Fertilizers Recommendation System for Disease Prediction

1. INTRODUCTION

1.1 Overview

Productivity, food security, sustainability, and environmental impact are handled using smart farming techniques. As the population level is increasing day by day food production plays a major role in it[1]. The quality of the food products is essential for good healthy human beings. For providing this healthy environment analysis of various factor that causes a productive loss in agriculture has to be done. Identification of diseased part, Image classification, Detection of an anomaly in plant leaves open wide area research named image processing in agriculture [2].

Identification of the plant diseases is the key to preventing the losses in the yield and quantity of the agricultural product. There are two main factors which cause disease in plants and they are pathogens and environmental condition. Pathogens are the main reason for causing diseases in the plant.

Virus is a living organism which has living cells in it and it affects the plant. The region affected by the virus in plant leaf and fruits are seen as yellow streaking, yellow spots, deformed leaves, and stunted growth [3]. The best way to prevent viral disease is the disposal of the viral affected region.

Fungi is also one of the main reason for the productivity loss in the plant. Ascomycetes and basidiomycetes are two main fungi mainly responsible for disease in the plant. Fungicides are widely used for controlling fungal infection in a plant [4]. Magnaporthe grisea which is commonly known as rice blast disease. Sclerotinia sclerotiorum is responsible for cotton rot. Oomycetes and Phytomyxea are the fungal-like organisms which contain destructive pathogen in the plant.

When a plant is infected by microscopic living organisms then it is a bacterial infection. Bacteria is a one-celled organism. The bacterially infected regions in plants can be seen as wilts, spots, and scabs. The spots which are caused by the blights can spread rapidly in plants. Tropical plants and vegetables are affected mainly by wilts. The absorption water by the plant is blocked by the bacterial infection. Some of the bacterial plant pathogens are Burkholderia and Proteobacteria.

Health monitoring and disease detection on plant is very critical for sustainable agriculture. Crop diseases identification remains difficult in many parts of the world due to the lack of the necessary infrastructure. It is very difficult to monitor the plant diseases manually. It requires tremendous amount of work, expertize in the plant diseases, and also require the excessive processing time. Hence, image processing and deep learning techniques are used for the detection of plant diseases. 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.

1.2 Purpose

Plant disease, especially on leaves, is one of the major factors that reduce the yield in both quality and quantity of the food crops. Finding the leaf disease is an important role to preserve agriculture. Smart analysis and Comprehensive prediction model in agriculture helps the farmer to yield right crop at the right time. The main benefits of the proposed system are as follows: Yield right crop at the right time, Balancing the crop production, control plant disease, Economic growth, and planning to reduce the crop scarcity. Hence to Detect and recognize the plant diseases and to recommend fertilizer it is necessary to provide symptoms in identifying the disease at its earliest.

2 LITERATURE SURVEY

2.1 Existing problem

Azadbakht et al. [5] used Gaussian process regression, random forest regression, support vector regression and boosted regression tree to analyze the performance for identifying leaf rust of wheat at canopy scale.

Elavarasan et al. [6] presented the models based on both supervised and unsupervised machine learning algorithm that were used for analyzing several factors like crop disease analysis, yield prediction and crop improvement.

Johannes et al. [7] predicted the symptoms Rust, septoria and tan disease in wheat plant using image processing algorithms. The images used for this analysis were taken from the mobile devices. Image-based automatic identification of plant disease mostly fails under real field condition.

Phadikar et al. [8] proposed an autonomous system for classifying disease in rice crop. It mainly extracts an infected region from the rice plant leaf. Information loss reduction and selection of an important feature are done through Roush Set Theory (RST).

Karadag et al. [9] proposed a technique that helps in identifying pepper fusarium disease. Light reflection from plant helps in knowing information about plant through specter radiometer. Four types of leaves were used for it namely mycorrhizal fungus, fusarium diseased, healthy and mycorrhizal leaves.

Liang et al. [10] proposed a technique based on the robust image-based plant disease diagnosis and severity estimation network (PD2SE-Net). It mainly focuses on practical diagnosis. This proposed technique also helps in measuring the severity of the disease.

Sandesh Raut and Karthik Ingale [11]proposed fast and accurate method for detection and classification of plant diseases. The proposed algorithm was tested on main five diseases on the plant they are Early Scorch, Cottony mold, Ashen Mold, Late scorch, Tiny Whiteness. The feature extracted was recognized through a pre-trained neural network.

Sanjay et al [12]proposed detection and classification of plant disease using image processing and artificial neural networks. The detection and classification of plant diseases was divided into three steps such as identification of infected object extraction of feature set of the infected leaf images and detection and classification the type of disease using ANN.

2.2 Proposed solution

The works specified in the literature have limitations in terms of accuracy, computation and resource management. Hence an automated and new fertilizers Recommendation System for crop disease prediction is introduced. This system identifies 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. 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

3 THEORITICAL ANALYSIS

3.1 Block diagram

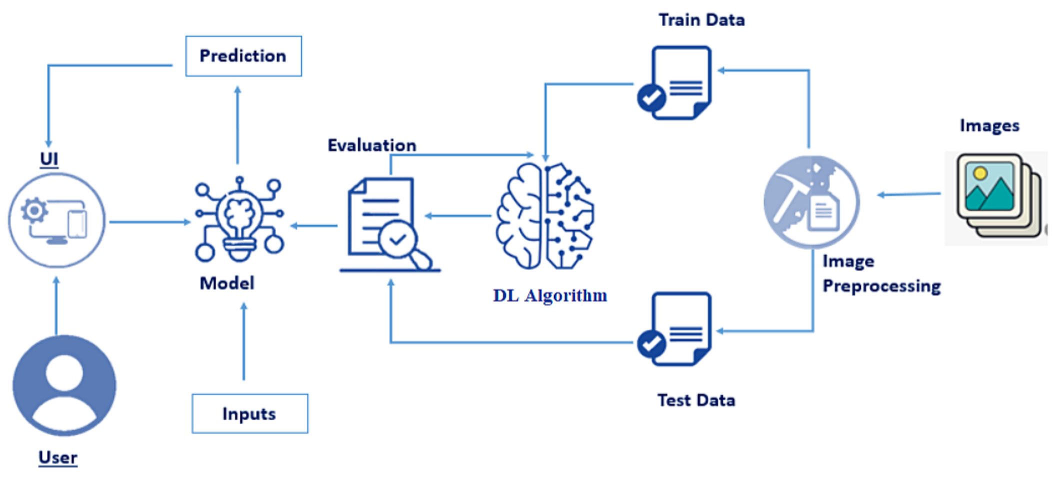


Figure 2: Architecture of the proposed system

The proposed approach is shown in Figure 2. The proposed approach consists of Data Collection, Image Preprocessing, and Model Building for Fruit Disease Prediction, Model Building for Vegetable Disease Prediction, Testing of the Models, Application Building and Training the Model on IBM.

3.2 Hardware / Software designing

3.1 Hardware Requirements

- Processor : 2.5 gigahertz (GHz) frequency or above.
- RAM : A minimum of 4 GB of RAM.
- Hard disk : A minimum of 20 GB of available space.
- Input Device : High resolution camera
- Monitor : Minimum Resolution 1024 X 768.

3.2 Software Requirements

- Python, Python Web Frame Works,
- ANN, CNN, IBM Cloud,
- IBM Watson Studio, IBM Machine Learning,
- Python-Flask

4. EXPERIMENTAL INVESTIGATIONS

The first step in the implementation of the proposed approach is Data Collection. Dataset can be collected from the source given by the admins. Train and Test folders are created with each folder having subfolders with leaf images of different plant diseases. Two datasets are used, two models are created, one to detect vegetable leaf diseases like tomato, potato, and pepper plants and the second model is for fruits diseases like corn, peach, and apple.

Image Preprocessing:

Once the data are collected, this data is used to train the model. Before training the model the images are preprocessed and then feed on to the model for training. Keras ImageDataGenerator class for image preprocessing is used. The ImageDataGenerator accepts the original data, randomly transforms it, and returns only the new, transformed data. The next step is to build model for fruit leaf disease detection

Model Building for Fruit Disease Prediction:

The model building from the augmented images includes the following. Importing the model building Libraries, Initializing the model, Adding CNN Layers, Adding Hidden Layer, Adding Output Layer, Configuring the Learning Process, Training and testing the model and saving the model.

Three layers are added for CNN

- Convolution layer
- Pooling layer
- Flattening layer

Convolution Layer:

The first layer of the neural network model, the convolution layer is added. The convolution layer, takes a number of 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). 32 3 x 3 convolution filters are used in this work.

Activation Function:

These are the functions that help to decide if the node need to be activated or not. These functions introduce non-linearity in the networks. Relu activation in used in the initial layers and softmax in final layer.

Pooling layer:

Max Pooling selects the maximum element from the region of the feature map covered by the filter. Thus, the output after the max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

Flatten layer:

The flatten layer is used to convert n-dimensional arrays to 1-dimensional arrays. This 1D array will be given as input to ANN fully connected layers.

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.

Compile the model:

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

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.

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.

Application Building:

After the model is built, the model is integrated 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

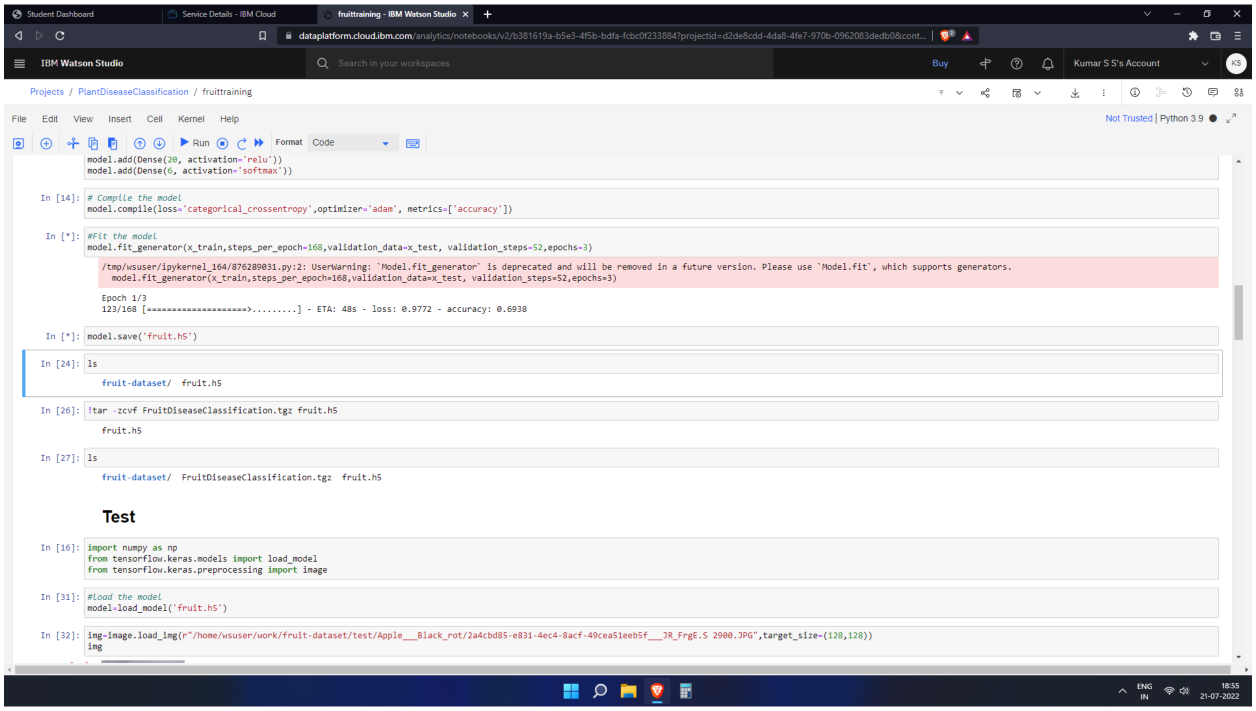
- HTML pages - front end
- Python script - Server-side script

Python script is created using spyder IDE

6. RESULT

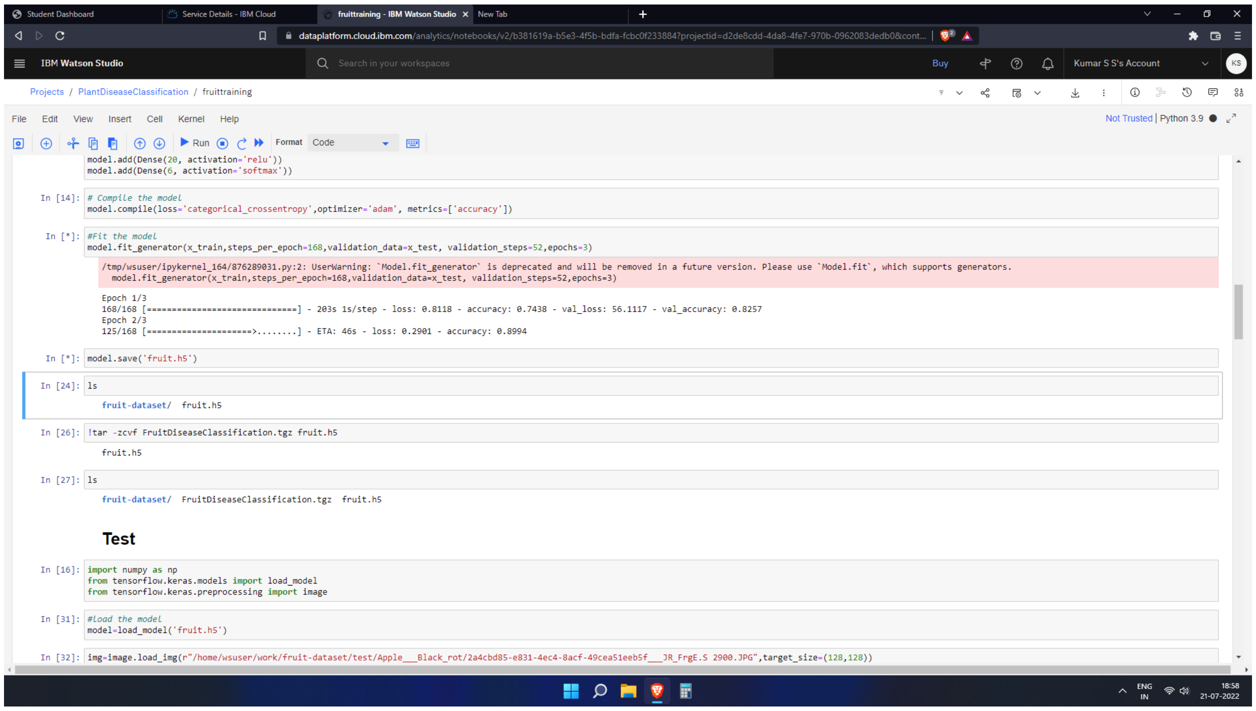
The screen shots of various steps of the automated system are shown below.

Figure 6.1-6.3 shows the training and testing of fruits disease classification using IBM Watson.



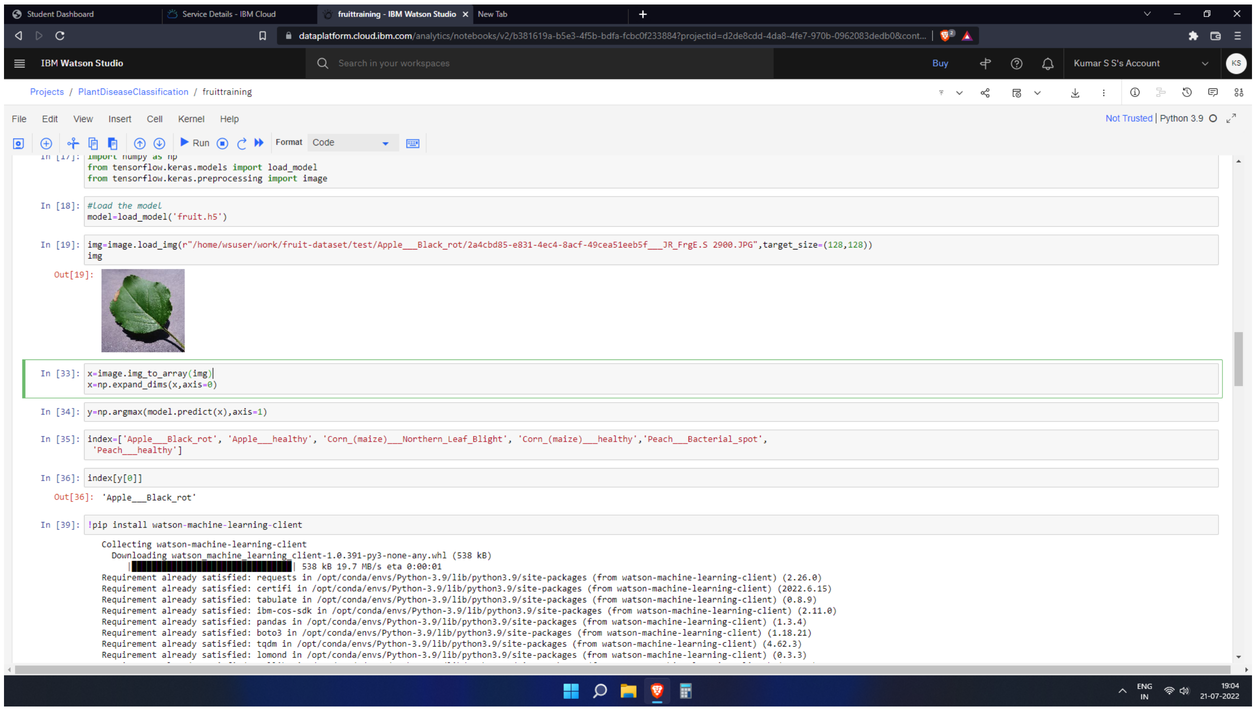
```
       model.add(Dense(20, activation='relu'))
       model.add(Dense(6, activation='softmax'))

In [14]: # Compile the model
         model.compile(loss='categorical_crossentropy',optimizer='adam', metrics=['accuracy'])

In [*]: #Fit the model
        model.fit_generator(x_train,steps_per_epoch=168,validation_data=x_test, validation_steps=52,epochs=3)

          /tmp/wsuser/ipykernel_164/876289031.py:2: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
            model.fit_generator(x_train,steps_per_epoch=168,validation_data=x_test, validation_steps=52,epochs=3)

          Epoch 1/3
          123/168 [===================>.........] - ETA: 48s - loss: 0.9772 - accuracy: 0.6938

In [*]: model.save('fruit.h5')

In [24]: ls

          fruit-dataset/  fruit.h5

In [26]: !tar -zcvf FruitDiseaseClassification.tgz fruit.h5

          fruit.h5

In [27]: ls

          fruit-dataset/  FruitDiseaseClassification.tgz  fruit.h5

          Test

In [16]: import numpy as np
         from tensorflow.keras.models import load_model
         from tensorflow.keras.preprocessing import image

In [31]: #load the model
         model=load_model('fruit.h5')

In [32]: img=image.load_img(r"/home/wsuser/work/fruit-dataset/test/Apple___Black_rot/2a4cbd85-e831-4ec4-8acf-49cea51eeb5f___JR_FrgE.S 2900.JPG",target_size=(128,128))
         img
```

Figure 6.1: Fruit Leaf Disease model training



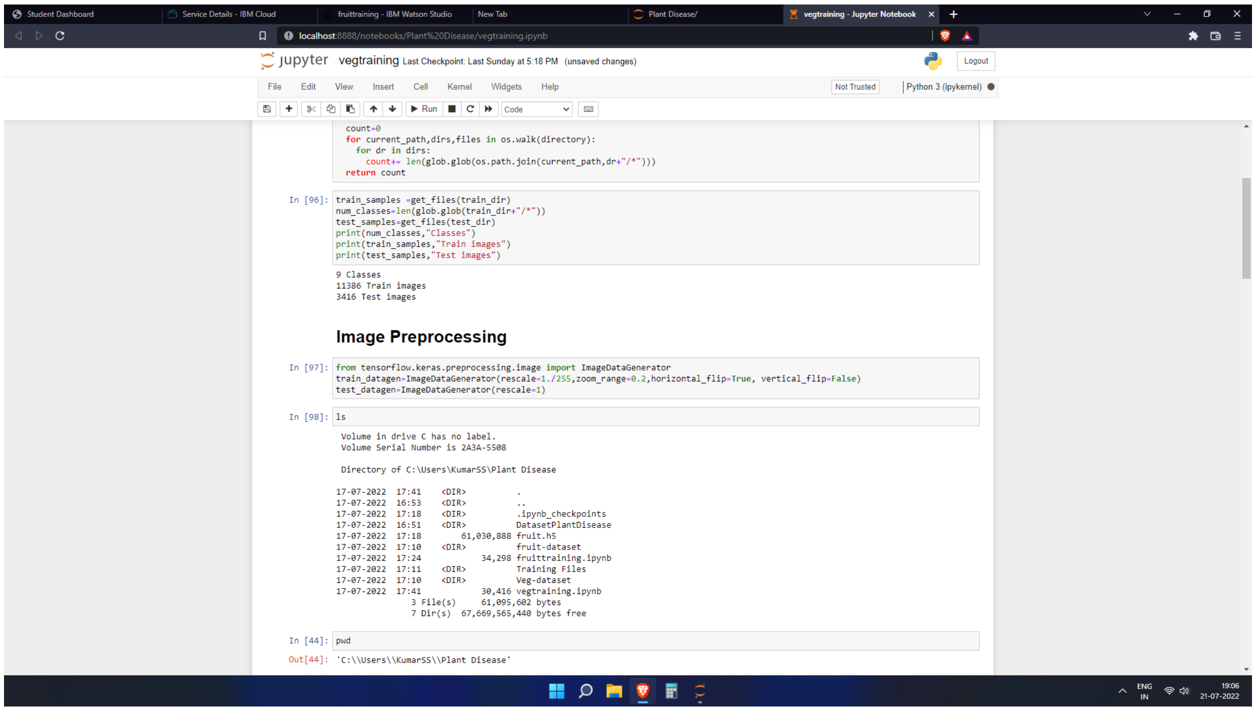
```
       model.add(Dense(20, activation='relu'))
       model.add(Dense(6, activation='softmax'))

In [14]: # Compile the model
         model.compile(loss='categorical_crossentropy',optimizer='adam', metrics=['accuracy'])

In [*]: #Fit the model
        model.fit_generator(x_train,steps_per_epoch=168,validation_data=x_test, validation_steps=52,epochs=3)

          /tmp/wsuser/ipykernel_164/876289031.py:2: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
            model.fit_generator(x_train,steps_per_epoch=168,validation_data=x_test, validation_steps=52,epochs=3)

          Epoch 1/3
          168/168 [==============================] - 203s 1s/step - loss: 0.8118 - accuracy: 0.7438 - val_loss: 56.1117 - val_accuracy: 0.8257
          Epoch 2/3
          125/168 [====================>.........] - ETA: 46s - loss: 0.2901 - accuracy: 0.8994

In [*]: model.save('fruit.h5')

In [24]: ls

          fruit-dataset/  fruit.h5

In [26]: !tar -zcvf FruitDiseaseClassification.tgz fruit.h5

          fruit.h5

In [27]: ls

          fruit-dataset/  FruitDiseaseClassification.tgz  fruit.h5

          Test

In [16]: import numpy as np
         from tensorflow.keras.models import load_model
         from tensorflow.keras.preprocessing import image

In [31]: #load the model
         model=load_model('fruit.h5')

In [32]: img=image.load_img(r"/home/wsuser/work/fruit-dataset/test/Apple___Black_rot/2a4cbd85-e831-4ec4-8acf-49cea51eeb5f___JR_FrgE.S 2900.JPG",target_size=(128,128))
```

Figure 6.2 : Training process of the model



```
In [17]: import numpy as np
         from tensorflow.keras.models import load_model
         from tensorflow.keras.preprocessing import image

In [18]: #load the model
         model=load_model('fruit.h5')

In [19]: img=image.load_img(r"/home/wsuser/work/fruit-dataset/test/Apple___Black_rot/2a4cbd85-e831-4ec4-8acf-49cea51eeb5f___JR_FrgE.S 2900.JPG",target_size=(128,128))
         img

Out[19]:

In [33]: x=image.img_to_array(img)
         x=np.expand_dims(x,axis=0)

In [34]: y=np.argmax(model.predict(x),axis=1)

In [35]: index=['Apple___Black_rot', 'Apple___healthy', 'Corn_(maize)___Northern_Leaf_Blight', 'Corn_(maize)___healthy','Peach___Bacterial_spot',
          'Peach___healthy']

In [36]: index[y[0]]

Out[36]: 'Apple___Black_rot'

In [39]: !pip install watson-machine-learning-client

           Collecting watson-machine-learning-client
             Downloading watson_machine_learning_client-1.0.391-py3-none-any.whl (538 kB)
                |████████████████████████████████| 538 kB 19.7 MB/s eta 0:00:01
           Requirement already satisfied: requests in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2.26.0)
           Requirement already satisfied: certifi in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2022.6.15)
           Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (0.8.9)
           Requirement already satisfied: ibm-cos-sdk in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2.11.0)
           Requirement already satisfied: pandas in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.3.4)
           Requirement already satisfied: boto3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.18.21)
           Requirement already satisfied: tqdm in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (4.62.3)
           Requirement already satisfied: lomond in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (0.3.3)
```

Figure 6.3Testing of the trained model with an image from the test dataset

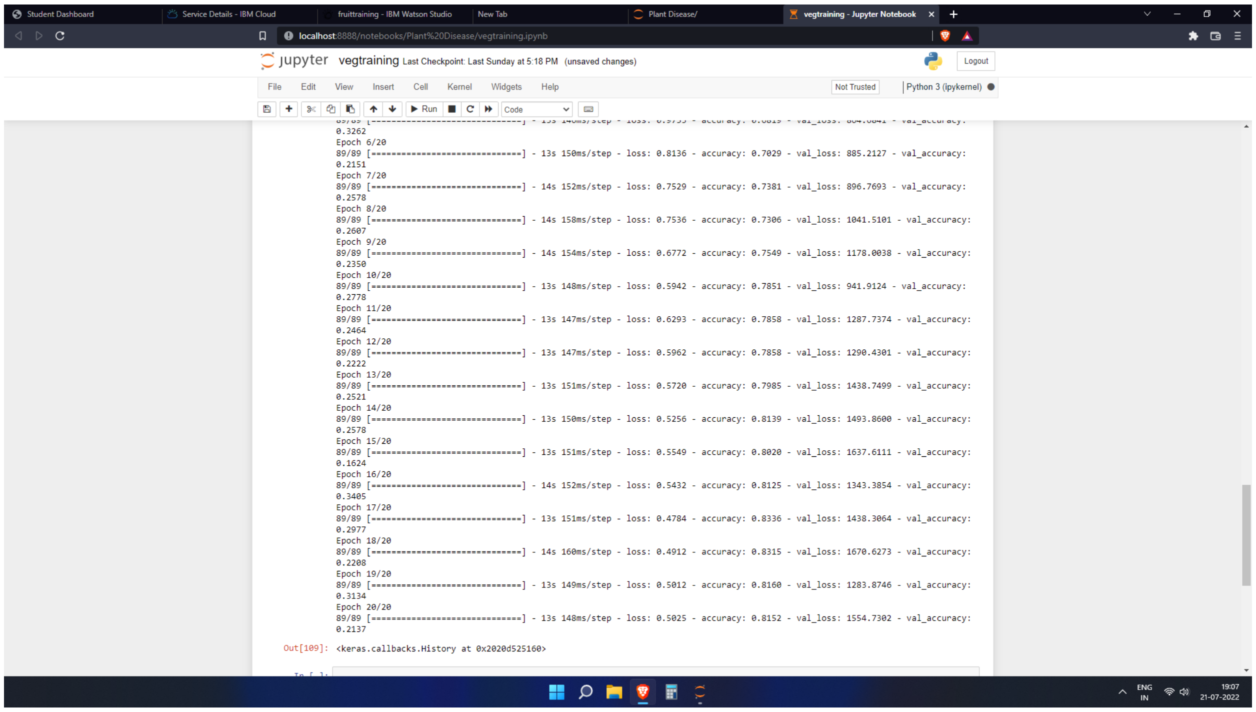


```
           count=0
           for current_path,dirs,files in os.walk(directory):
             for dr in dirs:
               count+= len(glob.glob(os.path.join(current_path,dr+"/*")))
           return count

In [96]: train_samples =get_files(train_dir)
         num_classes=len(glob.glob(train_dir+"/*"))
         test_samples=get_files(test_dir)
         print(num_classes,"Classes")
         print(train_samples,"Train images")
         print(test_samples,"Test images")

         9 Classes
         11386 Train images
         3416 Test images
```

Image Preprocessing

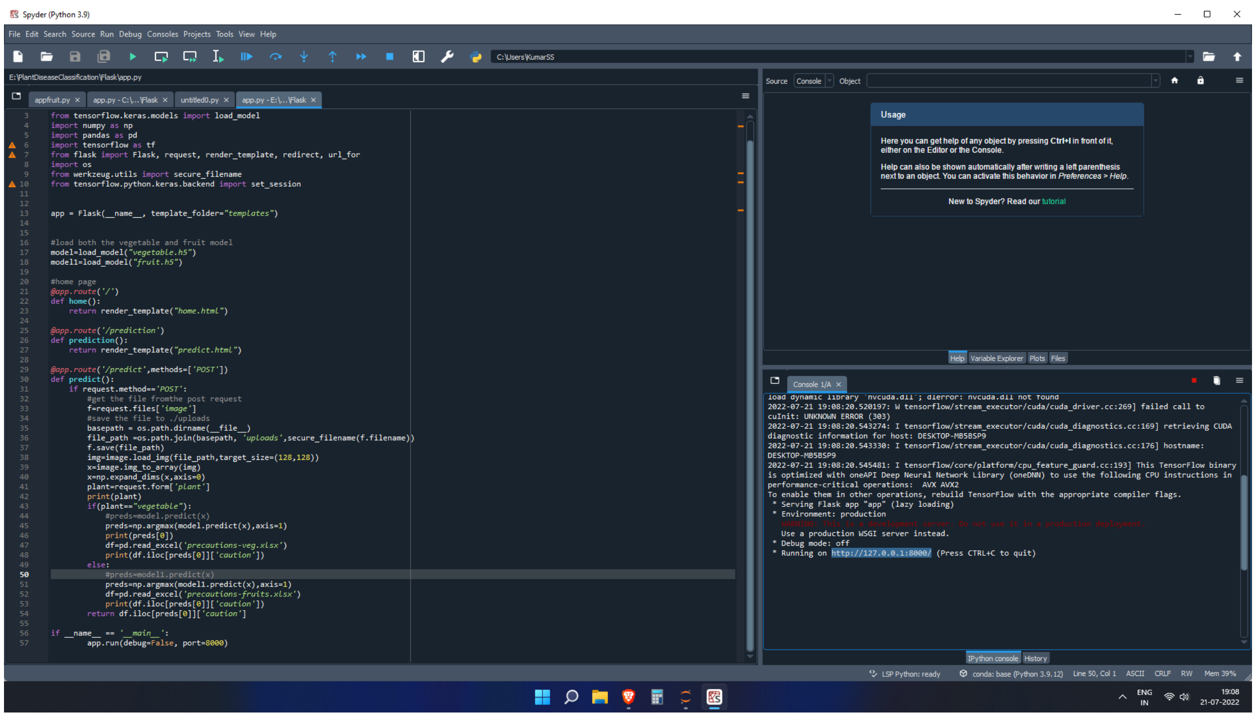
```
In [97]: from tensorflow.keras.preprocessing.image import ImageDataGenerator
         train_datagen=ImageDataGenerator(rescale=1./255,zoom_range=0.2,horizontal_flip=True, vertical_flip=False)
         test_datagen=ImageDataGenerator(rescale=1)

In [98]: ls

          Volume in drive C has no label.
          Volume Serial Number is 2A3A-5508

          Directory of C:\Users\KumarSS\Plant Disease

         17-07-2022  17:41    <DIR>          .
         17-07-2022  16:53    <DIR>          ..
         17-07-2022  17:18    <DIR>          .ipynb_checkpoints
         17-07-2022  16:51    <DIR>          DatasetPlantDisease
         17-07-2022  17:18        61,030,888 fruit.h5
         17-07-2022  17:10    <DIR>          fruit-dataset
         17-07-2022  17:24            34,298 fruittraining.ipynb
         17-07-2022  17:11    <DIR>          Training Files
         17-07-2022  17:10    <DIR>          Veg-dataset
         17-07-2022  17:41            30,416 vegtraining.ipynb
                        3 File(s)     61,095,602 bytes
                        7 Dir(s)  67,669,565,440 bytes free

In [44]: pwd

Out[44]: 'C:\\Users\\KumarSS\\Plant Disease'
```

Figure 6.4 Implementation of vegetable disease classification model using Jupyter



```
0.3262
Epoch 6/20
89/89 [==============================] - 13s 150ms/step - loss: 0.8136 - accuracy: 0.7029 - val_loss: 885.2127 - val_accuracy:
0.2151
Epoch 7/20
89/89 [==============================] - 14s 152ms/step - loss: 0.7529 - accuracy: 0.7381 - val_loss: 896.7693 - val_accuracy:
0.2578
Epoch 8/20
89/89 [==============================] - 14s 158ms/step - loss: 0.7536 - accuracy: 0.7306 - val_loss: 1041.5101 - val_accuracy:
0.2607
Epoch 9/20
89/89 [==============================] - 14s 154ms/step - loss: 0.6772 - accuracy: 0.7549 - val_loss: 1178.0038 - val_accuracy:
0.2350
Epoch 10/20
89/89 [==============================] - 13s 148ms/step - loss: 0.5942 - accuracy: 0.7851 - val_loss: 941.9124 - val_accuracy:
0.2778
Epoch 11/20
89/89 [==============================] - 13s 147ms/step - loss: 0.6293 - accuracy: 0.7858 - val_loss: 1287.7374 - val_accuracy:
0.2464
Epoch 12/20
89/89 [==============================] - 13s 147ms/step - loss: 0.5962 - accuracy: 0.7858 - val_loss: 1290.4301 - val_accuracy:
0.2222
Epoch 13/20
89/89 [==============================] - 13s 151ms/step - loss: 0.5720 - accuracy: 0.7985 - val_loss: 1438.7499 - val_accuracy:
0.2521
Epoch 14/20
89/89 [==============================] - 13s 150ms/step - loss: 0.5256 - accuracy: 0.8139 - val_loss: 1493.8600 - val_accuracy:
0.2578
Epoch 15/20
89/89 [==============================] - 13s 151ms/step - loss: 0.5549 - accuracy: 0.8020 - val_loss: 1637.6111 - val_accuracy:
0.1624
Epoch 16/20
89/89 [==============================] - 14s 152ms/step - loss: 0.5432 - accuracy: 0.8125 - val_loss: 1343.3854 - val_accuracy:
0.3405
Epoch 17/20
89/89 [==============================] - 13s 151ms/step - loss: 0.4784 - accuracy: 0.8336 - val_loss: 1438.3064 - val_accuracy:
0.2977
Epoch 18/20
89/89 [==============================] - 14s 160ms/step - loss: 0.4912 - accuracy: 0.8315 - val_loss: 1670.6273 - val_accuracy:
0.2208
Epoch 19/20
89/89 [==============================] - 13s 149ms/step - loss: 0.5012 - accuracy: 0.8160 - val_loss: 1283.8746 - val_accuracy:
0.3134
Epoch 20/20
89/89 [==============================] - 13s 148ms/step - loss: 0.5025 - accuracy: 0.8152 - val_loss: 1554.7302 - val_accuracy:
0.2137

Out[109]: <keras.callbacks.History at 0x2020d525160>
```

Figure 6.5 Training of vegetable disease classification model using Jupyter



```python
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__, template_folder="templates")


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

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

@app.route('/prediction')
def prediction():
    return render_template("predict.html")

@app.route('/predict',methods=['POST'])
def predict():
    if request.method=='POST':
        #get the file fromthe 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(model.predict(x),axis=1)
            print(preds[0])
            df=pd.read_excel('precautions-veg.xlsx')
            print(df.iloc[preds[0]]['caution'])
        else:
            #preds=model1.predict(x)
            preds=np.argmax(model1.predict(x),axis=1)
            df=pd.read_excel('precautions-fruits.xlsx')
            print(df.iloc[preds[0]]['caution'])
        return df.iloc[preds[0]]['caution']

if __name__ == '__main__':
        app.run(debug=False, port=8000)
```

Usage

Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

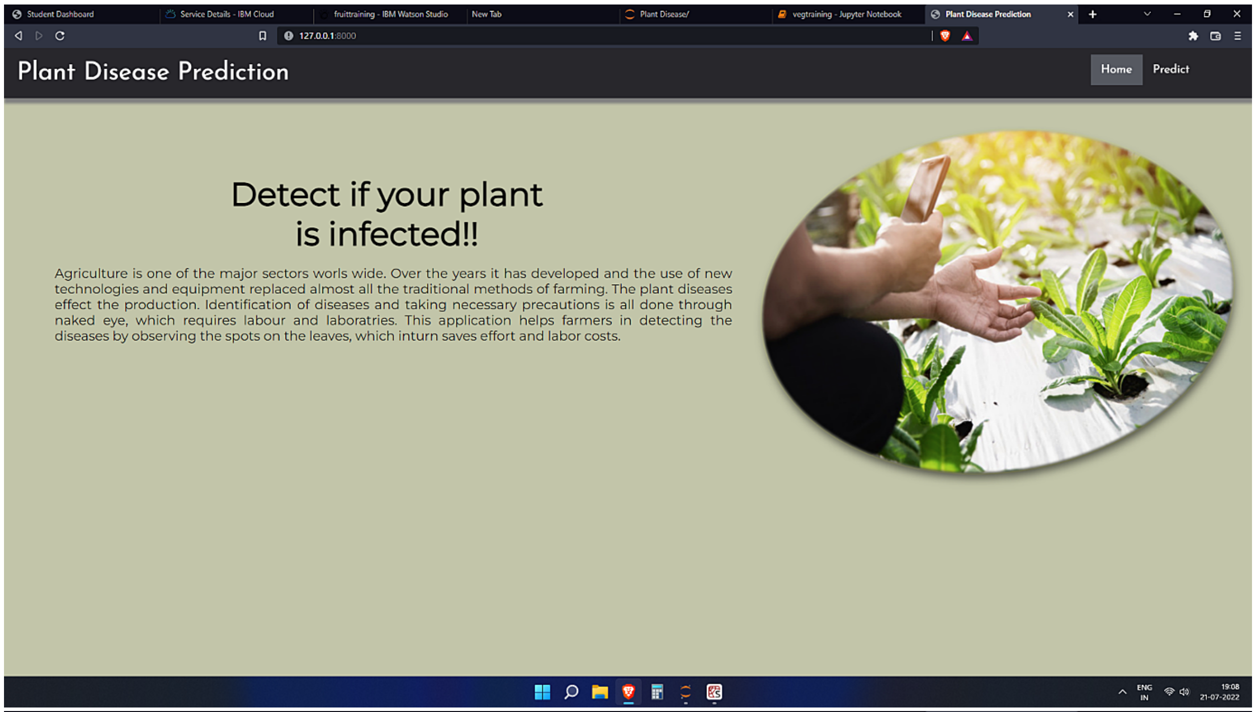
New to Spyder? Read our tutorial

```
load dynamic library 'nvcuda.dll'; dlerror: nvcuda.dll not found
2022-07-21 19:08:20.520197: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to
cuInit: UNKNOWN ERROR (303)
2022-07-21 19:08:20.543274: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA
diagnostic information for host: DESKTOP-MB5BSP9
2022-07-21 19:08:20.543330: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname:
DESKTOP-MB5BSP9
2022-07-21 19:08:20.545481: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary
is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in
performance-critical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:8000/ (Press CTRL+C to quit)
```

Figure 6.6 Creating web application using Spyder IDE

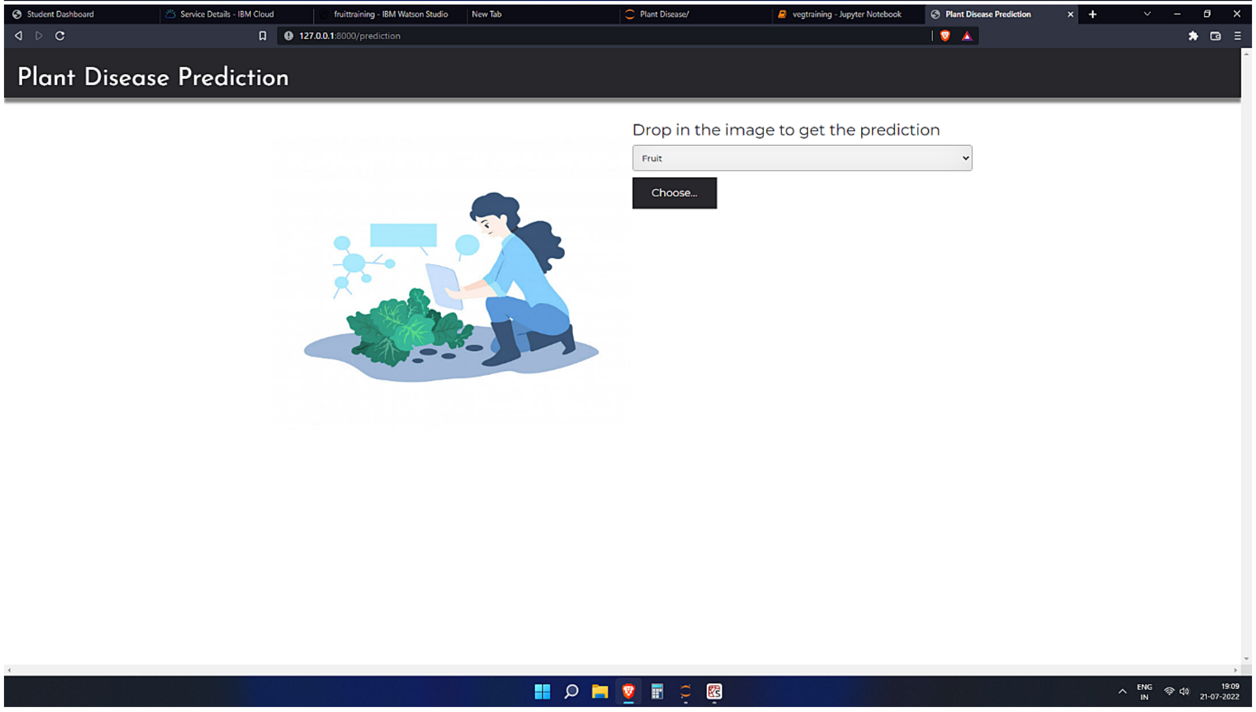
The trained and saved model fruits.h5 and vegetables.h5 was used for creating a web application using Flask. The process is shown is Figure 6.6.

Figure 6.7: Demo page of the web app



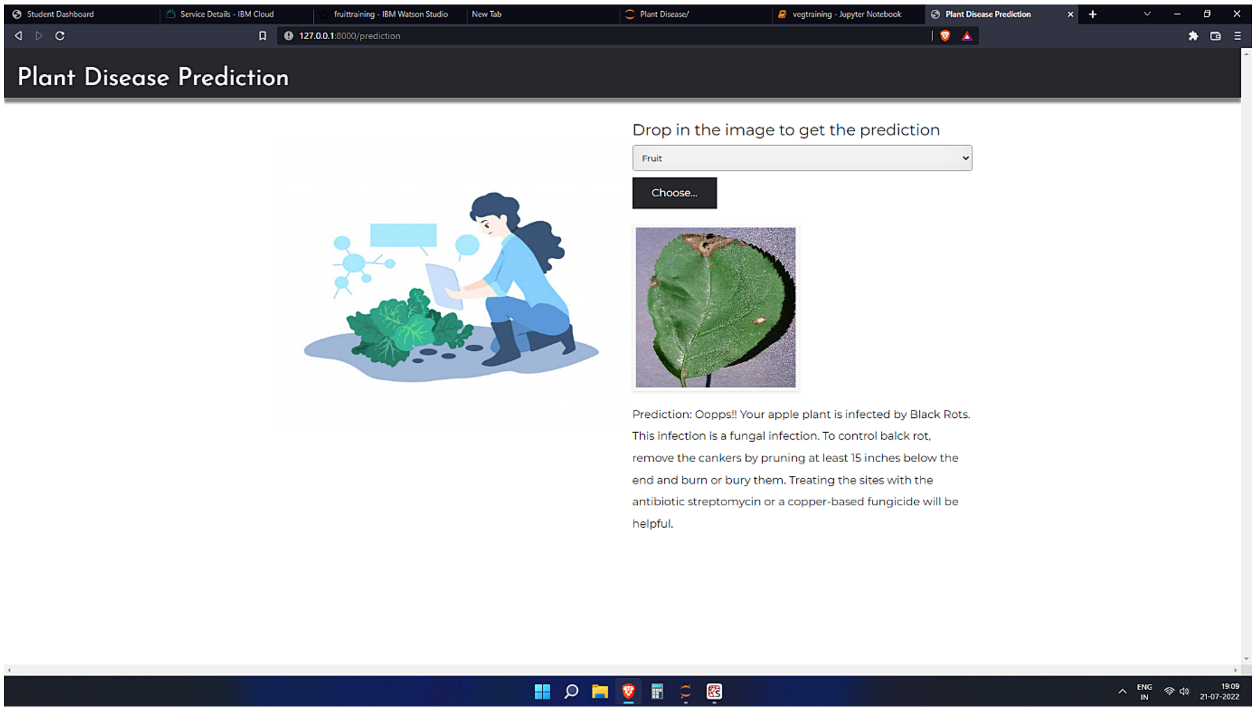
Detect if your plant is infected!!

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.



Drop in the image to get the prediction

Figure 6.8: Web app page asking to choose one type from vegetable and fruit



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.

Figure 6.9: Web app page predicting the disease and recommending the suitable fertilizer for the test data.

7 ADVANTAGES & DISADVANTAGES

The proposed system can help farmers to identify crop diseases at an early stage. The system also suggests the combination of fertilizers to be used by the farmers, based on the analysis done by the system. This can promote crop production and thereby increase the productivity. This can also protect the quality of soil, so that future cultivations can yield good productivity.

The limitation of the system is, there are cases where the system couldn’t identify the diseases correctly. Also the types of diseases considered are minimal. If the user interface allows to connect to the camera directly, it would be helpful to all the farmers.

8 APPLICATIONS

The proposed system can be applied in different areas,

1. Automated Disease Prediction
2. Automatic fertilizer recommendation
3. Disease prediction by mobile phones.
4. Smart Agriculture Solutions

9 CONCLUSION

Monitoring the health and identifying the disease on plants is essential for sustainable agriculture. An automated system using image processing and deep learning techniques was introduced to identify different diseases on plants by checking the symptoms shown on the leaves of the plant and suggest the precautions that can be taken for those diseases. Two datasets i) vegetable leaf diseases like tomato, potato, and pepper plants and the ii) fruits diseases like corn, peach, and apple was studied in this work. The system was trained and modelled using deep learning CNN. The trained models were used to create a web app through which farmers can easily identify the disease of the plant and get suggestions for the use of Fertilizers.

10 FUTURE SCOPE

The system can be further improved by using some recent networks. Transfer learning techniques can be applied to improve the accuracy. The system can also be extended to more number of disease prediction.

11 BIBILOGRAPHY

1. Graham M Turner, Kirsten A Larsen, Seona Candy, Sue Ogilvy, Jaithri Ananthapavan, Marj Moodie, Sarah W James, Sharon Friel, Chris J Ryan, and Mark A Lawrence. Squandering australias food security the environmental and economic costs of our unhealthy diet and the policy path were on. Journal of cleaner production, 195:1581–1599, 2018

2. Gittaly Dhingra, Vinay Kumar, and Hem Dutt Joshi. A novel computer vision based neutrosophic approach for leaf disease identification and classification. Measurement, 135:782–794, 2019.

3. Paula Tennant, Gustavo Fermin, and Jerome E Foster. Viruses: Molecular Biology, Host Interactions, and Applications to Biotechnology. Academic Press, 2018.

4. David B Collinge, Hans JL Jorgensen, Meike AC Latz, Andrea Manzotti, Fani Ntana, Edward Camilo Rojas Tayo, and Birgit Jensen. Searching for novel fungal biological control agents for plant disease control among endophytes. Endophytes for a Growing World, pages 25–51, 2019

5. Mohsen Azadbakht, Davoud Ashourloo, Hossein Aghighi, Soheil Radiom, and Abbas Alimohammadi. Wheat leaf rust detection at canopy scale under different lai levels using machine learning techniques. Computers and Electronics in Agriculture, 156:119–128, 2019.

6. Dhivya Elavarasan, Durai Raj Vincent, Vishal Sharma, Albert Y Zomaya, and Kathiravan Srinivasan. Forecasting yield by integrating agrarian factors and machine learning models: A survey. Computers and Electronics in Agriculture, 155:257–282, 2018

7. Alexander Johannes, Artzai Picon, Aitor Alvarez-Gila, Jone Echazarra, Sergio Rodriguez-Vaamonde, Ana D´ıez Navajas, and Amaia Ortiz-Barredo. Automatic plant disease diagnosis using mobile capture devices, applied on a wheat use case. Computers and electronics in agriculture, 138:200–209, 2017.

8. Santanu Phadikar, Jaya Sil, and Asit Kumar Das. Rice diseases classification using feature selection and rule generation techniques. Computers and electronics in agriculture, 90:76–85, 2013

9. Kerim Karada˘g, Mehmet Emin Tenekeci, Ramazan Ta¸saltın, and Ay¸sin Bilgili. Detection of pepper fusarium disease using machine learning algorithms based on spectral reflectance. Sustainable Computing: Informatics and Systems, 2019.

10. Qiaokang Liang, Shao Xiang, Yucheng Hu, Gianmarc Coppola, Dan Zhang, and Wei Sun. Pd2se-net: Computer-assisted plant disease diagnosis and severity estimation network. Computers and electronics in agriculture, 157:518–529, 2019

11. Sandesh Raut, Kartik Ingole , Review On Leaf Disease Detection Using Image Processing Techniques International Research Journal of Engineering and Technology (IRJET), Volume: 04 Issue: 04 | Apr -2017

12. Mr. Sanjay Mirchandani, Mihir Pendse, Prathamesh Rane, Ashwini Vedula, Plant Disease Detection And Classification Using Image Processing And Artificial Neural Networks, International Research Journal of Engineering and Technology (IRJET), Volume: 05 Issue: 06 | June 2018

APPENDIX

A. Source Code

Fruit disease training Model

```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0
```

```
# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It
includes your credentials.
# You might want to remove those credentials before you share the notebook.
client_5081f81d79d14bc98585e4995f36ea4f = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='ayQ-QLPt1QO3ZO_iHtnBUv8N2Lh6qRLO8LZ4_Fi-p3KG',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

streaming_body_2 =
client_5081f81d79d14bc98585e4995f36ea4f.get_object(Bucket='plantdiseaseclassifi
cation-donotdelete-pr-3jizmpkeyw1kkv', Key='fruit-dataset.zip')['Body']

# Your data file was loaded into a botocore.response.StreamingBody object.
# Please read the documentation of ibm_boto3 and pandas to learn more about
the possibilities to load the data.
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
# pandas documentation: http://pandas.pydata.org/
```

In [2]:

```
from io import BytesIO
import zipfile
unzip=zipfile.ZipFile(BytesIO(streaming_body_2.read()),'r')
file_paths=unzip.namelist()
for path in file_paths:
    unzip.extract(path)
```

In [3]:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [7]:

```
train_datagen=ImageDataGenerator(rescale=1./255,zoom_range=0.2,horizontal_flip=
True, vertical_flip=False)
test_datagen=ImageDataGenerator(rescale=1)
```

In [4]:

```
ls
fruit-dataset/
```

In [5]:

```
pwd
```

Out[5]:

```
'/home/wsuser/work'
```

In [8]:

```
x_train=train_datagen.flow_from_directory(r"/home/wsuser/work/fruit-
dataset/train",target_size=(128,128),class_mode='categorical',batch_size=32)
x_test=test_datagen.flow_from_directory(r"/home/wsuser/work/fruit-
dataset/test",target_size=(128,128),class_mode='categorical',batch_size=32)
Found 5384 images belonging to 6 classes.
Found 1686 images belonging to 6 classes.
```

In [9]:

```
x_train.class_indices
```

Out[9]:

```
{'Apple___Black_rot': 0,
 'Apple___healthy': 1,
 'Corn_(maize)___Northern_Leaf_Blight': 2,
 'Corn_(maize)___healthy': 3,
 'Peach___Bacterial_spot': 4,
 'Peach___healthy': 5}
```

In [10]:

```
#Import keras models and layers

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
```

In [11]:

```
model=Sequential()
model.add(Convolution2D(32,(3,3), input_shape=(128,128,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
```

In [12]:

```
model.summary()
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 32)      896

 max_pooling2d (MaxPooling2D  (None, 63, 63, 32)       0
 )

 flatten (Flatten)           (None, 127008)            0

=================================================================
Total params: 896
Trainable params: 896
Non-trainable params: 0
_________________________________________________________________
```

In [13]:

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

In [14]:

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

In [15]:

```
#Fit the model
model.fit_generator(x_train,steps_per_epoch=168,validation_data=x_test,
validation_steps=52,epochs=3)
/tmp/wsuser/ipykernel_164/876289031.py:2: UserWarning: `Model.fit_generator` is
deprecated and will be removed in a future version. Please use `Model.fit`,
which supports generators.
  model.fit_generator(x_train,steps_per_epoch=168,validation_data=x_test,
validation_steps=52,epochs=3)
Epoch 1/3
168/168 [==============================] - 203s 1s/step - loss: 0.8118 -
accuracy: 0.7438 - val_loss: 56.1117 - val_accuracy: 0.8257
Epoch 2/3
168/168 [==============================] - 201s 1s/step - loss: 0.2739 -
accuracy: 0.9049 - val_loss: 344.3285 - val_accuracy: 0.5150
Epoch 3/3
168/168 [==============================] - 202s 1s/step - loss: 0.2159 -
accuracy: 0.9277 - val_loss: 390.2771 - val_accuracy: 0.5571
```

Out[15]:

```
<keras.callbacks.History at 0x7f8e92092700>
```

In [16]:

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

In [24]:

```
ls
fruit-dataset/  fruit.h5
```

In [26]:

```
!tar -zcvf FruitDiseaseClassification.tgz fruit.h5
fruit.h5
```

In [27]:

```
ls
fruit-dataset/  FruitDiseaseClassification.tgz  fruit.h5
```

Test

In [17]:

```
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
```

In [18]:

```
#load the model
model=load_model('fruit.h5')
```

In [19]:

```
img=image.load_img(r"/home/wsuser/work/fruit-
dataset/test/Apple___Black_rot/2a4cbd85-e831-4ec4-8acf-
49cea51eeb5f___JR_FrgE.S 2900.JPG",target_size=(128,128))
img
```

Out[19]:

In [33]:

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

In [34]:

```
y=np.argmax(model.predict(x),axis=1)
```

In [35]:

```
index=['Apple___Black_rot', 'Apple___healthy',
'Corn_(maize)___Northern_Leaf_Blight',
'Corn_(maize)___healthy','Peach___Bacterial_spot',
 'Peach___healthy']
```

In [36]:

```
index[y[0]]
```

Out[36]:

```
'Apple___Black_rot'
```

In [39]:

```
!pip install watson-machine-learning-client
Collecting watson-machine-learning-client
  Downloading watson_machine_learning_client-1.0.391-py3-none-any.whl (538 kB)
     |████████████████████████████████| 538 kB 19.7 MB/s eta 0:00:01
Requirement already satisfied: requests in /opt/conda/envs/Python-
```

```
3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2.26.0)
Requirement already satisfied: certifi in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client)
(2022.6.15)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (0.8.9)
Requirement already satisfied: ibm-cos-sdk in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2.11.0)
Requirement already satisfied: pandas in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.3.4)
Requirement already satisfied: boto3 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client)
(1.18.21)
Requirement already satisfied: tqdm in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (4.62.3)
Requirement already satisfied: lomond in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (0.3.3)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.26.7)
Requirement already satisfied: s3transfer<0.6.0,>=0.5.0 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-
machine-learning-client) (0.5.0)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-
machine-learning-client) (0.10.0)
Requirement already satisfied: botocore<1.22.0,>=1.21.21 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-
machine-learning-client) (1.21.41)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from
botocore<1.22.0,>=1.21.21->boto3->watson-machine-learning-client) (2.8.2)
Requirement already satisfied: six>=1.5 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1-
>botocore<1.22.0,>=1.21.21->boto3->watson-machine-learning-client) (1.15.0)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk-
>watson-machine-learning-client) (2.11.0)
Requirement already satisfied: ibm-cos-sdk-core==2.11.0 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk-
>watson-machine-learning-client) (2.11.0)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->watson-
machine-learning-client) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from requests->watson-machine-learning-
client) (3.3)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from pandas->watson-machine-learning-client)
(2021.3)
Requirement already satisfied: numpy>=1.17.3 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from pandas->watson-machine-learning-client)
(1.20.3)
Installing collected packages: watson-machine-learning-client
Successfully installed watson-machine-learning-client-1.0.391
```

In [37]:

```
from ibm_watson_machine_learning import APIClient
wml_credentials={
    "url":"https://us-south.ml.cloud.ibm.com",
    "apikey":"79usIO-T1tcICWjvzF9sxfShUpnN9UWqIn0B960NTJlE"
}
```

In [38]:

```
client=APIClient(wml_credentials)
```

In [39]:

```
client
```

Out[39]:

```
<ibm_watson_machine_learning.client.APIClient at 0x7fbfb58dc430>
```

In [40]:

```
def guid_space_name(client, PlantDiseaseDeploy):
    space=client.spaces.get_details()
    return (next(item for item in space ['resources'] if item
['entity']['name']==PlantDiseaseDeploy)['metadata']['id'])
```

In [41]:

```
space_uid=guid_space_name(client, 'PlantDiseaseDeploy')
print(space_uid)
0a27f7f4-c72b-418f-a896-a0f57f6b1154
```

In [42]:

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

Out[42]:

```
'SUCCESS'
```

In [44]:

```
pip install tensorflow
Requirement already satisfied: tensorflow in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (2.7.2)
Requirement already satisfied: protobuf>=3.9.2 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from tensorflow) (3.19.1)
```

```
Requirement already satisfied: gast<0.5.0,>=0.2.1 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from tensorflow) (0.4.0)
Requirement already satisfied: termcolor>=1.1.0 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from tensorflow) (1.1.0)
Requirement already satisfied: astunparse>=1.6.0 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from tensorflow) (1.6.3)
Requirement already satisfied: typing-extensions>=3.6.6 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorflow)
(3.7.4.3)
Requirement already satisfied: h5py>=2.9.0 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from tensorflow) (3.2.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from tensorflow) (1.42.0)
Requirement already satisfied: keras<2.8,>=2.7.0 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from tensorflow) (2.7.0)
Requirement already satisfied: flatbuffers<3.0,>=1.12 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from tensorflow) (2.0)
Requirement already satisfied: tensorboard~=2.7 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from tensorflow) (2.7.0)
Requirement already satisfied: tensorflow-estimator<2.8,~=2.7.0 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorflow)
(2.7.0)
Requirement already satisfied: absl-py>=0.4.0 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from tensorflow) (0.12.0)
Requirement already satisfied: opt-einsum>=2.3.2 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from tensorflow) (3.3.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorflow)
(1.1.2)
Requirement already satisfied: wheel<1.0,>=0.32.0 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from tensorflow) (0.37.0)
Requirement already satisfied: google-pasta>=0.1.1 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: six>=1.12.0 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from tensorflow) (1.15.0)
Requirement already satisfied: wrapt>=1.11.0 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from tensorflow) (1.12.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.21.0 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorflow)
(0.23.1)
Requirement already satisfied: numpy>=1.14.5 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from tensorflow) (1.20.3)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorboard~=2.7-
>tensorflow) (0.4.4)
```

```
Requirement already satisfied: google-auth<3,>=1.6.3 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from tensorboard~=2.7->tensorflow) (1.23.0)
Requirement already satisfied: markdown>=2.6.8 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from tensorboard~=2.7->tensorflow) (3.3.3)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorboard~=2.7-
>tensorflow) (1.6.0)
Requirement already satisfied: werkzeug>=0.11.15 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from tensorboard~=2.7->tensorflow) (2.0.2)
Requirement already satisfied: setuptools>=41.0.0 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from tensorboard~=2.7->tensorflow) (58.0.4)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorboard~=2.7-
>tensorflow) (0.6.1)
Requirement already satisfied: requests<3,>=2.21.0 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from tensorboard~=2.7->tensorflow) (2.26.0)
Requirement already satisfied: rsa<5,>=3.1.4 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from google-auth<3,>=1.6.3->tensorboard~=2.7-
>tensorflow) (4.7.2)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from google-
auth<3,>=1.6.3->tensorboard~=2.7->tensorflow) (4.2.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from google-auth<3,>=1.6.3->tensorboard~=2.7-
>tensorflow) (0.2.8)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from google-auth-
oauthlib<0.5,>=0.4.1->tensorboard~=2.7->tensorflow) (1.3.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from pyasn1-modules>=0.2.1->google-
auth<3,>=1.6.3->tensorboard~=2.7->tensorflow) (0.4.8)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from requests<3,>=2.21.0->tensorboard~=2.7-
>tensorflow) (3.3)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from requests<3,>=2.21.0->tensorboard~=2.7-
>tensorflow) (1.26.7)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from requests<3,>=2.21.0->tensorboard~=2.7-
>tensorflow) (2022.6.15)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from
requests<3,>=2.21.0->tensorboard~=2.7->tensorflow) (2.0.4)
Requirement already satisfied: oauthlib>=3.0.0 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from requests-oauthlib>=0.7.0->google-auth-
oauthlib<0.5,>=0.4.1->tensorboard~=2.7->tensorflow) (3.2.0)
Note: you may need to restart the kernel to use updated packages.
```

In [48]:

```
client.software_specifications.list(100)
-------------------------------  ------------------------------------  ----
NAME                             ASSET_ID                              TYPE
default_py3.6                    0062b8c9-8b7d-44a0-a9b9-46c416adcbd9  base
kernel-spark3.2-scala2.12        020d69ce-7ac1-5e68-ac1a-31189867356a  base
pytorch-onnx_1.3-py3.7-edt       069ea134-3346-5748-b513-49120e15d288  base
scikit-learn_0.20-py3.6          09c5a1d0-9c1e-4473-a344-eb7b665ff687  base
spark-mllib_3.0-scala_2.12       09f4cff0-90a7-5899-b9ed-1ef348aebdee  base
pytorch-onnx_rt22.1-py3.9        0b848dd4-e681-5599-be41-b5f6fccc6471  base
ai-function_0.1-py3.6            0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda  base
shiny-r3.6                       0e6e79df-875e-4f24-8ae9-62dcc2148306  base
tensorflow_2.4-py3.7-horovod     1092590a-307d-563d-9b62-4eb7d64b3f22  base
pytorch_1.1-py3.6                10ac12d6-6b30-4ccd-8392-3e922c096a92  base
tensorflow_1.15-py3.6-ddl        111e41b3-de2d-5422-a4d6-bf776828c4b7  base
runtime-22.1-py3.9               12b83a17-24d8-5082-900f-0ab31fbfd3cb  base
scikit-learn_0.22-py3.6          154010fa-5b3b-4ac1-82af-4d5ee5abbc85  base
default_r3.6                     1b70aec3-ab34-4b87-8aa0-a4a3c8296a36  base
pytorch-onnx_1.3-py3.6           1bc6029a-cc97-56da-b8e0-39c3880dbbe7  base
pytorch-onnx_rt22.1-py3.9-edt    1d362186-7ad5-5b59-8b6c-9d0880bde37f  base
tensorflow_2.1-py3.6             1eb25b84-d6ed-5dde-b6a5-3fbdf1665666  base
spark-mllib_3.2                  20047f72-0a98-58c7-9ff5-a77b012eb8f5  base
tensorflow_2.4-py3.8-horovod     217c16f6-178f-56bf-824a-b19f20564c49  base
runtime-22.1-py3.9-cuda          26215f05-08c3-5a41-a1b0-da66306ce658  base
do_py3.8                         295addb5-9ef9-547e-9bf4-92ae3563e720  base
autoai-ts_3.8-py3.8              2aa0c932-798f-5ae9-abd6-15e0c2402fb5  base
tensorflow_1.15-py3.6            2b73a275-7cbf-420b-a912-eae7f436e0bc  base
pytorch_1.2-py3.6                2c8ef57d-2687-4b7d-acce-01f94976dac1  base
spark-mllib_2.3                  2e51f700-bca0-4b0d-88dc-5c6791338875  base
pytorch-onnx_1.1-py3.6-edt       32983cea-3f32-4400-8965-dde874a8d67e  base
spark-mllib_3.0-py37             36507ebe-8770-55ba-ab2a-eafe787600e9  base
spark-mllib_2.4                  390d21f8-e58b-4fac-9c55-d7ceda621326  base
xgboost_0.82-py3.6               39e31acd-5f30-41dc-ae44-60233c80306e  base
pytorch-onnx_1.2-py3.6-edt       40589d0e-7019-4e28-8daa-fb03b6f4fe12  base
default_r36py38                  41c247d3-45f8-5a71-b065-8580229facf0  base
autoai-ts_rt22.1-py3.9           4269d26e-07ba-5d40-8f66-2d495b0c71f7  base
autoai-obm_3.0                   42b92e18-d9ab-567f-988a-4240ba1ed5f7  base
pmml-3.0_4.3                     493bcb95-16f1-5bc5-bee8-81b8af80e9c7  base
spark-mllib_2.4-r_3.6            49403dff-92e9-4c87-a3d7-a42d0021c095  base
xgboost_0.90-py3.6               4ff8d6c2-1343-4c18-85e1-689c965304d3  base
pytorch-onnx_1.1-py3.6           50f95b2a-bc16-43bb-bc94-b0bed208c60b  base
autoai-ts_3.9-py3.8              52c57136-80fa-572e-8728-a5e7cbb42cde  base
```

```
spark-mllib_2.4-scala_2.11        55a70f99-7320-4be5-9fb9-9edb5a443af5  base
spark-mllib_3.0                   5c1b0ca2-4977-5c2e-9439-ffd44ea8ffe9  base
autoai-obm_2.0                    5c2e37fa-80b8-5e77-840f-d912469614ee  base
spss-modeler_18.1                 5c3cad7e-507f-4b2a-a9a3-ab53a21dee8b  base
cuda-py3.8                        5d3232bf-c86b-5df4-a2cd-7bb870a1cd4e  base
autoai-kb_3.1-py3.7               632d4b22-10aa-5180-88f0-f52dfb6444d7  base
pytorch-onnx_1.7-py3.8            634d3cdc-b562-5bf9-a2d4-ea90a478456b  base
spark-mllib_2.3-r_3.6             6586b9e3-ccd6-4f92-900f-0f8cb2bd6f0c  base
tensorflow_2.4-py3.7             65e171d7-72d1-55d9-8ebb-f813d620c9bb  base
spss-modeler_18.2                 687eddc9-028a-4117-b9dd-e57b36f1efa5  base
pytorch-onnx_1.2-py3.6            692a6a4d-2c4d-45ff-a1ed-b167ee55469a  base
spark-mllib_2.3-scala_2.11        7963efe5-bbec-417e-92cf-0574e21b4e8d  base
spark-mllib_2.4-py37              7abc992b-b685-532b-a122-a396a3cdbaab  base
caffe_1.0-py3.6                  7bb3dbe2-da6e-4145-918d-b6d84aa93b6b  base
pytorch-onnx_1.7-py3.7            812c6631-42b7-5613-982b-02098e6c909c  base
cuda-py3.6                        82c79ece-4d12-40e6-8787-a7b9e0f62770  base
tensorflow_1.15-py3.6-horovod    8964680e-d5e4-5bb8-919b-8342c6c0dfd8  base
hybrid_0.1                        8c1a58c6-62b5-4dc4-987a-df751c2756b6  base
pytorch-onnx_1.3-py3.7            8d5d8a87-a912-54cf-81ec-3914adaa988d  base
caffe-ibm_1.0-py3.6              8d863266-7927-4d1e-97d7-56a7f4c0a19b  base
spss-modeler_17.1                 902d0051-84bd-4af6-ab6b-8f6aa6fdeabb  base
do_12.10                          9100fd72-8159-4eb9-8a0b-a87e12eefa36  base
do_py3.7                          9447fa8b-2051-4d24-9eef-5acb0e3c59f8  base
spark-mllib_3.0-r_3.6             94bb6052-c837-589d-83f1-f4142f219e32  base
cuda-py3.7-opence                 94e9652b-7f2d-59d5-ba5a-23a414ea488f  base
nlp-py3.8                         96e60351-99d4-5a1c-9cc0-473ac1b5a864  base
cuda-py3.7                        9a44990c-1aa1-4c7d-baf8-c4099011741c  base
hybrid_0.2                        9b3f9040-9cee-4ead-8d7a-780600f542f7  base
spark-mllib_3.0-py38              9f7a8fc1-4d3c-5e65-ab90-41fa8de2d418  base
autoai-kb_3.3-py3.7               a545cca3-02df-5c61-9e88-998b09dc79af  base
spark-mllib_3.0-py39              a6082a27-5acc-5163-b02c-6b96916eb5e0  base
runtime-22.1-py3.9-do             a7e7dbf1-1d03-5544-994d-e5ec845ce99a  base
default_py3.8                     ab9e1b80-f2ce-592c-a7d2-4f2344f77194  base
tensorflow_rt22.1-py3.9          acd9c798-6974-5d2f-a657-ce06e986df4d  base
kernel-spark3.2-py3.9             ad7033ee-794e-58cf-812e-a95f4b64b207  base
autoai-obm_2.0 with Spark 3.0     af10f35f-69fa-5d66-9bf5-acb58434263a  base
default_py3.7_opence              c2057dd4-f42c-5f77-a02f-72bdbd3282c9  base
tensorflow_2.1-py3.7             c4032338-2a40-500a-beef-b01ab2667e27  base
do_py3.7_opence                   cc8f8976-b74a-551a-bb66-6377f8d865b4  base
autoai-kb_3.0-py3.6               d139f196-e04b-5d8b-9140-9a10ca1fa91a  base
spark-mllib_3.0-py36              d82546d5-dd78-5fbb-9131-2ec309bc56ed  base
autoai-kb_3.4-py3.8               da9b39c3-758c-5a4f-9cfd-457dd4d8c395  base
kernel-spark3.2-r3.6              db2fe4d6-d641-5d05-9972-73c654c60e0a  base
autoai-kb_rt22.1-py3.9            db6afe93-665f-5910-b117-d879897404d9  base
tensorflow_rt22.1-py3.9-horovod  dda170cc-ca67-5da7-9b7a-cf84c6987fae  base
```

```
autoai-ts_1.0-py3.7             deef04f0-0c42-5147-9711-89f9904299db  base
tensorflow_2.1-py3.7-horovod    e384fce5-fdd1-53f8-bc71-11326c9c635f  base
default_py3.7                   e4429883-c883-42b6-87a8-f419d64088cd  base
do_22.1                         e51999ba-6452-5f1f-8287-17228b88b652  base
autoai-obm_3.2                  eae86aab-da30-5229-a6a6-1d0d4e368983  base
do_20.1                         f686cdd9-7904-5f9d-a732-01b0d6b10dc5  base
scikit-learn_0.19-py3.6         f963fa9d-4bb7-5652-9c5d-8d9289ef6ad9  base
tensorflow_2.4-py3.8            fe185c44-9a99-5425-986b-59bd1d2eda46  base
------------------------------  ------------------------------------  ----
```

In [49]:

```
software_space_uid=client.software_specifications.get_uid_by_name('tensorflow_rt
22.1-py3.9')
software_space_uid
```

Out[49]:

```
'acd9c798-6974-5d2f-a657-ce06e986df4d'
```

In [52]:

```
model_details =
client.repository.store_model(model='FruitDiseaseClassification.tgz',
  meta_props={
              client.repository.ModelMetaNames.NAME:'PlantDiseaseDeploy',
              client.repository.ModelMetaNames.TYPE:'tensorflow_2.7',

client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_space_uid } )
```

In [53]:

```
model_id=client.repository.get_model_id(model_details)
```

In [54]:

```
model_id
```

Out[54]:

```
'13fa8062-da8c-4f6b-b6bb-d3f1e385af17'
```

In [57]:

```
client.repository.download(model_id, 'FruitDisease.tar.gz')
Successfully saved model content to file: 'FruitDisease.tar.gz'
```

Out[57]:

```
'/home/wsuser/work/FruitDisease.tar.gz'
```

In [58]:

```
ls
fruit-dataset/                  FruitDisease.tag.gb  fruit.h5
FruitDiseaseClassification.tgz  FruitDisease.tar.gz
```

In [ ]:

.

Vegetable disease training model

```
# Import Libraries
import os
import glob
import matplotlib.pyplot as plt
import numpy as np

# Keras API
import keras
from keras.models import Sequential
from keras.layers import Dense,Dropout,Flatten
from keras.layers import
Conv2D,MaxPooling2D,Activation,AveragePooling2D,BatchNormalization
from keras.preprocessing.image import ImageDataGenerator
```

In [94]:

```
train_dir ="Veg-dataset/train_set/"
test_dir="Veg-dataset/test_set/"
```

In [95]:

```
# function to get count of images
def get_files(directory):
  if not os.path.exists(directory):
    return 0
  count=0
  for current_path,dirs,files in os.walk(directory):
    for dr in dirs:
      count+= len(glob.glob(os.path.join(current_path,dr+"/*")))
  return count
```

In [96]:

```
train_samples =get_files(train_dir)
num_classes=len(glob.glob(train_dir+"/*"))
test_samples=get_files(test_dir)
print(num_classes,"Classes")
print(train_samples,"Train images")
print(test_samples,"Test images")
9 Classes
11386 Train images
3416 Test images
```

Image Preprocessing

In [97]:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen=ImageDataGenerator(rescale=1./255,zoom_range=0.2,horizontal_flip=
True, vertical_flip=False)
test_datagen=ImageDataGenerator(rescale=1)
```

In [98]:

```
ls
 Volume in drive C has no label.
 Volume Serial Number is 2A3A-5508

 Directory of C:\Users\KumarSS\Plant Disease

17-07-2022  17:41    <DIR>          .
17-07-2022  16:53    <DIR>          ..
17-07-2022  17:18    <DIR>          .ipynb_checkpoints
17-07-2022  16:51    <DIR>          DatasetPlantDisease
17-07-2022  17:18        61,030,888 fruit.h5
17-07-2022  17:10    <DIR>          fruit-dataset
17-07-2022  17:24            34,298 fruittraining.ipynb
17-07-2022  17:11    <DIR>          Training Files
17-07-2022  17:10    <DIR>          Veg-dataset
17-07-2022  17:41            30,416 vegtraining.ipynb
               3 File(s)     61,095,602 bytes
               7 Dir(s)  67,669,565,440 bytes free
```

In [44]:

```
pwd
```

Out[44]:

```
'C:\\Users\\KumarSS\\Plant Disease'
```

In [99]:

```
x_train=train_datagen.flow_from_directory('Veg-
dataset/train_set',target_size=(128,128),class_mode='categorical',batch_size=1
6)
Found 11386 images belonging to 9 classes.
```

In [100]:

```
x_test=test_datagen.flow_from_directory('Veg-
dataset/test_set',target_size=(128,128),class_mode='categorical',batch_size=2
6)
Found 3416 images belonging to 9 classes.
```

In [101]:

```
x_train.class_indices
```

Out[101]:

```
{'Pepper,_bell___Bacterial_spot': 0,
 'Pepper,_bell___healthy': 1,
 'Potato___Early_blight': 2,
 'Potato___Late_blight': 3,
 'Potato___healthy': 4,
 'Tomato___Bacterial_spot': 5,
 'Tomato___Late_blight': 6,
 'Tomato___Leaf_Mold': 7,
 'Tomato___Septoria_leaf_spot': 8}
```

CNN Implementation

In [102]:

```
#Import keras models and layers

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
```

In [103]:

```
model=Sequential()
model.add(Convolution2D(32,(3,3), input_shape=(128,128,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
```

In [104]:

```
model.summary()
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 126, 126, 32)      896

 max_pooling2d_4 (MaxPooling  (None, 63, 63, 32)       0
 2D)

 flatten_4 (Flatten)         (None, 127008)            0

=================================================================
Total params: 896
Trainable params: 896
Non-trainable params: 0
_________________________________________________________________
```

Hidden layers

In [105]:

```
model.add(Dense(300, activation='relu'))
model.add(Dense(150, activation='relu'))
model.add(Dense(75, activation='relu'))
model.add(Dense(9, activation='softmax'))
```

In [106]:

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

In [107]:

```
len(x_train)
```

Out[107]:

```
712
```

In [108]:

```
len(x_test)
```

Out[108]:

```
132
```

In [109]:

```
#Fit the model
model.fit_generator(x_train,steps_per_epoch=89,validation_data=x_test,
validation_steps=27,epochs=20)
#model.fit(x_train,steps_per_epoch=len(x_train),validation_data=x_test,
validation_steps=len(x_test),epochs=3)
Epoch 1/20
C:\Users\KumarSS\AppData\Local\Temp\ipykernel_11188\4058285867.py:2:
UserWarning: `Model.fit_generator` is deprecated and will be removed in a
future version. Please use `Model.fit`, which supports generators.
  model.fit_generator(x_train,steps_per_epoch=89,validation_data=x_test,
validation_steps=27,epochs=20)
89/89 [==============================] - 14s 154ms/step - loss: 2.4896 -
accuracy: 0.3111 - val_loss: 212.3326 - val_accuracy: 0.4188
Epoch 2/20
89/89 [==============================] - 13s 148ms/step - loss: 1.4210 -
accuracy: 0.5070 - val_loss: 255.8210 - val_accuracy: 0.4117
Epoch 3/20
89/89 [==============================] - 13s 147ms/step - loss: 1.1745 -
accuracy: 0.5997 - val_loss: 351.9586 - val_accuracy: 0.4103
Epoch 4/20
```

```
89/89 [==============================] - 13s 145ms/step - loss: 0.9475 -
accuracy: 0.6685 - val_loss: 797.9529 - val_accuracy: 0.2664
Epoch 5/20
89/89 [==============================] - 13s 146ms/step - loss: 0.9755 -
accuracy: 0.6819 - val_loss: 864.6841 - val_accuracy: 0.3262
Epoch 6/20
89/89 [==============================] - 13s 150ms/step - loss: 0.8136 -
accuracy: 0.7029 - val_loss: 885.2127 - val_accuracy: 0.2151
Epoch 7/20
89/89 [==============================] - 14s 152ms/step - loss: 0.7529 -
accuracy: 0.7381 - val_loss: 896.7693 - val_accuracy: 0.2578
Epoch 8/20
89/89 [==============================] - 14s 158ms/step - loss: 0.7536 -
accuracy: 0.7306 - val_loss: 1041.5101 - val_accuracy: 0.2607
Epoch 9/20
89/89 [==============================] - 14s 154ms/step - loss: 0.6772 -
accuracy: 0.7549 - val_loss: 1178.0038 - val_accuracy: 0.2350
Epoch 10/20
89/89 [==============================] - 13s 148ms/step - loss: 0.5942 -
accuracy: 0.7851 - val_loss: 941.9124 - val_accuracy: 0.2778
Epoch 11/20
89/89 [==============================] - 13s 147ms/step - loss: 0.6293 -
accuracy: 0.7858 - val_loss: 1287.7374 - val_accuracy: 0.2464
Epoch 12/20
89/89 [==============================] - 13s 147ms/step - loss: 0.5962 -
accuracy: 0.7858 - val_loss: 1290.4301 - val_accuracy: 0.2222
Epoch 13/20
89/89 [==============================] - 13s 151ms/step - loss: 0.5720 -
accuracy: 0.7985 - val_loss: 1438.7499 - val_accuracy: 0.2521
Epoch 14/20
89/89 [==============================] - 13s 150ms/step - loss: 0.5256 -
accuracy: 0.8139 - val_loss: 1493.8600 - val_accuracy: 0.2578
Epoch 15/20
89/89 [==============================] - 13s 151ms/step - loss: 0.5549 -
accuracy: 0.8020 - val_loss: 1637.6111 - val_accuracy: 0.1624
Epoch 16/20
89/89 [==============================] - 14s 152ms/step - loss: 0.5432 -
accuracy: 0.8125 - val_loss: 1343.3854 - val_accuracy: 0.3405
Epoch 17/20
89/89 [==============================] - 13s 151ms/step - loss: 0.4784 -
accuracy: 0.8336 - val_loss: 1438.3064 - val_accuracy: 0.2977
Epoch 18/20
89/89 [==============================] - 14s 160ms/step - loss: 0.4912 -
accuracy: 0.8315 - val_loss: 1670.6273 - val_accuracy: 0.2208
Epoch 19/20
```

```
89/89 [==============================] - 13s 149ms/step - loss: 0.5012 - accuracy: 0.8160 - val_loss: 1283.8746 - val_accuracy: 0.3134
Epoch 20/20
89/89 [==============================] - 13s 148ms/step - loss: 0.5025 - accuracy: 0.8152 - val_loss: 1554.7302 - val_accuracy: 0.2137
```

Out[109]:

```
<keras.callbacks.History at 0x2020d525160>
```

In [ ]:

Save the model

In [110]:

```
model.save('vegetable.h5')
```

In [111]:

```
ls
 Volume in drive C has no label.
 Volume Serial Number is 2A3A-5508

 Directory of C:\Users\KumarSS\Plant Disease

17-07-2022  17:47    <DIR>          .
17-07-2022  16:53    <DIR>          ..
17-07-2022  17:18    <DIR>          .ipynb_checkpoints
17-07-2022  16:51    <DIR>          DatasetPlantDisease
17-07-2022  17:18        61,030,888 fruit.h5
17-07-2022  17:10    <DIR>          fruit-dataset
17-07-2022  17:24            34,298 fruittraining.ipynb
17-07-2022  17:11    <DIR>          Training Files
17-07-2022  17:10    <DIR>          Veg-dataset
17-07-2022  17:47       457,981,136 vegetable.h5
17-07-2022  17:47            15,570 vegtraining.ipynb
               4 File(s)    519,061,892 bytes
               7 Dir(s)  67,206,922,240 bytes free
```

In [ ]:

Testing the model

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Import Models

In [25]:

```
import numpy as np
import keras
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import img_to_array
from keras.preprocessing import image
import tensorflow as tf
```

Load the saved model (fruit)

In [40]:

```
model=load_model('fruit.h5')
```

In [41]:

```
img=keras.utils.load_img("fruit-dataset/test/Apple___Black_rot/2a4cbd85-e831-
4ec4-8acf-49cea51eeb5f___JR_FrgE.S 2900.JPG",target_size=(128,128))
img
```

Out[41]:

In [42]:

```
x=tf.keras.utils.img_to_array(img)
x=np.expand_dims(x,axis=0)
y=np.argmax(model.predict(x),axis=1)
```

```
1/1 [==============================] - 0s 47ms/step
```

In [43]:

```
index=['Apple___Black_rot', 'Apple___healthy',
'Corn_(maize)___Northern_Leaf_Blight',
'Corn_(maize)___healthy','Peach___Bacterial_spot',
 'Peach___healthy']
```

In [44]:

```
index[y[0]]
```

Out[44]:

```
'Apple___Black_rot'
```

# Load the saved model (vegetable)

In [45]:

```
model=load_model('vegetable.h5')
```

In [51]:

```
img=keras.utils.load_img("Veg-
dataset/test_set/Tomato___Bacterial_spot/b5c376b6-3c0a-46d7-a053-
2b3a57168897___UF.GRC_BS_Lab Leaf 9003.JPG",target_size=(128,128))
img
```

Out[51]:

In [52]:

```
x=tf.keras.utils.img_to_array(img)
x=np.expand_dims(x,axis=0)
y=np.argmax(model.predict(x),axis=1)
```

```
1/1 [==============================] - 0s 26ms/step
```

In [53]:

```
index=['Pepper,_bell___Bacterial_spot','Pepper,_bell___healthy','Potato___Earl
y_blight','Potato___Late_blight','Potato___healthy','Tomato___Bacterial_spot',
 'Tomato___Late_blight','Tomato___Leaf_Mold','Tomato___Septoria_leaf_spot']
```

In [54]:

```
index[y[0]]
```

Out[54]:

```
'Tomato___Septoria_leaf_spot'
```

In [ ]:

Flask app creation

```
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__, template_folder="templates")


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

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

@app.route('/prediction')
def prediction():
    return render_template("predict.html")

@app.route('/predict',methods=['POST'])
def predict():
    if request.method=='POST':
        #get the file fromthe 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(model.predict(x),axis=1)
            print(preds[0])
            df=pd.read_excel('precautions-veg.xlsx')
            print(df.iloc[preds[0]]['caution'])
        else:
            #preds=model1.predict(x)
            preds=np.argmax(model1.predict(x),axis=1)
            df=pd.read_excel('precautions-fruits.xlsx')
            print(df.iloc[preds[0]]['caution'])
        return df.iloc[preds[0]]['caution']

if __name__ == '__main__':
        app.run(debug=False, port=8000)
```