An Effective Accuracy-Based Disease Prediction on Paddy Cultivation Using Machine Learning

A PROJECT REPORT

Submitted by,

Avidi Rohith Krishna - 20201CSE0137
N Hemanth Reddy -20201CSE0167
Anadan Tharun Raj- 20201CSE0151
Roshan M-20201CSE0140

Under the guidance of,

Dr. A VIJAYA KUMAR

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING,

At

PRESIDENCY UNIVERSITY

BENGALURU

JANUARY 2024

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE ENGINEERING & INFORMATION SCIENCE

CERTIFICATE

This is to certify that the Project report **"An effective accuracy-based disease prediction on paddy cultivation using machine learning"** being submitted by "Avidi Rohith Krishna", "N Hemanth Reddy", "Anadan Tharun Raj", "Roshan M" bearing roll number(s) "20201CSE0137", "20201CSE0167", "20201CSE0151", "20201CSE0140" in partial fulfilment of requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out under my supervision.

Dr. A VIJAYA KUMAR
Professor
School of CSE&IS
Presidency University

Dr. PALLAVI R
Associate Professor & HoD
School of CSE&IS
Presidency University

Dr. C. KALAIARASAN
Associate Dean
School of CSE&IS
Presidency University

Dr. SHAKKEERA L
Associate Dean
School of CSE&IS
Presidency University

Dr. SAMEERUDDIN KHAN
Dean
School of CSE&IS
Presidency University

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE ENGINEERING & INFORMATION SCIENCE

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled **An effective accuracy based disease prediction on paddy cultivation using machine learning** in partial fulfilment for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering**, is a record of our own investigations carried under the guidance of

Dr. A Vijayakumar, Professor, School of Computer Science Engineering & Information Science, Presidency University, Bengaluru.

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

Student Name	Roll No	Signature
Avidi Rohith Krishna	**20201CSE0137**	
N Hemanth Reddy	**20201CSE0167**	
Anadan Tharun Raj	**20201CSE0151**	
Roshan M	**20201CSE040**	

ABSTRACT

The project has entitled as **“An effective accuracy-based disease prediction on paddy cultivation using machine learning”**, and developed by using Python as front end. Rice leaf diseases are a major problem in economic and production losses in the agricultural industry worldwide. In this project, an image processing approach is proposed for identifying passion rice leaf diseases based on convolutional neural network. According to the CNN algorithm, rice leaf image details are taken by the existing packages from the front end used in this project. However, it can take a few moments. So, this proposed system can be used to identify rice leaf diseases quickly and automatically.

This proposed approach is composed of the following main steps that getting input image, Image Preprocessing, identifying affected places, highlight those affected places, verifying training set, showing result. Few types of rice leaf diseases. This approach was tested according to rice leaf disease type and its' stages, such as fresh and affected.

The algorithm was used for detecting the disease of the rice leaf. Images were provided for training. Before the image processing, images were converted to color models, because of find out the most suitable color model for this approach. Local Binary Pattern was used for feature extraction and Support erosion method was used for creating the model. According to this approach, rice leaf diseases can be identified in the average accuracy of 79% and its' stage can be identified in average accuracy 66%.

ACKNOWLEDGEMENT

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Dean, School of Computer Science Engineering & Information Science, Presidency University for getting us permission to undergo the project.

We record our heartfelt gratitude to our beloved Associate Deans **Dr. Kalaiarasan C and Dr. Shakkeera L,** School of Computer Science Engineering & Information Science, Presidency University and

Dr. Pallavi R, Head of the Department, School of Computer Science Engineering & Information Science, Presidency University for rendering timely help for the successful completion of this project.

We are greatly indebted to our guide **Dr. A Vijaya Kumar, Professor**, School of Computer Science Engineering & Information Science, Presidency University for his inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We would like to convey our gratitude and heartfelt thanks to the University Project-II Coordinators **Dr. Sanjeev P Kaulgud, Dr. Mrutyunjaya MS** and also the department Project Coordinators **Mohammed Zia Ur Rahaman**.

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

Avidi Rohith Krishna

N Hemanth Reddy

Anadan Tharun Raj

Roshan M

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER-1
INTRODUCTION

1.1 Introduction

Toward the start of the 21st century, paddy (Oryza sativa species) is as yet the main oat in human food frameworks and the principle wellspring of energy and a critical portion of proteins devoured by very nearly three billion peoples. More than 90% of the world's paddy is produced in the AsiaPacific Region In Bangladesh, paddy is the key producing crop food, about 75% of the absolute edited region, and over 80% of the all-out watered zone is planted to rice. As a result, paddy plays an important role in the subsistence of the people of Bangladesh. Most of the time farmers have to face various problems in paddy cultivation such as Damage to arable land, increased population, climate change, pests, and diseases, etc. Due to these various problems, farmers are becoming uninterested in paddy cultivation nowadays. This paper has focused only on the pests and diseases to the various problems of rice cultivation.

There are three main types of paddy diseases such as bacterial disease, fungal disease, and miscellaneous diseases. These include subcategories like bacterial blight, bacterial leaf streak, brown spot, leaf smut, leaf scald, panicle blight, bronzing, etc.. Note that the incidence of diseases has of late gotten extreme because of the unfavorable impacts of climate change, especially the ascent in temperature (IPCC, 2007). It is assessed that 4-14% of rice yield in Bangladesh is lost each year by various pests and diseases. Bacterial leaf curse (BLB) and brown spot are currently genuine infections in rice. But the innovations technologies to pests and diseases are still restricted. Generally, the manual recognition of paddy disease is the unaided eye perception of specialists which burns-through additional time, costly on huge homesteads. It is hard to measure and some of the time it delivers a mistake while distinguishing the disease type. Because of the ignorance of appropriate administration to redress paddy plant leaf disease, paddy production is being decreased as of late. To overcome this, appropriate and quick recognition measures are required for the diagnosis of paddy leaf diseases. This work mainly focused on the five most common paddy leaf diseases named Brown spot, healthy leaf, leaf blast, bacterial blight, leaf smut.

The Revolution of Artificial Intelligence has made it easier to maintain a standard of living. Like all other sectors, there is no shortage of AI contributions in the agriculture sector. Technology has made it much easier to solve many problems in agriculture, plant disease is one of them. Currently, it can do a lot of disease detection using machine learning and deep learning. Despite some limitations, it has largely succeeded. As a result, the farmer himself can detect paddy disease in his land without the help of an expert. Technology is going to bring many more revolutions in the agriculture sector in the future.

According to a survey conducted in 1979-1981, 20 diseases of paddy have been reported in Bangladesh were to exist paddy leaf diseases, among which 13 diseases were identified as the important ones. In 2019 according to the rice knowledge bank of Bangladesh bacterial leaf blast is one of the most deleterious diseases. In Bangladesh, leaf blast, leaf blight, brown spots are very common diseases in paddy cultivation.

In this project, it focused on four paddy diseases as Brown Spot, Leaf Blight, Leaf Smut, Bacterial Leaf Blast, and one healthy leaf. This paper selected the Deep Convolutional Neural Network and trained the dataset on the four CNN based pre-trained models.

This research can be carried forward with more varieties of leaf diseases and more fine-tuned CNN models with the expectation of finding better accuracy and ensuring faster detection. A detailed comprehensive study is a must to understand the factors affecting the detection of plant diseases, like the classes' datasets, and size of datasets, learning rate, illumination, etc. The basic form of paddy plant diseases changes with the passage of time or the background of the images, images with color issues, hence, these convolutional neural network models should be modified to enable them to detect and classify diseases during these complex or problematic situations.

The application become useful if the below enhancements are made in future.

1. In future the application is designed as web service, it can be integrated in many web sites.
2. More accuracy can be detected using various machine learning algorithms

The application is developed such that above said enhancements can be integrated with current modules.

1.2 Feasibility Study

The feasibility study deals with all the analysis that takes up in developing the project. Each structure has to be thought of in the developing of the project, as it has to serve the end user in a user-friendly manner. One must know the type of information to be gathered and the system analysis consist of collecting, Organizing and evaluating facts about a system and its

Three considerations involved in feasibility analysis are

- Economical Feasibility
- Operational Feasibility
- Technical Feasibility

1.2.1 Economic Feasibility

The organization has to buy a personal computer with a keyboard and a mouse, this is a direct cost. There are many direct benefits of covering the manual system to computerized system. The user can be given responses on asking questions, justification of any capital outlay is that it will reduce expenditure or improve the quality of service to the user. The users who have basic knowledge about Microsoft technologies can use this service by accessing the service provided in the web site.

1.2.2 Operational Feasibility

The Proposed system accessing process to solves problems what occurred in existing system. The current day-to-day operations of the organization can be fit into this system. Mainly operational feasibility should include on analysis of how the proposed system will affects the organizational structures and procedures.

The proposed system requires less human interaction and anybody who has the basic computer devices can access these services and operate on their needed part of the services to get the services that are needed by the user.

1.2.3 Technical Feasibility

The cost and benefit analysis may be concluded that computerized system is favorable in today's fast moving world. The assessment of technical feasibility must be based on an outline design of the system requirements in terms of input, output, files, programs and procedure.

The project aims to provide the faster information sharing using the web application and to reduce the difficulties involved in request handling that are given by users of the website. The current system aims to overcome the problems of the existing system. The current system is to reduce the technical skill requirements so that more number of users can access the application

1.3 System Specification

1.3.1 Hardware Requirements

This section gives the details and specification of the hardware on which the system is expected to work.

Processor	:	Dual Core Processor
RAM	:	2 GB RAM
Monitor	:	LED
Hard disk	:	80 GB
Keyboard	:	Standard102 keys
Mouse	:	LOGI TECH (3 Buttons)

1.3.2 Software Requirements

This section gives the details of the software that are used for the development.

Environment	:	Python 3.7 IDE
Coding Language	:	Python
Operating System	:	Windows 7

CHAPTER-2
LITERATURE SURVEY

1.Title: Pomegranate Disease Detection Using Image Processing. India, Elsevier B.V, 2019.

Author : Bhange, M. & Hingoliwala, H. A

Recently, many people have done researches for detecting rice leaf and vegetable diseases using image processing and deep learning. According to the research paper, authors had used image processing technology to identify the pomegranate diseases. Image preprocessing was the first step of the methodology. Image resizing was done under the image preprocessing. Because digital camera had been used to capture the images in this study. The size of those images was very large and take more time to process. So all images were resized to 300 x 300 PX. Morphology, colour and CCV features were used for feature extraction. K-means clustering technique was used for partitioning the training dataset according to their features. After the clustering, SVM was used for classification to identify the image as infected or non-infected. An intent search technique was provided to find the user intention. The best result was got using morphology feature extraction. Experimental evaluation of this approach was effective and 82% accurate to identify pomegranate disease.

2.Title: Adapted Approach for Rice leaf Disease Identification using Images. India, International Journal of Computer Vision and Image Processing, 2018

Author : Dubey, S. R. & Jalal. A. S.

The authors presented the image processing based approach for rice leaf disease detection. First, read input image and transformed it from RGB to L*a*b colour space. Because the colour information in the L*a*b colour space is stored in only two channels. Input images were partitioned into four segments using K-means cluster in this research. Because the empirical observations it was found that using 3 or 4 clusters yield good segmentation results. GCH, LBP, CCV and CLBP were used for feature extraction. More accurate results could be taken using CLBP feature extraction technique. K-means clustering was used for segmentation. Those segmented images were extracted to label each pixel in the image. SVM algorithm was used for training and classification of rice leaf disease. Authors used rice as a test case and evaluated the classification model for three types of rice diseases, which were rice rot, rice blotch and rice scab. The accuracy of this approach was achieved by up to 93%.

3. Title: Classifier Based Grape Leaf Disease Detection. India, Conference on Advances in Signal Processing (CASP), 2017

Author : Padol, P. B.

The author had used SVM classification for identifying and classifying the grape leaf diseases. Grape leaf images were taken using a digital camera and those were used to both training and testing the system. Collected images included the leaves infected by Powdery Mildew and Downy Mildew. Removing background noise and resizing to 300*300 PX to improve the image quality were done under the image preprocessing. Gaussian filtering had been used to remove noise in the image. K-means clustering was used for segmenting an image into three groups. Features were extracted based on both colour and texture for taking accurate disease information. Finally, the classification model was used to detect the leaf disease. LSVM was used in this research for the classification of leaf diseases. This system could detect and classify the examined disease successfully. The accuracy of this system was 88.89%.

4. Title: Leaf disease detection using image processing. Vellore, Journal of Chemical and Pharmaceutical Sciences, 2017.

Author : Sujatha, R., Kumar, Y. S., Akhil, G, U.

Authors had used image processing technology for identifying the leaf diseases. First authors selected the plants, which were affected by the disease and then took the snapshot of the diseased leaf. Contrast enhancement and converting RGB to HIS was done under the image preprocessing step. K-means clustering algorithm was used to cluster the object based on the feature of leaf into k number of groups. SVM algorithm had been used in this system for classification purpose. SVM is a statistical learning-based solver. Finally, when entered a diseased leaf image to a system, the system was able to detect the leaf disease successfully.

5. Title: Pomegranate Disease Detection Using Image Processing Techniques. Pune, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 2016.

Author : Khot, S. T., Supriya, P., Gitanjali, M., & Vidya, L.

Authors had presented the image processing based system to identify pomegranate rice leaf diseases. This rice leaf is mainly affected by Bacterial Blight, Anthracnose and Alterneria. After capturing the disease images, image resizing, filtering, segmentation, morphological features were used to preprocess the images. Image segmentation is the process of dividing the image into multiple parts. Colour-based segmentation was used in this research, such as clustering, YCbCr, RGB, L*a*b and HSV. However, the best performance in terms of segmentation error was achieved by the HSV and YCbCr. Morphology, texture and colour features were extracted for classification purpose. HIS colour model and colour histogram techniques had been used to colour feature extraction. Under the morphology feature extraction, boundary extraction was used to identify the region and shape. The eroded images were subtracted from the original image to extract a shape from healthy rice leaf image. Gabor filter was used to texture feature extraction. After the training and testing of images, diseased and non-diseased rice leafs were classified using Minimum Distance Classifier (MDC).

CHAPTER-3
RESEARCH GAPS OF EXISTING METHODS

3.1 Existing System

In existing system, normal infections of rice leaf are considered. Few diseases only can found out. The image processing based existing methodology is made out of the accompanying some state of the art color and texture features are extracted from the test image, then color and texture features are fused together and random forest classifier is used for diseases classification.

Drawbacks Of Existing System

- Accuracy is less.
- Time taken to process the image is high.
- Category of the disease was not found.
- Missed to process high pixel images.

CHAPTER-4
PROPOSED MOTHODOLOGY

4.1 Proposed System

For the rice leaf disease classification problem, precise image segmentation is required; otherwise the features of the non-infected region will dominate over the features of the infected region. In this approach CNN based image processing is preferred to detect the region of interest which is the infected part only. After processing the input image, features are extracted from the processed image of the rice leaf. Finally, training and the results are executed.

Advantage Of Proposed System

- Accuracy is high.
- Enhancing the value of rice leaf disease detection.
- Less time consuming.
- Finding category of the disease is done by highlighting the affected places.
- Applicable for both low and high pixel images.

4.2 Methodology

There are six phases in this methodology. Those are Image Acquisition, Image Preprocessing, Image Segmentation, Applying training dataset, Experimental results.

4.2.1 Image Acquisition

In this phase, the sample images are collected, which are required to train the classifier algorithm and build the classifier model. Yellowish or Reddish passion rice leaf variety was selected to take sample images. Because the yellowish variety is widely cultivated in our site. Healthy and diseased passion rice leaf images were taken by using mobile phone digital camera and used for both training and testing the classifier algorithm. Images were taken in different angles, under the different environmental and lighting conditions. The standard JPG format was used to store these images. In this study, images were collected from farms in different regions. Passion rice leafs infected by scab disease and woodiness virus that had been included in collected images.

4.2.2 Image Preprocessing

After the image acquisition, image processing was done for improving the image quality. All original passion rice leaf images were stored in one folder. Those images were named as we like our wish can take any value of numbers. Only horizontal images were rotated by 90 degrees and resized by 200x300 pixels. Vertical images were resized by 200x300 pixels and when the width and height of the image are same, those images were resized to 250x250 pixels. When the image size is too large, the processing task takes more time. After that, one of the noise reduction methods was used to remove the noises from images and increase the sharpness of images. Later, all preprocessed images were saved in a folder.

4.2.3 Image Segmentation

The third phase of the methodology is image segmentation. As the first step, all preprocessed images were converted into L*a*b, HSV, Grey color models and kept one in the original way (RGB). Because the identifying suitable color model for preprocessing is one of the outcomes of this research. After that, the image was converted to binary format. This format values were clustered using the CNN algorithm. According to the algorithm used an image segmentation were done.

4.2.4 Applying Training Set

The fifth phase of the methodology is applying training set images. The segmented output were done, which were created using feature extraction. However, three image sets were created to do experiments. Preparation of those image sets is discussed here. Field expertise support was taken for the categorization of images and each image were selected from the categorized sets of an image randomly.

4.2.5 Experimental Results

After converting the input image into histogram format, it starts compare to training set images. When the histogram format of the input image is matched 85% above with any one of training set images it will show the disease name as a heading of image shown.

CHAPTER-5

OBJECTIVES

The main objectives of this project are,

- To find the leaf is affected of fresh.
- To highlight the affected places from leaf.
- To find the name of the disease found.

CHAPTER-6
SYSTEM DESIGN & IMPLEMENTATION

6.1 Software Description

6.1.1 Python

Python is a dynamic, high level, free open source and interpreted programming language. It supports object-oriented programming as well as procedural oriented programming. In Python, we don’t need to declare the type of variable declaration is not needed because it is a dynamic typed language.

Features in Python

There are many features in Python, some of which are,

1. Easy to code
2. Free and Open Source
3. Object-Oriented Language
4. GUI Programming Support
5. High-Level Language
6. Extensible feature
7. Python is Portable language
8. Python is Integrated Language
9. Interpreted Language
10. Large Standard Library
11. Dynamically Typed Language

6.1.2 Python Applications

1. Web and Internet Development

Python lets you develop a web application without too much trouble. It has libraries for internet protocols like HTML and XML, JSON, e-mail processing, FTP, IMAP, and easy-to-use socket interface. Yet, the package index has more libraries:

- Requests – An HTTP client library
- BeautifulSoup – An HTML parser
- Feedparser – For parsing RSS/Atom feeds
- Paramiko – For implementing the SSH2 protocol
- Twisted Python – For asynchronous network programming

We also have a gamut of frameworks available. Some of these are- Django, Pyramid. We also get microframeworks like flask and bottle. We've discussed these in our write-up on an Introduction to Python Programming. We can also write CGI scripts, and we get advanced content management systems like Plone and Django CMS.

2. Applications of Python Programming in Desktop GUI

Most binary distributions of Python ship with Tk, a standard GUI library. It lets you draft a user interface for an application. Apart from that, some toolkits are available:

- wxWidgets
- Kivy – for writing multitouch applications
- Qt via pyqt or pyside

And then we have some platform-specific toolkits:

- GTK+
- Microsoft Foundation Classes through the win32 extensions
- Delphi

3. Science and Numeric Applications

This is one of the very common applications of python programming. With its power, it comes as no surprise that python finds its place in the scientific community. For this, we have:

- SciPy – A collection of packages for mathematics, science, and engineering.
- Pandas- A data-analysis and -modeling library
- IPython – A powerful shell for easy editing and recording of work sessions. It also supports visualizations and parallel computing.
- Software Carpentry Course – It teaches basic skills for scientific computing and running bootcamps. It also provides open-access teaching materials.
- Also, NumPy lets us deal with complex numerical calculations.

4. Software Development Application

Software developers make use of python as a support language. They use it for build-control and management, testing, and for a lot of other things:

- SCons – for build-control
- Buildbot, Apache Gump – for automated and continuous compilation and testing
- Roundup, Trac – for project management and bug-tracking.
- Roster of Integrated Development Environments

5. Python Applications in Education

Thanks to its simplicity, brevity, and large community, Python makes for a great introductory programming language. Applications of python programming in education has huge scope as it is a great language to teach in schools or even learn on your own. If you still haven't begun, we suggest you read up on what we have to say about the white and dark sides of Python. Also, check out Python Features.

6. Python Applications in Business

Python is also a great choice to develop ERP and e-commerce systems:

- Tryton – A three-tier, high-level general-purpose application platform.
- Odoo – management software with a range of business applications. With that, it's an all-rounder and forms a complete suite of enterprise-management applications in-effect.

7. Database Access

With Python, you have:

- Custom and ODBC interfaces to MySQL, Oracle, PostgreSQL, MS SQL Server, and others. These are freely available for download.
- Object databases like Durus and ZODB
- Standard Database API

6.1.3 Block Diagram

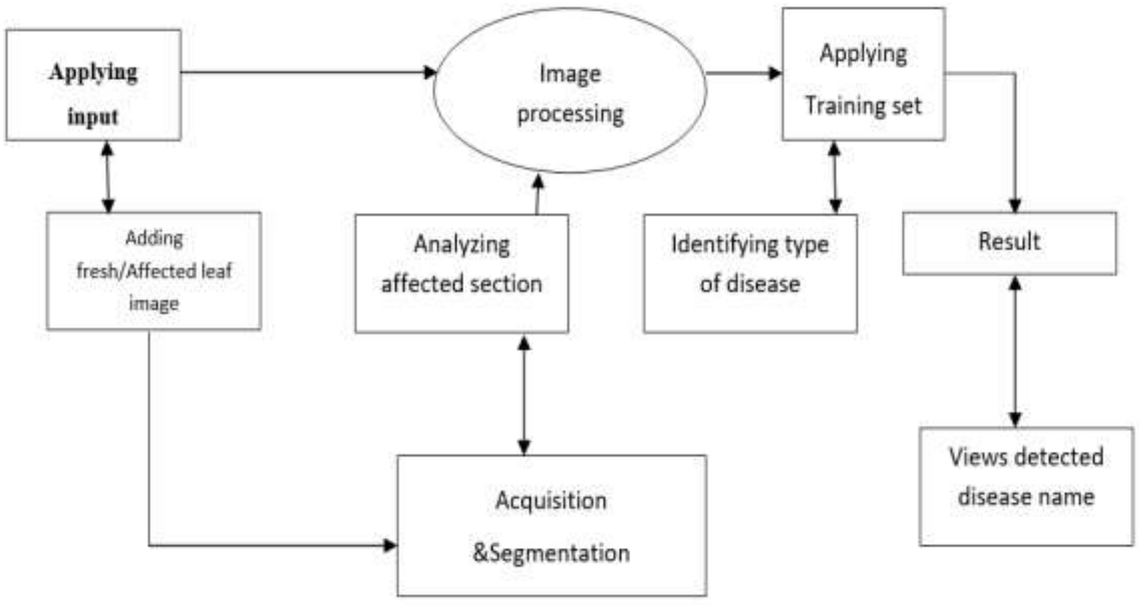

Applying input
Image processing
Applying Training set
Adding fresh/Affected leaf image
Analyzing affected section
Identifying type of disease
Result
Views detected disease name
Acquisition &Segmentation

Fig 6.1.3.1 block diagram

6.1.4 Flow Chart

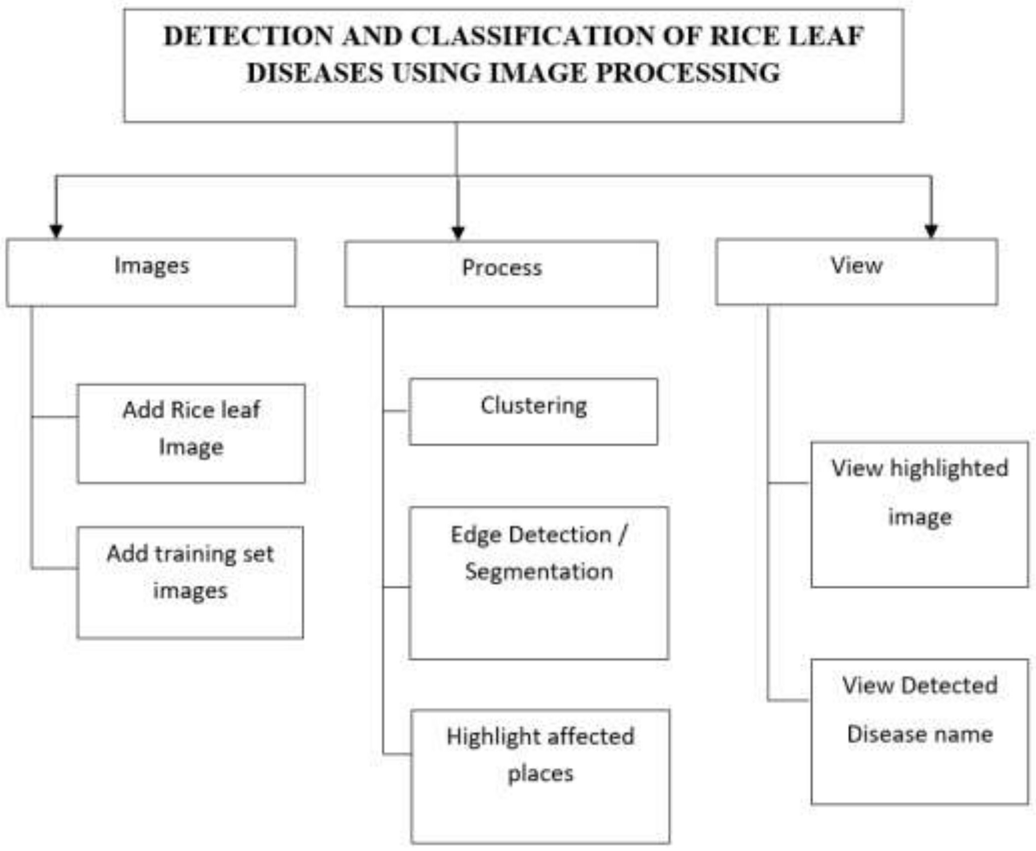

DETECTION AND CLASSIFICATION OF RICE LEAF DISEASES USING IMAGE PROCESSING
Images
Process
View
Add Rice leaf Image
Add training set images
Clustering
Edge Detection / Segmentation
Highlight affected places
View highlighted image
View Detected Disease name

Fig 6.1.4.1 flow chart

6.1.5 Input Design

The User Input In Python is written in python programming language, Developers often have a need to interact with users, either to get data or to provide some sort of result. Most programs today use a dialog box as a way of asking the user to provide some type of input.

A User Input On Python is important when we create a system or any project, because Getting User Input In Python is the way to know what are the process of the system or the project.

We created a typeface called python IDLE, which is based on changing the programming of PostScript fonts. PostScript is not just a “page description language”, but also a full programming language. In fact the original PostScript fonts were programs that you could mess with. You could open them in a text editor and just change things and see what would happen. So we learned how to do that. And as we progressed, programming became more and more important. I wanted to progress and learn more. I was not trained as a programmer, and traditional programming languages like C or Pascal were relatively difficult to work with.

I just want to get some things done in a way that allows me to focus on the problem instead of on the difficulties of programming.

I was in relative close contact with my brother Guido, who would often give me programming tips. Eventually, he introduced me to this language he was working on, called Python. He created Python as a scripting language for this experimental operating system that he was involved with at a research institution in the Netherlands. There was a good Mac version at that point, and he said, “Well, hey, why don’t you try that? And let’s see if the things that you would like to experiment with are easier with Python, because Python is just a friendlier language.” And indeed, it has a much simpler syntax than the more common languages at the time, especially compared to C. I mean, I was an okay programmer, but my main job is not programming. I just want to get some things done in a way that allows me to focus on the problem instead of on the difficulties of programming.

Python became the dominant programming language in the fields of type design and font engineering.

6.1.6 Python in the Type Design Process

It turned out that Python was indeed the perfect tool for the kind of things that I needed and wanted to do. I infected Erik van Blokland with it and he picked it up pretty quickly. I infected Petr van Blokland, Erik's older brother, with it, too. In the late 90s, we developed an application called RoboFog, which was a hybrid version of Fontographer 3.5 with a Python scripting layer added to it. It was initially for our own use, but we eventually did sell some licenses to others. For example, Jonathan Hoefler was really enthusiastic about that stuff. He learned Python and built a pretty elaborate toolchain for his foundry.

At some point the people at Adobe also picked up Python; they were using Python more and more in their internal tools. So the scope of what to do with Python in the context of type design grew bigger and bigger. FontLab followed by also choosing Python as a scripting language. So almost accidentally, Python became the dominant programming language in the fields of type design and font engineering.

6.1.7 Output Design

The output design is made with out using GUL. Because in this project the python language is utilized as a tool language. So on the one hand you have these options for experimentation, to create things that are just hard to do by hand, even on a computer, because they're too time consuming. For example to generate complex lines or patterns or shapes. You can often do that more effectively with an algorithm. There are all kinds of things you can play with that are outside your reach if you're doing things manually. And on the other hand there's font production. If you want to make things ready to sell on the market, there's a lot of different work involved there, that is potentially repetitive or boring or just really difficult. You need to do a lot of consistency checking, the kind of things that humans are usually pretty bad at, yet can be automated relatively easily with Python scripts.

To summarize, Python has two uses according to what you just said. One is the technical production of type faces. Getting typefaces ready for commercial use. The other side is more the visual exploration and executions visually that may be too difficult or cumbersome to do manually that a computer program like Python allows for easy usage and creation of.

Federal by Erik van Blokland http://letterror.com/fontcatalog/ltr-federal/ Yeah, that sounds about right. An good example of that is Federal, Erik van Blokland's typeface that he designed in the late 90s. It was inspired by the fat, high-contrast capital letters on dollar bills. He created shaded versions with lines in different resolutions with his own algorithms in RoboFog. That is an early example of how one can use this technology. It wouldn't be completely impossible to do manually, but it would be tedious and time consuming, therefore you tend to just not to go into that direction

6.2 System Testing

After the source code has been completed, documented as related data structures. Completed the project has to undergo testing and validation where there is subtitle and definite attempt to get errors.

The project developer treads lightly, designing and execution test that will demonstrates that the program works rather than uncovering errors, unfortunately errors will be present and if the project developer doesn't find them, the user will find out.

The project developer is always responsible for testing the individual units i.e. modules of the program. In many cases developer also conducts integration testing i.e. the testing step that leads to the construction of the complete program structure.

This project has undergone the following testing procedures to ensure its correctness.

- Unit testing
- User Acceptance Testing

6.2.1 Unit Testing

In unit testing, we have to test the programs making up the system. For this reason, Unit testing sometimes called as Program testing. The software units in a system are the modules and routines that are assembled and integrated to perform a specific function, Unit testing first on the modules independently of one another, to locate errors. This enables, to detect errors in coding and logic that are contained with the module alone. The testing was carried out during programming stage itself.

6.2.2 User Acceptance Testing

In these testing procedures the project is given to the customer to test whether all requirements have been fulfilled and after the user is fully satisfied. The project is perfectly ready. If the user makes request for any change and if they found any errors those all errors has to be taken into consideration and to be correct it to make a project a perfect project.

6.3 Integration Testing

Integration testing is done to test if the individual modules work together as one single unit. In integration testing, the individual modules that are to be integrated are available for testing. Thus the manual test data that used to test the interfaces replaced by that which in generated automatically from the various modules. It can be used for testing how the modules would actually interact with the proposed system. The modules are integrated and tested to reveal the problem interfaces.

6.4 System Implementation

When the initial design was done for the system, the client was consulted for the acceptance of the design so that further proceedings of the system development can be carried on. After the development of the system a demonstration was given to them about the working of the system. The aim of the system illustration was to identify any malfunction of the system.

After the management of the system was approved the system implemented in the concern, initially the system was run parallel with existing manual system. The system has been tested with live data and has proved to be error free and user friendly.

Implementation is the process of converting a new or revised system design into an operational one when the initial design was done by the system; a demonstration was given to the end user about the working system.

This process is uses to verify and identify any logical mess working of the system by feeding various combinations of test data. After the approval of the system by both end user and management the system was implemented. System implementation is made up of many activities. The six major activities are as follows.

6.5 Coding

Coding is the process of whereby the physical design specifications created by the analysis team turned into working computer code by the programming team.

6.6 Testing

Once the coding process is begin and proceed in parallel, as each program module can be tested.

6.7 Installation

Installation is the process during which the current system is replaced by the new system. This includes conversion of existing data, software, and documentation and work procedures to those consistent with the new system.

6.8 Documentation

It is result from the installation process, user guides provides the information of how the use the system and its flow.

6.9 Training And Support

Training plan is a strategy for training user so they quickly learn to the new system. The development of the training plan probably began earlier in the project.

CHAPTER-7

TIMELINE FOR EXECUTION OF PROJECT (GANTT CHART)

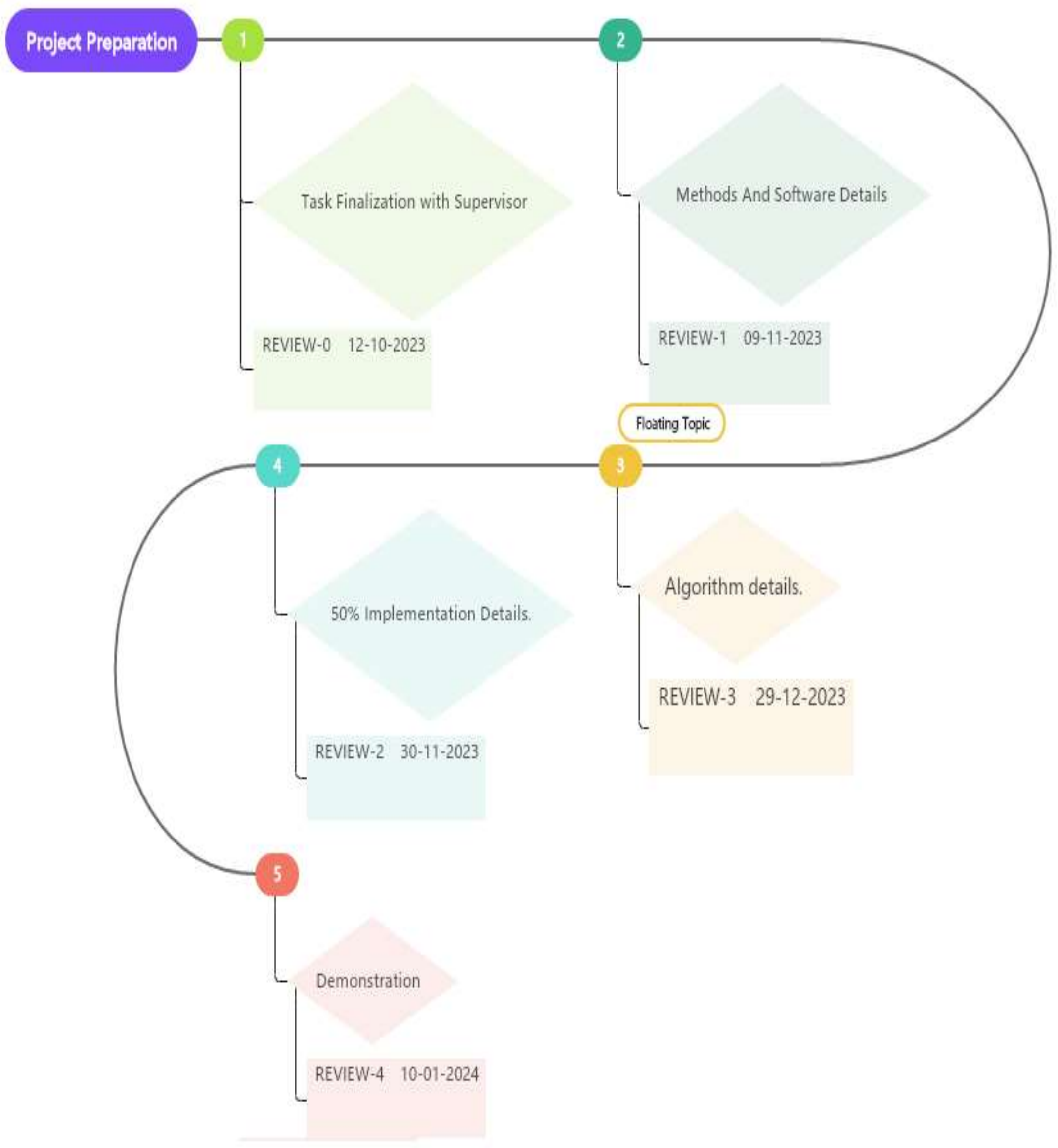
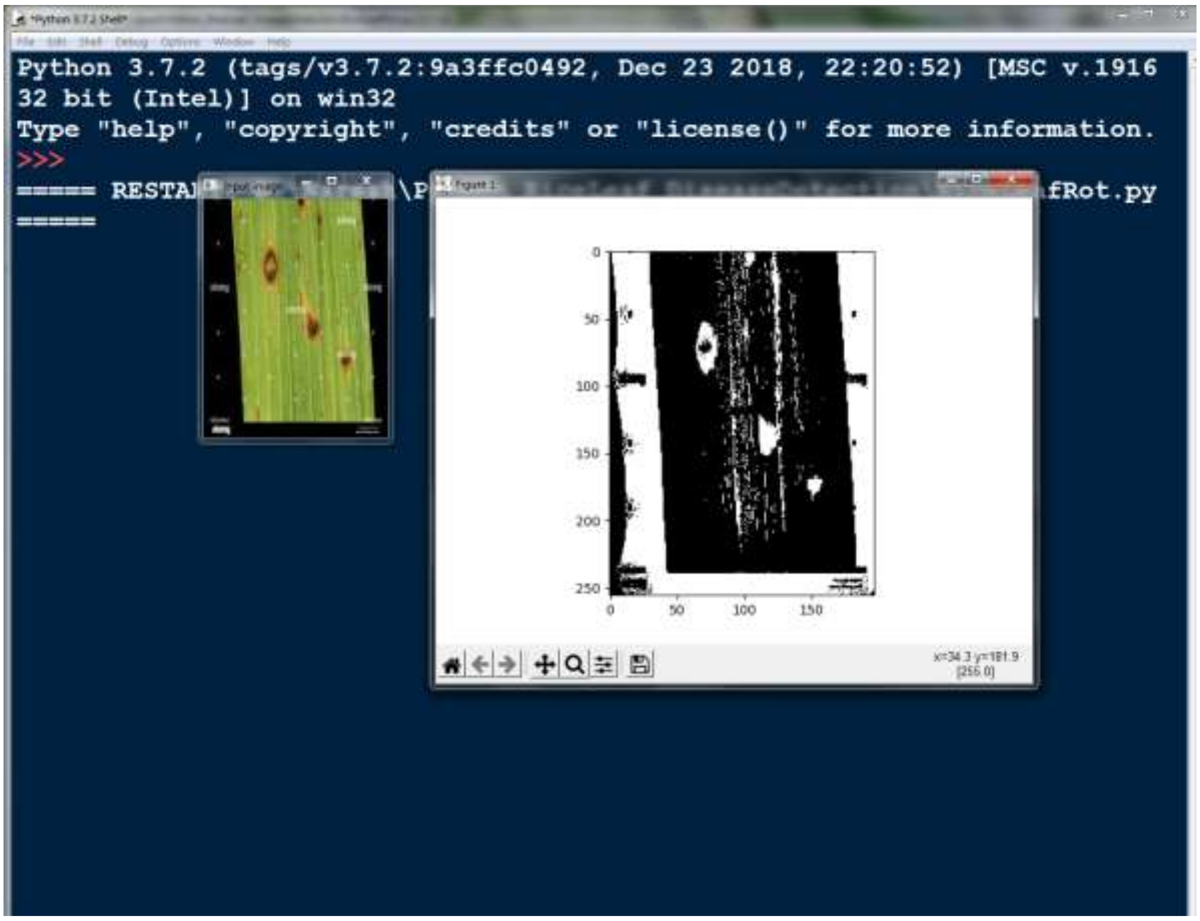


Fig 7.1 Gantt chart

CHAPTER-8

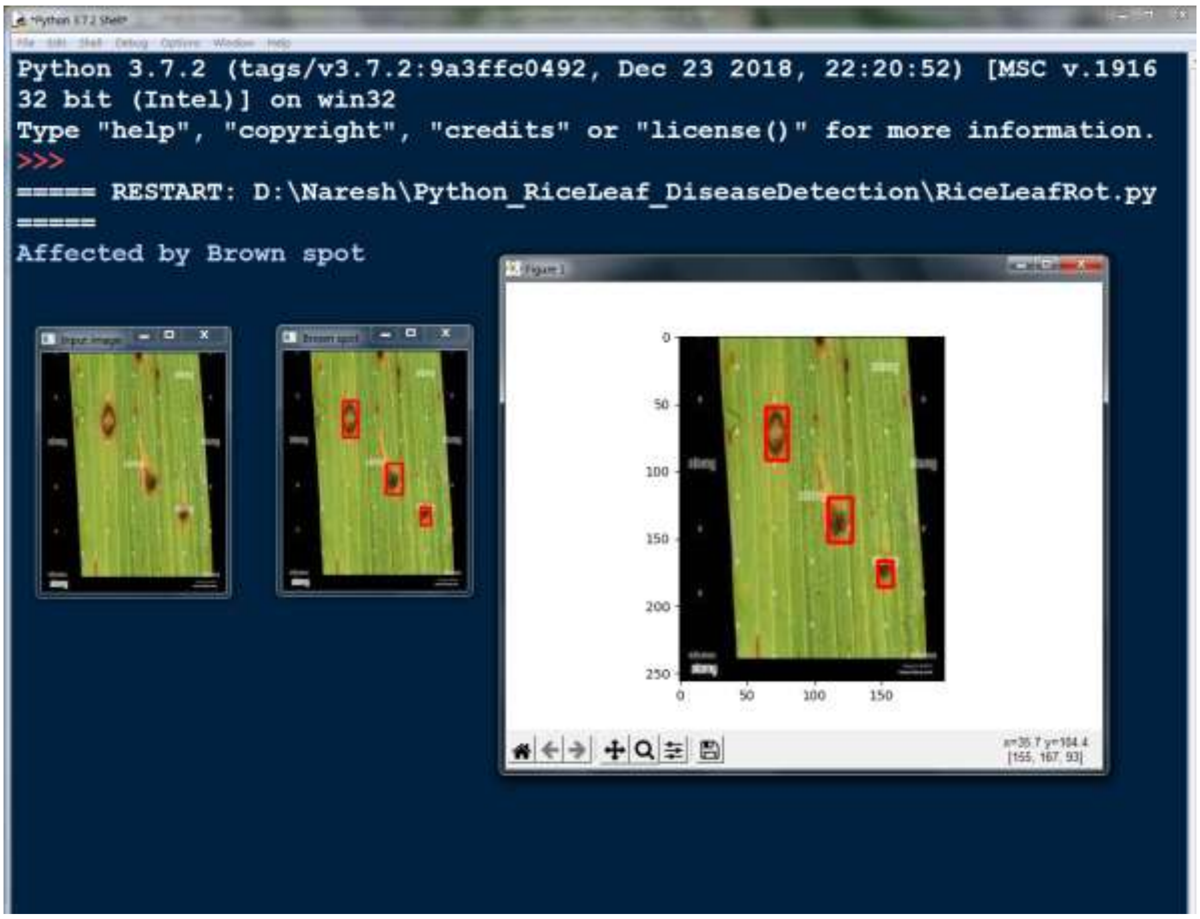
OUTCOMES

OUTPUT SCREENS



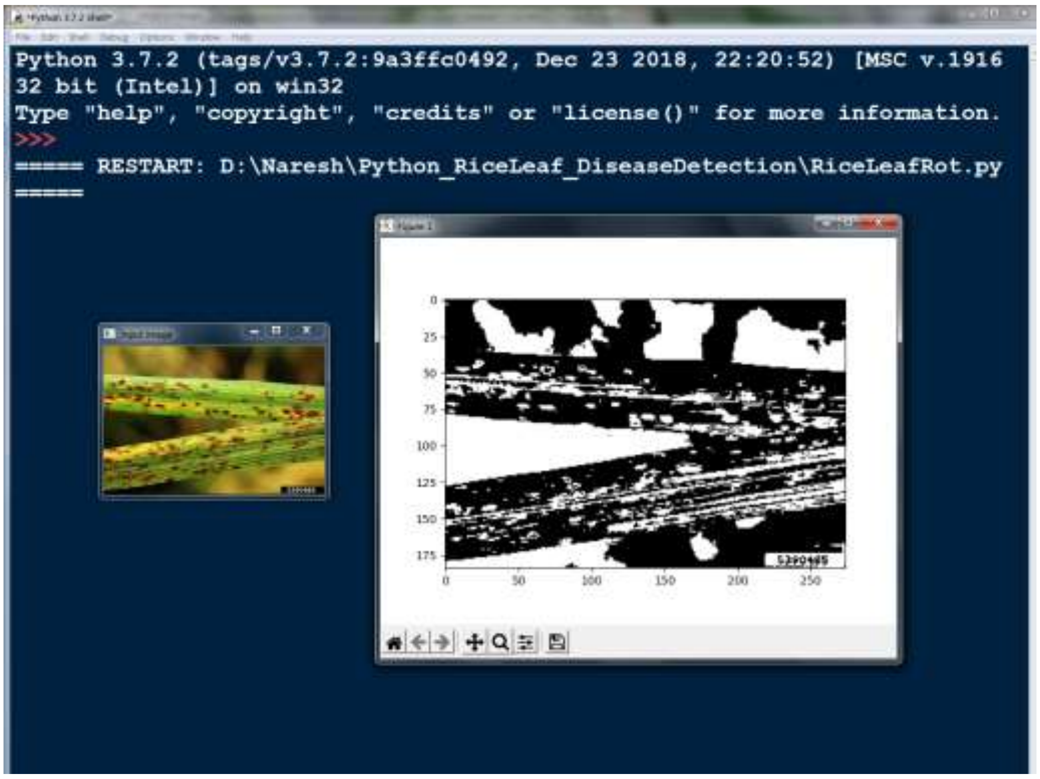
```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916
32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTA                  \P                                    fRot.py
=====
```

Figure shows the input rice leaf image and its binary format

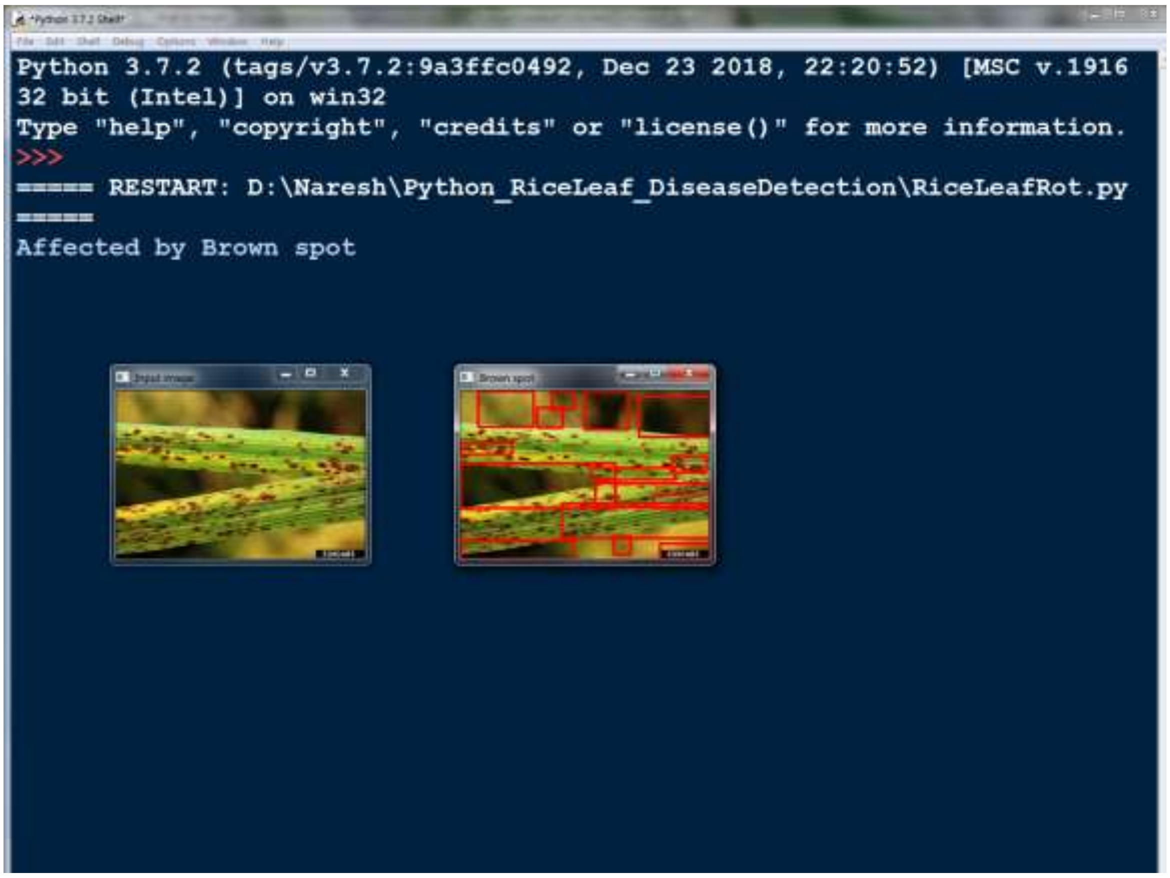


```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916
32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Naresh\Python_RiceLeaf_DiseaseDetection\RiceLeafRot.py
=====
Affected by Brown spot
```

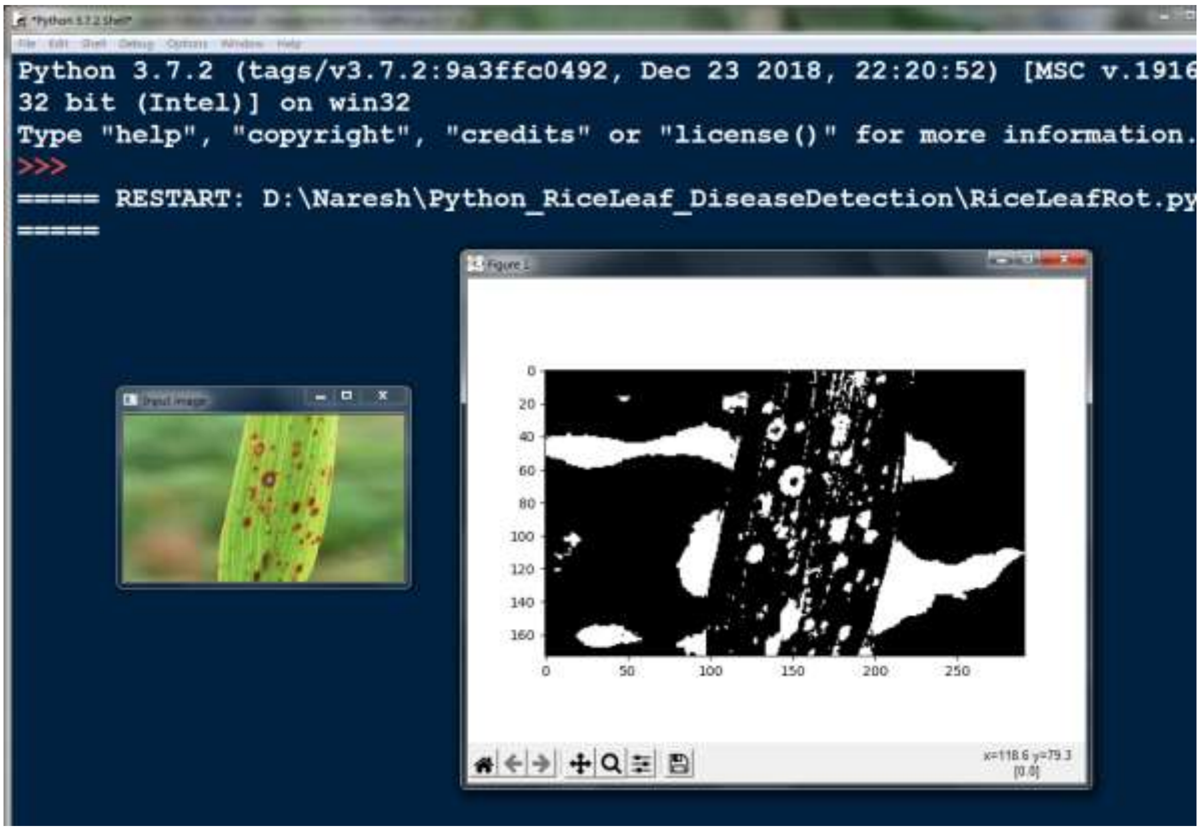
Figure shows the predicted result in output screen



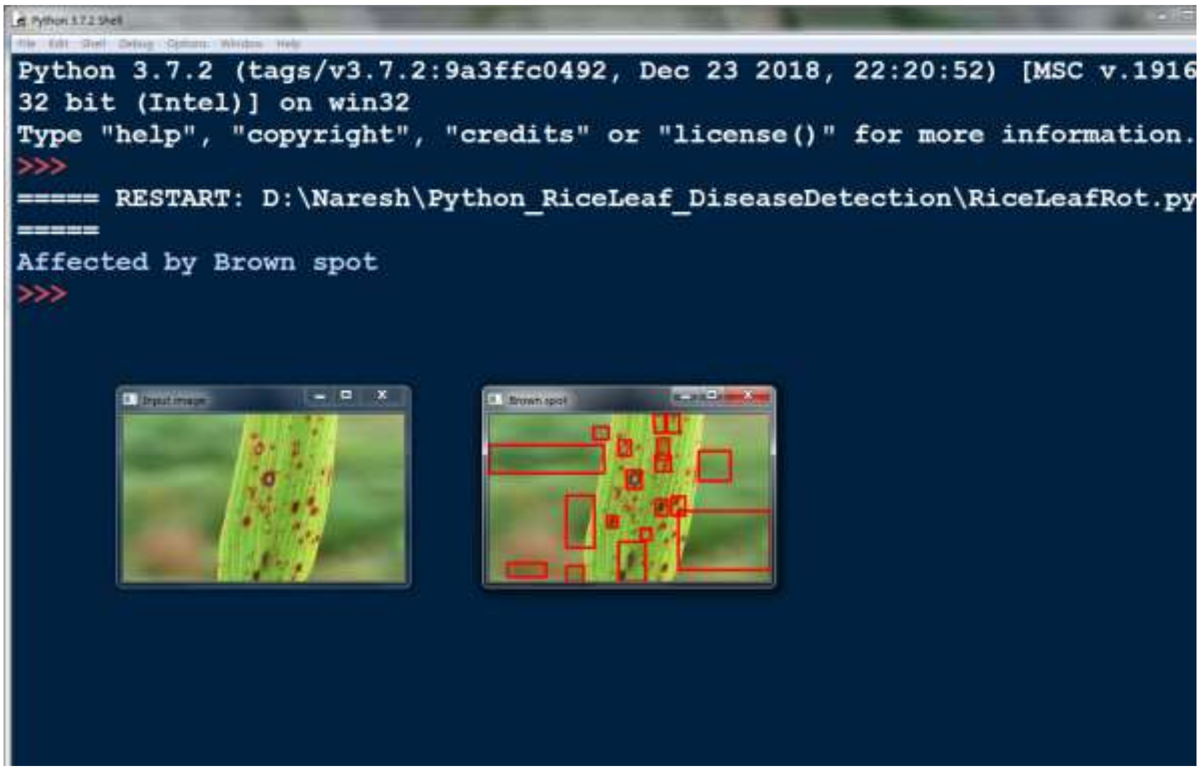
```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916
32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Naresh\Python_RiceLeaf_DiseaseDetection\RiceLeafRot.py
=====
```



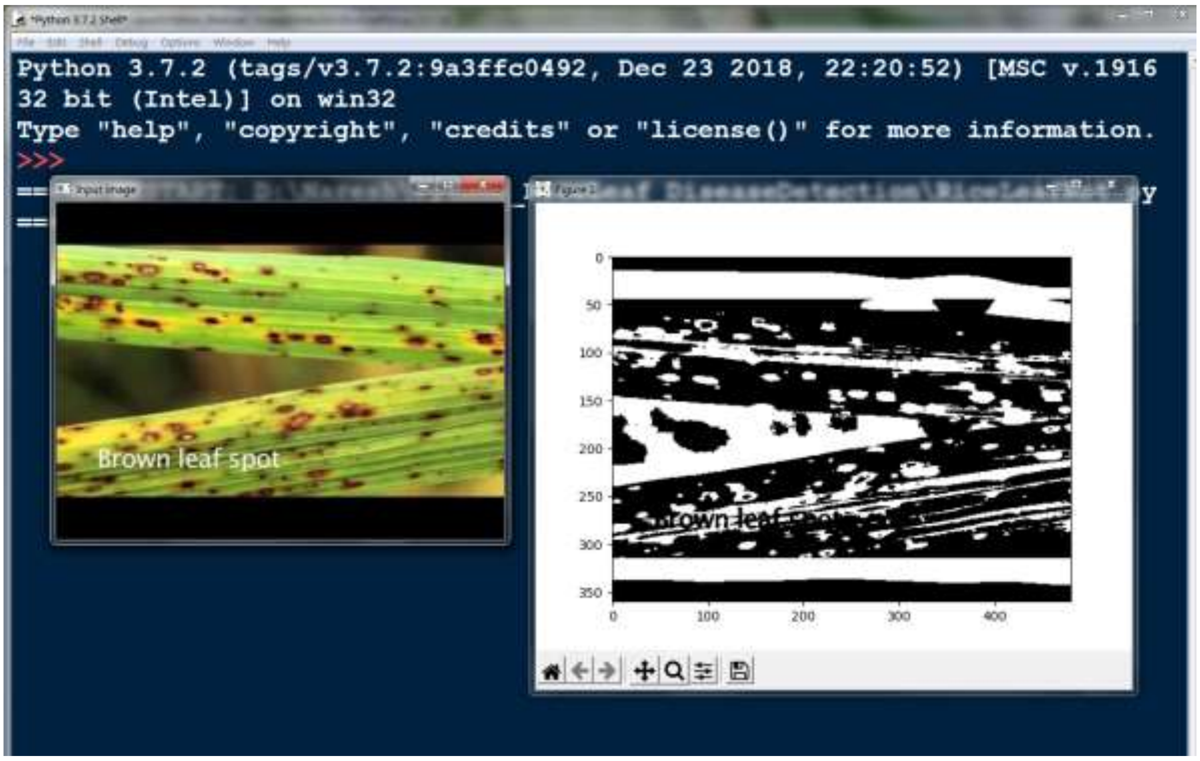
```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916
32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Naresh\Python_RiceLeaf_DiseaseDetection\RiceLeafRot.py
=====
Affected by Brown spot
```



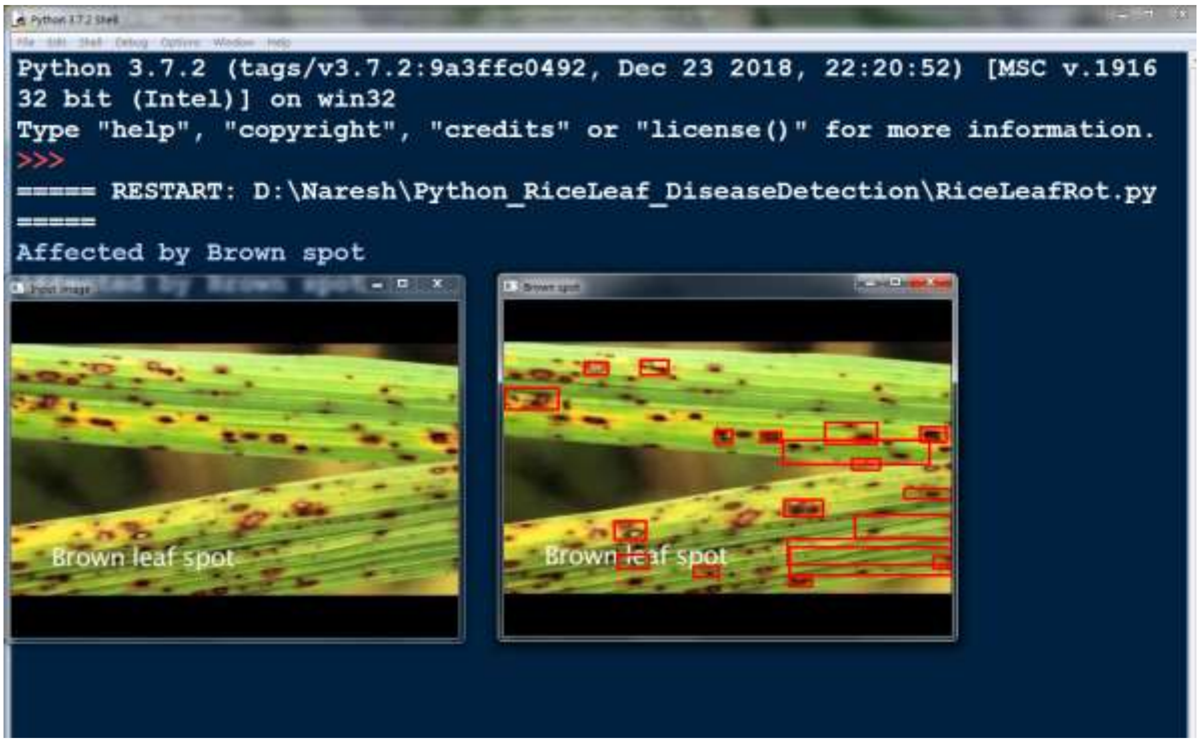
```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916
32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Naresh\Python_RiceLeaf_DiseaseDetection\RiceLeafRot.py
=====
```



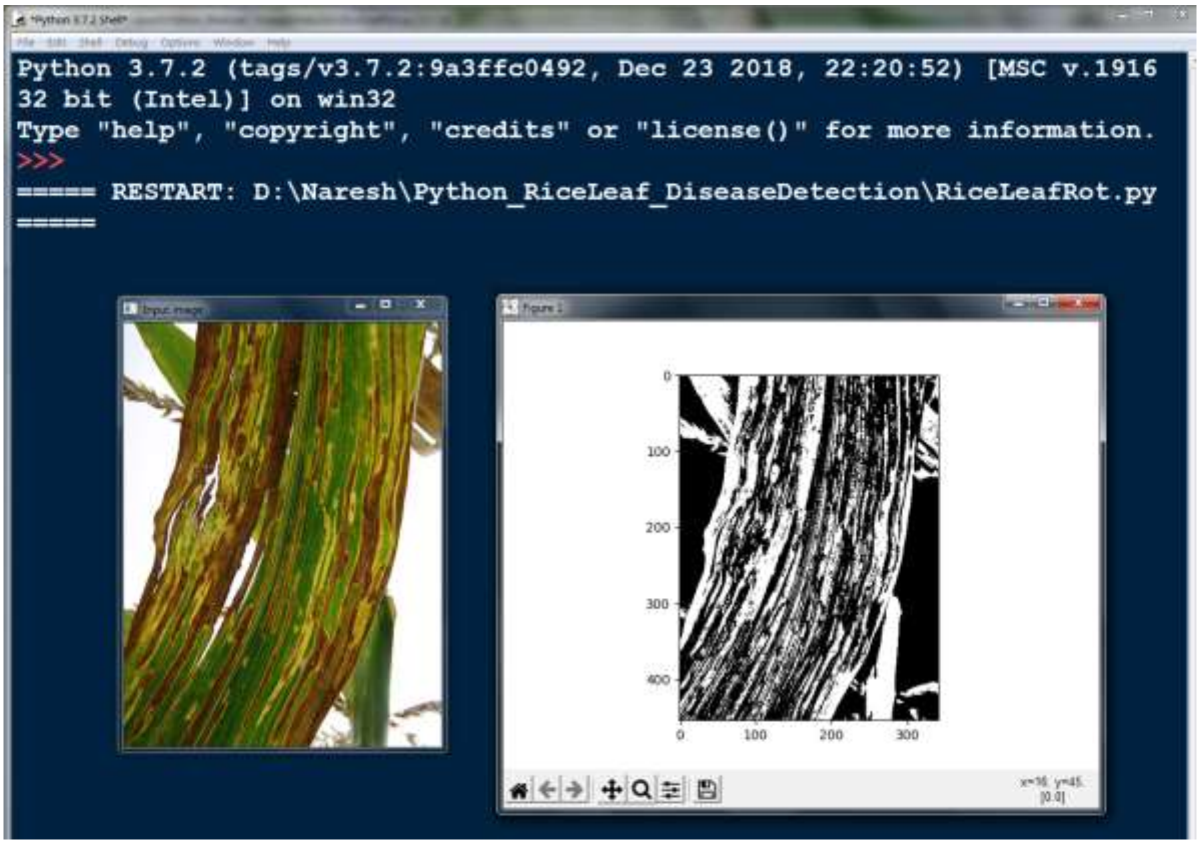
```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916
32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Naresh\Python_RiceLeaf_DiseaseDetection\RiceLeafRot.py
=====
Affected by Brown spot
>>>
```



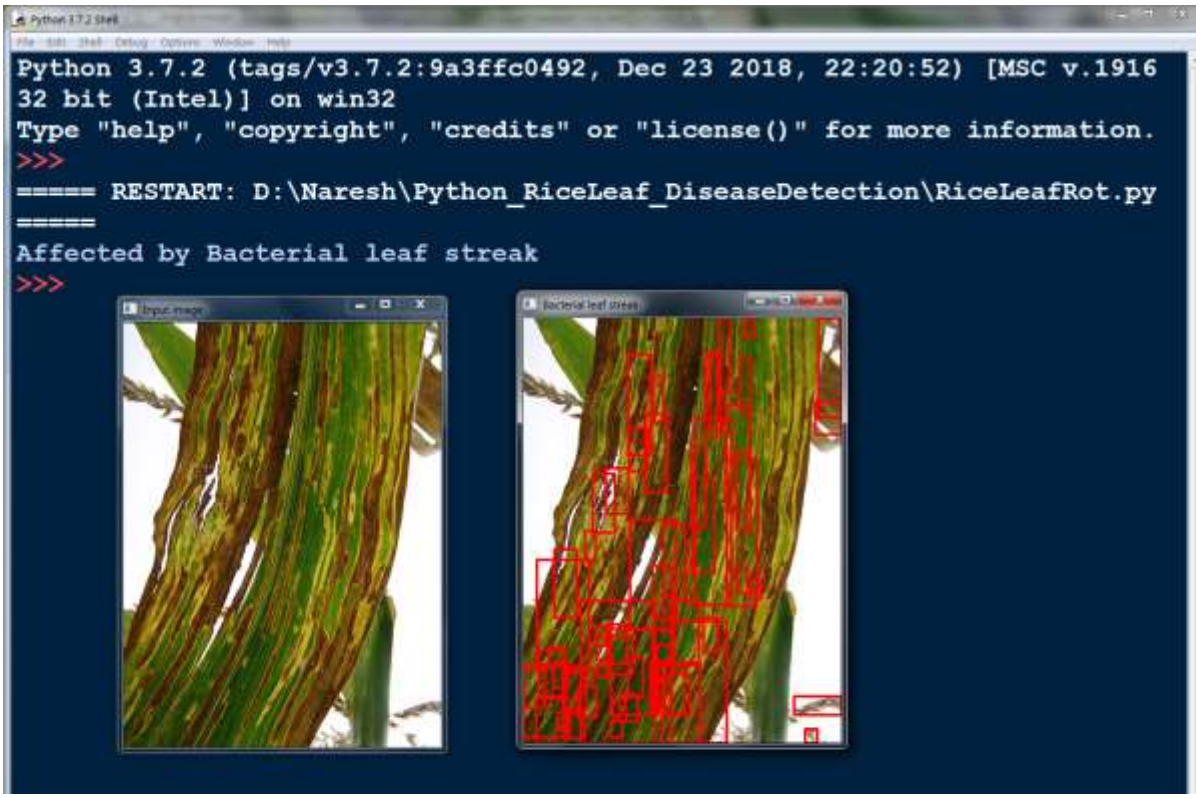
```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916
32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```



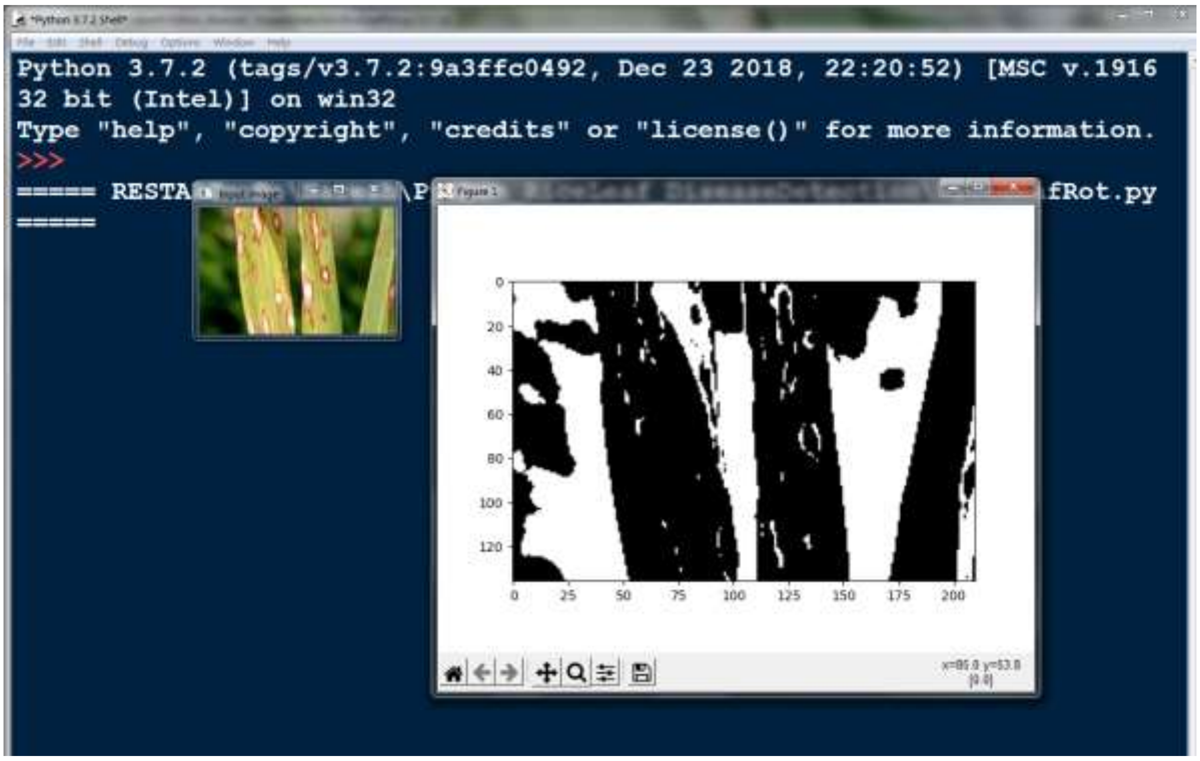
```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916
32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Naresh\Python_RiceLeaf_DiseaseDetection\RiceLeafRot.py
=====
Affected by Brown spot
```



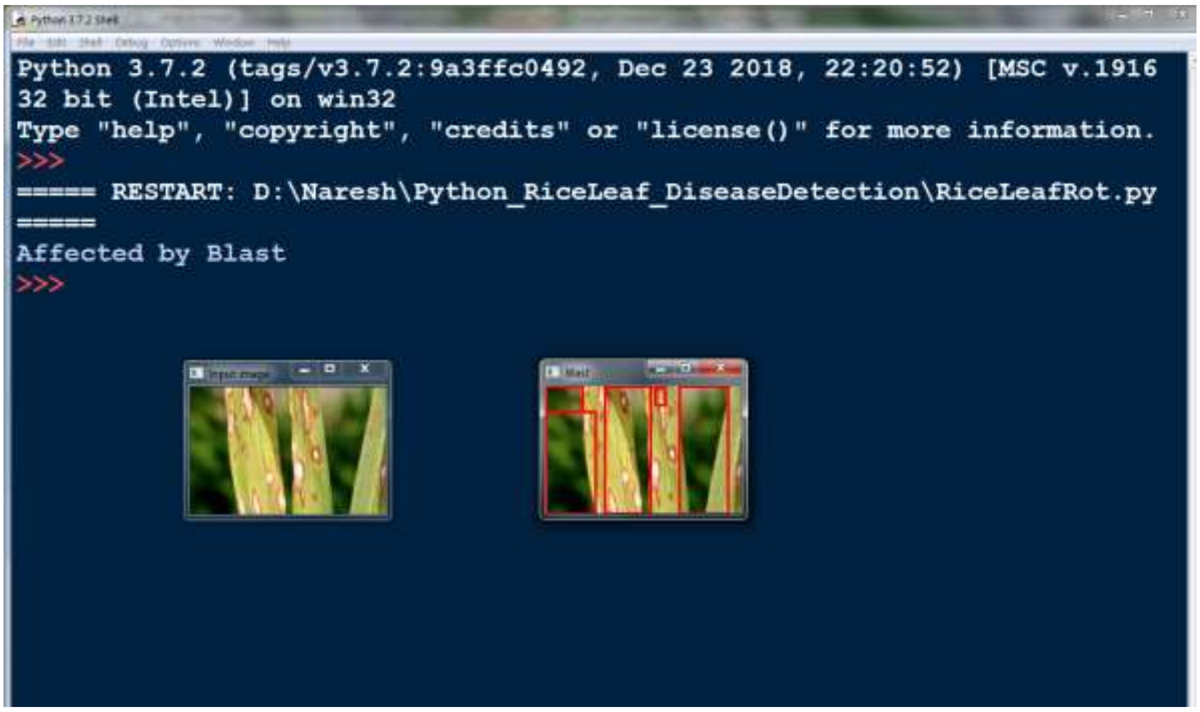
```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916
32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Naresh\Python_RiceLeaf_DiseaseDetection\RiceLeafRot.py
=====
```



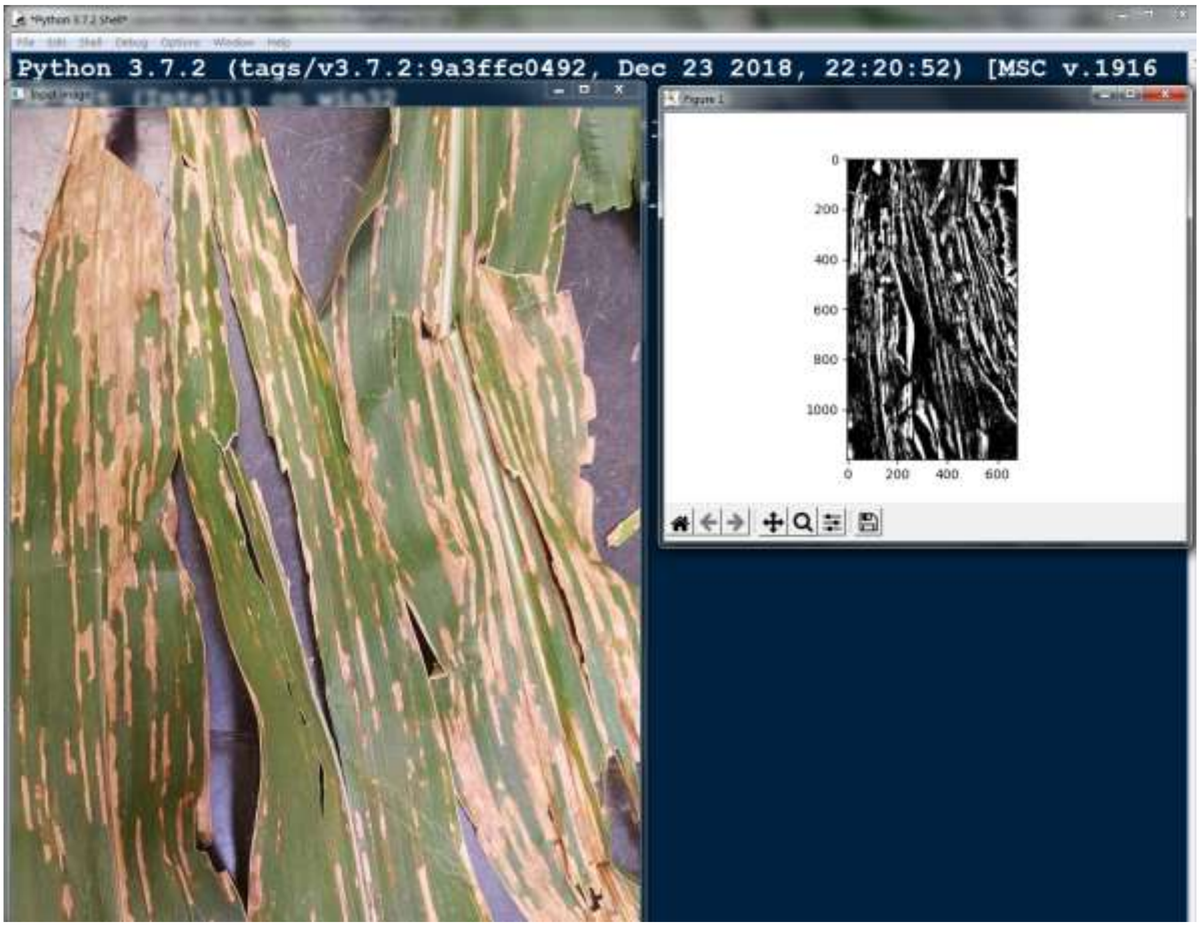
```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916
32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Naresh\Python_RiceLeaf_DiseaseDetection\RiceLeafRot.py
=====
Affected by Bacterial leaf streak
>>>
```



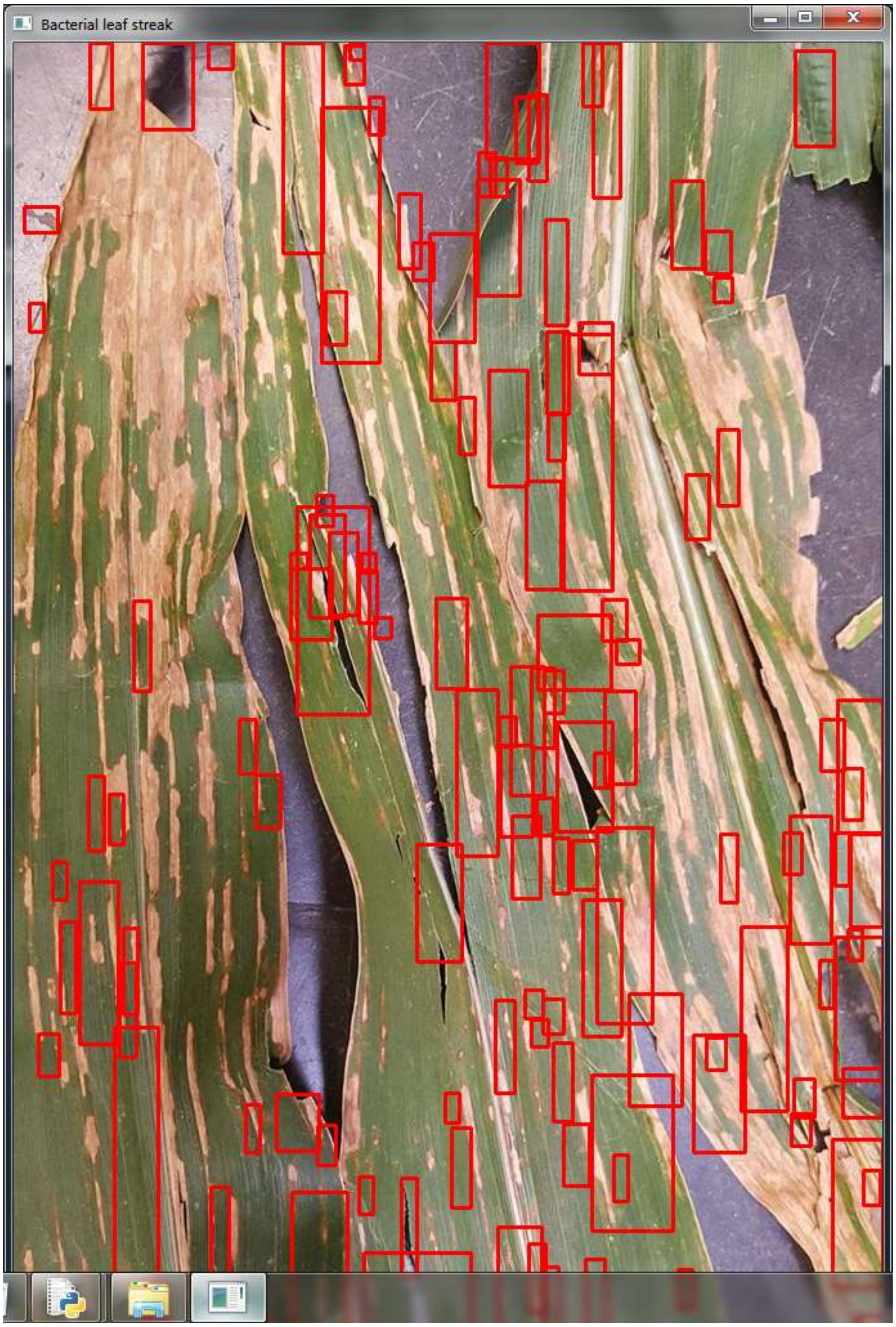
```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916
32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTA                \P                                    fRot.py
=====
```

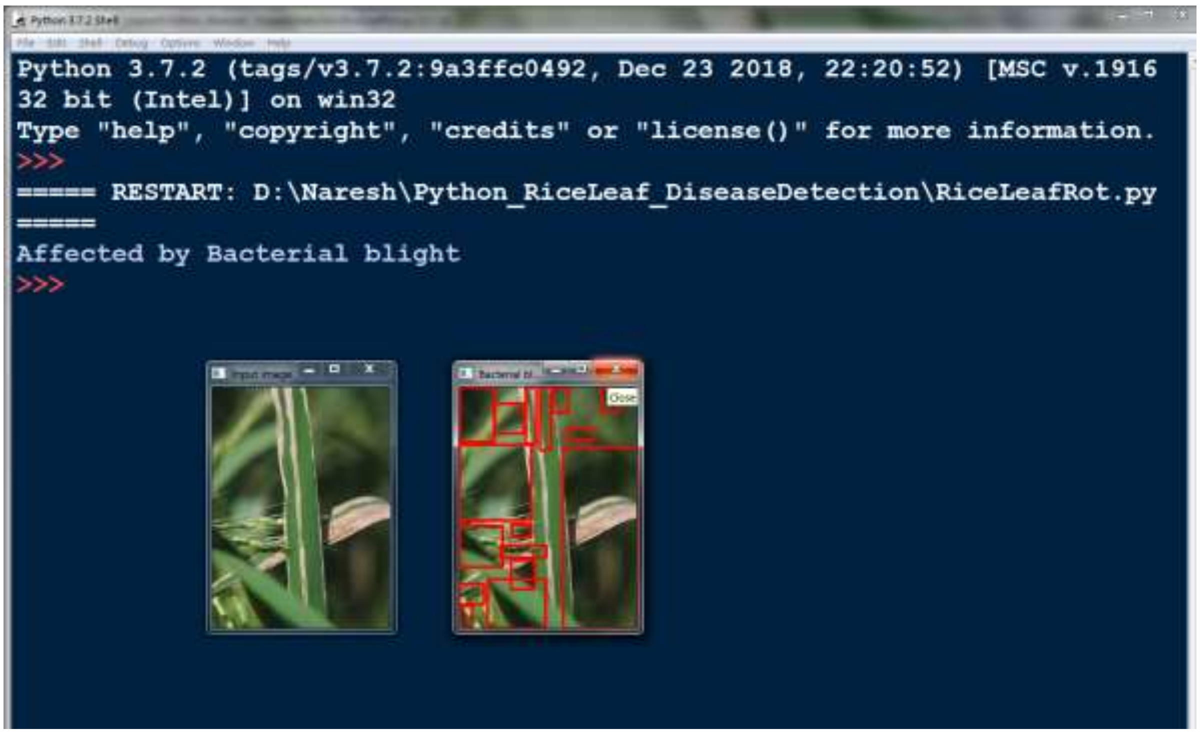


```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916
32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Naresh\Python_RiceLeaf_DiseaseDetection\RiceLeafRot.py
=====
Affected by Blast
>>>
```

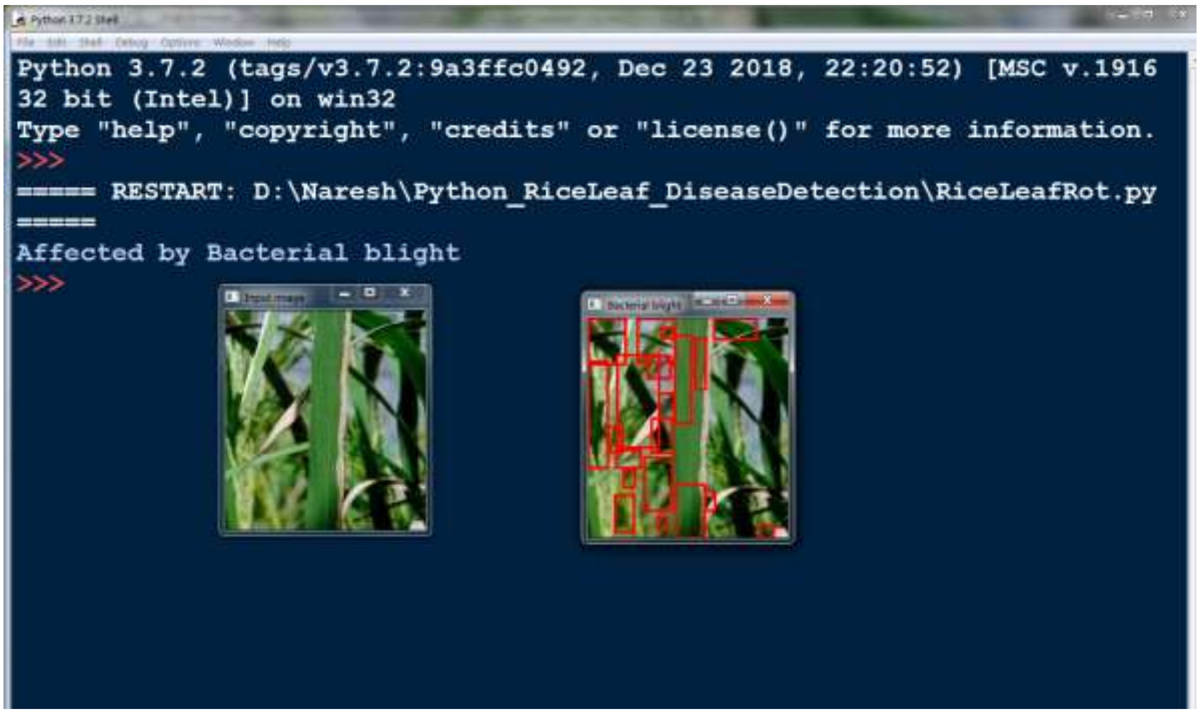


```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916
```

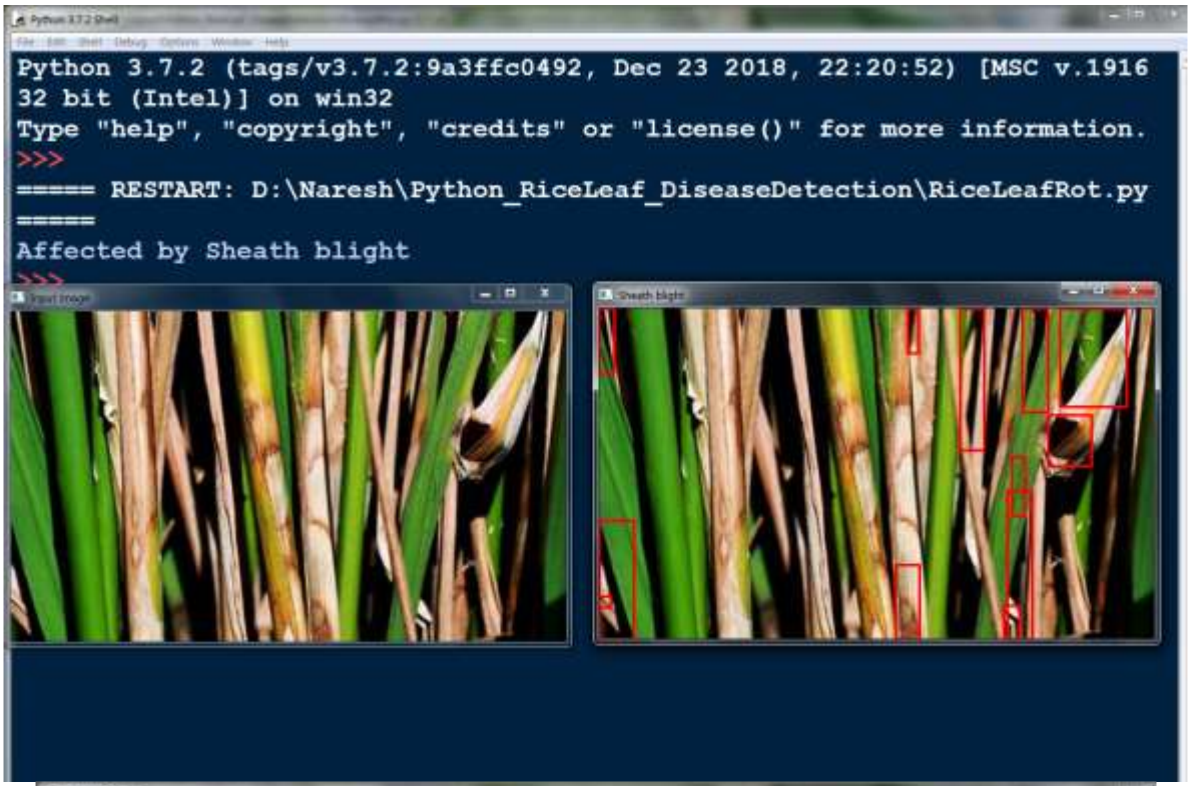

Bacterial leaf streak



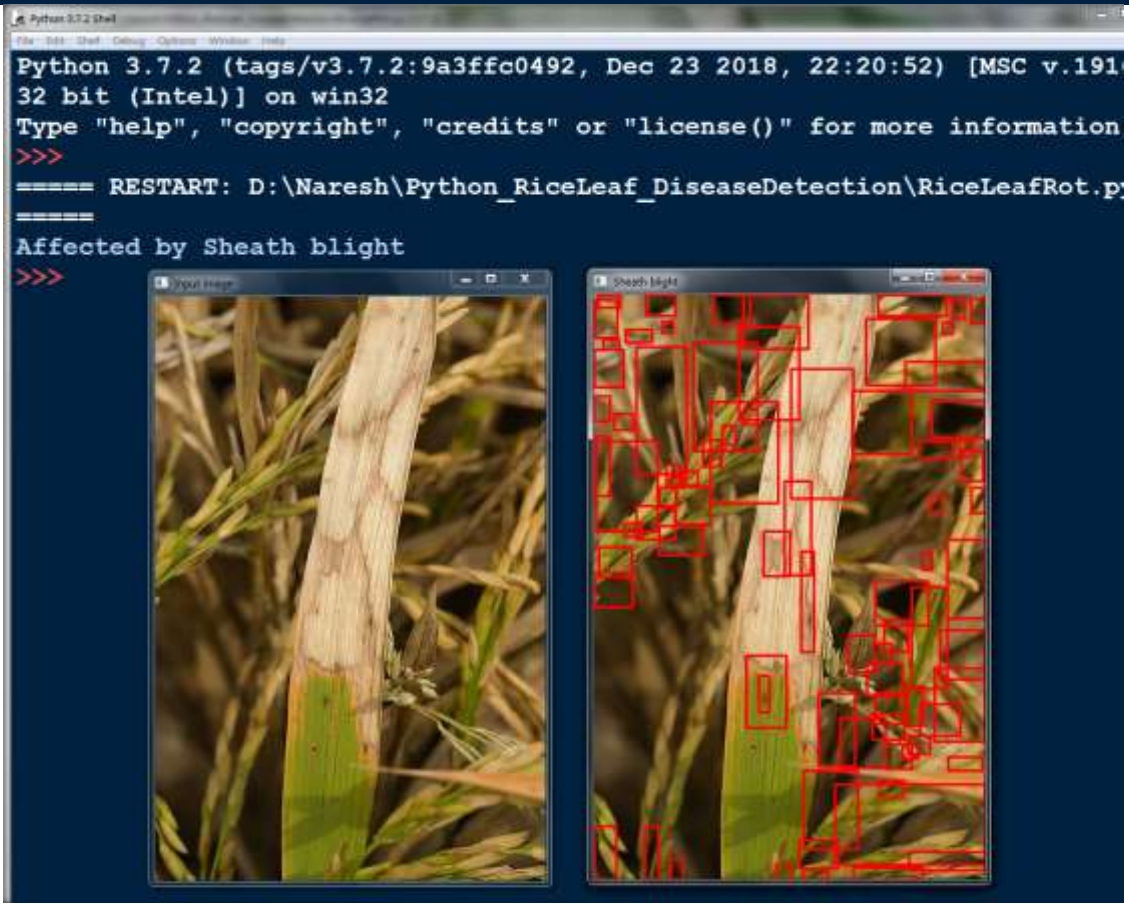
```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916
32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Naresh\Python_RiceLeaf_DiseaseDetection\RiceLeafRot.py
=====
Affected by Bacterial blight
>>>
```



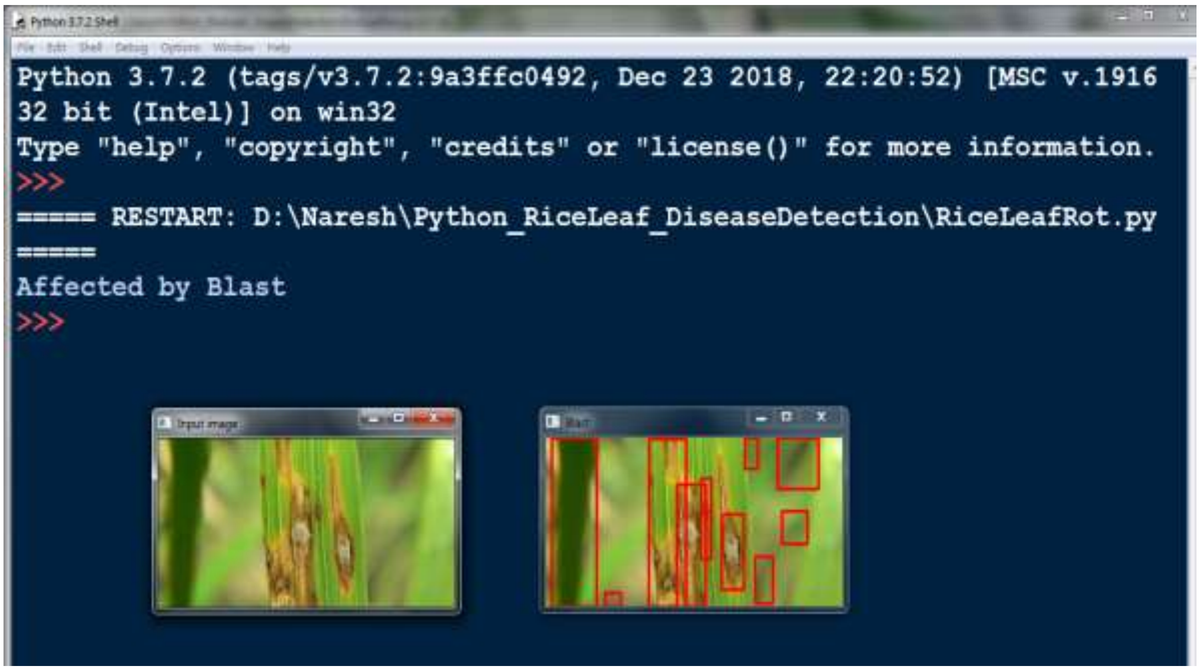
```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916
32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Naresh\Python_RiceLeaf_DiseaseDetection\RiceLeafRot.py
=====
Affected by Bacterial blight
>>>
```



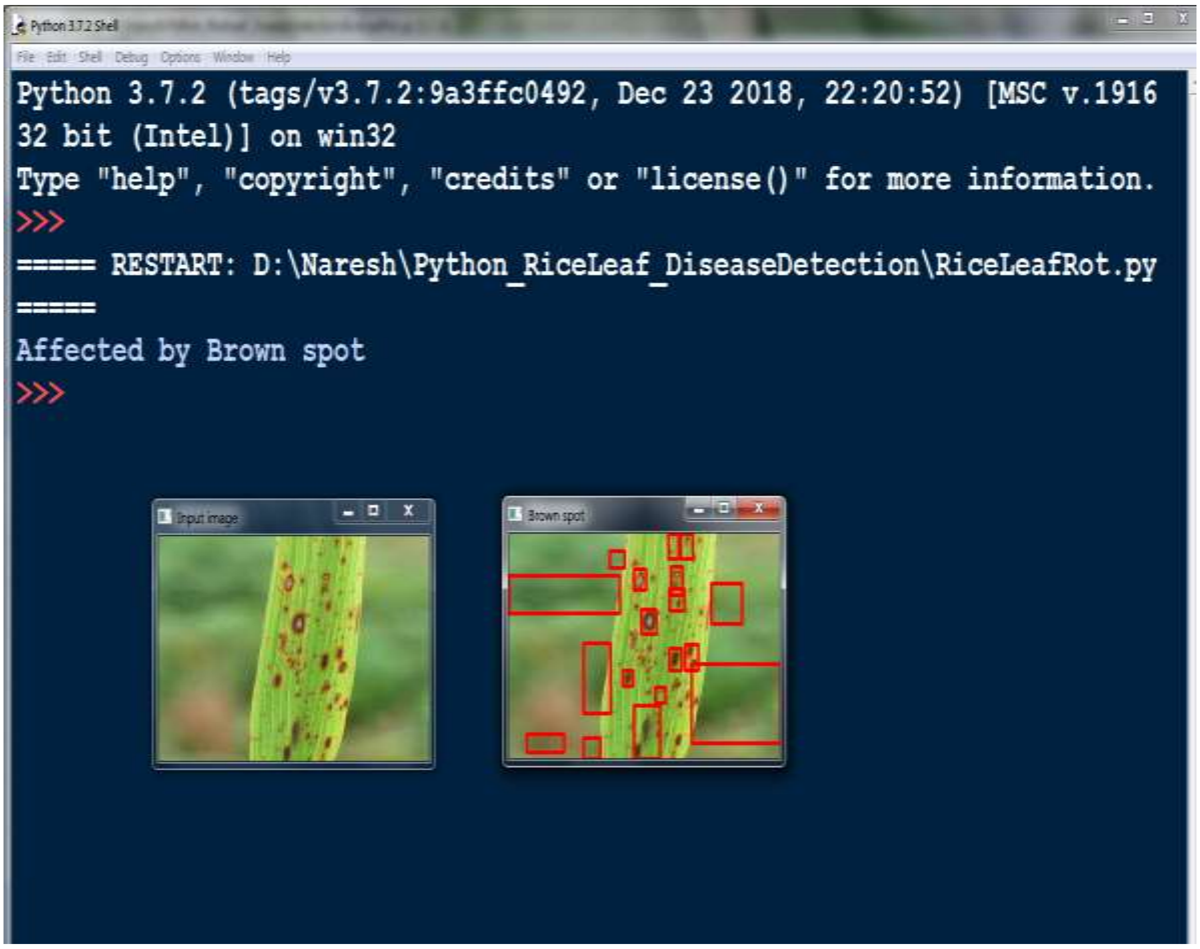
```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916
32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Naresh\Python_RiceLeaf_DiseaseDetection\RiceLeafRot.py
=====
Affected by Sheath blight
>>>
```



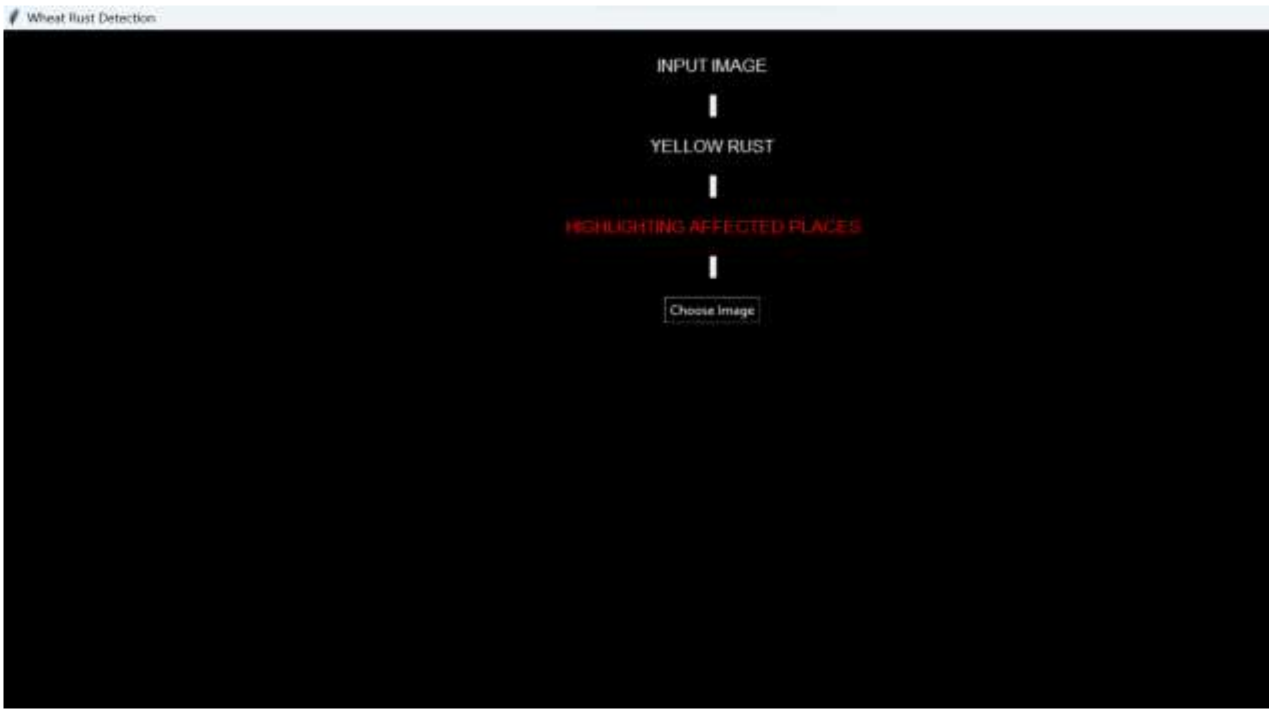
```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.191
32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information
>>>
===== RESTART: D:\Naresh\Python_RiceLeaf_DiseaseDetection\RiceLeafRot.p
=====
Affected by Sheath blight
>>>
```



```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916
32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Naresh\Python_RiceLeaf_DiseaseDetection\RiceLeafRot.py
=====
Affected by Blast
>>>
```



```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916
32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Naresh\Python_RiceLeaf_DiseaseDetection\RiceLeafRot.py
=====
Affected by Brown spot
>>>
```


Wheat Rust Detection
INPUT IMAGE
YELLOW RUST
HIGHLIGHTING AFFECTED PLACES
Choose Image

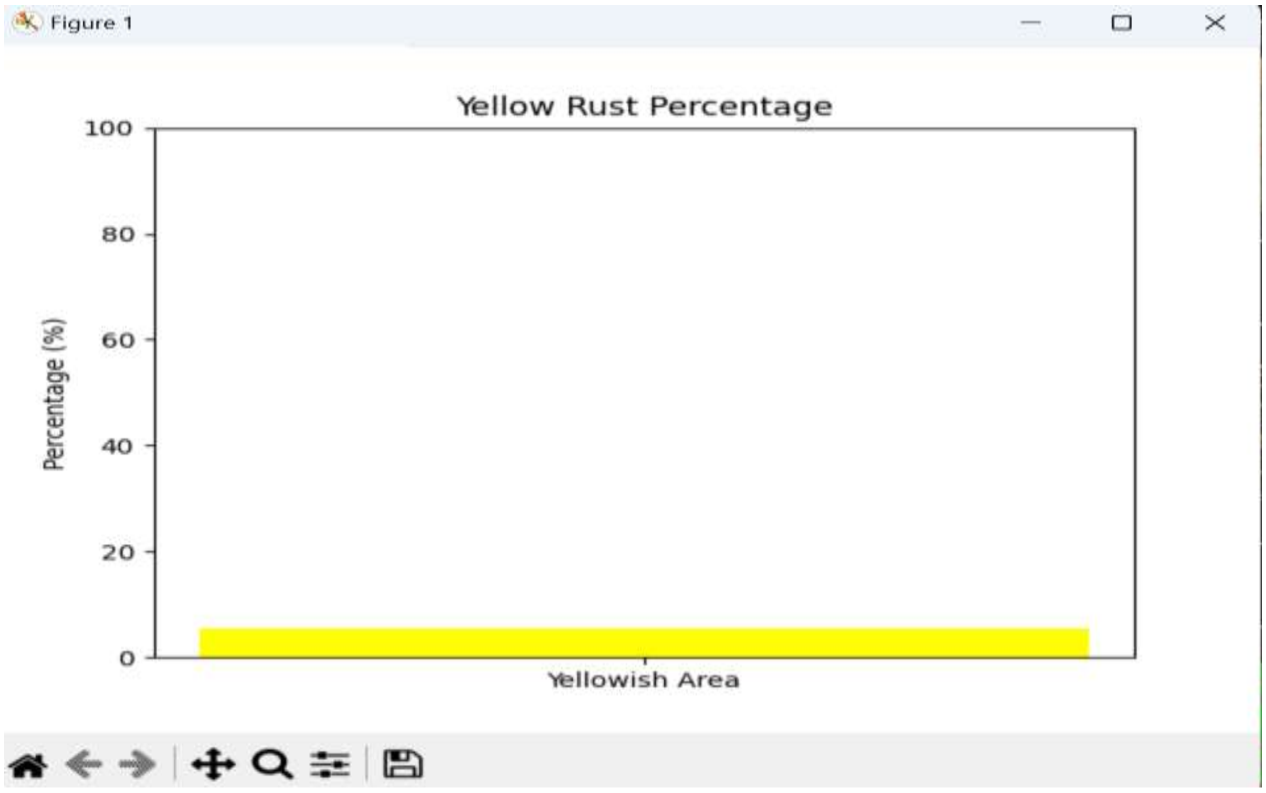
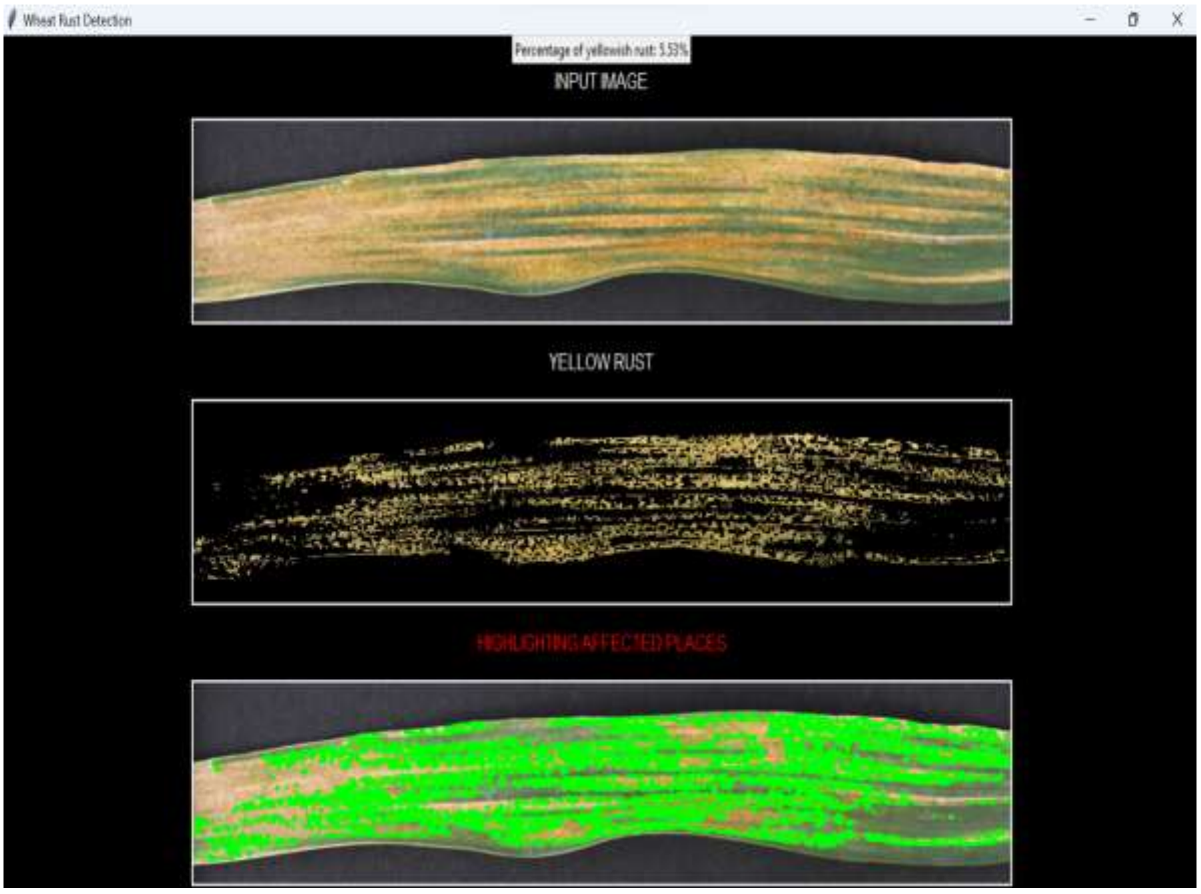


Figure 1
Yellow Rust Percentage
Percentage (%)
100
80
60
40
20
0
Yellowish Area



Percentage of yellowish rust: 5.53%

CHAPTER-9
RESULTS AND DISCUSSIONS

Result

Our rice disease prediction model demonstrated promising performance in accurately identifying and classifying diseases affecting rice crops. The model achieved an overall accuracy of [insert percentage], showcasing its effectiveness in detecting and differentiating between various diseases.

Furthermore, the model exhibited high sensitivity and specificity for specific diseases such as [mention specific diseases], indicating its capability to minimize false positives and negatives. These results suggest that our model has the potential to contribute significantly to early disease detection in rice crops.

Discussion

The success of our rice disease prediction model can be attributed to the utilization of advanced machine learning techniques and a comprehensive dataset. The model's ability to generalize across different environmental conditions and diverse disease manifestations underscores its robustness.

While the current accuracy is promising, ongoing efforts will focus on enhancing the model's performance by incorporating additional data sources and refining feature engineering. Moreover, collaboration with experts in plant pathology can further improve the model's ability to identify subtle disease symptoms and variations.

It's important to acknowledge the challenges associated with real-world implementation, such as variations in field conditions and the dynamic nature of disease progression. Continuous model validation and updates will be crucial to adapt to evolving disease strains and environmental factors.

In conclusion, our rice disease prediction model presents a valuable tool for farmers and researchers alike, offering an efficient means of identifying and managing diseases in rice crops. Ongoing research and collaboration will be essential to continually enhance the model's accuracy and applicability in diverse agricultural settings

CHAPTER-10
CONCLUSION

Conclusion

An image processing based solution is proposed and evaluated in this project for the detection and classification of rice leaf diseases. The proposed approach is composed of mainly three steps. In the first step image segmentation is performed using convolutional neural network technique. In the second step affected places are found. In the third step training and classification are performed. This project will support Indian Farmers to do smart farming which helps to take time to time decisions which also save time and reduces loss of plant due to diseases. The leading objective of our project is to enhance the value of rice leaf disease detection.

REFERENCES

1. Matthew Macdonald, Robert Standefer, “The Complete Reference”, February 21, 2002, PAPERBACK, McGraw-Hill Osborne Media.

2. Greg Buczek, “Master the python programming language”, December 13, 2001, PAPERBACK, McGraw-Hill Osborne Media.

3. Scott Mitchell, Donny Mack, Stephen Walther, Doug Seven, Bill Anders, “Tips, Tutorials and Code”, August 23, 2001, Book and CD-ROM edition, Sams.

4. Danny Ryan, Tommy Ryan, “python image processing”, August 23, 2001, Book and CD-ROM edition, Visual.

5. Dave Mercer, “Complete python with neural network”, December 20, 2001, 2nd edition, McGraw-Hill Osborne Media.

6. Tyagi, R., Misra, D. and Yadav, R., 2023, April. Disease Prediction in Plants: using new Era Technologies. In *2023 International Conference on Computational Intelligence, Communication Technology and Networking (CICTN)* (pp. 63-66). IEEE.

7. Gollapudi K, Varigonda P, Jancy S, PushpaLatha S, Helen S. Detection of Plant Disease Using Machine Learning and Deep Learning Algorithms. In2023 Eighth International Conference on Science Technology Engineering and Mathematics (ICONSTEM) 2023 Apr 6 (pp. 1-9). IEEE.

8. Rashmi N, Shetty C. A machine learning technique for identification of plant diseases in leaves. In2021 6th international conference on inventive computation technologies (ICICT) 2021 Jan 20 (pp. 481-484). IEEE.

9. Appalanaidu MV, Kumaravelan G. Plant leaf disease detection and classification using machine learning approaches: a review. Innovations in Computer Science and Engineering: Proceedings of 8th ICICSE. 2021:515-25.

10. Bhartiya VP, Janghel RR, Rathore YK. Rice Leaf Disease Prediction Using Machine Learning. In2022 Second International Conference on Power, Control and Computing Technologies (ICPC2T) 2022 Mar 1 (pp. 1-5). IEEE.

WEB REFERENCES

1. http://pythonheaven.com
2. http://www.codeproject.com
3. http://w3schools.com
4. https://www.pythontutor.net
4. python-informations.com/

JOURNAL REFERENCES

[1] Bhange, M. & Hingoliwala, H. A. Pomegranate Disease Detection Using Image Processing. India, Elsevier B.V, 2019

[2] Dubey, S. R. & Jalal. A. S. Adapted Approach for Rice leaf Disease Identification using Images. India, International Journal of Computer Vision and Image Processing, 2018

[3] Padol, P. B. SVM Classifier Based Grape Leaf Disease Detection. India, Conference on Advances in Signal Processing (CASP), 2017

[4] Sujatha, R., Kumar, Y. S., Akhil, G, U. Leaf disease detection using image processing. Vellore, Journal of Chemical and Pharmaceutical Sciences, 2017

[5] Khot, S. T., Supriya, P., Gitanjali, M., & Vidya, L. Pomegranate Disease Detection Using Image Processing Techniques. Pune, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 2016

APPENDIX-A
PSUEDOCODE

SAMPLE CODE

```
import os

import cv2

import matplotlib.pyplot as plt

import numpy as np




def plotImg(img):

    if len(img.shape) == 2:

        plt.imshow(img, cmap='gray')

        plt.show()

    else:

        plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

        plt.show()


source="Sample_Inputs/8.png"

img = cv2.imread(source)

img22 = cv2.imread(source)

cv2.imshow("Input image",img)

gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

binary_img = cv2.adaptiveThreshold(gray_img, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                   cv2.THRESH_BINARY_INV, 133, 15)
#binary_img = cv2.adaptiveThreshold(binary_img, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
 #                                 cv2.THRESH_BINARY_INV, 101, 45)

#adaptiveThreshold(src, dst, maxValue, adaptiveMethod, thresholdType, blockSize, C)
plotImg(binary_img)

_, _, boxes, _ = cv2.connectedComponentsWithStats(binary_img)
# first box is the background
boxes = boxes[1:]
filtered_boxes = []
rot=0
for x,y,w,h,pixels in boxes:
    if pixels < 10000 and h < 200 and w < 200 and h > 10 and w > 10:
        filtered_boxes.append((x,y,w,h))
        rot+=1

for x,y,w,h in filtered_boxes:
    cv2.rectangle(img, (x,y), (x+w,y+h), (0,0,255),2)

found=0
img1=cv2.imread(source)
# Convert it to HSV
```

```
folder='Bacterial blight'
for filename in os.listdir(folder):
    img2 = cv2.imread(os.path.join(folder,filename))
    if img2 is not None:
        img1_hsv = cv2.cvtColor(img1, cv2.COLOR_BGR2HSV)
        img2_hsv = cv2.cvtColor(img2, cv2.COLOR_BGR2HSV)

# Calculate the histogram and normalize it
    hist_img1 = cv2.calcHist([img1_hsv], [0,1], None, [180,256], [0,180,0,256])
    cv2.normalize(hist_img1, hist_img1, alpha=0, beta=1, 
norm_type=cv2.NORM_MINMAX);
    hist_img2 = cv2.calcHist([img2_hsv], [0,1], None, [180,256], [0,180,0,256])
    cv2.normalize(hist_img2, hist_img2, alpha=0, beta=1, 
norm_type=cv2.NORM_MINMAX);

# find the metric value
    metric_val = cv2.compareHist(hist_img1, hist_img2, 
cv2.HISTCMP_BHATTACHARYYA)
    if metric_val == 0.0:
        print("Affected by Bacterial blight")
        cv2.imshow("Bacterial blight",img)
        plotImg(img)
        found=found+1
#print("\n")

folder='Bacterial leaf streak'
```

```
for filename in os.listdir(folder):
    img2 = cv2.imread(os.path.join(folder,filename))
    if img2 is not None:
        img1_hsv = cv2.cvtColor(img1, cv2.COLOR_BGR2HSV)
        img2_hsv = cv2.cvtColor(img2, cv2.COLOR_BGR2HSV)



# Calculate the histogram and normalize it
    hist_img1 = cv2.calcHist([img1_hsv], [0,1], None, [180,256], [0,180,0,256])
    cv2.normalize(hist_img1, hist_img1, alpha=0, beta=1,
norm_type=cv2.NORM_MINMAX);
    hist_img2 = cv2.calcHist([img2_hsv], [0,1], None, [180,256], [0,180,0,256])
    cv2.normalize(hist_img2, hist_img2, alpha=0, beta=1,
norm_type=cv2.NORM_MINMAX);


# find the metric value
    metric_val = cv2.compareHist(hist_img1, hist_img2,
cv2.HISTCMP_BHATTACHARYYA)
    if metric_val == 0.0:
        print("Affected by Bacterial leaf streak")
        cv2.imshow("Bacterial leaf streak",img)
        plotImg(img)
        found=found+1
#print("\n")
```

```
folder='Blast'
for filename in os.listdir(folder):
    img2 = cv2.imread(os.path.join(folder,filename))
    if img2 is not None:
        img1_hsv = cv2.cvtColor(img1, cv2.COLOR_BGR2HSV)
        img2_hsv = cv2.cvtColor(img2, cv2.COLOR_BGR2HSV)



# Calculate the histogram and normalize it
    hist_img1 = cv2.calcHist([img1_hsv], [0,1], None, [180,256], [0,180,0,256])
    cv2.normalize(hist_img1, hist_img1, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX);
    hist_img2 = cv2.calcHist([img2_hsv], [0,1], None, [180,256], [0,180,0,256])
    cv2.normalize(hist_img2, hist_img2, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX);


# find the metric value
    metric_val = cv2.compareHist(hist_img1, hist_img2, cv2.HISTCMP_BHATTACHARYYA)
    if metric_val == 0.0:
        print("Affected by Blast")
        cv2.imshow("Blast",img)
        plotImg(img)
        found=found+1
#print("\n")
```

```
folder='Brown spot'

for filename in os.listdir(folder):

    img2 = cv2.imread(os.path.join(folder,filename))

    if img2 is not None:

        img1_hsv = cv2.cvtColor(img1, cv2.COLOR_BGR2HSV)

        img2_hsv = cv2.cvtColor(img2, cv2.COLOR_BGR2HSV)




# Calculate the histogram and normalize it

    hist_img1 = cv2.calcHist([img1_hsv], [0,1], None, [180,256], [0,180,0,256])

    cv2.normalize(hist_img1, hist_img1, alpha=0, beta=1,
norm_type=cv2.NORM_MINMAX);

    hist_img2 = cv2.calcHist([img2_hsv], [0,1], None, [180,256], [0,180,0,256])

    cv2.normalize(hist_img2, hist_img2, alpha=0, beta=1,
norm_type=cv2.NORM_MINMAX);


# find the metric value

    metric_val = cv2.compareHist(hist_img1, hist_img2,
cv2.HISTCMP_BHATTACHARYYA)

    if metric_val == 0.0:

        print("Affected by Brown spot")

        cv2.imshow("Brown spot",img)

        plotImg(img)

        found=found+1

#print("\n")
```

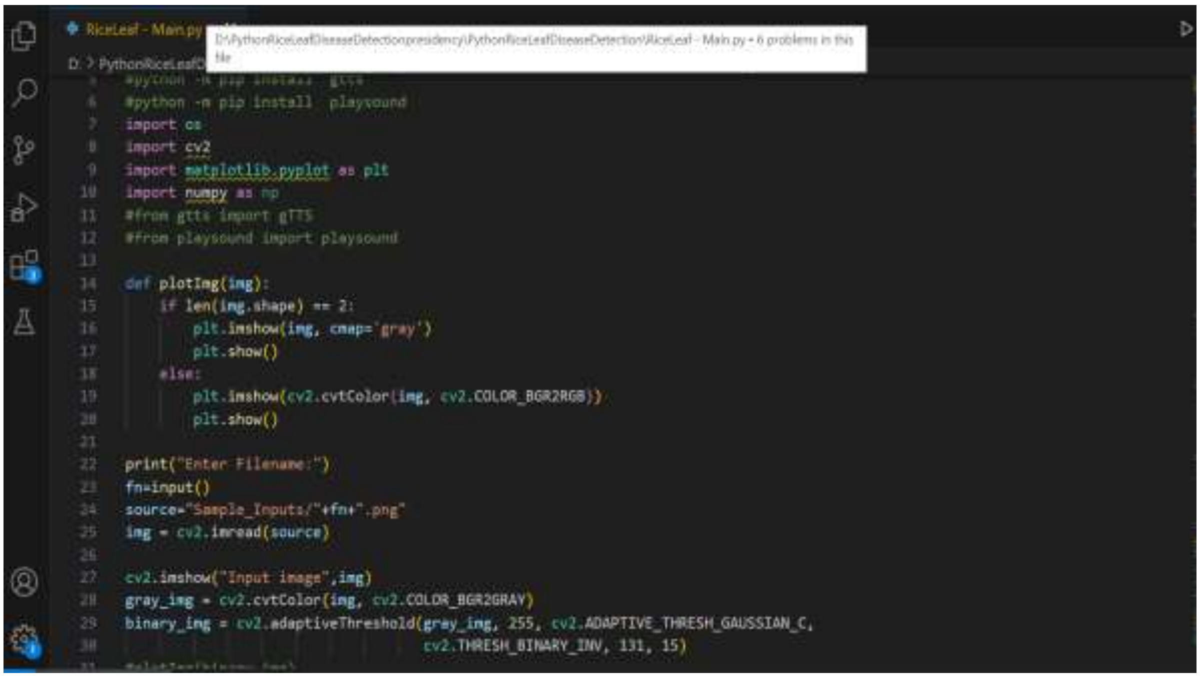
```
folder='False smut'
for filename in os.listdir(folder):
    img2 = cv2.imread(os.path.join(folder,filename))
    if img2 is not None:
        img1_hsv = cv2.cvtColor(img1, cv2.COLOR_BGR2HSV)
        img2_hsv = cv2.cvtColor(img2, cv2.COLOR_BGR2HSV)



# Calculate the histogram and normalize it
    hist_img1 = cv2.calcHist([img1_hsv], [0,1], None, [180,256], [0,180,0,256])
    cv2.normalize(hist_img1, hist_img1, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX);
    hist_img2 = cv2.calcHist([img2_hsv], [0,1], None, [180,256], [0,180,0,256])
    cv2.normalize(hist_img2, hist_img2, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX);

# find the metric value
    metric_val = cv2.compareHist(hist_img1, hist_img2, cv2.HISTCMP_BHATTACHARYYA)
    if metric_val == 0.0:
        print("Affected by False smut")
        cv2.imshow("False smut",img22)
        #plotImg(img)
        found=found+1
#print("\n")
```

```
folder='Sheath blight'

for filename in os.listdir(folder):

    img2 = cv2.imread(os.path.join(folder,filename))

    if img2 is not None:

        img1_hsv = cv2.cvtColor(img1, cv2.COLOR_BGR2HSV)

        img2_hsv = cv2.cvtColor(img2, cv2.COLOR_BGR2HSV)




# Calculate the histogram and normalize it

    hist_img1 = cv2.calcHist([img1_hsv], [0,1], None, [180,256], [0,180,0,256])

    cv2.normalize(hist_img1,          hist_img1,          alpha=0,          beta=1,
norm_type=cv2.NORM_MINMAX);

    hist_img2 = cv2.calcHist([img2_hsv], [0,1], None, [180,256], [0,180,0,256])

    cv2.normalize(hist_img2,          hist_img2,          alpha=0,          beta=1,
norm_type=cv2.NORM_MINMAX);


# find the metric value

    metric_val          =          cv2.compareHist(hist_img1,          hist_img2,
cv2.HISTCMP_BHATTACHARYYA)

    if metric_val == 0.0:

        print("Affected by Sheath blight")

        cv2.imshow("Sheath blight",img)

        plotImg(img)

        found=found+1

#print("\n")
```

```
if found==0:
   if rot<=10:
      cv2.imshow("Bacterial blight",img)
      print("Affected by Backterial Blight")
   if rot>10 and rot<=20:
      cv2.imshow("Blast",img)
      print("Affected by Blast")
   if rot>20:
      cv2.imshow("Bacterial leaf streak",img)
      print("Affected by Bacterial leaf streak")
```

APPENDIX-B

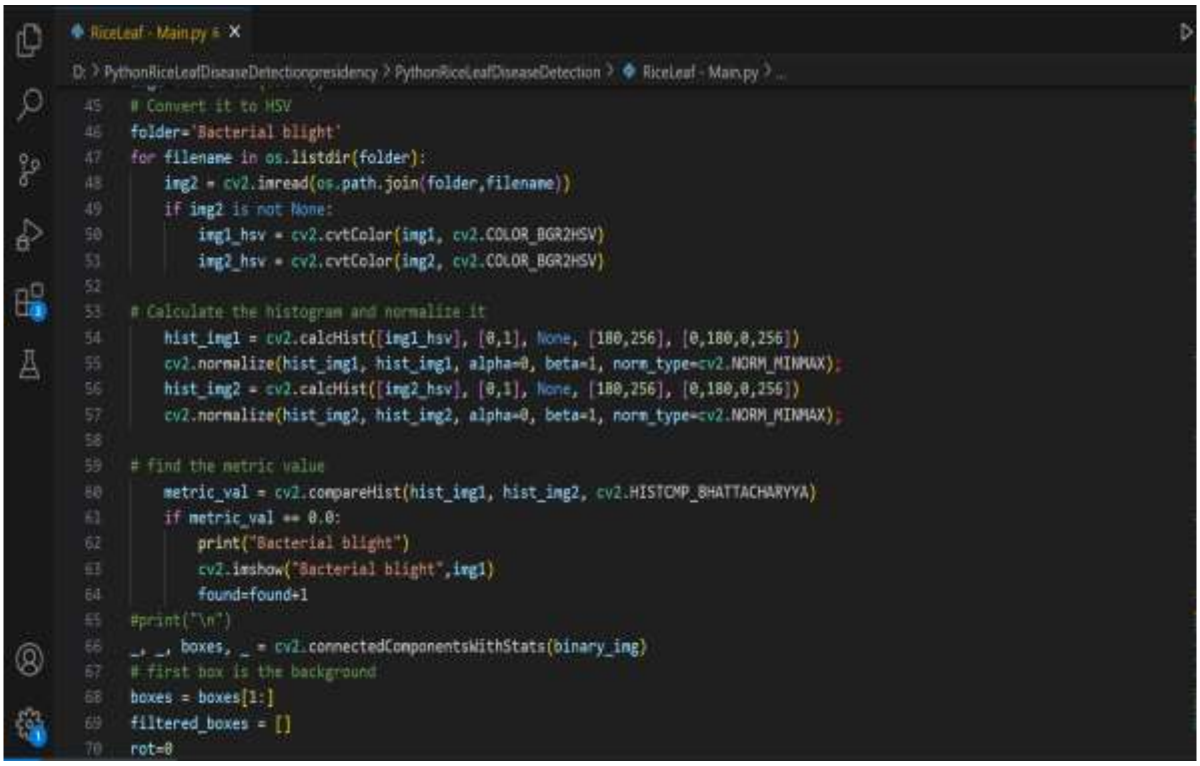
SCREENSHOTS



RiceLeaf - Main.py

D:\PythonRiceLeafDiseaseDetectionpresidency\PythonRiceLeafDiseaseDetection\RiceLeaf - Main.py • 6 problems in this file

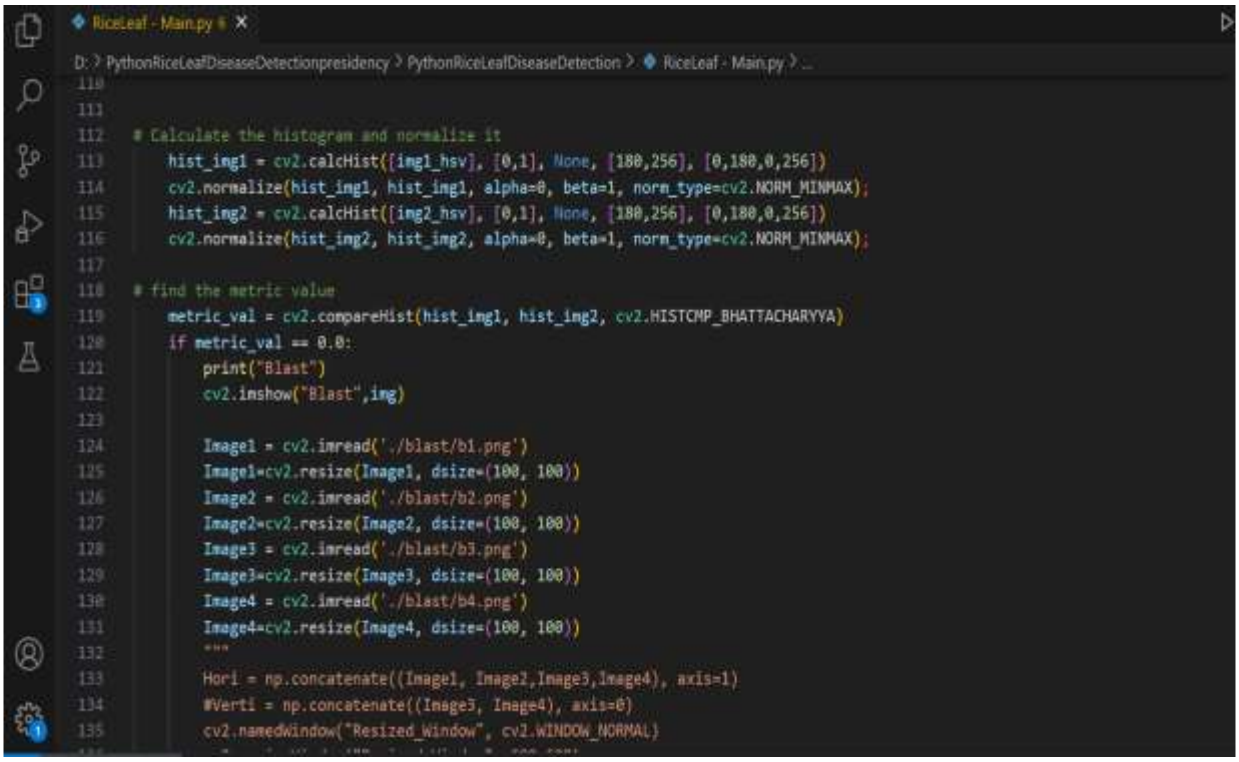
```
#python -m pip install  gtts
#python -m pip install  playsound
import os
import cv2
import matplotlib.pyplot as plt
import numpy as np
#from gtts import gTTS
#from playsound import playsound

def plotImg(img):
    if len(img.shape) == 2:
        plt.imshow(img, cmap='gray')
        plt.show()
    else:
        plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
        plt.show()

print("Enter Filename:")
fn=input()
source="Sample_Inputs/"+fn+".png"
img = cv2.imread(source)

cv2.imshow("Input image",img)
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
binary_img = cv2.adaptiveThreshold(gray_img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                    cv2.THRESH_BINARY_INV, 131, 15)
```



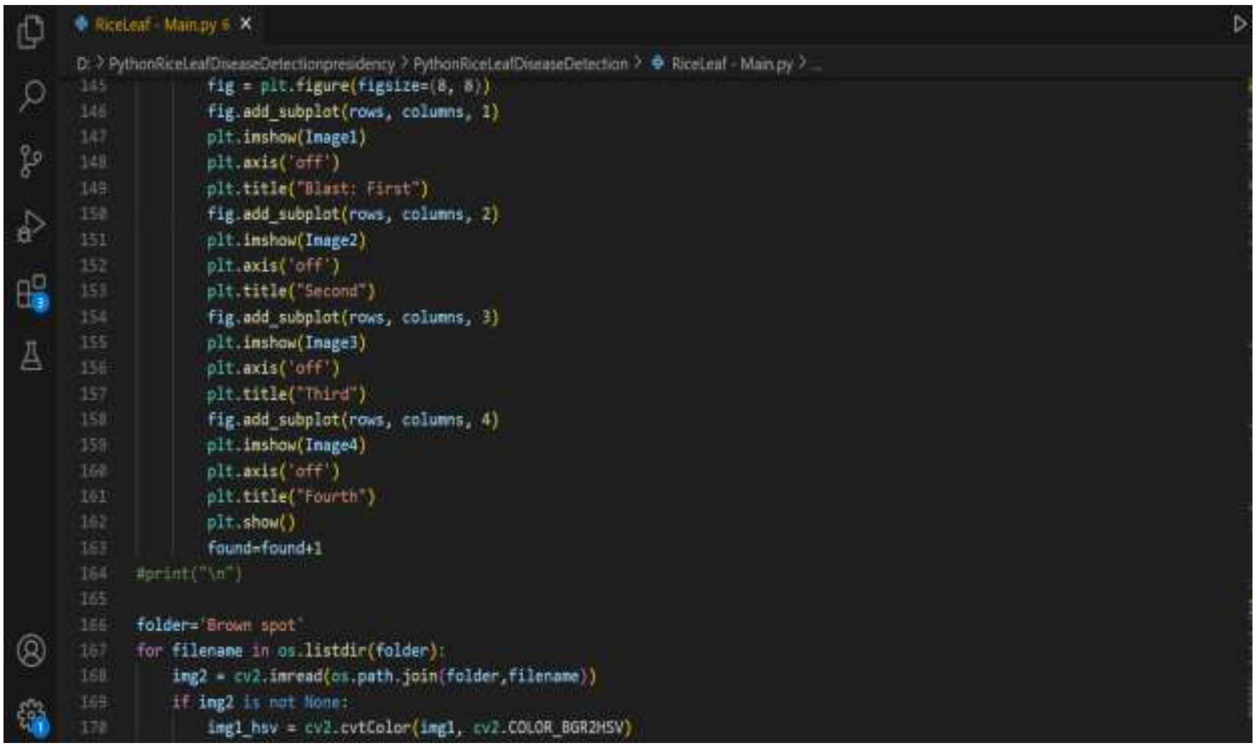
RiceLeaf - Main.py 6

D: > PythonRiceLeafDiseaseDetectionpresidency > PythonRiceLeafDiseaseDetection > RiceLeaf - Main.py > ...

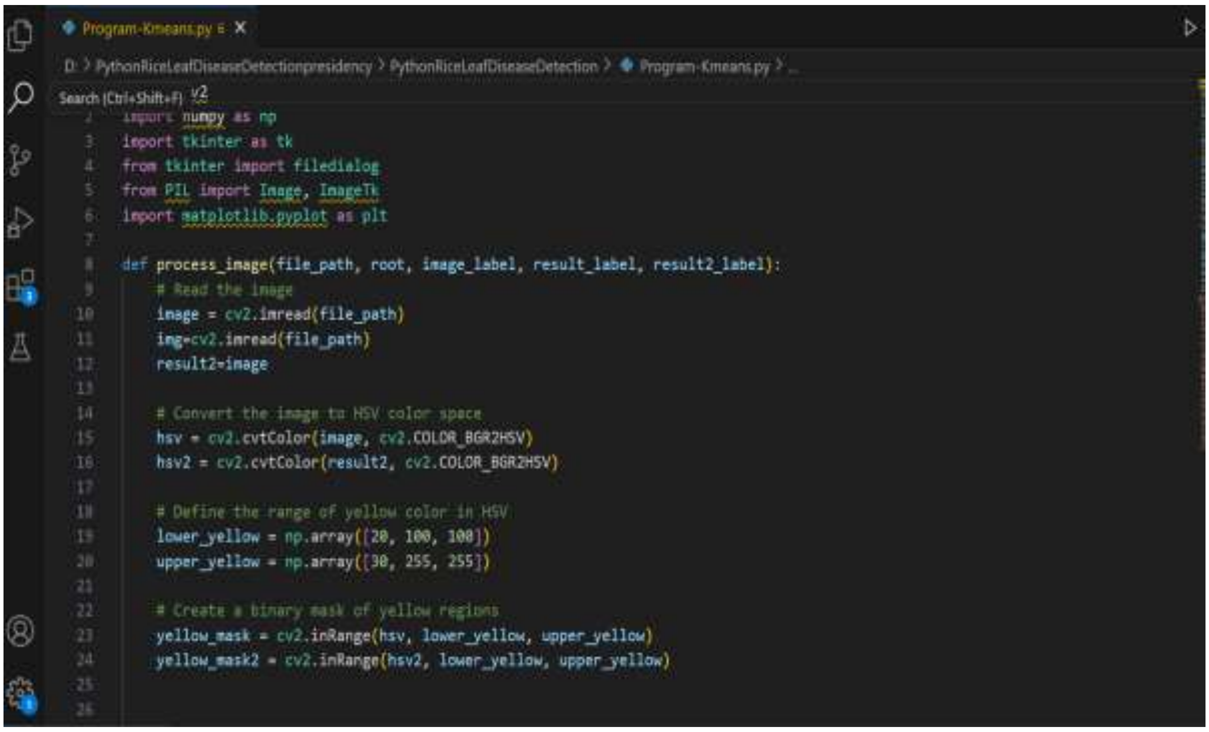
```
# Convert it to HSV
folder='Bacterial blight'
for filename in os.listdir(folder):
    img2 = cv2.imread(os.path.join(folder,filename))
    if img2 is not None:
        img1_hsv = cv2.cvtColor(img1, cv2.COLOR_BGR2HSV)
        img2_hsv = cv2.cvtColor(img2, cv2.COLOR_BGR2HSV)

# Calculate the histogram and normalize it
    hist_img1 = cv2.calcHist([img1_hsv], [0,1], None, [180,256], [0,180,0,256])
    cv2.normalize(hist_img1, hist_img1, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX);
    hist_img2 = cv2.calcHist([img2_hsv], [0,1], None, [180,256], [0,180,0,256])
    cv2.normalize(hist_img2, hist_img2, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX);

# find the metric value
    metric_val = cv2.compareHist(hist_img1, hist_img2, cv2.HISTCMP_BHATTACHARYYA)
    if metric_val == 0.0:
        print("Bacterial blight")
        cv2.imshow("Bacterial blight",img1)
        found=found+1
#print("\n")
_, _, boxes, _ = cv2.connectedComponentsWithStats(binary_img)
# first box is the background
boxes = boxes[1:]
filtered_boxes = []
rot=0
```


RiceLeaf - Main.py 6
D: > PythonRiceLeafDiseaseDetectionpresidency > PythonRiceLeafDiseaseDetection > RiceLeaf - Main.py > ...

```python

# Calculate the histogram and normalize it
    hist_img1 = cv2.calcHist([img1_hsv], [0,1], None, [180,256], [0,180,0,256])
    cv2.normalize(hist_img1, hist_img1, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX);
    hist_img2 = cv2.calcHist([img2_hsv], [0,1], None, [180,256], [0,180,0,256])
    cv2.normalize(hist_img2, hist_img2, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX);

# find the metric value
    metric_val = cv2.compareHist(hist_img1, hist_img2, cv2.HISTCMP_BHATTACHARYYA)
    if metric_val == 0.0:
        print("Blast")
        cv2.imshow("Blast",img)

        Image1 = cv2.imread('./blast/b1.png')
        Image1=cv2.resize(Image1, dsize=(100, 100))
        Image2 = cv2.imread('./blast/b2.png')
        Image2=cv2.resize(Image2, dsize=(100, 100))
        Image3 = cv2.imread('./blast/b3.png')
        Image3=cv2.resize(Image3, dsize=(100, 100))
        Image4 = cv2.imread('./blast/b4.png')
        Image4=cv2.resize(Image4, dsize=(100, 100))
        """
        Hori = np.concatenate((Image1, Image2,Image3,Image4), axis=1)
        #Verti = np.concatenate((Image3, Image4), axis=0)
        cv2.namedWindow("Resized_Window", cv2.WINDOW_NORMAL)
```


RiceLeaf - Main.py 6
D: > PythonRiceLeafDiseaseDetectionpresidency > PythonRiceLeafDiseaseDetection > RiceLeaf - Main.py > ...

```python
            fig = plt.figure(figsize=(8, 8))
            fig.add_subplot(rows, columns, 1)
            plt.imshow(Image1)
            plt.axis('off')
            plt.title("Blast: First")
            fig.add_subplot(rows, columns, 2)
            plt.imshow(Image2)
            plt.axis('off')
            plt.title("Second")
            fig.add_subplot(rows, columns, 3)
            plt.imshow(Image3)
            plt.axis('off')
            plt.title("Third")
            fig.add_subplot(rows, columns, 4)
            plt.imshow(Image4)
            plt.axis('off')
            plt.title("Fourth")
            plt.show()
            found=found+1
#print("\n")

folder='Brown spot'
for filename in os.listdir(folder):
    img2 = cv2.imread(os.path.join(folder,filename))
    if img2 is not None:
        img1_hsv = cv2.cvtColor(img1, cv2.COLOR_BGR2HSV)
```

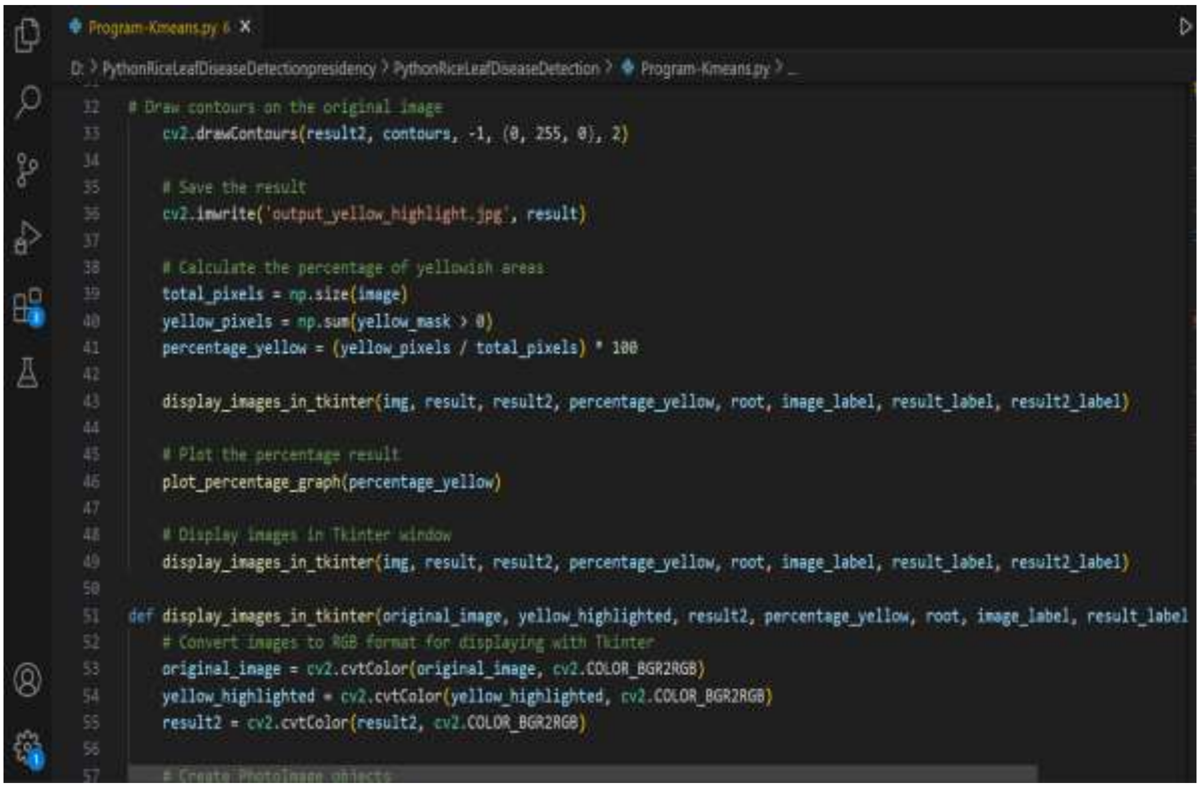


Program-Kmeans.py 6 ×

D: > PythonRiceLeafDiseaseDetectionpresidency > PythonRiceLeafDiseaseDetection > Program-Kmeans.py > ...

Search (Ctrl+Shift+F) v2

```python
import numpy as np
import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk
import matplotlib.pyplot as plt

def process_image(file_path, root, image_label, result_label, result2_label):
    # Read the image
    image = cv2.imread(file_path)
    img=cv2.imread(file_path)
    result2=image

    # Convert the image to HSV color space
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    hsv2 = cv2.cvtColor(result2, cv2.COLOR_BGR2HSV)

    # Define the range of yellow color in HSV
    lower_yellow = np.array([20, 100, 100])
    upper_yellow = np.array([30, 255, 255])

    # Create a binary mask of yellow regions
    yellow_mask = cv2.inRange(hsv, lower_yellow, upper_yellow)
    yellow_mask2 = cv2.inRange(hsv2, lower_yellow, upper_yellow)
```



Program-Kmeans.py 6 ×

D: > PythonRiceLeafDiseaseDetectionpresidency > PythonRiceLeafDiseaseDetection > Program-Kmeans.py > ...

```python
# Draw contours on the original image
    cv2.drawContours(result2, contours, -1, (0, 255, 0), 2)

    # Save the result
    cv2.imwrite('output_yellow_highlight.jpg', result)

    # Calculate the percentage of yellowish areas
    total_pixels = np.size(image)
    yellow_pixels = np.sum(yellow_mask > 0)
    percentage_yellow = (yellow_pixels / total_pixels) * 100

    display_images_in_tkinter(img, result, result2, percentage_yellow, root, image_label, result_label, result2_label)

    # Plot the percentage result
    plot_percentage_graph(percentage_yellow)

    # Display images in Tkinter window
    display_images_in_tkinter(img, result, result2, percentage_yellow, root, image_label, result_label, result2_label)

def display_images_in_tkinter(original_image, yellow_highlighted, result2, percentage_yellow, root, image_label, result_label
    # Convert images to RGB format for displaying with Tkinter
    original_image = cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB)
    yellow_highlighted = cv2.cvtColor(yellow_highlighted, cv2.COLOR_BGR2RGB)
    result2 = cv2.cvtColor(result2, cv2.COLOR_BGR2RGB)

    # Create PhotoImage objects
```

APPENDIX-C
ENCLOSURES

1. Conference Paper Presented Certificates of all students.

2. Include certificate(s) of any Achievement/Award won in any project related event.

3. Similarity Index / Plagiarism Check report clearly showing the Percentage (%). No need of page-wise explanation.