

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY,
BELGAUM, KARNATAKA**



**PROJECT REPORT
ON
“EARLY DETECTION OF PLANT DISEASES”**

Submitted in partial fulfilment of the requirement for the award of the degree of
BACHELOR OF ENGINEERING

**IN
COMPUTER SCIENCE AND ENGINEERING**

Submitted by

USN

NAME

2SD16CS008

AISHWARYA SAJJANAR

2SD16CS064

PAYAL GHORPADE

2SD16CS076

SALONI PORWAL

2SD16CS107

SUJIT BHOSALE

**Under the Guidance of
Dr. PUSHPALATHA S. NIKKAM**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
S.D.M. COLLEGE OF ENGINEERING & TECHNOLOGY,
DHARWAD-580002
2019-2020**

**S.D.M COLLEGE OF ENGINEERING & TECHNOLOGY,
DHARWAD –580002**



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CERTIFICATE

Certified that the project work and presentation entitled “**EARLY DETECTION OF PLANT DISEASES**” is a bonafide work carried out by **Aishwarya Sajjanar (2SD16CS008)**, **Payal Ghorpade (2SD16CS064)**, **Saloni Porwal (2SD16CS076)**, **Sujit Bhosale (2SD16CS107)** students of S. D. M. College of Engineering & Technology, Dharwad, in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belgaum**, during the year 2019-2020. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the department library. The Project has been approved, as it satisfies the academic requirements in respect of project report prescribed for the said degree.

Dr. Pushpalatha S. Nikkam

Project Guide

Dr. U P Kulkarni

HoD-CSE

External Viva

Name of Examiners

Signature with date

1. _____

2. _____

DECLARATION

We hereby declare that the dissertation work titled “**Early Detection of Plant Diseases**”, has been carried out under the guidance of **Dr. Pushpalatha S. Nikkam**, Assistant Professor, Department of Computer Science and Engineering, S.D.M. College of Engineering and Technology, Dharwad, in partial fulfillment of the degree of Bachelor of Engineering in Computer Science and Engineering from Visvesvaraya Technological University, Belgaum, Karnataka, during the academic year 2019–2020.

I also declare that I have not submitted this dissertation to any other university for the award of any other degree.

Place : Dharwad

Date:

NAME

SIGNATURE

1. AISHWARYA SAJJANAR
2. PAYAL GHORPADE
3. SALONI PORWAL
4. SUJIT BHOSALE

ACKNOWLEDGEMENT

We consider it a privilege to express our sincere gratitude and respect to all those who guided and inspired us throughout this project.

We express our deep sense of gratitude to our project supervisor **Dr. Pushpalatha S. Nikkam**, Assistant Professor, Department of Computer Science and Engineering, S. D. M. College of Engineering and Technology, Dharwad, for her insightful guidance, priceless intellectual motivation throughout our work. We learnt many moral and ethical standards in academic and non-academic issues from her. She is truly a source of inspiration to pursue a research-oriented career. The successful completion of this project owes to her coordination and streamlining of the project progress.

We wish to express our indebtedness to **Prof. Indira R. Umarji**, project coordinator, Assistant Professor, Department of Computer Science and Engineering, S. D. M. College of Engineering and Technology, Dharwad, for her valuable suggestion and constant support.

we have great pleasure in expressing our heartfelt gratitude to members of the project review committee **Dr. Umakant P. Kulkarni**, Professor & Head, **Dr. Shrihari M. Joshi**, **Prof. Ranganath G. Yadvad**, Dept. of CSE, for their constructive comments and scrutiny which enabled us in enhancing our work. We wish to offer our sincere thanks to all Faculty members of CSE Department, SDM College of Engineering, Dharwad for their support and encouragement.

Finally, heartfelt thanks to all **our family members** and **friends** without whose support we would not have completed our project.

ABSTRACT

The aim of the proposed work is to detect the diseases of tomato leaves using the simplest approach while making use of minimal computing resources to achieve accurate results by comparing different algorithms - k-Nearest Neighbor, Decision Tree, Support vector machine and Convolutional neural network to predict the type of disease encountered. An efficient, accurate and user-friendly system is built to help farmers detect diseases and consider the required steps to overcome further affects.

PROBLEM STATEMENT

Early Detection of Plant Diseases Using Digital Image Processing Techniques.

An efficient development of the project on the agriculture sector provides early detection of plant diseases and helps to increase production and contribute to the commercialization of farmer's crops.

Aim of the project is to compare different algorithms namely k-Nearest Neighbor, Decision tree, Support Vector Machine, Convolutional Neural Network to identify the algorithm that accurately predicts the type of disease.

Contents

DECLARATION	i
ACKNOWLEDGEMENT	ii
ABSTRACT	iii
PROBLEM STATEMENT	iv
INTRODUCTION	1
1.1 Overview.....	1
1.2 Objective.....	1
1.3 Plant description	2
1.4 Technical Overview	3
1.4.1 k-nearest neighbor (k NN)	3
1.4.3 Decision Tree Classifier	3
1.4.2 Support Vector Machine (SVM).....	3
1.4.2 Convolution neural network (CNN)	3
LITERATURE SURVEY	4
DETAILED DESIGN	8
3.1 Methodologies	8
3.2 KNN Architecture.....	12
3.3 Decision Tree Architecture	13
3.4 SVM Architecture.....	14
3.5 CNN Architecture.....	15
PROJECT SPECIFIC REQUIREMENTS	16
4.1 Functional and Non-Functional Requirements	16
4.1.1 Functional Requirements.....	16
4.1.2 Non-Functional Requirements.....	16
4.2 Basic Requirements	17
4.2.1 Hardware Requirements.....	17
4.2.2 Software Requirements	17
IMPLEMENTATION	18
5.1 k-NN algorithm.....	18
5.1.1 Reading database of images and Classifying	18

5.1.2 Creating a Training dataset.....	18
5.1.3 Randomly classify the image after training	19
5.1.4 Accuracy Calculation.....	19
5.1.5 Prediction.....	20
5.2 Decision Tree Classifier Algorithm	21
5.2.1 Algorithm	21
5.2.2 Reading images from specified folder and extracting the features from it.	21
5.2.3 Creating a Training dataset.....	22
5.2.4 The Decision tree classifier model.....	22
5.2.5 Accuracy Calculation.....	23
5.2.6 Prediction.....	23
5.3 SVM algorithm	24
5.3.1 Reading images from specified folder	24
5.3.2 Creating a Training dataset.....	24
5.3.3 Randomly classify the image after training	25
5.3.4 Accuracy Calculation.....	25
5.3.5 Graph - Confusion Metrics.....	26
5.3.6 Prediction.....	26
5.2 Convolutional Neural Network (CNN) Algorithm	28
5.4.1 Four layers of CNN	28
5.4.2 Compiling the Model, Fitting the dataset into the model and Training the model	28
5.4.3 Accuracy Calculation.....	29
5.4.4 Predictions	29
TEST CASES / TESTING.....	31
6.1 k-NN Algorithm.....	31
6.2 Decision Tree Classifier	32
6.3 SVM Algorithm.....	33
6.4 CNN Algorithm	34
CONCLUSION	35
REFERENCES	36

LIST OF FIGURES

<u>FIG.NO</u>	<u>FIGURE</u>	<u>PAGE NO.</u>
3.1	System Architecture	8
3.2	RGB to HSV and Gray scale	9
3.3	Image segmentation	10
3.4	Extracted features in decision tree	10
3.5	Pickle file	10
3.6	Feature extraction in CNN	11
3.7	k-NN System architecture	12
3.8	Decision tree System architecture	13
3.9	SVM System architecture	14
3.10	CNN System architecture	15
5.1	k-NN accuracy graph	19
5.2	Prediction using k-NN	20
5.3	Accuracy graph of Decision tree	23
5.4	Prediction using Decision tree	23
5.5	SVM accuracy graph	25
5.6	Prediction using SVM	27
5.7	Prediction using CNN	30

INTRODUCTION

1.1 Overview

Agriculture is the backbone of Indian economy with around 70% of the population earning their livelihood directly or indirectly from this sector. It is a critical sector as it plays major part in the development of the Indian economy and food security for the vast population of the country. Plant diseases are a major threat to the productivity of crops, which affects food security and reduces the profit of farmers. Identifying the diseases in plants is that the key to avoiding losses by proper feeding measures to cure the diseases early and avoiding the reduction in productivity or profit. A special attention for development planning in the field of agriculture is needed which will eventually help the sector to sustain, hence a sustainable agriculture would be the choice.





1.2 Objective

The main objective is to:

1. Learn the properties of Digital Image Processing.
2. Implement four different algorithms.
3. Make comparative analysis.
4. Evaluate the performance of the algorithm.

1.3 Plant description

The different diseases used in this project include Bacterial spot, Early Blight, Late blight, and Leaf mold. A few characteristic differentiating features are listed below:

Disease	Characteristic feature	Image
Bacterial Spot	<ul style="list-style-type: none">-caused by four species of <i>Xanthomonas</i>-leads to defoliation, sun-scalded fruit, and yield loss-leaf lesions are initially circular and water-soaked and may be surrounded by a faint yellow halo- spots are dark brown to black and circular on leaves and stems	
Early Blight (<i>Alternaria solani</i>)	<ul style="list-style-type: none">-early blight is caused by <i>Alternaria solani</i>-small necrotic spots that appear dry and papery.-as lesions enlarge they usually produce concentric rings giving the lesion a target-like appearance.-also cause cankers on the tomato stems	
Late Blight	<ul style="list-style-type: none">- caused by the fungus <i>Phytophthora infestans</i>-first appears on the lower, older leaves as water-soaked, gray-green spots-spots darken and a white fungal growth forms on the undersides-the spores are introduced by infected tubers, transplants or seeds	
Leaf Mold	<ul style="list-style-type: none">- caused by the fungus <i>Passalora fulva</i>-the oldest leaves are infected first.-pale greenish-yellow spots, usually less than 1/4 inch, with no definite margins, form on upper sides of leaves.-olive-green to brown velvety mold forms on the lower leaf surface below leaf spots.	

1.4 Technical Overview

The proposed decision-making system utilizes image content characterization and supervised classifier type back propagation with k-Nearest Neighbors (KNN), Convolution neural network (CNN), Support Vector Machine (SVM), Decision Tree Classifier. Image processing techniques for the decision analysis involves pre-processing, feature extraction and classification stage.

1.4.1 k-nearest neighbor (k NN)

The k-nearest neighbors' algorithm (k-NN) could also be a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples within the feature space. The output depends on whether k-NN is employed for classification or regression.

1.4.3 Decision Tree Classifier

Decision Tree classifier algorithms that can be used for classification or regression predictive modeling problems. Classification and Regression Trees (CART) algorithm is a classification algorithm for building a decision tree based on Gini's impurity index as splitting criterion.

1.4.2 Support Vector Machine (SVM)

The algorithm creates a line or a hyper-plane which separates the information into classes. A support vector machine (SVM) may be a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving SVM model sets of labeled training data for each category, they are able to categorize new image.

1.4.2 Convolution neural network (CNN)

A convolutional neural network (CNN) may be a sort of artificial neural network utilized in image recognition and processing that's specifically designed to process pixel data. CNN have their “neurons” arranged more like network form.

LITERATURE SURVEY

Indriani, Kusuma, Rachmawanto state that it's necessary to acknowledge the acceptable pattern to work out the amount of maturity. one in all the popularity patterns of tomatoes image is to use texture and color analysis. Texture analysis may be processed mistreatment the grey Level Co-occurrence Matrix (GLCM) technique. moreover, for color analysis one in all the ways which will be used is Hue, Saturation, worth (HSV). By utilizing HSV, associate degree object with an explicit color may be detected and cut back the influence of the intensity of sunshine from outside. The results of GLCM and HSV calculations may be classified to work out the maturity level of tomatoes by mistreatment K-Nearest Neighbor (K-NN) one in all basic and easy classification technique that utilizes the space (k) as a comparison of the similarity level of the image [1].

Shivali Sharma, Er. Varinderjit Kaur, Dr. Naveen Dhillon mention in their paper that early recognition of malady in plants may be fruitful. it will be accomplished with continues observance of plant by consultants and it needs high price just in case with giant farms. This study developed a KNN-SVM primarily based disease recognition system by mistreatment the method of feature extraction with the assistance of Color, form and Gabor filter feature extraction model, clump by applying KNN and SVM primarily based classification. The planned technique is 0.3077 and 0.0192 severally which implies that error rate has reduced and Accuracy is 98.0769% and 92.3077% severally that shows improvement within the potency. Consequently, the planned technique performs aptly in terms of police work a malady in plants pictures [2].

There are several types of plant disease could cause several losses to the production of crops. The presence of the pathogen depends on the favorable environment conditions and varieties of crops grown, which is the reason for occurrence and prevalence of plant diseases. There are various disease management programs which will help to scale back losses in yields and grain quality. From the three or four decades, research in the field of plant disease recognition and classification contributed greatly. The correct identification of diseases on early bases can help in preventing losses produce a high-quality yield of great good grain. Image recognition of plant diseases is a widely concern now these days, which develops the visual applications and also the reason for the popularization of digital technologies. The classification accuracy of the six different kinds of tomato plant images is yielding classification 97.3%. The tomato plant images categorized into Healthy, and 4 types of unhealthy diseases, mainly early blight, late blight, leaf mold, bacterial spot. Further, subdivided into five types of disease and healthy. These images segmented by using otsu's method into segmented binary and color images. Then total 24 combined features extracted and fed into classification tree. The result is 97.3% classification accuracy as a classifier [3].

The area of automation of plant disease recognition using computer vision is widely growing extensively. The study proposed and conducted within the paper is predicated on the thought to automate the disease recognition within the mean of digital image processing. The algorithm includes capturing of six differing types of disease infected leaf images, image standardization, image segmentation, feature extraction, normalization and recognition by classification employing a decision tree. In the feature extraction phase, total ten intensity based statistical features computed for every disease. These extracted features submitted to the choice tree for classification. The algorithm evaluates the plant disease recognition accuracy using classification technique called decision tree. The decision tree-based classifier follows the criteria of if-then rules. We considered five types of tomato plant diseases and healthy images for the implementation. Features and submitted to classifier-based decision tree. We assumed six classes for recognition include disease infected and healthy. The recognition accuracy of bacterial canker: 84.6%, bacterial leaf spot:69.2%, late blight: 80.7%, septoria leaf spot: 92.3%, leaf curl:70% and healthy recognized with 70%. The overall recognition accuracy yielded 78% and that is quite satisfactory with an error of incorrect recognition is 22% [4].

Detection of plant disease is essential research topic. The plant diseases are caused by fungi, bacteria, and viruses. The most common method used by farmers for detection of is by naked eyes, which required experience which totally depends on knowledge of their ancestors. Another economical option is discussed in this paper that is using image processing. All the diseases have common characteristics that they change some morphological characteristics such as color or shape. These changes can be extracted through image segmentation and classification of different disease through Multiclass SVM algorithm. There are many methods and algorithms that are being implemented on different plants for different diseases. Three steps that are followed by many of the authors: (1) Image Pre-processing: At initial stage raw image is taken as input and smoothen through various techniques like thresholding, grey scale conversion, RGB to HIS, various filters like median filter, CIBAL color model to remove noise and many more image smoothing techniques. (2) Image segmentation: the disease affected area is segmented from the leaf. For segmentation various methods are used such as k-nearest neighbor method, triangle threshold method and simple threshold methods. (3) Disease classification: For disease classification numerous algorithms are available. SVM is the most commonly used algorithm [5].

The proposed SVM-based tomato diseases detection approach is comprised of the following four fundamental building phases: (1) Image acquisition phase: The proposed SVM based tomato diseases detection approach begins with acquiring and collecting digital images (2) Pre-processing phase: To improve the quality of the images and to make the feature extraction phase more reliable prepare, some image-processing techniques are applied including leaf image isolation, image resizing and background removing. (3) Feature extraction phase: texture-based features are extracted, and represented during a database and (4) Classification phase: The last phase is that the classification and prediction of new objects, it is dependent on the SVM classifier [6].

The convolution neural network model to discover tomato pests and diseases supported VGG16 and transfer learning. The VGG16 deep model is discharged by the University of Oxford within the 2014 ILSVRC competition and has achieved the championship for the year's image recognition cluster. Because the original task, VGG16 is trained with one.26 million pictures to classify 1000-category pictures, whereas the target task is to classify 10- class leaf pictures of tomato diseases. This paper builds a picture dataset of tomato diseases and pests, that contains leaf pictures of 10 common tomato diseases in China furthermore as some traditional ones. The dataset includes a complete of 11 classes, 7040 pictures, every class has 640 pictures. The detection model's area unit trained to classify the tomato diseases and pests by transfer learning technology that achieves a median classification accuracy of eighty-nine [7].

The planned methodology consists of 3 major steps: knowledge acquisition, pre-processing and classification. The photographs used for the implementation of the planned methodology were non-inheritable from a in public obtainable dataset referred to as Plant Village, as mentioned earlier. Within the next step, the photographs were re-sized to a typical size before feeding it into the classification model. The ultimate step is that the classification of the input pictures with the employment of a small variation of the deep learning convolutional neural network (CNN) commonplace model referred to as the LeNet that consists of the convolutional, activation, pooling and totally connected layers. Neural network models use automatic feature extraction to assist within the classification of the input image into various malady categories. This planned system has achieved a median accuracy of 94-95% indicating the practicability of the neural network approach even beneath unfavorable conditions [8].

DETAILED DESIGN

3.1 Methodologies

1. Pre-processing
2. Color Space Conversion
3. Color and Texture Features Extraction

Proposed methodology starts with the acquisition of captured images. After acquisition of image, pre-processing and filtering is done to remove unwanted noise from the image. The pre-processed and filtered image is converted into the binary image and its color features are extracted. Based on feature extraction results estimation of the affected area is done. Depending on percentage of the affected area grading and classification is done.

SYSTEM ARCHITECTURE

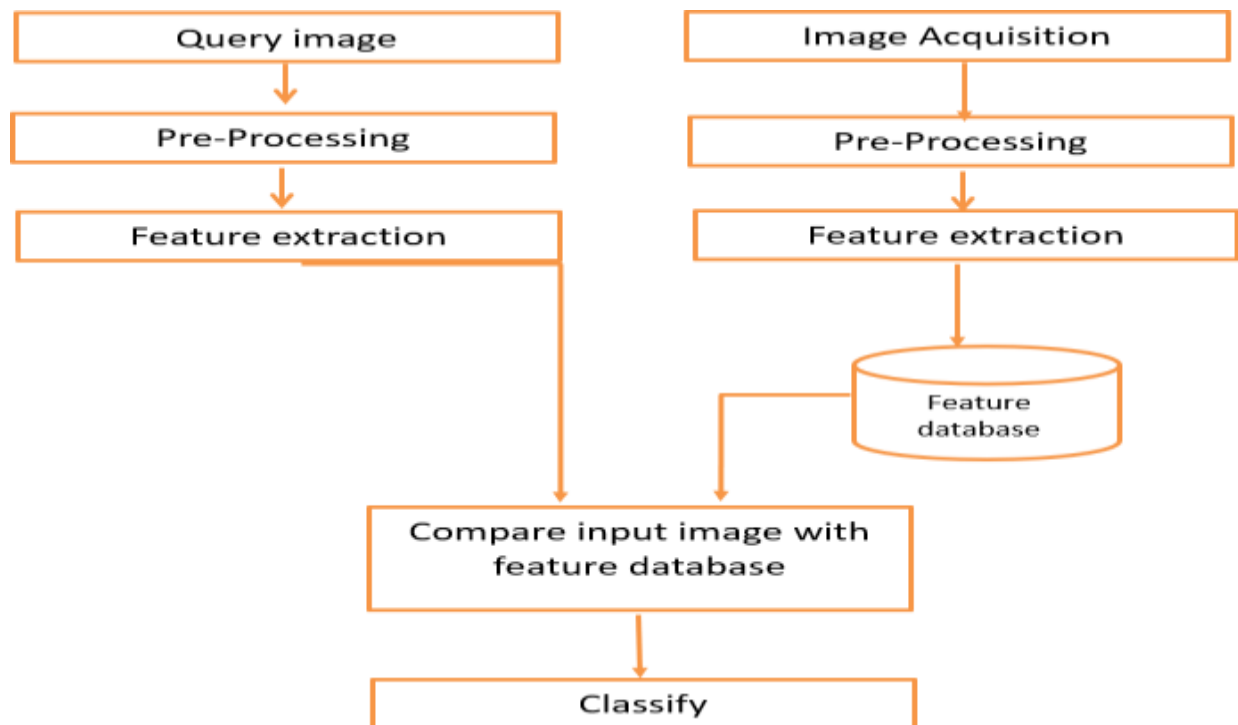


Fig 3.1 – System Architecture

Image Acquisition:

Images of varied leaves are acquired employing a camera with 16+20 MP resolution for better quality. This image is in RGB (Red, Green and Blue) form. The created database consists of 500 to 1000 acquired images consisting of four diseases namely bacterial spot, early blight, late blight, leaf mold and healthy leaves.

Image Pre-processing:

To remove noise in an image or other object removal, different pre-processing techniques are taken into account such as conversion of RGB image into HSV, gray scale (Binary image) and smoothing of image. Image clipping of the leaf image to get the interested image region is done using image resize. The conversion of RGB images into the binary images using color conversion equation.

$$f(x) = 0.2989 * R + 0.5870 * G + 0.114 * B$$

The conversion of RGB images into the HSV images using color conversion equation.

$$h = (60 * ((g - b) / \text{diff}) + 360) \% 360$$

$$s = (\text{diff} / c_{\text{max}}) * 100$$

$$v = c_{\text{max}} * 100$$

where $c_{\text{max}} = \max(r, \max(g, b))$

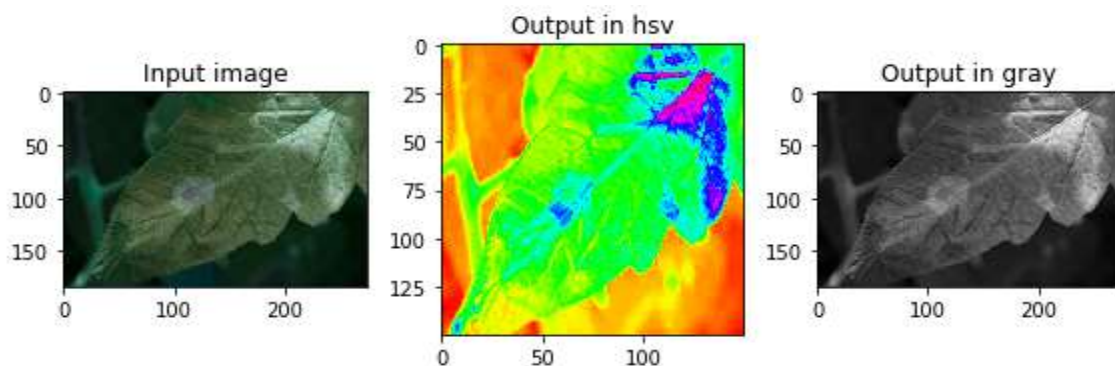
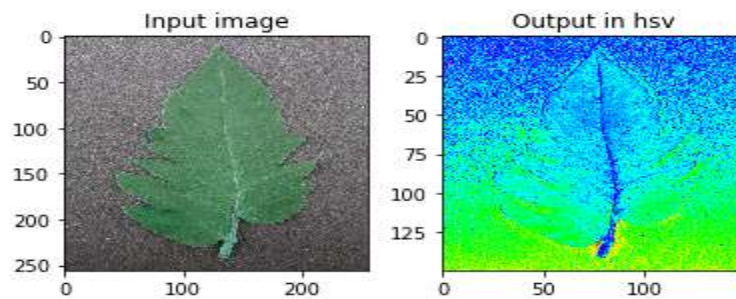


Fig 3.2 – RGB to HSV and gray scale

Image Segmentation:

Image segmentation is the sequence of steps involved in dividing a digital image into numerous segments. The objective of the separation is to make it simple and/or alter the picture of an image into more meaningful and simpler to examine. Image segmentation is essentially used to place objects and boundaries in the images. In this phase, a binary masked image is converted into HSV image.

Fig 3.3 – Image segmentation



Feature Extraction:

Feature extraction plays a crucial role in the identification of an object. Color, intensity, height, width, saturation, and contrast, along with homogeneity and dissimilarity, are the features utilized in disease detection. In algorithms such as k-NN, SVM, and decision tree, the features are stored in comma-separated value (CSV) format. k-NN and SVM can also be implemented using pickle files.

ID	contrast	dissimilarity	homogeneity	ASM	energy	hue	saturation	value	label
0	586.685	17.8590074	0.07000097	0.000182	0.01349	104.3016	119.0275	39.74677	0
102	151.3342	7.0808364	0.2604688	0.000619	0.02487	86.87231	129.007	57.1711	1
200	254.9454	8.99866728	0.19827198	0.060581	0.024111	30.11552	157.2398	64.44061	2
300	117.3821	7.88662684	0.1377316	0.000459	0.021425	104.2605	120.5343	53.73763	3
400	170.9116	9.04710478	0.13595666	0.000257	0.016021	60.79973	100.6843	59.28644	4
499	152.5495	8.46979167	0.14739863	0.000339	0.018405	106.1108	112.8797	42.86357	4

Fig 3.4 – Extracted features in decision tree

```
X1[1]
array([[137],
       [150],
       [120],
       ...,
       [133],
       [130],
       [129]])
```

Fig 3.5 – Pickle file

Feature extraction in CNN:

The input image is a matrix that specifies the color of various pixels in terms of the amount of red, green, blue components. Every pixel value is between 0 and 255. A kernel or a convolutional filter is a square matrix that specifies spatial weights of various pixels in an image. Ex:

$$\text{Gaussian filter } 3 \times 3 \text{ matrix } \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad \text{Sharpen filter } 3 \times 3 \text{ matrix } \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Convolution layer of a matrix involves laying a matrix over another and then calculating the weighted sum of all the pixel values as shown below: Consider a 5 x 5 whose image pixel values are 0, 1 and filter matrix 3 x 3

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

 *

0	-1	0
-1	5	-1
0	-1	0

Then the convolution of 5 x 5
image matrix multiplies with 3
x 3 filter matrix which is
called “Feature Map”.

3	1	3
-2	2	1
-2	2	3

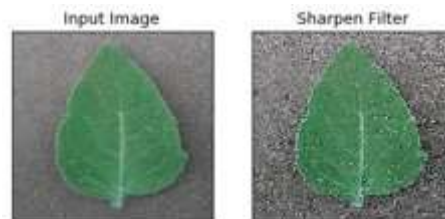


Fig 3.6 – Feature extraction in CNN

Feature database:

It consists of the feature extracted image of plant leaf. It is used as a reference to compare the input query image to that of the feature dataset. The dataset is further trained. On multiple training, the program becomes well versed with the different features of the images. The extracted features from the image will be stored in the database which is used for classification.

Testing of query image:

It compares the input image with feature database and classifies the disease that the plant has acquired.

Classify:

Depending on percentage of the affected area grading and classification is done.

3.2 KNN Architecture

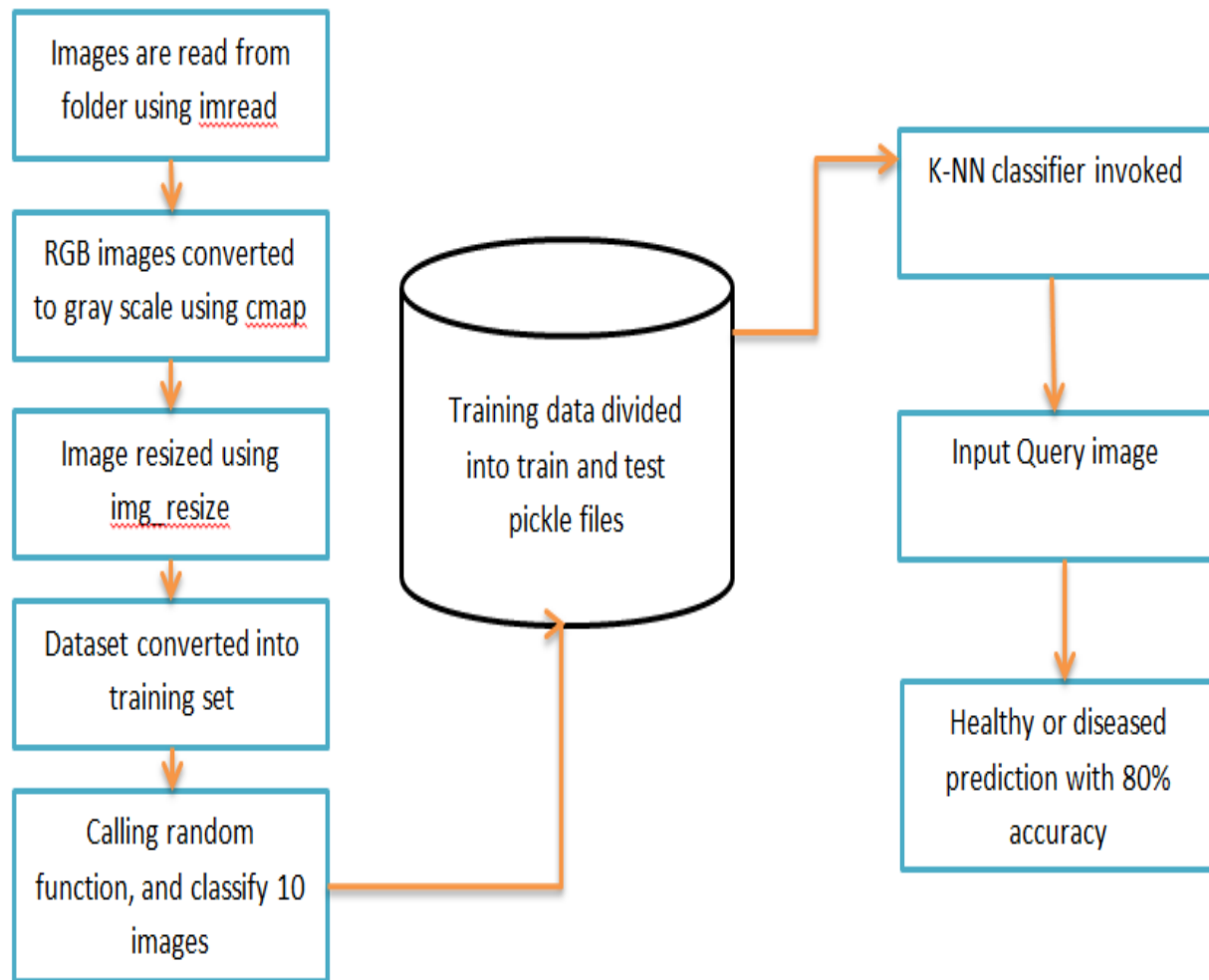


Fig 3.7 – KNN System Architecture

The architecture describes the reading of images using `imread ()` function and converting the image to gray scale and resizing it. These images are split into train and test set. Using Knn classifier the input query image is compared with the extracted feature dataset and prediction of the image is obtained that is healthy or diseased leaf plant with 80% accuracy.

3.3 Decision Tree Architecture

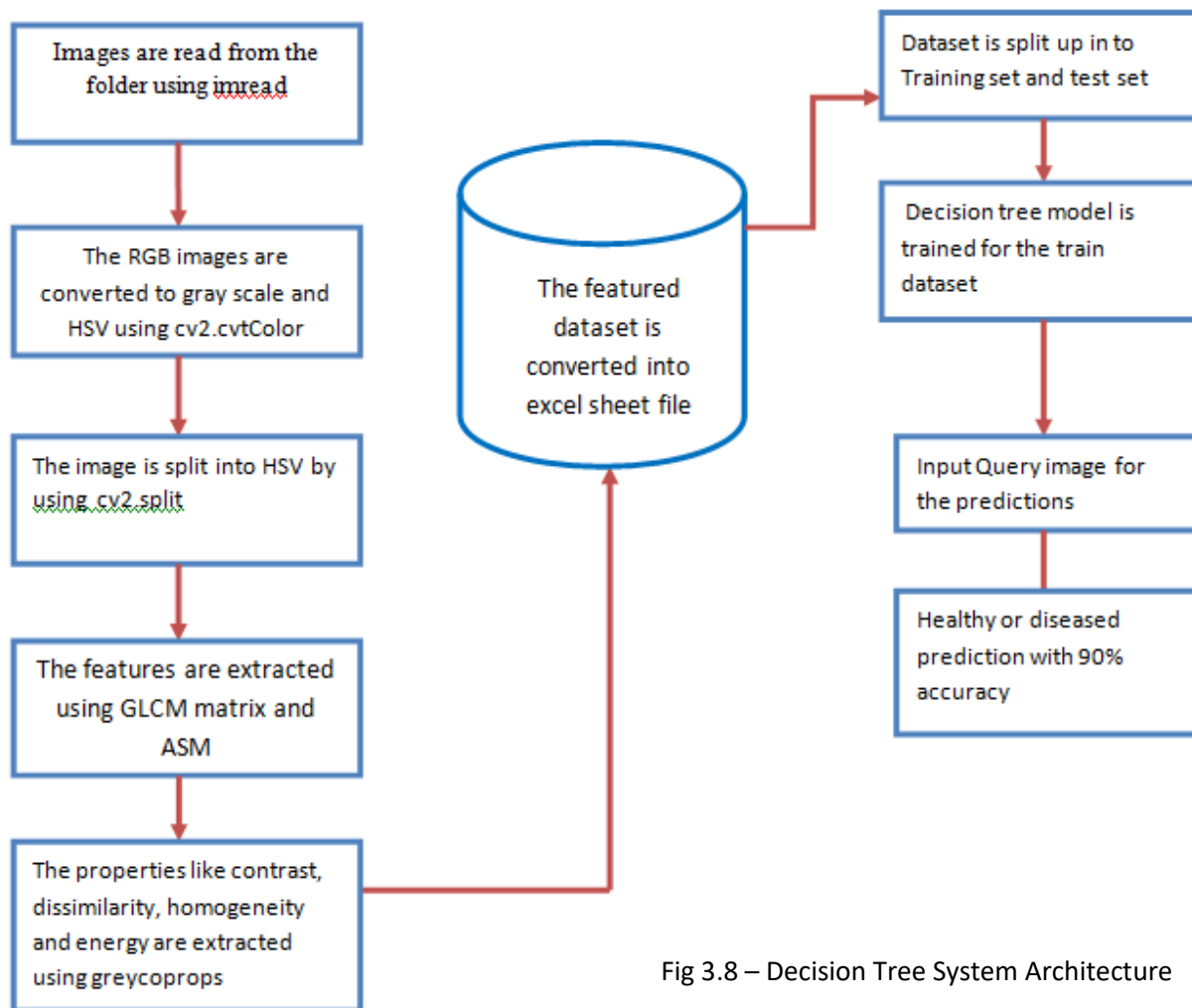


Fig 3.8 – Decision Tree System Architecture

The architecture describes about reading the images using `imread ()` function and converting the image to gray scale, HSV and resizing it. These images are split into train and test set. Using Decision tree classifier, the input query image is compared with the extracted feature dataset and prediction of the image is obtained that is healthy or diseased leaf plant with 90% accuracy.

3.4 SVM Architecture

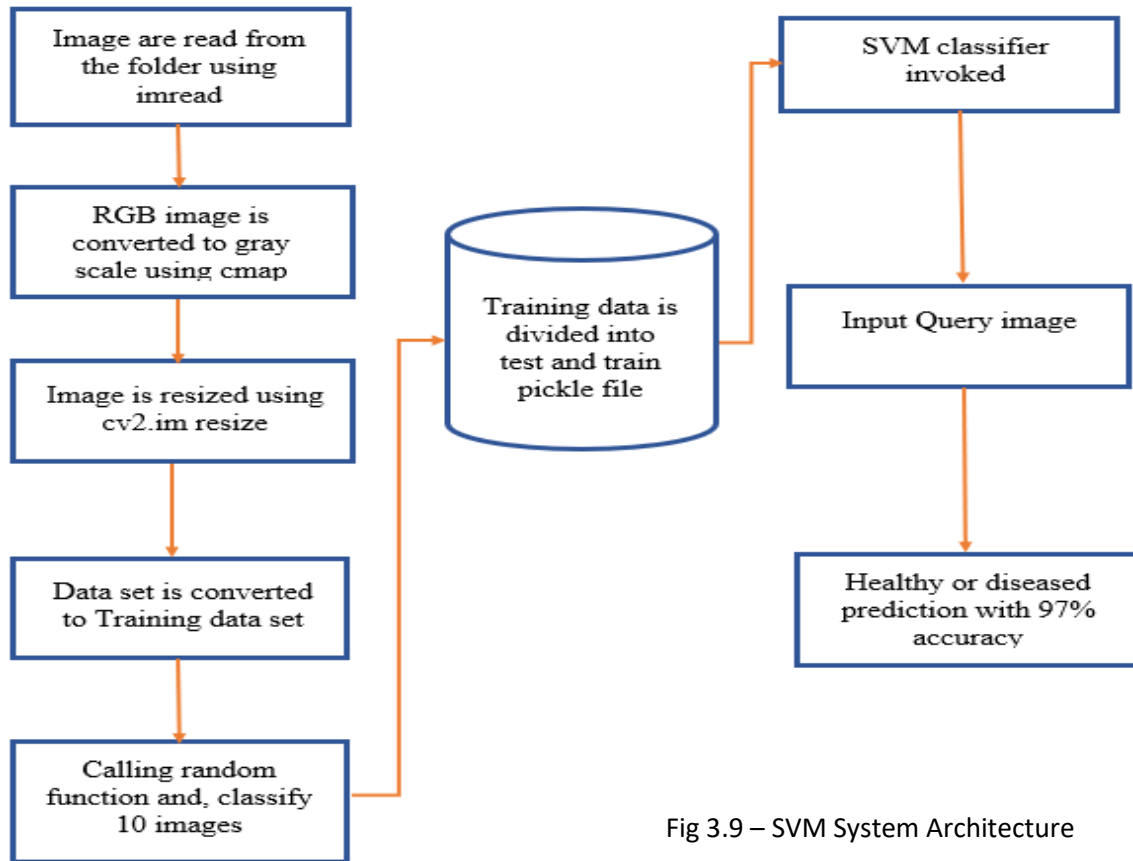


Fig 3.9 – SVM System Architecture

The SVM architecture describes how the image is read using `imread()` function and converting the image to gray scale and resizing it using `cv2.imresize`. These images are split into train and test set. Using SVM classifier the input query image is compared with the extracted feature dataset and prediction of the image is obtained that is healthy or diseased leaf plant with 97% accuracy.

3.5 CNN Architecture

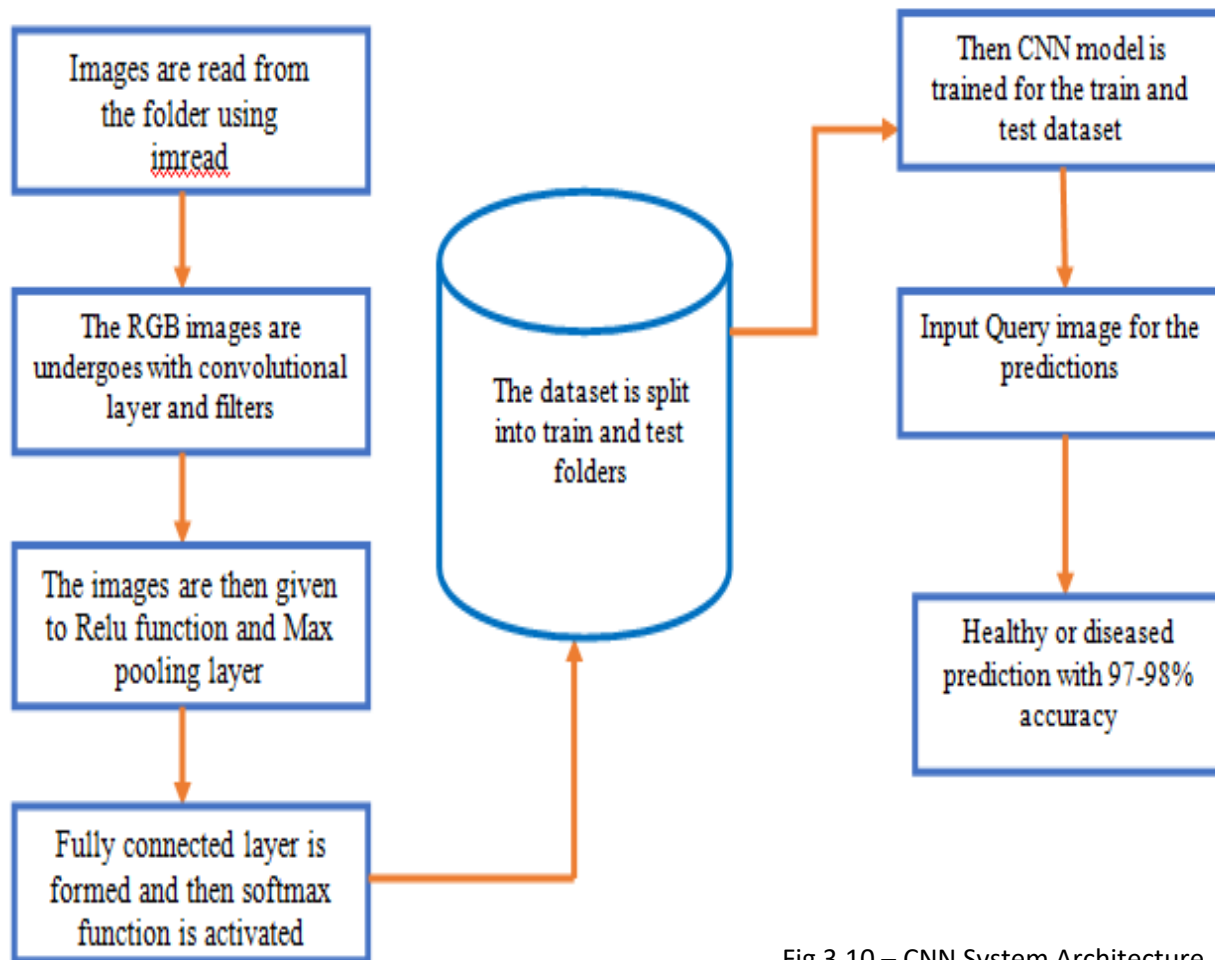


Fig 3.10 – CNN System Architecture

The architecture describes about reading the images using `imread ()` function and applying the CNN layers to it and resining it. These images are split into train and test set. Using CNN classifier, the input query image is compared with the extracted feature dataset and prediction of the image is obtained that is healthy or diseased leaf plant with 97-98% accuracy.

PROJECT SPECIFIC REQUIREMENTS

4.1 Functional and Non-Functional Requirements

4.1.1 Functional Requirements:

Inputs: Acquired dataset consisting of healthy and unhealthy plant leaf to train the system, crop details, and an input image for detecting the disease.

Behavior: once the farmer inputs the plant image, the image is processed and compared with the image stored in the database to identify if the plant is diseased or not. If diseased, classification of the disease type is done.

Output: Detection of diseases at an early stage to help the farmers to take the required precautions.

4.1.2 Non-Functional Requirements:

Performance: The application must be reliable and efficient. As soon as the input image is entered, processing should happen at a fast pace and accurately deliver the type of disease.

Data Integrity: The data entered, which is to be stored in the database must be stored accurately. The data when retrieved for comparison must be of the similar plant as input by the farmer.

Accessibility: The applications must be easy to access and easy to use.

Platform Compatibility: The software must execute in any digital or a computing platform.

4.2 Basic Requirements

4.2.1 Hardware Requirements: -

RAM: 8GB

Using more intense programs, such as image editors, multi-tab browsers with several add ons, generally benefit from having a greater amount of RAM

Processor: 8th Gen Intel core i5

Memory: 1TB

Ram: 16GB

High resolution Camera: 16 + 20 MP

Pixel segmentation and image analysis requires high resolution image.

Input device – phone, camera

Output device – Monitor

4.2.2 Software Requirements: -

Python – Version 3.7(64 bit)

Jupyter Notebook – Version 6.0.3

Libraries - Sklearn, Pandas, OpenCV, numpy, matplotlib

Data - Information about the crops and the type of disease

Operating System – Windows

IMPLEMENTATION

5.1 k-NN algorithm

5.1.1 Reading database of images and Classifying

```
import numpy as np
import matplotlib.pyplot as plt
import os
import cv2

DATADIR = "C:/Users/admin/Desktop/test"
CATEGORIES = ["BACTERIAL_SPOT", "EARLY_BLIGHT", "HEALTHY", "LATE_BLIGHT", "LEAF_MOLD"]

for category in CATEGORIES:
    path = os.path.join(DATADIR, category)
    for x in range(1,9):
        for imp in os.listdir(path):
            img_array = cv2.imread(os.path.join(path,imp), cv2.IMREAD_GRAYSCALE)
            plt.imshow(img_array, cmap="gray")
            plt.show()
            break
        break
    break
```

5.1.2 Creating a Training dataset

```
training_data = []

def create_training_data():
    for category in CATEGORIES:
        path = os.path.join(DATADIR, category)
        class_num = CATEGORIES.index(category)
        for imp in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path,imp), cv2.IMREAD_GRAYSCALE)
                new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
                training_data.append([new_array, class_num])
                #print(training_data[10])
            except Exception as e:
                pass

create_training_data()

print(len(training_data))
```

75

- The images from the folder are read in a loop and are all considered for training the program. All images in the folder are used for training.

5.1.3 Randomly classify the image after training

```

classify = []
for sample in training_data[:10]:
    #print(sample[1])
    classify.append(sample[1])
#print("new")
for classify in training_data[:20]:
    #print(classify[1])
    if classify[1] == 0:
        print(classify[1], "Bacterial_spot")
    elif classify[1] == 1:
        print(classify[1], "Early_Blight")
    elif classify[1] == 2:
        print(classify[1], "Healthy")
    elif classify[1] == 3:
        print(classify[1], "Late_Blight")
    else:
        print(classify[1], "Leaf_Mold")

```

```

4 Leaf_Mold
3 Late_Blight
3 Late_Blight
2 Healthy
2 Healthy
3 Late_Blight
1 Early_Blight
2 Healthy
3 Late_Blight
2 Healthy
2 Healthy
1 Early_Blight
0 Bacterial_spot

```

OUTPUT

- Calling the random() function, the program will consider a random of 10 images from the training data set and try to classify them into their respective folders.

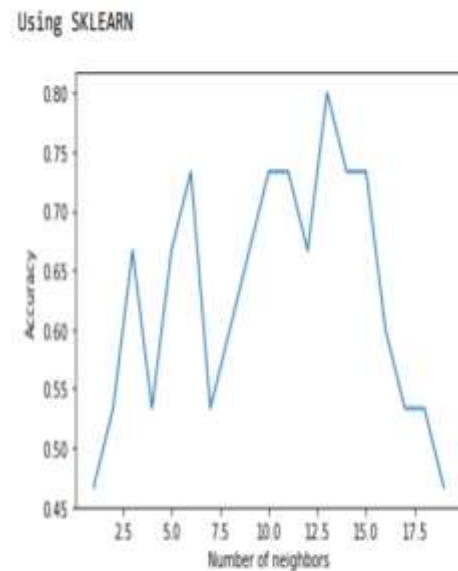
5.1.4 Accuracy Calculation

```

print("Using SKLEARN")
lix = []
liy = []
index=1
acc=0
from sklearn.neighbors import KNeighborsClassifier
for k in range(1, 20):
    neigh = KNeighborsClassifier(n_neighbors=k)
    neigh.fit(X_train, y_train)
    liy.append(neigh.score(X_test, y_test))
    if liy[k-1]>acc:
        acc=liy[k-1]
        index=k-1
    lix.append(k)

plt.plot(lix, liy)
plt.xlabel('Number of neighbors')
plt.ylabel('Accuracy')
plt.show()
print("max acc at k="+str(index+1)+" acc of "+str(acc))

```



max acc at k=13 acc of 0.8

Fig 5.1: k-NN Accuracy graph

5.1.5 Prediction

```
print("Predicting custom image")
labels = ["BACTERIAL_SPOT", "EARLY_BLIGHT", "HEALTHY", "LATE_BLIGHT", "LEAF_MOLD"]
img = cv2.imread(r"C:/Users/DELL/Desktop/lb1.JPG")
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_pred = cv2.resize(img, (120, 120))
img_pred1 = np.reshape(img_pred, (1, img_pred.shape[0]*img_pred.shape[1]))
#plt.imshow(img_pred)
e=knn.predict(img_pred1)
#print(labels[(e[0])])
NUM_ROWS = 1
IMGs_IN_ROW = 3
f, ax = plt.subplots(NUM_ROWS, IMGs_IN_ROW, figsize=(8,8))
img1 = cv2.imread(r"C:/Users/DELL/Desktop/lb1.JPG")
ax[0].imshow(img1)
ax[1].imshow(img_pred, cmap="hsv")
img3 = cv2.imread(r"C:/Users/DELL/Desktop/lb1.JPG")
img3 = cv2.cvtColor(img3, cv2.COLOR_BGR2GRAY)
ax[2].imshow(img3, cmap="gray")

ax[0].set_title('original image')
ax[1].set_title('Output in hsv')
ax[2].set_title('Output in gray')

title = 'predicting custom image'
#f.suptitle(title, fontsize=16)
plt.tight_layout()
plt.show()
print(labels[(e[0])])
```

Predicting custom image

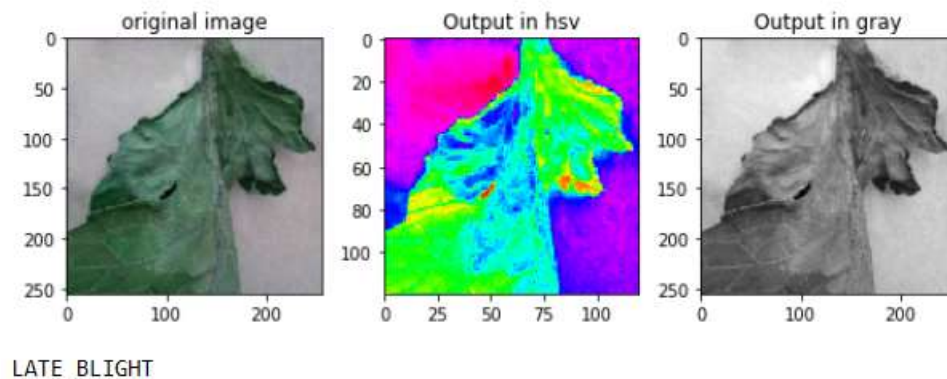


FIG 5.2: Prediction Using k-NN

5.2 Decision Tree Classifier Algorithm

5.2.1 Algorithm: STEPS FOR DECISION TREE GENERATION:

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using Angular Second Moment (ASM).
(Angular Second Moment: Energy, also called Angular Second Moment and Uniformity is a measure of textural Uniformity of an image)

Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

5.2.2 Reading images from specified folder and extracting the features from it.

```
INPUT_SCAN_FOLDER="C://Users//G3IN//Desktop//project and reports//delete//"+folder+"//"
image_folder_list = os.listdir(INPUT_SCAN_FOLDER)
for i in range(len(image_folder_list)):
    abc = cv2.imread(INPUT_SCAN_FOLDER+image_folder_list[i])
    gray_image = cv2.cvtColor(abc, cv2.COLOR_BGR2GRAY)
    hsv = cv2.cvtColor(abc, cv2.COLOR_BGR2HSV)
    h,s,v = cv2.split(hsv)
    h_mean = np.mean(h)
    h_mean = np.mean(h_mean)

    s_mean = np.mean(s)
    s_mean = np.mean(s_mean)

    v_mean = np.mean(v)
    v_mean = np.mean(v_mean)

    # images = images.f.arr_0
    print(image_folder_list[i])

    glcmMatrix = (greycomatrix(gray_image, [1], [0], levels=2 ** 8))
    for j in range(0, len(proList)):
        properties[j] = (greycoprops(glcmMatrix, prop=proList[j]))

    features = np.array(
        [properties[0], properties[1], properties[2], properties[3], properties[4],h_mean,s_mean,v_mean,
```

Feature Extraction aims to scale back the number of features during a dataset by creating new features from the prevailing ones (and then discarding the first features). These new reduced set of features should then be able to summarize most of the information contained in the original set of features.

5.2.3 Creating a Training dataset

Splitting the train and test set

```
#print(abc.head())
X=abc.iloc[:,1:]
#print(X)
Y=X.iloc[:,-1]
#print(Y)                                     # Stores labels in Y variable
X=X.iloc[:,0:-1]                             # Stores features in X variable
X_train, X_test, y_train, y_test = train_test_split(X,Y,test_size=0.22,random_state=17)
y_train = y_train.astype('int')#as we are splitting the data into 2 parts same way we hav
y_test = y_test.astype('int')
```

5.2.4 The Decision tree classifier model.

```
[('CART', DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                                max_features=None, max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, presort=False,
                                random_state=None, splitter='best'))]
```


5.2.5 Accuracy Calculation

```
for name,model in models:
    for n in range(0,9): #range for trials
        model.fit(X_train,y_train)
        liy.append(model.score(X_test, y_test))
        if liy[n-1]>accuracy:
            acc=liy[n-1]
            index=n-1
        lix.append(n)

plt.plot(lix, liy)
plt.ylabel("accuracy")
plt.xlabel("no. of trials")
plt.show()
```

the Accuracy of the model is 0.9043478260869565
CART 90.43478260869566

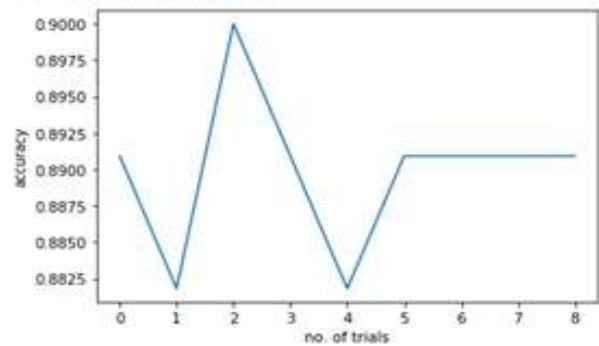


FIG 5.3: Accuracy Graph of Decision Tree

The respective code plots a comparative accuracy graph stating the no of trials at which the accuracy achieved is the highest. From the output, it can be concluded that on considering the test_set split=1.22, the highest accuracy of 90% can be achieved.

5.2.6 Prediction

```
new_image_features = extract_feature(r"C:/Users/G3IN/Desktop/project and reports/database1/test_set/late_blight/")
print(new_image_features)
new_predicted = model.predict(np.array([new_image_features]))
print(folders[new_predicted[0]])
print("label is",new_predicted)
img3 = cv2.imread(r"C:/Users/G3IN/Desktop/project and reports/database1/test_set/late_blight/0a4b3cde-c83a-4c83-")
img3 = cv2.cvtColor(img3, cv2.COLOR_BGR2RGB)
plt.imshow(img3)
plt.show()
```

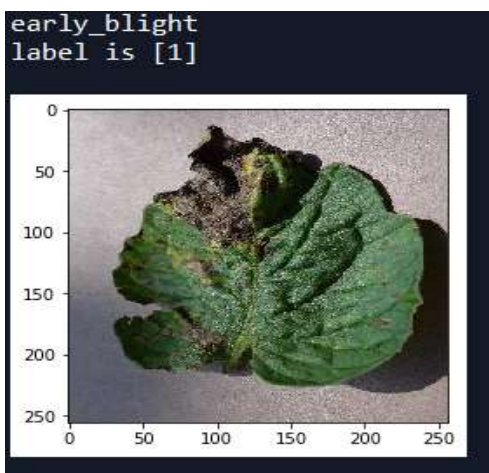


FIG 5.4: Prediction Using Decision Tree

5.3 SVM algorithm

5.3.1 Reading images from specified folder

```
import numpy as np
import matplotlib.pyplot as plt
import os
import cv2

#reading the dataset
DATADIR = "C:/Users/aishwarya/Desktop/Images2"
CATEGORIES = ["Bacterial_spot", "Early_blight", "HEALTHY", "Late_blight", "Leaf_mold"]

for category in CATEGORIES:
    label = CATEGORIES.index(category)
for category in CATEGORIES:
    path = os.path.join(DATADIR, category)
    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
        plt.imshow(img_array, cmap="gray")
        plt.show()
        break
    break
```

5.3.2 Creating a Training dataset

```
training_data = []

def create_training_data():
    for category in CATEGORIES:
        path = os.path.join(DATADIR, category)
        class_num = CATEGORIES.index(category)
        for imp in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path, imp), cv2.IMREAD_GRAYSCALE)
                new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
                training_data.append([new_array, class_num])
                #print(training_data[10])
            except Exception as e:
                pass

create_training_data()

print(len(training_data))
```

75

The images from the folder are read in a loop and are all considered for training the program. All images in the folder are used for training.

5.3.3 Randomly classify the image after training

```
import random
#randomly takes the training_data value and append in classify
random.shuffle(training_data)
classify = []
for sample in training_data[:10]:
    classify.append(sample[1])
#predicting which disease
for classify in training_data[:20]:
    if classify[1]==0:
        print(classify[1], "Bacterial_spot")
    if classify[1]==1:
        print(classify[1], "Early blight")
    if classify[1]==2:
        print(classify[1], "Healthy")
    if classify[1]==3:
        print(classify[1], "Late blight")
    if classify[1]==4:
        print(classify[1], "Leaf mold")
```

```
2 Healthy
3 Late blight
4 Leaf mold
4 Leaf mold
3 Late blight
0 Bacterial_spot
2 Healthy
3 Late blight
3 Late blight
4 Leaf mold
2 Healthy
2 Healthy
0 Bacterial_spot
2 Healthy
2 Healthy
3 Late blight
2 Healthy
3 Late blight
0 Bacterial_spot
2 Healthy
```

Calling the random () function, the program will consider a random of 10 images from the training data set and try to classify them into their respective folders.

5.3.4 Accuracy Calculation

```
from sklearn import svm
from sklearn import metrics

# Fitting SVC Classification to the Training set with poly kernel
# poly kernel is popular in image processing.
classifier=svm.SVC(kernel='poly',gamma='auto',C=2)
classifier.fit(X_train,Y_train)

# Predicting the Test set results
Y_predict=classifier.predict(X_test)

#evaluation metric of the classifier on test data produced when score() is called.
accuracy = classifier.score(X_test, Y_test)*100
print("Accuracy: ",accuracy)

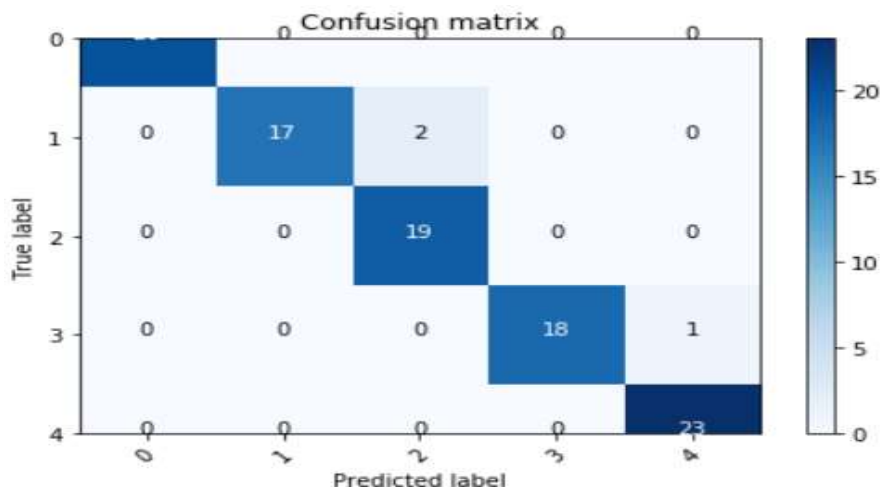
print("Precision Score : ",metrics.precision_score(Y_test, Y_predict,average='macro')*100)
print("Recall Score : ",metrics.recall_score(Y_test, Y_predict,average='macro')*100)
```

```
Accuracy: 97.0
Precision Score : 97.26190476190476
Recall Score : 96.84210526315789
```

FIG 5.5: Accuracy using SVM

5.3.5 Graph - Confusion Metrics

The diagonal elements represent the amount of points that the anticipated label is adequate to truth label, while off-diagonal elements are people who are mislabeled by the classifier. The higher the diagonal values of the confusion matrix the higher, indicating many correct predictions.



5.3.6 Prediction

```
labels = ["BACTERIAL_SPOT", "EARLY_BLIGHT", "HEALTHY", "LATE_BLIGHT", "LEAF_MOLD"]
img = cv2.imread(r"C:/Users/aishwarya/Desktop/h1.JPG")
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_pred = cv2.resize(img, (150, 150))
img_pred2 = np.reshape(img_pred, (1, img_pred.shape[0]*img_pred.shape[1]))
e=model.predict(img_pred2)

NUM_ROWS = 1
IMGs_IN_ROW = 3
f, ax = plt.subplots(NUM_ROWS, IMGs_IN_ROW, figsize=(8,8))
img1 = cv2.imread(r"C:/Users/aishwarya/Desktop/h1.JPG")
ax[0].imshow(img1)
ax[1].imshow(img_pred, cmap="hsv")
img3 = cv2.imread(r"C:/Users/aishwarya/Desktop/h1.JPG")
img3 = cv2.cvtColor(img3, cv2.COLOR_BGR2GRAY)
ax[2].imshow(img3, cmap="gray")
```

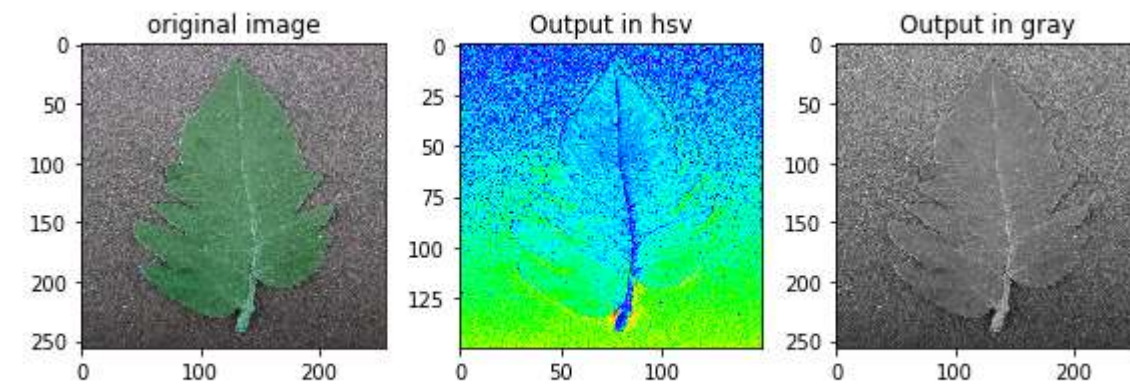
```

ax[0].set_title('original image')
ax[1].set_title('Output in hsv')
ax[2].set_title('Output in gray')

title = 'predicting custom image'|
#f.suptitle(title, fontsize=16)
plt.tight_layout()
plt.show()
print(labels[(e[0])])

```

predicting custom image



HEALTHY

FIG 5.6: Prediction Using SVM

5.2 Convolutional Neural Network (CNN) Algorithm

5.4.1 Four layers of CNN

```
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense

# Initialising the CNN
classifier = Sequential()

# Step 1 - Convolution          (128 or 256)          #(row,col,dimension)
#                               #(256, 256, 3) if GPU
classifier.add(Convolution2D(32, (5, 5), input_shape = (64, 64, 3), activation = 'relu'))

# Step 2 - Pooling
classifier.add(MaxPooling2D(pool_size = (3, 3)))

# Adding a second convolutional layer
classifier.add(Convolution2D(32, (5, 5), activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))

#Adding a third convolutional layer
classifier.add(Convolution2D(128, (3, 3), activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Step 3 - Flattening
classifier.add(Flatten())

# Step 4 - Full connection  128-Experiment with it
classifier.add(Dense(units = 500, activation = 'relu'))
classifier.add(Dense(units = 3, activation = 'softmax'))
```

5.4.2 Compiling the Model, Fitting the dataset into the model and Training the model

```
# Compiling the CNN
classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
#loss = 'categorical_crossentropy' for more than two categories
print(classifier.summary())
# Part 2 - Fitting the CNN to the images

from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.3,
                                   zoom_range = 0.3,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)

training_set = train_datagen.flow_from_directory(r'C:/Users/HP/Documents/loginware/major_project/database1/training_set',
                                                target_size = (64, 64),
                                                batch_size = 32,
                                                class_mode = 'categorical')

test_set = test_datagen.flow_from_directory(r'C:/Users/HP/Documents/loginware/major_project/database1/test_set/',
                                           target_size = (64, 64),
                                           batch_size = 32,
                                           class_mode = 'categorical')

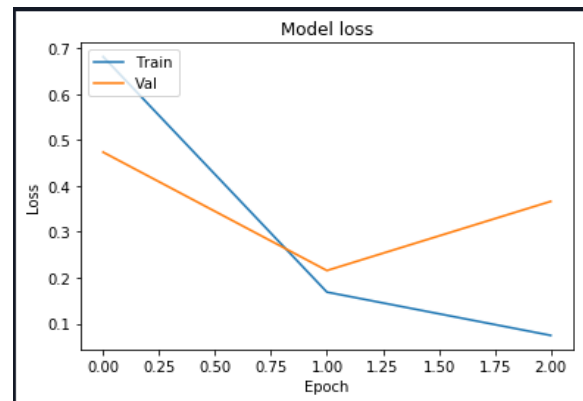
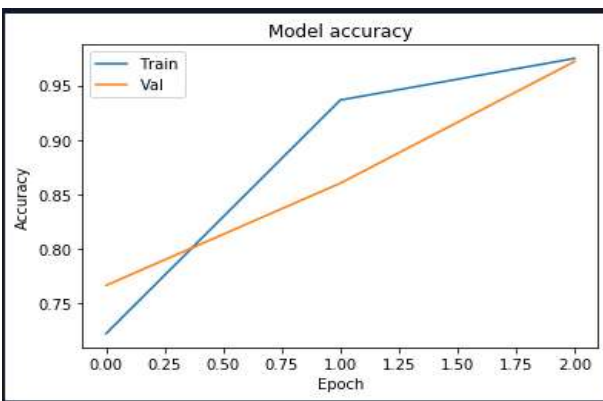
print (training_set.class_indices)
print (test_set.class_indices)

history=classifier.fit_generator(training_set,
                                steps_per_epoch = 100, # number of training images
                                epochs = 1,
                                validation_data = test_set,
                                validation_steps = 110) #number of test images
```

5.4.3 Accuracy Calculation

```
import matplotlib.pyplot as plt
# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()

# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()
```



5.4.4 Predictions

```
import cv2
import matplotlib.pyplot as plt
from keras.models import load_model
model = load_model('healthy.h5')
import numpy as np
from keras.preprocessing import image
img3 = cv2.imread(r'C:/Users/G3IN/Desktop/project/image_dataset_3/test_set/Healthy/GH_HL Leaf 283.JPG')
img3 = cv2.cvtColor(img3, cv2.COLOR_BGR2RGB)
img3 = cv2.resize(img3, (64,64))
img4 = np.reshape(img3, [1,64,64,3])
plt.imshow(img3)
plt.show()
disease = model.predict_classes(img4)
print(disease)
prediction = disease[0]
print ("class is",prediction)
if prediction==0:
    print("Healthy leaf")
elif prediction==1:
    print("bacterial_spot")
elif prediction==2 :
    print("early_blight")
elif prediction==3 :
    print("late_blight")
else :
    print("leaf_mold")
```

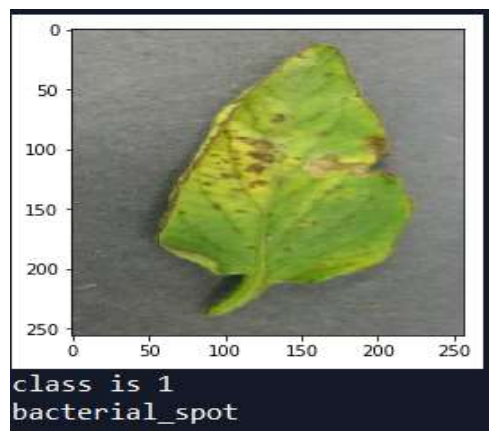
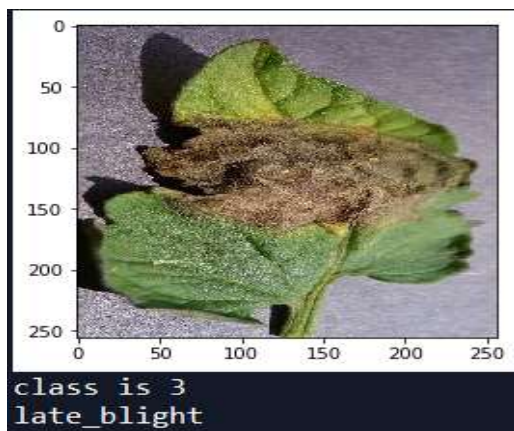
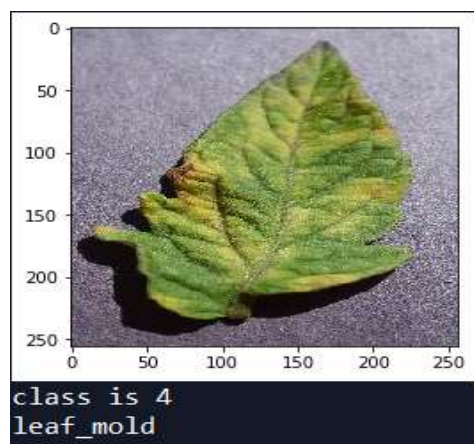
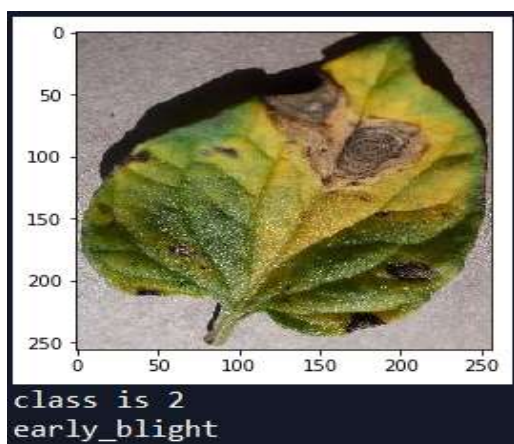



FIG 5.7: Prediction Using CNN

TEST CASES / TESTING

6.1 k-NN Algorithm

- 1) When input image is healthy, the program predicts it correctly.

```
print("Predicting custom image")
labels = ["BACTERIAL_SPOT", "EARLY_BLIGHT", "HEALTHY", "LATE_BLIGHT", "LEAF_MOLD"]
img = cv2.imread(r"C:/Users/admin/Desktop/h1.JPG")
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_pred = cv2.resize(img, (150, 150))
img_pred = np.reshape(img_pred, (1, img_pred.shape[0]*img_pred.shape[1]))
e=knn.predict(img_pred)
print(labels[(e[0])])
```

Predicting custom image
HEALTHY

- 2) When input image is diseased and disease is late blight, the program predicts it correctly.

```
print("Predicting custom image")
labels = ["BACTERIAL_SPOT", "EARLY_BLIGHT", "HEALTHY", "LATE_BLIGHT", "LEAF_MOLD"]
img = cv2.imread(r"C:/Users/admin/Desktop/lb1.JPG")
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_pred = cv2.resize(img, (150, 150))
img_pred = np.reshape(img_pred, (1, img_pred.shape[0]*img_pred.shape[1]))
e=knn.predict(img_pred)
print(labels[(e[0])])
```

Predicting custom image
LATE_BLIGHT

- 3) When input image is diseased and disease is early blight, the program predicts it correctly

```
print("Predicting custom image")
labels = ["BACTERIAL_SPOT", "EARLY_BLIGHT", "HEALTHY", "LATE_BLIGHT", "LEAF_MOLD"]
img = cv2.imread(r"C:/Users/admin/Desktop/eb1.JPG")
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_pred = cv2.resize(img, (150, 150))
img_pred = np.reshape(img_pred, (1, img_pred.shape[0]*img_pred.shape[1]))
e=knn.predict(img_pred)
print(labels[(e[0])])
```

Predicting custom image
EARLY_BLIGHT

- 4) When input image is diseased and disease is bacterial spot, the program predicts it correctly

```
print("Predicting custom image")
labels = ["BACTERIAL_SPOT", "EARLY_BLIGHT", "HEALTHY", "LATE_BLIGHT", "LEAF_MOLD"]
img = cv2.imread(r"C:/Users/admin/Desktop/bs1.JPG")
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_pred = cv2.resize(img, (150, 150))
img_pred = np.reshape(img_pred, (1, img_pred.shape[0]*img_pred.shape[1]))
e=knn.predict(img_pred)
print(labels[(e[0])])
```

Predicting custom image
BACTERIAL_SPOT

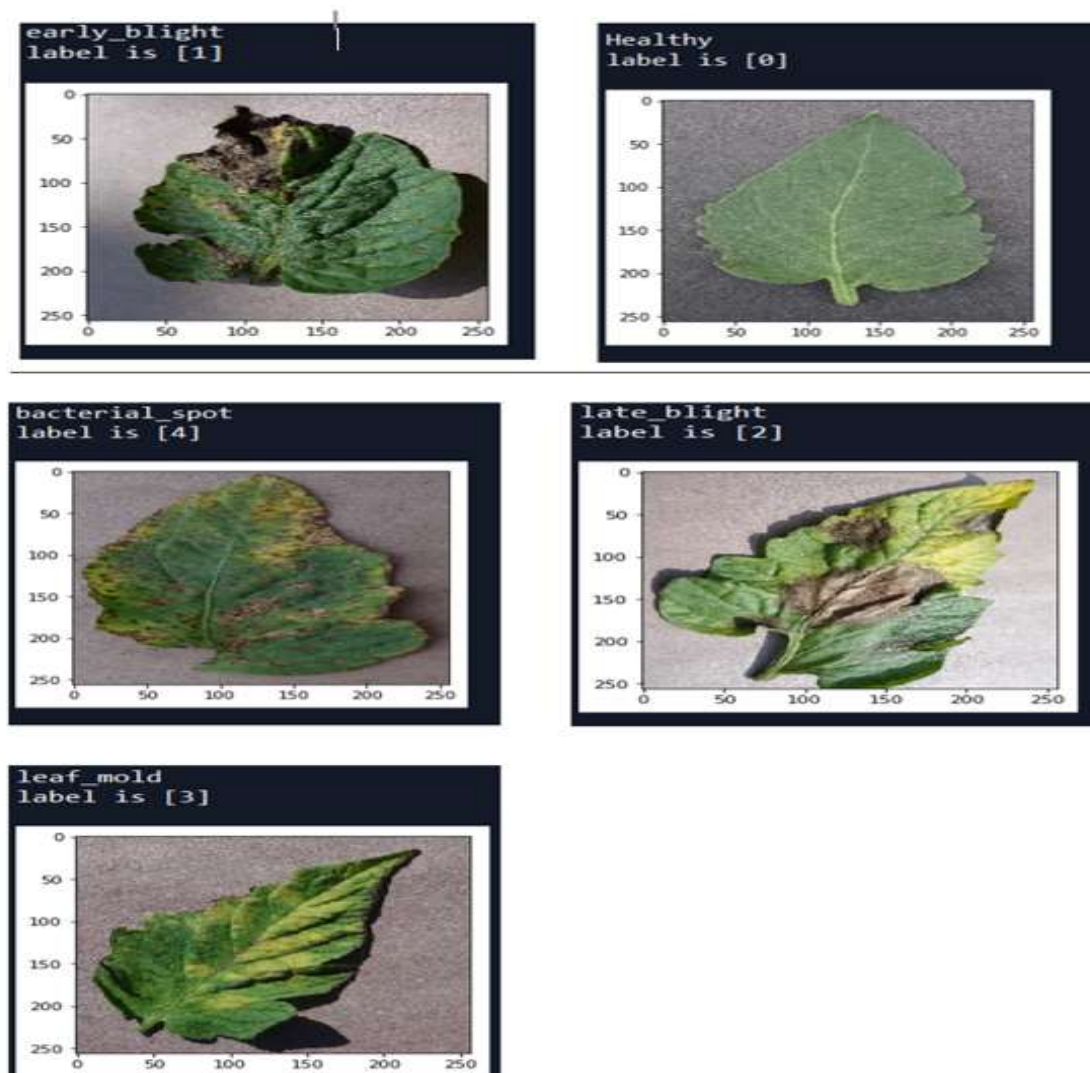
- 5) When input image is diseased and disease is leaf mold, the program predicts incorrect.

```
print("Predicting custom image")
labels = ["BACTERIAL_SPOT", "EARLY_BLIGHT", "HEALTHY", "LATE_BLIGHT", "LEAF_MOLD"]
img = cv2.imread(r"C:/Users/admin/Desktop/lm1.JPG")
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_pred = cv2.resize(img, (150, 150))
img_pred = np.reshape(img_pred, (1, img_pred.shape[0]*img_pred.shape[1]))
e=knn.predict(img_pred)
print(labels[e[0]])
```

Predicting custom image
BACTERIAL_SPOT

The accuracy of kNN algorithm is 80%. The first four predictions are correct whereas the 5th prediction is faulty considering 20% loss of accuracy.

6.2 Decision Tree Classifier



Decision tree is giving an apt prediction for given input images with 90% accuracy.

6.3 SVM Algorithm

```
print("Predicting custom image")
labels = ["BACTERIAL_SPOT", "EARLY_BLIGHT", "HEALTHY", "LATE_BLIGHT", "LEAF_MOLD"]
img = cv2.imread(r"C:/Users/aishwarya/Desktop/bs1.JPG")

img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_pred = cv2.resize(img, (150, 150))
img_pred = np.reshape(img_pred, (1, img_pred.shape[0]*img_pred.shape[1]))
e=model.predict(img_pred)
print(labels[(e[0])])
```

Predicting custom image
BACTERIAL_SPOT

```
print("Predicting custom image")
labels = ["BACTERIAL_SPOT", "EARLY_BLIGHT", "HEALTHY", "LATE_BLIGHT", "LEAF_MOLD"]
img = cv2.imread(r"C:/Users/aishwarya/Desktop/h1.JPG")
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_pred = cv2.resize(img, (150, 150))
img_pred = np.reshape(img_pred, (1, img_pred.shape[0]*img_pred.shape[1]))
e=model.predict(img_pred)
print(labels[(e[0])])
```

Predicting custom image
HEALTHY

```
print("Predicting custom image")
labels = ["BACTERIAL_SPOT", "EARLY_BLIGHT", "HEALTHY", "LATE_BLIGHT", "LEAF_MOLD"]
img = cv2.imread(r"C:/Users/aishwarya/Desktop/lb1.JPG")
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_pred = cv2.resize(img, (150, 150))
img_pred = np.reshape(img_pred, (1, img_pred.shape[0]*img_pred.shape[1]))
e=model.predict(img_pred)
print(labels[(e[0])])
```

Predicting custom image
LATE_BLIGHT

```
print("Predicting custom image")
labels = ["BACTERIAL_SPOT", "EARLY_BLIGHT", "HEALTHY", "LATE_BLIGHT", "LEAF_MOLD"]
img = cv2.imread(r"C:/Users/aishwarya/Desktop/lm1.JPG")
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_pred = cv2.resize(img, (150, 150))
img_pred = np.reshape(img_pred, (1, img_pred.shape[0]*img_pred.shape[1]))
e=model.predict(img_pred)
print(labels[(e[0])])
```

Predicting custom image
LEAF_MOLD

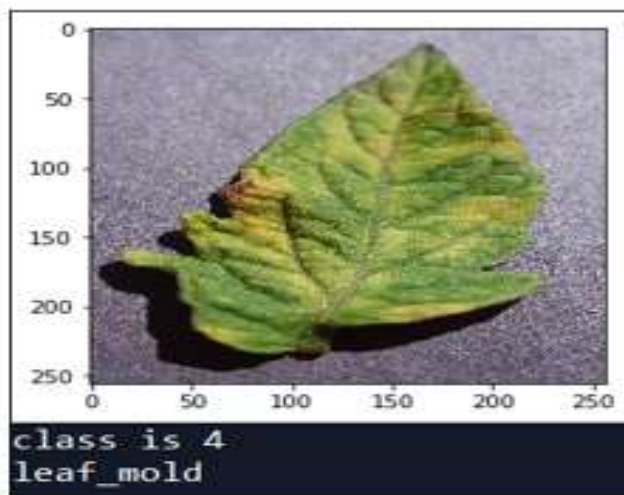
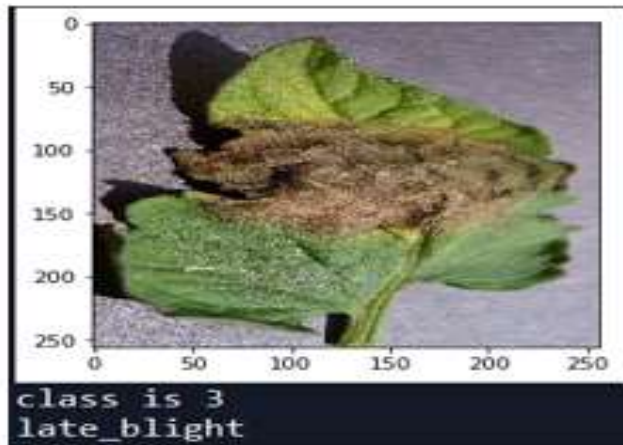
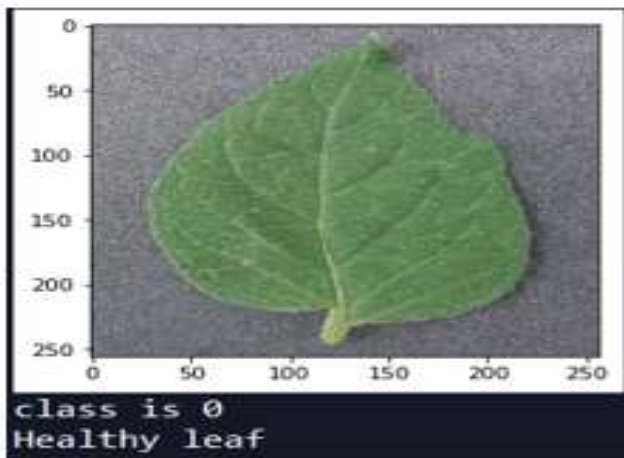
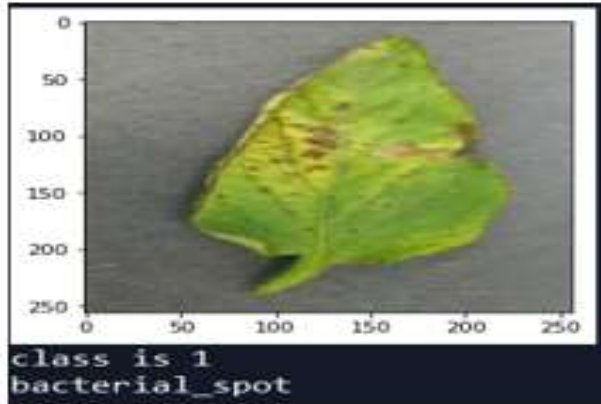
```
print("Predicting custom image")
labels = ["BACTERIAL_SPOT", "EARLY_BLIGHT", "HEALTHY", "LATE_BLIGHT", "LEAF_MOLD"]
img = cv2.imread(r"C:/Users/aishwarya/Desktop/el1.JPG")
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_pred = cv2.resize(img, (150, 150))
img_pred = np.reshape(img_pred, (1, img_pred.shape[0]*img_pred.shape[1]))
e=model.predict(img_pred)
print(labels[(e[0])])
```

Predicting custom image
EARLY_BLIGHT

The SVM algorithm is 97%. For the given respective image, it is predicting as Healthy, late blight, leaf mold correctly. Given an unknown image the SVM algorithm is predicting it as early blight where as it is healthy image this due the 3% loss of SVM algorithm.

6.4 CNN Algorithm

The CNN algorithm gives 97-98% accuracy. For the given input images, the algorithm predicts healthy and also the diseased leaf correctly.



CONCLUSION

The detection and classification of diseases in the plant leaves are important for the successful cultivation of plants and this can be done with the image processing. This project discusses various techniques for disease detection using K Nearest Neighbor, Decision Tree, Support vector machine, Convolutional Neural Network algorithms for leaves classification. For the particular diseases, the precaution measures to be fed has been given here to make plant healthy from the unhealthy stage, where the healthy plant leads to higher productivity and leads to increase in their profit.

From the implemented codes in this project it can be noticed that a machine trained with CNN and SVM classifier can give accuracy up to 97%. An efficient and user-friendly system is implemented for the classification and detection of tomato leaf diseases.

REFERENCES

- [1] Indriani, Kusuma, Rachmawanto , “Tomatoes Classification Using K-NN Based on GLCM and HSV Color Space”, Dian Nuswantoro University Semarang, Indonesia-2017
- [2] Shivali Sharma, Er. Varinderjit Kaur, Dr. Naveen Dhillon, “Plant Disease Classification with KNN-SVM Classification”, International Journal of Advance Engineering and Research Development Volume 5, Issue 05, May -2018
- [3] H. Sabrol and K. Satish, “Tomato Plant Disease Classification in Digital Images using Classification Tree”, Published: 06 April 2016
- [4] H. Sabrol and S. Kumar, “Intensity based feature extraction for tomato plant disease recognition by classification using decision tree”, Published: 9, September 2
- [5] Sagar Vetall, R.S. Khule- “Tomato Plant Disease Detection using Image Processing”, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 6, Issue 6, June 2017.
- [6] Usama Mokhtar*, Mona A. S. Alit, Aboul Ella Hassenian, Hesham Hefny- “Tomato leaves diseases detection approach based on support vector machines”, 2015 IEEE.
- [7] Jia Shijie Electronic information college Dalian Jiaotong University Dalian, Chinajsj@djtu.edu.cnJia Peiyi School of Mechatronics Information Engineering Shandong University Weihai, Chinapp_luck@163.comHu Sipping Electronic & information college Dalian Jiaotong University Dalian, Chinajsj@djtu.edu.cnLiuHaiboElectronic&information college Dalian Jiaotong University Dalian, Chinajsj@djtu.edu.cn “Automatic Detection of Tomato Diseases and Pests Based on Leaf Images”978-1-5386-3524-7/17/\$31.00 ©2017 IEEE.
- [8] Prajwala TM, Alla Pranathi, Kandiraju Sai Ashritha, Nagaratna B. Chittaragi*, Shashidhar G. Koolagudi Department of Computer Science and Engineering, National Institute of Technology Karnataka, Surathkal Email: {prajwala.tm, pakasarka, ashrita1615, nbchittaragi}@gmail.com, koolagudi@nitk.ac.in. “Tomato Leaf Disease Detection using Convolutional Neural Networks” 978-1-5386-6835-1/18/\$31.00 ©2018 IEEE.