PROJECT REPORT

Fertilizers Recommendation System for Disease Prediction

Team ID: PNT2022TMID49029

TEAM MEMBERS:

P.NAGAJOTHI (921619106041)

M.BHARATHI PRIYA (921619106009)

T.DIVYADHARSHINI(921619106018)

S.YUVASHREE(921619106067)

ABSTRACT:

- ✓ Agriculture is the most important sector in today's life. Most plants are affected by a wide variety of bacterial and fungal diseases. Diseases on plants placed a major constraint on the production and a major threat to food security. Hence, early and accurate identification of plant diseases is essential to ensure high quantity and best quality. In recent years, the number of diseases on plants and the degree of harm caused has increased due to the variation in pathogen varieties, changes in cultivation methods, and inadequate plant protection techniques.
- ✓ 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.

Purpose:

- ✓ To Detect and recognize the plant diseases and to recommend fertilizer, it is necessary to provide symptoms in identifying the disease at its earliest. Hence the authors proposed and implemented new fertilizers Recommendation System for crop disease prediction.

LITERATURE SURVEY:

Existing Problem:

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

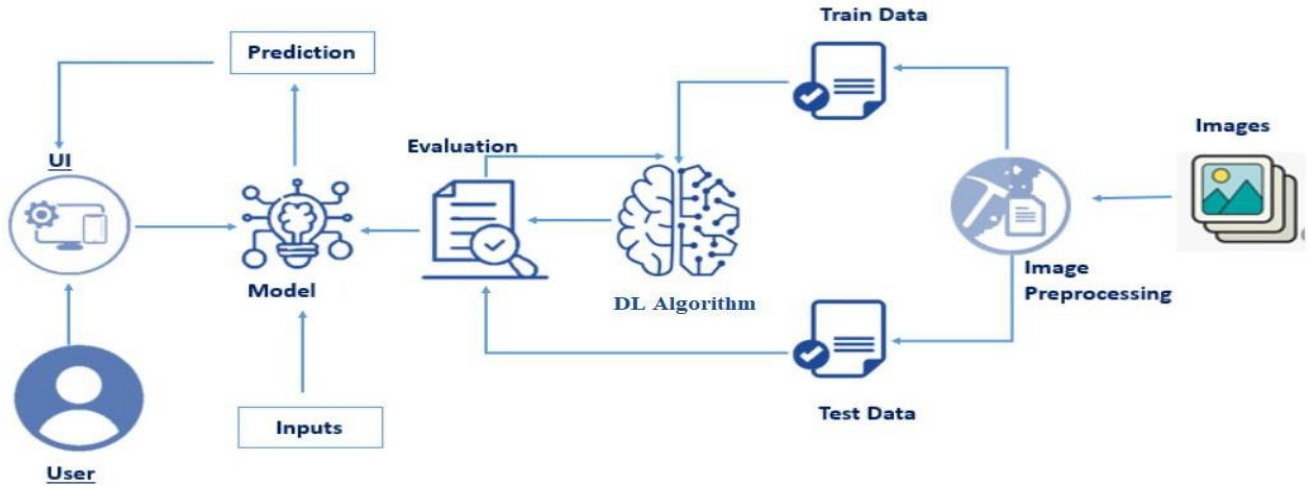
Proposed solution:

So, we have built Web Application where:

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

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

TECHNICAL ARCHITECTURE:



HARDWARE / SOFTWARE REQUIREMENTS:

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

Software's:

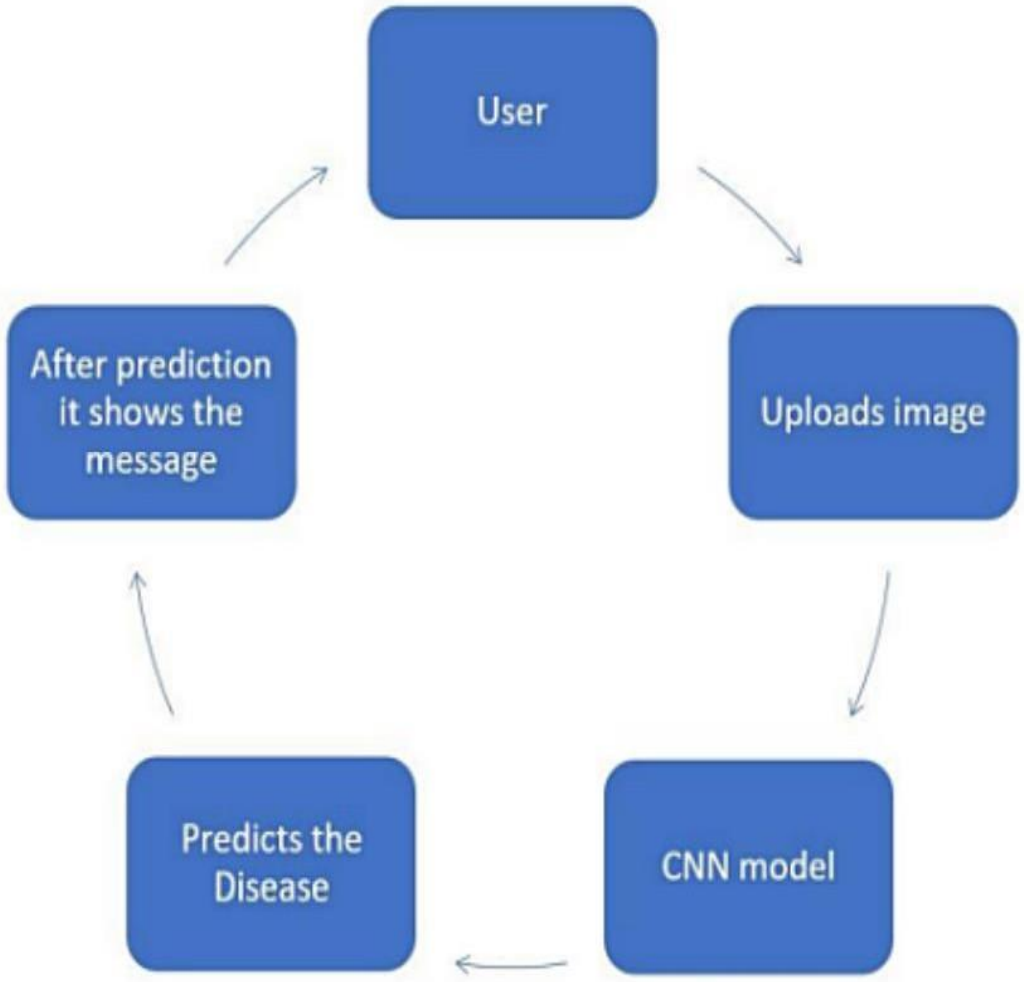
- Anaconda Navigator
- py charm
- Visual studio code
- Jupiter notebook
- IBM Watson studio

Packages:

- Tensor flow
- Keras
- Flask
- numpy
- Pandas

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

FLOWCHART:



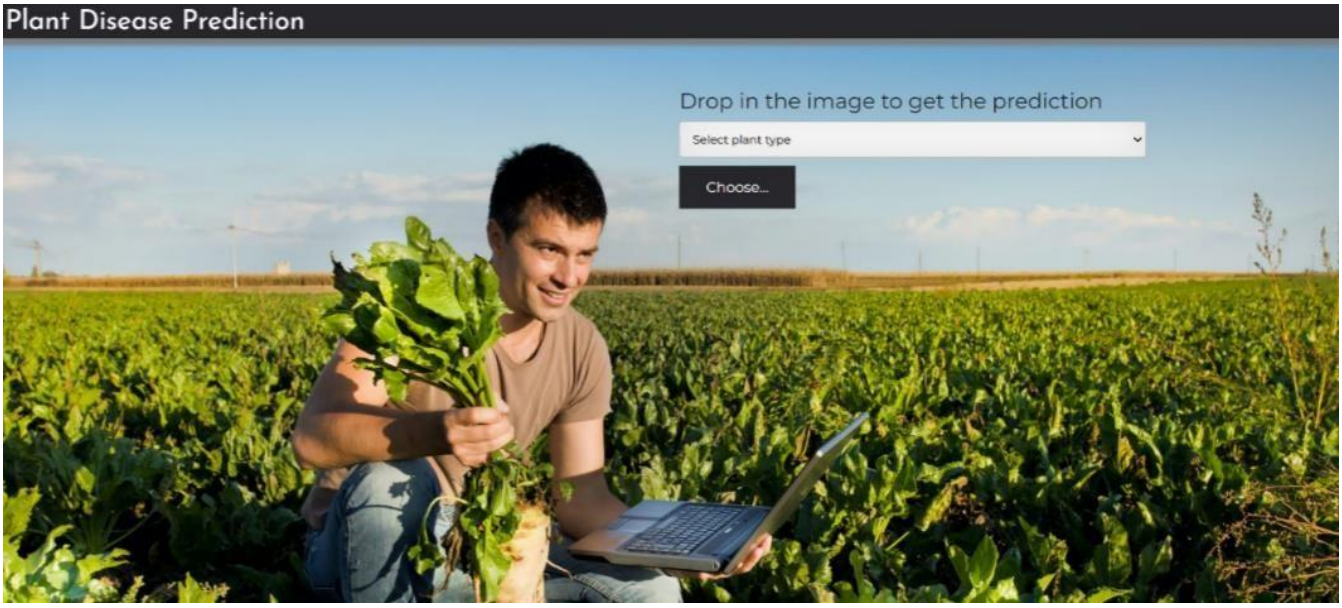
To accomplish the above task, you must complete the below activities and tasks:

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

OUTPUT:

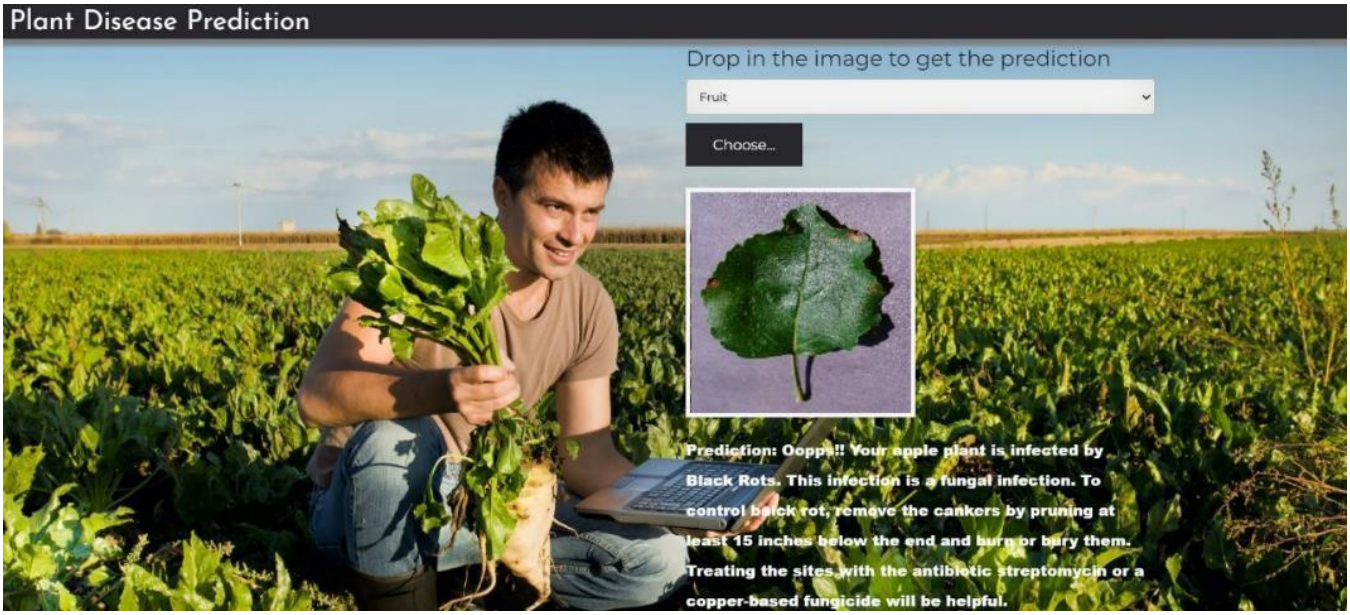
Home Page:

Prediction Page:

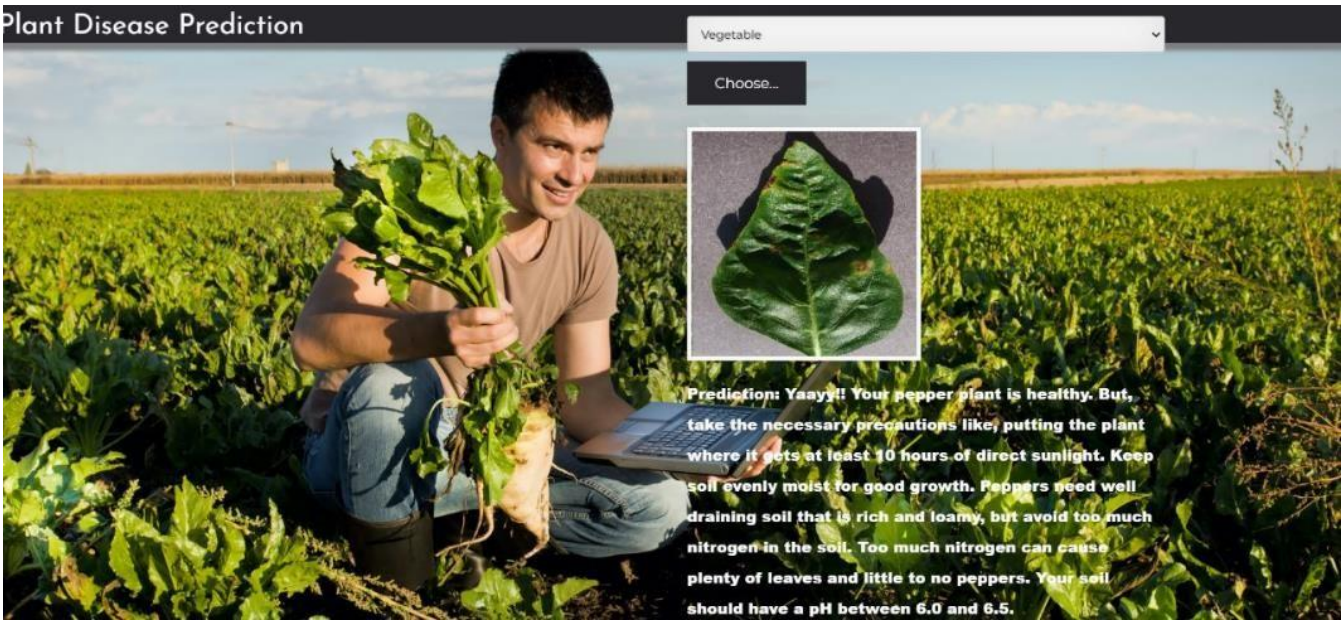


Result Page:

- ⇨ **Fruit:**



- ⇨ **Vegetable:**



ADVANTAGES:

- The proposed model could predict the disease just from the image of a part icular plant.
- Easy to use UI.
- Model has some good accuracy in detecting the plant just by taking the input(leaf).

APPLICATIONS:

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

CONCLUSION:

- Agriculture is the most important sector in today's life. Most plants are affected by a wide variety of bacterial and fungal diseases. Diseases on plants placed a major constraint on the production and a major threat to food security. Hence, early and accurate identification of plant diseases is essential to ensure high quantity and best quality.
- In recent years, the number of diseases on plants and the degree of harm caused has increased due to the variation in pathogen varieties, changes in cultivation methods, and inadequate plant protection techniques. Usage of such applications could help the farmers to necessary precautions so that they don't face any loss as such.

FUTURE SCOPE:

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

- This can be also done in Mobile applications like android, ios. It helps in many ways to improve the agriculture in cultivation of crops and predict the correct fertilizers to the crops.

APPENDIX:

Source Code:

```
In [1]: ls
```

```
In [2]: pwd
```

```
Out[2]: '/home/wsuser/work'
```

```
In [8]: !pip install keras==2.7.0
        !pip install tensorflow==2.5.0
```

```
Collecting keras==2.7.0
  Using cached keras-2.7.0-py2.py3-none-any.whl (1.3 MB)
Installing collected packages: keras
  Attempting uninstall: keras
    Found existing installation: Keras 2.2.4
    Uninstalling Keras-2.2.4:
      Successfully uninstalled Keras-2.2.4
Successfully installed keras-2.7.0
Requirement already satisfied: tensorflow==2.5.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (2.5.0)
Requirement already satisfied: protobuf>=3.9.2 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorflow==2.5.
0) (3.19.1)
Requirement already satisfied: h5py~=3.1.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorflow==2.5.0)
(3.1.0)
Requirement already satisfied: astunparse~=1.6.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorflow==2.
5.0) (1.6.3)
Requirement already satisfied: keras-nightly~=2.5.0.dev in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorf
low==2.5.0) (2.5.0.dev2021032900)
Requirement already satisfied: termcolor~=1.1.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorflow==2.
5.0) (1.1.0)
Requirement already satisfied: flatbuffers~=1.12.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorflow==
2.5.0) (1.12)
Requirement already satisfied: wrapt~=1.12.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorflow==2.5.0)
(1.12.1)
Requirement already satisfied: six~=1.15.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorflow==2.5.0)
```

```
Requirement already satisfied: typing-extensions~=3.7.4 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorf
low==2.5.0) (3.7.4.3)
Requirement already satisfied: keras-preprocessing~=1.1.2 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tenso
rflow==2.5.0) (1.1.2)
Requirement already satisfied: absl-py~=0.10 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorflow==2.5.0)
(0.12.0)
Requirement already satisfied: grpcio~=1.34.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorflow==2.5.
0) (1.34.1)
Requirement already satisfied: numpy~=1.19.2 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorflow==2.5.0)
(1.19.5)
Requirement already satisfied: google-pasta~=0.2 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorflow==2.
5.0) (0.2.0)
Requirement already satisfied: wheel~=0.35 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorflow==2.5.0)
(0.37.0)
Requirement already satisfied: opt-einsum~=3.3.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorflow==2.
5.0) (3.3.0)
Requirement already satisfied: gast==0.4.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorflow==2.5.0)
(0.4.0)
Requirement already satisfied: tensorboard~=2.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorflow==2.
5.0) (2.7.0)
Requirement already satisfied: google-auth<3,>=1.6.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorboar
d~=2.5->tensorflow==2.5.0) (1.23.0)
Requirement already satisfied: markdown>=2.6.8 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorboard~=2.5
->tensorflow==2.5.0) (3.3.3)
Requirement already satisfied: werkzeug>=0.11.15 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorboard~=
2.5->tensorflow==2.5.0) (2.0.2)
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~=2.5->tensorflow==2.5.0) (0.6.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from te
nsorboard~=2.5->tensorflow==2.5.0) (1.6.0)
Requirement already satisfied: setuptools>=41.0.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorboard~=
2.5->tensorflow==2.5.0) (58.0.4)
Requirement already satisfied: requests<3,>=2.21.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from tensorboard~
```

Image Augmentation

In [9]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [10]:
```
train_datagen=ImageDataGenerator(rescale=1./255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
```

In [11]:
```
test_datagen=ImageDataGenerator(rescale=1./255)
```

In [12]:
```
ls
```

In [13]:
```
pwd
```

Out[13]: '/home/wsuser/work'

In [14]:
```
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_4ff9f1114db24196a9abd4f5c1f0b60a = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='j4lNXssktSSxQiDx3pbNR_eFi1SMCDE6MFnBQ_EmNCDM',
    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_1 = client_4ff9f1114db24196a9abd4f5c1f0b60a.get_object(Bucket='trainmodel-donotdelete-pr-cbqe37eh8gzesa', Key='fr

# 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 [15]:
```
from io import BytesIO
import zipfile
unzip = zipfile.ZipFile(BytesIO(streaming_body_1.read()), "r")
file_paths = unzip.namelist()
for path in file_paths:
    unzip.extract(path)
```

In [16]:
```
pwd
```

Out[16]: '/home/wsuser/work'

In [17]:
```
import os
filenames = os.listdir('/home/wsuser/work/fruit-dataset/train')
```

In [18]:
```
x_train=train_datagen.flow_from_directory("/home/wsuser/work/fruit-dataset/train",target_size=(128,128),class_mode='categorical'
```

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

In [ ]:
```

```

In [19]:
```
x_test=test_datagen.flow_from_directory(r"/home/wsuser/work/fruit-dataset/test",target_size=(128,128),
                                        class_mode='categorical',batch_size=24)
```

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

In [20]:
```
x_train.class_indices
```

Out[20]:
```
{'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

In [21]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Convolution2D,MaxPooling2D,Flatten
```

In [24]:
```
model=Sequential()
```

In [25]:
```
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
```

In [26]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [27]:
```
model.add(Flatten())
```

In [28]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (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 [29]:
```
32*(3*3*3+1)
```

Out[29]:
```
896
```

Hidden Layers

In [30]:
```
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
```

Output Layer

In [31]:
```
model.add(Dense(6,activation='softmax'))
```

In [32]:
```
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

In [33]:
```
len(x_train)
```

Out[33]:
```
225
```

In [34]:
```
1238/24
```

Out[34]:
```
51.583333333333336
```

In [35]:
```
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_164/1582812018.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future ver
sion. 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
225/225 [==============================] - 118s 520ms/step - loss: 0.8920 - accuracy: 0.8094 - val_loss: 0.2273 - val_accuracy:
0.9235
Epoch 2/10
225/225 [==============================] - 116s 515ms/step - loss: 0.2367 - accuracy: 0.9179 - val_loss: 0.2056 - val_accuracy:
0.9324
Epoch 3/10
225/225 [==============================] - 116s 517ms/step - loss: 0.1970 - accuracy: 0.9337 - val_loss: 0.4972 - val_accuracy:
0.8754
Epoch 4/10
225/225 [==============================] - 117s 521ms/step - loss: 0.1688 - accuracy: 0.9422 - val_loss: 0.2279 - val_accuracy:
0.9217
Epoch 5/10
225/225 [==============================] - 116s 516ms/step - loss: 0.1438 - accuracy: 0.9487 - val_loss: 0.1685 - val_accuracy:
0.9484
Epoch 6/10
225/225 [==============================] - 117s 518ms/step - loss: 0.1362 - accuracy: 0.9556 - val_loss: 0.1176 - val_accuracy:
0.9662
Epoch 7/10
225/225 [==============================] - 116s 515ms/step - loss: 0.1282 - accuracy: 0.9590 - val_loss: 0.5466 - val_accuracy:

Epoch 8/10
225/225 [==============================] - 116s 514ms/step - loss: 0.1282 - accuracy: 0.9597 - val_loss: 0.1194 - val_accuracy:
0.9620
Epoch 9/10
225/225 [==============================] - 116s 514ms/step - loss: 0.1141 - accuracy: 0.9616 - val_loss: 0.1478 - val_accuracy:
0.9508
Epoch 10/10
225/225 [==============================] - 116s 516ms/step - loss: 0.0927 - accuracy: 0.9695 - val_loss: 0.0772 - val_accuracy:
0.9751
```

Out[35]:
```
<keras.callbacks.History at 0x7f71e8184070>
```

Saving Model

In [36]:
```
ls
```

```
fruit-dataset/
```

In [37]:
```
model.save('fruit.h5')
```

In [40]:
```
!tar -zcvf Train-model_new.tgz fruit.h5
```

```
fruit.h5
```

In [39]:
```
ls -1
```

```
fruit-dataset/
fruit.h5
Train-model_new.tgz
```

IBM Cloud Deployment Model

In [41]:
```
!pip install watson-machine-learning-client --upgrade
```

```
Collecting watson-machine-learning-client
  Downloading watson_machine_learning_client-1.0.391-py3-none-any.whl (538 kB)
     |████████████████████████████████| 538 kB 21.2 MB/s eta 0:00:01
Requirement already satisfied: tqdm in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-cli
ent) (4.62.3)
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-
client) (2022.9.24)
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: 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-learn
ing-client) (2.11.0)
Requirement already satisfied: pandas in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-c
lient) (1.3.4)
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-c
lient) (0.3.3)
Requirement already satisfied: boto3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-cl
ient) (1.18.21)
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->wa
tson-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 boto
core<1.22.0,>=1.21.21->boto3->watson-machine-learning-client) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas->watson-mach
ine-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-mac
hine-learning-client) (1.19.5)
Installing collected packages: watson-machine-learning-client
Successfully installed watson-machine-learning-client-1.0.391
```

In [43]:
```
from ibm_watson_machine_learning import APIClient

wml_credentials = {
                    "url": "https://us-south.ml.cloud.ibm.com",
                   "apikey":"0P3XkyCFYqABnc48BNG2ReoGAJy-oDXDRuULl4Y_zFxa"
                  }

client = APIClient(wml_credentials)
```

In [44]:
```
client = APIClient(wml_credentials)
```

In [45]:
```
def guid_from_space_name(client, space_name):
    space = client.spaces.get_details()
    return(next(item for item in space['resources'] if item['entity']["name"]==space_name)['metadata']['id'])
```

In [46]:
```
space_uid = guid_from_space_name(client, 'Trainmodel')
print("Space UID = " + space_uid)
```

```
Space UID = 616c7d74-e99b-4c09-9922-27394a62c2d0
```

In [47]:
```
client.set.default_space(space_uid)
```

Out[47]:
```
'SUCCESS'
```

In [48]:
```
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
kernel-spark3.3-r3.6          1c9e5454-f216-59dd-a20e-474a5cdf5988  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
kernel-spark3.3-py3.9         2b7961e2-e3b1-5a8c-a491-482c8368839a  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
---------------------------  ---------------------------------  ----
Note: Only first 50 records were displayed. To display more use 'limit' parameter.
```

In [51]:
```
software_space_uid = client.software_specifications.get_uid_by_name("tensorflow_rt22.1-py3.9")
software_spec_uid
```

Out[51]: '1eb25b84-d6ed-5dde-b6a5-3fbdf1665666'

In [54]:
```
ls
```

```
fruit-dataset/  fruit.h5  Train-model_new.tgz
```

In [56]:
```
model_details = client.repository.store_model(model= 'Train-model_new.tgz',
    meta_props={
        client.repository.ModelMetaNames.NAME:"CNN",
        client.repository.ModelMetaNames.TYPE:"tensorflow_2.7",
        client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_space_uid}
    )
```

In [57]:
```
model_id = client.repository.get_model_id(model_details)
```

In [58]:
```
model_id
```

Out[58]: 'd0aeb6a2-e89c-4f8d-bf2f-a28ca4ea3cca'

In [60]:
```
ls
```

```
fruit-dataset/  fruit.h5  Train-model_new.tgz
```

Test The Model

In [54]:
```
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
```

In [55]:
```
model=load_model('fruit.h5')
```

In [68]:
```
img=image.load_img(r"C:\Users\Sree Ram\Desktop\ibm\Dataset Plant Disease\fruit-dataset\fruit-dataset\test\Apple___healthy\0adc1c5
```

In [69]:
```
img
```

Out[69]:

In [70]:
```
img=image.load_img(r"C:\Users\Sree Ram\Desktop\ibm\Dataset Plant Disease\fruit-dataset\fruit-dataset\test\Apple___healthy\0adc1c5
img
```

Out[70]:

In [71]:
```
x=image.img_to_array(img)
```

In [72]:
```
x
```

Out[72]:
```
array([[[ 99.,  86., 106.],
        [101.,  88., 108.],
        [118., 105., 125.],
        ...,
        [ 92.,  83., 102.],
        [ 93.,  84., 103.],
        [ 89.,  80.,  99.]],

       [[ 96.,  83., 103.],
        [ 87.,  74.,  94.],
        [102.,  89., 109.],
        ...,
        [ 88.,  79.,  98.],
        [ 89.,  80.,  99.],
        [ 83.,  74.,  93.]],
```

```
        ...,
        [ 88.,  79.,  98.],
        [ 89.,  80.,  99.],
        [ 83.,  74.,  93.]],

       [[ 86.,  73.,  93.],
        [ 88.,  75.,  95.],
        [ 98.,  85., 105.],
        ...,
        [107.,  98., 117.],
        [ 96.,  87., 106.],
        [ 96.,  87., 106.]],

       ...,

       [[172., 175., 194.],
        [173., 176., 195.],
        [175., 178., 197.],
        ...,
        [179., 180., 198.],
        [184., 185., 203.],
        [179., 180., 198.]],

       [[172., 175., 194.],
        [170., 173., 192.],
        [173., 176., 195.],
        ...,
        [178., 179., 197.],
        [182., 183., 201.],
        [178., 179., 197.]],

       [[169., 172., 191.],
        [166., 169., 188.],
        [168., 171., 190.],
        ...,
          ...,
          [187., 188., 206.],
          [185., 186., 204.],
          [186., 187., 205.]]], dtype=float32)
```

In [73]:
```
x=np.expand_dims(x,axis=0)
```

In [74]:
```
x
```

Out[74]:
```
array([[[[ 99.,  86., 106.],
         [101.,  88., 108.],
         [118., 105., 125.],
         ...,
         [ 92.,  83., 102.],
         [ 93.,  84., 103.],
         [ 89.,  80.,  99.]],

        [[ 96.,  83., 103.],
         [ 87.,  74.,  94.],
         [102.,  89., 109.],
         ...,
         [ 88.,  79.,  98.],
         [ 89.,  80.,  99.],
         [ 83.,  74.,  93.]],

        [[ 86.,  73.,  93.],
         [ 88.,  75.,  95.],
         [ 98.,  85., 105.],
         ...,
         [107.,  98., 117.],
         [ 96.,  87., 106.],
         [ 96.,  87., 106.]],

        ...,
```

```
...,
[187., 188., 206.],
[185., 186., 204.],
[186., 187., 205.]]], dtype=float32)
```

In [75]:
```
y=np.argmax(model.predict(x),axis=1)
```

```
1/1 [==============================] - 0s 105ms/step
```

In [76]:
```
x_train.class_indices
```

Out[76]:
```
{'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 [77]:
```
index=['Apple___Black_rot','Apple___healthy','Corn_(maize)___Northern_Leaf_Blight','Corn_(maize)___healthy','Peach___Bacterial_sp
```

In [78]:
```
index[y[0]]
```

Out[78]:
```
'Apple___healthy'
```

In [82]:
```
img=image.load_img(r"C:\Users\Sree Ram\Desktop\ibm\Dataset Plant Disease\fruit-dataset\fruit-dataset\test\Peach___healthy\0a2ed4
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','Peach___Bacterial_sp
index[y[0]]
```

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

Out[82]:
```
'Corn_(maize)___healthy'
```

In [83]:
```
import os
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from flask import Flask,render_template,request
```

In [61]:
```
app=Flask(__name__)

model=load_model("fruit.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___Bact
        text="The Classified Fruit disease is : " +str(index[pred[0]])
    return text
if __name__=='__main__':
    app.run(debug=False)
```

REFERENCE:

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