

AICTE BUILD A THON 2022

20.07.2022 – 26.07.2022

Project Report

Under

The title

Fertilizers Recommendation System for Disease Prediction

Submitted by

Dr. R GUNABALAN

Associate Professor

School of Electrical Engineering

Vellore Institute of Technology – Chennai



AUGUST 2022

1. Introduction

Agriculture sector plays a major role in today's life. Plants are affected by a wide variety of bacterial and fungal diseases. It limits the production and a major threat to food security. Hence, early and accurate identification of plant diseases is essential to ensure high quantity food products. In recent years, chemical fertilizers are used and day by day, the number of diseases and the severity has increased due to the variation in pathogen varieties, changes in cultivation methods and inadequate plant protection techniques. An automated system is necessary to identify different diseases on plants by checking the symptoms shown on the leaves of the plant. Deep learning techniques are preferred to identify the diseases and suggest the precautions that can be taken for those diseases.

2. Objectives

The main objectives of the project work are to:

- Pre-process the images
- Applying CNN algorithm to the dataset to detect the disease and find the accuracy of the model
- Build web applications using the Flask framework

3. Block diagram

The block diagram of the project work is shown in Figure 1. The dataset is collected to detect plant disease using convolution neural network (CNN). The dataset is divided into train data and test data. The network is trained and tested from the collected images and the disease in the plant is detected. The accuracy is based on the trained dataset.

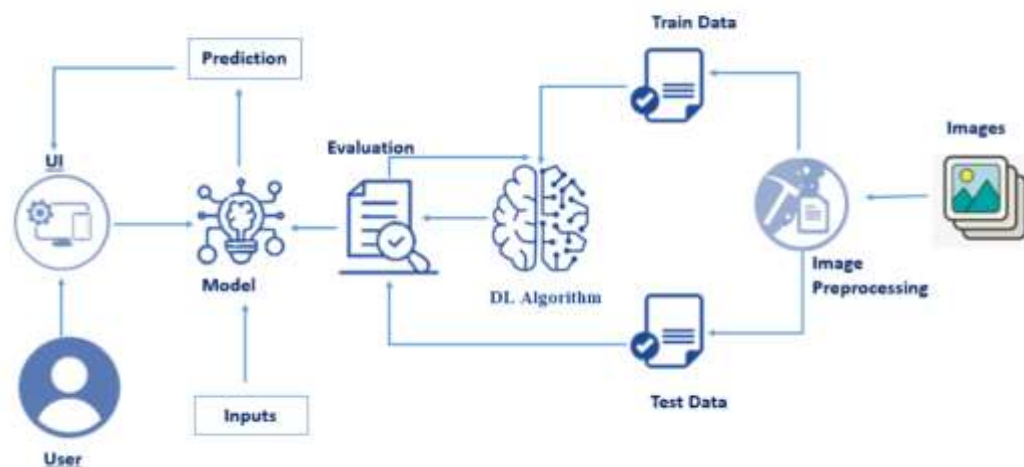


Figure 1 block diagram for disease detection using CNN

4. Methodology

The given dataset is downloaded. The dataset folder contains two folders for the fruit and vegetable dataset which again contains a test and train folder, each of them have images of different diseases. The coding is written in jupyter notebook using python. Different neural network layers are added. The images are trained and tested for plants and fruits. The accuracy of the model is noted.

4.1 Necessary steps:

- a. Dataset is downloaded
- b. Dataset is classified into train and test sets
- c. Neural network layers are added
- d. The images are loaded for training the model
- e. The model is tested using the dataset
- f. The model and its dependencies are saved
- g. The .h5 file is downloaded
- h. The accuracy of the model is noted.
- i. Web application using a flask is built and integrated with the model built.

4.2 Requirements

To build the deep learning models, the following packages are required:

Tensorflow: It is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers can easily build and deploy ML powered applications.

Keras : Keras leverages various optimization techniques to make high level neural network API easier. The main features are:

- Consistent
- Simple and extensible API
- Minimal structure
- Supports multiple platforms and backends
- User-friendly framework that runs on both CPU and GPU

- Highly scalability of computation.

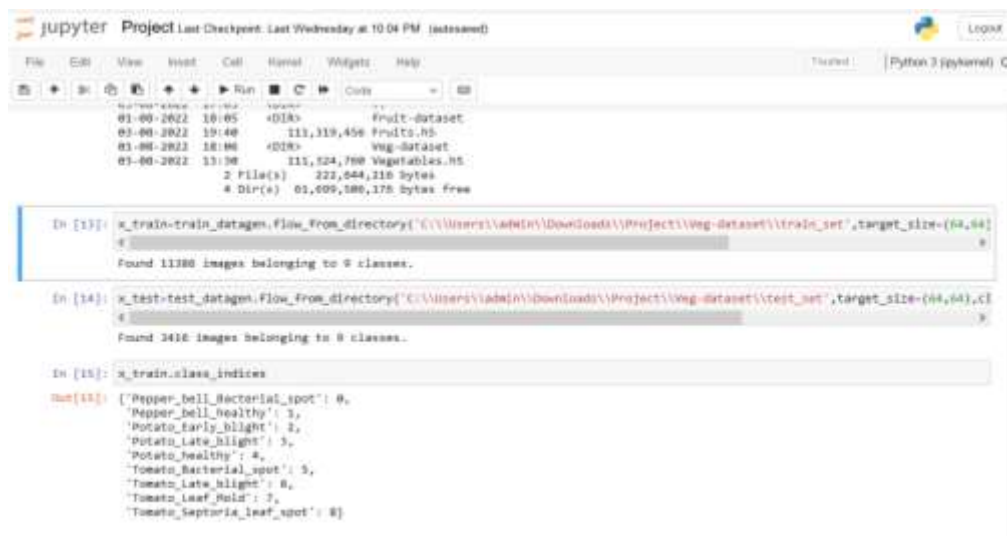
4.3 Algorithm steps

- Image Pre-processing
- Import the libraries
- Initializing the model
- ADD CNN layers
- Train and save the model
- Test both models

5. Results and Discussion

5.1 Plant disease detection

The model is built using python coding, written in jupyter notebook. Figure 2 shows the screenshot of the coding written for plant disease detection. A part of the coding is presented in Figure 2. 'relu' activation function is used for training the dataset. The length of the train dataset is 356 for each epoch. Single epoch is used for training the data. The accuracy for 5 epochs is 89.55%. The accuracy can be increase further by increasing epochs. The screenshot of the trained dataset accuracy is presented in Figure 3. After building the model, it is trained and tested using the dataset. The test results screenshot is provided in Figure 4. The image size of each dataset is reframed into 64*64. The original and the reframed images of the test dataset are presented in Figure 5.



```
jupyter Project Last Checkpoint: Last Wednesday at 10:04 PM (autosaved)
Python 3 (ipykernel)

File Edit View Insert Cell Format Help Trusted Python 3 (ipykernel)

01-00-2022 10:05 <DIR> Fruit-dataset
03-00-2022 10:40 111,319,456 Fruits.H5
03-00-2022 10:00 <DIR> Veg-dataset
03-00-2022 13:30 111,824,780 Vegetables.H5
2 File(s) 222,044,236 Bytes
4 Dir(s) 61,609,580,178 Bytes free

In [13]: x_train=train_datagen.flow_from_directory('C:\\Users\\ADMIN\\Downloads\\Project\\Veg-dataset\\train_set',target_size=(64,64),
Found 11380 images belonging to 8 classes.

In [14]: x_test=test_datagen.flow_from_directory('C:\\Users\\ADMIN\\Downloads\\Project\\Veg-dataset\\test_set',target_size=(64,64),cl
Found 3416 images belonging to 8 classes.

In [15]: x_train.class_indices
Out[15]: {'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_Rolt': 7,
'Tomato_Septoria_leaf_spot': 8}
```

Figure 2 Python coding for plant disease detection in jupyter notebook

```

In [19]: model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

In [20]: len(x_train)
Out[20]: 356

In [21]: len(x_test)
Out[21]: 107

In [22]: model.fit(x_train, steps_per_epoch=len(x_train), validation_data=x_test, validation_steps=len(x_test), epochs=1)
356/356 [=====] - 929s 3s/step - loss: 1.1485 - accuracy: 0.5388 - val_loss: 0.6079 - val_accuracy: 0.7857

Out[22]: <keras.callbacks.History at 0x2258091a3b>

In [28]: ls
Volume in drive C has no label,
Volume Serial Number is B060-38E3

Directory of C:\Users\Admin\Downloads\Project

01-08-2022  20:38    <DIR>          .
01-08-2022  20:30    <DIR>          ..
01-08-2022  18:05    <DIR>          fruit-dataset
01-08-2022  18:00    <DIR>          Vow-dataset

```

(a) Single epoch

```

In [22]: len(x_train)
Out[22]: 356

In [23]: len(x_test)
Out[23]: 107

In [24]: model.fit(x_train, steps_per_epoch=len(x_train), validation_data=x_test, validation_steps=len(x_test), epochs=5)

Epoch 1/5
356/356 [=====] - 744s 1s/step - loss: 1.0807 - accuracy: 0.6195 - val_loss: 0.7210 - val_accuracy: 0.7387
y: 0.7387
Epoch 2/5
356/356 [=====] - 307s 2s/step - loss: 0.5580 - accuracy: 0.6888 - val_loss: 0.4720 - val_accuracy: 0.8387
y: 0.8387
Epoch 3/5
356/356 [=====] - 550s 2s/step - loss: 0.4823 - accuracy: 0.8368 - val_loss: 0.5110 - val_accuracy: 0.8249
y: 0.8249
Epoch 4/5
356/356 [=====] - 573s 1s/step - loss: 0.3760 - accuracy: 0.8601 - val_loss: 0.2675 - val_accuracy: 0.9139
y: 0.9139
Epoch 5/5
356/356 [=====] - 727s 2s/step - loss: 0.2800 - accuracy: 0.8981 - val_loss: 0.2082 - val_accuracy: 0.8955
y: 0.8955

Out[24]: <keras.callbacks.History at 0x1a6bb2a3a3b>

```

(b) Five epochs

Figure 3 Accuracy of tested dataset by CNN

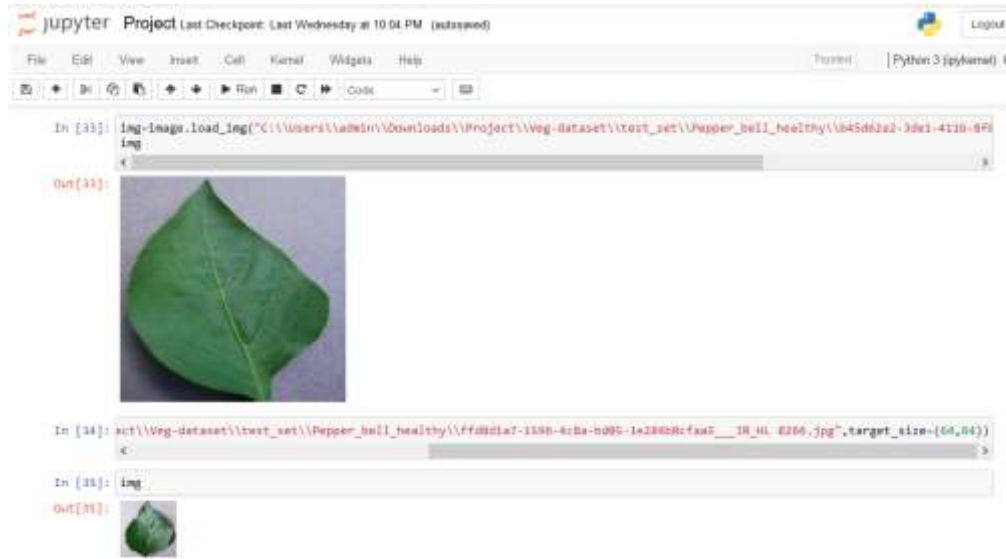


Figure 4 Original and reframed image in the model

The images are converted into arrays. The maximum value is 255 and minimum value depends on the image colour combination (RGB). The array structure is shown in Figure 5. Once trained, the network is tested with various images. The screenshot of the test results are shown in Figure 6. The CNN network identifies the disease correctly and the accuracy is good.

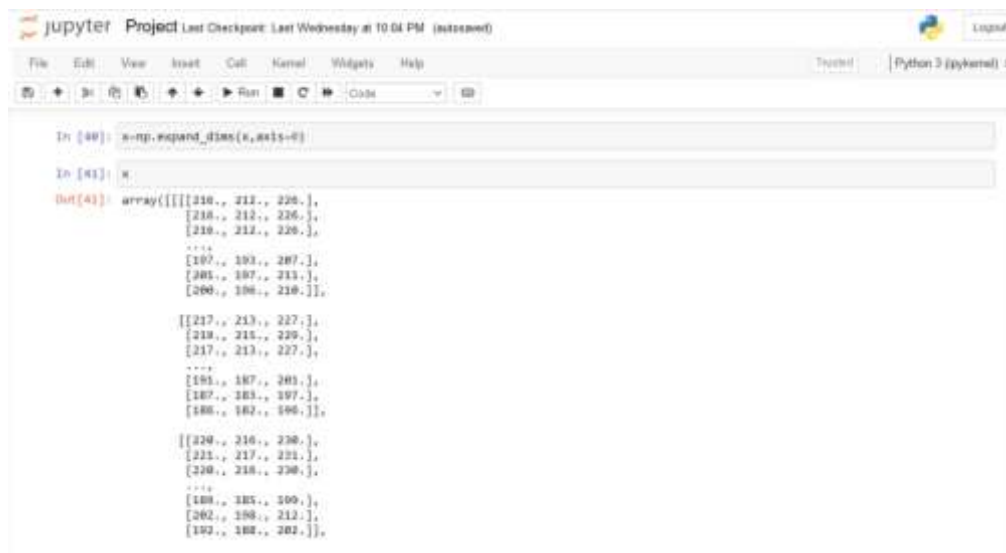


Figure 5 Image array structure built in python coding

```

index=['Pepper_bell_Bacterial_spot', 'Pepper_bell_healthy', 'Potato_Early_blight', 'Potato_Late_blight', 'Potato_healthy', 'Tomato_healthy']
index[y[0]]
1/1 [=====] - 8s 31ms/step

Out[48]: 'Potato_Early_blight'

In [49]: img=image.load_img('C:\\Users\\admin\\Downloads\\Project\\Veg-dataset\\test_set\\Tomato_Late_blight\\HS_Late_6-6624.jpg'),target_size=(64,64)
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_blight', 'Potato_Late_blight', 'Potato_healthy', 'Tomato_healthy']
index[y[0]]
1/1 [=====] - 8s 31ms/step

Out[49]: 'Potato_Early_blight'

In [50]: img=image.load_img('C:\\Users\\admin\\Downloads\\Project\\Veg-dataset\\test_set\\Tomato_Sectoria_leaf_spot\\Keller-St_CG_189.jpg'),target_size=(64,64)
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_blight', 'Potato_Late_blight', 'Potato_healthy', 'Tomato_healthy']
index[y[0]]
1/1 [=====] - 8s 31ms/step

Out[50]: 'Tomato_Sectoria_leaf_spot'

```

Figure 6 Test results for plant disease detection using CNN

5.2 Fruit disease detection

The disease detection for fruit dataset using python coding is done. A part of the coding is shown in Figure 7. The number of images for training is 5384 under 6 classes and 1686 images for testing. The trained results are shown in Figure 8. The accuracy is 97.57% and 10 epochs are used for training the data. Once, it is trained, the validation is done by testing the dataset. The tested results are shown in Figure 9 and it is confirmed that the built model works successfully.

```

03-08-2022 17:05 <DIR> ..
03-08-2022 18:05 <DIR> fruit-dataset
03-08-2022 19:48 111,219,456 Fruits.H5
03-08-2022 18:00 <DIR> Veg-dataset
05-08-2022 14:57 111,324,768 Vegetables.H5
2 File(s) 222,544,216 bytes
4 Dir(s) 61,685,266,752 bytes free

In [10]: try('C:\\Users\\admin\\Downloads\\Project\\fruit-dataset\\train',target_size=(64,64),class_mode='categorical',batch_size=32)
6
Found 5384 Images belonging to 6 classes.

In [11]: try('C:\\Users\\admin\\Downloads\\Project\\fruit-dataset\\test',target_size=(64,64),class_mode='categorical',batch_size=32)
6
Found 1686 Images belonging to 6 classes.

In [12]: x_train,class_indices
Out[12]: ['Apple_Black_rot': 0,
'Apple_healthy': 1,
'Corn_(maize)_Northern_Leaf_Blight': 2,
'Corn_(maize)_healthy': 3,
'Peach_Bacterial_spot': 4,
'Peach_healthy': 5]

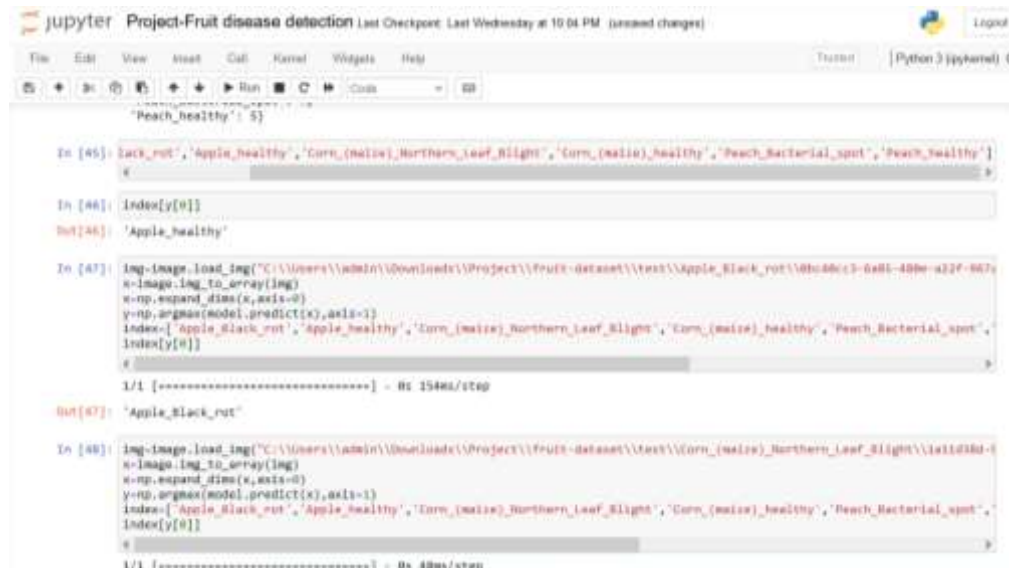
CNN

```

Figure 7 Coding for fruit disease detection using CNN



Figure 8 Training results for fruit disease detection using CNN



(a)


```

Index=[ 'Apple_Black_rot', 'Apple_healthy', 'Corn_(maize)_Northern_Leaf_Blight', 'Corn_(maize)_healthy', 'Peach_Bacterial_spot',
Index[y[0]]
1/1 [=====] - 0s 48ms/step

Out[48]: 'Corn_(maize)_healthy'

In [49]: img=image.load_img('C:\\Users\\admin\\Downloads\\Project\\fruit-dataset\\test\\Peach_Bacterial_spot\\B07F254-8215-4006-954C
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_spot',
Index[y[0]]
1/1 [=====] - 0s 31ms/step

Out[49]: 'Peach_Bacterial_spot'

In [50]: img=image.load_img('C:\\Users\\admin\\Downloads\\Project\\fruit-dataset\\test\\Corn_(maize)_healthy\\2a8f1c3-5c54-481c-a01f
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_spot',
Index[y[0]]
1/1 [=====] - 0s 118ms/step

Out[50]: 'Corn_(maize)_healthy'

```

(b)

Figure 9 Test results for fruit disease detection using CNN

5.3 IBM Deployment

The fruit disease detection is implemented in IBM cloud. All files are saved in the local system itself. Necessary resources are accessed from IBM. Watson-machine-learning-client is installed for local system access. The screenshot of IBM implementation is shown in Figure 10. The model is saved successfully.

```

In [79]: software_spec_uid=client.software_specifications.get_uid_by_name('tensorflow/tf2.1-py3.9')

In [80]: software_spec_uid

Out[80]: 'acd9c798-697a-5d2f-a657-ce88e9860fad'

In [81]: !tar -xvf Project.tar.gz Fruits.tar.gz
# Fruits.tar.gz

In [82]: model_details=client.repository.store_model(model='Project.tar.gz',
meta_props={
client.repository.ModelMetaNames.NAME:'Fruit disease detection',
client.repository.ModelMetaNames.TYPE:'tensorflow 2.1',
client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid
})

In [83]: model_id=client.repository.get_model_id(model_details)

In [84]: model_id

Out[84]: '8c7fa8d-879d-4119-8b2f-68a8bde061a'

In [85]: client.repository.download(model_id,'Fruits.tar.gz')
Successfully saved model content to file: 'Fruits.tar.gz'

Out[85]: 'C:\\Users\\admin\\Downloads\\Project\\Fruits.tar.gz'

```

Figure 10 IBM deployment in local system

5.4 FLASK Deployment

The fruit disease detection is implemented in Flask. All files are saved in the local system itself. Necessary resources are accessed from IBM. Watson_machine_learning is installed for local system access. The coding is written in Jupiter notebook. After the model is saved successfully, the implementation in flask is done. The coding is written using spyter. The screenshot of Flask implementation is shown in Figure 11. The final implementation in html coding is shown in Figure 12.

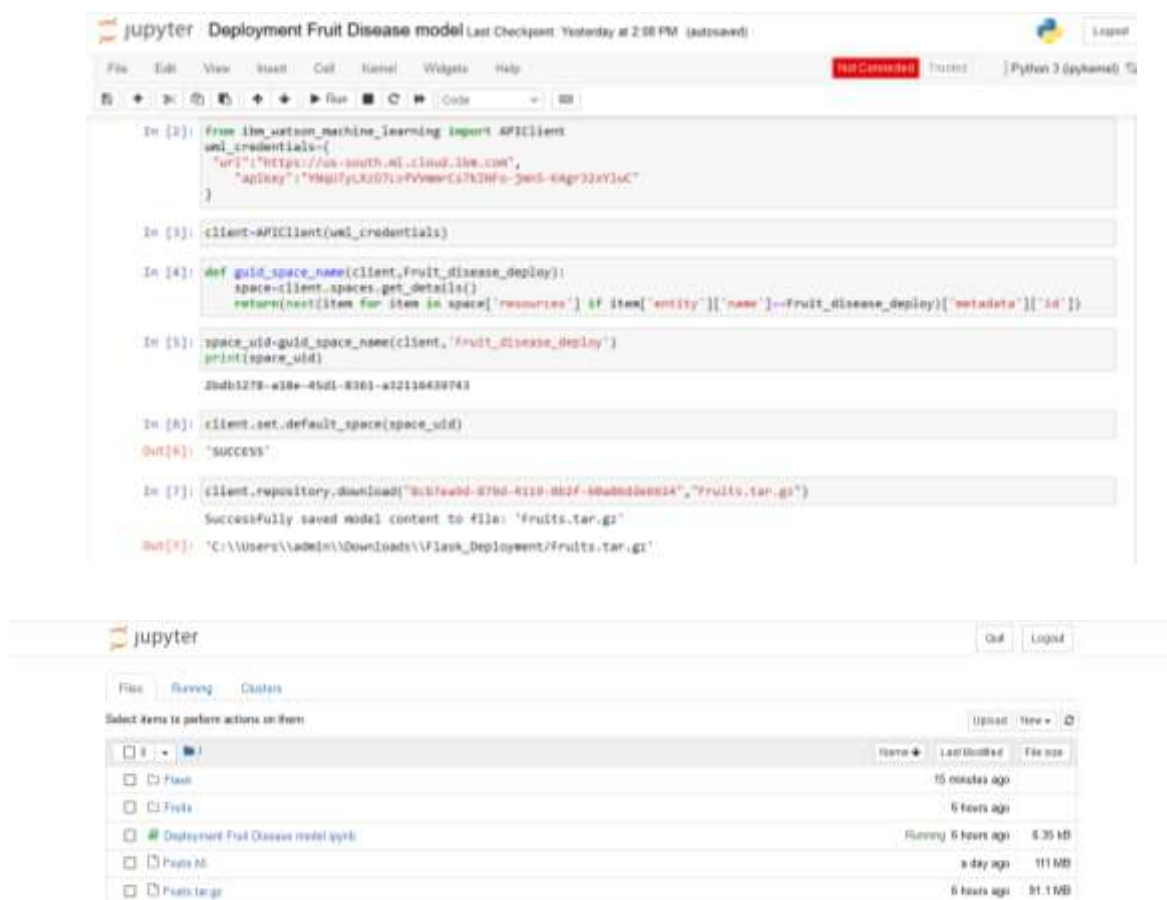


Figure 11 Flask deployment in local system

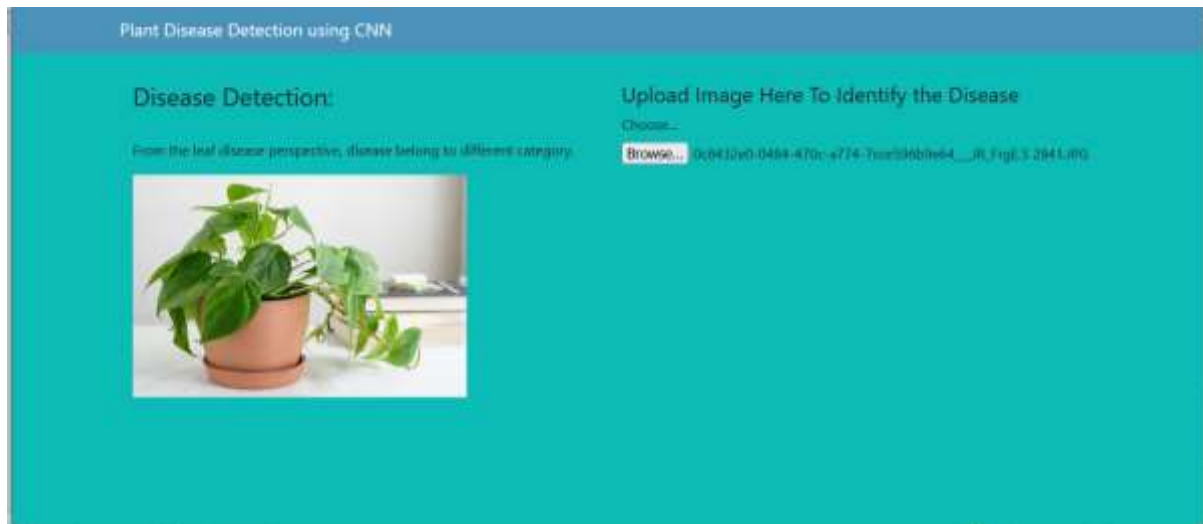


Figure 12 Html deployment of plant disease detection using CNN

6. Conclusion

The disease of a plant is identified from the train dataset and validated and tested using test dataset. The built model identifies the disease using the leaf. The model was built using CNN algorithm and written in jupyter notebook using python code. It is a good experience and learnt about artificial intelligence and machine learning.

Acknowledgment

Author thank AICTE and smartinternz trainers for their valuable input to complete the training successfully and complete the project work.

References

1. Recorded video lecture by smartinetnz