Industry Oriented Mini Project Report

On

PLANT GROWTH ANALYSIS

Submitted in partial fulfillment of the requirements for the award of Degree of

Bachelor of Technology

In

Computer Science and Engineering

By

LAKKOJU MADHU	**(17241A05F3)**
KARTHIK CHALAMALASETTI	**(17241A05D2)**
KIRAN TALUKA	**(18245A0532)**

Under the Guidance of

MRS. S. BHARGAVI LATHA

Assistant Professor

Department of Computer Science and Engineering

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Autonomous)

Bachupally, Kukatpally, Hyderabad-500090.

2019-2020

ABSTRACT

Many plants have been unknown to present generation men. Sometimes, we think that a plant growing in our garden without our interference is a weed and an un-useful one. But there lies the problem. What if we could know what type of plant it is and what growth phase it is in?

We could use it to help us in our lives. We could take care of the plant well and get the most utilization out of it. There is a lot of Information about billions of plants around us, in a lot of books by many botanists. What if we could use that data and make people explore it using an application which helps us find out about the plant it is and understand many unknown insights about it. All this data gets out to be a very useful and vulnerable resource to humans.

We are able to say that a plant is growing well and its corresponding growth phase by perceiving with our eyes and knowing about that plant in prior. The same can be implemented using Image Processing which is similar to perceiving with eyes and using machine learning which is similar to the prior knowledge we have.

CHAPTER 1
INTRODUCTION

1.1 Introduction

Plants being the greatest resource of many utilities to nature, are to be taken care of. Plant growth has been a fascinating study to many biologists and common people. We could analyze the plant growth and using the latest technology, we could also grow plants with care. Generally most of us try to grow plants but we may or may not be knowing the specifics and uses of the plant completely. Moreover we don't exactly know how a specific plant species transforms in their growth.

Many plants have become extinct due to changes in their growth expectancies and habitual conditions. We could actually save the plants we are growing now. We have got a lot of data about each plant species, their growth pattern, their reproductive strategies and many more. This could be used along with various technologies used in the present day to ascertain their growth pattern and escalate their growth rate. This data generated could be used to make robots nurture plant growth in the future without the influence of humans.

1.2 Domain Description

The Project solely depends on two main Domains, which are Image Processing and Machine Learning.

1.2.1 Digital Image:

An Image is a Two - Dimensional Function or Matrix F(x, y) where x and y are regarded as spatial coordinates, along with an amplitude F at a coordinate (x, y). This Function denotes the Image Intensity at that particular (x, y). This kind of representation is called a Digital Image. A Digital Image is said to be composed of a finite number of elements called Pixels. A Pixel is an Intensity of a position at a particular coordinate value. Numerous Pixels combine to form an Image which we are able to perceive through our eyes. Each Image might constitute a different range of pixel count.

Some of the Image Resolution formats are

0.3 Megapixel	(640 x 480)
1.3 Megapixel	(1392 x 1024)
2.0 Megapixel	(1600 x 1200)
3.3 Megapixel	(2080 x 1542)
5.0 Megapixel	(2580 x 1944)
6.0 Megapixel	(2816 x 2112)
8.0 Megapixel	(3264 x 2448)
12.5 Megapixel	(4080 x 3072)
32 Megapixel	(6464 x 4864)

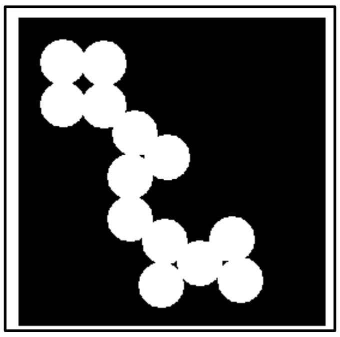
Where (An x B) denotes the size of Image in Pixels count in a 2 D Array Format.

1.2.2 Types of Images:

Image could be categorized in these different formats which define the Image storage in the Database thereby denoting the Two Dimensional Function.

1. **BINARY IMAGE**

A Binary Image uses a 2 - Label representation for denoting the pixel elements for an Image. The Labels are 0 and 1 which refers to Black (0) and White (1) colors only. This is called a Monochrome, which either makes a pixel element completely Black or completely White.



These Binary Images are stored as Bit Maps which are a very tightly packed array of bits. This type of Image needs a very less space. These Binary Images compress well and work well with different algorithms.

Fig: 3.1 - Binary Image

2. **BLACK AND WHITE IMAGE / GRAYSCALE IMAGE**

The Image consists of Black and White color only. In this Image, a pixel is a sample that holds only an intensity value that ranges from a predefined set of values. The Intensity denoted by each pixel denotes the value of Amount of Light that pixel carries. This Intensity contrast ranges from Black (Weak Intensity) to White (Strongest Intensity).

These also can be regarded as One Bit Bitonal Black- and - white Image. This might also be seen as Bi - Level Image. This Image has different shades of Gray ranging between White and

Black.

Fig: 3.2 - Gray Scale Image

3. **8 BIT COLOR FORMAT IMAGE**

As the name suggests, the Colors are 8 bit as stored in computers, where three bits of Red, three bits of Green and two bits of Blue. This Format of Image holds a range of 0 to 255 for each pixel. In this Image type, The Colors are regarded as Numbers from 0 to 255.

For Example, 0 is Black, 255 is White, 127 is Gray, etc.

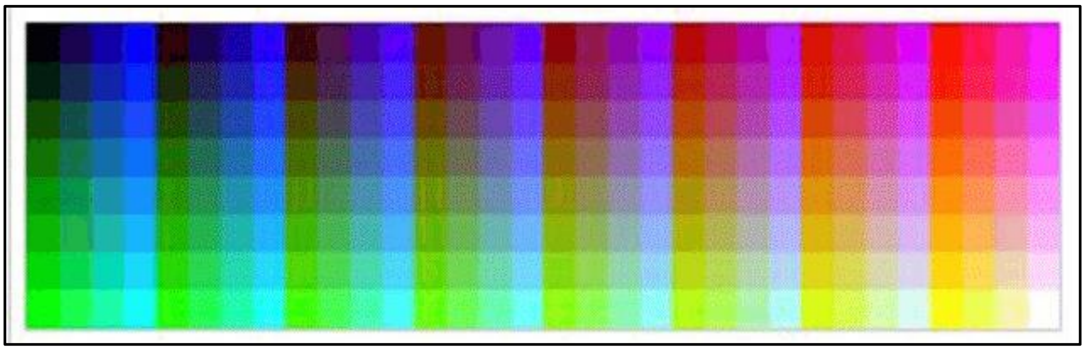


Fig: 3.3: 8 Bit Color Format Image

This format of Image also requires a low amount of Memory and since this format is similar to the Computer background byte type, it is used in many different applications like in Personal Computers, Gaming Consoles. To use the full potential of this format, color maps are used.

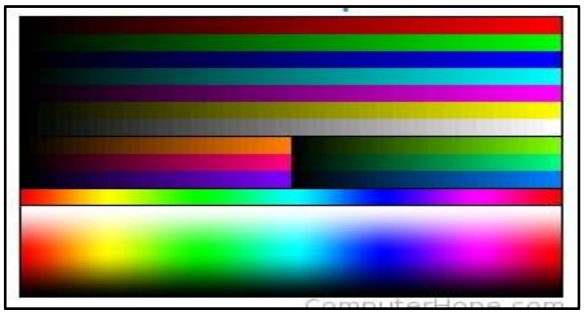


Fig: 3.4: 16 Bit Color Format Image

4. 16 BIT COLOR FORMAT IMAGE

A 16 Bit Color Image Format is a peculiar Image file which supports more colors than 8 bit color format. This Image file format can support 65,536 different colors. This can be represented as RGB (Red, Green, and Blue) format

Each Red, Green, Blue colors are assigned a set of bits from an Image Pixel. Here, 5 bits are used for Red Color, 6 Bits are used for Green and 5 Bits for Blue Color. This enables an Image with High Color variations and can be used for various high end applications.

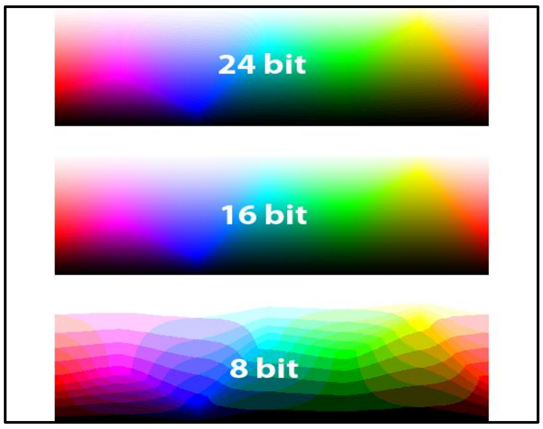


Fig: 3.5: Pixel Format Comparison in different Image Types

5. **24 BIT COLOR FORMAT IMAGE**

24 Bit color format image is also known as True Color Format. This is similar to 16 bit Color Format. In this format, the 24 bits are equally distributed among the Red, Green and Blue Colors. i.e., each color has 8 bits in each pixel. The range of colors possible to support is 0 - 2^24. This has three different matrices each for R, G, and B.

1.2.3 Image Processing

A profound technical definition of Image Processing defines it this way, “Analysis and Manipulation of Digitized Images to Enhance Quality and obtain Specifics from the image which is useful in some way “.

Image Processing is a procedure to enhance a digital image in its quality and improve the readability of the image and use it to draw some useful information that could be used at any instance for any purpose.

Image Processing allows people to make some readings from the image and using a program, can use that insights for further implementations in the real world. These insights of an image can be noted and can be used with Machine Learning and Neural Networks to make predictions and alterations needed in order to create self-learning systems.

Image processing has been used in innumerous applications in recent times. This has been growing exponentially from a few years now. This has got wide applications in many technical areas.

1.2.4 Machine Learning

Machine Learning is a way of predicting some values based on its previously obtained or recorded data and compute the future happenings where a pattern or a similarity in the previous data exists.

Machine Learning is a method of data analysis that can be used to automate or assess an Analytical Model. This is a branch of Artificial Intelligence which explains the idea that says that systems can be made to learn from data, identify similarities, formulate patterns and make decisions with minimal Human Intervention after deployment in the respective Industry.

Machine Learning is implemented along with many different technologies like Data Mining, Image Processing, Text Processing, and many more. This has even got a lot of attention to predict different categorical data by many means possible. All the sectors are now using Machine Learning to improve their businesses and to foresee their position with abundant data.

Machine Learning Algorithms can be broadly classified into 4 categories.

1. Supervised Machine Learning
   This is a process of predicting a given data using a previously analyzed, tagged and validated data. In this process, the data taken in the prior is called Training data which contains some data which is labelled with the value to be predicted by the algorithm and this prior data can be used to predict a set of unforeseen data. This is highly accurate for those inputs which are in the range specified by the Labelled data. Various kinds of regressions and classification models fall under this category.

2. Unsupervised Machine Learning
   A process in which the Data set used for training does not need labels. The algorithms learn about the data on their own. This exerts more computational overhead due to innumerous complex Data manipulations. This comes handy when there are unknown data patterns hidden in the data. This category requires minimal human intervention than the supervised version. Data Clustering is a major part in this type of unsupervised category. The accuracy depends on the data and also the algorithm one chooses.

3. Semi Supervised Machine Learning
   Semi Supervised, as the name suggests is not a supervised or unsupervised algorithm completely. This is a combination of the two kinds. The training data might contain labelled and unlabeled data as well which drive to the conclusion of a prediction.

4. Reinforcement learning
   Reinforcement learning is most of a Trial and error type. This type of algorithm generally learns by making predictions and actions, on calculating errors and re modifying its model iteratively until a most accurate fit is found.

1.3 Application

The Project reserves its application to the detection of the plant family and the corresponding Plant parts' Growth Phase. This system could be used as a backend for various applications that can be used to obtain the plant species or to understand the plant requirements to help the growth commence fruitfully.

1.4 Input Output Design

The Input output format to be designed is very simple. We design a system which takes an image of a leaf and displays the plant's species and its phase of growth along with some data about that plant.

CHAPTER - 2
SYSTEM ANALYSIS

2.1 Objective of the Project

The Main Objective of this project is to attain patterns in the growth of plant parts of different species with predefined set of conditions. The Main Plant part which is very much required by the plants is Leaf. Leaves show differences in their Color, Texture and size while their growth is at a stretch. The Young Leaves to the Adult leaves are targeted and using three labels, we predict the Growth phase.
This application derives an underlying pattern in the leaf growth and can be used to predict the growth phase. Thereby helping the Gardener to take some reasonable cautions to reserve his/her flora.

2.2 Problem Statement

The System should derive the patterns in the leaf growth and derive their growth quotient. Developing a Machine Learning based Application which takes the Image of a leaf and derives the growth phase in which that leaf is in. The Image produced by the user is supposed to tell whether the leaf is in Vegetative phase, Young phase or Adult Phase depending on the insights of the plant part.

Existing System

Generally there are no automated systems that analyze the plant growth and act accordingly. But we do have some information about the plants in books written by Botanists.

Limitations:

1. Inability of laymen to Understand terminology of scientists.
2. No Automation happened in this area.

Proposed System

This System uses Image Processing and Machine Learning together to predict the plant part growth phase.
In this, an application is created which is based on Image processing and Machine Learning using Python. This can be used to record data from the plants grown by the experts and draw insights from this data.
This can be used by a user who provides the image and expects the plant species and the growth phase the plant part is in and also gets some data about its growth and suggests some measures that help the plant grow well in the given conditions.

Scope of the Proposed System

1. The User can use this system to identify the Plant Species family and the current plant part phase.
2. This System upon integrating with much more methods of accurate predictions, can be used to predict much more different species by training with abundant amounts of data and images.
3. This system, once created can be used for any number of species by training the model.

2.3 Feasibility Study

Financial Feasibility

Being a machine learning application, it will have no associated hosting cost. Since the system does consist of a multimedia data transfer, therefore the bandwidth required for the operation is optimal, but not that high as the input required is just a single image per connection. The system will follow with all freeware software standards. No cost will be charged from the potential customers. Bug fixes and associated risks will have an associated cost.
Concatenation of several modules in both the Image processing and Machine Learning, makes it beneficial for the customers to instantly identify the classification of several plants without any physical need.

Technical Feasibility

The Plant growth Analysis project is a complete client server based machine application using image processing. The main technologies and tools that are associated with Plant Growth Analysis are:

- Python
- Tkinter

These are freely available and are manageable. The time limitations of the front-end development and the ease of implementing using these technologies are synchronized.

Resource Feasibility

Resources required for the project include

- Programming Device (Laptop).
- Hosting Data Storage space
- Programming Tools(freely available)

Social / Legal Feasibility

As the Plant Growth Analysis uses only freely available development tools, and provides the system as an open source system. Only the maintenance cost in updating the database will be charged from the potential customers.
The python software libraries that are used in this system are free software libraries.

2.4 Software / Hardware Requirements

The System requirements can be classified into categories broadly as Software requirements and Hardware requirements. These satisfy all the needs of the system proposed.

2.4.1 Hardware Requirements

- **CAMERA Equipment**
  Camera is the most essential requirement which serves us in picturing the plant parts for Image Processing.
- **DATA STORAGE Equipment**
  The Images produced are to be structured and stored in order to maintain the data structuring and lessen the effects of losing data insights.

2.4.2 Software Requirements

- **PYTHON Based Modules**
  Some Modules like cv2, numpy, pandas are used.
  - **CV2**
    CV stands for Computer Vision. This Python based CV2 module holds different methods for performing Image Processing. This is used for various Image pre-processing techniques and for drawing insights from the digital Image being considered for drawing insights.
  - **NUMPY**
    Numpy module is a python based module which helps in calculating with high precision. This module helps in manipulation of data in the form of 2D Arrays, which are the base for the digital images.
    This has N-D Array Objects for Data Manipulations, this has sophisticated methods useful in many different Mathematical and scientific applications like Random Numbers, Algebraic Operations, Transforms, etc.,.
  - **PANDAS**
    Pandas module is a python based module which helps in data storage and large data manipulations upto 1GB. This is used for efficient structuring the data in an optimized way. This works similar to the relational databases. The data is stored in the form of Tables and matrices.
  - **SKLEARN**
    The Sci-Kit Learn Module is a machine Learning based module which provides Implementations of various Machine Learning Algorithms along with Accuracy, Precision calculating metrics. This module supports scientific and numerical libraries like Numpy and Scipy. Some of the algorithms implemented in this Library are Random Forests, K Nearest Neighbours, Support Vector Machine, SVR, Decision Trees, Spectral clustering, Mean Shift, K- Means, Principal Component Analysis,etc.,.
    Some of the metrics are Cross Validation, PCA, ICA, LDA, etc.
- **PYTHON DEVELOPMENT**
  The Main course of the Project is based on Python. We use the Modules for Processing the images and Data Modelling.
  This could be satisfied by using Spyder IDE, Jupyter Notebooks, etc.
- **GUI**
  The Front End of the project has to be developed using Tkinter for Python. This enables us to add Graphic related widgets for better User experience.
- **SOCKET**
  The Socket Module in the python library is used to develop the client server implementation of the System. This enables us to create a virtual server and a virtual client which can be run parallely on the same device or different devices.

2.5 Schematic Diagrams

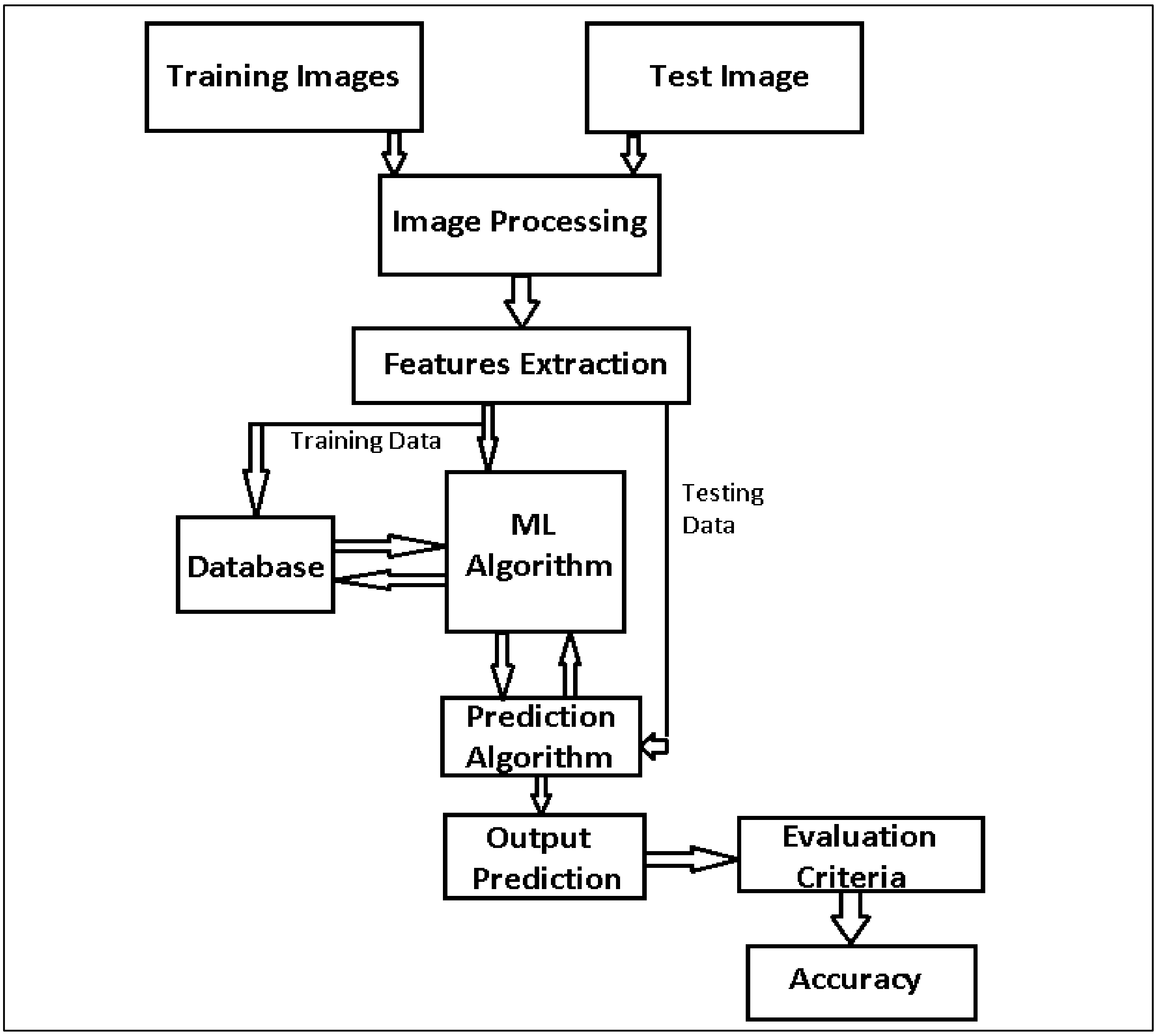


Fig 2.5.1 Project Design Block Diagram

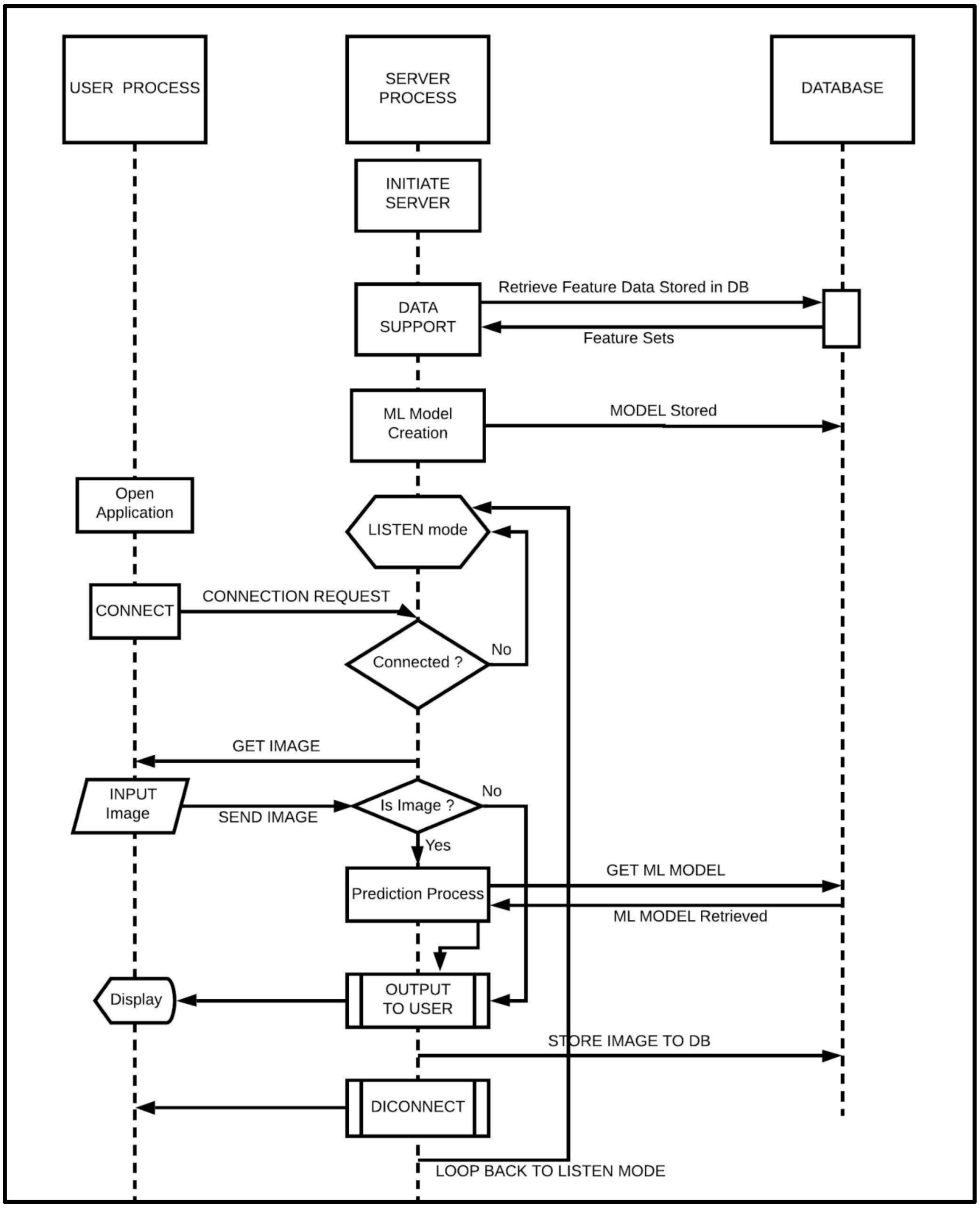


Fig 2.5.2 Sequence Diagram

The Sequence Diagram explains the way the components are interconnected to each other and satisfy the user request.

This Sequence can be divided into the following parts to demonstrate their operational behavior

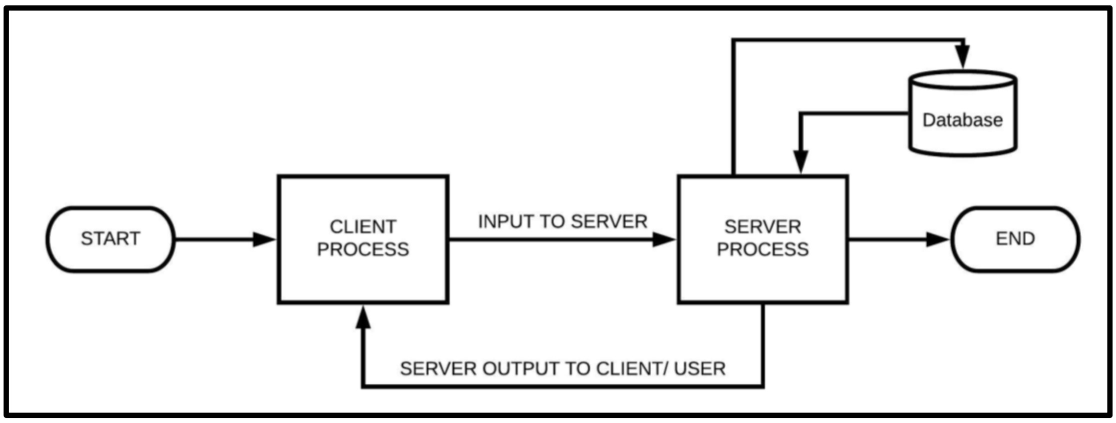


Fig 2.5.3 Client Server Implementation

This module explains the Client server relation in the System. Here, the client system is not given any kind of massive overheads of large computations. The Server takes the Image from the client / user and sends back the predicted output to the corresponding input.

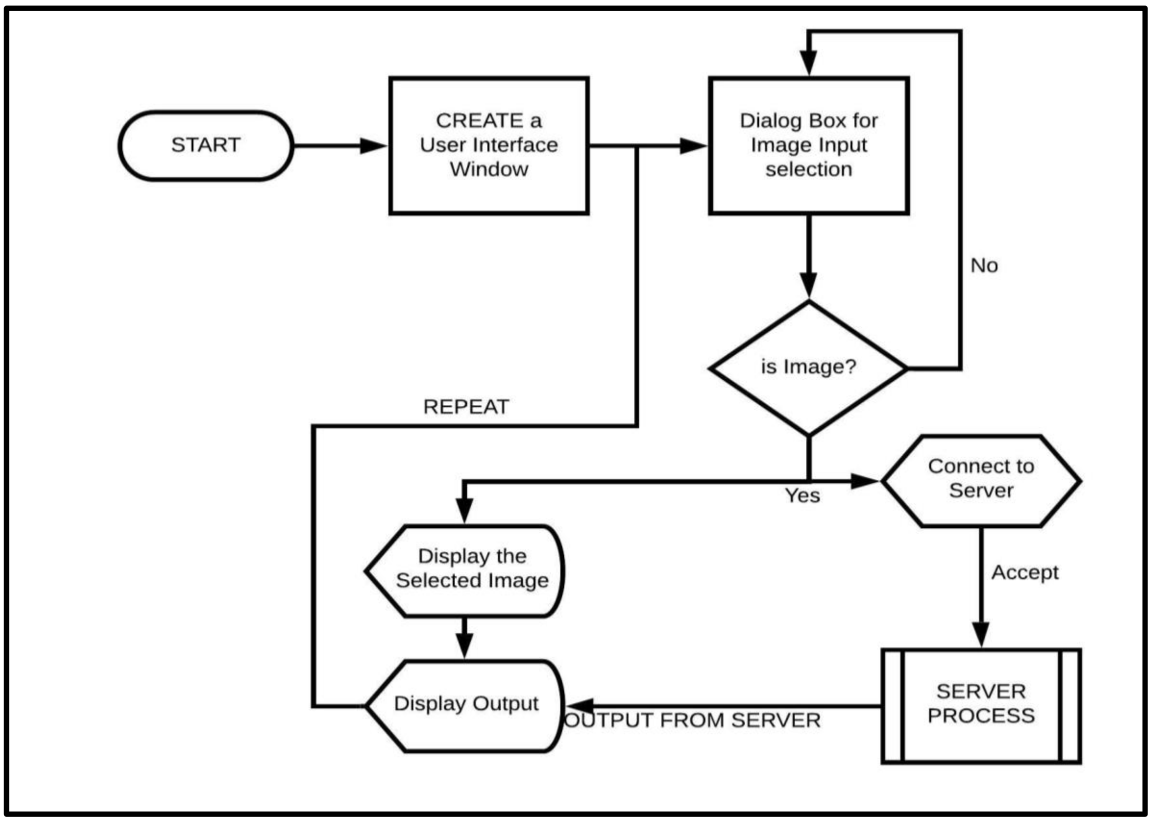


Fig 2.5.4: Client Implementation

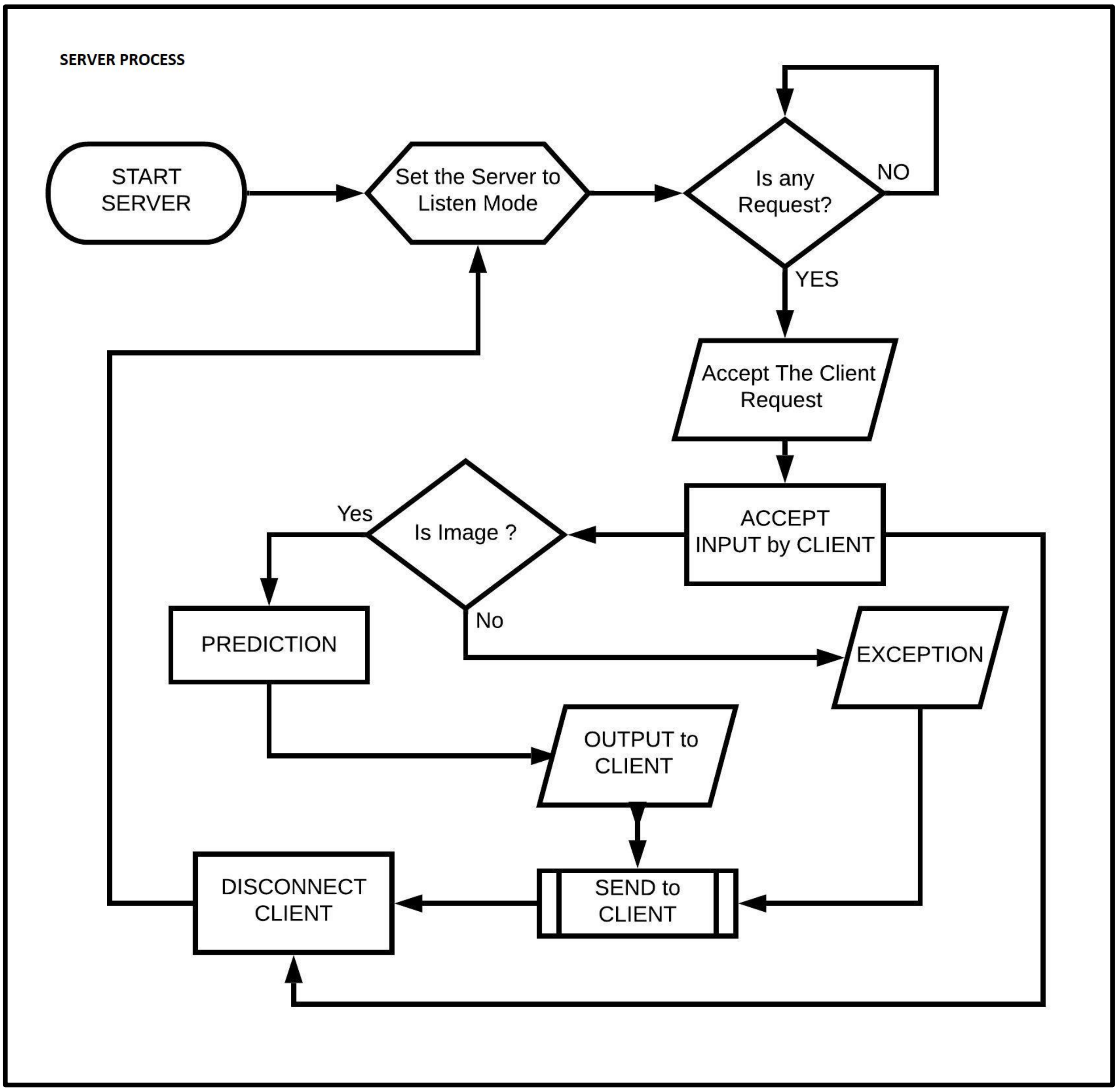


Fig 2.5.5: Server Implementation

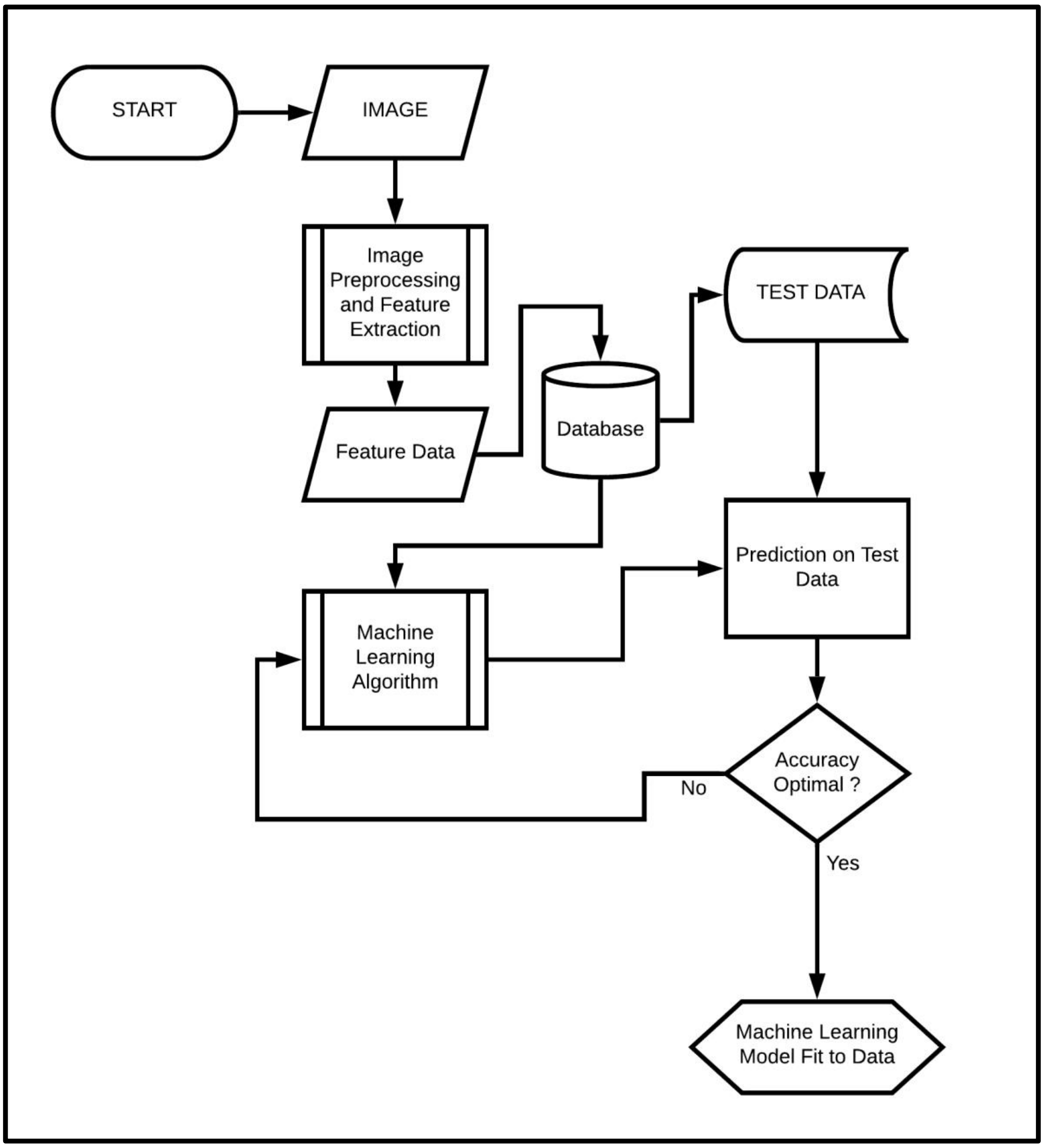

START
IMAGE
Image Preprocessing and Feature Extraction
Feature Data
Database
TEST DATA
Machine Learning Algorithm
Prediction on Test Data
Accuracy Optimal ?
No
Yes
Machine Learning Model Fit to Data

Fig: 2.5.6 Machine Learning Model

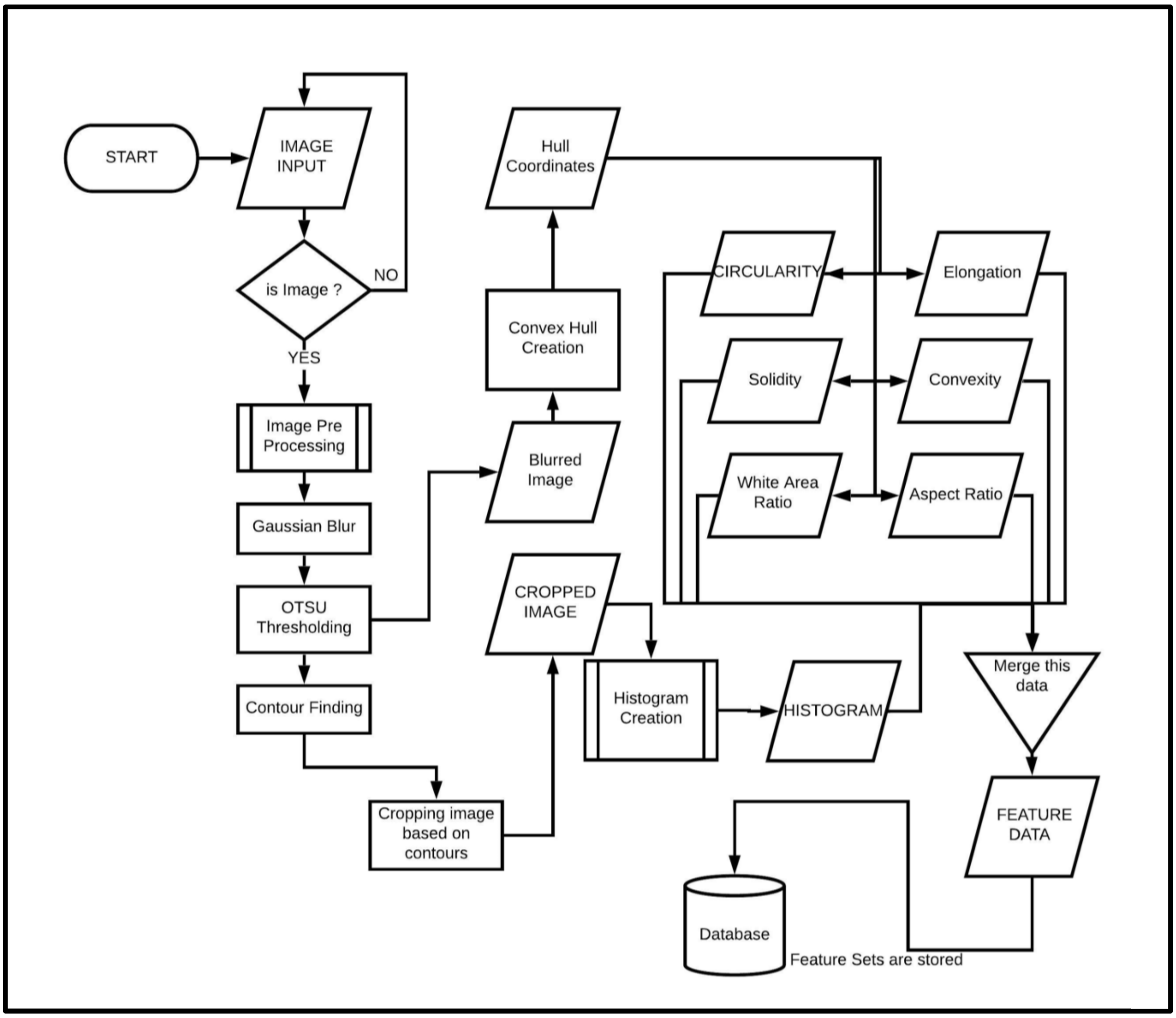


Fig 2.5.7: Image Pre Processing and Feature Extraction

CHAPTER - 3
IMPLEMENTATION

3.1 ARCHITECTURE

The Project architecture can be briefly described as a Client server based architectural pattern where a small amount of text data is stored at client device and a very small amount of data is being made to pass over the network.

3.2 MODULES DESCRIPTION

The Project has been broadly categorized into three modules.

1. Image Pre Processing and Feature Extraction
2. Machine Learning Model based on Feature Data
3. Front End and Data Storage.

3.2.1 Image Pre Processing and Feature Extraction

Images being the main backbone of the project, there is a need to draw the insights from the Image data. This Module takes care of the Image pre Processing and Feature Extraction part. Each and every Image, either a Training Image or a Test Image has to undergo through this module. This Module can be internally classified into two major categories.

3.2.1.1 Image Pre Processing

The Images are taken through a sequence of steps which enhance the Image quality and improve the ailments that are required from the Image. This happens in the following way.

1. **Digital Image Acquisition**

Digital Image Acquisition is a process of developing a digitally encoded virtual representation of the Objects. This can also be termed as **Imaging**. This defines the format of the Image, Image storage type, etc. This can be simplified to this, "To Transform a Real World View / an Object into an array of Intensity values which define the Image at different spatial coordinates. This is mostly automatic these days. Nevertheless, if we are to create a new Image using the traces of another image for internal purposes, this becomes a helping hand.

2. **Reading and Resizing Images / Compression**

The Images obtained after acquisition are to be resized according to the system capacity. The Image is read as a 2D array of varying Intensity values which is resized prior to the processing. This helps in faster access to the Image properties and a speeder manipulation of Image insights.

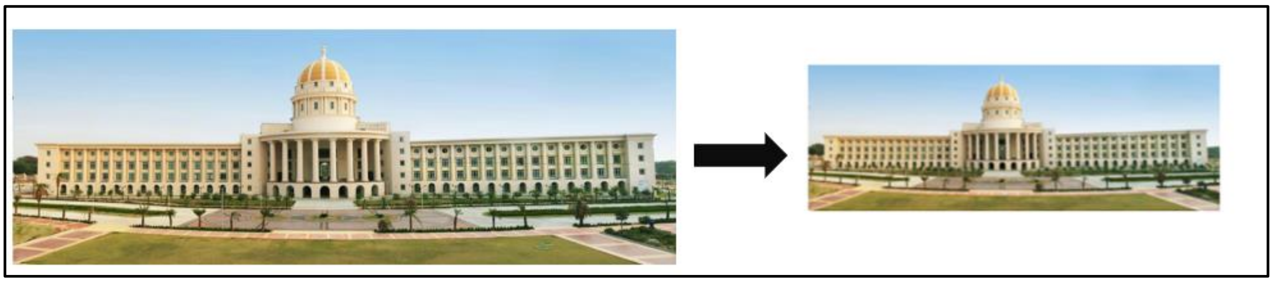


Fig 3.1: Image Compression

3. **Denoising Image**

   The Images resized are to be denoised in order to reduce complexity in the image and enrich the image with accurate values of intensity in each pixel. This stage smoothens the image and reduces sharpness of the image. This is similar to blurring an image or viewing an image with a translucent screen in between.

   1. Opening Operation
   2. Closing Operation

   These Opening and Closing operations serve the purpose of Denoising. These are especially required in the Field. The images might contain various types of noises. They might be an unwanted minute particle or a shadow of the curves on the surface of the object to be imaged or any other stuff.

Morphological Operations:

Opening Operation:

The Opening Operation goes through 2 steps where we first erode the Image and then dilate the resulting image. This results in expulsion of protrusions in the main Image read or it can also be called as removing internal noise of the image

Closing Operation:

The Closing operation will be the reverse of the Opening Operation. This performs Dilation and then Erosion. The holes or depressions present in the Main Image will be eliminated or smoothened. Here, the Contours also called as the edges of an object will be smoothened along with narrow breaks in the image if any.

Erosion Operation:

The Erosion operation proceeds in such a way that the image is considered along with another empty image of a predefined fixed pixel size which is comparatively very less. This temporary image is mapped with each possible two dimensional sub array of the image. During this process, the output 2D arrays' value at (x, y) becomes the minimum value of the mapped sub array. Here, The Lower intensity value is prioritized. This eliminates spots and breaks.

Dilation Operation:

This operation is very similar to Erosion operation. The change which is seen is the consideration of the maximum value of the mapped pixel values. This prioritizes the maximum values of pixels. Here, the white light is much used. This eliminates the background of the image.

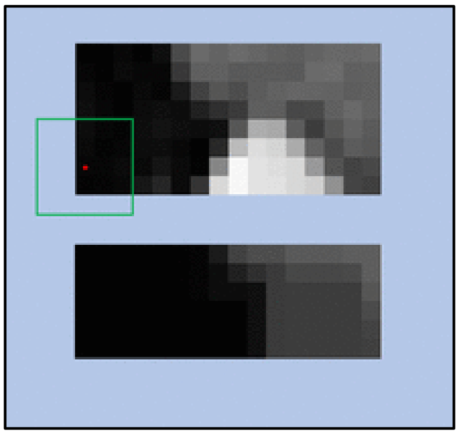


Fig 3.2 Erosion Operation

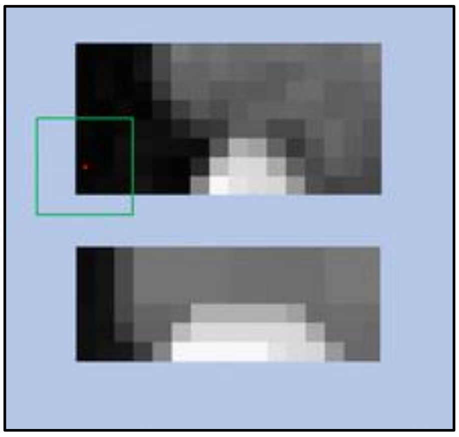


Fig 3.3 Dilation Operation

Pseudo code for Erosion Operation

```
Start Erosion operation
Create a black empty image of 5X5 size
Iteratively check for all rows in the Image considered
For Each Row and Column,
        Map this empty image on the Main Image
        Check for the minimum Intensity values in that mapped array
        Overwrite the column of the main image with this Maximum intensity
        Proceed to the next column in the same row
Finally return the modified image
End
```

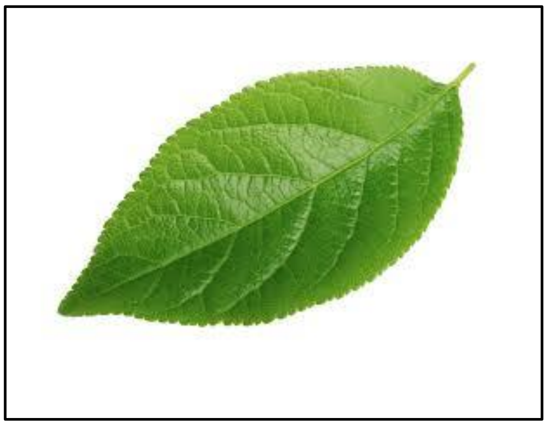
4. **Segmentation of Image**

Image Segmentation by name represents the separation of parts of an image based on some X factor. Segmentation is a base for finding out multiple objects in the Image at a single pass rather than checking the whole image for each and every object to be found from that image. This helps in separating the 2D array of images based on the pixel values whose place in the segmented image relies solely on the X factor we use.

The X factor said in the above discussion might vary according to our requirements. A very common method is called **Thresholding.** In this method, we define bounds in the image array and convert the image to Binary (Black / white only). There are different methods of defining Threshold values.

1. Maximum Entropy Method of Thresholding
2. Balanced Histogram Thresholding
3. OTSU method of Thresholding (depends on Variance)
4. K Means Clustering for Thresholding

Segmentation can also be done in some more other ways, Motion and Interactive Segmentation, Compression based Segmentation, Histogram based Segmentation, P-Tile thresholding, Clustering & Dual Clustering, Region



Growing Segmentation and Partial Differentiation based segmentation, Variational segmentation, Graph Partitioning segmentation, Model based, Semi-Automatic segmentation, Trainable segmentation (Neural Networks) and many more.

Fig 3.4: Original Image

Simple Thresholding:

The Simple Thresholding calculates the Global boundary/ Global Threshold value. The whole Images' segmentation depends on this Threshold value.

In the Simple thresholding we have different kinds of mappings like TRUNC, BINARY, BINARY_INV, TOZERO (to zero), TOZERO_INV (To zero Inverted), etc. Their methods are self-explained by their names.

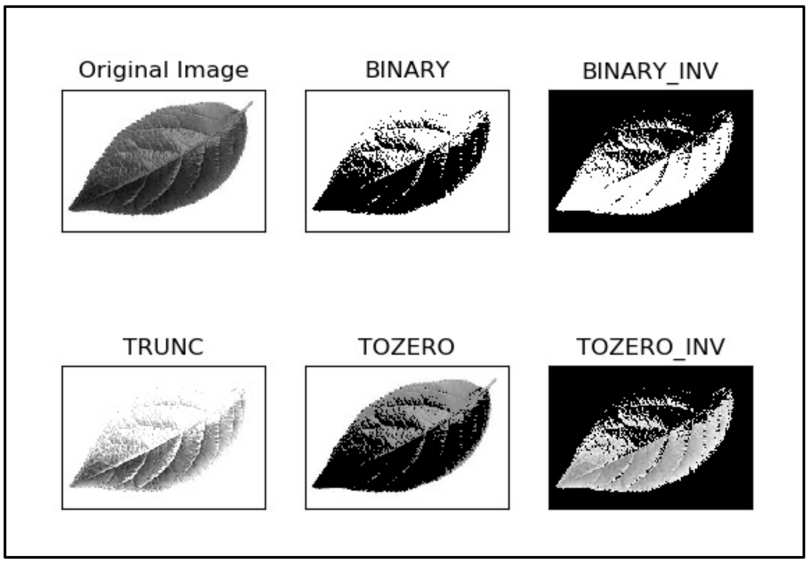


Fig 3.5: Simple Thresholding with various mappings

Adaptive Thresholding:

This type of Thresholding calculates the Threshold value to sub parts of the Image and this local threshold value is used in segmentation of the image. This type of segmentation helps in defining the sharp parts in the image as shown in the image. There are different methods with varied filtering approaches like Adaptive Mean Thresholding, Adaptive Gaussian Thresholding, etc.

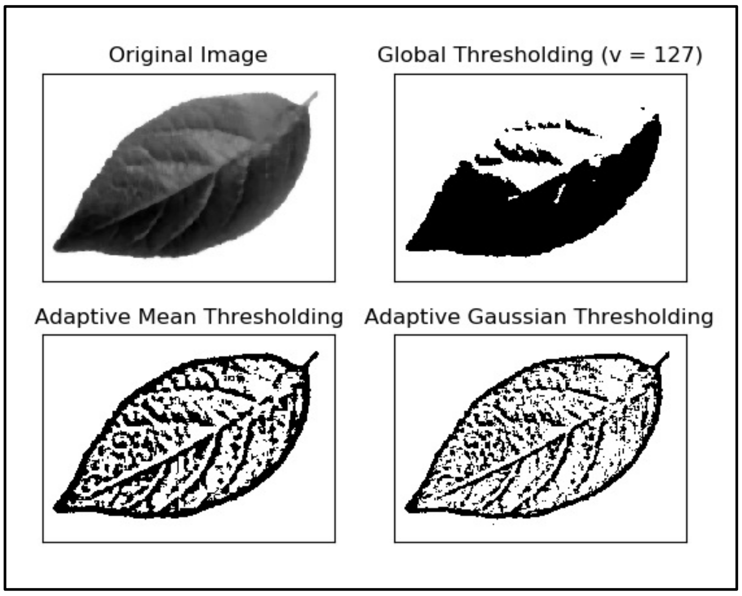

Original Image
Global Thresholding (v = 127)
Adaptive Mean Thresholding
Adaptive Gaussian Thresholding

Fig 3.6: Adaptive Thresholding and Types

OTSU Type Thresholding:

The OTSU type thresholding method follows the principle of finding the minimal spread in the histogram to the left and right of the Threshold value considered. This can also be defined in this way "The Threshold value considered must have **maximum Between Class variance** and **minimum Within Class Variance**". This yields much more precise Thresholding than others. This, upon use with different Filters, yields a pretty much accurate segmented image.

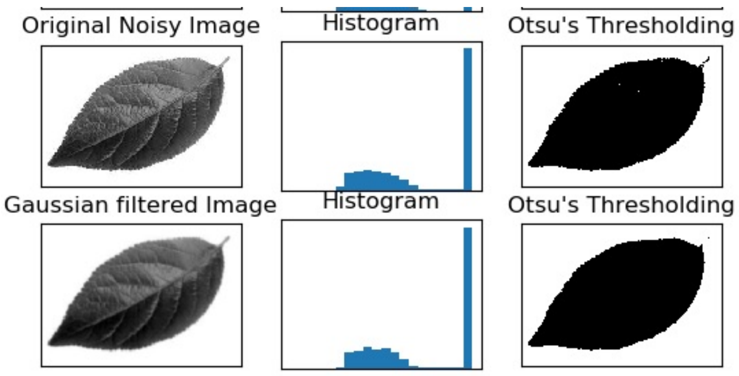


Fig 3.7: OTSU Type Thresholding

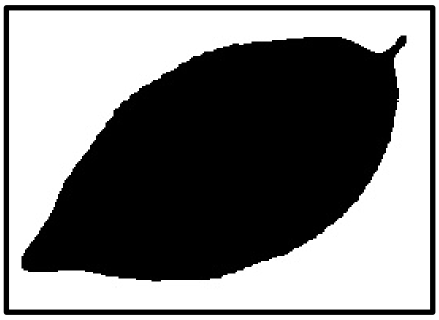


Fig 3.8: OTSU Threshold with Gaussian Filter

Pseudo code for Image segmentation

```
Start Segmentation
Accept the Image
Set the mode to OTSU and Binary
Create a Histogram from this Image
For each point in histogram range
        Calculate spread of left part in histogram
        Calculate spread of right part in histogram
        If these values are optimal till now
                Update optimal values
Return the optimal values
End
```

5. Boundary and Region Representation :

We now consider the segmented image and find the boundaries to the object in the image. The Image that segmented as per our requirement is taken and the Boundaries are found from this segmented image. These boundaries are marked where there is a drastic difference in the pixel- intensity value. These are also called as Contours. These mark our outline of the object.

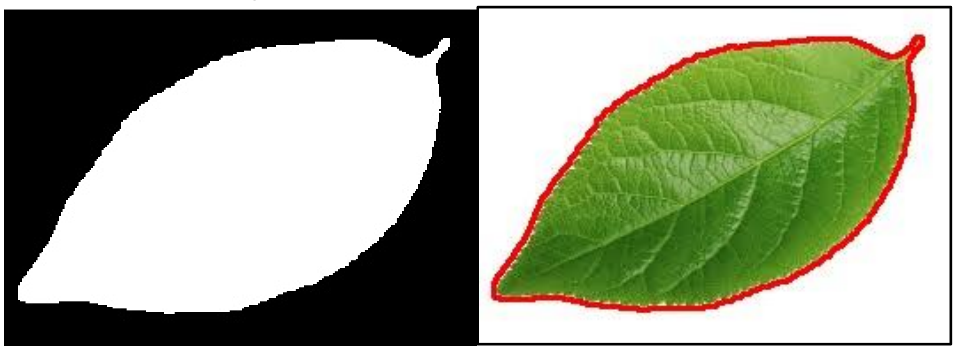


Fig 3.9: OTSU Inverted Image & Contours Marking in the Image

Here, The OTSU Threshold Image is inverted and the difference in pixels is mapped to an empty image and thus the Outline is implanted onto the Original image and thus the contours of the Object is found. These contours are used to the next step.

Pseudo code for Contours Finding

```
Start Contours Finding
Accept Image
For each pixel in the image
        If the pixel falls in the threshold values
                Mark the pixel as a contour coordinate
        Otherwise continue
Return Marked coordinates
End
```

6. **Cropping the Image for Object**

   We consider the Image with contours marked and draw out the extremes of the Contours from the Image object. These extremes of the object are used to crop the image so as to contain the whole object in the image. In our perspective, we should make sure that all the Image is contained in the cropped image.

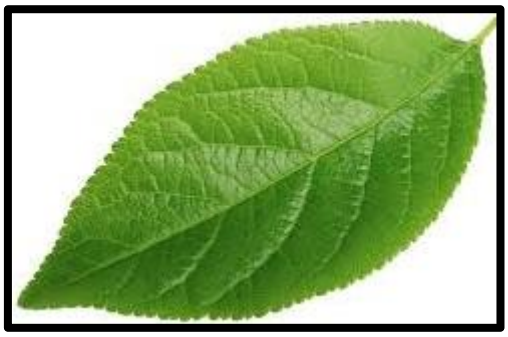


Fig 3.10: Cropped Image

7. **Histogram**

   The Cropped Image is used to pull out the Histogram of Intensity values to ascertain the Texture and color shade levels. These Histogram values represent a massive insight in the Image. A predefined number of bins is defined while finding the histogram so that there is a definite notion of varying intensities.

   Pseudo code for Histogram

```
Start Histogram
Create a Histogram List with 64 bins
Define ranges of 0 - 256 for 64 bins
For each pixel
    Check the intensity of pixel
    Add this to the corresponding bin
    Increment the bin count value
Return Histogram List
End
```

8. **Approximating the Shape of the Leaf / Object**

   The Contours/ borders of the object are considered and their values are approximated based on a predefined value. This reduces the complexity in the Contour points. This will be used to extract feature shapes and use it as a feature.

   Pseudo code for Convex Hull Creation

```
Start Hull Creation
Consider Contour points
Note the first point of contours
For each point in contours
    Check if there is a steep drop / increase in the distance or slope
        Note this point
        Add it to hull points list
    Otherwise continue
Return the hull list
End
```

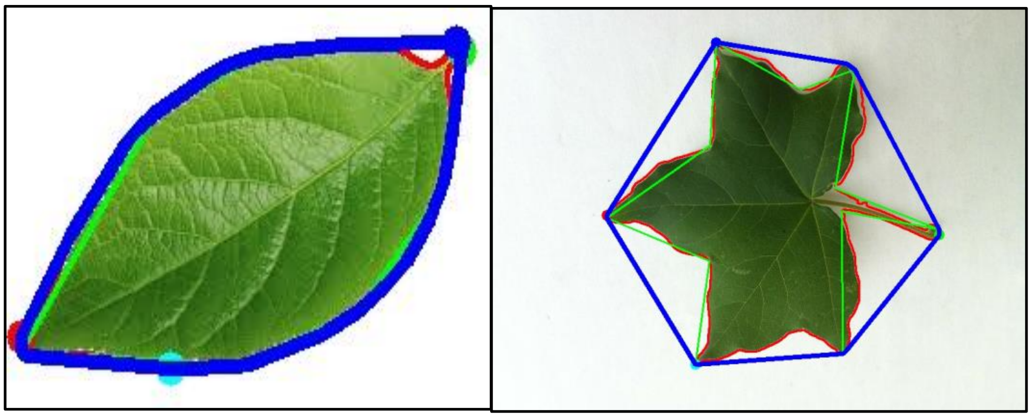


Fig 3.11: Hull Construction to Leaf

3.2.1.2 Feature Extraction

The Image is a vulnerable source of various features corresponding to shape and size. These features are used in constructing the Machine Learning Model.

Features pertaining to Shape of the Leaf:

1. **Aspect Ratio**
2. **White Area Ratio / Elongation**
3. **Perimeter to Area Ratio**
4. **Perimeter to Hull Ratio**
5. **Hull Area Ratio / Solidity**
6. **Circularity**
7. **Equivalent Diameter of leaf, etc.**

These features explain how the Leaf structure and shape are. These features can be used to differentiate between different family leaves and also can be used to find the growth phase of the leaf within the same species of plant.

1. **Aspect Ratio**

   Aspect Ratio is a Geometric ratio which defines a ratio of the object sizes of different dimensions. Generally the Aspect ratio is defined as the Ratio of Width to Height or the ratio of the objects' longer side to its shorter side. This ratio is of various kinds as per the requirement. We encapsulate the Hull into a Rectangular box and calculate its aspect ratio and use it as a feature.

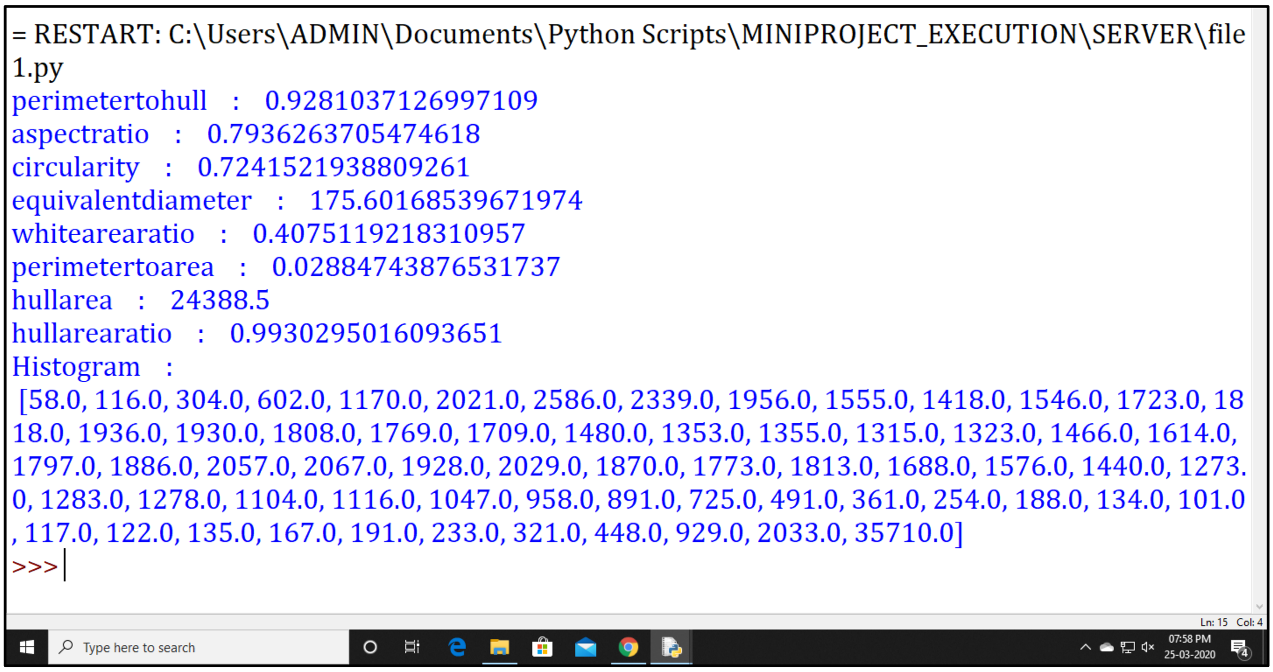
2. **White Area Ratio / Elongation**

   White Area Ratio is a ratio which pertains to the existing white area in the Image which is not a part of the object that is to be focused on. This ratio can also be called the Floor area Ratio. This is defined as the area of the object/ leaf divided by the area of the rectangular box encapsulating it.

3. **Perimeter to Area Ratio**

   The perimeter to area ratio measures the complexity hidden in the leaf curvature and shape. This is an application of Morphometric. This also defines the compactness in the leaf. As the name suggests, it is the ratio of the perimeter of the leaf to the area of the leaf calculated using the contours of the leaf.

4. **Perimeter to Hull Ratio / Convexity**
   The Convexity explains the difference between the hull and the main object in the image. If the hull is the same as the main object, the convexity values will be close to 1 otherwise it is a fraction. This can be defined as a ratio of Hull perimeter to the leaf perimeter.
5. **Hull area Ratio / Solidity**
   Solidity also called the Hull area ratio defines the density of the object. This is the ratio of the areas of leaf and the hull. As the hull is the approximated structure which encapsulates the object, it has slightly extra white space. This ratio helps in figuring out what percentage of the whole hull is the leaf.
6. **Equivalent diameter**
   The Equivalent Diameter is the length of a line which extends between the extremes in a leaf. This denotes the length of the leaf. This is calculated based on the circularity of the leaf as the Square root of a 4 times leaf area divided by pi.
7. **Circularity**
   The Circularity defines the roundness in the image object. This defines the exterior shape of the leaf and this is helpful to ascertain a shape feature which is very much useful in differentiating leaves in species.



```
= RESTART: C:\Users\ADMIN\Documents\Python Scripts\MINIPROJECT_EXECUTION\SERVER\file
1.py
perimetertohull  :  0.9281037126997109
aspectratio  :  0.7936263705474618
circularity  :  0.7241521938809261
equivalentdiameter  :  175.60168539671974
whitearearatio  :  0.4075119218310957
perimetertoarea  :  0.028847438765317 37
hullarea  :  24388.5
hullarearatio  :  0.9930295016093651
Histogram  :
 [58.0, 116.0, 304.0, 602.0, 1170.0, 2021.0, 2586.0, 2339.0, 1956.0, 1555.0, 1418.0, 1546.0, 1723.0, 18
18.0, 1936.0, 1930.0, 1808.0, 1769.0, 1709.0, 1480.0, 1353.0, 1355.0, 1315.0, 1323.0, 1466.0, 1614.0,
1797.0, 1886.0, 2057.0, 2067.0, 1928.0, 2029.0, 1870.0, 1773.0, 1813.0, 1688.0, 1576.0, 1440.0, 1273.
0, 1283.0, 1278.0, 1104.0, 1116.0, 1047.0, 958.0, 891.0, 725.0, 491.0, 361.0, 254.0, 188.0, 134.0, 101.0
, 117.0, 122.0, 135.0, 167.0, 191.0, 233.0, 321.0, 448.0, 929.0, 2033.0, 35710.0]
>>>|
```

These Shape features explain the size and shape of the Leaf.

Fig 3.12: Features computed towards dataset creation

Feature pertaining to the Texture of the leaf

1. **HISTOGRAM**
   Histogram defines the Intensity values which are separated in the form of a predefined set of bins to approximate the colors closely. This defines the color range and color frequency of the leaf.

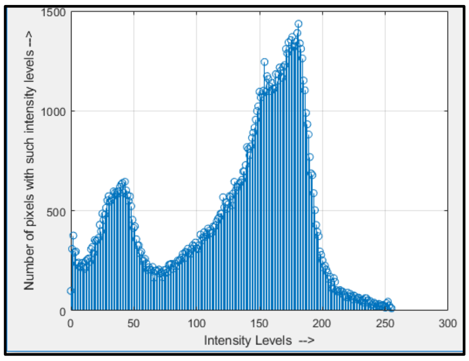


Fig 3.13: Histogram

All these features are noted and used in the prediction of the Species and phase of the plant.

Pseudo code for Histogram

```
Start Histogram
Create a Histogram List with 64 bins
Define ranges of 0 - 256 for 64 bins
For each pixel
        Check the intensity of pixel
        Add this to the corresponding bin
        Increment the bin count value
Return Histogram List
End
```

Pseudo code for Feature Extraction

```
Start Feature Extraction
Accept the Image Path
Read the Image from the path given
Perform Erosion operation
Blur the Image using the Gaussian Blur
Perform Image Segmentation on this blurred image
Obtain threshold values from segmentation
Obtain Contours using the threshold values
Find the Extremities of the contours
After all extremes are found, crop the Image
Draw and store the Histogram
Approximate contour values
Calculate Area, Perimeter
Use Contour values and draw a Convex Hull covering all contours
Calculate ratios required
Return these ratios
End
```

3.2.2 Machine Learning Model

Machine Learning models are used in prediction of labels which define a set of features individually/ uniquely. These Models create a buffer decision tree or a classifier model to predict the category where the corresponding image features fall into.
Various Models have been used for high precision and accuracy to the model and Majority of them are considered for evaluations.

1. **K Nearest Neighbors Classifier**
   K Nearest Neighbors Classifier uses the feature data of the large training data available and sorts the data such that the similarities in the data are prioritized and grouped together. This Classifier takes the input feature data of the image to be predicted and moves through the training data and finds the position where the feature set can be inserted such that the similarities are larger among the neighbors by computing distances to each feature data i.e., distance must be minimal.
   There are different kinds of distance functions that can be used to serve this purpose.
   1. Euclidean Distance
   2. Manhattan Distance
   3. Murkowski Distance
   4. Hamming Distance (for Categorical values)

2. **Naive Bayes Classifier**
   Naive Bayes Classifier uses the prominent Bayes theorem to predict the labels. As the name suggests this classifier uses naive and independent assumptions between the features of the dataset. This generates a Probabilistic model which guesses the probability of each label and returns the more likely label.
   This can also be used with some masks to better the estimation. Some of them are
   1. Simple Bayes Classifier and Independence Bayes
   2. Multinomial Naive Bayes
   3. Gaussian Naive Bayes
   4. Bernoulli Naive bays
   5. Semi Supervised parameter estimation. Etc.,

3. **Support Vector Machine**
   A Support vector machine is a Machine Learning model which uses the training data in plotting the coordinates regarding the data and fit a hyper plane so as to minimize the difference between each point and hyper plane. The orientation of this hyper plane depends on the data labels and also the size of the dataset. This is a discriminative type classifier.
   The different kernel kinds of SVM’s are
   1. Linear kernel with soft and Hard Margins
   2. Polynomial kernel
   3. Radial Bias Function Kernel / Gaussian Kernel
   4. Sigmoid Kernel

   The extensions defined to SVM are SVC (Support vector Clustering), Multiclass SVM, Transductive SVM, Structured SVM, Bayesian SVM, etc.

4. **Decision Trees**
   The Decision tree classifiers are highly adaptive and computationally effective. They use a tree to structure the given training data. This tree building is based on various comparisons made by the classifier while building the model. This tree uses Hunt's algorithm during this process. The underlying work is in such a way that the working data / train data is splitted into sub parts and used to create decision nodes.

   These are of 2 kinds
   1. Classification Trees
   2. Regression Trees

5. **Voting Classifier / Wrapper**
   All the above explained classifiers are of single hierarchy. There is no acceptance offered by some other person or classifier that negotiates with the classifier prediction. This has been masked in this classifier by using a vote method and returning the maximum valued label. This ensures lower error and less over fit by combining predictions of multiple machine learning algorithms and wrap them up in parallel and exclude the unwanted or minority type label.
   There are 2 kinds of Voting
   1. Hard Voting / Majority Voting
   2. Soft Voting
   3. Weighted Voting

   This classifier takes the models like K-NN, Decision tree Naive bayes, SVM, etc. as arguments and does voting among them.

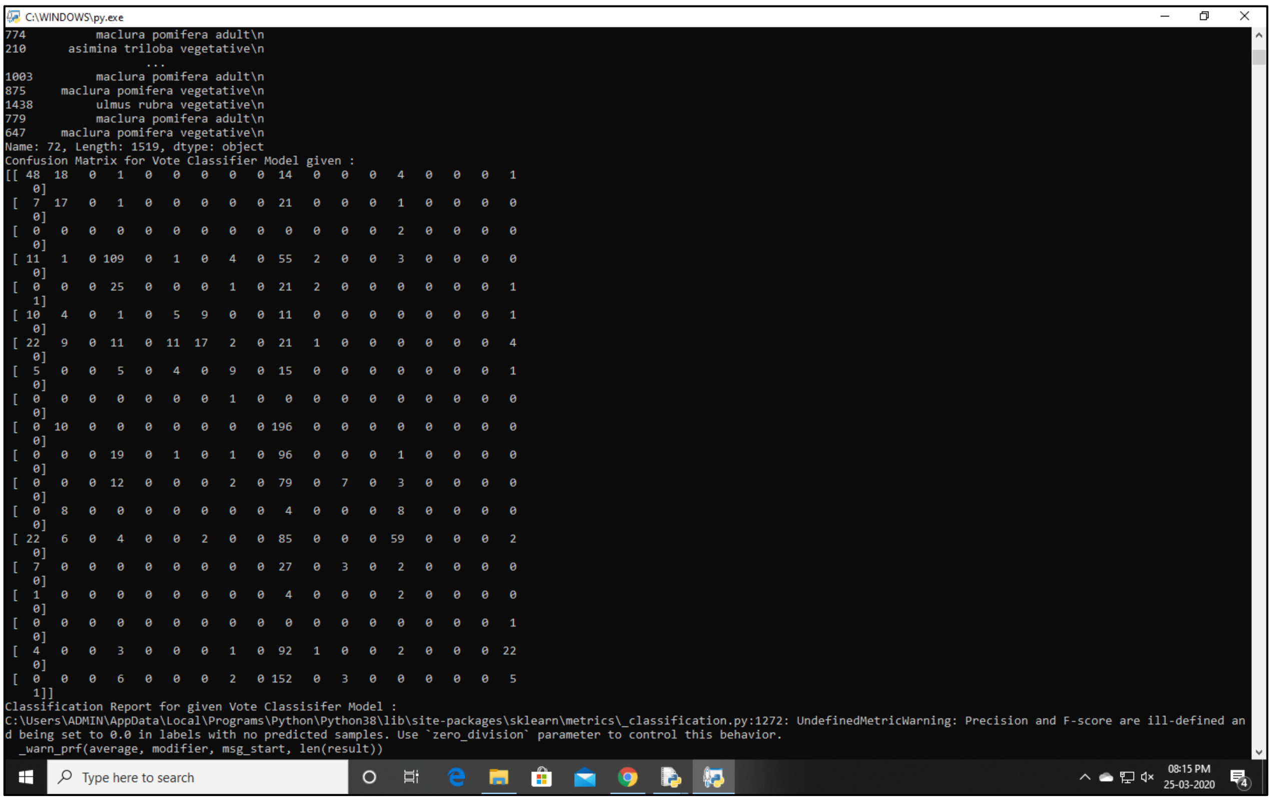
   Pseudo code for Voting Classifier Model

```
Start Voting Classifier Model
Accept Training Sets
Create a ML model for Naive Bayes
Create a ML model for K Nearest Neighbors
Create a ML model for Linear Support Vector Machine
Create a ML model for Polynomial Support Vector Machine
Create a ML model for Radial Bias Support Vector Machine
Create a ML model for Sigmoid Support Vector Machine
Create a ML model for Decision Tree Classifier
Add these Models as estimators to Voting Classifier
Create the Model for Voting Classifier with these estimators inserted into it
Fit this Model with the Training set
Return this Voting classifier Model
End
```

   Pseudo code for Voting Classifier Prediction

```
Start Voting Classifier Prediction
Accept the features for the Test
Predict the output using Test Set for the given Model Voting Classifier
Predict the output for all the estimators in the Model
```

```
    For each output of estimators
        Perform hard voting
        Update Output label when needed
    Return the Output label
    End
```



```
774          maclura pomifera adult\n
210      asimina triloba vegetative\n
                       ...
1003         maclura pomifera adult\n
875     maclura pomifera vegetative\n
1438         ulmus rubra vegetative\n
779          maclura pomifera adult\n
647     maclura pomifera vegetative\n
Name: 72, Length: 1519, dtype: object
Confusion Matrix for Vote Classifier Model given :
[[ 48  18   0   1   0   0   0   0   0  14   0   0   0   4   0   0   0   1
    0]
 [  7  17   0   1   0   0   0   0   0  21   0   0   0   1   0   0   0   0
    0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   2   0   0   0   0
    0]
 [ 11   1   0 109   0   1   0   4   0  55   2   0   0   3   0   0   0   0
    0]
 [  0   0   0  25   0   0   0   1   0  21   2   0   0   0   0   0   0   1
    1]
 [ 10   4   0   1   0   5   9   0   0  11   0   0   0   0   0   0   0   1
    0]
 [ 22   9   0  11   0  11  17   2   0  21   1   0   0   0   0   0   0   4
    0]
 [  5   0   0   5   0   4   0   9   0  15   0   0   0   0   0   0   0   1
    0]
 [  0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
    0]
 [  0  10   0   0   0   0   0   0   0 196   0   0   0   0   0   0   0   0
    0]
 [  0   0   0  19   0   1   0   1   0  96   0   0   0   1   0   0   0   0
    0]
 [  0   0   0  12   0   0   0   2   0  79   0   7   0   3   0   0   0   0
    0]
 [  0   8   0   0   0   0   0   0   0   4   0   0   0   8   0   0   0   0
    0]
 [ 22   6   0   4   0   0   2   0   0  85   0   0   0  59   0   0   0   2
    0]
 [  7   0   0   0   0   0   0   0   0  27   0   3   0   2   0   0   0   0
    0]
 [  1   0   0   0   0   0   0   0   0   4   0   0   0   2   0   0   0   0
    0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1
    0]
 [  4   0   0   3   0   0   0   1   0  92   1   0   0   2   0   0   0  22
    0]
 [  0   0   0   6   0   0   0   2   0 152   0   3   0   0   0   0   0   5
    1]]
Classification Report for given Vote Classisifer Model :
C:\Users\ADMIN\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetricWarning: Precision and F-score are ill-defined an
d being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Fid 3.14: Confusion Matrix for Voting Classifier Implementation

3.2.3 Frontend and Backend

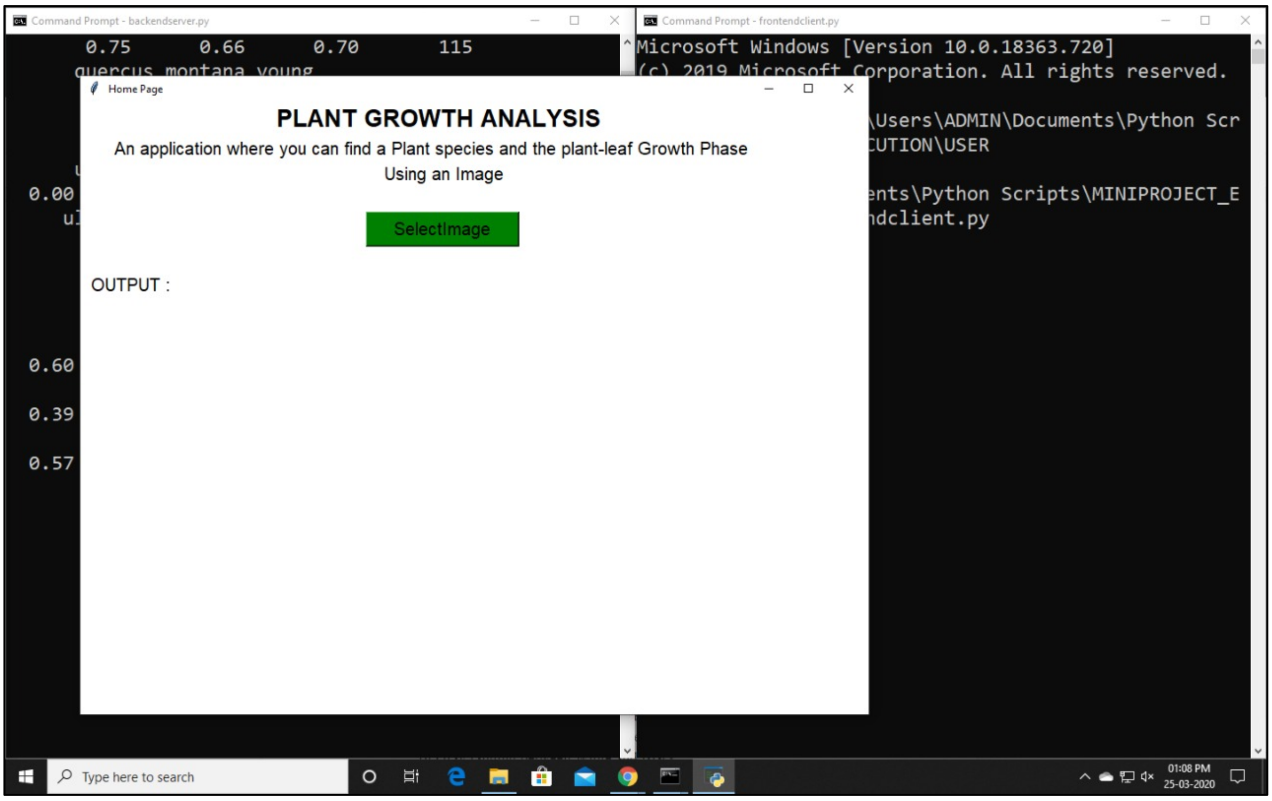
The Project System needs a User Interface to work upon. This arises a need to develop a Front End UI to the System. On par with this front end, there is a need of setting up a Backend which takes up the main part of project execution. These parts are to be linked with a medium.

3.2.3.1 Frontend Development

The Front End of the system has been developed using Tkinter-Python. This supports various kinds of GUI applications like Buttons, Labels, Text Boxes, Dialogue boxes, etc., similar to the Applets of Java. The User front end has been added with required buttons and dialogue boxes that support Image selection for Prediction.

Pseudo code for Client Frontend

```
Start Client Frontend
Create a Tkinter Window
Add Button and a Button Event
Loop Infinitely
    Button Event for Button Clicked
        Open a Dialog Box
        Accept an Image file from the Dialog box
        Initiate Client Execution with this Image
Close the Window
End
```



Command Prompt - backendserver.py
Command Prompt - frontendclient.py
Home Page
SelectImage
Type here to search
01:08 PM
25-03-2020

```
   0.75         0.66         0.70         115
  quercus_montana young
0.00
0.60
0.39
0.57
```

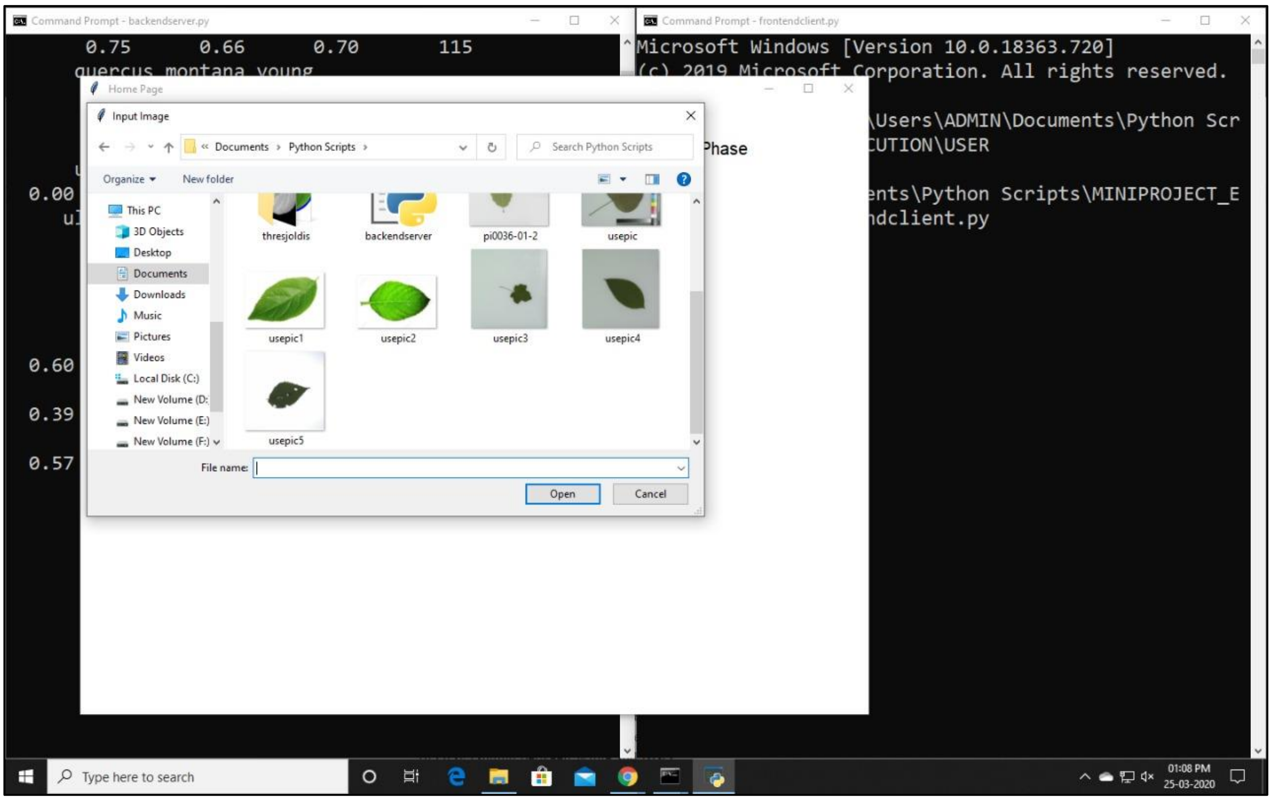
```
Microsoft Windows [Version 10.0.18363.720]
(c) 2019 Microsoft Corporation. All rights reserved.
\Users\ADMIN\Documents\Python Scr
CUTION\USER
ents\Python Scripts\MINIPROJECT_E
ndclient.py
```

PLANT GROWTH ANALYSIS

An application where you can find a Plant species and the plant-leaf Growth Phase Using an Image

OUTPUT :

Fig 3.15: Tkinter Window for Client



```
   0.75         0.66         0.70         115
  quercus_montana young
0.00
0.60
0.39
0.57
```

```
Microsoft Windows [Version 10.0.18363.720]
(c) 2019 Microsoft Corporation. All rights reserved.
\Users\ADMIN\Documents\Python Scr
CUTION\USER
ents\Python Scripts\MINIPROJECT_E
ndclient.py
```

Fig 3.16: Dialog Box for Image Input selection

3.2.3.2 Backend Development

The Backend of the system has been developed with Python. All the computations happening during the execution are with Python and with some modules like numpy, pandas, cv2, etc.

Pseudo code for Backend for Dataset

```
Start Backend for Dataset
Create a Data Storage for storing the dataset
Take the Image set
For each image in the Image set
    Obtain the features using Feature Extract
    Add these features to the data storage file
Commit the Data Storage file
End
```

Pseudo code for Data Support

```
Start Data Support
Use the Data Storage created in Backend for Dataset
Read this storage and structure them
Split the Dataset into train and test sets
And return these train test splits
End
```

3.2.3.3 Client - Server Implementation

In the Real Time applications, the user needs the assistance of a server for computations. In the same way, a Client-server Model has been implemented where the client sends the Image to the server and the server performs the necessary actions and returns the output to the user. This ensures better implementations and also the server dataset can be improved using the images sent by the user by tagging the image and adding the features to the set.

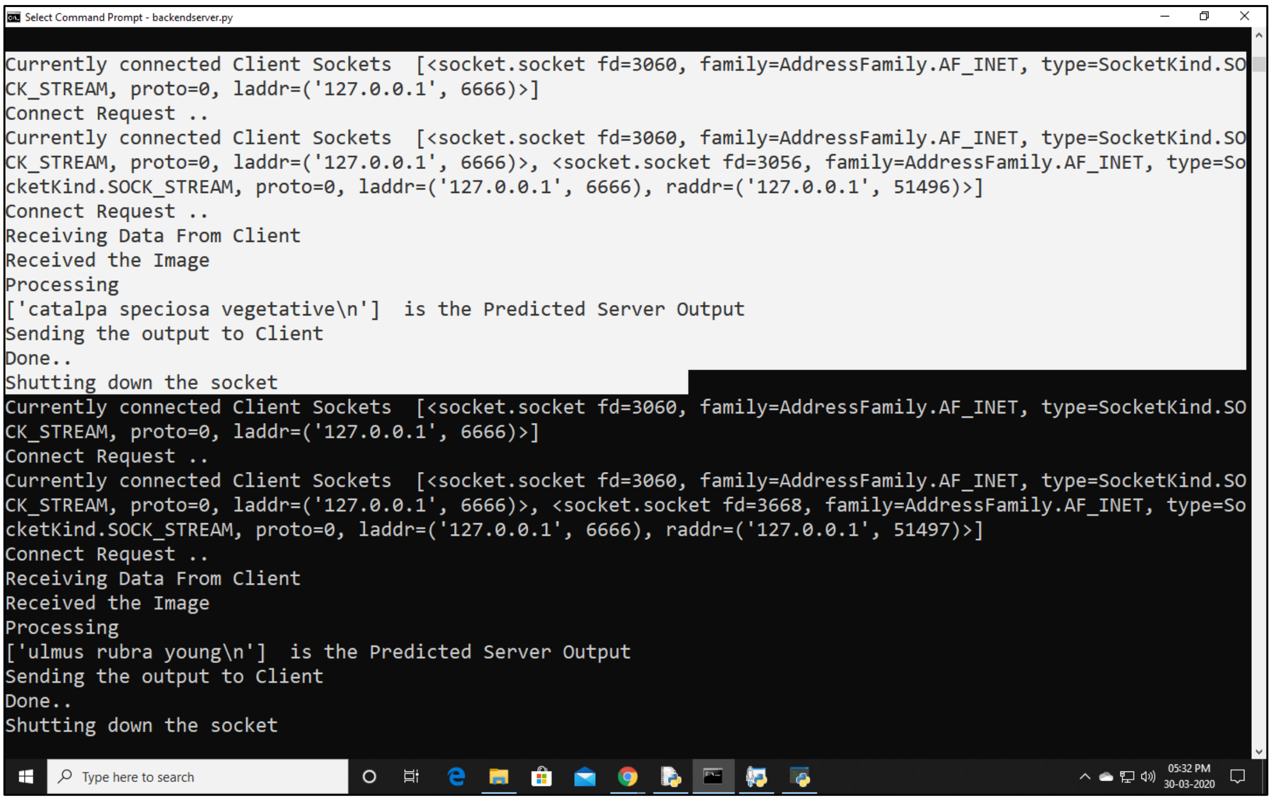
Server Execution

The server program starts its execution at first. After making some pre computations, it gets ready for the executions demanded by the client. The Server resides in the Listen Mode from this point. Whenever it receives a connection request, the server accepts the connection and pauses the Listen mode. Then takes the image that has been sent by the user and performs feature extraction and then predicts the label. This predicted label is sent in the form of bytes encoded. And then loops back to its Listen mode for further connection requests by clients. An Image counter is used to keep track of the images that are being predicted and also the images are saved with this counter as a part in the name. This enables us to have these images stored. At regular intervals, we could use these images to improve the database existing with the server.

Pseudo code for Server

```
Start Server
Initiate execution of Data Support
Load the Prediction - Vote Model
Set the server to LISTEN mode
Check for the Connection requests by clients
```

```
        If a request is found
                Accept the Request
                Establish Connection
                Block the server Listen Mode
                Accept the Image from the Client
                Call the Prediction method
                Take the output of the prediction method
                Send this output back to the Client
                Disconnect the Client
                Store this Image at a Buffer for further use
                Loop back to LISTEN Mode
        If No request is found
                Wait for a connection Request
Loop this Process Infinitely
End
```



```
Currently connected Client Sockets  [<socket.socket fd=3060, family=AddressFamily.AF_INET, type=SocketKind.SO
CK_STREAM, proto=0, laddr=('127.0.0.1', 6666)>]
Connect Request ..
Currently connected Client Sockets  [<socket.socket fd=3060, family=AddressFamily.AF_INET, type=SocketKind.SO
CK_STREAM, proto=0, laddr=('127.0.0.1', 6666)>, <socket.socket fd=3056, family=AddressFamily.AF_INET, type=So
cketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 6666), raddr=('127.0.0.1', 51496)>]
Connect Request ..
Receiving Data From Client
Received the Image
Processing
['catalpa speciosa vegetative\n']  is the Predicted Server Output
Sending the output to Client
Done..
Shutting down the socket
Currently connected Client Sockets  [<socket.socket fd=3060, family=AddressFamily.AF_INET, type=SocketKind.SO
CK_STREAM, proto=0, laddr=('127.0.0.1', 6666)>]
Connect Request ..
Currently connected Client Sockets  [<socket.socket fd=3060, family=AddressFamily.AF_INET, type=SocketKind.SO
CK_STREAM, proto=0, laddr=('127.0.0.1', 6666)>, <socket.socket fd=3668, family=AddressFamily.AF_INET, type=So
cketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 6666), raddr=('127.0.0.1', 51497)>]
Connect Request ..
Receiving Data From Client
Received the Image
Processing
['ulmus rubra young\n']  is the Predicted Server Output
Sending the output to Client
Done..
Shutting down the socket
```

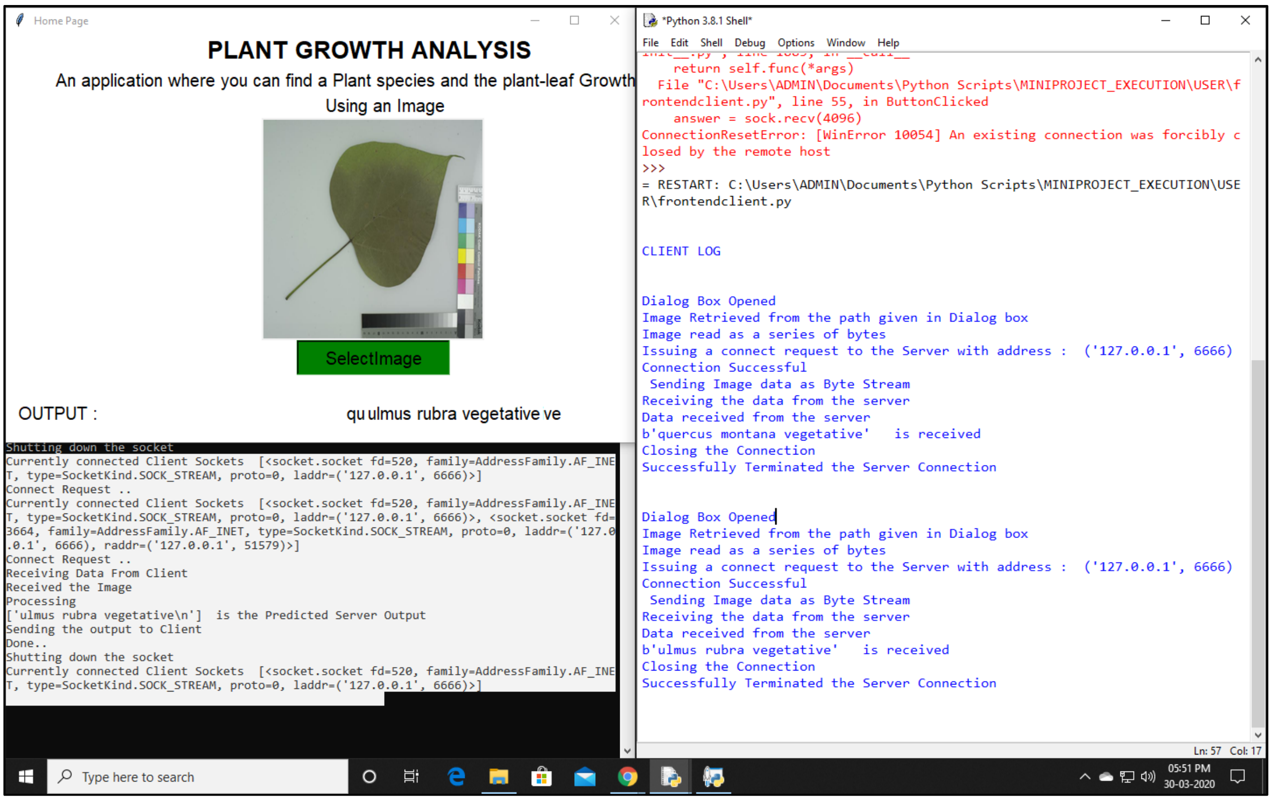
Fig 3.17: Server Process demonstration

Client Execution

The Client program when executed, creates and opens a window which contains a dialog box to select the image to be sent to the server. When the image is selected, a connection request is sent to the server using a port address and a socket address. The selected image is converted into bytes and this byte stream is sent to the server. The server performs the computations needed and sends back the predicted output in the form of byte stream. This Byte stream is converted back to string and displayed as output at the clients' GUI window.

Pseudo code for Client

```
Start Client Execution
Take the Image for Prediction
Raise a Connection Request to server
Send the Image to the server
Accept the answer sent by the server
Terminate Connection
Display the answer as output to the user
End
```



Home Page
PLANT GROWTH ANALYSIS
An application where you can find a Plant species and the plant-leaf Growth
Using an Image
SelectImage
OUTPUT :
qu ulmus rubra vegetative ve
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Type here to search
Ln: 57 Col: 17
05:51 PM
30-03-2020

```
Shutting down the socket
Currently connected Client Sockets  [<socket.socket fd=520, family=AddressFamily.AF_INE
T, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 6666)>]
Connect Request ..
Currently connected Client Sockets  [<socket.socket fd=520, family=AddressFamily.AF_INE
T, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 6666)>, <socket.socket fd=
3664, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0
.0.1', 6666), raddr=('127.0.0.1', 51579)>]
Connect Request ..
Receiving Data From Client
Received the Image
Processing
['ulmus rubra vegetative\n']  is the Predicted Server Output
Sending the output to Client
Done..
Shutting down the socket
Currently connected Client Sockets  [<socket.socket fd=520, family=AddressFamily.AF_INE
T, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 6666)>]
```

```
    return self.func(*args)
  File "C:\Users\ADMIN\Documents\Python Scripts\MINIPROJECT_EXECUTION\USER\f
rontendclient.py", line 55, in ButtonClicked
    answer = sock.recv(4096)
ConnectionResetError: [WinError 10054] An existing connection was forcibly c
losed by the remote host
>>> 
= RESTART: C:\Users\ADMIN\Documents\Python Scripts\MINIPROJECT_EXECUTION\USE
R\frontendclient.py


CLIENT LOG


Dialog Box Opened
Image Retrieved from the path given in Dialog box
Image read as a series of bytes
Issuing a connect request to the Server with address :  ('127.0.0.1', 6666)
Connection Successful
 Sending Image data as Byte Stream
Receiving the data from the server
Data received from the server
b'quercus montana vegetative'   is received
Closing the Connection
Successfully Terminated the Server Connection


Dialog Box Opened
Image Retrieved from the path given in Dialog box
Image read as a series of bytes
Issuing a connect request to the Server with address :  ('127.0.0.1', 6666)
Connection Successful
 Sending Image data as Byte Stream
Receiving the data from the server
Data received from the server
b'ulmus rubra vegetative'   is received
Closing the Connection
Successfully Terminated the Server Connection
```

Fig 3.18: Client Process Demonstration

3.3 INPUT AND OUTPUT ANALYSIS

The Inputs to the project can be broadly categorized into two types.

Input to the Servers' Dataset

The Inputs to the Dataset include the Images of the leaves of different species and their labels that define their Species family and their growth phase. This helps the project to create a well-structured model for prediction.

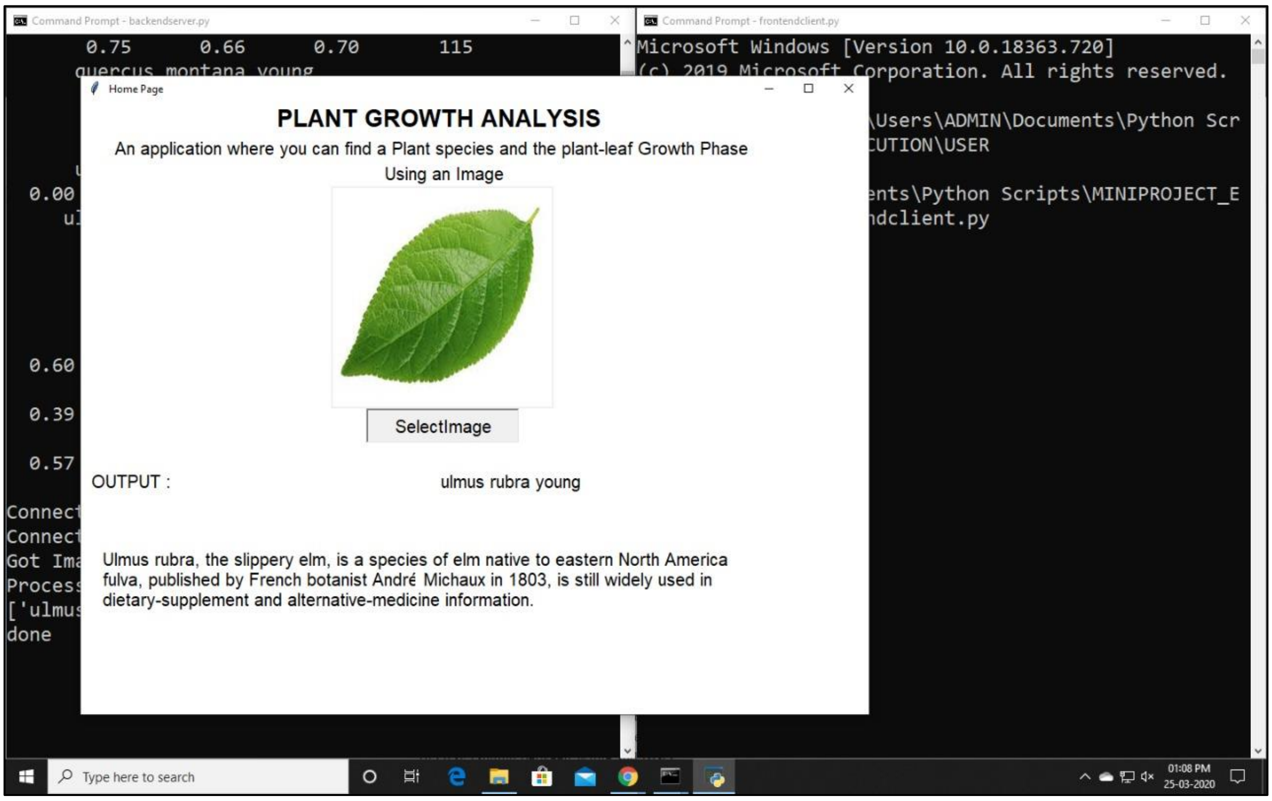
This requires a large dataset for the accurate prediction. The Images that are being used are of two kinds. Some are the images that are captured without external noise pertaining in the image. Some of the images are those with external noise and disturbances added to the image which improve the bias in the image and help the model to predict precisely when the test image consists of noise. These Images undergo computations to extract the features.

Input by User

The User Input is taken using a Tkinter window which is assisted by the Dialogue Box. This is sent to the server in the encoded Binary Format. The server takes the data as a collection of bytes and creates an image at the server and uses it as the Input to its ML models.

Output to the User

The Output that is obtained is the Plant Species and the growth phase of the leaf which is being given as input. Here, the computations take place at the server. This data predicts the species and the vegetative phase. This prediction is sent to the server in the form of encoded bytes and this predicted output is displayed on to the Tkinter Window.



Command Prompt - backendserver.py

```
      0.75       0.66       0.70        115
     quercus montana young

 0.00
    ul

 0.60

 0.39

 0.57

Connect
Connect
Got Ima
Process
['ulmus
done
```

Command Prompt - frontendclient.py

```
Microsoft Windows [Version 10.0.18363.720]
(c) 2019 Microsoft Corporation. All rights reserved.

\Users\ADMIN\Documents\Python Scr
CUTION\USER

ents\Python Scripts\MINIPROJECT_E
ndclient.py
```

Home Page

PLANT GROWTH ANALYSIS

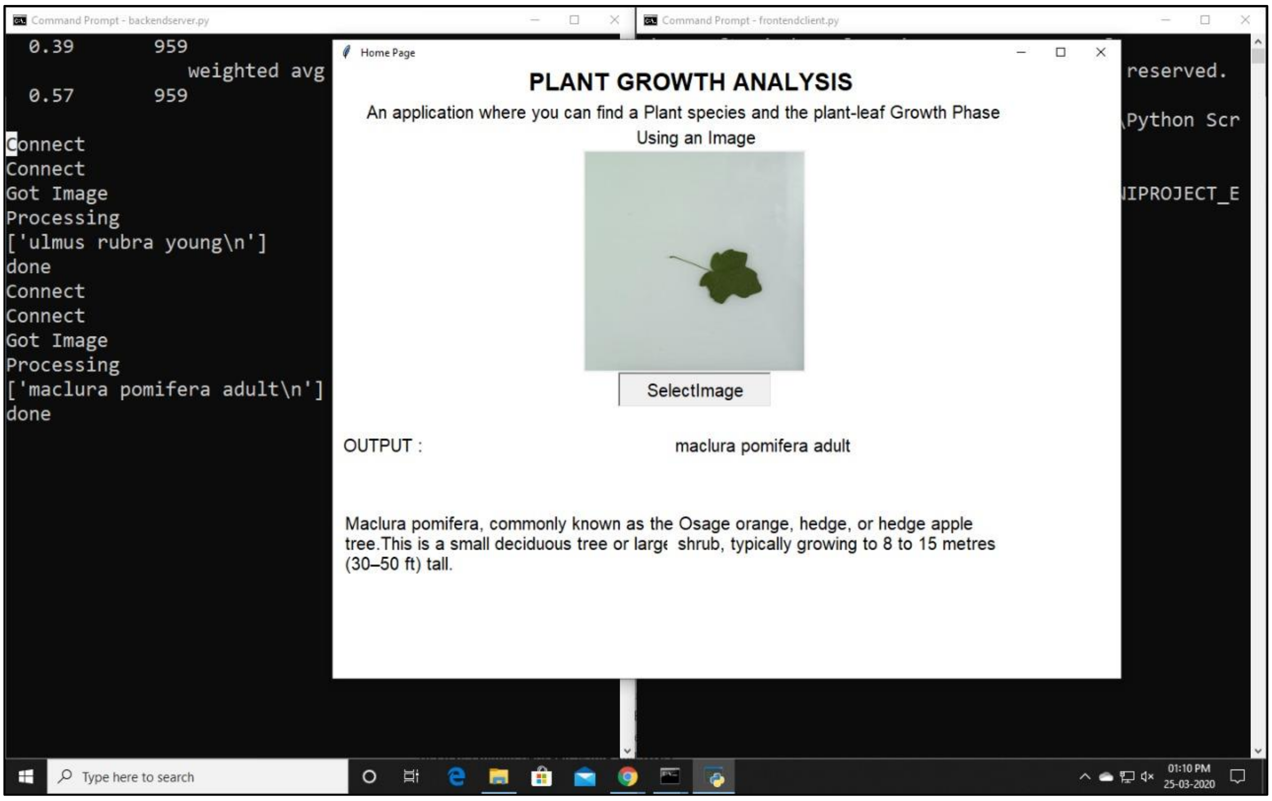
An application where you can find a Plant species and the plant-leaf Growth Phase Using an Image

SelectImage

OUTPUT : ulmus rubra young

Ulmus rubra, the slippery elm, is a species of elm native to eastern North America fulva, published by French botanist André Michaux in 1803, is still widely used in dietary-supplement and alternative-medicine information.

Fig 3.19: Output to the User / Client



Command Prompt - backendserver.py

```
  0.39       959
                weighted avg
  0.57       959

Connect
Connect
Got Image
Processing
['ulmus rubra young\n']
done
Connect
Connect
Got Image
Processing
['maclura pomifera adult\n']
done
```

Command Prompt - frontendclient.py

```
reserved.

Python Scr

IPROJECT_E
```

Home Page

PLANT GROWTH ANALYSIS

An application where you can find a Plant species and the plant-leaf Growth Phase Using an Image

SelectImage

OUTPUT : maclura pomifera adult

Maclura pomifera, commonly known as the Osage orange, hedge, or hedge apple tree.This is a small deciduous tree or large shrub, typically growing to 8 to 15 metres (30–50 ft) tall.

Fig 3.20: Server Log and Client Output

CHAPTER - 4
CONCLUSION AND FUTURE ENHANCEMENTS

4.1 Conclusion

The Plants are living creatures of the Kingdom Plantae which prepare their starch using a pigment which is present in the leaves. The Chlorophyll amounts change during the course of the plant growth. The chlorophyll, at the beginning stages of the growth, is lesser than average and within a very few days of time, the chlorophyll grows in amount and becomes abundant during the leaf growth. This chlorophyll may or may not be of different color in different species of plant. But these species can have different shapes, sizes, texture, vein structures, etc. These Differences in these species and their Chlorophyll colors and levels at optimum state becomes the base of the project. The shape, size, circularity, solidity, elongation, convexity features define the size and shape features of the leaf. These features primarily distinguish between the species of plants. The texture and color intensities are also noted which identify the chlorophyll levels of the leaf and can be used to distinguish between the species due to difference in colors of chlorophyll pigments and then between the growth phases as the chlorophyll levels increase during the progression of growth. These features are extracted using the technology and fed to a machine learning model and it is made to fit to the total database expecting higher accuracy and precision.

The separation of control over the use of this application is achieved by implementing the Client and server architecture so that there is no redundancy in storage of data at multiple places and reducing the runtime at the user end. This also helps us in reducing the external resources needed by the user to perform the tasks during the execution. This improves quality of service. A Graphical interface has been used to the user end to improve usability and accessibility to the user.

4.2 Future Enhancements

The Project can be further improved using a Neural Network based reading of the image like Convolutional Neural Networks. This helps in reading the image in layers and helps in drawing hidden features like Vein Structure, Vein texture, vein split counts. Layered texture changes and many more which drastically increase the accuracy and precision. Some types of Neural Networks like Multi layered perceptrons, Radial basis Networks, Modular networks, Probabilistic networks, etc., can be used to speed up and improve the accuracy of the outputs. This can be extended along with Integration with Disease finding software and with IoT, plants can be taken care of by bots with a very minimal Human Involvement. Much more Integrations can be added to improve technical optimizations and automations. The same System can be added with a mobile application to upgrade the user accessibility and improve the system availability.

REFERENCES

1. **Sangeet Saha1 , Chandrajit pal2 , Rourab paul3 , Satyabrata Maity 4 , Suman Sau**5"A brief experience on journey through hardware developments for image processing and its applications on Cryptography" Dept of Computer Science & Engineering 1, A. K. Choudhury School of Information Technology 2,3,4,5 University Of Calcutta, Kolkata, India 92, A.P.C Road,Kolkata-700009

2. **J.Marot, Y.Caulier, A.Kuleschov, K. Spinnler, and S. Bourennane Fraunhofer , Mark Nixon & Alberto Aguado,** PContour Detection for Industrial Image Processing by Means of level Set Methods Institute IIS Am Wolfsmantel

3. **Sebastian Raschka,** Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning University of Wisconsin–Madison Department of Statistics November 2018

4. EG Learned Miller. Introduction to ComputerVision. University of Massachusetts (2011)

5. **Hany Farid,** Fundamentals of Image Processing, Dartmouth College 2010.

6. **Alan D Moore**, Python GUI programming with Tkinter, PacktPublishings 2018

7. **EG Learned Miller**, Introduction to computer vision, MIT 2011

8. **Mark Lutz,** Learning Python O'Reilly Media, June 2013

9. **Aurelien Geron**, Hands on Machine learning using sci-kit learn, O'Reilly 2017

10. https://docs.python.org/ Python Implementation details

11. https://kite.com/python/docs/sklearn Sci Kit Learn implementation for Python.

12. https://docs.opencv.org/master/ Open Cv Documentation for Python

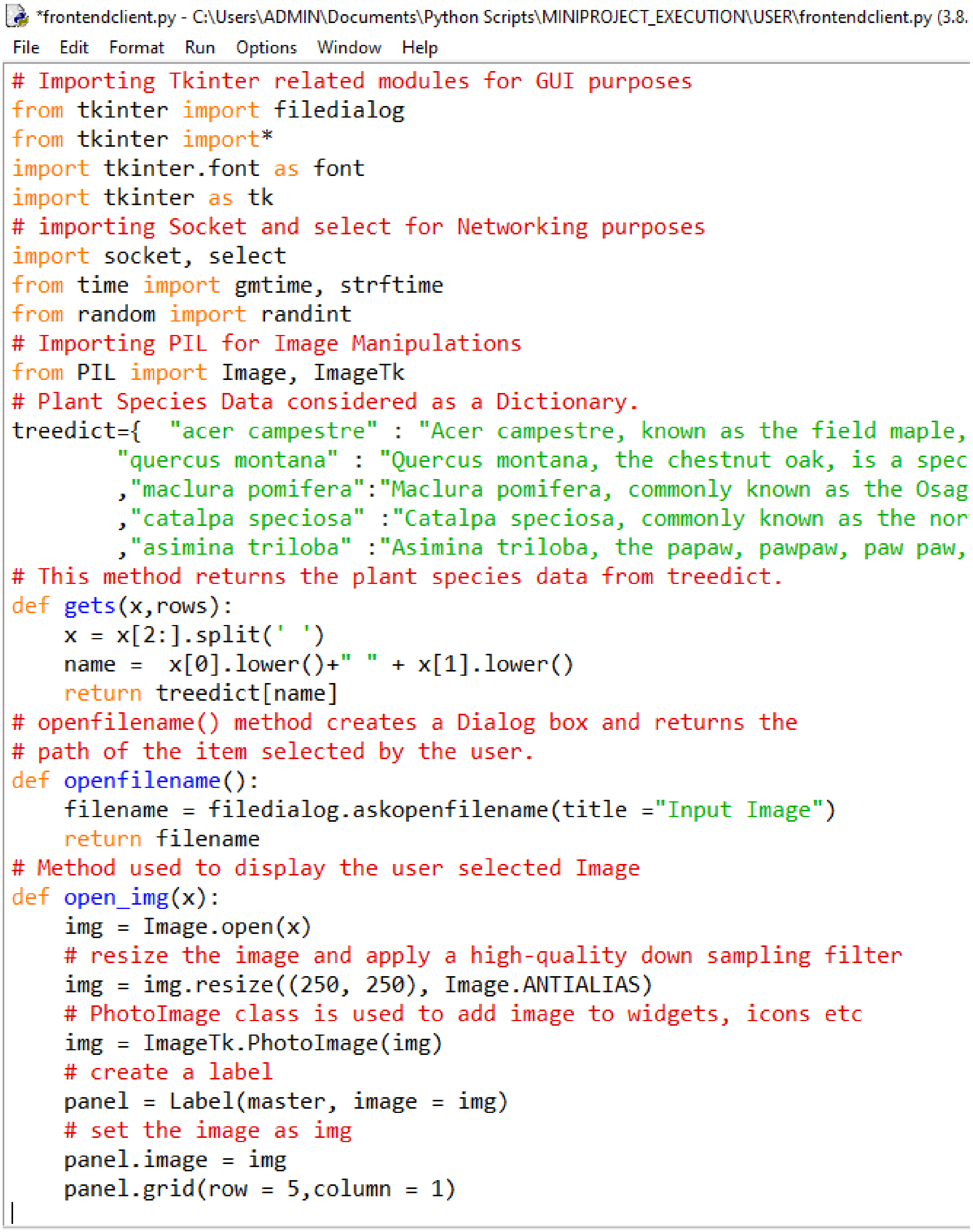
13. Feature selection for machine Learning https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c

14. https://towardsdatascience.com/the-5-feature-selection-algorithms-every-data-scientist-need-to-know-3a6b566efd2 Feature selection for Machine learning based projects.

15. https://realpython.com/python-sockets/#multi-connection-client-and-server Client server implementation guide

SAMPLE CODES

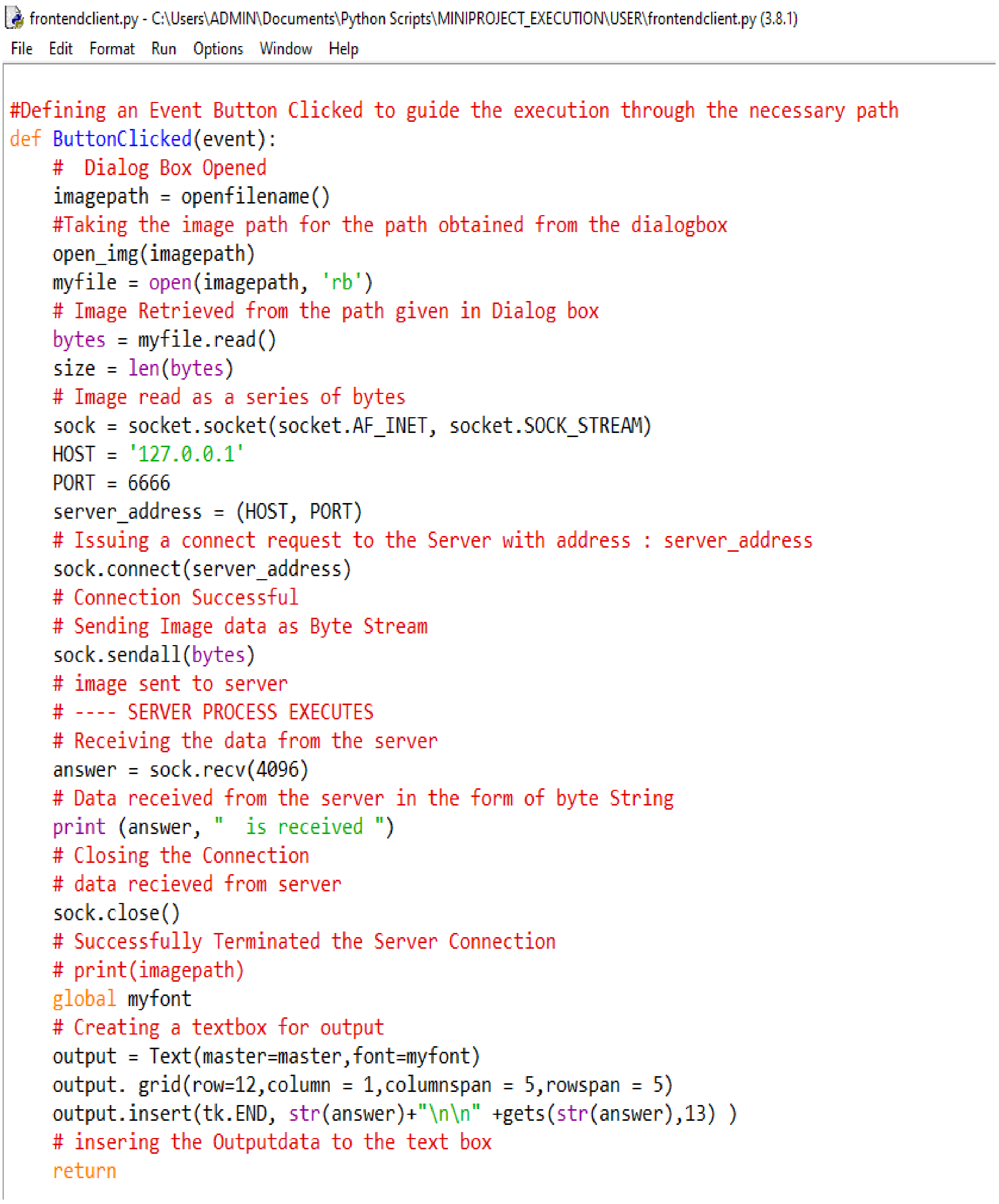
Client Execution :



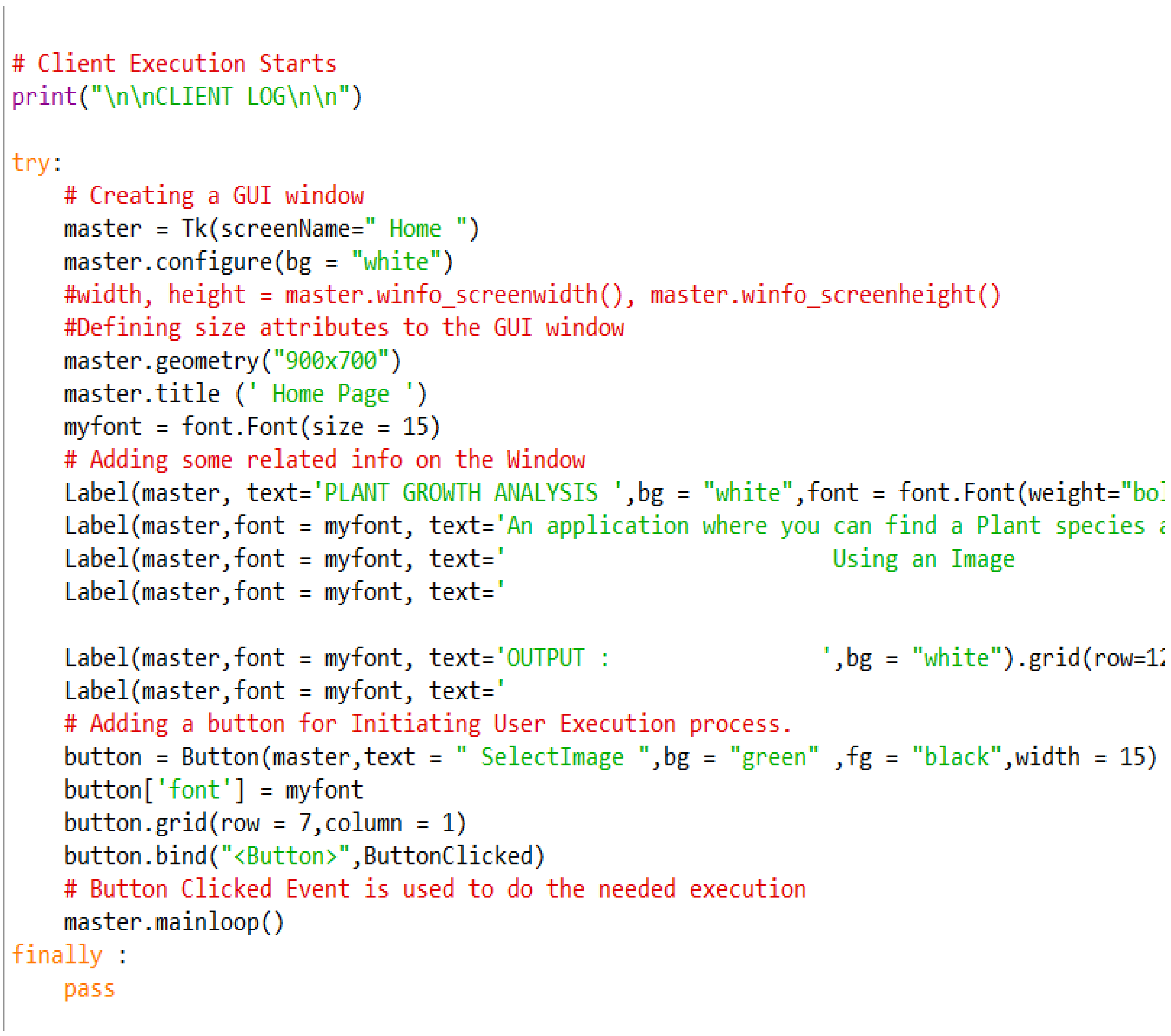
*frontendclient.py - C:\Users\ADMIN\Documents\Python Scripts\MINIPROJECT_EXECUTION\USER\frontendclient.py (3.8.

```
# Importing Tkinter related modules for GUI purposes
from tkinter import filedialog
from tkinter import*
import tkinter.font as font
import tkinter as tk
# importing Socket and select for Networking purposes
import socket, select
from time import gmtime, strftime
from random import randint
# Importing PIL for Image Manipulations
from PIL import Image, ImageTk
# Plant Species Data considered as a Dictionary.
treedict={  "acer campestre" : "Acer campestre, known as the field maple,
        "quercus montana" : "Quercus montana, the chestnut oak, is a spec
        ,"maclura pomifera":"Maclura pomifera, commonly known as the Osag
        ,"catalpa speciosa" :"Catalpa speciosa, commonly known as the nor
        ,"asimina triloba" :"Asimina triloba, the papaw, pawpaw, paw paw,
# This method returns the plant species data from treedict.
def gets(x,rows):
    x = x[2:].split(' ')
    name =  x[0].lower()+" " + x[1].lower()
    return treedict[name]
# openfilename() method creates a Dialog box and returns the
# path of the item selected by the user.
def openfilename():
    filename = filedialog.askopenfilename(title ="Input Image")
    return filename
# Method used to display the user selected Image
def open_img(x):
    img = Image.open(x)
    # resize the image and apply a high-quality down sampling filter
    img = img.resize((250, 250), Image.ANTIALIAS)
    # PhotoImage class is used to add image to widgets, icons etc
    img = ImageTk.PhotoImage(img)
    # create a label
    panel = Label(master, image = img)
    # set the image as img
    panel.image = img
    panel.grid(row = 5,column = 1)
```

Button Clicked Event/ Method where the whole Execution occurs at client side.


frontendclient.py - C:\Users\ADMIN\Documents\Python Scripts\MINIPROJECT_EXECUTION\USER\frontendclient.py (3.8.1)
File Edit Format Run Options Window Help

```
#Defining an Event Button Clicked to guide the execution through the necessary path
def ButtonClicked(event):
    #  Dialog Box Opened
    imagepath = openfilename()
    #Taking the image path for the path obtained from the dialogbox
    open_img(imagepath)
    myfile = open(imagepath, 'rb')
    # Image Retrieved from the path given in Dialog box
    bytes = myfile.read()
    size = len(bytes)
    # Image read as a series of bytes
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    HOST = '127.0.0.1'
    PORT = 6666
    server_address = (HOST, PORT)
    # Issuing a connect request to the Server with address : server_address
    sock.connect(server_address)
    # Connection Successful
    # Sending Image data as Byte Stream
    sock.sendall(bytes)
    # image sent to server
    # ---- SERVER PROCESS EXECUTES
    # Receiving the data from the server
    answer = sock.recv(4096)
    # Data received from the server in the form of byte String
    print (answer, "  is received ")
    # Closing the Connection
    # data recieved from server
    sock.close()
    # Successfully Terminated the Server Connection
    # print(imagepath)
    global myfont
    # Creating a textbox for output
    output = Text(master=master,font=myfont)
    output. grid(row=12,column = 1,columnspan = 5,rowspan = 5)
    output.insert(tk.END, str(answer)+"\n\n" +gets(str(answer),13) )
    # insering the Outputdata to the text box
    return
```

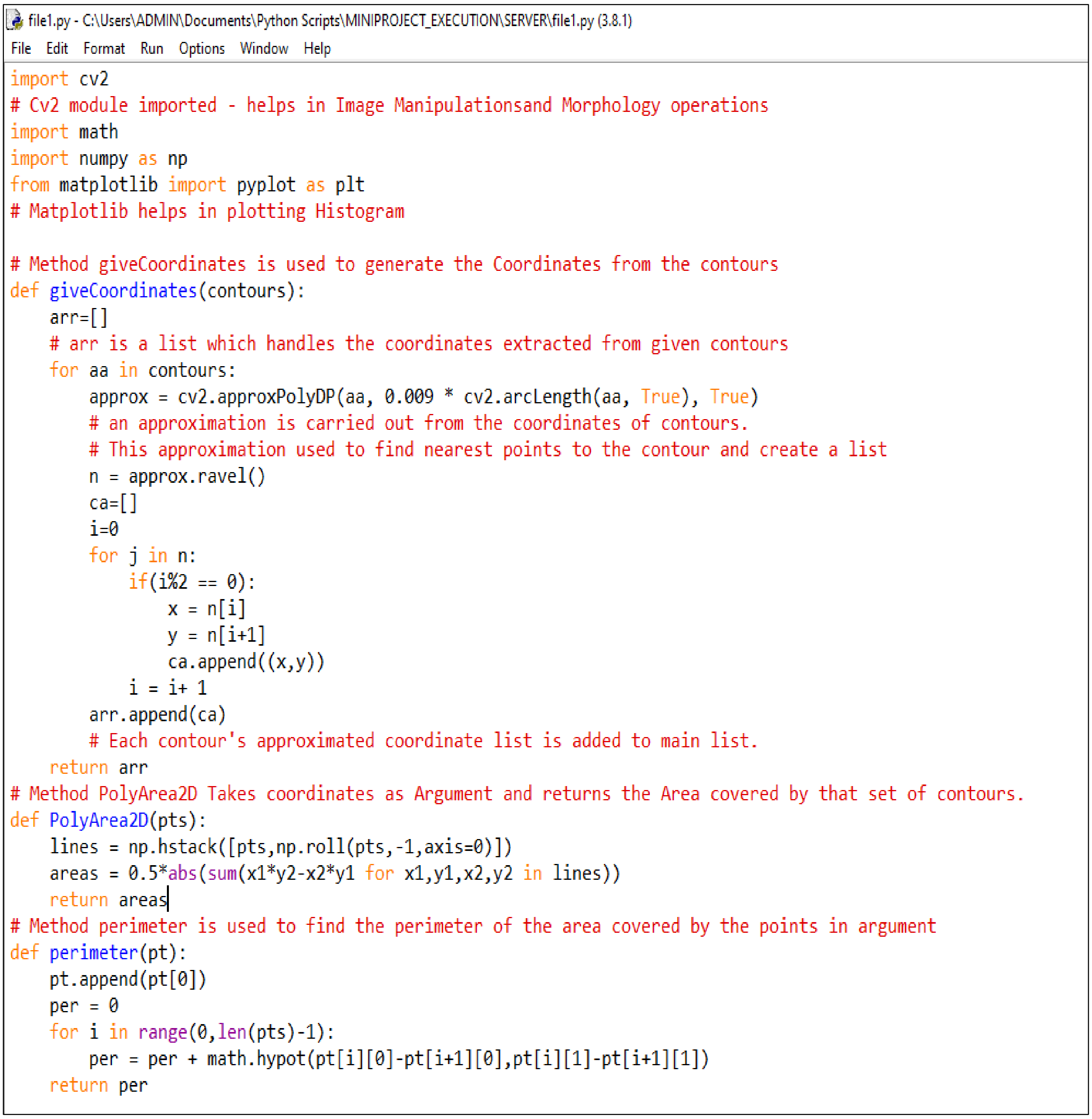


```
# Client Execution Starts
print("\n\nCLIENT LOG\n\n")

try:
    # Creating a GUI window
    master = Tk(screenName=" Home ")
    master.configure(bg = "white")
    #width, height = master.winfo_screenwidth(), master.winfo_screenheight()
    #Defining size attributes to the GUI window
    master.geometry("900x700")
    master.title (' Home Page ')
    myfont = font.Font(size = 15)
    # Adding some related info on the Window
    Label(master, text='PLANT GROWTH ANALYSIS ',bg = "white",font = font.Font(weight="bo
    Label(master,font = myfont, text='An application where you can find a Plant species 
    Label(master,font = myfont, text='                         Using an Image
    Label(master,font = myfont, text='

    Label(master,font = myfont, text='OUTPUT :                ',bg = "white").grid(row=1
    Label(master,font = myfont, text='
    # Adding a button for Initiating User Execution process.
    button = Button(master,text = " SelectImage ",bg = "green" ,fg = "black",width = 15)
    button['font'] = myfont
    button.grid(row = 7,column = 1)
    button.bind("<Button>",ButtonClicked)
    # Button Clicked Event is used to do the needed execution
    master.mainloop()
finally :
    pass
```

Client GUI for Client side Execution

Server Execution:

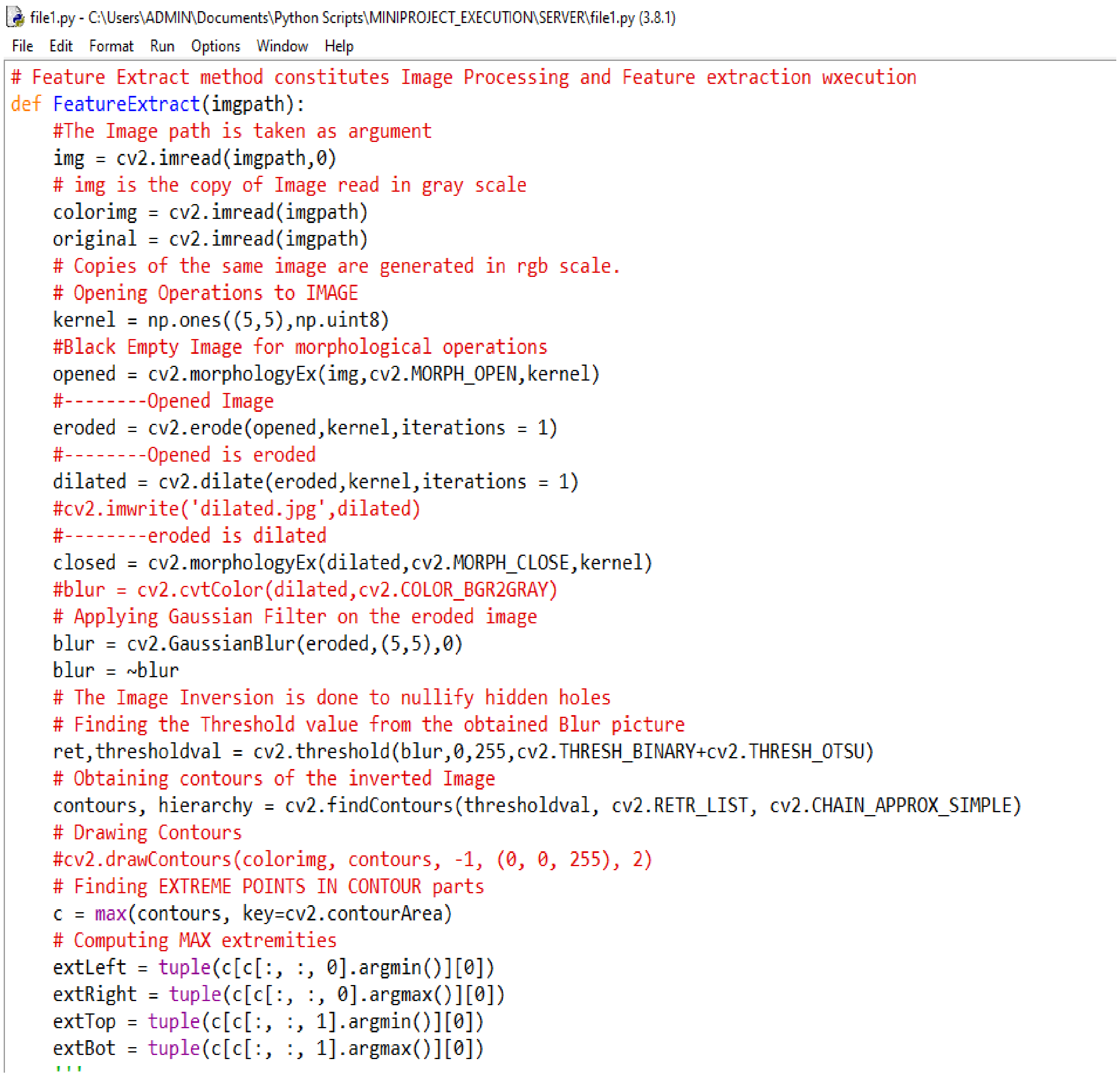
Module “file1”


file1.py - C:\Users\ADMIN\Documents\Python Scripts\MINIPROJECT_EXECUTION\SERVER\file1.py (3.8.1)
File Edit Format Run Options Window Help

```python
import cv2
# Cv2 module imported - helps in Image Manipulationsand Morphology operations
import math
import numpy as np
from matplotlib import pyplot as plt
# Matplotlib helps in plotting Histogram

# Method giveCoordinates is used to generate the Coordinates from the contours
def giveCoordinates(contours):
    arr=[]
    # arr is a list which handles the coordinates extracted from given contours
    for aa in contours:
        approx = cv2.approxPolyDP(aa, 0.009 * cv2.arcLength(aa, True), True)
        # an approximation is carried out from the coordinates of contours.
        # This approximation used to find nearest points to the contour and create a list
        n = approx.ravel()
        ca=[]
        i=0
        for j in n:
            if(i%2 == 0):
                x = n[i]
                y = n[i+1]
                ca.append((x,y))
            i = i+ 1
        arr.append(ca)
        # Each contour's approximated coordinate list is added to main list.
    return arr
# Method PolyArea2D Takes coordinates as Argument and returns the Area covered by that set of contours.
def PolyArea2D(pts):
    lines = np.hstack([pts,np.roll(pts,-1,axis=0)])
    areas = 0.5*abs(sum(x1*y2-x2*y1 for x1,y1,x2,y2 in lines))
    return areas
# Method perimeter is used to find the perimeter of the area covered by the points in argument
def perimeter(pt):
    pt.append(pt[0])
    per = 0
    for i in range(0,len(pts)-1):
        per = per + math.hypot(pt[i][0]-pt[i+1][0],pt[i][1]-pt[i+1][1])
    return per
```

Feature Extract Method

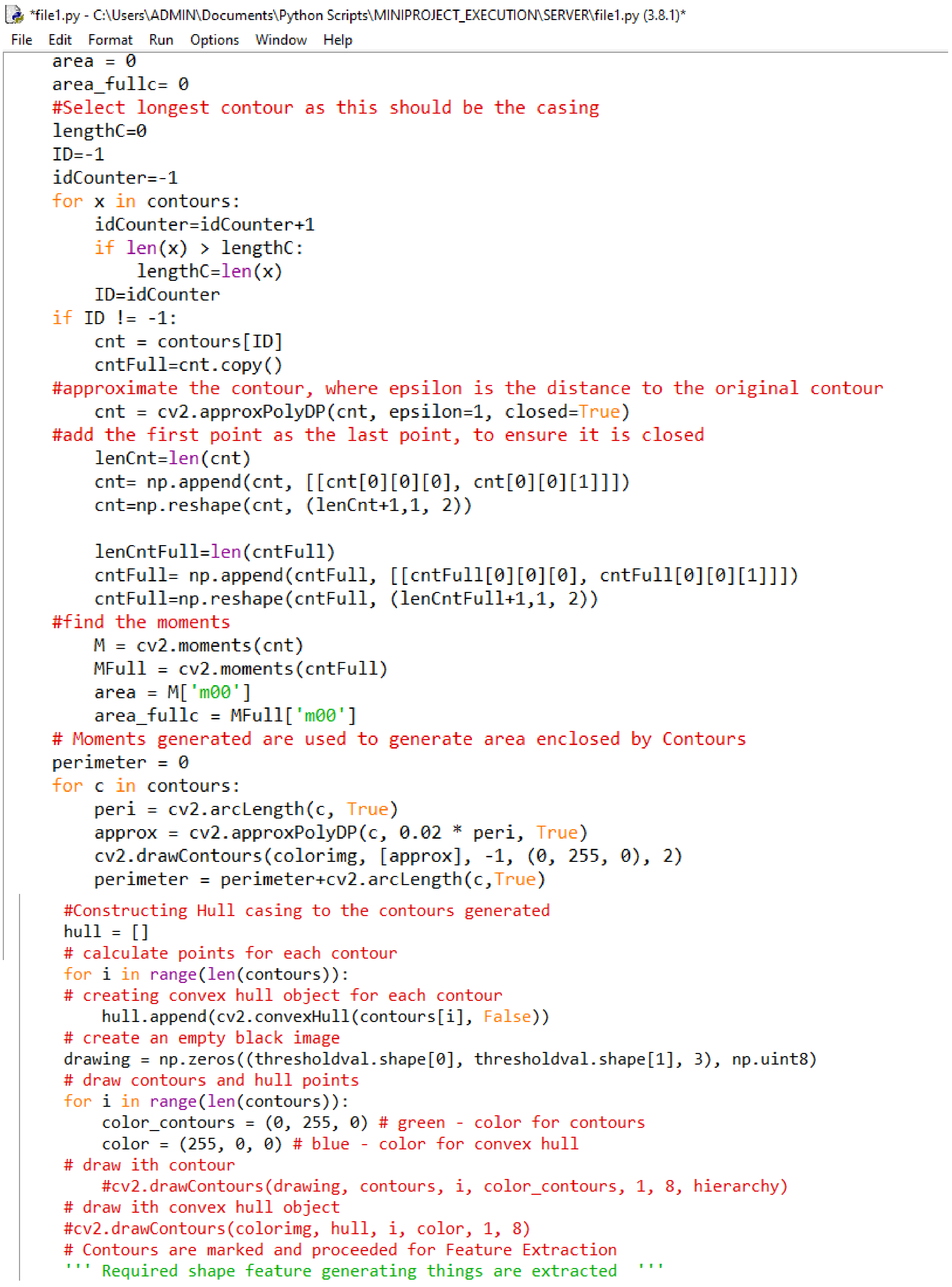
```
'''
# MARKING the Extremeties for Contours
cv2.circle(colorimg, extLeft, 8, (0, 0, 255), -1)
cv2.circle(colorimg, extRight, 8, (0, 255, 0), -1)
cv2.circle(colorimg, extTop, 8, (255, 0, 0), -1)
cv2.circle(colorimg, extBot, 8, (255, 255, 0), -1)
'''
# CROPPING THE image based on the extremities
cropped = original[extTop[1]:extBot[1],extLeft[0]:extRight[0]].copy()
# Obtaining the Color Histogram of the cropped picture.
(hist,bins,patches) = plt.hist(cropped.ravel(),bins = 64,range=(0,256),density=0)
#  hist is the array of the top points in the Histogram
```



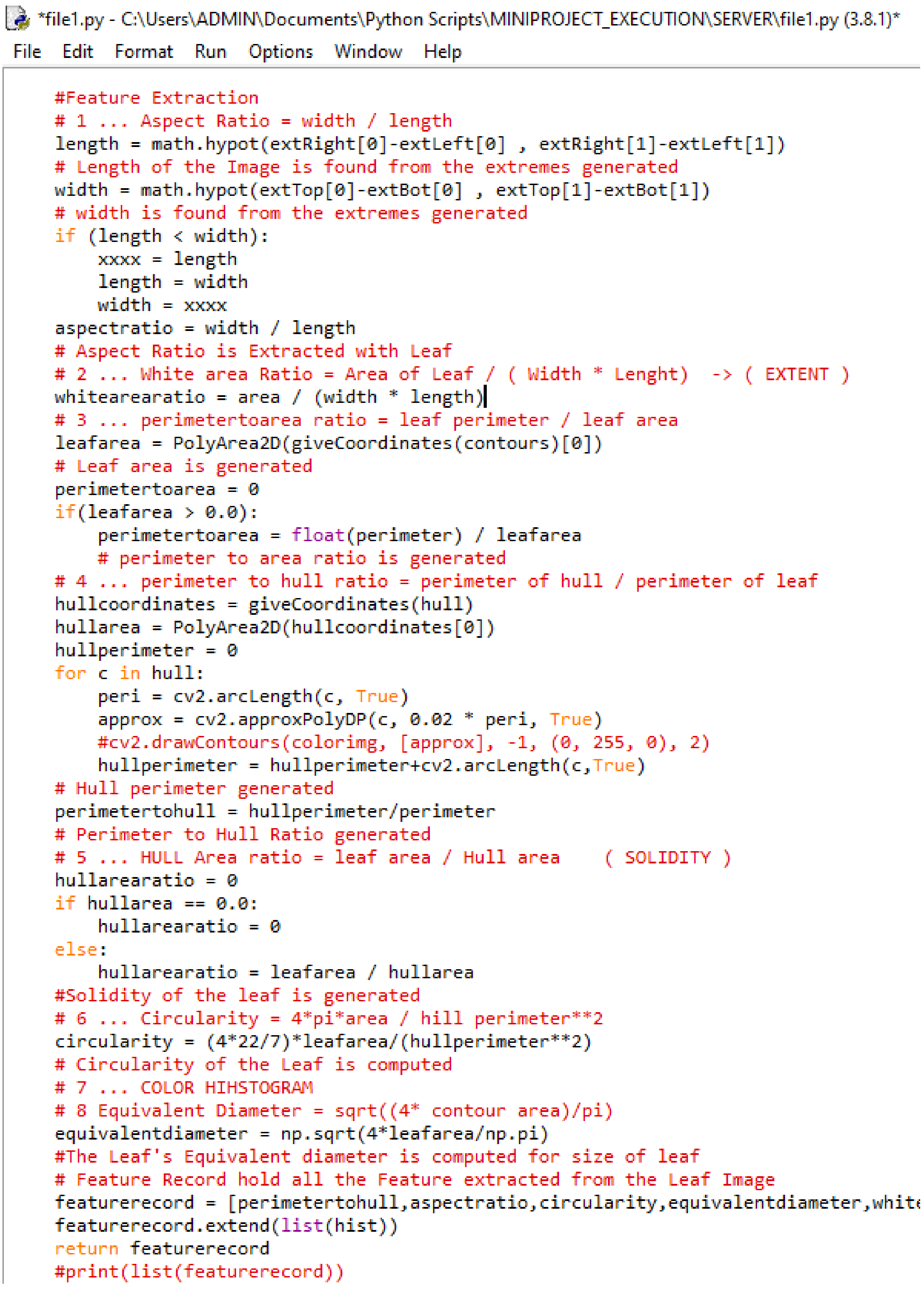
file1.py - C:\Users\ADMIN\Documents\Python Scripts\MINIPROJECT_EXECUTION\SERVER\file1.py (3.8.1)

File Edit Format Run Options Window Help

```python
# Feature Extract method constitutes Image Processing and Feature extraction wxecution
def FeatureExtract(imgpath):
    #The Image path is taken as argument
    img = cv2.imread(imgpath,0)
    # img is the copy of Image read in gray scale
    colorimg = cv2.imread(imgpath)
    original = cv2.imread(imgpath)
    # Copies of the same image are generated in rgb scale.
    # Opening Operations to IMAGE
    kernel = np.ones((5,5),np.uint8)
    #Black Empty Image for morphological operations
    opened = cv2.morphologyEx(img,cv2.MORPH_OPEN,kernel)
    #--------Opened Image
    eroded = cv2.erode(opened,kernel,iterations = 1)
    #--------Opened is eroded
    dilated = cv2.dilate(eroded,kernel,iterations = 1)
    #cv2.imwrite('dilated.jpg',dilated)
    #--------eroded is dilated
    closed = cv2.morphologyEx(dilated,cv2.MORPH_CLOSE,kernel)
    #blur = cv2.cvtColor(dilated,cv2.COLOR_BGR2GRAY)
    # Applying Gaussian Filter on the eroded image
    blur = cv2.GaussianBlur(eroded,(5,5),0)
    blur = ~blur
    # The Image Inversion is done to nullify hidden holes
    # Finding the Threshold value from the obtained Blur picture
    ret,thresholdval = cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
    # Obtaining contours of the inverted Image
    contours, hierarchy = cv2.findContours(thresholdval, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
    # Drawing Contours
    #cv2.drawContours(colorimg, contours, -1, (0, 0, 255), 2)
    # Finding EXTREME POINTS IN CONTOUR parts
    c = max(contours, key=cv2.contourArea)
    # Computing MAX extremities
    extLeft = tuple(c[c[:, :, 0].argmin()][0])
    extRight = tuple(c[c[:, :, 0].argmax()][0])
    extTop = tuple(c[c[:, :, 1].argmin()][0])
    extBot = tuple(c[c[:, :, 1].argmax()][0])
    '''
```


file1.py - C:\Users\ADMIN\Documents\Python Scripts\MINIPROJECT_EXECUTION\SERVER\file1.py (3.8.1)
File Edit Format Run Options Window Help

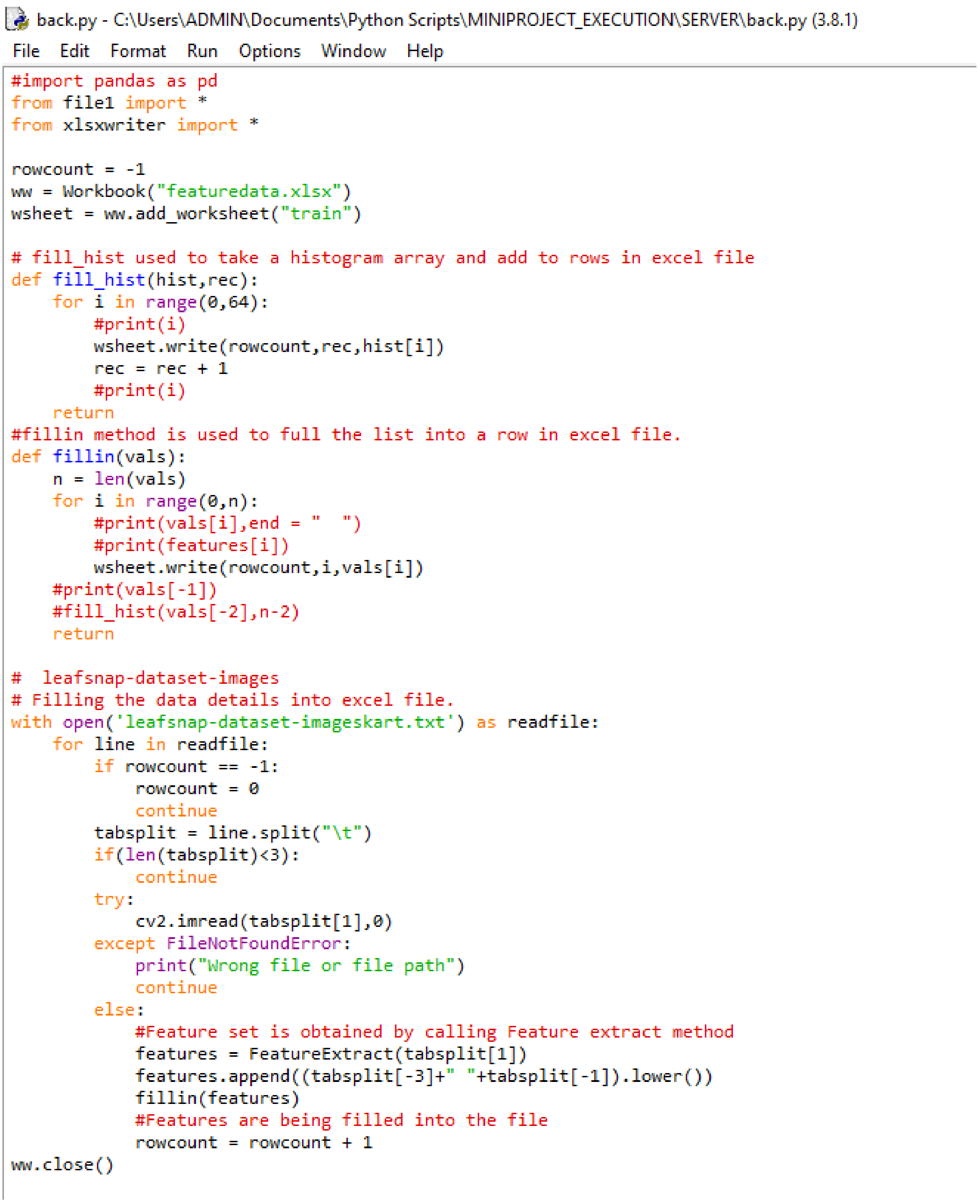
```
    area = 0
    area_fullc= 0
    #Select longest contour as this should be the casing
    lengthC=0
    ID=-1
    idCounter=-1
    for x in contours:
        idCounter=idCounter+1
        if len(x) > lengthC:
            lengthC=len(x)
        ID=idCounter
    if ID != -1:
        cnt = contours[ID]
        cntFull=cnt.copy()
    #approximate the contour, where epsilon is the distance to the original contour
        cnt = cv2.approxPolyDP(cnt, epsilon=1, closed=True)
    #add the first point as the last point, to ensure it is closed
        lenCnt=len(cnt)
        cnt= np.append(cnt, [[cnt[0][0][0], cnt[0][0][1]]])
        cnt=np.reshape(cnt, (lenCnt+1,1, 2))

        lenCntFull=len(cntFull)
        cntFull= np.append(cntFull, [[cntFull[0][0][0], cntFull[0][0][1]]])
        cntFull=np.reshape(cntFull, (lenCntFull+1,1, 2))
    #find the moments
        M = cv2.moments(cnt)
        MFull = cv2.moments(cntFull)
        area = M['m00']
        area_fullc = MFull['m00']
    # Moments generated are used to generate area enclosed by Contours
    perimeter = 0
    for c in contours:
        peri = cv2.arcLength(c, True)
        approx = cv2.approxPolyDP(c, 0.02 * peri, True)
        cv2.drawContours(colorimg, [approx], -1, (0, 255, 0), 2)
        perimeter = perimeter+cv2.arcLength(c,True)
    #Constructing Hull casing to the contours generated
    hull = []
    # calculate points for each contour
    for i in range(len(contours)):
    # creating convex hull object for each contour
        hull.append(cv2.convexHull(contours[i], False))
    # create an empty black image
    drawing = np.zeros((thresholdval.shape[0], thresholdval.shape[1], 3), np.uint8)
    # draw contours and hull points
    for i in range(len(contours)):
        color_contours = (0, 255, 0) # green - color for contours
        color = (255, 0, 0) # blue - color for convex hull
    # draw ith contour
        #cv2.drawContours(drawing, contours, i, color_contours, 1, 8, hierarchy)
    # draw ith convex hull object
    #cv2.drawContours(colorimg, hull, i, color, 1, 8)
    # Contours are marked and proceeded for Feature Extraction
    ''' Required shape feature generating things are extracted  '''
```



file1.py - C:\Users\ADMIN\Documents\Python Scripts\MINIPROJECT_EXECUTION\SERVER\file1.py (3.8.1)
File Edit Format Run Options Window Help

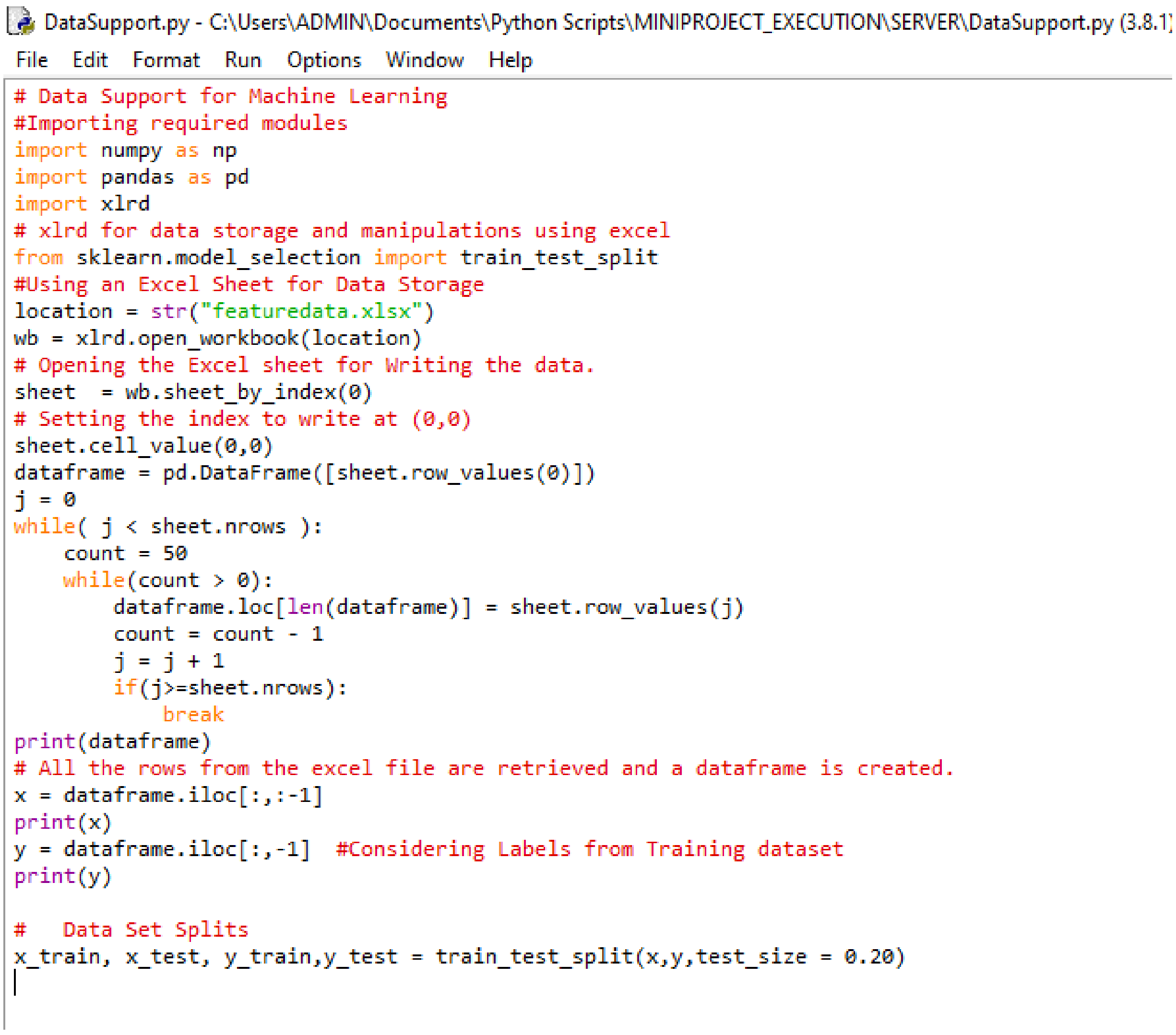
```
    #Feature Extraction
    # 1 ... Aspect Ratio = width / length
    length = math.hypot(extRight[0]-extLeft[0] , extRight[1]-extLeft[1])
    # Length of the Image is found from the extremes generated
    width = math.hypot(extTop[0]-extBot[0] , extTop[1]-extBot[1])
    # width is found from the extremes generated
    if (length < width):
        xxxx = length
        length = width
        width = xxxx
    aspectratio = width / length
    # Aspect Ratio is Extracted with Leaf
    # 2 ... White area Ratio = Area of Leaf / ( Width * Lenght)  -> ( EXTENT )
    whitearearatio = area / (width * length)
    # 3 ... perimetertoarea ratio = leaf perimeter / leaf area
    leafarea = PolyArea2D(giveCoordinates(contours)[0])
    # Leaf area is generated
    perimetertoarea = 0
    if(leafarea > 0.0):
        perimetertoarea = float(perimeter) / leafarea
        # perimeter to area ratio is generated
    # 4 ... perimeter to hull ratio = perimeter of hull / perimeter of leaf
    hullcoordinates = giveCoordinates(hull)
    hullarea = PolyArea2D(hullcoordinates[0])
    hullperimeter = 0
    for c in hull:
        peri = cv2.arcLength(c, True)
        approx = cv2.approxPolyDP(c, 0.02 * peri, True)
        #cv2.drawContours(colorimg, [approx], -1, (0, 255, 0), 2)
        hullperimeter = hullperimeter+cv2.arcLength(c,True)
    # Hull perimeter generated
    perimetertohull = hullperimeter/perimeter
    # Perimeter to Hull Ratio generated
    # 5 ... HULL Area ratio = leaf area / Hull area    ( SOLIDITY )
    hullarearatio = 0
    if hullarea == 0.0:
        hullarearatio = 0
    else:
        hullarearatio = leafarea / hullarea
    #Solidity of the leaf is generated
    # 6 ... Circularity = 4*pi*area / hill perimeter**2
    circularity = (4*22/7)*leafarea/(hullperimeter**2)
    # Circularity of the Leaf is computed
    # 7 ... COLOR HIHSTOGRAM
    # 8 Equivalent Diameter = sqrt((4* contour area)/pi)
    equivalentdiameter = np.sqrt(4*leafarea/np.pi)
    #The Leaf's Equivalent diameter is computed for size of leaf
    # Feature Record hold all the Feature extracted from the Leaf Image
    featurerecord = [perimetertohull,aspectratio,circularity,equivalentdiameter,white
    featurerecord.extend(list(hist))
    return featurerecord
    #print(list(featurerecord))
```

Module “back” for Feature Handling and Data Handling


back.py - C:\Users\ADMIN\Documents\Python Scripts\MINIPROJECT_EXECUTION\SERVER\back.py (3.8.1)
File Edit Format Run Options Window Help

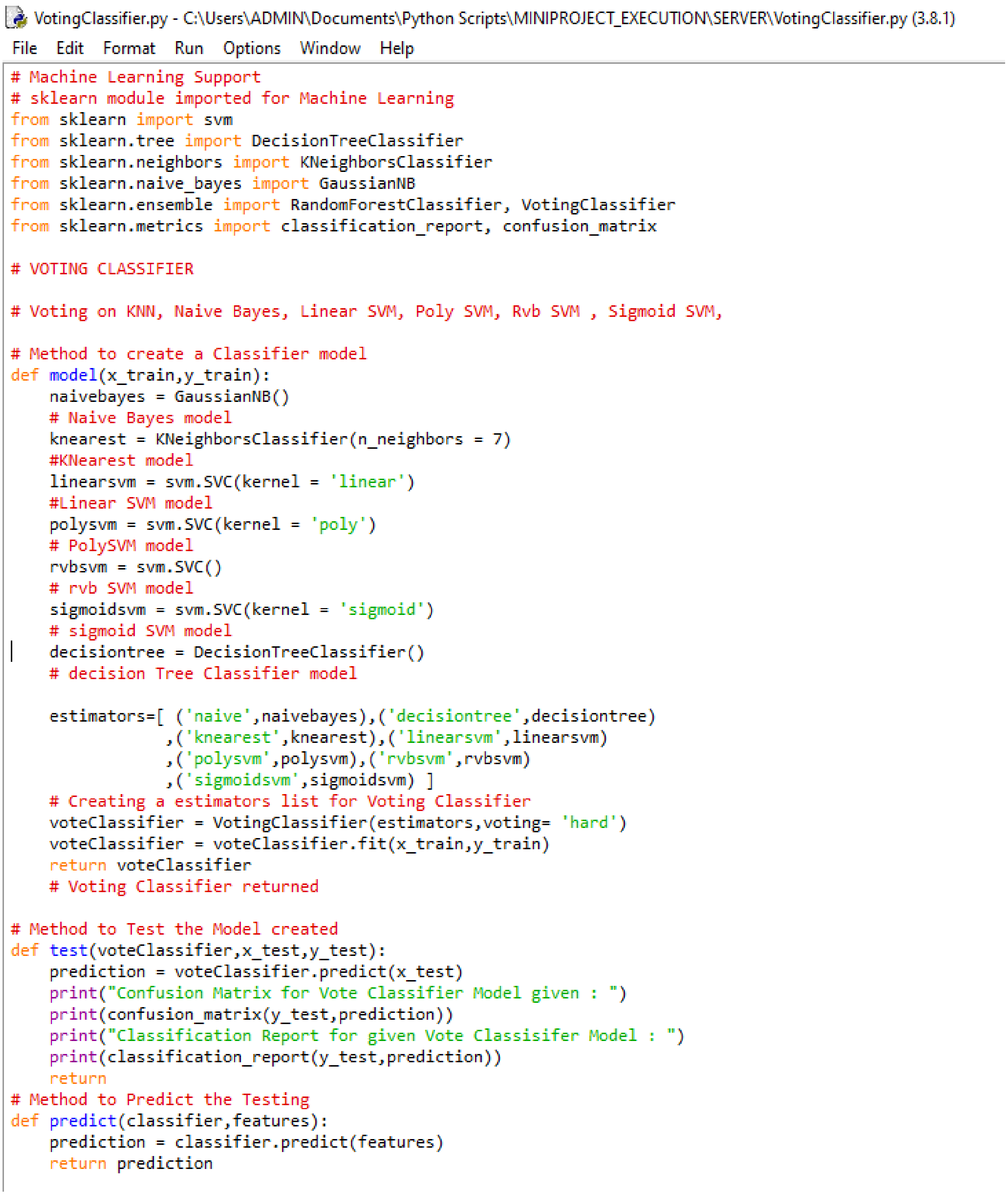
```
#import pandas as pd
from file1 import *
from xlsxwriter import *

rowcount = -1
ww = Workbook("featuredata.xlsx")
wsheet = ww.add_worksheet("train")

# fill_hist used to take a histogram array and add to rows in excel file
def fill_hist(hist,rec):
    for i in range(0,64):
        #print(i)
        wsheet.write(rowcount,rec,hist[i])
        rec = rec + 1
        #print(i)
    return
#fillin method is used to full the list into a row in excel file.
def fillin(vals):
    n = len(vals)
    for i in range(0,n):
        #print(vals[i],end = "  ")
        #print(features[i])
        wsheet.write(rowcount,i,vals[i])
    #print(vals[-1])
    #fill_hist(vals[-2],n-2)
    return

#  leafsnap-dataset-images
# Filling the data details into excel file.
with open('leafsnap-dataset-imageskart.txt') as readfile:
    for line in readfile:
        if rowcount == -1:
            rowcount = 0
            continue
        tabsplit = line.split("\t")
        if(len(tabsplit)<3):
            continue
        try:
            cv2.imread(tabsplit[1],0)
        except FileNotFoundError:
            print("Wrong file or file path")
            continue
        else:
            #Feature set is obtained by calling Feature extract method
            features = FeatureExtract(tabsplit[1])
            features.append((tabsplit[-3]+" "+tabsplit[-1]).lower())
            fillin(features)
            #Features are being filled into the file
            rowcount = rowcount + 1
ww.close()
```

Module DataSupport


DataSupport.py - C:\Users\ADMIN\Documents\Python Scripts\MINIPROJECT_EXECUTION\SERVER\DataSupport.py (3.8.1)
File Edit Format Run Options Window Help

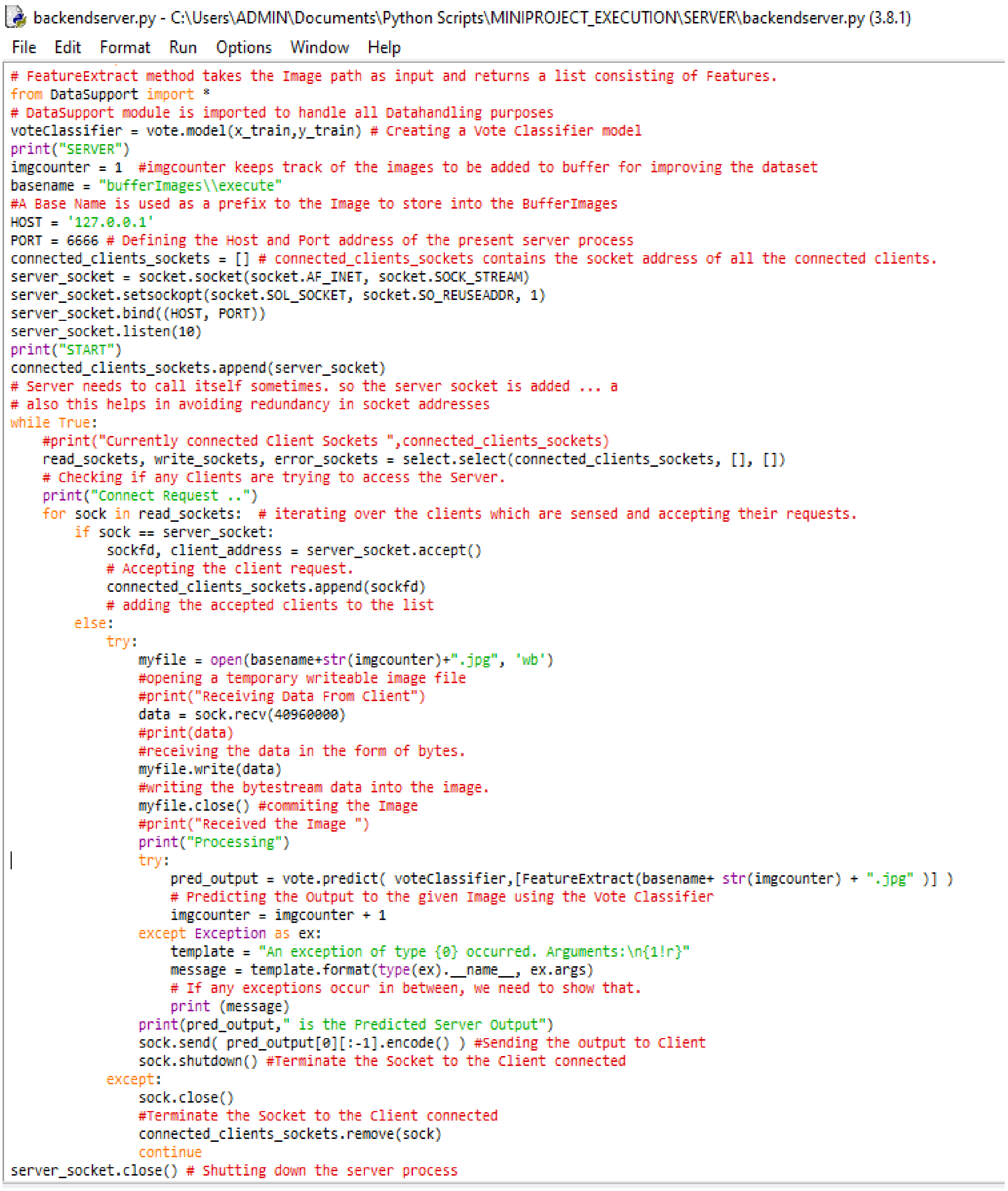
```
# Data Support for Machine Learning
#Importing required modules
import numpy as np
import pandas as pd
import xlrd
# xlrd for data storage and manipulations using excel
from sklearn.model_selection import train_test_split
#Using an Excel Sheet for Data Storage
location = str("featuredata.xlsx")
wb = xlrd.open_workbook(location)
# Opening the Excel sheet for Writing the data.
sheet  = wb.sheet_by_index(0)
# Setting the index to write at (0,0)
sheet.cell_value(0,0)
dataframe = pd.DataFrame([sheet.row_values(0)])
j = 0
while( j < sheet.nrows ):
    count = 50
    while(count > 0):
        dataframe.loc[len(dataframe)] = sheet.row_values(j)
        count = count - 1
        j = j + 1
        if(j>=sheet.nrows):
            break
print(dataframe)
# All the rows from the excel file are retrieved and a dataframe is created.
x = dataframe.iloc[:,:-1]
print(x)
y = dataframe.iloc[:,-1]  #Considering Labels from Training dataset
print(y)

#   Data Set Splits
x_train, x_test, y_train,y_test = train_test_split(x,y,test_size = 0.20)
```

Module Voting Classifier


VotingClassifier.py - C:\Users\ADMIN\Documents\Python Scripts\MINIPROJECT_EXECUTION\SERVER\VotingClassifier.py (3.8.1)
File Edit Format Run Options Window Help

```
# Machine Learning Support
# sklearn module imported for Machine Learning
from sklearn import svm
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.metrics import classification_report, confusion_matrix

# VOTING CLASSIFIER

# Voting on KNN, Naive Bayes, Linear SVM, Poly SVM, Rvb SVM , Sigmoid SVM,

# Method to create a Classifier model
def model(x_train,y_train):
    naivebayes = GaussianNB()
    # Naive Bayes model
    knearest = KNeighborsClassifier(n_neighbors = 7)
    #KNearest model
    linearsvm = svm.SVC(kernel = 'linear')
    #Linear SVM model
    polysvm = svm.SVC(kernel = 'poly')
    # PolySVM model
    rvbsvm = svm.SVC()
    # rvb SVM model
    sigmoidsvm = svm.SVC(kernel = 'sigmoid')
    # sigmoid SVM model
    decisiontree = DecisionTreeClassifier()
    # decision Tree Classifier model

    estimators=[ ('naive',naivebayes),('decisiontree',decisiontree)
                ,('knearest',knearest),('linearsvm',linearsvm)
                ,('polysvm',polysvm),('rvbsvm',rvbsvm)
                ,('sigmoidsvm',sigmoidsvm) ]
    # Creating a estimators list for Voting Classifier
    voteClassifier = VotingClassifier(estimators,voting= 'hard')
    voteClassifier = voteClassifier.fit(x_train,y_train)
    return voteClassifier
    # Voting Classifier returned

# Method to Test the Model created
def test(voteClassifier,x_test,y_test):
    prediction = voteClassifier.predict(x_test)
    print("Confusion Matrix for Vote Classifier Model given : ")
    print(confusion_matrix(y_test,prediction))
    print("Classification Report for given Vote Classisifer Model : ")
    print(classification_report(y_test,prediction))
    return
# Method to Predict the Testing
def predict(classifier,features):
    prediction = classifier.predict(features)
    return prediction
```

Module backend server – Execution


backendserver.py - C:\Users\ADMIN\Documents\Python Scripts\MINIPROJECT_EXECUTION\SERVER\backendserver.py (3.8.1)
File Edit Format Run Options Window Help

```
# FeatureExtract method takes the Image path as input and returns a list consisting of Features.
from DataSupport import *
# DataSupport module is imported to handle all Datahandling purposes
voteClassifier = vote.model(x_train,y_train) # Creating a Vote Classifier model
print("SERVER")
imgcounter = 1  #imgcounter keeps track of the images to be added to buffer for improving the dataset
basename = "bufferImages\\execute"
#A Base Name is used as a prefix to the Image to store into the BufferImages
HOST = '127.0.0.1'
PORT = 6666 # Defining the Host and Port address of the present server process
connected_clients_sockets = [] # connected_clients_sockets contains the socket address of all the connected clients.
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
server_socket.bind((HOST, PORT))
server_socket.listen(10)
print("START")
connected_clients_sockets.append(server_socket)
# Server needs to call itself sometimes. so the server socket is added ... a
# also this helps in avoiding redundancy in socket addresses
while True:
    #print("Currently connected Client Sockets ",connected_clients_sockets)
    read_sockets, write_sockets, error_sockets = select.select(connected_clients_sockets, [], [])
    # Checking if any Clients are trying to access the Server.
    print("Connect Request ..")
    for sock in read_sockets:  # iterating over the clients which are sensed and accepting their requests.
        if sock == server_socket:
            sockfd, client_address = server_socket.accept()
            # Accepting the client request.
            connected_clients_sockets.append(sockfd)
            # adding the accepted clients to the list
        else:
            try:
                myfile = open(basename+str(imgcounter)+".jpg", 'wb')
                #opening a temporary writeable image file
                #print("Receiving Data From Client")
                data = sock.recv(40960000)
                #print(data)
                #receiving the data in the form of bytes.
                myfile.write(data)
                #writing the bytestream data into the image.
                myfile.close() #commiting the Image
                #print("Received the Image ")
                print("Processing")
                try:
                    pred_output = vote.predict( voteClassifier,[FeatureExtract(basename+ str(imgcounter) + ".jpg" )] )
                    # Predicting the Output to the given Image using the Vote Classifier
                    imgcounter = imgcounter + 1
                except Exception as ex:
                    template = "An exception of type {0} occurred. Arguments:\n{1!r}"
                    message = template.format(type(ex).__name__, ex.args)
                    # If any exceptions occur in between, we need to show that.
                    print (message)
                print(pred_output," is the Predicted Server Output")
                sock.send( pred_output[0][:-1].encode() ) #Sending the output to Client
                sock.shutdown() #Terminate the Socket to the Client connected
            except:
                sock.close()
                #Terminate the Socket to the Client connected
                connected_clients_sockets.remove(sock)
                continue
server_socket.close() # Shutting down the server process
```