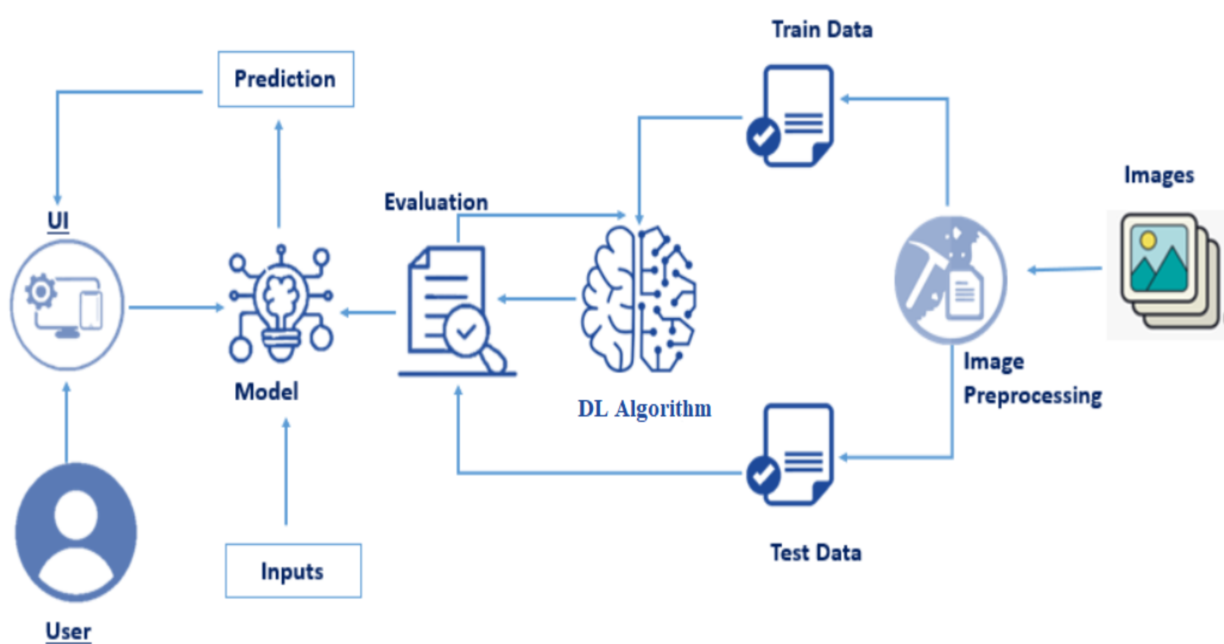
Introduction

In today's society, agriculture is the most significant industry. An extensive range of bacterial and fungal diseases harm the majority of plants. Plant diseases severely limited productivity and posed a serious threat to food security. To achieve maximum quantity and optimum quality, early and accurate identification of plant diseases is crucial. The variety of pathogen strains, adjustments to production practises, and insufficient plant protection systems have all contributed to an increase in the number of plant diseases in recent years, as well as the severity of the damage they inflict.
An automated technique is now available to recognise many plant diseases by examining the symptoms seen on the plant's leaves. In order to identify diseases and provide preventative measures, deep learning algorithms are applied.

Architecture



Smart Internz

Project Objectives

At the conclusion of this task, you will comprehend:

- Image Preprocessing
- the dataset is subjected to the CNN algorithm.
- how deep learning systems identify the illness.
- You will be able to determine how to assess the model's correctness.
- The Flask framework will let you construct web applications.

Software & Packages Used

For applications involving data science and machine learning, Anaconda Navigator is a free and open-source distribution of the Python and R programming languages. It can be set up on Linux, macOS, and Windows. A cross-platform, open-source package management system is called Conda. JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, and Visual Studio Code are just a few of the excellent tools that come with Anaconda. We are utilising a Jupiter laptop and Spyder for this project.

We need the following packages in order to develop deep learning models:

TensorFlow: TensorFlow is an open-source, end-to-end machine learning platform. Researchers can advance the state-of-the-art in machine learning thanks to its extensive, adaptable ecosystem of tools, libraries, and community resources, while developers can simply create and deploy ML-powered applications.

Keras: To make high level neural network API simpler and more effective, Keras makes use of a variety of optimization approaches. The following functionalities are supported by it:

- an extensible, straightforward, and consistent API.
- Simple structure makes it simple to get the desired effect.
- It supports a variety of backends and platforms.
- It is an easy-to-use framework that utilises both the CPU and GPU.
- extremely scalable computation.

Methodology

- Farmers can communicate with the portal using a web application that is established.
- utilises the user interface to upload pictures of a sick leaf
- Our developed model analyses the Illness and recommends using fertilisers by the farmer.

We have to finish the following activities and tasks in order to fulfil the aforementioned task.

- **Download the dataset using following link:**

https://drive.google.com/file/d/1fxs7ptI6zh7NTbCOZARKZ7AmYKjnprrY/view?usp=sharing

- Image Preprocessing: Image data augmentation is a method for artificially increasing the size of a training dataset by producing altered versions of the dataset's photographs.Through the ImageDataGenerator class, the Keras deep learning neural network library offers the ability to fit models with the addition of image data. Importing the libraries that the programme will require typically comes first. Import the ImageDataGenerator Library from that library, then import the Keras library.

✓ **Import ImageDataGenerator Library and Configure it**

✓ **Apply ImageDataGenerator functionality to Train and Test set**

- **Model Building For Fruit Disease Prediction:** Now that we have the enhanced and pre-processed image data, let's start creating our model. This activity entails the following phases.

✓ Import the model building Libraries

✓ Initializing the model

✓ Adding CNN Layers

✓ Adding Hidden Layer

✓ Adding Output Layer

✓ Configure the Learning Process

✓ Training and testing the model

- ✓ Saving the model

- **Model Building For Vegetable Disease Prediction:**
  - ✓ Import ImageDataGenerator Library and Configure it
  - ✓ Apply ImageDataGenerator functionality to Train and Test set
  - ✓ Import the required model building libraries
  - ✓ initialize the model
  - ✓ Add the convolution layer
  - ✓ Add the pooling layer
  - ✓ Add the flatten layer
  - ✓ Adding the dense layers
  - ✓ Compile the model
  - ✓ Fit and save the model
  - ✓ Calculate accuracy and loss.

- **Test the model:**
  - ✓ Import the packages and load the saved model
  - ✓ Load the test image, pre-process it and predict

- **Application Building:**

We will incorporate the model into a web application once it has been developed so that common users can use it as well. The first step for new users is to register in the portal. Users can log in after registering to peruse the photographs and look for sickness.

We now need to construct:

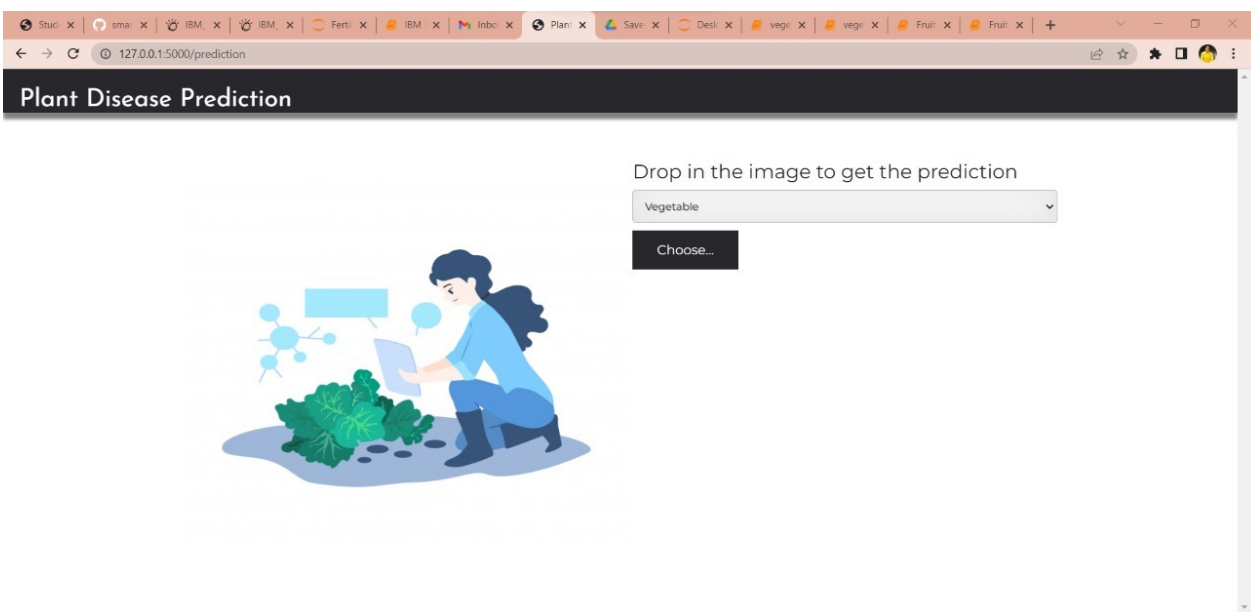
- ✓ HTML front end pages
- ✓ Server-side script written in Python

- **Train the model on IBM.**

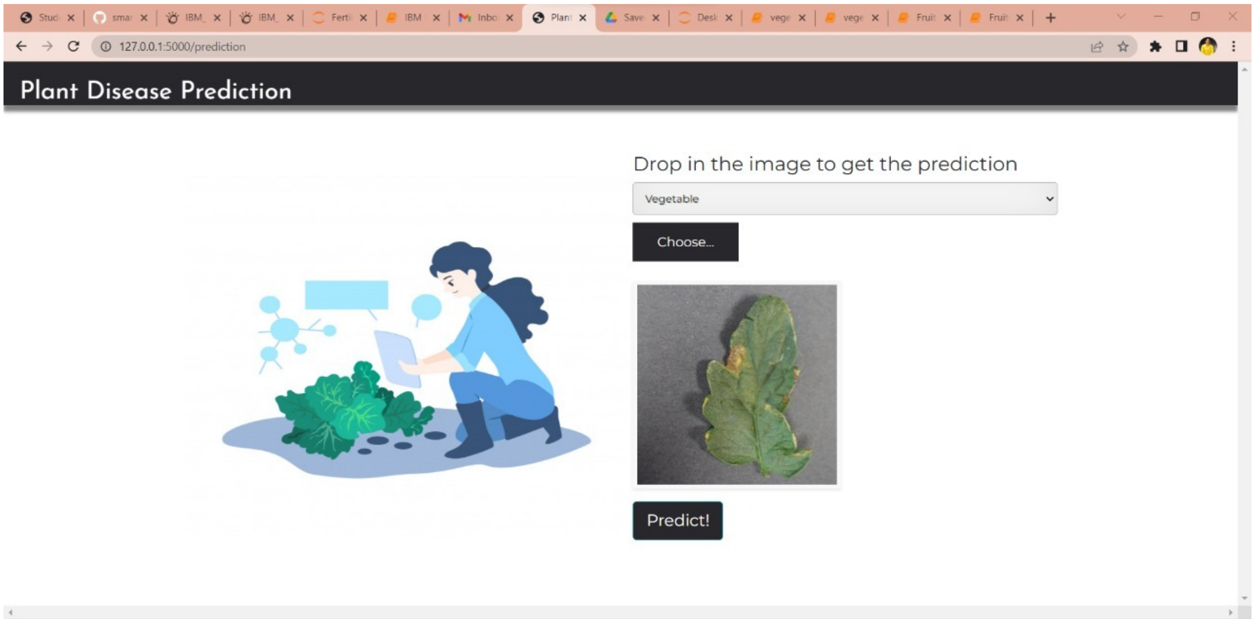
Results

Web applications using the Flask framework (home.html)

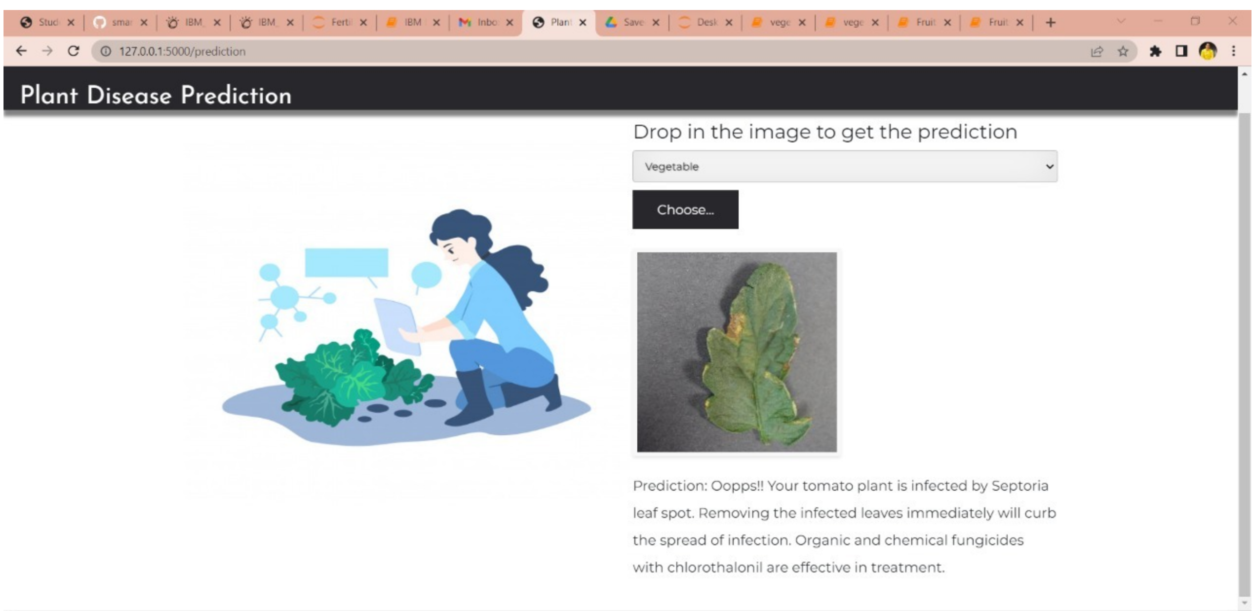
Web applications using the Flask framework (predict.html)



Web applications using the Flask framework (predict.html)



Web applications using the Flask framework (predict.html)



Conclusion

The most significant industry is agriculture, and a vast range of bacterial and fungal diseases harm most plants. Plant diseases severely limited productivity and posed a serious threat to food security. To achieve maximum quantity and optimum quality, early and accurate identification of plant diseases is crucial.

Smart Internz

Appendix

Model Building For Vegetable Disease Prediction (vegetable_training.ipynb)

jupyter vegetable_training Last Checkpoint: Last Sunday at 6:45 PM (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run | Code

In [1]:
```
from keras.preprocessing.image import ImageDataGenerator
```

In [2]:
```
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
```

In [3]:
```
test_datagen=ImageDataGenerator(rescale=1./255)
```

In [4]:
```
x_train=train_datagen.flow_from_directory('C:/Users/Saurabh Sharma/Desktop/Project Build-A-Thon/Project/Dataset Plant Disease/Veg
class_mode='categorical',batch_size=16)
```

```
Found 11386 images belonging to 9 classes.
```

In [5]:
```
x_test=test_datagen.flow_from_directory('C:/Users/Saurabh Sharma/Desktop/Project Build-A-Thon/Project/Dataset Plant Disease//Veg-
class_mode='categorical',batch_size=16)
```

```
Found 3416 images belonging to 9 classes.
```

In [6]:
```
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 [7]:
```
model=Sequential()
```

In [8]:
```
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
```

In [9]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [10]:
```
model.add(Flatten())
```

In [11]:
```
model.add(Dense(units=300,activation='relu'))
model.add(Dense(units=150,activation='relu'))
model.add(Dense(units=75,activation='relu'))
model.add(Dense(units=9,activation='softmax'))
```

In [12]:
```
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

In [13]:
```
model.fit_generator(x_train,steps_per_epoch=89,epochs=20,validation_data=x_test,validation_steps=27)
```

```
C:\Users\SAURAB~1\AppData\Local\Temp/ipykernel_10356/174847055.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=89,epochs=20,validation_data=x_test,validation_steps=27)
```

```
Epoch 1/20
89/89 [==============================] - 31s 345ms/step - loss: 2.8369 - accuracy: 0.2746 - val_loss: 1.5583 - val_accuracy: 0.
4838
Epoch 2/20
89/89 [==============================] - 31s 345ms/step - loss: 1.3561 - accuracy: 0.5323 - val_loss: 1.3340 - val_accuracy: 0.
5671
Epoch 3/20
89/89 [==============================] - 29s 329ms/step - loss: 1.1275 - accuracy: 0.6074 - val_loss: 1.2351 - val_accuracy: 0.
5694
Epoch 4/20
89/89 [==============================] - 29s 324ms/step - loss: 0.9758 - accuracy: 0.6440 - val_loss: 0.9521 - val_accuracy: 0.
6551
Epoch 5/20
89/89 [==============================] - 28s 318ms/step - loss: 0.9002 - accuracy: 0.6861 - val_loss: 0.7767 - val_accuracy: 0.
7245
Epoch 6/20
89/89 [==============================] - 30s 340ms/step - loss: 0.8297 - accuracy: 0.7184 - val_loss: 0.8054 - val_accuracy: 0.
7060
Epoch 7/20
89/89 [==============================] - 31s 343ms/step - loss: 0.7284 - accuracy: 0.7374 - val_loss: 0.7404 - val_accuracy: 0.
6991
Epoch 8/20
89/89 [==============================] - 29s 324ms/step - loss: 0.7045 - accuracy: 0.7409 - val_loss: 1.1024 - val_accuracy: 0.
6134
Epoch 9/20
89/89 [==============================] - 31s 345ms/step - loss: 0.6471 - accuracy: 0.7676 - val_loss: 0.6744 - val_accuracy: 0.
7569
Epoch 10/20
89/89 [==============================] - 29s 328ms/step - loss: 0.6768 - accuracy: 0.7683 - val_loss: 0.6454 - val_accuracy: 0.
7708
Epoch 11/20
89/89 [==============================] - 28s 315ms/step - loss: 0.6648 - accuracy: 0.7556 - val_loss: 0.6540 - val_accuracy: 0.
7662
Epoch 12/20
89/89 [==============================] - 30s 340ms/step - loss: 0.6026 - accuracy: 0.7858 - val_loss: 0.6508 - val_accuracy: 0.
7616
Epoch 13/20
89/89 [==============================] - 28s 317ms/step - loss: 0.5792 - accuracy: 0.7795 - val_loss: 0.6020 - val_accuracy: 0.
7917
Epoch 14/20
89/89 [==============================] - 29s 324ms/step - loss: 0.5801 - accuracy: 0.7999 - val_loss: 0.3342 - val_accuracy: 0.
8889
Epoch 15/20
89/89 [==============================] - 29s 324ms/step - loss: 0.5816 - accuracy: 0.7928 - val_loss: 0.5745 - val_accuracy: 0.
7940
Epoch 16/20
89/89 [==============================] - 30s 332ms/step - loss: 0.4989 - accuracy: 0.8216 - val_loss: 0.6760 - val_accuracy: 0.
7731
Epoch 17/20
89/89 [==============================] - 29s 323ms/step - loss: 0.5500 - accuracy: 0.8097 - val_loss: 0.4692 - val_accuracy: 0.
8102
Epoch 18/20
89/89 [==============================] - 28s 319ms/step - loss: 0.5084 - accuracy: 0.8174 - val_loss: 0.3491 - val_accuracy: 0.
8796
```

```
Epoch 19/20
89/89 [==============================] - 29s 323ms/step - loss: 0.5063 - accuracy: 0.8287 - val_loss: 0.4064 - val_accuracy: 0.
8634
Epoch 20/20
89/89 [==============================] - 30s 334ms/step - loss: 0.4626 - accuracy: 0.8392 - val_loss: 0.4001 - val_accuracy: 0.
8588
```

Out[13]:
```
<keras.callbacks.History at 0x24821141b50>
```

In [14]:
```python
model.save('vegetable.h5')
```

In [15]:
```python
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

 dense (Dense)               (None, 300)               38102700

 dense_1 (Dense)             (None, 150)               45150

 dense_2 (Dense)             (None, 75)                11325

 dense_3 (Dense)             (None, 9)                 684

=================================================================
Total params: 38,160,755
Trainable params: 38,160,755
```

Model Testing For Vegetable Disease Prediction (vegetable_Tesing.ipynb)

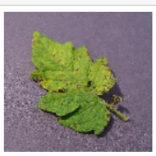
In [1]:
```python
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
import numpy as np
```

In [2]:
```python
model=load_model('vegetable.h5')
```

In [3]:
```python
img=image.load_img('C:/Users/Saurabh Sharma/Desktop/Project Build-A-Thon/Project/Dataset Plant Disease/Veg-dataset/Veg-dataset/te
```

In [4]:
```python
img
```

Out[4]:



In [5]:
```python
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
```

In [6]:
```python
pred=np.argmax(model.predict(x),axis=1)
```

```
1/1 [==============================] - 0s 453ms/step
```

```
In [7]: from keras.preprocessing.image import ImageDataGenerator
        train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
        test_datagen=ImageDataGenerator(rescale=1./255)
        x_train=train_datagen.flow_from_directory('C:/Users/Saurabh Sharma/Desktop/Project Build-A-Thon/Project/Dataset Plant Disease/Veg
        class_mode='categorical',batch_size=16)
        x_test=test_datagen.flow_from_directory('C:/Users/Saurabh Sharma/Desktop/Project Build-A-Thon/Project/Dataset Plant Disease//Veg-
        class_mode='categorical',batch_size=16)

        Found 11386 images belonging to 9 classes.
        Found 3416 images belonging to 9 classes.

In [8]: x_train.class_indices

Out[8]: {'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}

In [9]: index=['Pepper,_bell___Bacterial_spot','Pepper,_bell___healthy','Potato___Early_blight','Potato___Late_blight','Potato___healthy'
              'Tomato___Bacterial_spot','Tomato___Late_blight','Tomato___Leaf_Mold','Tomato___Septoria_leaf_spot']

In [10]: index[pred[0]]

Out[10]: 'Tomato___Septoria_leaf_spot'
```

Model Building For Fruits Disease Prediction (Fruit-Training.ipynb)

```
In [1]: from keras.preprocessing.image import ImageDataGenerator

In [2]: train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)

In [3]: test_datagen=ImageDataGenerator(rescale=1./255)

In [4]: %pwd

Out[4]: 'C:\\Users\\Saurabh Sharma\\Desktop\\Project Build-A-Thon\\Project'

In [5]: x_train=train_datagen.flow_from_directory('C:/Users/Saurabh Sharma/Desktop/Project Build-A-Thon/Project/Dataset Plant Disease/fru
        class_mode='categorical',batch_size=32)

        Found 5384 images belonging to 6 classes.

In [6]: x_test=test_datagen.flow_from_directory('C:/Users/Saurabh Sharma/Desktop/Project Build-A-Thon/Project/Dataset Plant Disease/fruit
        class_mode='categorical',batch_size=32)

        Found 1686 images belonging to 6 classes.

In [7]: 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 [8]: model=Sequential()

In [9]: model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
```

In [10]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [11]:
```
model.add(Flatten())
```

In [18]:
```
model.add(Dense(40,activation='relu'))
model.add(Dense(20, activation='relu'))
model.add(Dense(6,activation='softmax'))
```

In [13]:
```
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

In [14]:
```
model.fit_generator(x_train,steps_per_epoch=168,epochs=3,validation_data=x_test,validation_steps=52)
```

```
C:\Users\SAURAB~1\AppData\Local\Temp/ipykernel_10636/1229104227.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=168,epochs=3,validation_data=x_test,validation_steps=52)
```

```
Epoch 1/3
168/168 [==============================] - 99s 578ms/step - loss: 0.9372 - accuracy: 0.7083 - val_loss: 0.3136 - val_accuracy:
0.8882
Epoch 2/3
168/168 [==============================] - 94s 556ms/step - loss: 0.3465 - accuracy: 0.8815 - val_loss: 0.4098 - val_accuracy:
0.8732
Epoch 3/3
168/168 [==============================] - 93s 550ms/step - loss: 0.2828 - accuracy: 0.9073 - val_loss: 0.1605 - val_accuracy:
0.9549
```

Out[14]:
```
<keras.callbacks.History at 0x20064d0de20>
```

In [15]:
```
model.save('fruit.h5')
```

In [16]:
```
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

 dense (Dense)               (None, 40)                5080360

 dense_1 (Dense)             (None, 20)                820

 dense_2 (Dense)             (None, 6)                 126

=================================================================
Total params: 5,082,202
Trainable params: 5,082,202
Non-trainable params: 0
_________________________________________________________________
```

Model Testing For Fruits Disease Prediction (Fruits-Tesing.ipynb)

In [1]:
```python
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
import numpy as np
```

In [2]:
```python
model=load_model('fruit.h5')
```

In [4]:
```python
img=image.load_img('C:/Users/Saurabh Sharma/Desktop/Project Build-A-Thon/Project/Dataset Plant Disease/fruit-dataset/fruit-datase
```

In [5]:
```python
img
```

Out[5]:

In [6]:
```python
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
```

In [13]:
```python
pred=np.argmax(model.predict(x),axis=1)
```

```
1/1 [==============================] - 0s 41ms/step
```

In [18]:
```python
from keras.preprocessing.image import ImageDataGenerator
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
x_train=train_datagen.flow_from_directory('C:/Users/Saurabh Sharma/Desktop/Project Build-A-Thon/Project/Dataset Plant Disease/fru
class_mode='categorical',batch_size=32)
x_test=test_datagen.flow_from_directory('C:/Users/Saurabh Sharma/Desktop/Project Build-A-Thon/Project/Dataset Plant Disease/fruit
class_mode='categorical',batch_size=32)
```

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

In [19]:
```python
x_train.class_indices
```

Out[19]:
```
{'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 [20]:
```python
index=['Apple___Black_rot','Apple___healthy','Corn_(maize)___Northern_Leaf_Blight','Corn_(maize)___healthy','Peach___Bacterial_sp
```

In [21]:
```python
index[pred[0]]
```

Out[21]:
```
'Apple___healthy'
```

Smart Internz

IBM Deployment Model Download (IBM Deployment Model Download.ipynb)

In [1]:
```
pip install ibm_watson_machine_learning
```

```
Requirement already satisfied: ibm_watson_machine_learning in c:\users\saurabh sharma\anaconda3\lib\site-packages (1.0.229)Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: pandas<1.4.0,>=0.24.2 in c:\users\saurabh sharma\anaconda3\lib\site-packages (from ibm_watson_machine_learning) (1.3.4)
Requirement already satisfied: certifi in c:\users\saurabh sharma\anaconda3\lib\site-packages (from ibm_watson_machine_learning) (2021.10.8)
Requirement already satisfied: requests in c:\users\saurabh sharma\anaconda3\lib\site-packages (from ibm_watson_machine_learning) (2.26.0)

Requirement already satisfied: packaging in c:\users\saurabh sharma\anaconda3\lib\site-packages (from ibm_watson_machine_learning) (21.0)
Requirement already satisfied: lomond in c:\users\saurabh sharma\anaconda3\lib\site-packages (from ibm_watson_machine_learning) (0.3.3)
Requirement already satisfied: urllib3 in c:\users\saurabh sharma\anaconda3\lib\site-packages (from ibm_watson_machine_learning) (1.26.7)
Requirement already satisfied: tabulate in c:\users\saurabh sharma\anaconda3\lib\site-packages (from ibm_watson_machine_learning) (0.8.10)
Requirement already satisfied: ibm-cos-sdk==2.11.* in c:\users\saurabh sharma\anaconda3\lib\site-packages (from ibm_watson_machine_learning) (2.11.0)
Requirement already satisfied: importlib-metadata in c:\users\saurabh sharma\anaconda3\lib\site-packages (from ibm_watson_machine_learning) (4.8.1)
Requirement already satisfied: ibm-cos-sdk-core==2.11.0 in c:\users\saurabh sharma\anaconda3\lib\site-packages (from ibm-cos-sdk==2.11.*->ibm_watson_machine_learning) (2.11.0)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in c:\users\saurabh sharma\anaconda3\lib\site-packages (from ibm-cos-sdk==2.11.*->ibm_watson_machine_learning) (0.10.0)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in c:\users\saurabh sharma\anaconda3\lib\site-packages (from ibm-cos-sdk==2.11.*->ibm_watson_machine_learning) (2.11.0)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in c:\users\saurabh sharma\anaconda3\lib\site-packages (from ibm-cos-sdk-core==2.11.0->ibm-cos-sdk==2.11.*->ibm_watson_machine_learning) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in c:\users\saurabh sharma\anaconda3\lib\site-packages (from pandas<1.4.0,>=0.24.2->ibm_watson_machine_learning) (2021.3)
Requirement already satisfied: numpy>=1.17.3 in c:\users\saurabh sharma\anaconda3\lib\site-packages (from pandas<1.4.0,>=0.24.2->ibm_watson_machine_learning) (1.20.3)
Requirement already satisfied: six>=1.5 in c:\users\saurabh sharma\anaconda3\lib\site-packages (from python-dateutil<3.0.0,>=2.1->ibm-cos-sdk-core==2.11.0->ibm-cos-sdk==2.11.*->ibm_watson_machine_learning) (1.16.0)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\saurabh sharma\anaconda3\lib\site-packages (from requests->ibm_watson_machine_learning) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\saurabh sharma\anaconda3\lib\site-packages (from requests->ibm_watson_machine_learning) (3.2)
Requirement already satisfied: zipp>=0.5 in c:\users\saurabh sharma\anaconda3\lib\site-packages (from importlib-metadata->ibm_watson_machine_learning) (3.6.0)
Requirement already satisfied: pyparsing>=2.0.2 in c:\users\saurabh sharma\anaconda3\lib\site-packages (from packaging->ibm_watson_machine_learning) (3.0.4)
```

In [2]:
```
from ibm_watson_machine_learning import APIClient
wml_credentials={
    "url":"https://us-south.ml.cloud.ibm.com",
    "apikey":"6gGrTqhp16JtoQo_kdxCqPnd23T2rzpPtIP49BmEAR6h"
}
```

In [3]:
```
client=APIClient(wml_credentials)
```

In [4]:
```
client
```

Out[4]:
```
<ibm_watson_machine_learning.client.APIClient at 0x200e7838460>
```

In [6]:
```
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 [7]:
```
space_uid=guid_from_space_name(client,'Plants Disease')
print(space_uid)
```

```
49c73313-300d-4017-8df3-1cbcec09cc26
```

In [8]:
```
client.set.default_space(space_uid)
```

Out[8]:
```
'SUCCESS'
```

In [10]:
```
client.repository.download("de34c718-bf6a-40dd-ae19-4e7b5670dc5e","plant-classification.tar.gz")
```

```
Successfully saved model content to file: 'plant-classification.tar.gz'
```

Out[10]:
```
'C:\\Users\\Saurabh Sharma\\Desktop\\Project Build-A-Thon\\Fertilizers Recommendation System For Disease Prediction/plant-classification.tar.gz'
```

IBM Flask Application Deployment (application.py on Spyder)

```
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__)

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

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

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

@app.route('/predict',methods=['GET','POST'])
def predict():
    if request.method=='POST':
        # Get the file name from the post request
        f=request.files['image']
        # Save the file to ./uploads
        basepath=os.path.dirname(__file__)
```

```
        filepath=os.path.join(basepath,'uploads',secure_filename(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)
        plant=request.form['plant']
        print(plant)
        if(plant=="vegetable"):
            preds=model1.predict(x)
            print(preds)
            preds=np.argmax(model1.predict(x),axis=1)

index=['Pepper,_bell___Bacterial_spot','Pepper,_bell___healthy','Potato___Early_blight','Pot
ato___Late_blight','Potato___healthy',

'Tomato___Bacterial_spot','Tomato___Late_blight','Tomato___Leaf_Mold','Tomato___Sept
oria_leaf_spot']
            index[preds[0]]
            df=pd.read_excel('precautions - veg.xlsx')
            print(df.iloc[preds[0]]['caution'])
        else:
            preds=model.predict(x)
            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)
```