
PROJECT REPORT

Title of the Project:
FERTILIZER RECOMMENDATION SYSTEM FOR DISEASE PREDICTION

Submitted by:
Dr. J. Jeya Jeevahan, M.E., Ph.D.,
Assistant Professor,
Department of Mechanical Engineering,
Sathyabama Institute of Science and Technology, Chennai.

Submitted for the partial fulfilment of:
Project Build-a-thon, FDP conducted by IBM and SmartInternz in association with AICTE.

Report Submitted on:
05-August-2022

---

1. INTRODUCTION

The fertilizer recommendation system is a kind of AI program built for predicting the kind of disease a fruit/vegetable plants are affected. Based on the disease affected by the plants, the type of fertilizer may be recommended. The fertilizer recommendation system may be helpful in the agricultural sector to identify the type of plant diseases from the images of leaf itself. Hence, the fertilizer recommendation system can be much beneficial to the society.

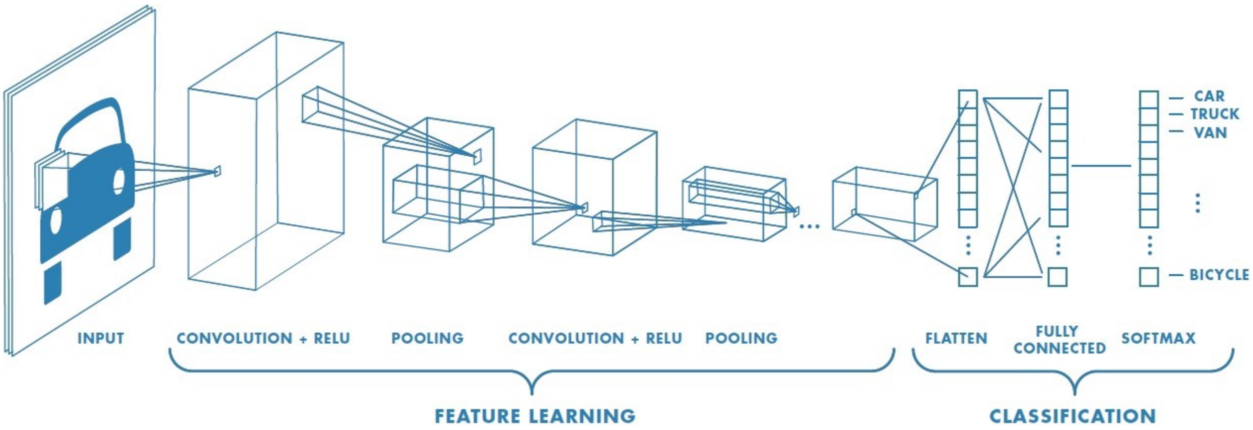
The purpose of this project work is to predict the type of plant through the developed AI model.

2. PROJECT OBJECTIVES

The objective of this project work is to develop the AI model based system to predict the plant diseases. In order to identify the type of plant disease, two datasets were used: (a) fruit leaves, and (b) vegetable leaves. As CNN model is more suitable in image recognition, CNN model is proposed for determining the type of plant disease.

3. CONVOLUTIONAL NEURAL NETWORK (CNN) MODEL

CNN is a type of neural network model which is more suitable for image recognition and classification. Unlike the classical image recognition where you define the image features yourself, CNN takes the image's raw pixel data, trains the model, then extracts the features automatically for better classification. CNN image classifications takes an input image, process it and classify it under certain categories (Eg., apple, banana, orange and so on). In CNN, each input image will pass it through a series of convolution layers with filters (Kernals), Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between 0 and 1.



In this method, the input preprocessed image is provided into convolution layer. The parameters are chosen, the filters with strides and padding were applied, if required. The convolution on the image is performed and ReLU activation is applied to the matrix. The pooling is then performed to reduce dimensionality size. If required, as many convolutional layers are added. Flattening of the output was performed then, and the feed is given to a fully connected layer (FC Layer). Finally, the output is obtained using an activation function (Logistic Regression with cost functions). The result is usually an image classification or recognition.

4. PROCEDURE/METHODOLOGY FOLLOWED

Step 1: Launching IBM Jupyter Notebook:
Go to "IBM Cloud Machine Learning" resource, and launch "IBM Cloud Pak for Data". In the opening window, click on "New Project" and go to "Create". In the next page, give the project name and other required information. Now the IBM Jupyter Notebook is opened. Type the Python coding to get started.

Step 2: Creating the Image Dataset:
Load of the image file (Eg. Fruits, Vegetables, etc.) in Zip file format using Object Space and Streaming Body options available under 0001 icon. Unzip the image file using python coding, and create the image dataset in IBM cloud. The dataset contains two subfolders, namely, train and test. The training folder dataset will be used for training the model, and test dataset will be used for testing whether or not the model is predicting the images correctly or not.

Step 3: Image Augmentation:
After the image dataset is created, the next step is image augmentation. Using tensorflow, the training dataset images as well as test dataset images are preprocessed. In preprocessing, the colour images are converted into greyscale, and the size of the images are converted into 128x128 pixels. A suitable batch size is selected for both training dataset and test dataset.

Step 4: Setting Up the Convolution Layer in the CNN model:
Using tensorflow library, the CNN model is developed. It contains the following funtions:

- Convolution funtion with 3x3 window size moving across the 128x128 pixel images, and "RELU" activation function.
- Pooling function with MaxPooling type with the pool size of 2x2.
- Flatten funtion.

Step 5: Setting Up the Hidden Layer in the CNN model:
In this step, the dense layer is set up with 150 and 50 with "RELU" activation funtion.

Step 6: Setting Up the Output Layer in the CNN model:

In this step, two layers are added. First, a dense layer is created with the required number of categories and "SOFTMAX" activation function. Then, a loss funtion is created with the categorical_crossentropy loss function and "ADAM" optimizer.

Step 7: Train the model:

Once the convolution, hidden and output layers are modelled, it is time to train the model with the training dataset images. It is done with the help of model fit generator function with the required number of steps, validation of data, accuracy, and so on. As the model is trained in steps, the accuracy increases in the prediction and loss function reduces. It is good to fit the model with greater than 90% accuracy.

Step 8: Save the model:

The trained model is saved as .h5 file. The file will be added in IBM cloud.

Step 9: Test the model:

In this step, the .h5 model file is first loaded. Then, the image, for which the prediction is required, is processed to have greyscale and 128x128 pixel size, and fed into the model in array form. The model predicts the image, and throws the classification index as the output.

Step 10: Deployment of the model in IBM Watson Machine Learning:

The following steps are followed to deploy the model in IBM Watson Machine Learning software:

- Install IBM Watson Machine Learning Client, and provide the credentials for verification (URL and API Key).
- Create GUID space, and set it as the default location for storing the model.
- Convert the .h5 file into .tgz model file, and create the dataset.
- Create the model ID in tar.gb file format. The model is sucessfully deployed now.

Step 11: Integrating the CNN model in Browser using Flask Library:

The following steps are followed to integrate the deployed CNN model to a web browser:

- Open Offline Spyder Notebook.

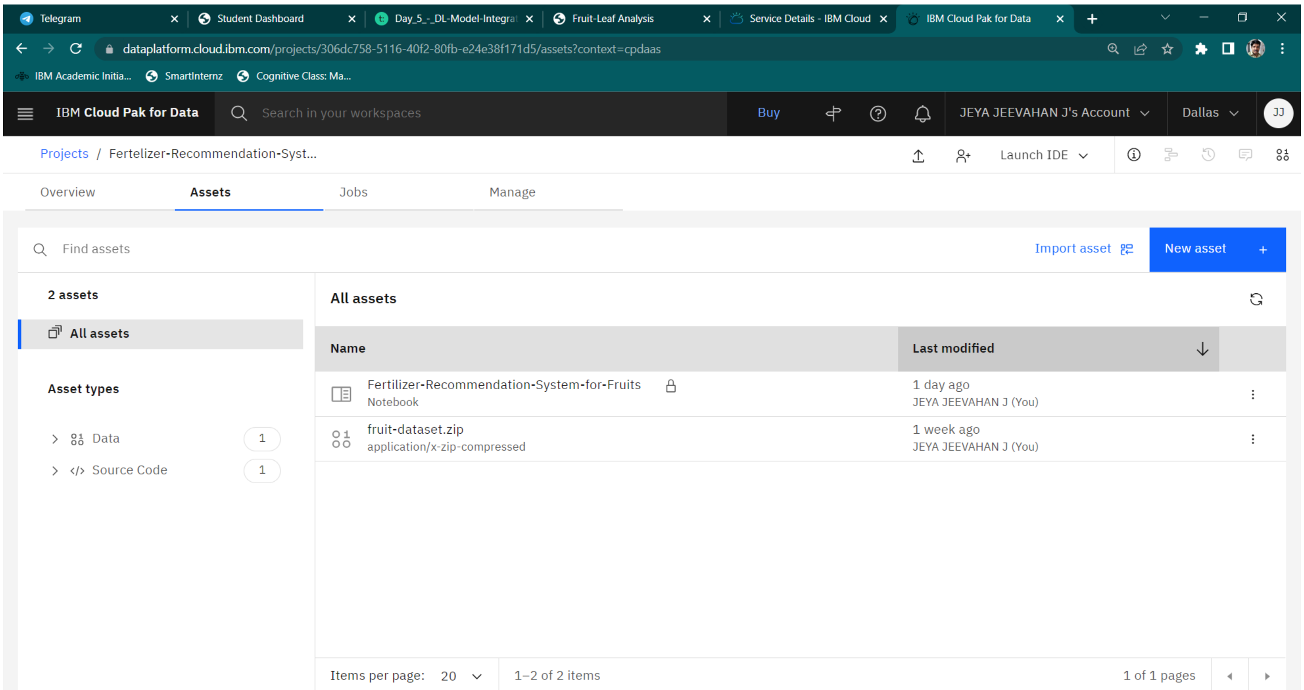
- Add Flask template folder and .h5 file developed using IBM Jupyter notebook.
- Load the model (.h5 file), and set prediction steps.
- Run the program, and check for the URL.

Step 12: Launch the Model in Web Browser and Verify:

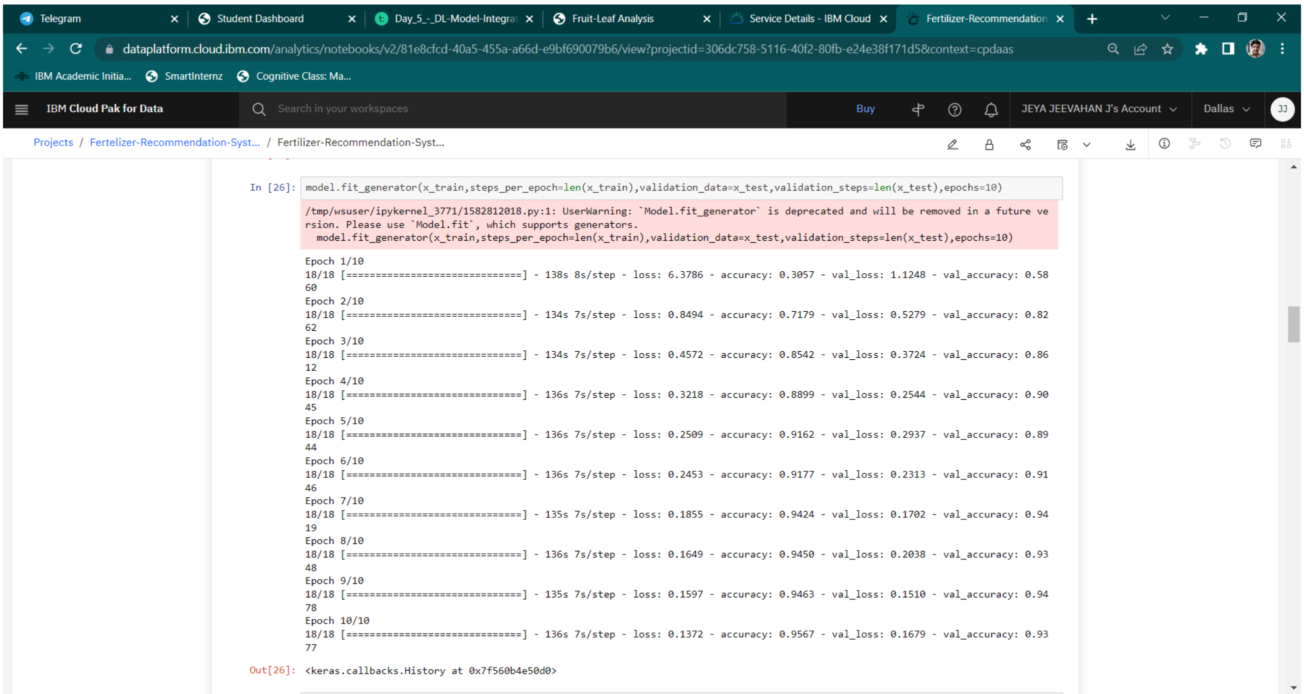
It's time to launch the model in the web browser, and test the images. Go to browser, and launch the URL. Now, you will be able to see the website with upload option. Upload the test image from local disk, and click on 'Predict'. Now, the model is run, and the predicted result is displayed.

5. OUTPUT/SCREENSHOTS OF FRUIT DISEASE PREDICTION

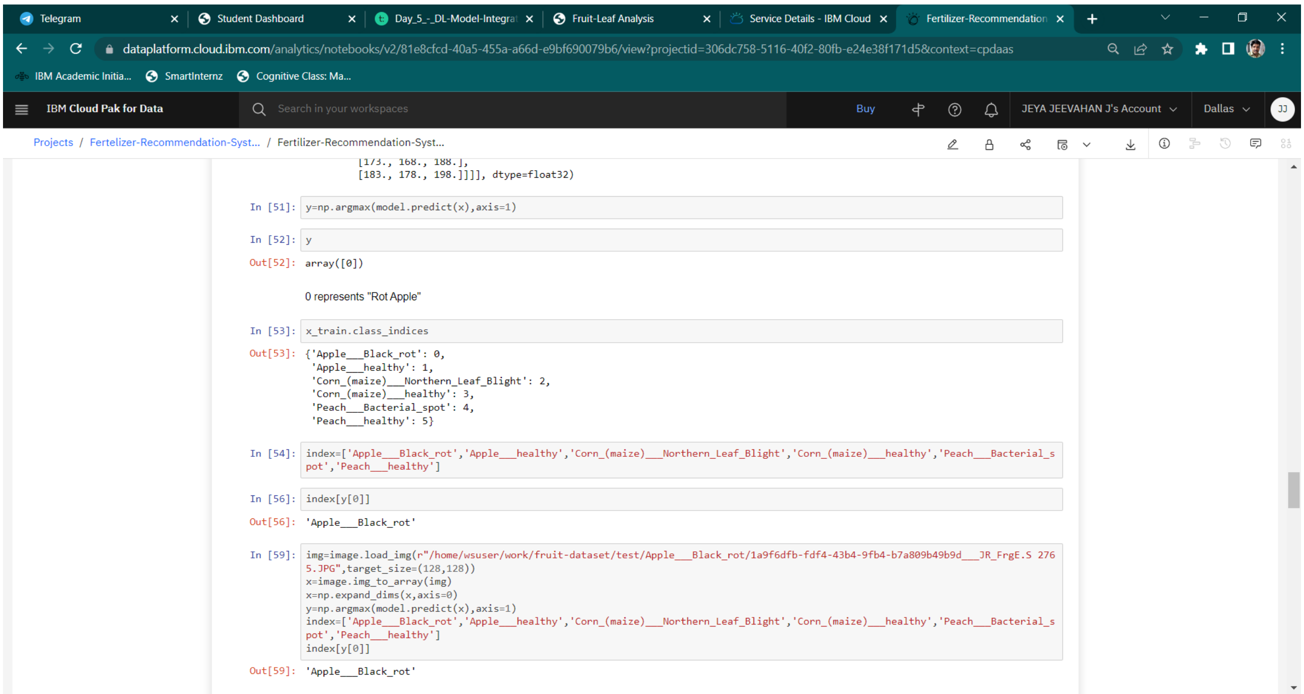
Screenshot: Project Space with Notebook and Fruit-Dataset



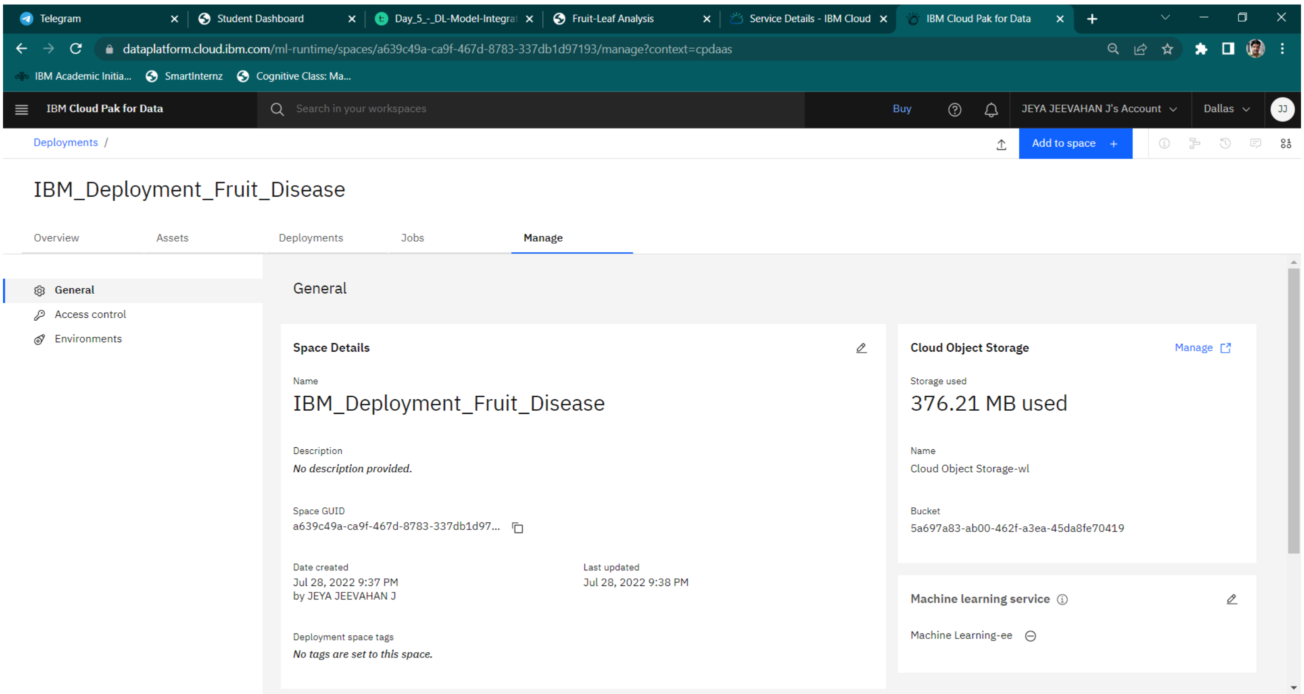
Screeshot: Training the CNN model with 93% accuracy



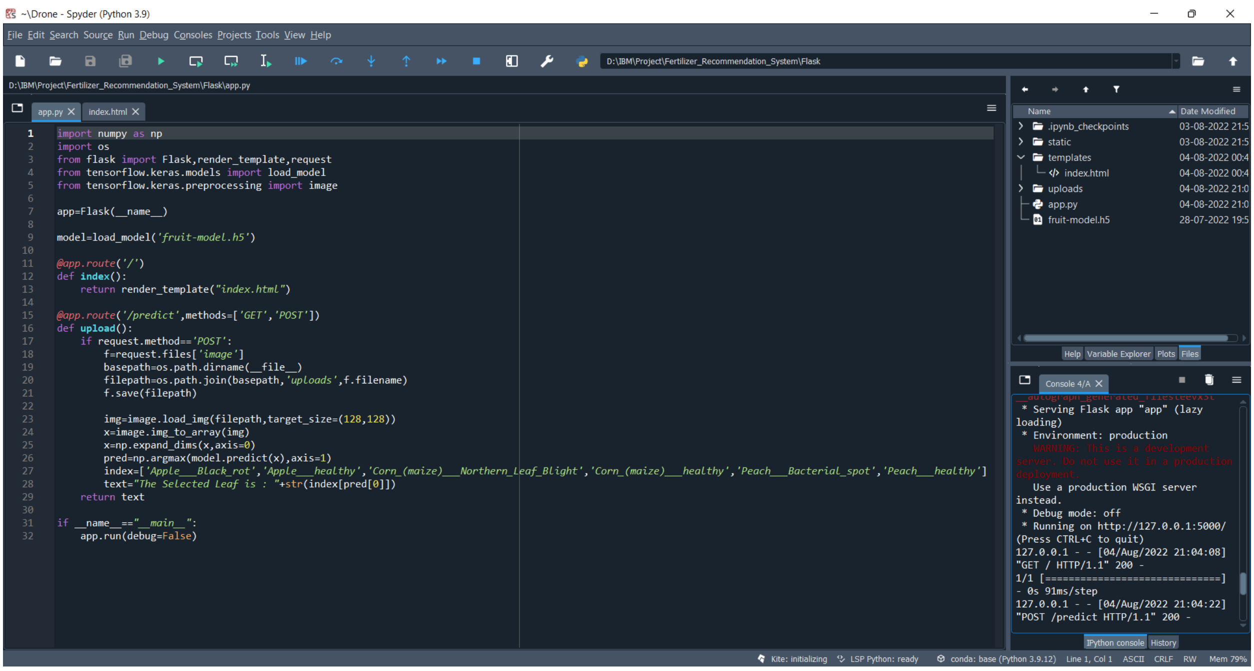
Screenshot: Testing the developed model with any test image.



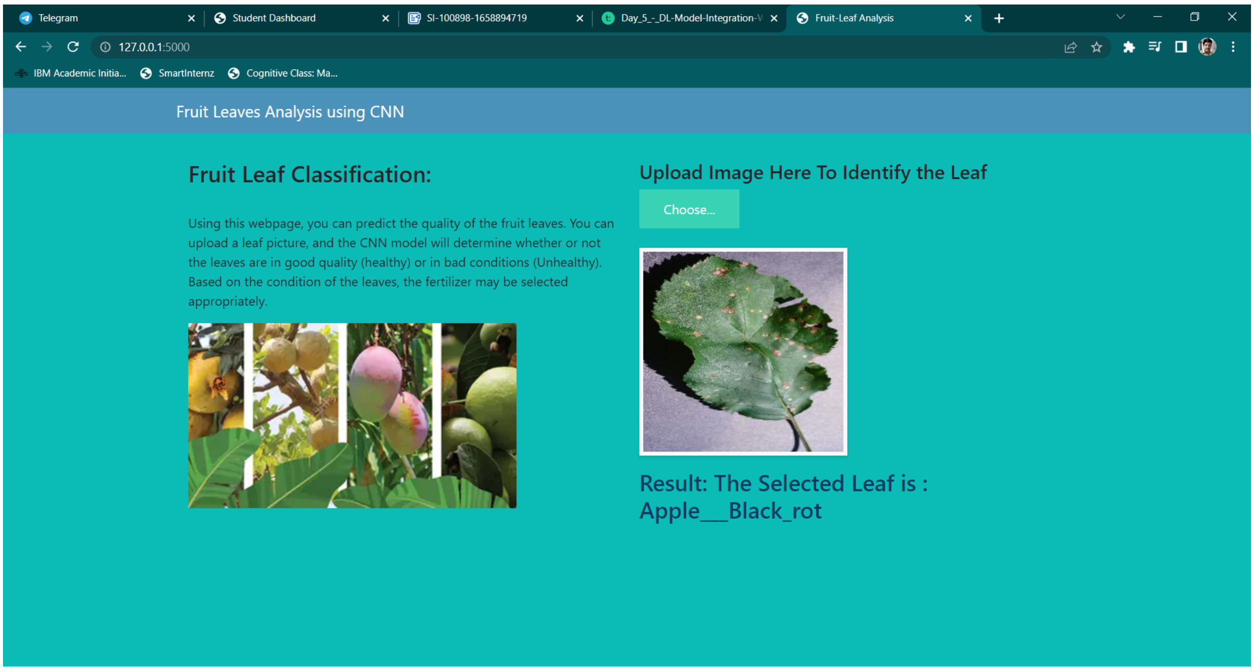
Screenshot: IBM Deployment



Screenshot: Run the developed model using Flask environment in Spyder

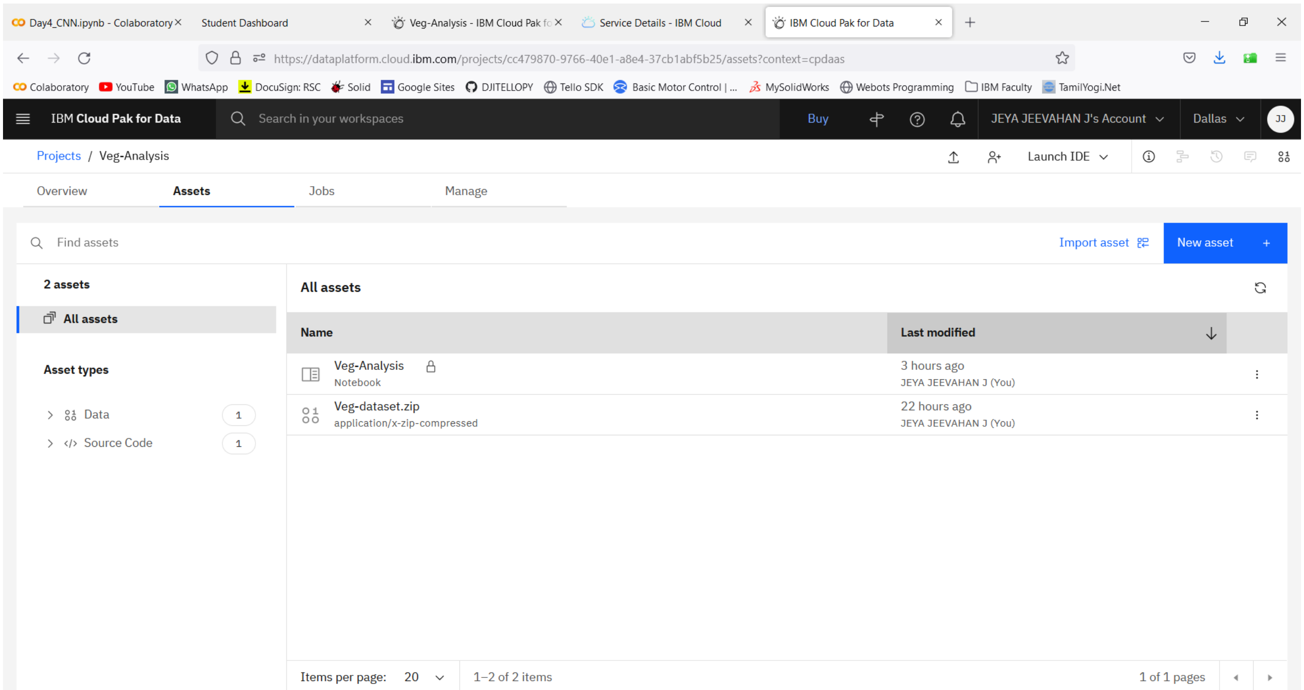


Screenshot: Output showing the correct prediction of Fruit Leaf Disease in URL.

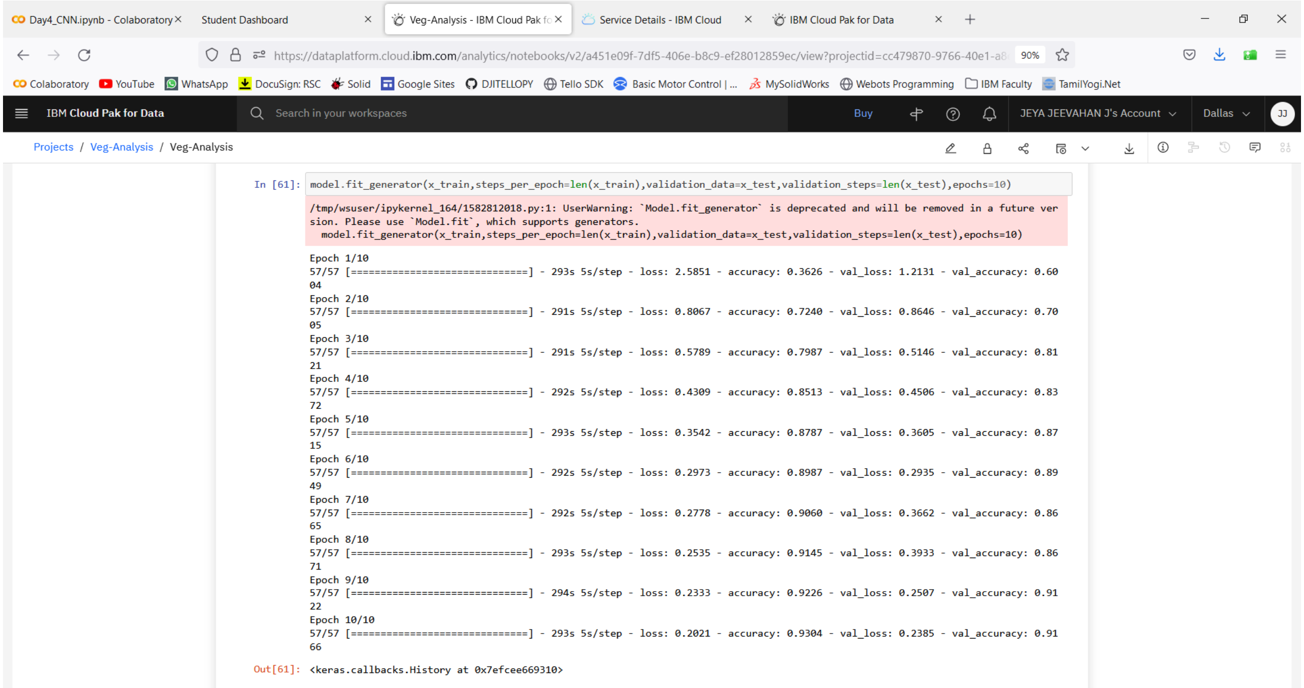


6. OUTPUT/SCREENSHOTS OF VEGETABLE DISEASE PREDICTION

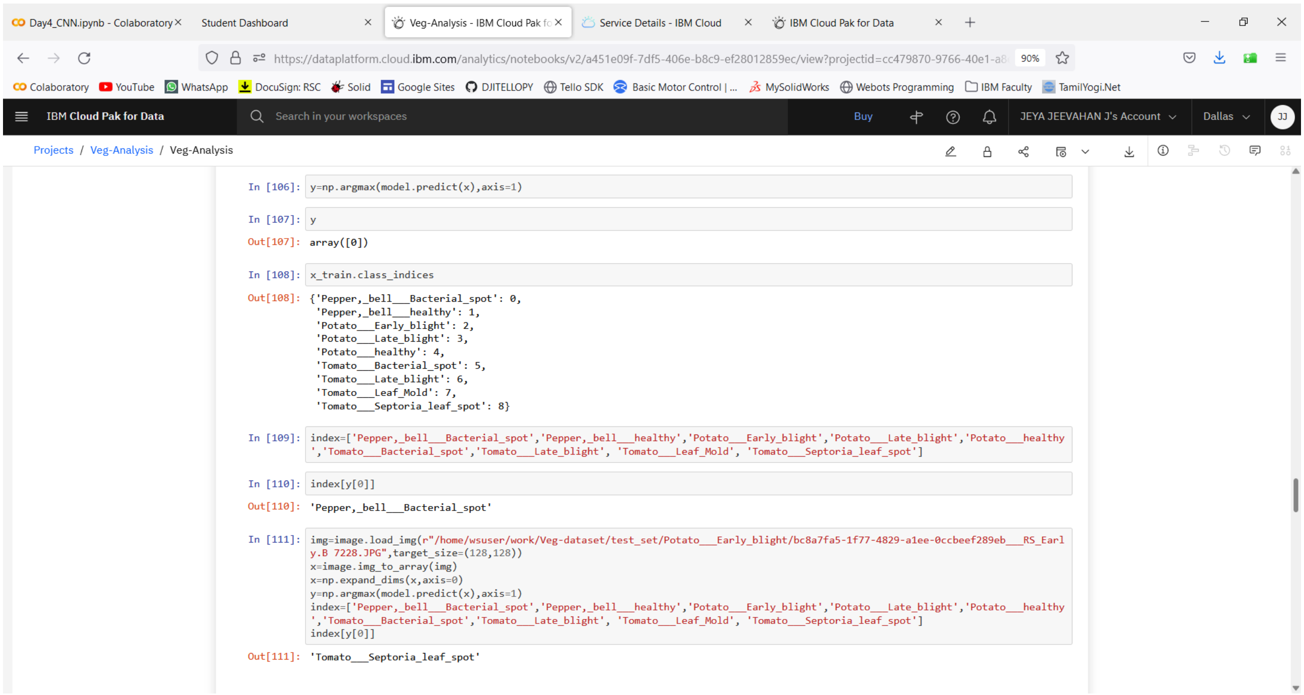
Screenshot: Project Space with Notebook and Fruit-Dataset



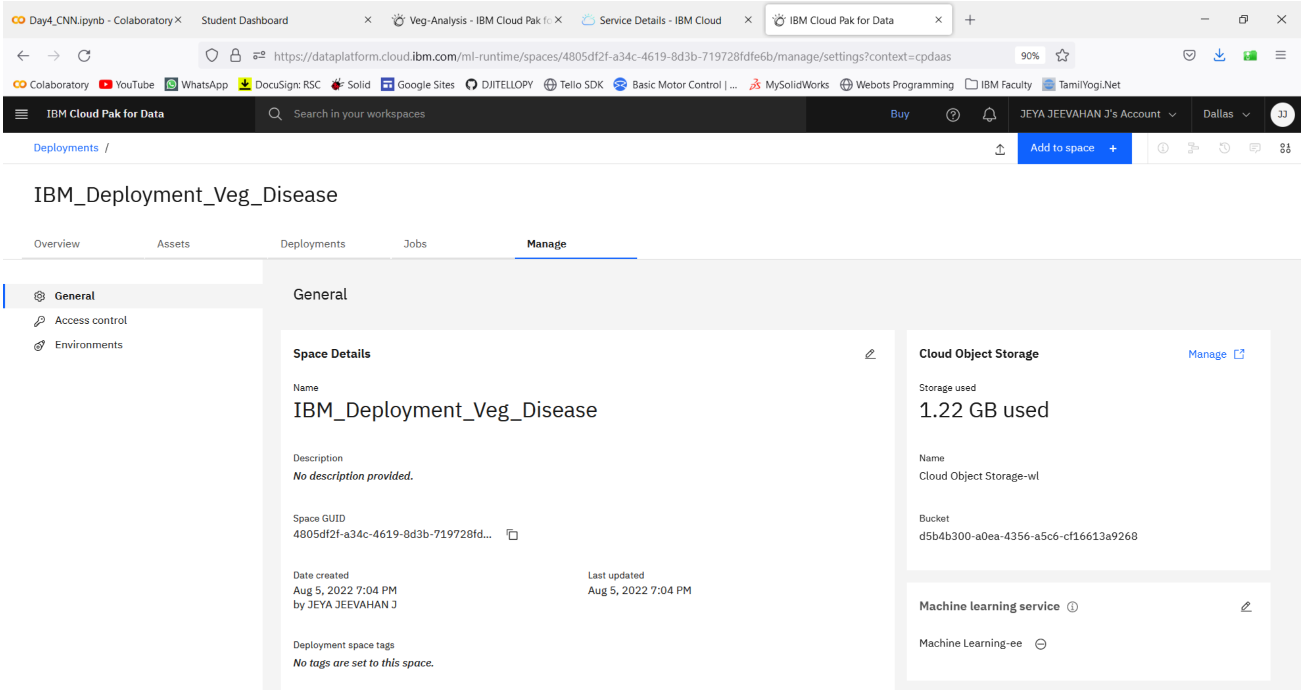
Screeshot: Training the CNN model with 93% accuracy



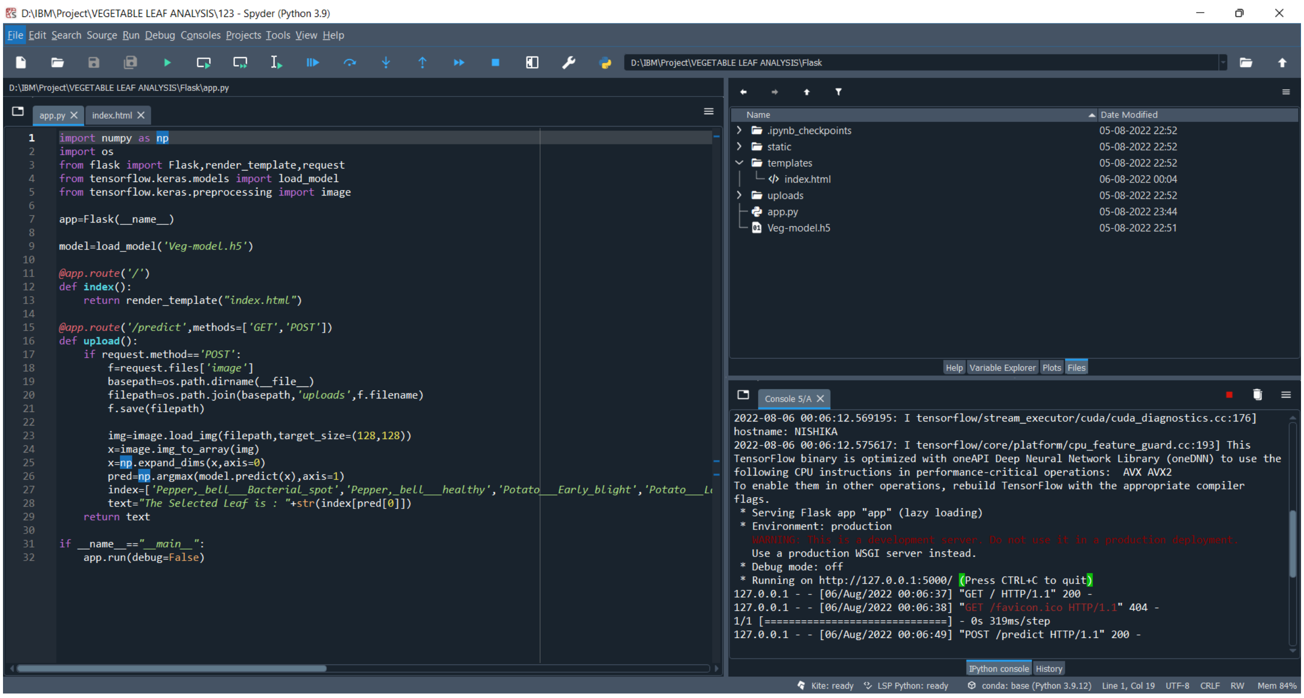
Screenshot: Testing the developed model with any test image.



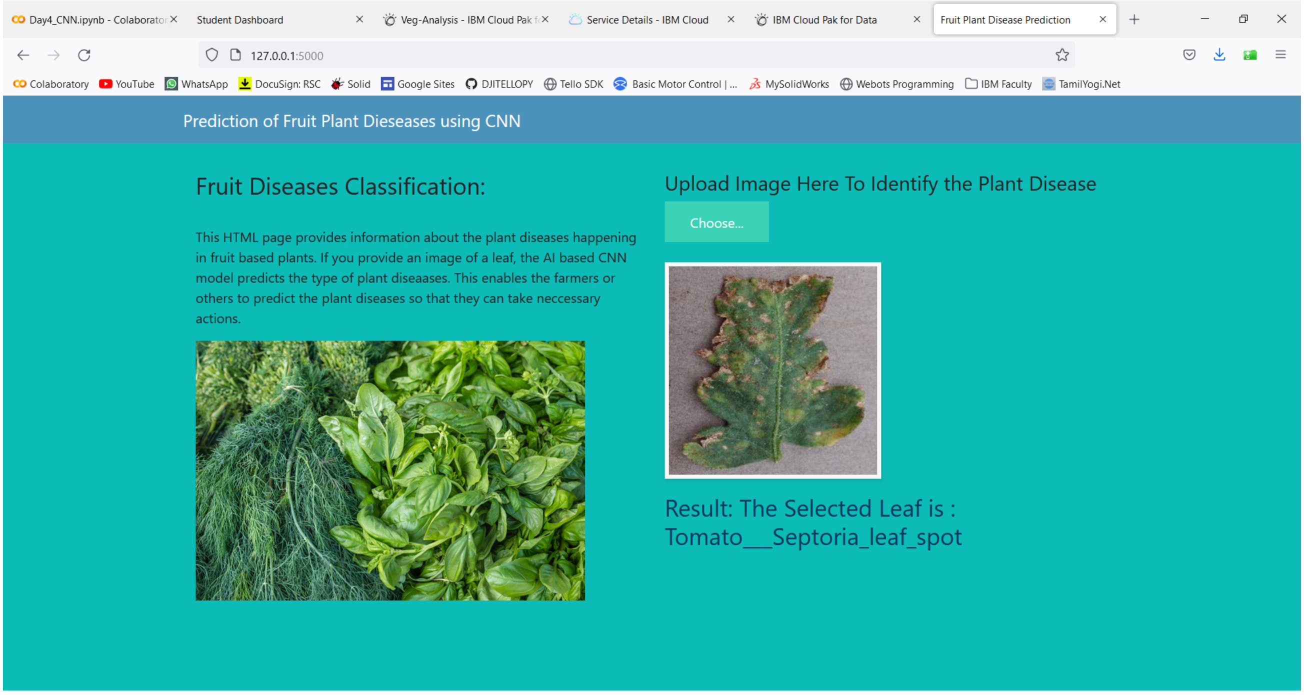
Screenshot: IBM Deployment



Screenshot: Run the developed model using Flask environment in Spyder



Screenshot: Output showing the correct prediction of Vegetable Leaf Disease in URL.



7. CONCLUSION

The CNN models were successfully developed for predicting the plant diseases for fruits and vegetables related plants. This prediction can give information on whether or not the leaves are healthy. If they are not healthy, the type of diseases will be identified by the CNN model and the same is displayed. The CNN models created in this project work gave the image classification with more than 90% accuracy, and the same can be used in the prediction of plant diseases in an efficient manner.

APPENDIX

A. SOURCE CODE FOR FRUIT DISEASE PREDICTION

A1. IBM Jupyter Notebook

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

Load the Image Dataset

```
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_520603f795bd4c36bda2fa2f9d06f6a8 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='FuPeYouN7pXgz-Losv6x-xJhGC5nyxO5NFzBtwG6aDR6',
    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_520603f795bd4c36bda2fa2f9d06f6a8.get_object(Bucket='fertelizerrecommendationsystem-
donotdelete-pr-2jbmdaxyrbwa8f', 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/
```

Unzip the folder

```
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)
ls
fruit-dataset/
pwd
'/home/wsuser/work'
```

Image Augmentation

```
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./255)
ls
fruit-dataset/
pwd
'/home/wsuser/work'
x_train=train_datagen.flow_from_directory(r"/home/wsuser/work/fruit-
dataset/train",target_size=(128,128),class_mode='categorical',batch_size=300)
Found 5384 images belonging to 6 classes.
x_test=test_datagen.flow_from_directory(r"/home/wsuser/work/fruit-
dataset/test",target_size=(128,128),class_mode='categorical',batch_size=100)
Found 1686 images belonging to 6 classes.
x_train.class_indices
{'Apple___Black_rot': 0,
 'Apple___healthy': 1,
 'Corn_(maize)___Northern_Leaf_Blight': 2,
 'Corn_(maize)___healthy': 3,
 'Peach___Bacterial_spot': 4,
 'Peach___healthy': 5}
```

CNN

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

```
model=Sequential()
model.add(Convolution2D(64,(3,3),input_shape=(128,128,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.summary()
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 64)      1792      
                                                                 
 max_pooling2d (MaxPooling2D  (None, 63, 63, 64)       0         
 )                                                               
                                                                 
 flatten (Flatten)           (None, 254016)            0         
                                                                 
=================================================================
Total params: 1,792
Trainable params: 1,792
Non-trainable params: 0
_________________________________________________________________
```

Hidden Layer

```
model.add(Dense(150,activation='relu'))
model.add(Dense(50,activation='relu'))
```

Output Layer

```
model.add(Dense(6,activation='softmax')) # 6 is the number of catagories
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
len(x_train)
# NUMBER OF IMAGES DIVIDED BY BATCH SIZE
18
model.fit_generator(x_train,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len(x_test),epochs=10)
/tmp/wsuser/ipykernel_3771/1582812018.py:1: 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=len(x_train),validation_data=x_test,validation_steps=len(x_test),epochs=10)
Epoch 1/10
18/18 [==============================] - 138s 8s/step - loss: 6.3786 - accuracy: 0.3057 - val_loss: 1.1248 - val_accuracy: 0.5860
Epoch 2/10
18/18 [==============================] - 134s 7s/step - loss: 0.8494 - accuracy: 0.7179 - val_loss: 0.5279 - val_accuracy: 0.8262
Epoch 3/10
18/18 [==============================] - 134s 7s/step - loss: 0.4572 - accuracy: 0.8542 - val_loss: 0.3724 - val_accuracy: 0.8612
Epoch 4/10
18/18 [==============================] - 136s 7s/step - loss: 0.3218 - accuracy: 0.8899 - val_loss: 0.2544 - val_accuracy: 0.9045
Epoch 5/10
18/18 [==============================] - 136s 7s/step - loss: 0.2509 - accuracy: 0.9162 - val_loss: 0.2937 - val_accuracy: 0.8944
Epoch 6/10
18/18 [==============================] - 136s 7s/step - loss: 0.2453 - accuracy: 0.9177 - val_loss: 0.2313 - val_accuracy: 0.9146
```

Epoch 7/10
18/18 [==============================] - 135s 7s/step - loss: 0.1855 - accuracy: 0.9424 - val_loss: 0.1702 - val_accuracy: 0.9419
Epoch 8/10
18/18 [==============================] - 136s 7s/step - loss: 0.1649 - accuracy: 0.9450 - val_loss: 0.2038 - val_accuracy: 0.9348
Epoch 9/10
18/18 [==============================] - 135s 7s/step - loss: 0.1597 - accuracy: 0.9463 - val_loss: 0.1510 - val_accuracy: 0.9478
Epoch 10/10
18/18 [==============================] - 136s 7s/step - loss: 0.1372 - accuracy: 0.9567 - val_loss: 0.1679 - val_accuracy: 0.9377
<keras.callbacks.History at 0x7f560b4e50d0>
ls
fruit-dataset/
model.save('fruit-model.h5')
ls
fruit-dataset/ fruit-model.h5

Test the Model

import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
# Load the model
model=load_model('fruit-model.h5')
pwd
'/home/wsuser/work'
img=image.load_img(r"/home/wsuser/work/fruit-dataset/test/Apple___Black_rot/1a9f6dfb-fdf4-43b4-9fb4-b7a809b49b9d___JR_FrgE.S 2765.JPG",target_size=(128,128))
img

x=image.img_to_array(img)
x
array([[[87., 78., 105.],
        [122., 113., 140.],
        [120., 111., 138.],
        ...,
        [84., 76., 100.],
        [123., 115., 139.],
        [87., 79., 103.]],

       [[131., 123., 147.],
        [76., 68., 92.],
        [111., 103., 127.],
        ...,
        [101., 93., 117.],
        [97., 89., 113.],
        [130., 122., 146.]],

       [[115., 105., 130.],

```
        [111., 101., 126.],
        [125., 115., 140.],
        ...,
        [120., 112., 136.],
        [105.,  97., 121.],
        [ 78.,  70.,  94.]],

       ...,

       [[166., 159., 177.],
        [174., 167., 185.],
        [172., 165., 183.],
        ...,
        [186., 181., 201.],
        [188., 183., 203.],
        [191., 186., 206.]],

       [[166., 159., 177.],
        [183., 176., 194.],
        [183., 176., 194.],
        ...,
        [176., 171., 191.],
        [174., 169., 189.],
        [190., 185., 205.]],

       [[177., 170., 188.],
        [178., 171., 189.],
        [173., 166., 184.],
        ...,
        [183., 178., 198.],
        [173., 168., 188.],
        [183., 178., 198.]]], dtype=float32)
x=np.expand_dims(x,axis=0)
x
array([[[[ 87.,  78., 105.],
         [122., 113., 140.],
         [120., 111., 138.],
         ...,
         [ 84.,  76., 100.],
         [123., 115., 139.],
         [ 87.,  79., 103.]],

        [[131., 123., 147.],
         [ 76.,  68.,  92.],
         [111., 103., 127.],
         ...,
         [101.,  93., 117.],
         [ 97.,  89., 113.],
         [130., 122., 146.]],

        [[115., 105., 130.],
         [111., 101., 126.],
         [125., 115., 140.],
         ...,
         [120., 112., 136.],
         [105.,  97., 121.],
         [ 78.,  70.,  94.]],
```

```
        ...,

        [[166., 159., 177.],
         [174., 167., 185.],
         [172., 165., 183.],
         ...,
         [186., 181., 201.],
         [188., 183., 203.],
         [191., 186., 206.]],

        [[166., 159., 177.],
         [183., 176., 194.],
         [183., 176., 194.],
         ...,
         [176., 171., 191.],
         [174., 169., 189.],
         [190., 185., 205.]],

        [[177., 170., 188.],
         [178., 171., 189.],
         [173., 166., 184.],
         ...,
         [183., 178., 198.],
         [173., 168., 188.],
         [183., 178., 198.]]]], dtype=float32)
y=np.argmax(model.predict(x),axis=1)
y
array([0])
0 represents "Rot Apple"
x_train.class_indices
{'Apple___Black_rot': 0,
 'Apple___healthy': 1,
 'Corn_(maize)___Northern_Leaf_Blight': 2,
 'Corn_(maize)___healthy': 3,
 'Peach___Bacterial_spot': 4,
 'Peach___healthy': 5}
index=['Apple___Black_rot','Apple___healthy','Corn_(maize)___Northern_Leaf_Blight','Corn_(maize)___healthy','Pe
ach___Bacterial_spot','Peach___healthy']
index[y[0]]
'Apple___Black_rot'
img=image.load_img(r"/home/wsuser/work/fruit-dataset/test/Apple___Black_rot/1a9f6dfb-fdf4-43b4-9fb4-
b7a809b49b9d___JR_FrgE.S 2765.JPG",target_size=(128,128))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
y=np.argmax(model.predict(x),axis=1)
index=['Apple___Black_rot','Apple___healthy','Corn_(maize)___Northern_Leaf_Blight','Corn_(maize)___healthy','Pe
ach___Bacterial_spot','Peach___healthy']
index[y[0]]
'Apple___Black_rot'
```

IBM Deployment

```
!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 17.2 MB/s eta 0:00:01
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: requests in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from
watson-machine-learning-client) (2.26.0)
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: boto3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from
watson-machine-learning-client) (1.18.21)
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: 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: tabulate in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from
watson-machine-learning-client) (0.8.9)
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: 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: 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: 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: 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
from ibm_watson_machine_learning import APIClient
wml_credentials={
   "url":"https://us-south.ml.cloud.ibm.com",
   "apikey":"rYTfECgGLTX62qHksGC-xucEGWt_yWy_THLJbkU3WOqx"
}
client=APIClient(wml_credentials)
client
<ibm_watson_machine_learning.client.APIClient at 0x7f858c5c6280>
def guid_space_name(client,IBM_Deployment_Fruit_Disease):
   space=client.spaces.get_details()
   return(next(item for item in space['resources'] if
item['entity']['name']==IBM_Deployment_Fruit_Disease)['metadata']['id'])
space_uid=guid_space_name(client,'IBM_Deployment_Fruit_Disease')
print(space_uid)
a639c49a-ca9f-467d-8783-337db1d97193
client.set.default_space(space_uid)
'SUCCESS'
client.software_specifications.list()
```

```
----------------------------  ----------------------------------  ----
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
----------------------------  ----------------------------------  ----
Note: Only first 50 records were displayed. To display more use 'limit' parameter.
software_space_uid=client.software_specifications.get_uid_by_name('tensorflow_rt22.1-py3.9')
software_space_uid
'acd9c798-6974-5d2f-a657-ce06e986df4d'
ls
```

fruit-dataset/ fruit-model.h5
!tar -zcvf fruit-model.tgz fruit-model.h5
fruit-model.h5
ls
fruit-dataset/ fruit-model.h5 fruit-model.tgz
model_details=client.repository.store_model(model='fruit-model.tgz',
    meta_props={
        client.repository.ModelMetaNames.NAME:"CNN_Model",
        client.repository.ModelMetaNames.TYPE:'tensorflow_2.7',
        client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_space_uid
    })
model_id=client.repository.get_model_id(model_details)
model_id
'468d1eb1-3800-4b0e-8207-61ab4a0036a2'
client.repository.download(model_id,'fruit-model.tar.gb')
Successfully saved model content to file: 'fruit-model.tar.gb'
'/home/wsuser/work/fruit-model.tar.gb'
ls
fruit-dataset/ fruit-model.h5 fruit-model.tar.gb fruit-model.tgz

A2. Spyder IDE

```
import numpy as np
import os
from flask import Flask,render_template,request
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image

app=Flask(__name__)

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

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

@app.route('/predict',methods=['GET','POST'])
def upload():
    if request.method=='POST':
        f=request.files['image']
        basepath=os.path.dirname(__file__)
        filepath=os.path.join(basepath,'uploads',f.filename)
        f.save(filepath)

        img=image.load_img(filepath,target_size=(128,128))
        x=image.img_to_array(img)
        x=np.expand_dims(x,axis=0)
        pred=np.argmax(model.predict(x),axis=1)

index=['Apple___Black_rot','Apple___healthy','Corn_(maize)___Northern_Leaf_Blight
','Corn_(maize)___healthy','Peach___Bacterial_spot','Peach___healthy']
        text="The Selected Leaf is : "+str(index[pred[0]])
    return text

if __name__=="__main__":
    app.run(debug=False)
```

B. SOURCE CODE FOR VEGETABLE DISEASE PREDICTION

A1. IBM Jupyter Notebook

```
pwd
'/home/wsuser/work'
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_520603f795bd4c36bda2fa2f9d06f6a8 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='Z7FhoL0Uz9mKepzPieFt8TPouKLVCTnph1c-wKY4nX36',
    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_5 =
client_520603f795bd4c36bda2fa2f9d06f6a8.get_object(Bucket='veganalysis-
donotdelete-pr-mgno2nzecteeky', Key='Veg-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/
```

Unzip the folder

```
from io import BytesIO
import zipfile
unzip=zipfile.ZipFile(BytesIO(streaming_body_5.read()),'r')
file_paths=unzip.namelist()
for path in file_paths:
    unzip.extract(path)
ls
Veg-dataset/
pwd
'/home/wsuser/work'
```

Image Augmentation

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen=ImageDataGenerator(rescale=1./255,zoom_range=0.2,horizontal_flip=Tru
e,vertical_flip=False)
test_datagen=ImageDataGenerator(rescale=1./255)
ls
Veg-dataset/
pwd
'/home/wsuser/work'
x_train=train_datagen.flow_from_directory(r"/home/wsuser/work/Veg-
dataset/train_set",target_size=(128,128),class_mode='categorical',batch_size=200)
Found 11386 images belonging to 9 classes.
x_test=test_datagen.flow_from_directory(r"/home/wsuser/work/Veg-
```

```
dataset/test_set",target_size=(128,128),class_mode='categorical',batch_size=50)
Found 3416 images belonging to 9 classes.
x_train.class_indices
{'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

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Convolution2D,MaxPooling2D,Flatten
model=Sequential()
model.add(Convolution2D(64,(3,3),input_shape=(128,128,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.summary()
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 126, 126, 64)      1792

 max_pooling2d_1 (MaxPooling  (None, 63, 63, 64)       0
 2D)

 flatten_1 (Flatten)         (None, 254016)            0

=================================================================
Total params: 1,792
Trainable params: 1,792
Non-trainable params: 0
_________________________________________________________________
```

Hidden Layer

```
model.add(Dense(150,activation='relu'))
model.add(Dense(50,activation='relu'))
```

Output Layer

```
model.add(Dense(9,activation='softmax')) # 9 is the number of catagories
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy
'])
len(x_train)
# NUMBER OF IMAGES DIVIDED BY BATCH SIZE
57
model.fit_generator(x_train,steps_per_epoch=len(x_train),validation_data=x_test,va
lidation_steps=len(x_test),epochs=10)
/tmp/wsuser/ipykernel_164/1582812018.py:1: 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=len(x_train),validation_data=x_test,va
lidation_steps=len(x_test),epochs=10)
```

```
Epoch 1/10
57/57 [==============================] - 293s 5s/step - loss: 2.5851 - accuracy:
0.3626 - val_loss: 1.2131 - val_accuracy: 0.6004
Epoch 2/10
57/57 [==============================] - 291s 5s/step - loss: 0.8067 - accuracy:
0.7240 - val_loss: 0.8646 - val_accuracy: 0.7005
Epoch 3/10
57/57 [==============================] - 291s 5s/step - loss: 0.5789 - accuracy:
0.7987 - val_loss: 0.5146 - val_accuracy: 0.8121
Epoch 4/10
57/57 [==============================] - 292s 5s/step - loss: 0.4309 - accuracy:
0.8513 - val_loss: 0.4506 - val_accuracy: 0.8372
Epoch 5/10
57/57 [==============================] - 293s 5s/step - loss: 0.3542 - accuracy:
0.8787 - val_loss: 0.3605 - val_accuracy: 0.8715
Epoch 6/10
57/57 [==============================] - 292s 5s/step - loss: 0.2973 - accuracy:
0.8987 - val_loss: 0.2935 - val_accuracy: 0.8949
Epoch 7/10
57/57 [==============================] - 292s 5s/step - loss: 0.2778 - accuracy:
0.9060 - val_loss: 0.3662 - val_accuracy: 0.8665
Epoch 8/10
57/57 [==============================] - 293s 5s/step - loss: 0.2535 - accuracy:
0.9145 - val_loss: 0.3933 - val_accuracy: 0.8671
Epoch 9/10
57/57 [==============================] - 294s 5s/step - loss: 0.2333 - accuracy:
0.9226 - val_loss: 0.2507 - val_accuracy: 0.9122
Epoch 10/10
57/57 [==============================] - 293s 5s/step - loss: 0.2021 - accuracy:
0.9304 - val_loss: 0.2385 - val_accuracy: 0.9166
<keras.callbacks.History at 0x7efcee669310>
ls
Veg-dataset/
model.save('Veg-model.h5')
ls
fruit-model.h5  fruit-model.tar.gb  fruit-model.tgz  Veg-dataset/  Veg-model.h5
```

Test the Model

```
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
# Load the model
model=load_model('Veg-model.h5')
pwd
'/home/wsuser/work'
img=image.load_img(r"/home/wsuser/work/Veg-
dataset/test_set/Tomato___Bacterial_spot/b6d72c2e-9e41-4215-af23-
8e84d94d839f___UF.GRC_BS_Lab Leaf 9065.JPG",target_size=(128,128))
img
```

```
x=image.img_to_array(img)
x
array([[[125., 126., 128.],
        [126., 127., 129.],
        [124., 125., 127.],
        ...,
        [118., 119., 123.],
        [116., 117., 121.],
        [110., 111., 115.]],

       [[126., 127., 129.],
        [126., 127., 129.],
        [123., 124., 126.],
        ...,
        [115., 116., 120.],
        [115., 116., 120.],
        [112., 113., 117.]],

       [[129., 130., 132.],
        [127., 128., 130.],
        [121., 122., 124.],
        ...,
        [119., 120., 124.],
        [119., 120., 124.],
        [117., 118., 122.]],

       ...,

       [[121., 123., 122.],
        [128., 130., 129.],
        [123., 125., 124.],
        ...,
        [ 99.,  99.,  97.],
        [103., 103., 101.],
        [109., 109., 107.]],

       [[115., 117., 116.],
        [112., 114., 113.],
        [109., 111., 110.],
        ...,
        [111., 111., 109.],
        [111., 111., 109.],
        [107., 107., 105.]],

       [[119., 121., 120.],
        [121., 123., 122.],
        [123., 125., 124.],
        ...,
        [103., 103., 101.],
        [104., 104., 102.],
        [109., 109., 107.]]], dtype=float32)
x=np.expand_dims(x,axis=0)
x
array([[[[125., 126., 128.],
         [126., 127., 129.],
         [124., 125., 127.],
         ...,
         [118., 119., 123.],
```

```
         [116., 117., 121.],
         [110., 111., 115.]],

        [[126., 127., 129.],
         [126., 127., 129.],
         [123., 124., 126.],
         ...,
         [115., 116., 120.],
         [115., 116., 120.],
         [112., 113., 117.]],

        [[129., 130., 132.],
         [127., 128., 130.],
         [121., 122., 124.],
         ...,
         [119., 120., 124.],
         [119., 120., 124.],
         [117., 118., 122.]],

        ...,

        [[121., 123., 122.],
         [128., 130., 129.],
         [123., 125., 124.],
         ...,
         [ 99.,  99.,  97.],
         [103., 103., 101.],
         [109., 109., 107.]],

        [[115., 117., 116.],
         [112., 114., 113.],
         [109., 111., 110.],
         ...,
         [111., 111., 109.],
         [111., 111., 109.],
         [107., 107., 105.]],

        [[119., 121., 120.],
         [121., 123., 122.],
         [123., 125., 124.],
         ...,
         [103., 103., 101.],
         [104., 104., 102.],
         [109., 109., 107.]]]], dtype=float32)
y=np.argmax(model.predict(x),axis=1)
y
array([0])
x_train.class_indices
{'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}
index=['Pepper,_bell___Bacterial_spot','Pepper,_bell___healthy','Potato___Early_b
```

```
light','Potato___Late_blight','Potato___healthy','Tomato___Bacterial_spot','Tomat
o___Late_blight', 'Tomato___Leaf_Mold', 'Tomato___Septoria_leaf_spot']
index[y[0]]
'Pepper,_bell___Bacterial_spot'
img=image.load_img(r"/home/wsuser/work/Veg-
dataset/test_set/Potato___Early_blight/bc8a7fa5-1f77-4829-a1ee-
0ccbeef289eb___RS_Early.B 7228.JPG",target_size=(128,128))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
y=np.argmax(model.predict(x),axis=1)
index=['Pepper,_bell___Bacterial_spot','Pepper,_bell___healthy','Potato___Early_b
light','Potato___Late_blight','Potato___healthy','Tomato___Bacterial_spot','Tomat
o___Late_blight', 'Tomato___Leaf_Mold', 'Tomato___Septoria_leaf_spot']
index[y[0]]
'Tomato___Septoria_leaf_spot'
```

IBM Deployment

```
!pip install watson-machine-learning-client
Requirement already satisfied: watson-machine-learning-client in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (1.0.391)
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: requests in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2.26.0)
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: tqdm in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (4.62.3)
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: boto3 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.18.21)
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: urllib3 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.26.7)
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: 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: 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-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: 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: 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: 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: 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)
from ibm_watson_machine_learning import APIClient
wml_credentials={
    "url":"https://us-south.ml.cloud.ibm.com",
    "apikey":"6Nog_LsS2bKRhyx-TXVzEcTnb-yakzm3pPDo82orK0st"
}
client=APIClient(wml_credentials)
client
<ibm_watson_machine_learning.client.APIClient at 0x7efccc10d700>
def guid_space_name(client,IBM_Deployment_Veg_Disease):
    space=client.spaces.get_details()
    return(next(item for item in space['resources'] if item['entity']['name']==IBM_Deployment_Veg_Disease)['metadata']['id'])
space_uid=guid_space_name(client,'IBM_Deployment_Veg_Disease')
print(space_uid)
4805df2f-a34c-4619-8d3b-719728fdfe6b
client.set.default_space(space_uid)
'SUCCESS'
client.software_specifications.list()
-----------------------------  ------------------------------------  ----
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
-----------------------------  ------------------------------------  ----
Note: Only first 50 records were displayed. To display more use 'limit' parameter.
software_space_uid=client.software_specifications.get_uid_by_name('tensorflow_rt22.
1-py3.9')
software_space_uid
'acd9c798-6974-5d2f-a657-ce06e986df4d'
ls
fruit-model.h5  fruit-model.tar.gb  fruit-model.tgz  Veg-dataset/  Veg-model.h5
!tar -zcvf Veg-model.tgz Veg-model.h5
Veg-model.h5
ls
fruit-model.h5      fruit-model.tgz  Veg-model.h5
fruit-model.tar.gb  Veg-dataset/     Veg-model.tgz
model_details=client.repository.store_model(model='Veg-model.tgz',
    meta_props={
                client.repository.ModelMetaNames.NAME:"CNN_Model",
                client.repository.ModelMetaNames.TYPE:'tensorflow_2.7',

client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_space_uid
    })
model_id=client.repository.get_model_id(model_details)
model_id
'4799586d-1f3c-444f-abb0-e3ff94438704'
client.repository.download(model_id,'Veg-model.tar.gb')
Successfully saved model content to file: 'Veg-model.tar.gb'
'/home/wsuser/work/Veg-model.tar.gb'
ls
fruit-model.h5      fruit-model.tgz  Veg-model.h5      Veg-model.tgz
fruit-model.tar.gb  Veg-dataset/     Veg-model.tar.gb
```

A2. Spyder IDE

```
import numpy as np
import os
from flask import Flask,render_template,request
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image

app=Flask(__name__)

model=load_model('Veg-model.h5')

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

@app.route('/predict',methods=['GET','POST'])
def upload():
   if request.method=='POST':
      f=request.files['image']
      basepath=os.path.dirname(__file__)
      filepath=os.path.join(basepath,'uploads',f.filename)
      f.save(filepath)

      img=image.load_img(filepath,target_size=(128,128))
      x=image.img_to_array(img)
      x=np.expand_dims(x,axis=0)
      pred=np.argmax(model.predict(x),axis=1)

index=['Pepper,_bell___Bacterial_spot','Pepper,_bell___healthy','Potato___Early_blight','Potato___Late_blight','Pota
to___healthy','Tomato___Bacterial_spot','Tomato___Late_blight', 'Tomato___Leaf_Mold',
'Tomato___Septoria_leaf_spot']
      text="The Selected Leaf is : "+str(index[pred[0]])
   return text

if __name__=="__main__":
   app.run(debug=False)
```
