# LITCHI LEAF DISEASES RECOGNITION BY USING IMAGE PROCESSING

**BY**

**IMDADUL HAQUE SHAZU**
ID: 161-15-6870

**MD. MOHSIN ALIM**
ID: 171-15-9276

**MD. MAHBUB ALAM**
ID: 171-15-9283

This Report Submitted as Partial Completion of the Requirements for the Degree of Bachelor of Science in Computer Science & Engineering

**Supervised by**

**MD. TAREK HABIB**

Assistant Professor

Department of Computer Science and Engineering

Daffodil International University

**Co-Supervised by**

**GAZI ZAHIRUL ISLAM**
Assistant Professor

Department of Computer Science and Engineering
Daffodil International University


**DAFFODIL INTERNATIONAL UNIVERSITY**

**DHAKA, BANGLADESH**

**31 JANUARY 2021**

# APPROVAL

This Project titled “Litchi Leaf Disease Recognition by Using Image Processing”, submitted by Md. Imdadul Haque ID No: 161-15-6870 Md. Mohsin Alim ID No:171-15-9276 and Md. Mahbub Alam ID No: 171-15-9283 to the Department of Computer Science and Engineering, Daffodil International University, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on 31.01.2021

## BOARD OF EXAMINERS

**Dr. Touhid Bhuiyan** Chairman

**Professor and Head**

Department of Computer Science and Engineering

Faculty of Science & Information Technology

Daffodil International University

Moushumi Zaman Bonny Internal Examiner

Assistant Professor

Department of computer Science and Engineering

Faculty of Science & Information Technology

Daffodil International University

Md. Sazzadur Ahmed — Internal Examiner

Assistant Professor

Department of computer Science and Engineering

Faculty of Science & Information Technology

Daffodil International University

Dr. Md Arshad Ali — External Examiner

**Associate Professor**

Department of Computer Science and Engineering

Hajee Mohammad Danesh Science and Technology University

# DECLARATION

We hereby announce that, this project has been done by us under the supervision of **Md. Tarek Habib, Assistant Professor, and Department of CSE at Daffodil International University**. We also announce that this project or any a part of this project hasn't been submitted any other place for award or any kind of degree/diploma.

**Supervised by:**

**Md. Tarek Habib**
Assistant Professor,
Department of CSE,

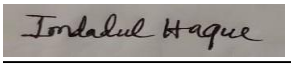
Daffodil International University

**Co- Supervised by:**

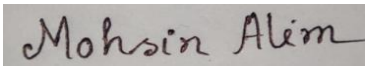
**Gazi Zahirul Islam**
Assistant Professor
Department of CSE,

Daffodil International University

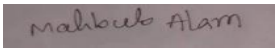
**Submitted by:**



**Md. Imdadul Haque Shazu**
ID: 161-15-6870



**Md. Mohsin Alim Akash**
ID: 171-15-9276



**Md. Mahabub Alam**
ID: 171-15-9283

Department of CSE,
Daffodil International University

# ACKNOWLEDGEMENT

Alhamdulillah for everything, we are thankful to almighty Allah, the most gracious & favorable, to gives us the scope of completing our project.

We would like to thankful to our honorable project supervisor, **Md. Tarek Habib (**Assistant Professor), for his tireless support, continuous guidance, and priceless advice throughout our project work. His rigorous mind unto technical support & his obtainment of solving sudden problems have aided us inventing our way of complete this project. Every time he tries to boost up our mind. Without his suggestion, inspiration and endurance, we couldn't be able to effectively complete our project. We are so lucky and pleased to have him as our supervisor.

We would like to reveal our passionate thankfulness to our family. Thanks a lot for their support and unbounded love to us. They always with us when we go through bad situations, they give inspiration & lot of support to us. Without their support, inspiration, this project work would not have completed. So, we decided to our project is dedicated to them.

We would like to thankful all the faculty member of Computer Science and Engineering, Daffodil International University.

# ABSTRACT

Bangladesh is an agriculture-based country. Most of the people are depends on agriculture. Our Bangladeshi economy also extensively deepens on agriculture. In this term of condition this is become very important to grow our crops and fruits we need to increase their yields. In spite of being drastic and change able climate condition Bangladeshi people and farmers are fighting to grow their crops and fruits in critical way. As an agricultural country Bangladeshi people love fruits, but this is matter of sorrow that our fruits quality and quantity are decreasing for various disease. People of our country are finding many new rare diseases of our local fruits but we are failed to recognize these rare disease and quantity of this problem raised day by day. So a proper treatment or recovery is needed against this problem. As a Bangladeshi people, this is very challenging problem for us to detect this uncommon disease and need classification of these problems. As we are living in an era of technology so obviously technology can play vital role to recognize these diseases. As growing a good plant is depending on plant's leaf so at first this is very important to recognized leaf disease. As a result, we can keep the leaves disease free and fruits also will be being disease free. In our research we are working to recognize leaf diseases. In Bangladesh Litchi is the most popular fruit so we are very interested to work for litchi leaf disease. Hence, we can take part in our Bangladeshi economy by keeping our litchi fruit disease free. For ensuring the freshness of leaves, we work with modern image processing tools that can help us a lot. It is so tough to detect all leaves disease by observing them with eye. Our technology comes with a modern technique named Image Processing. For this purpose, we are using Machine Vision Based Image Processing & CNN (Convolutional Neural Network) Algorithm.

# TABLE OF CONTENTS

# LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

## 1.1 Introduction

Many Bangladeshi people make their living from fruit. Even we are also related directly or indirectly with this sector. In spite of being 8th most populated country in the world, Agriculture sector plays a very important role in our employment more than 47% employee are related in this sector but this sector is not so good in our country. A major part of the GDP around 13% come from agriculture sector. The fruit sector is the biggest sector and manifold economic areas in Bangladesh agriculture. In our country most the farmer is illiterate and they have no idea about the cause of unexpected loss of our economy and how it can impact on our GDP. Agricultural sector consists of cultivating crops animal, animal farming and fishing. If you want to ensure a long-term security of food then we need keep a friendly agricultural system and this only possible by maintaining a sustainable, value able environment. There are many seasonal fruits in Bangladesh, as example Mango, jackfruit, litchi, papaya is produced on around 79% of the harvested area. Fruit cultivation is a traditional part of Bangladeshi agriculture. Litchi's harvested period is May – July. According to Bangladeshi tradition these months are called "Modhu Mash". 46% of fruits is harvested during this "Modhu Mash". At present total annual production of litchi is about 12800 MT. After Completing our demands litchi can also take part in Bangladeshi remittance by exporting in different country. According to the Food and Agriculture Organization (FAO) of the UN, commercially the production and business of litchi is increasing day by day in a higher rate. In our country litchi tree are planted in most courtyards. We need to take care of litchi to sustain our progress. In our research we are working to detect the litchi leaf disease so that the litchi tree can be kept trouble fee and the litchi disease free. We are using a modern technology which belongs to machine learning and this Image Processing technique with CNN Algorithm which can give expected outcome, this will show us which leaf is disease free and which have disease. In this system some images can be captured by camera or any type of camera device then it is uploaded to the system and the system will analyze what is more. The systems fervidity expelled by using Image Processing techniques and this expelled feature give us propose model to predict which type of disease, the captured lives images is having. Our propose model based on CNN which can give us a good accuracy of recognized of the leaf disease.

## 1.2 Motivation

Bangladesh is transcendently a horticultural nation where the agribusiness area assumes a crucial part in quickening financial development. It is hence critical to have a beneficial, practical, and climate neighborly horticultural framework to guarantee long haul food security for individuals. Bangladesh flourishes with a huge assortment of tropical and sub-tropical organic products. Significant organic products, similar to mango, banana, jackfruit, pineapple, papaya, litchi, and jujube are created on 79 percent of the collected zone. Litchi is one of my most favorite fruit. I have a litchi tree in my house but somehow the leaf of the tree is turned into stains. And the stains are increasing day by day for each leaf as a result the tree cannot produce enough fruits. And I see at the end of the situation the leaf is damaging too much and many times in this reason the litchi tree are die. To see this situation, I have an interest in my mind how to solve this problem.in this research to the use of modern technology we should solve this problem.

## 1.3 Objectives

i. Improve agricultural sector.
ii. Data collection.
iii. Identify disease and solution.
iv. To reduce cost of the farmer.
v. Give suggestion to the farmer to produce quality fruits.

## 1.4 Expected Outcome

1. Development of agriculture site.

2. We can easily detect the disease.

2. Use of image processing the error rate is very law.

3. Our farmer can take a step at proper time.

4. Our farmer can produce quality fruits.

5. Encouraged unemployed people to the agriculture sector.

# CHAPTER 2

# BACKGROUND

## 2.1 Introduction

Recognizing of litchi leaf disease by images processing is a process for recognizing the litchi leaf disease. For reorganization at first, we need to some image and select this image as data basic on some criteria as like as Data Collection, Data Preprocessing, Data Selection and Accuracy. Different types of techniques analyzed work will be discussed.

In this chapter the details about the present work, related work, research summery and details about the scope of our work. Here is a brief description about our target and challenges that we have faced.

## 2.2 Related Work

Now a day's image reorganization researcher has been performed and develop system to recognize the disease of fruit leaves. In a research-Machine Vision based papaya disease recognition by Md. Tarek Habib, this system is performed through K-Means clustering, some experiment has been performed by them to show the utility of the propose expert system. By this system the disease attacked region segment out from the capture image. In this work they have achieved 90% accuracy classification.

In another research which was performed to recognize carrot disease. In this research, researcher proposed model based on Multiclass SVM which gave a fruitful result. Disease recognition by image processing techniques is very important to keep our fruits tree disease free. To recognize the disease image processing can be utilized. In our work we use CNN to classify the images and we have used keras to generate our profound model. From a research Hussain et al. proposed and algorithm based on CNN for fruit recognition in this research. He has got 99% accuracy but it was without any extraction. Another machine vision-based fruit classification has been proposed by Zawbaa et al. and Naik and Patel. In this research they have used SVM system. We are going to use CNN because when data is in huge quantity, this CNN algorithm can classify data or image layer by layer.

## 2.3 Research Summary

In this research here is demonstrated that recognize the litchi leaf disease by classification of images. The system will give a better outcome containing with the better accuracy. We will use CNN (Convolutional Neural Network) to classify images pixel to pixel. By this research people will be helped to know the rare diseases of litchi. It will increase our litchi production also.

## 2.4 Challenges

- Although there were much previous documentation or research paper but the litchi leaf disease recognition work is very rare.
- As we collected the raw images for our research so it was tough for us to collect all raw image of leaves from litchi tree because we need a lot of images but the sick leaf is very tough to find and we fined the leaves physically going from here to there.
- The processing of this system was quite difficult with a normal configured device or computer.
- All the images not actually so we needed to give our data in a fixed format so that the system can classify the image properly.
- Sometimes it took a long time to finish the particular process
- As we work with Tensor flow, this library has updated day by day and we had to change our used library with the updating of tensor flow.
- At the time of this corona pandemic, it was very difficult for us to collect the data physically.
- We have faced seasonal problem because this litchi leaf diseases are not available in all season.
- Each used library had to give an importance
- Finally, we can say that processing technique is very challenging and very difficult because the system requires high configured device.

# CHAPTER 3

# Research Methodology

## 3.1 Introduction:

Here are many algorithms in computer science for image processing. We have chosen CNN (Convolutional Neural Network) to classify the 2D images layer by layer. As we work with three litchi leaf disease so form the images of disease leaves this method can classify images efficiently it will address disease from the images of the leaves and determine specific disease like “Leaf Blight “, “Leaf Mite” and” Red Rust”. We have studied to know about these litchi diseases. Some related work about images processing and classification already have some fruitful observation.

## 3.2 Research subject and instrumentation:

Our research work can give a clear concept of our research area in this part we are going to deploy and design, perfect data collection, a proper o data and model training and then the model application to work. For completing our work, we have used windows platform. For implementation of our work we have used python programming language and also used many library packages- CV2, tqdm, matplotlib: pyplot ,opencv, csv, Sklearn ,keras ,numpy and Tensflow 2.3.0. We have used virtual environment and **Jupyter notebook**. We have chosen python because this programming language is very reliable for fast testing of any complex algorithm and for Machine Learning application. If we didn’t use virtual environment, then we faced so much problem about installing library, packages. To avoid this problem, we have used virtual environment.

## 3.3 Data collection procedure:

In our work we have made a new data set for training the proposed network. We have collected leaf images from visiting many places at our village. All of the images are in JPG format and obviously the resolution is not same. In our data set we have about 331 images with three categories of disease like “Leaf Blight”, “Leaf Mite” and “Red Rust”. Out of 331 images, 80% of data for training and 20% for testing.


Figure 3.1: Sample of our dataset before processing (Leaf Blight)


Figure 3.2: Sample of our dataset before processing (Leaf Mite)

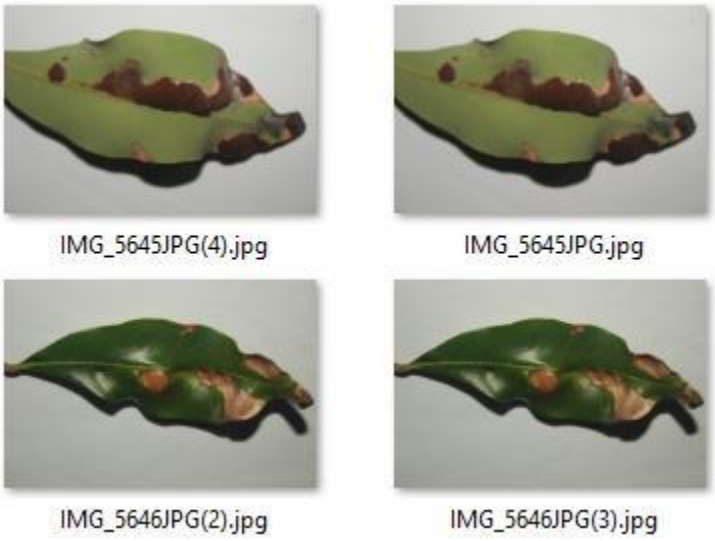



Figure 3.3: Sample of our dataset before processing (Red Rust)

## 3.4 Data processing

This is the challenge to collect images as data and process them. When we collected our images, those images were not in a suitable format to use for image processing and this is difficult for us to correct it one by one. So, we have used a program to sort the images and give them a suitable format to use for the processing. This is an important think to rearrange the data for training and to get better accuracy. Before the data processing our data, size and format is in unorganized. After sorting this we have got our data format organized. Hare is a sample of organized and modified data set. Data/Image name also accepted now.


Figure 3.4: Sample of our sorted dataset (Leaf Blight)


Figure 3.5: Sample of our sorted dataset (Leaf Mite)


Figure 3.6: Sample of our sorted dataset (Red Rust)

# CHAPTER 4

## Working procedure

**4.1 Introduction:** Here we have discussed about working process from the first step. A brief description to set up the environment, activation, Installing Jupyter Notebook.

**Virtual environment activation:**

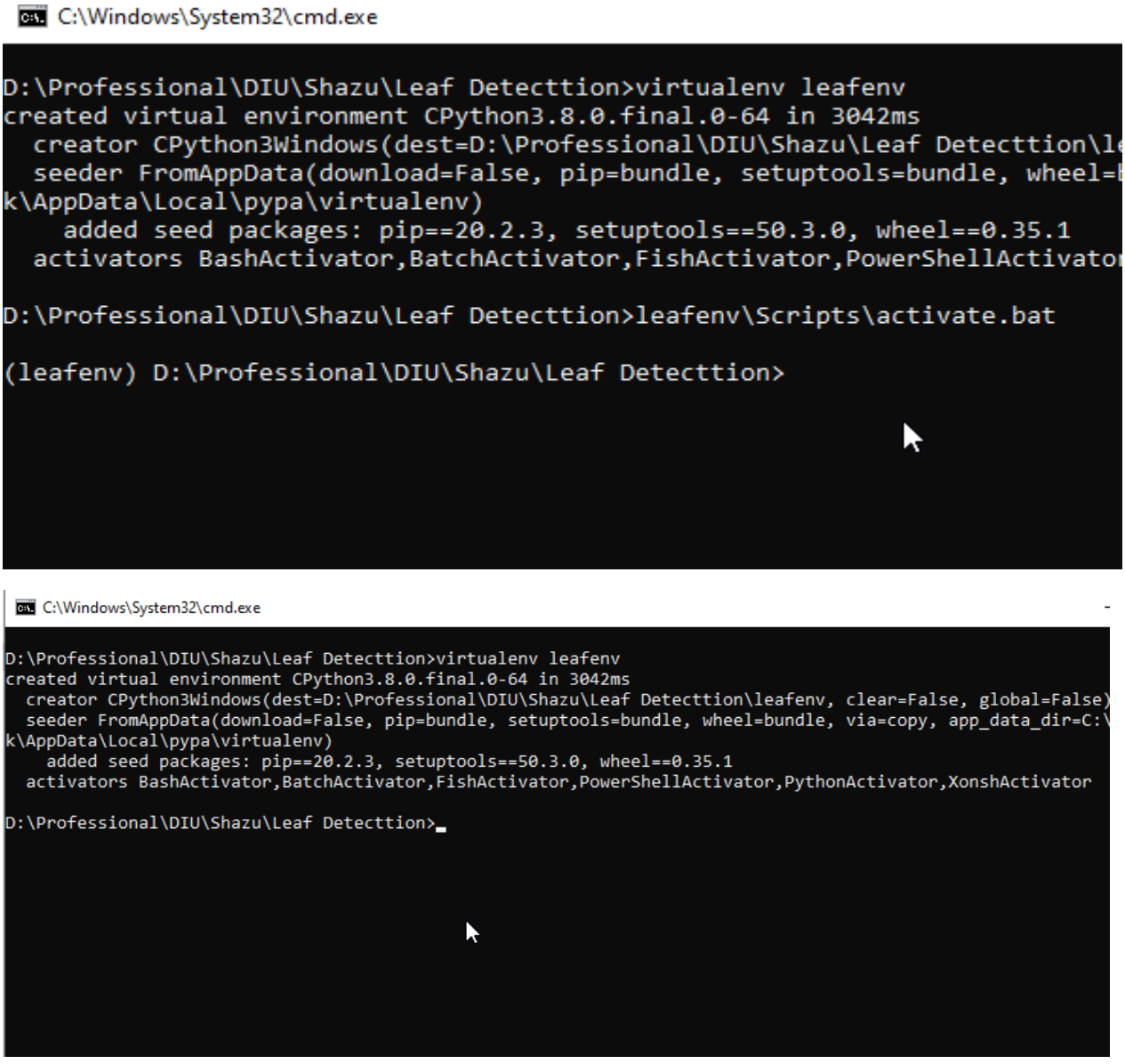



Figure 4.1: Virtual Environment Activating

After activating the virtual environment, we installed the **Jupyter notebook**.

Virtual environment is very friendly and here is no problem for package installation, library installation. All of are included here

**Jupyter Notebook installation-**

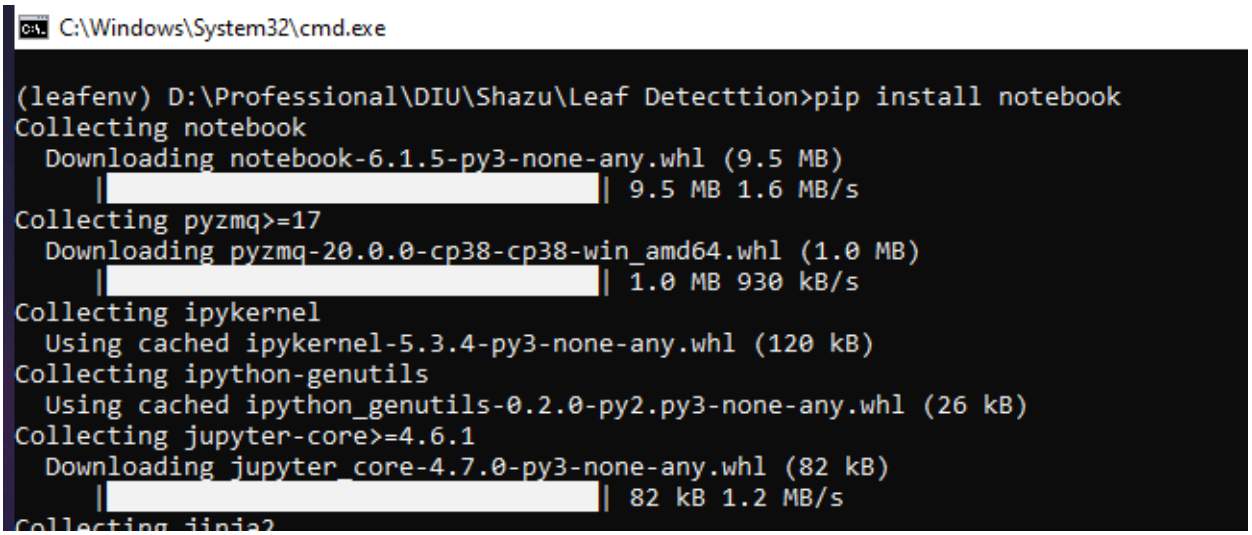



Figure 4.2: Jupyter Notebook Installation

**Opening Notebook--**

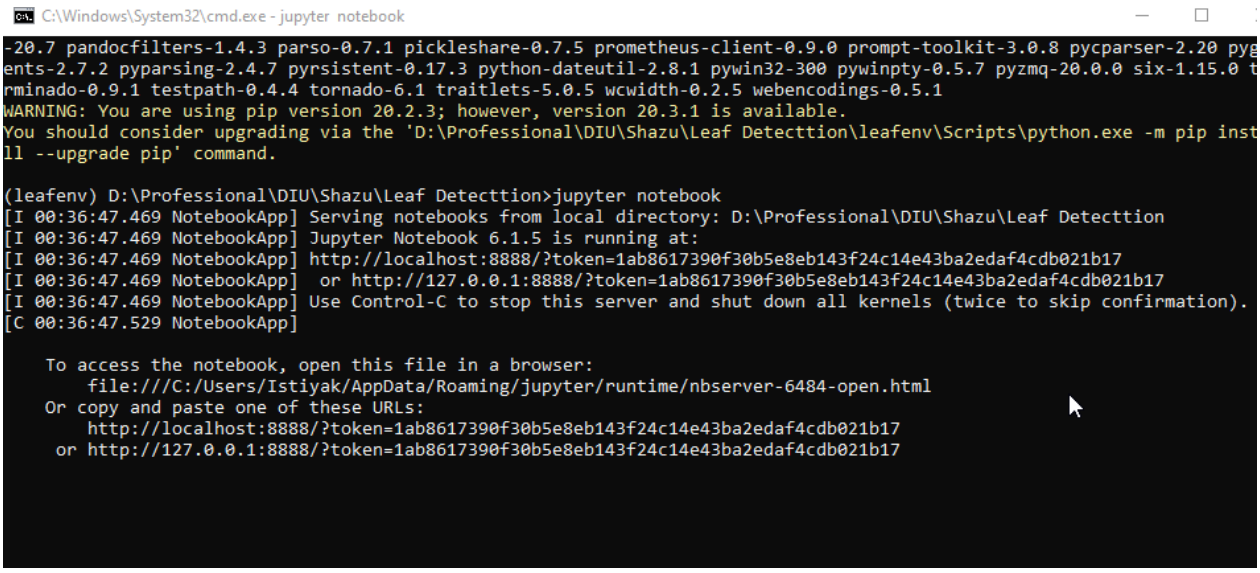



Figure 4.3: Opening Notebook

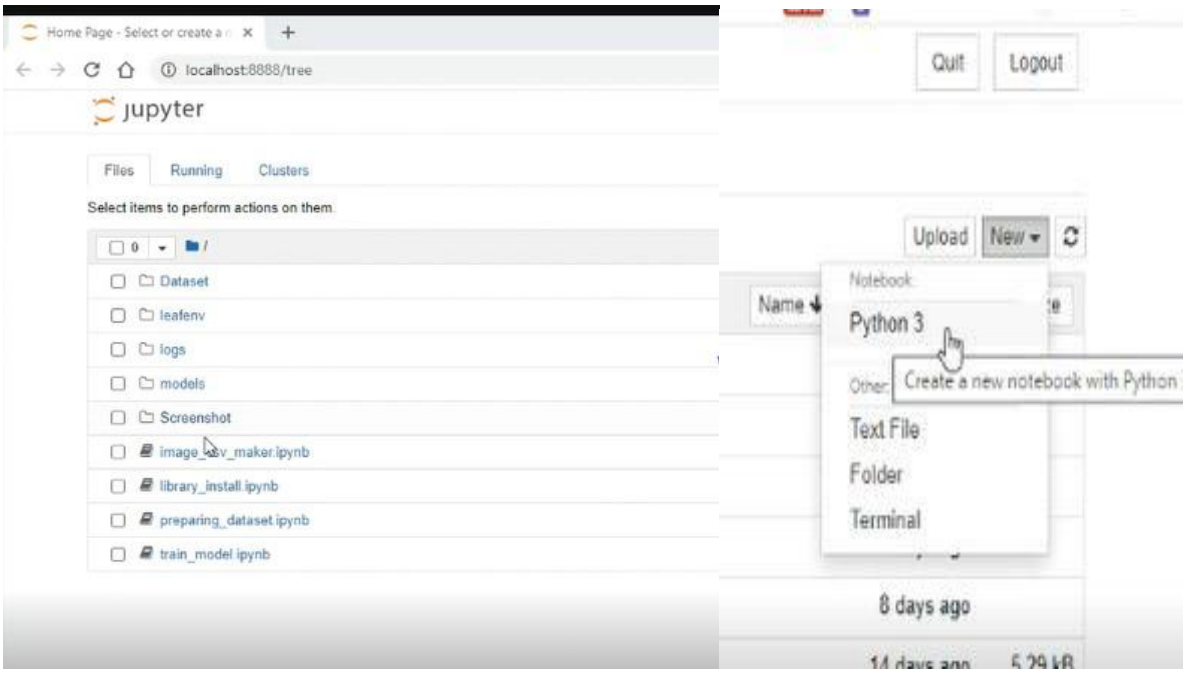



Figure 4.4: Jupyter Notebook Interface

This a sample of Jupyter notebook interface. Here we have created our python file and started to work implementation.

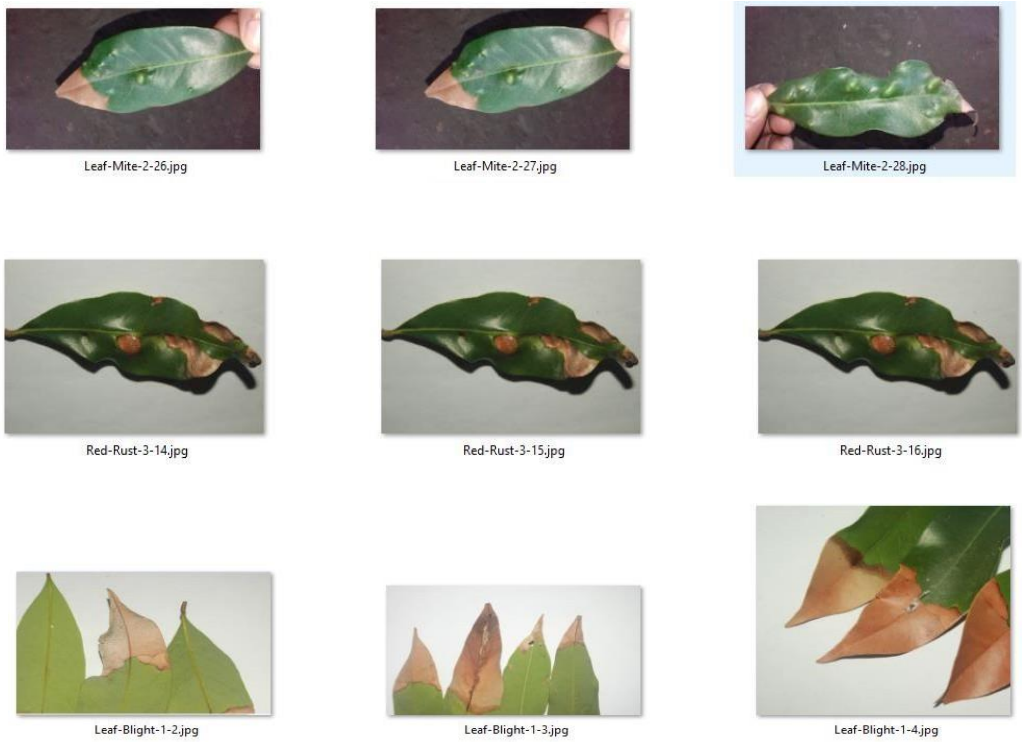



Figure 4.5: Sample of testing dataset


Figure 4.6: Datasets for Result Show

Images are name as 1, 2, 3. I also could know which image belongs to which disease. So, after learning the machine it will say as output that is Leaf Blight/ Leaf Mite/ Red Rust.

## 4.2 Required Library Installation

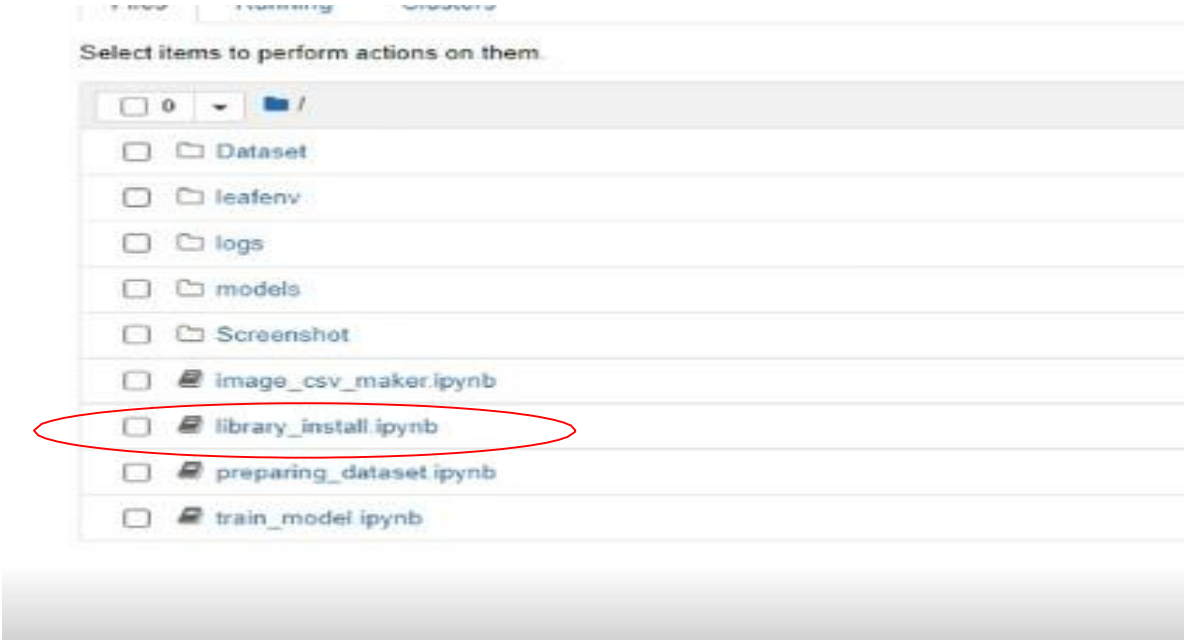



Figure 4.7: Library Installation

**4.2.1 OpenCV –** Compute Vision is the process by which we can understand any images and videos and this is also understood that how this image will be manipulated. This is highly used in Artificial Intelligence. openCV is the library of computer vision, machine learning and image processing. This is used actually for any type recognition like image recognition, face recognition, handwrite recognition, self-car driving, counting traffic, medical image analysis, object

recognition etc. At the point when it incorporated with different libraries, for example, Numpuy, python is equipped for preparing the OpenCV cluster structure for investigation.

**4.2.2 Numpy:** This is a python library utilized for working with arrays. Here we are using 20.0.3 version of numpy. Numpy is 50x faster to serve any array object. Numpy array is called ndarray. This provides a lot of supporting function very easy to work with ndarray. Our collected images have value, height width. These were in an array and these were modified by numpy.

**4.2.3 Tqdm : tqdm** is the library of python with is very helpful to show the progress simple and hassle free. Here we need tqdm to see our progress very smartly. Loop designing.

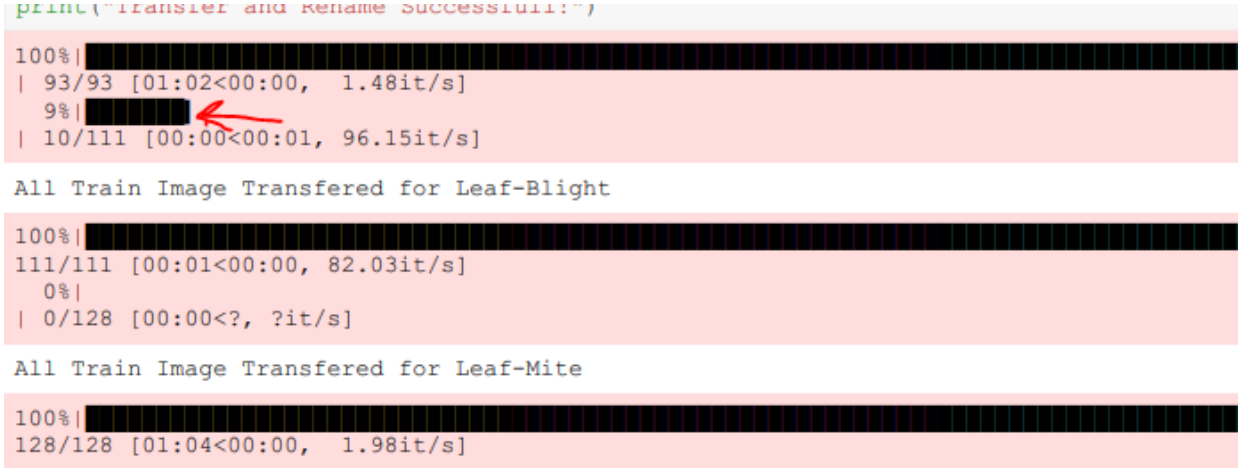



Figure 4.8: Progress showing

**4.2.4 CSV**: Python csv is used for arrange the tabular data. Data populated by this library from excel file data.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | FileName | Deseas | index | width | height |
| 2 | Leaf-Blight-1-3.jpg | Leaf-Blight | 1 | 466 | 812 |
| 3 | Leaf-Blight-1-4.jpg | Leaf-Blight | 1 | 488 | 528 |
| 4 | Leaf-Blight-1-5.jpg | Leaf-Blight | 1 | 4000 | 6000 |
| 5 | Leaf-Blight-1-6.jpg | Leaf-Blight | 1 | 4000 | 6000 |
| 6 | Leaf-Blight-1-7.jpg | Leaf-Blight | 1 | 4000 | 6000 |

Figure 4.9: CSV Dataset

**4.2.5 TensorFlow 2.3.0**: In our project we have used **2.3.0** version of TensorFlow because it occurred mismatch with graphics card. So, we are using **2.3.0** version. This is a python library

produced by Google. Here we are using it to build our main model. This is an open-source library and this will work so faster. This library is the all-in-one type library by which we can use for image processing, data processing, image classification, model recognition, natural language processing. This is a very powerful framework that functionality implements series of nodes and each and every node will expose the mathematical operation. With the entire series of nodes graph can be generated. Though the installation of tensor flow more minor libraries also installed.

**4.2.6 Keras:** This is a high-level API that works as a TensorFlow function and Machine Learning library. This have been configured to work with Python and can be worked without any modification and configured. We work with keras with fulfilling the requirements – installing python, numpy, installing Tensorflow 2.3.0. This library also helped us to generate the model.

**4.2.7 Sklearn(Scikit-learn):** This library is used as machine learning library with python programming language. For our image processing it can give features, furthermore classification, clustering algorithm, K-means clustering and DBSCAN. This library is built to interoperate with the python numerical and scientific libraries NumPy & SciPy.

Generate the train set:

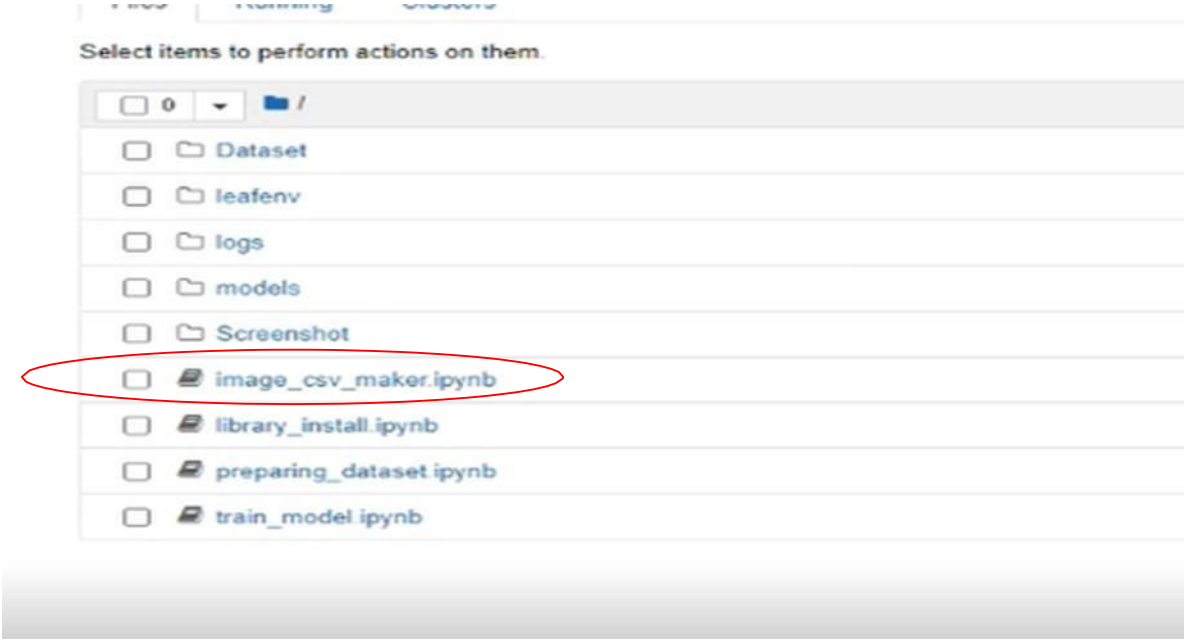



Figure 5.0: csv Dataset for train

Here we have generated a train set.

### 4.2.8 Image_CSV_maker:

```
In [21]:
import os
import cv2
import numpy as np
from tqdm import tqdm

In [22]:
```

Figure 5.1: Initially Import Libraries

To do any work with Python first work is import the required packages. os for operating system to access the folder, cv2 for image analysis, numpy for data processing, tqdm for loop design.

```
In [22]:
DESEASES= ["Leaf-Blight","Leaf-Mite","Red-Rust"]
mainDirectory = "Dataset/MainImages/"
trainDirectory = "Dataset/train_data/"

In [36]:
deseas_images = []
deseas_index = []
deseas_name =[]
np_sets =  ["FileName", "Deseas", "Index","width","height"]
```

Figure 5.2: Image Directory and Indexing for Data

We have assigned our image directory of folder where the system can find the images and where the train dataset will be stored.

Then the csv file will be generated by indexing as Filename, Deseas, Index, width, height.

```
In [37]:
dIndex=0

for deseas in DESEASES:
    dIndex=dIndex+1
    imgNameIndex =1
    for image_path in tqdm(os.listdir(mainDirectory+deseas)):
        imgNameIndex =imgNameIndex+1
        img = cv2.imread(mainDirectory+deseas+'/'+image_path)
        width, height = img.shape[:2]
        filename = deseas + '-' + str(dIndex) + '-'+str(imgNameIndex)+'.jpg'
        filePath = trainDirectory+filename
        cv2.imwrite(filePath, img)
        deseas_images.append(filename)
        deseas_name.append(deseas)
        deseas_index.append(dIndex)
        np_sets.append([filename,deseas,str(dIndex),str(width),str(height)])
    print("All Train Image Transfered for "+deseas)

print("Transfer and Rename Successfull!")
```

```
100%|██████████████████████████████████████████████████████████████████████████████
| 93/93 [01:02<00:00,  1.48it/s]
  9%|███████
| 10/111 [00:00<00:01, 96.15it/s]

All Train Image Transfered for Leaf-Blight

100%|██████████████████████████████████████████████████████████████████████████████|
111/111 [00:01<00:00, 82.03it/s]
  0%|
| 0/128 [00:00<?, ?it/s]

All Train Image Transfered for Leaf-Mite

100%|██████████████████████████████████████████████████████████████████████████████|
128/128 [01:04<00:00,  1.98it/s]

All Train Image Transfered for Red-Rust
Transfer and Rename Successfull!
```

Figure 5.3: Checking and Counting our Data/Images

Here are accessed the directory of images and we have 331 images in total, the checking count of the image will same number of images.

**4.2.9 Image.shape** function is used to get the images height and width.

Then it will generate new train set of data in sorted. Loop progress has been shown very nicely.

```
In [38]:
arr = np.array(np_sets)

<ipython-input-38-84adb5597a12>:1: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray
  arr = np.array(np_sets)

In [41]:
import csv

In [42]:
myFile = open('Dataset/excels/train_set.csv', 'w')
with myFile:
    writer = csv.writer(myFile)
    writer.writerows(arr)
```

Figure 5.4: Convert to numpy array

arr = np.array(np_sets) - by this we set the array and convert it to numpy sets Then we can get as CSV data set and this csv also can be used in future. With myFile: writer = csv.writer(myFile) writer.writerows(arr) – write the csv file from ndarray

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | FileName | Deseas | index | width | height |
| 2 | Leaf-Blight-1-3.jpg | Leaf-Blight | 1 | 466 | 812 |
| 3 | Leaf-Blight-1-4.jpg | Leaf-Blight | 1 | 488 | 528 |
| 4 | Leaf-Blight-1-5.jpg | Leaf-Blight | 1 | 4000 | 6000 |
| 5 | Leaf-Blight-1-6.jpg | Leaf-Blight | 1 | 4000 | 6000 |
| 6 | Leaf-Blight-1-7.jpg | Leaf-Blight | 1 | 4000 | 6000 |
| 7 | Leaf-Blight-1-8.jpg | Leaf-Blight | 1 | 4000 | 6000 |
| 8 | Leaf-Blight-1-9.jpg | Leaf-Blight | 1 | 4000 | 6000 |
| 9 | Leaf-Blight-1-10.jpg | Leaf-Blight | 1 | 4000 | 6000 |
| 10 | Leaf-Blight-1-11.jpg | Leaf-Blight | 1 | 6000 | 4000 |
| 11 | Leaf-Blight-1-12.jpg | Leaf-Blight | 1 | 6000 | 4000 |
| 12 | Leaf-Blight-1-13.jpg | Leaf-Blight | 1 | 6000 | 4000 |
| 13 | Leaf-Blight-1-14.jpg | Leaf-Blight | 1 | 4000 | 6000 |
| 14 | Leaf-Blight-1-15.jpg | Leaf-Blight | 1 | 4000 | 6000 |
| 15 | Leaf-Blight-1-16.jpg | Leaf-Blight | 1 | 4000 | 6000 |
| 16 | Leaf-Blight-1-17.jpg | Leaf-Blight | 1 | 6000 | 4000 |
| 17 | Leaf-Blight-1-18.jpg | Leaf-Blight | 1 | 6000 | 4000 |
| 18 | Leaf-Blight-1-19.jpg | Leaf-Blight | 1 | 6000 | 4000 |
| 19 | Leaf-Blight-1-20.jpg | Leaf-Blight | 1 | 4000 | 6000 |

Figure 5.5: csv dataset with images index no. and height, width

**4.3 Dataset preparing:** In this portion we have done our maximus work

```
In [1]:
import numpy as np
import pandas as pd
import  matplotlib.pyplot as plt
from tqdm import tqdm
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing import image
```

Figure 5.6: Import Required Library

Here, required library packages have been installed. Panda is used for visualizing the data. Matplotlib is used to plot the data by visualizing on graph.

```
In [2]:
data = pd.read_csv('Dataset/excels/train_set_mod.csv')
data.shape

Out[2]:

(331, 5)
```

Figure 5.7: Count csv dataset

Then the system will read the csv file and give the out with 331 number of images and 5 columns csv file. 1st row was the column name and this is understood by the system automatically.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | FileName | Deseas | index | width | height |
| 2 | Leaf-Blight-1-3.jpg | Leaf-Blight | 1 | 466 | 812 |
| 3 | Leaf-Blight-1-4.jpg | Leaf-Blight | 1 | 488 | 528 |
| 4 | Leaf-Blight-1-5.jpg | Leaf-Blight | 1 | 4000 | 6000 |
| 5 | Leaf-Blight-1-6.jpg | Leaf-Blight | 1 | 4000 | 6000 |

Figure 5.8: csv dataset

Then we have taken all the height and width and we have gotten 331 height and width from the images. We need both height and width because of input layer

**Input Layer** is consisting of neurons. Every input is called neuron.

i) The input layer passes the information straightforwardly to the primary concealed layer where the information is increased by the principal shrouded layer's loads.

ii) The input layer goes the information through the initiation work prior to passing it on. The information is then increased by the principal concealed layer's loads.

iii) The information layer has its own loads that duplicate the approaching information. The information layer at that point goes the information through the actuation work prior to passing it on. The information is then increased by the primary concealed layer's loads

After plot the data on a figure we have got this

```
plt.scatter(heights, widths)
plt.show()
```

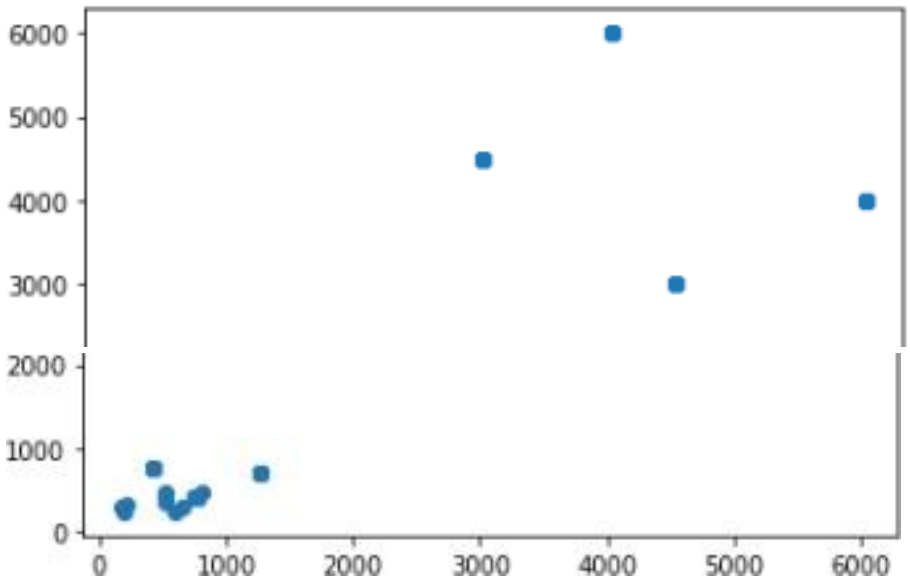



Figure 5.8: Graphical figure of data

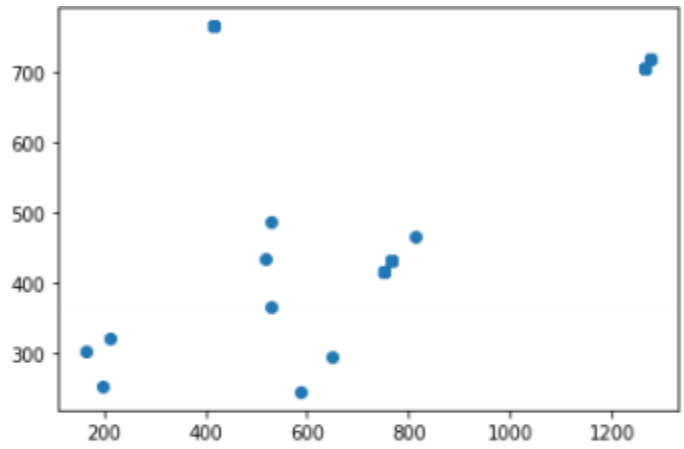
Maximum data according to the height weight are addressed in 1000.

Then if we filter these data then we will get another graphical figure.

```
In [8]:
filtered_data = pd.read_csv('Dataset/excels/train_set_mod.csv')[lambda x: x['width'] < 1
000]

In [9]:
plt.scatter(filtered_data['height'], filtered_data['width'])
plt.show()
```

Figure 5.9: csv data read from csv dataset




```
In [10]:
filtered_data.shape
Out[10]:
(113, 5)
```

Figure 6.0: Filtered data

Here we have got 113 images.

| | FileName | Deseas | index | width | height |
|---|---|---|---|---|---|
| 0 | Leaf-Blight-1-3.jpg | Leaf-Blight | 1 | 466 | 812 |
| 1 | Leaf-Blight-1-4.jpg | Leaf-Blight | 1 | 488 | 528 |
| 92 | Leaf-Mite-2-2.jpg | Leaf-Mite | 2 | 765 | 416 |
| 93 | Leaf-Mite-2-3.jpg | Leaf-Mite | 2 | 765 | 416 |
| 94 | Leaf-Mite-2-4.jpg | Leaf-Mite | 2 | 765 | 416 |

Figure 6.1: sample of 113 filtered images

```
In [12]:
data['width'].mean()
Out[12]:
2948.081570996979

In [13]:
filtered_data['width'].mean()
Out[13]:
635.5309734513274
```

Figure 6.2: mean of filtered data (width)

In perspective of mean after calculating mean for 331 we have got 2948.08157996979 because the number of images width above 1000 is much and after calculating under the 1000 we have got 635.

```
In [14]:
data['height'].mean()
Out[14]:
3461.5045317220543

In [15]:
filtered_data['height'].mean()
Out[15]:
617.3274336283185
```

Figure 6.3: mean of filtered data (height)

In perspective of height, we have got mean 3461.504531 above 1000 and got 617.327433662 under 1000. This was not fruitful and it shows so much difference so we need to calculate the median

after calculating median of height and width we have got same shaped data. So, we can work with it.

```
plt.scatter(heights, widths)
plt.xlabel("heights")
plt.ylabel("widths")
plt.plot([0,max(heights)],[0,max(widths)])
plt.show()
```

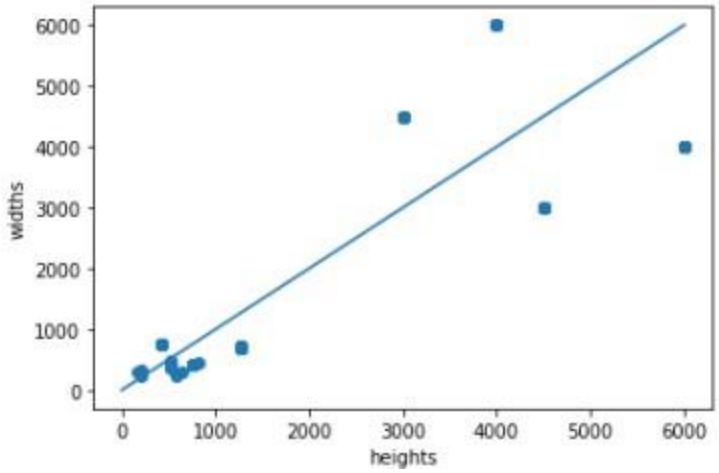



Figure 6.4: plotting figure according to height, width

After the linear regression of data and plotting figure is generated. This is safe to work with the point which points are addressed near the line.

```
all_deseases = data['Deseas']
```

```
In [26]:
```

```
unic_deseases = list(set(all_deseases))
unic_deseases
```

```
Out[26]:
```

```
['Red-Rust', 'Leaf-Blight', 'Leaf-Mite']
```

```
In [27]:
```

```
all_index = data['index']
unic_index = list(set(data['index']))
unic_index
```

```
Out[27]:
```

```
[1, 2, 3]
```

Figure 6.5: checking unique 3 diseases

After checking all unique diseases, we have got our all data is right here is no mistake. The number of diseases is three.

## 4.4 Proposed Methodology

**4.4.1 Convolutional neural Network (CNN)** is a particular kind of artificial neural network. This is special for deep learning that utilizes perceptron, a grinding AI unit calculation, managed learning, to break down various sorts of information. CNN operatin for the most part work relies upon contributions for separating design acknowledgment and it functions admirably with information that has a spatial relationship CNN likewise has a learnable boundary like neural network This CNN work with some layer input layer,output layer and various output layers.

We can apply in Images, Classification forecast issues, face recognition, object recognition, and so forth There are a few layers to group a picture, for example, CNN's first layer is the info layer. This layer will read the images pixel to pixel. Various kinds of pooling layers , This will put the images in a reliable shape for the system.

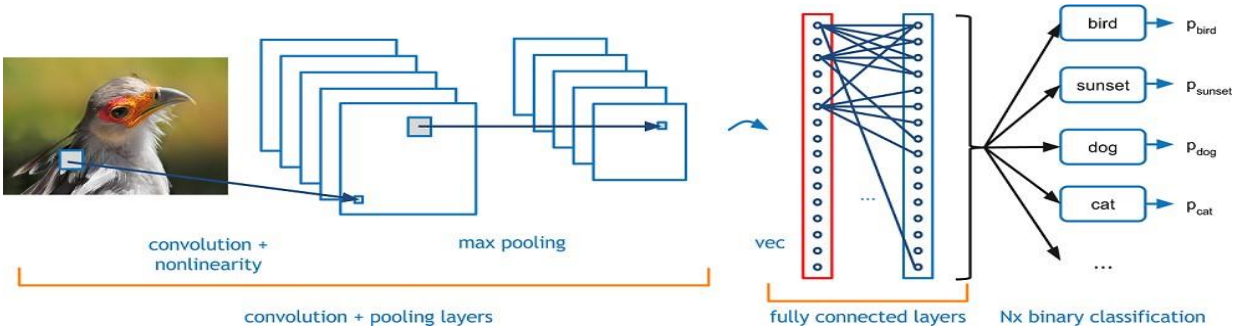



Figure 6.6: CNN based image classification

**4.4.2 Pooling layer: Pooling layer** is a non-straight layer that isolates the component of the info and diminishes the quantity of boundaries, controlling overfitting and most significant data is saved. We can characterize the size of the PoolingLayer that can eliminate superfluous highlights and keep the fundamental highlights. Here are three sorts of pooling layer MaxPooling, AveragePooling, and MinPooling.

- MaxPooling: It picks just the greatest worth contained in the pooling window. In CNN engineering MaxPooing is generally utilized for perceiving important highlights on the grounds that MaxPooling gives a superior come about because of AveragePooling and MinPooling layer.

- AveragePooling: It picks just Average worth contained in the pooling window.
- MinPooling: It picks just Minimum worth contained in the pooling window.

**4.4.3 Flatten layer: Flatten layer** in the middle of the convolutional layer and completely associated layer, there's a 'Smooth' layer. It's changing over all the resultant 2-dimensional clusters into a 1D element vector, this activity is called smoothing. This smoothing structure makes a solitary long consistent direct vector to be utilized by the thick layer for the last arrangement layer.

**4.4.4 Fully connected layer: Fully connected layer** is the last stage for a CNN organization, it speaks to the component vector for the information. In FC layer includes loads, predispositions, and neurons. It interfaces neurons in a single layer to neurons in another layer. FC layers recombine every neuron to efficiently and precisely order each information. It is utilized to arrange pictures between various classifications via preparing. FC layers can be stood out from Multilayer Perceptron (MLP) where every neuron has total connections with all past layer initiations.

## 4.5 :Train Methodology

In this context we will discuss about the trainning methodology. By training, machine will understand the data which I have input for learn. So how to train, how will be trained data and how is it processed are described here.

```
In [29]:
Train_Height = 635
Train_Width = 617

In [30]:
X=[]
for i in tqdm(range(data.shape[0])):
    path = 'Dataset/train_data/'+data['FileName'][i]
    img = image.load_img(path,target_size=(Train_Height,Train_Width,3))
    img = image.img_to_array(img)
    img = img/255.0
    X.append(img)
X= np.array(X)
100%|████████████████████████████████████████████████████████████████████████████| 33
1/331 [00:22<00:00, 14.57it/s]

In [31]:
X.shape
Out[31]:
(331, 635, 617, 3)
```

Figure 6.7: Trained for classification

Here we have choosen a figure with 635/617, then load the images. We have devided these images by 255.0 because we have color images. Then the system will take the pixels value between 1 and 0 because the machine can't understans any integer value. This gives the output 331,635,617,3 this is th 3 dimentional value and every index have contained 3 values red, green and blue. Here are going 635*617*3= 1175385 neurons as input.

```
ddf = ddf.groupby('index')
```

```
In [40]:
```

```
ddf = ddf.first()
ddf.head()
```

```
Out[40]:
```

| index | FileName | Deseas | width | height |
|---|---|---|---|---|
| 1 | Leaf-Blight-1-3.jpg | Leaf-Blight | 466 | 812 |
| 2 | Leaf-Mite-2-2.jpg | Leaf-Mite | 765 | 416 |
| 3 | Red-Rust-3-2.jpg | Red-Rust | 4000 | 6000 |

Figure 6.8: Groupby data set

```
In [52]:
```

```
new_data = pd.read_csv('Dataset/excels/train_set_mod_processed.csv')
new_data.head()
```

```
Out[52]:
```

| | Filename | Deseas | Leaf-Blight | Leaf-Mite | Red-Rust |
|---|---|---|---|---|---|
| 0 | Leaf-Blight-1-3.jpg | Leaf-Blight | 1 | 0 | 0 |
| 1 | Leaf-Blight-1-4.jpg | Leaf-Blight | 1 | 0 | 0 |
| 2 | Leaf-Blight-1-5.jpg | Leaf-Blight | 1 | 0 | 0 |
| 3 | Leaf-Blight-1-6.jpg | Leaf-Blight | 1 | 0 | 0 |
| 4 | Leaf-Blight-1-7.jpg | Leaf-Blight | 1 | 0 | 0 |

Figure 6.9: machine realization data set for Leaf Blight

This is the new generating dataset for realization of machine.

## Model Training Methodology

```
model = Sequential()
model.add(Conv2D(16,(3,3),activation='relu',input_shape=x_train[0].shape))
model.add(BatchNormalization())
model.add(MaxPool2D(2,2))
model.add(Dropout(0.2))


model.add(Conv2D(32,(3,3),activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(2,2))
model.add(Dropout(0.3))


model.add(Conv2D(64,(3,3),activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(2,2))
model.add(Dropout(0.4))


model.add(Conv2D(128,(3,3),activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(2,2))
model.add(Dropout(0.5))



model.add(Flatten())
model.add(Dense(128,activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
```

Figure 7.0: Training Model

Here er have used CNN, Relu function by which images will be classified, a model also will be generated.

**4.5.1 Rectified Linear Units (ReLU):** ReLU is the most generally utilized enactment work while planning networks today. ReLU work is nonlinear and considers back propagation. The steady slope of ReLUs brings about quicker learning since it doesn't initiate all the neurons simultaneously like as though the info is negative it will change over into zero and that neuron doesn't get actuated. So, couple of neurons are dynamic at a time, not all neurons, at this explanation ReLU a lot simpler, quicker and make more productive. More organic roused to

prepare.

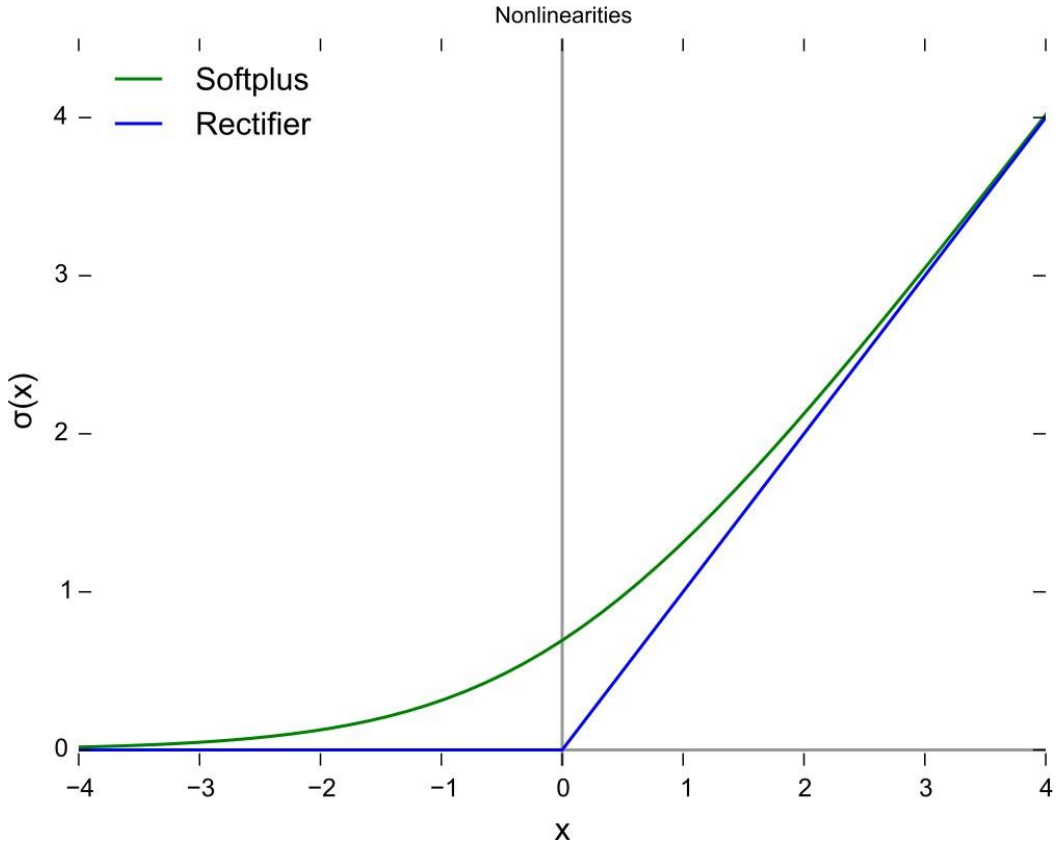



Figure 7.1: ReLu Function.

After testing and training our data set, we have got this model by which can evaluate and get the training result.

```
In [11]:
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 633, 615, 16)      448
_________________________________________________________________
batch_normalization (BatchNo (None, 633, 615, 16)      64
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 316, 307, 16)      0
_________________________________________________________________
dropout (Dropout)            (None, 316, 307, 16)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 314, 305, 32)      4640
_________________________________________________________________
batch_normalization_1 (Batch (None, 314, 305, 32)      128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 157, 152, 32)      0
_________________________________________________________________
dropout_1 (Dropout)          (None, 157, 152, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 155, 150, 64)      18496
_________________________________________________________________
batch_normalization_2 (Batch (None, 155, 150, 64)      256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 77, 75, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 77, 75, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 75, 73, 128)       73856
_________________________________________________________________
batch_normalization_3 (Batch (None, 75, 73, 128)       512
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 37, 36, 128)       0
_________________________________________________________________
dropout_3 (Dropout)          (None, 37, 36, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 170496)            0
_________________________________________________________________
dense (Dense)                (None, 128)               21823616
_________________________________________________________________
batch_normalization_4 (Batch (None, 128)               512
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 387
=================================================================
Total params: 21,922,915
Trainable params: 21,922,179
Non-trainable params: 736
_________________________________________________________________
```

Figure 7.2: Model summery

# CHAPTER 6

# EXPERIMENTAL RESULTS AND DISCUSSION

**5.1 Introduction:** In this section we will discuss about performance evaluation of our model, accuracy level.

After 15 number of epoch, we have got that accuracy –

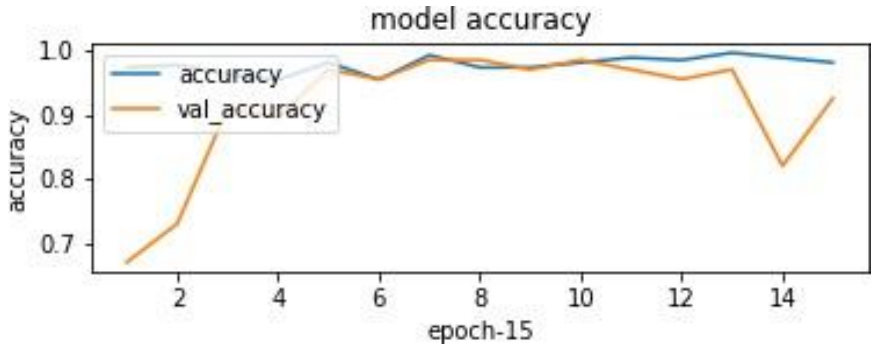



Figure 7.3: Accuracy and Performance evaluation

We are seeing that there is not more difference between accuracy and Val accuracy, so this time of model is accepted.

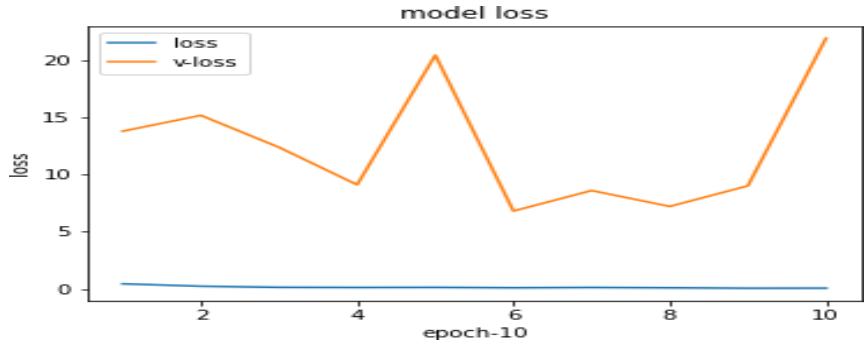



Figure 7.4: Accuracy and Performance evaluation

Here is some epoch loss, so at 15 epoch we have got the best accuracy.

## 5.2 Result-

Now we have got the result for the recognition of 3 diseases. Here is the result for 1st disease- Leaf-Blight. Our proposed system has recognized 99.31% of Leaf-Blight.

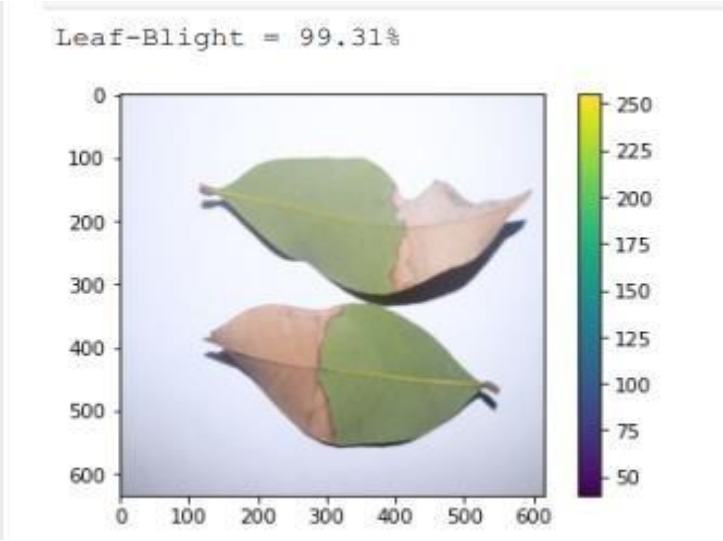



Figure 7.5: 99.31% accuracy for Leaf -Blight

Then, here is the result for 2nd disease – Leaf-Mite. This is the great result that 100% recognition of Leaf-Mite disease.

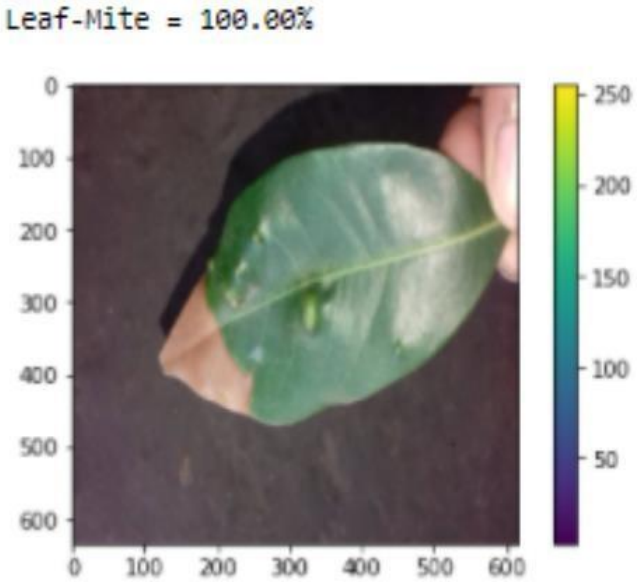



Figure 7.6: 100.00% accuracy for Leaf -Mite

At 3rd we have got the result for Red Rust and we have got 99.88%

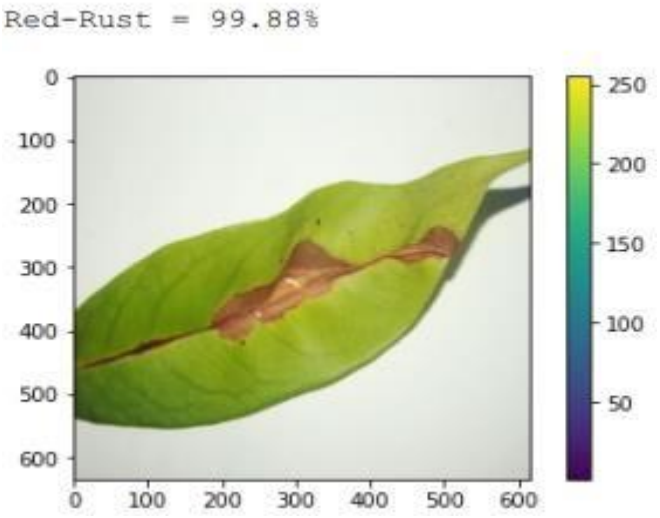



Figure 7.7: 99.88% accuracy for Red Rust

Actually this is very tough to recognize properly but within 3 diseases we can recognize one disease perfectly with the accuracy of 100.00%

## 5.3 Requirements Accessories:

After the description of our CNN methodology and completely trained our model these are as requirements to work for image processing

- Operating System (Windows 10)
- Hard Disk (minimum 1000)
- Ram (Minimum 8 GB)
- GPU(Recommended) – we have used 16GB 1660 GPU

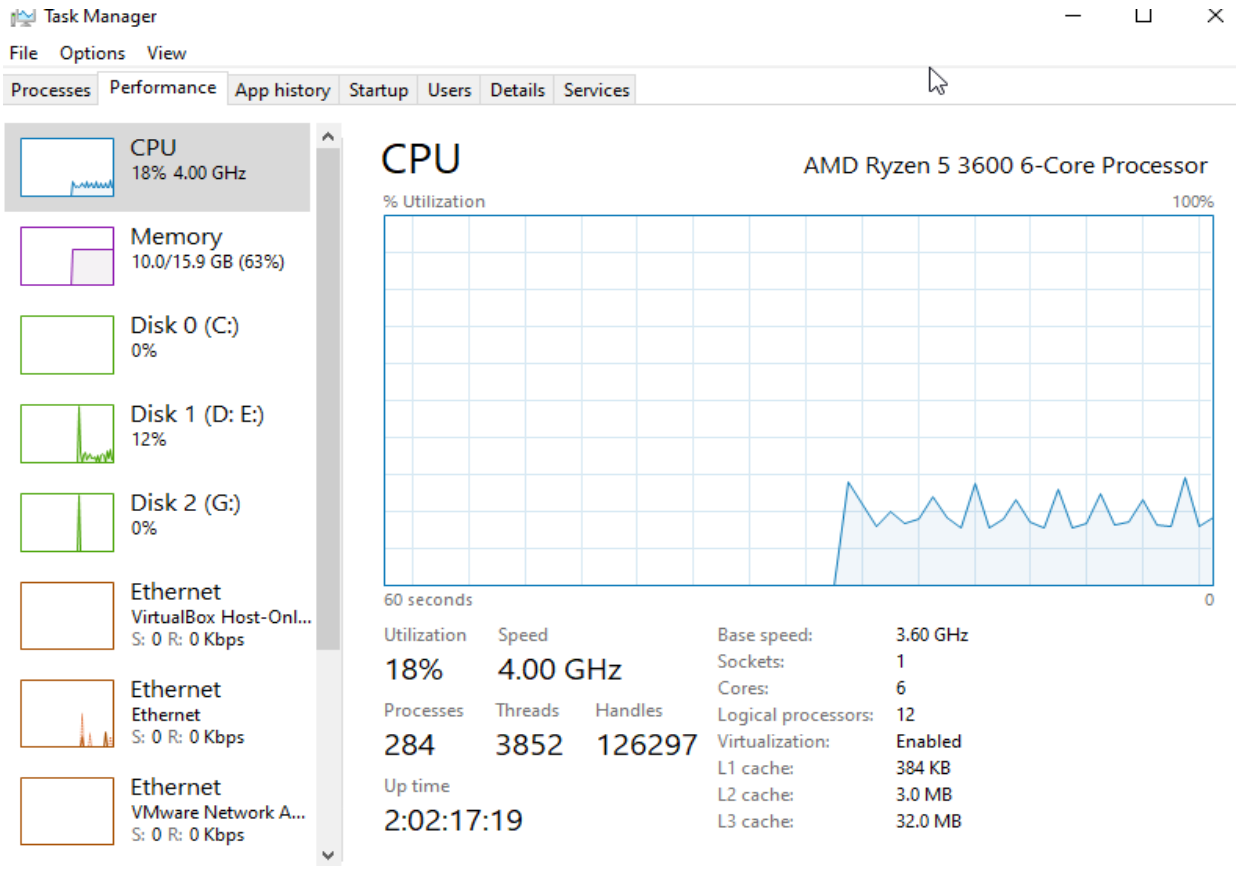



Figure 7.8: CPU Performance at the time of working

## 6.4 Developing Tools

- Python Environment (Virtual environment used)
- Jupyter notebook

# CHAPTER 6

# CONCLUSION & FUTURE WORK

**7.1 Conclusion:** In this paper we have discussed about data preparing, data processing, Test and Train methodology, CNN and its layers. Our model is so simple but we have got a better accuracy. At the end I can say using CNN any image processing system can be fruitful and give the efficient result.

**7.2 Future work:** In future anybody can work with our dataset because we are creating a csv test dataset, filtered dataset by which more progress of that report could be done. From the images any researchers can be helped for his research progress because we prepared our dataset sorted. We have used CNN but if anybody want to use more strong approach to apply it can give more accuracy.

## REFERENCES

[1] M. T. Habib, A. Majumder, A. Z. M. Jakaria, M. Akter, Md. S. Uddin, F. Ahmed "Machine vision-based papaya disease recognition". Journal of King Saud University – Computer and Information Sciences (2018), Volume 32, Issue 3, Pages 300-309, March 2020,

[2] Um, T.T., Babakeshizadeh, V. and Kulić, D., 2017, September. Exercise motion classification from large-scale wearable sensor data using convolutional neural networks.IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE. (pp. 2385-2390), 2017

[3] An in-depth exploration of automated jackfruit disease recognition Md. Tarek Habib, Md. Jueal Mia, Mohammad Shorif Uddin, Farruk Ahmed, page no 343- 353, 5 May 2020.

[4] Automated carrot disease recognition: computer vision approach by Anup Majumder, Md. Tarek Habib, Papiya Hossain Lima, Saifuddin Sourav, Rabindra Nath Nandi, 7 (4), 5790-5797, 2018.

[5] Computer Vision Based Local Fruit Recognition Md. Robel Mia, Md. Jueal Mia, Anup Majumder, Soummo Supriya, Md. Tarek Habib,International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-9 Issue-1, October 2019

[6] Islam, M.S., Foysal, F.A., Neehal, N., Karim, E. and Hossain, S.A., 2018. InceptB: a CNN Based classification approach for recognizing traditional Bengali games. Procedia computer science, 143, pp.595-602, 2018.

[7] Recognition of Paddy Plant Diseases Based on Histogram Oriented Gradient Features, International Journal of Advanced Research in Computer and Communication Engineering Vol. 5, Issue 3, March 2016.