3.3 Applying Image Processing Techniques and Artificial Neural Network to Detect Plant Diseases.

The present work recommends a system for identifying plant diseases early and accurately, using image processing techniques and artificial neural network. Agriculturalists suffer great obstacles in changing from one disease control policy to another. Depend On absolute naked eye to detect diseases can be costly, a range of plant diseases cause a significant risk to the farming sector by reducing the life of the plants.[27] the present work is targeted to develop a simple disease recognition system for plant diseases.

3.3.1 Proposed Methodology

The block diagram of the proposed system is shown in below Fig. 14. The step by step proposed approach consists of leaf image database collection, it is started with capturing the images, but in our project, we will use a Dataset obtained from "Mendeley Data" website [28], since we do not have these materials, in this work we will be focusing on five different diseases for different vegetables, then segmented the affected Area using automatic thresholding value after converted image to L*a*b* Color Space. Now we have the interest area texture to extract the features using GLCM method and make some statistical calculations After All, the feature values are fed as input to the SVM classifier to classify the given image.

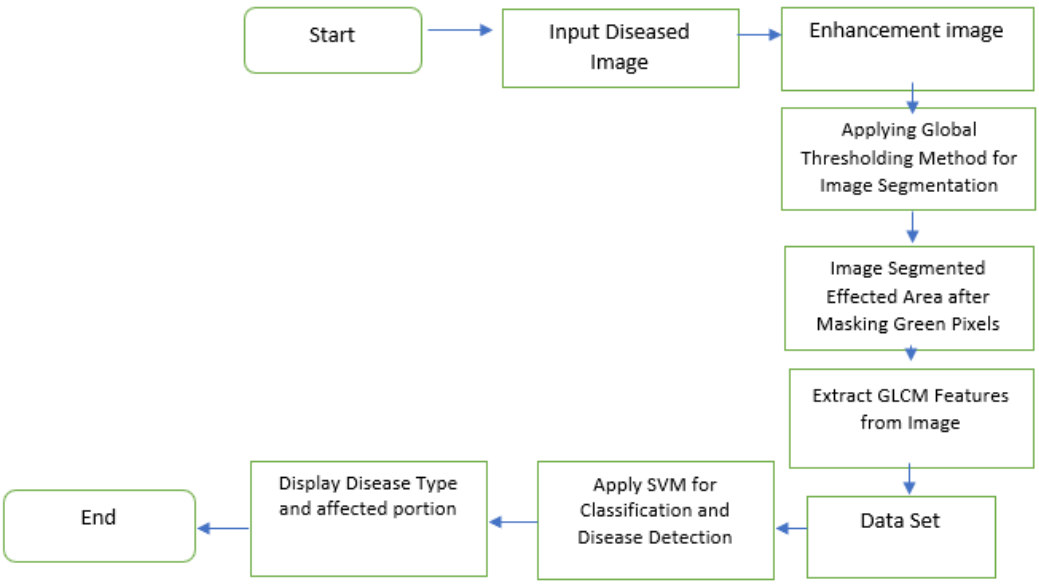

Figure 14- Framework of the proposed system

3.3.1.1 Image Acquisition

Image acquisition is the first step in the proposed method of digital image processing, and it is described as capturing the image through a digital camera and stores it in digital media for further MATLAB operations [29] . In this system, the available images from the digital camera or internet are also valid. Total 7028 image samples are captured of five classes shown in Fig.15, Pepper bell Bacterial spot, Potato Early blight, Potato Late blight, Tomato Bacterial spot, and Tomato Late blight diseases. Late blight and Early blight diseases are commonly found on tomato and potato plants [30], so, selecting the same disease for two different plants making some challenging for classification.

Plant Diseases	No. of Images used for Testing	No. of Images used for Training
Pepper bell Bacterial spot	970	22
Potato Early blight	970	30
Potato Late blight	970	30
Tomato Bacterial spot	2100	27
Tomato Late blight	1870	39
Total	6880	148

Figure 15- Plant diseases dataset

3.3.1.2 Image Preprocessing

Preprocessing stage is to enhance the image by removing noise and remove unwanted distortions, it does not add data, it positively enhances the image features that are important for further processing. The stored images are resized to a standard size(255x255).

Image enhancement is carried out for enhancing the contrast. The histogram equalization which distributes the intensities of the images is applied to the image to enhance the plant disease images. Fig.16. the contrast of the image is low in this case before applying Histogram Equalization. Moreover, as expected, the brighter pixel intensities were compressed which means it will be suitable for darkening.

Figure 16- Contrast Enhancement using Histogram Equalization.

3.3.1.3 Image Segmentation

Image segmentation is one of the essential approaches of digital image processing that used to simplify and change the representation of an image into something that is more meaningful and simpler to analyze, and to distinguish the object of interest from the background as shown in Fig.17, first it is good practice to select a color space before choosing the segmentation method since we act with color images. We select CIE L*a*b*, or CIELAB, The L* which represents a perceptual lightness, maximum value is 100 which represents white, and zero is the minimum value, which represents black. The a* and b* have no exact limits. Positive a* is red. Negative a* is green. Positive b* is yellow. Negative b* is blue. The Lab color space allows you to quantify the differences between points plotted in the color space correspond to visible differences between the colors plotted [31] . This means that the values give you an independent value representing that color. from this point, we can choose the threshold values for each color by visual inspection of the image based on your area of interest, in this technique, instead of using trial and error, we will use a graphical interface that using a slider to see the result immediately from MATLAB, called Color Thresholder in the image processing toolbox.

We will use Lab color space for each channel histogram to notify the peaks and valleys that represent each color in the a*, and b* channels histogram to allocate the cluster in the image, and masking the green pixels , by shifting the scroll of a* channel to zero to make sure we are masking all green pixels as shown in Fig.18.This is done in sense that the green colored pixels regularly represent the healthy parts of the leaf , and they do not add any valuable information to disease classification. Furthermore, this significantly reduces the processing time.

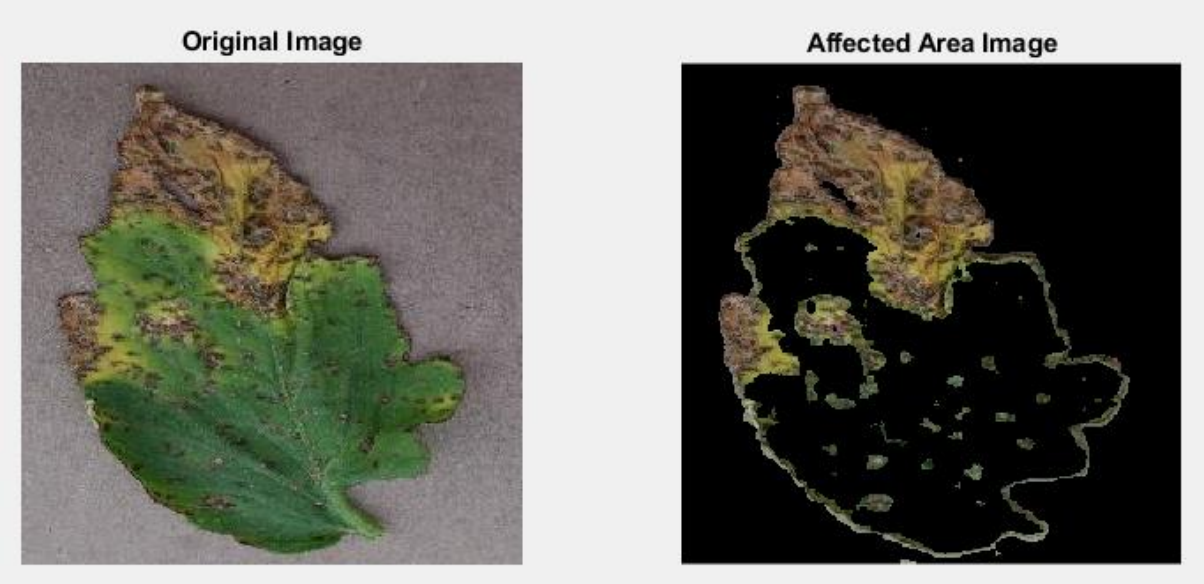

Figure 17 -Segmented affected Area for Tomato Bacterial Disease

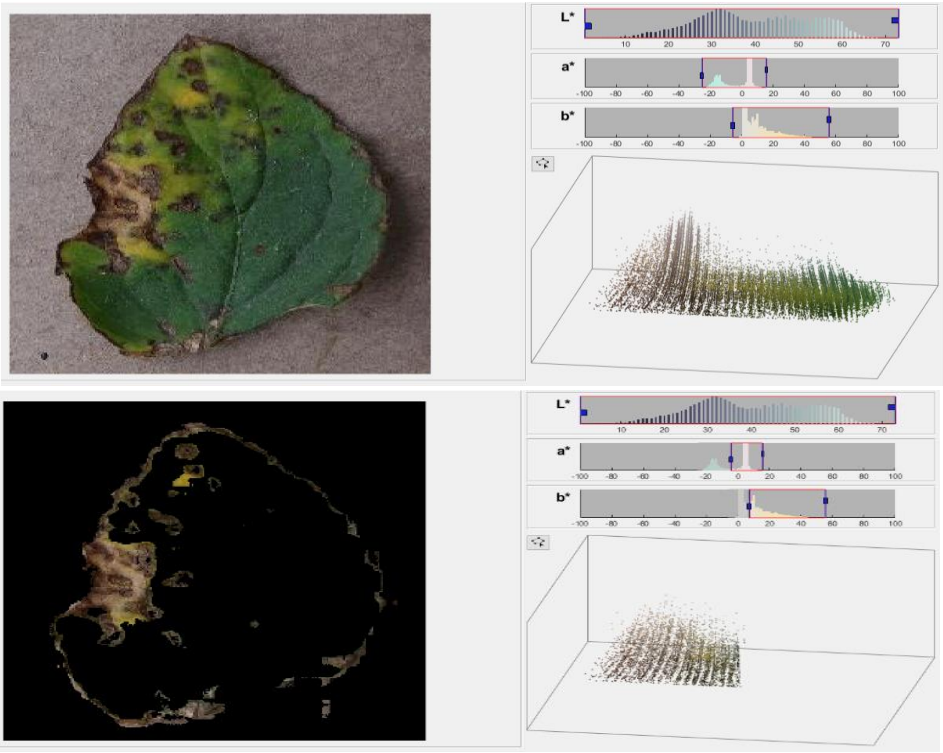

Figure 18 -Color Thresholder Application

3.3.1.4 Feature Extraction

Feature extraction is a process involving number of properties required to describe a large set of data accurately. Features are quantitative measures of texture that describe salient features in the image, there are two categories of features in the spatial domain; first-order statistical features and second-order statistical features, the first order depends only on pixel values and do not take the distribution of pixels into consideration, while the second-order statistical depends on both pixel values and the spatial inter-relationship. Statistical texture features are

be given by gray level co-occurrence matrix (GLCM) [32], it is a common texture-based feature extraction method. The GLCM determines the textural correlation between pixels in distance and angle as show in Fig.19. The GLCM properties of an image are expressed as a matrix with the same number of rows and columns as the gray values in the image. The components of this matrix depend on the frequency of the two specific pixels. Both pixel pairs can vary depending on their neighborhood distribution. These matrix elements' values depending on the gray value of the rows and columns, the matrix would be larger If the intensity values increasing. This creates a time-consuming process load, the GLCM features used in this project are contrast, correlation, energy, homogeneity, mean, standard deviation, entropy, RMS, variance, smoothness, skewness, and inverse difference movement. The procedure in this work includes first, convert each segmented image to Grayscale image so we can calculate the co-occurrence matrix by "graycomatrix" MATLAB function, and extract the statistical texture features, and make a vector of features for every sample image for each class until the last one, now we must combine these vector feature in one big dataset and use it for future training.

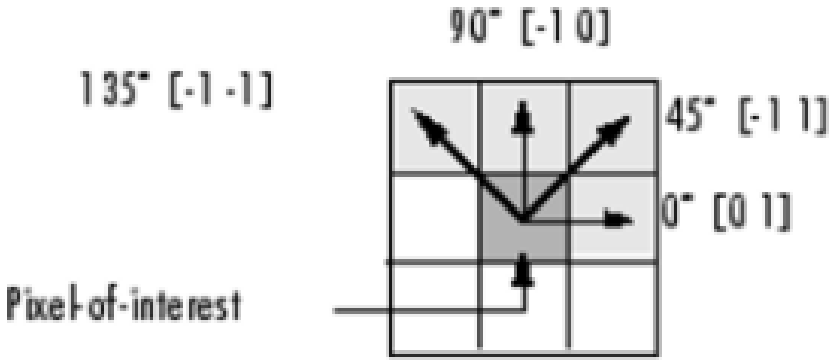

Figure 19 -GLCM depend on both pixel values and the spatial inter-relationship.

3.3.1.5 Training & Classification

the classification is performed through using Support Vector Machine (SVM). A support vector machine creates a hyper-plane or multi hyper-planes in all dimensional space, which can be utilized for classification, regression, and other different errands. SVM is supervised machine learning with related learning algorithms that examine the information and recognize patterns, used for classification analysis. Given a set of training data, and marked each sample belonging to one of the categories, Multiclass SVM aims to allocate labels to classes by applying SVM. The dominant approach to distinguish classes well is to convert the single multiclass problem into multiple two-class classification problems, this may also support the separation of classes, disease classification may face some challenges to recognize since it can mix up with other classes, this kind of problems can be eased by construct binary classifiers which distinguish between one of the labels and the rest (one vs all) shown in Fig.20(a), or among every pair of classes (one vs one) Fig.20(b). [33]

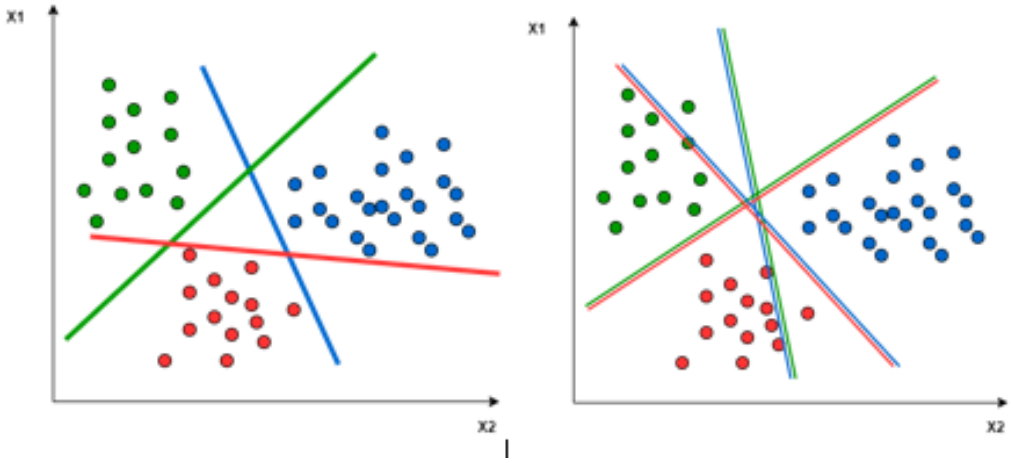

Figure 20 -(a) One-to-all approach (b) One-to-one approach [34].

Once extracting texture features and save it in (.mat) file, the file contains a (6880x13) matrix representing 6880 samples from all the five classes. Each row had 13 columns representing the 13 texture features for each sample image. Labeling the classes Done separately and the training dataset have been arranged, each class given a unique number (one, two, three, four, or five) which represented the class (*i.e.,* Tomato Late blight). "1" represented Pepper bell Bacterial spot disease, "2" represented Potato Early blight disease, "3" represented Potato Late blight disease, "4" represented Tomato Bacterial spot disease, "5" represented Tomato Late blight disease as shown if the Fig.21, then, a software program was created in MATLAB that would

train the Classifier using the “Train_Feat.mat”, and then use the test texture feature vector that represent the image that would be predicted to perform the classification task as shown in Fig. 22.

label	Plant Diseases	No. of Samples
1	Pepper bell Bacterial spots	970
2	Potato Early blight	970
3	Potato Late blight	970
4	Tomato Bacterial spots	2100
5	Tomato Late blight	1870
	Total	6880

Figure 21 -labeling the arranged training dataset.

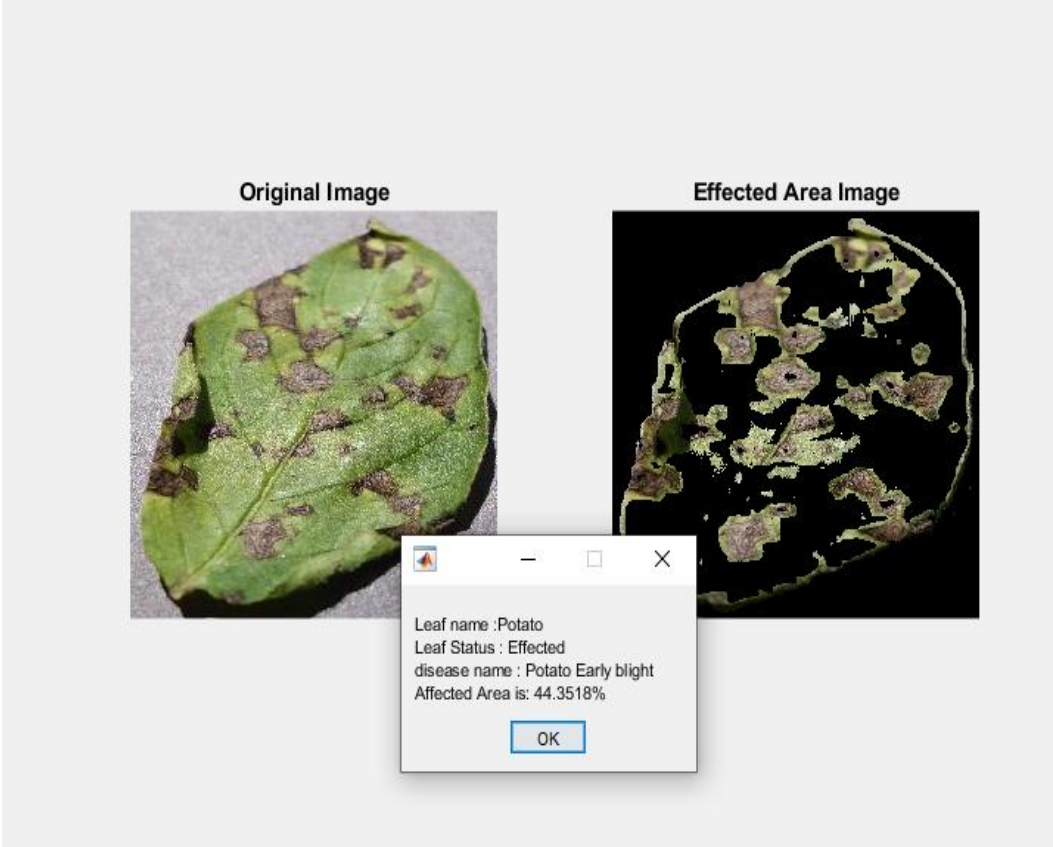

Figure 22 -prediction classification message box

that takes user input from boxes and uploads it to the S3 bucket, after confirming that the main interface displays correct banners, saves the user input correctly and uploads it when a button is pressed, we next create secondary windows, these will be displaying the current status of the greenhouse and the image recognition part, for reading the values of the greenhouse when the read values button is pressed the output values will be downloaded from the S3 bucket and displayed to the user in a new window, finally in the main window there is a "disease detection" button, when pressed this button will download the greenhouse snapshot and a MATLAB code is called from python and the image will be segmented and the leaves of the plants inside the greenhouse will be tested for disease and the result will be displayed to the user.

4.1.5 Disease Detection Result and Testing

The classification performance of the SVM was compared to other machine learning classifiers to determine if SVM is the best technique. Classification Learner App in MATLAB was used to train the plant diseases with the same features used in the classifier.

To do the same training and testing in Classification Learner App. All the 22 machine learning algorithms in the Classification Learner App were selected and the 10-fold cross-validation (C.V) with one-to-one approach was used to set the training and testing data for the model. The 10-fold C.V technique was used to set 10% data for testing and 90% data for training. The 10-fold C.V also means that the training and testing process is repeated ten times. The results of the classification using the 22 machine learning algorithms are shown in Table 2. that the highest classification accuracy is 82.4%, which was produced by Cubic SVM, followed by Quadratic SVM with 81.7% and the third in rank is the Fine Gaussian SVM with 75.2%.

Table 2 -22 machine learning algorithms using 10-fold cross-validation.

Classifier	Classifier type	Accuracy (%)	Prediction speed (Objects/Seconds)	Training time (Seconds)
Decision Trees	Fine Tree	67.7	290000	1.5564017
	Medium Tree	63.5	270000	1.377095
	Coarse Tree	53.7	290000	1.2342071
Discriminant Analysis	Linear Discriminant	68.0	240000	0.6222891
	Quadratic Discriminant (Using Diagonal Covariance)	64.8	180000	1.0536454
SVM	Linear SVM	72.8	110000	19.262819
	Quadratic SVM	81.7	410000	190.18886
	Cubic SVM	82.4	46000	804.12537
	Fine Gaussian SVM	75.2	6500	14.577755
	Medium Gaussian SVM	73.3	8100	11.1521127
	Coarse Gaussian SVM	66.2	5500	12.299001
KNN	Fine KNN	68.2	12000	3.4496906
	Medium KNN	71.2	11000	3.3252121
	Coarse KNN	68.7	6500	4.7121804
	Cosine KNN	71.0	11000	4.1591818
	Cubic KNN	71.6	540	60.0427419
	Weighted KNN	72.2	11000	5.4975716
Ensemble classifiers	Boosted Trees	66.3	42000	23.9368547
	Bagged Trees	73.4	200000	17.4178874
	Subspace Discriminant	64.7	18000	9.3814331
	Subspace KNN	57.0	6900	12.862472
	RUS Boosted Trees	65.2	39000	25.2798088

In Cubic SVM classification, 776 images of Pepper bell Bacterial spots disease are correctly classified, and misclassified samples are 193. The correct classification rate for Pepper bell Bacterial spots disease is 80.1%. For Potato Early blight disease, 853 samples are correctly classified, and misclassified samples are 117. The correct classification rate for Potato Early blight disease is 87.9%. For Potato Late blight disease, 691 samples are correctly classified, and misclassified samples are 279. The correct classification rate for Potato Late blight disease is 71.2%. For Tomato Bacterial spots disease, 1896 samples are correctly classified out of 2100. The correct classification for Tomato Bacterial spots disease is 90.3%. For Tomato Late blight disease, 1455 samples are correctly classified out of 1870. The correct classification rate for

Tomato Late blight disease is 77.8%. The overall system classification rate for the above leaf samples is 82.4% and the error rate of the system is 17.6%. as shown in Fig.25.

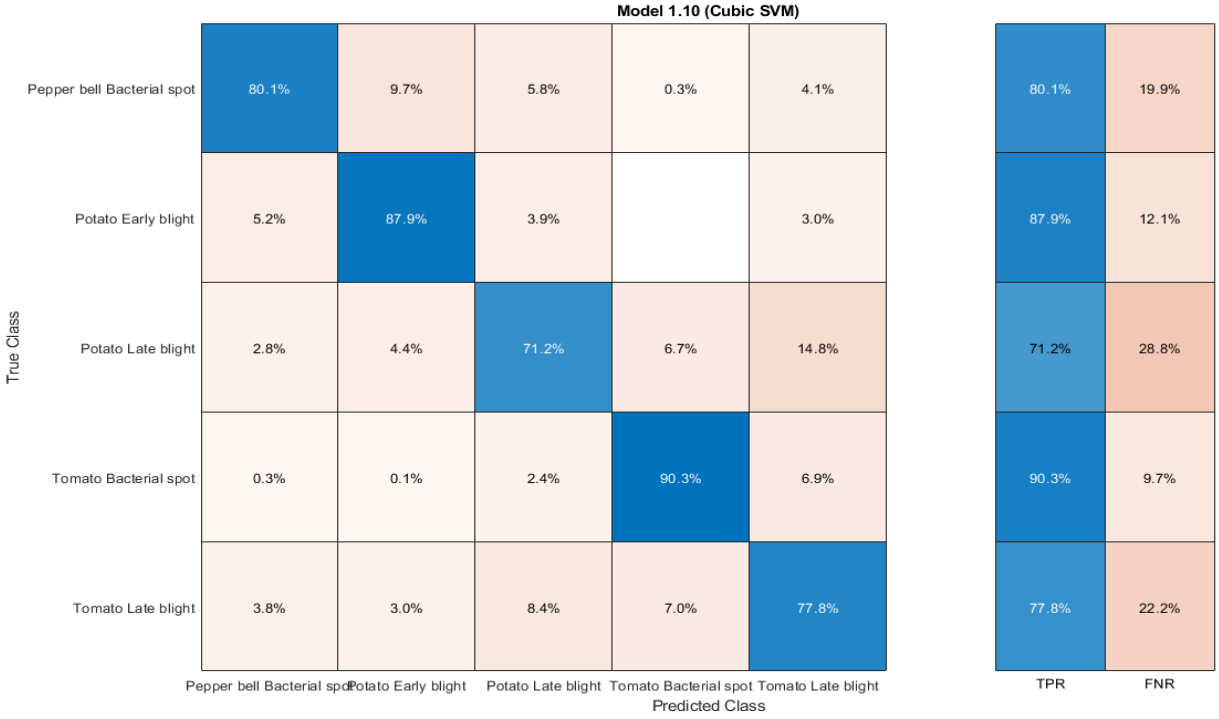

Figure 25 -Confusion Matrix for Cubic SVM Classification

4.2 Challenges

There is several problems that limit this project, for once the use of cheap hobbyist Arduino sensors while they may be good for trainees they don't provide precision and are very prone to sending corrupted data, this happens but not as frequently and for reasons beyond our control, secondly the poor processing capabilities of the RPI made it frustrating to debug problems, sometimes the OS would freeze and a reboot was necessary the use of custom made light Linux distros could solve this problem especially since Raspbian OS (RPI main OS) is filled with bloatware that could slow down the RPI, debugging the Arduino could be challenging especially since using Serial communication requires us to do so inside the RPI graphical interface and that was a very frustrating experience considering the 1 GB ram and the buggy Raspbian OS graphical interface, also a problem is when the user enters large values as threshold values while this doesn't brick the system it does create undesirable outputs, however the biggest problem facing the system is the USB serial communication where sometimes it can send or receive corrupted data that will leave the Arduino system acting randomly even

APPENDICES

Appendix A:

Libraries and Toolboxes Used:

Multiple libraries were used in the project as shown in Fig.26 these libraries provide useful functions and modules, these include:

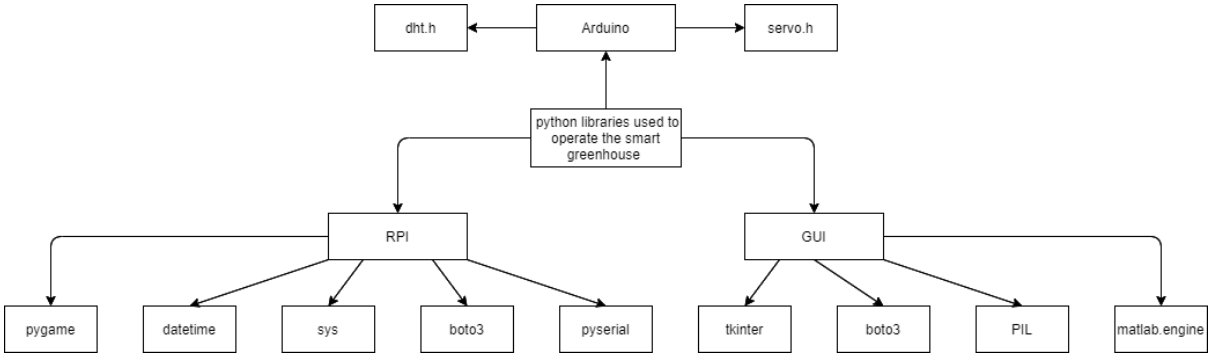

Figure 26 -libraries used in the project.

Libraries that will run on the RPI segment:

- Boto3: boto3 was used to access S3 bucket storage where the configuration and output values are stored.
- Pyserial: pyserial is used to access the serial port that is connected to the Arduino.
- Datetime: used to print the current date and time to the output file in the RPI.
- Sys: used to exit the script when all the response has been captured.
- Pygame: used to access the web cam and take a snapshot of inside the greenhouse.

Libraries used in the GUI segment:

- Tkinter: python standard GUI building tool.
- PIL: used to display images inside the GUI environment.
- Boto3: used to download output values and latest snapshot, also uploads threshold values to cloud.
- MATLAB. engine: standard MATLAB engine imported to python that will call a MATLAB script from python.

Libraries used in Arduino segment:

- Dht.h: Arduino library for the dht humidity and temperature sensor.
- Servo.h: Arduino library for the servo motors.

Toolboxes used in MATLAB segment:

- Image processing toolbox: used in segmentation and feature extraction.
- Statistics and machine learning toolbox: used in disease detection and classification.

Appendix B:

Code Segments:

scripts that will run on the RPI segment:

- Loop.sh: this bash script will run in a loop and will run the following scripts in order: download.py, tx_rx.py, photo.py, upload.py.
- Download.py: will download the threshold values from the S3 bucket.
- Tx_rx.py: will open the threshold.txt file read it and serially send its content in a start: NUM1:NUM2: NUM3:NUM4 format and reads the messages from the serial port and write it to the output.txt file.
- Photo.py: will take a snapshot of inside the greenhouse.
- Upload.py: will upload both the snapshot and output.txt to the S3 bucket.
- Secret.py: contains the secret access key and the access key for the S3 bucket.

The GUI segment:

- GUI.py: graphical user interface that will take the threshold values from the user, download output file from the S3 bucket and display its content, also will download the snapshot from the S3 bucket and display it to the user, also will run Multi_SVM.m on demand.

The disease detection segment:

- Diseases_Features.m: MATLAB script for establishing the Training Dataset.
- Multi_SVM.m: MATLAB script for disease detection affecting the leaves.
- TrainingSet : arranged Training set folder.
- TestSet : Testing set folder.
- Train_Feat.mat: Training Dataset file.
- Train_Feat.xlsx: Training Dataset file used in Classification Learner App.

Appendix C:

User Manual:

note that at least MATLAB 2018 and python 3.7 must be downloaded for the GUI script to run, also MATLAB engine must be installed on user PC, to do this open MATLAB and on the command prompt:

1) `cd (fullfile(matlabroot,'extern','engines','python'))`
2) `system ('python setup.py install')`

Also note that the directory of Multi_SVM.m must be changed inside GUI.py to point to the script location.

To run the system:

1) Connect the external power supply and USB connection to the Arduino and the other end to the RPI.
2) Run loop.sh on the RPI.
3) Open GUI on user PC.
4) Enter desired threshold values then click “submit” button.
5) To view latest status in greenhouse, click “read current data from greenhouse” button.
6) To display latest snapshot of inside the greenhouse, click “display latest snapshot” button.
7) For classification disease from image click” Diagnose” button then select the image.