LEAF DOCTOR

Abstract -
Our project "Detecting Leaf Disease" is the latest solution that is developed using MATLAB to solve the major problems in agriculture right now in case of automating disease detection in plants and crop management. The system accurately detects the diseases in plant leaves, combining the technology of the advanced image processing techniques with convolutional neural networks (CNN). It acts as a real-time due tool requisite for farmers and agricultural professionals.
The system works through an image processing pipeline that prepares raw images, identifies key features, and enhances the diagnostic process. Using algorithms for segmentation, feature extraction, and image enhancement, it performs reliably under various environmental conditions and across different plant species. CNNs, a deep learning method, allows the system to recognize complex patterns in leaf textures, colors, and structures, which are essential for identifying diseases. After training on large datasets, the model becomes capable of distinguishing even subtle differences in leaves, leading to highly accurate classification and diagnosis of plant diseases

Keywords: plant disease detection, convolutional neural networks, image processing, deep learning, agriculture, MATLAB

I. INTRODUCTION

Damage to plant diseases has drastic adverse effects to many countries' economies because of their contributions towards high agricultural production losses, including the quality and quantity of agricultural products; this also makes it very difficult for monitoring such a large farm, thus making it more tough for farmers. Symptoms perceived by farmers usually differ from one another. Indeed the vast knowledge it would take to develop these symptoms makes it impossible for farmers to recognize all of them. Typically farmers detect and name these diseases through manual inspection; an unfortunately time-consuming, labor-intensive process.

Farmers now focus on improving the quantity and quality of harvested crops as well as minimizing the amount of pesticide used. Due to the aforementioned adversity, automated systems have sprung forth that utilize these new emerging technologies in the early detection of plant diseases. The first attempts to classify diseases through plant images relied primarily on digital imaging processing.

This dataset analysis was well assimilated within several image processing methods forming a constant analysis of crops from sowing until harvesting. The other technologies can be called References fast, precise, and less expensive when compared to manual inspection methods for the detection of plant diseases. According to smart farming improvisation, image processing, machine learning (ML), and deep learning (DL) algorithms improve the understanding of plant disease from different perspectives. Image processing has been effective alone without ascertaining much accuracy in the cost-effectiveness, and basically, these will be further upended in the shape of cost-effectiveness and accuracy through the usage of ML techniques.

The Flow diagram below shows who the project works and methodology used to do this project

.

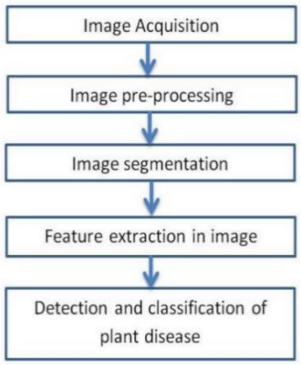

Figure.1 Flow diagram of proposed system.

The most used ML techniques include supervised learning, which relies on labeled datasets for training, and unsupervised learning, where unlabeled data is used to discover patterns without human intervention. The selection of the right algorithm and input type is crucial for achieving the desired results. In the context of plant disease detection, using image data from datasets presents a more complex challenge than traditional numerical data, but it also provides a more powerful tool for disease identification.

III PROPOSED METHODOLOGY

This paper has come up with ideas on the methodology proposed. Some of the best ways of leaf disease detection is convolutional neural network techniques. CNNs are very great in the classification of image patterns due to heuristic features such as automatic learning of hierarchical representations of features such as patterns and textures. CNN is trained on larger datasets containing diseased and healthy leaves in the process of training. Thus, the model does not require feature engineering but will learn these features by itself and, thus, will be able to adapt to changes brought about by different diseases and variances in leaves. Use of transfer learning from large pre-trained datasets such as ImageNet allows one to fine-tune the CNN model toward the solving of their problem and thus achieving high accuracy, especially under cases where there is a limited amount of labeled data.
The proposed system in the present work MH-DMAT against MATLAB is a replica of that which uses sophisticated methods/processes of image processing with the operations of CNN to recognize with very high precision plant diseases. For example, capture high-resolution images of the infected parts of leaves with special noise removal and

improvement image processing techniques and feature extraction to stress parts related to the disease. And the heart of this system is CNN trained on datasets comprising healthy and diseased leaves. This network would know how to recognize any differential minute points and features in a plant, clinically accurate classification for different environment conditions and plant species. MATLAB has streamlined this process since it provides a complete environment for developing, training, and deploying the CNN, making it effective for the real-time environment of applications in agriculture.

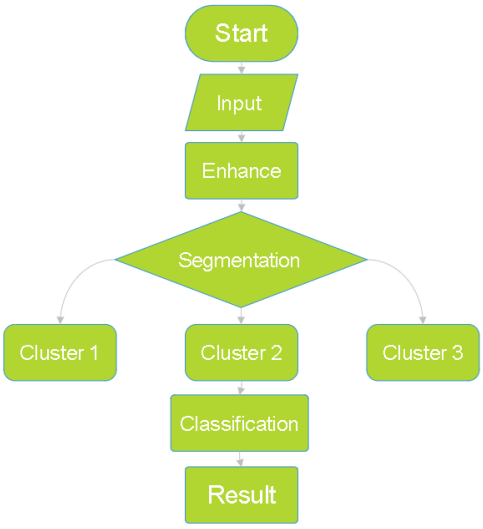

Figure.2 Methodology

The system employs RGB to HSI (Hue, Saturation, Intensity) conversion to better analyze color features, which are critical in disease detection. The RGB image is split into its red, green, and blue components and converted to HSI format using mathematical formulas. This transformation allows the system to accurately assess color differences, helping in the detection of diseases based on variations in leaf color. The HSI model is more aligned with human visual perception, making it ideal for analyzing plant health.

III.1 Feature Extraction:

In plant disease detection, feature extraction is essential, focusing on characteristics like colour, texture, morphology, and structure. A significant method is colour co-occurrence, which captures both texture and colour attributes. This process begins by converting RGB images of leaves into the HIS (Hue, Intensity, Saturation) colour space, separating colour information from intensity for a detailed analysis.

Next, colour co-occurrence matrices are generated for each channel—Hue, Saturation, and Intensity—by examining the spatial relationships between pixels. This approach identifies not only the presence of colours but also their arrangements, which can indicate specific diseases.

The resulting matrices provide vital features, allowing the extraction of statistical measures such as contrast, correlation, and energy. These metrics enhance the detection process by improving accuracy in distinguishing healthy from diseased tissues, even amidst lighting variations. By employing colour co-occurrence methods, researchers can develop sophisticated algorithms for early disease identification, ultimately fostering better agricultural practices and crop management.

$$X = 0.5\ \{(R-G) + (R-G)\}$$

$$Y = \sqrt{(R-G)^2 + (R-B)\ (G-B)}$$

$$\theta = \text{acos}\left(\frac{x}{y}\right)$$

$$H = \{\theta \quad \text{if } B<G\}$$

$$360-\theta \quad B>G$$

$$S = 1 - \frac{3}{(R+G+B)} * [\min (R,G,B)]$$

$$I = \frac{1}{3}(R+G+B)$$

Figure.3 Feature Extraction Formula

IV. WORKING OF THE SYSTEM

The system for identifying diseases in plants captures a leaf image uploaded by the user, which is processed to auto resize it into the standard dimension of 256x256 pixels for uniformity of analysis. Enhancement of critical information in the picture is then carried out using the contrast adjustment of the image through [imadjust]. The procedure serves to prepare the image for the next phases of identification of areas of interest. On preparing it, Otsu's method was utilized to segment the picture. This method basically extracts the leaf from its background and results in a binary image in which leaves appear in white, while the background is in black. It's thus easy to concentrate on the analysis of the leaf's characteristics.

The system hence enters the feature extraction stage. The user is required to indicate the region of interest (ROI) alleged to be diseased on the leaf. If the segmented image is still in RGB format, it will be converted to grayscale for easier extraction. The Gray Level Co-occurrence Matrices (GLCMs) are computed using the gray image, which allows the extraction of detail-oriented image features like contrast, correlation, energy, and homogeneity as well as other statistical parameters including mean, standard deviation, entropy, and skewness. A comprehensive description of the health condition of the leaves is then created in the 'feat disease' identifying array for future investigation.

Figure.4 Disease Detection Method

For classification, the system employs machine learning techniques. It loads pre-existing training data, which includes both features and corresponding labels, from a file. The system then uses a multi-class Support Vector Machine (SVM) classifier to predict the type of disease present based on the extracted features. Once the classification is complete, the system displays the results, informing the user of the identified disease affecting the leaf.
Finally, the detected disease type is shown in a message dialog box, offering the user an easy-to-understand conclusion about the health of the plant. This entire process, from image preprocessing to machine learning classification, works seamlessly to provide an efficient solution for detecting plant diseases with a high degree of accuracy.

V. ALGORITHMS USED

A.K-MEANS ALGORITHM

The K-means algorithm is a renowned algorithm for clustering under unsupervised machine learning. It is aimed at partitioning the dataset into K separate clusters, representing each of the clusters using a centroid. One of the most important objectives is to bring together similar data points. Because of this, it has a great significance in many applications, including image segmentation, where pixels are grouped together using feature similarities of pixels. First, the user decides on a number of clusters, K, and then the K cluster centroids are initialized randomly. After that, every data point will be assigned to one of the nearest centroids based on a certain distance metric, generally Euclidean distance. After that, whenever all points are put according to assigned centroids, they will be recalculated by taking the average positions of the data points in each cluster. Keep iterating between the two assignment steps until either the centroids don't shift or the maximum number of iterations set beforehand is reached.

In the case of image segmentation, K-means clustering generates K classes or clusters by contextualizing the features for each pixel, while minimizing the squared distance between pixels and respective cluster center to obtain distinct classes for every image pixel. Essentially, the technique minders the segmentation of pixels in an image in such a way that the foreground object, such as a leaf, is differentiated from the background. The enhancement in the segmentation is often done by Otsu's thresholding method. This algorithm of Otsu converts a grayscale image into a binary form, with pixels less than a particular threshold marked black (0) and all above it being marked white (1).

B. MultiSVM

The multisvm function defined in the code implements a multi-class Support Vector Machine (SVM) classifier in the one-vs-all way which is one of the methods for dealing with multi-class classifications issues. It works by converting the multi-class problem into a number of binary classification tasks. For each class label in the given dataset, it trains a binary SVM classifier which is supposed to distinguish between samples of that category and samples from all other categories. Again, this is done iteratively as the SVM model considers one class to be the positive instance and all others as negative instances. That way, it develops many binary classifiers for every specific category label. After binary classifiers have been trained for every class, one-vs-all strategy is applied. This means, for each sample test, the model evaluates outputs from all binary classifiers and assigns the label with the maximum confidence score. Essentially, it means that maximum output among binary classifiers will be the predicted label for the sample test, transforming multiclass classification into successive binary decisions.

The function then iterates over each test sample, utilizing the trained SVM classifiers to predict the class label. For each sample, the class with the highest confidence score is chosen as the final classification result. This approach is particularly effective for datasets with multiple classes, providing robust classification performance by reducing the complexity into simpler binary tasks. In the end, the function outputs the predicted class labels for the test samples, effectively determining which class each sample belongs to based on the highest output score from the SVM classifiers.

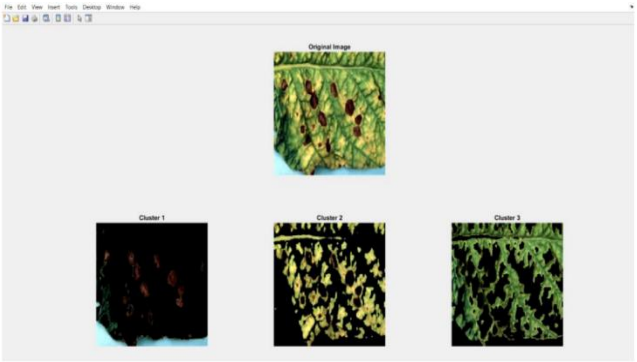

Figure.5 Image Segmentation Diagram

C. Support Vector Machines (SVM)

The Support Vector Machines (SVM) is an extremely capable and flexible algorithm in the machine learning arena targeting its application at both classification and the regression levels. More specifically, the application of SVM models is applicable in class-based issues whereby data is associated with particular class names. The model does this by simply identifying the hyperplane as the best separation between the different classes in the feature space. The optimum hyperplane, therefore, is expected to maximize the margin criterion with the classes, thus ensuring that the classification boundary is as wide as possible for better generalization on unseen data.

The separating hyperplane would act as a decision boundary in binary classification that separates the set of data points belonging to one class from those that belong to the other. This is what would define the kernel trick, and that is what makes SVM different: it can learn non-linearly separable data rather than just changing input features into higher dimensions. SVMs employ specific kernel functions, such as the polynomial kernel or the RBF kernel, which transform the input space into high defined dimensional feature space. In

such transformed space, SVM learns the linear decision boundary, even if the original data is not linearly separable.

In training, the SVM optimizes the position of the hyperplane to maximize the margin but with the least classification errors at the margin boundaries. Now, the SVM classifier has been trained, and this can be put into action to classify, or predict, the class labels of previously unseen data points, establishing which side of the hyperplane those points fall on. In summary, SVM is a very useful tool for using an efficient basis for separating classes in classification- either data than linear or non-linear means using a kernel function. It has got usage in most real-life tasks in machine learning.

VI. Frameworks Utilized

MATLAB GUI

This MATLAB GUI offers simple, intuitive interfacing for the entire process of leaf image processing, extraction of features, and classification of diseases in a MATLAB GUI application. At the beginning of the application, the new user is greeted with a demo video that shows the critical working of the tool and guides them on how to go through the workflow. This introduction helps users get an idea of what the app does and how they can proceed, even if they do not know the tool well.
The interface allows users to upload leaf images, enhancing the image quality using contrast adjustment. Through k-means clustering, the application enables users to segment regions of interest (ROIs) within the leaf images, which are displayed for visual inspection.

KEY FEATURE

A key feature of the application is its ability to extract critical image characteristics, such as contrast, correlation, energy, and homogeneity, from the segmented regions. These extracted features are displayed within the GUI for easy analysis. Moreover, the application includes a function to evaluate the accuracy of a support vector machine (SVM) model trained on leaf image features. Cross-validation techniques are employed to assess the model's accuracy, and the results are shown to the user for deeper insight into the model's performance.
Users can also classify the segmented ROIs using a multi-class SVM model, which determines whether the leaf is healthy or diseased, and if diseased, identifies the specific condition. Additionally, relevant data such as the affected area and extracted features from the classified region are presented, helping with diagnosis.

DEMO VIDEO

The included demo video plays automatically when the GUI is launched, offering a visual guide to the application's functionality. It shows users how to upload a leaf image, enhance contrast, segment regions, extract features, and classify diseases. By following the video, users can quickly understand the workflow and make the most of the application's capabilities without needing to refer to extensive documentation.

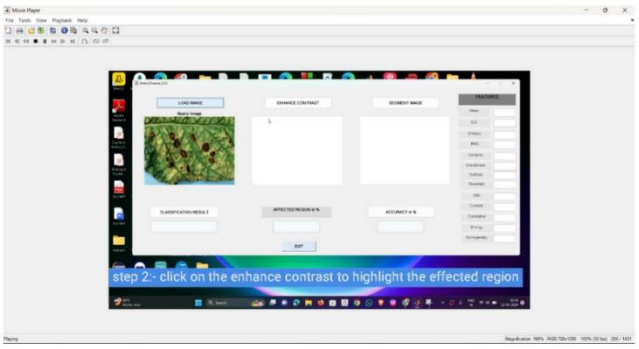

Figure.6 MATLAB GUI and DEMO VIDEO OUTPUT

VII. Result and Discussion

The implementation of the leaf disease detection system using MATLAB yielded promising results across various stages of the analysis process as mentioned below

Step 1: Upload the Leaf Image Click on the **Load Image** button to upload an image of a diseased leaf. This step initializes the process by importing the input image into the system for further analysis.

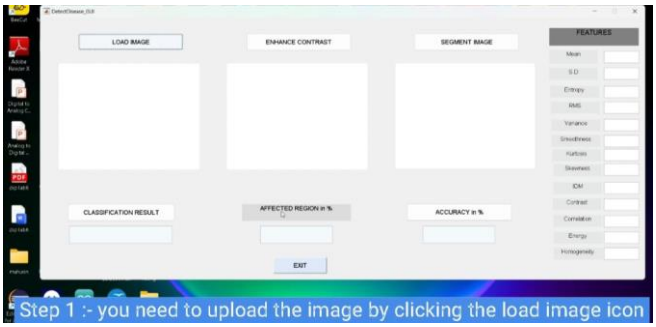

Figure.7.1 LOAD IMAGE

Step 2: Enhance the Image Click on the **Enhancement** button to apply contrast enhancement. This step enhances the visibility of the diseased areas, making them stand out more clearly. By using the imadjust function, the contrast of the image is improved, which helps highlight the affected regions, making it easier for segmentation and feature extraction.

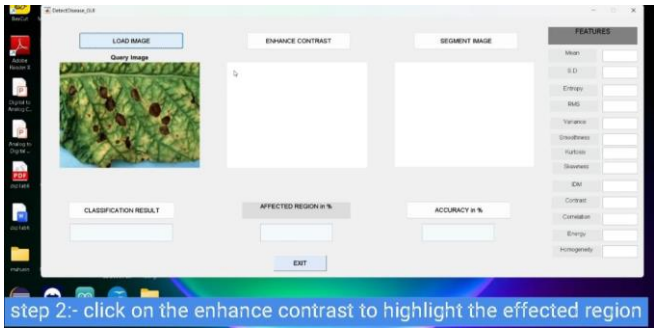

Figure.7.2 ENCHANCE CONTRAST

Step 3: Segment the Image Click on the **Segmentation** button, which applies Otsu's thresholding method to separate

the leaf area from the background. This process produces three clusters, dividing the image into distinct regions.

Figure.7.3 SEGEMENT RANGE

Step 4: Analyse the Segments After segmentation, you will observe the clusters. **See which cluster** shows the diseased area more clearly. The segmented regions (clusters) represent different parts of the leaf, and the diseased area will typically be more visible in one of the clusters.

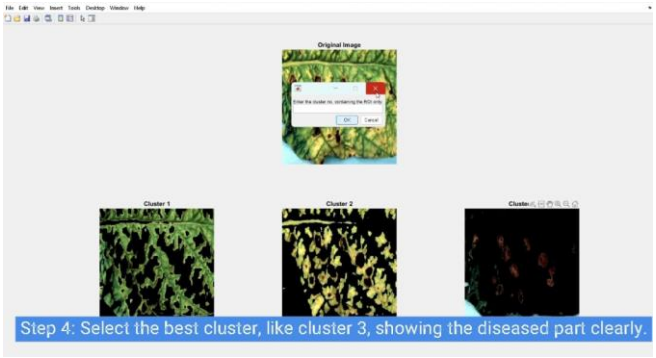

Figure.7.4 CLUSTER Selection

Step 5: Enter Cluster Number Once you identify the cluster where the disease is most visible, a dialog box will appear. **Enter the cluster number** that corresponds to the region where the disease is prominent. This step informs the system which part of the leaf to focus on for feature extraction and classification.

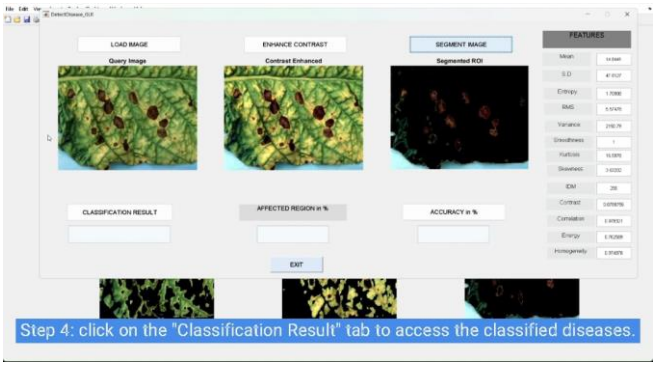

Figure.7.5 CLASSIFICATION RESULT

Step 6: Classify the Disease Click on the **Classify** or **Result** button to initiate the classification process. The system will extract texture features like contrast, correlation, energy, and homogeneity using Gray Level Co-occurrence Matrices (GLCMs). After feature extraction, the multi-class Support Vector Machine (SVM) classifier will distinguish between healthy leaves and various disease classes,

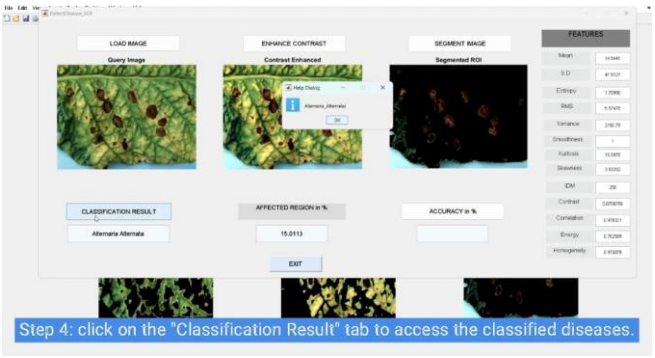

Figure.7.6 AFFECTED REGION in %

Step 7: Check Accuracy Finally, click on the **Affected Region Accuracy** button to calculate the accuracy of the classification result. Giving insight into the reliability of the classification.

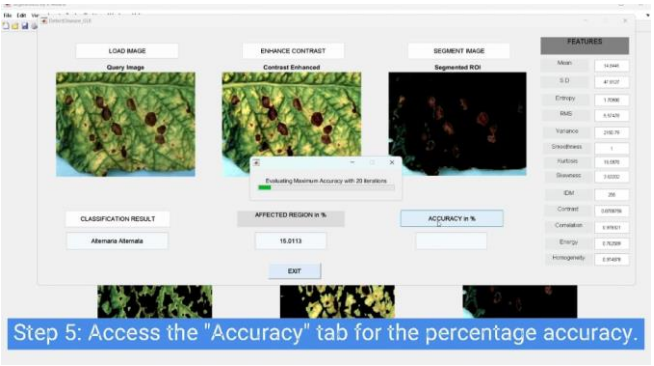

Figure.7.7 ACCURACY in %

This process allows for an efficient and automated way to detect leaf diseases using MATLAB, guiding the user through each step for seamless plant health management. The practical implications of this system include its potential use in precision agriculture, enabling farmers to identify and respond to plant diseases early, reducing crop

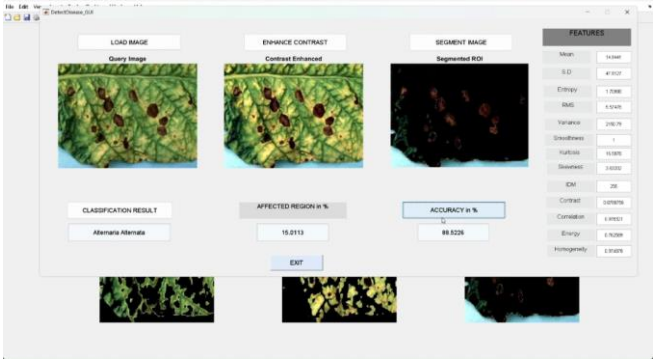

Figure.7.8 FINAL RESULT

VIII. CONCLUSION

We have built a project on automatic plant disease detection using MATLAB. The methods involved are image processing and convolutional neural networks that could improve early detection of diseases, which increases crop yields. The system provides early information of diseases; hence, it prevents or reduces the spread of diseases in addition to the reduction of pesticides use.The pipeline begins with preprocessing steps like noise reduction, colour normalization, and contrast enhancement, followed by feature extraction that identifies textures and colours. The system is designed to work across different environmental conditions and plant species, thanks to its robust preprocessing techniques and CNN architecture. Convolutional layers extract features, while pooling layers reduce data dimensions, leading to accurate classification.

With deep learning, the system enhances accuracy and reduces the time spent on manual inspections. Traditional methods are slow and prone to errors, but our automated system can quickly analyse thousands of images, providing real-time feedback to farmers. This scalable, efficient system supports sustainable agriculture, aiming to increase global food security and productivity.

IX TRAINING PIPELINE FOR AUTOMATED LEAF DISEASE DETECTION

To train the machine learning algorithms for the leaf disease detection system, the process began with comprehensive dataset preparation. High-resolution images of healthy and diseased plant leaves were gathered from existing datasets or captured in the field. These images underwent preprocessing steps, including noise reduction, contrast enhancement, and RGB to HSI transformation, to enhance colour feature analysis. Segmentation techniques, such as K-means clustering, were employed to isolate diseased regions by grouping similar pixels, and the region of interest (ROI) was identified for extracting disease-relevant features.

Feature extraction focused on identifying texture, colour, and morphological traits. Using Gray Level Co-occurrence Matrices (GLCM), statistical features like contrast, correlation, energy, and homogeneity were computed. The HSI color space further facilitated detailed colour analysis by separating hue, saturation, and intensity. For classification, a multi-class Support Vector Machine (SVM) was trained using the one-vs-all strategy. Binary SVMs were trained to differentiate each class from all others, with extracted features as input. Hyperparameters, such as kernel types, were optimized to achieve maximum accuracy.

For deep learning, pre-trained Convolutional Neural Networks (CNNs), such as ImageNet, were fine-tuned to adapt to the nuances of leaf diseases. The model's layers were adjusted to recognize unique textures and structures, and data augmentation techniques like rotation, scaling, and flipping were used to increase dataset size and improve model robustness. The system's performance was validated by splitting the dataset into training and testing subsets and evaluating metrics like accuracy, precision, and recall. Cross-validation ensured consistency in the models.

Finally, the trained models were integrated into a MATLAB-based GUI, enabling real-time classification. The GUI allowed users to upload images, process inputs, extract features, and classify diseases seamlessly. This comprehensive pipeline ensured the algorithms were robust, accurate, and efficient in automating the detection and classification of leaf diseases.

X PROFICIENCY IN MATLAB SIMULATION FOR LEAF DISEASE DETECTION

I have strong experience in using MATLAB as a simulation tool, particularly for developing solutions to real-world problems. In my project on leaf disease detection, I used MATLAB extensively to design a complete workflow that combines image processing, machine learning, and deep learning techniques. I handled tasks like preprocessing high-resolution images through noise reduction, contrast enhancement, and RGB to HSI transformation, and I used K-means clustering to segment images and isolate regions of interest for analysis.

I leveraged MATLAB's feature extraction tools, such as Gray Level Co-occurrence Matrices (GLCM), to calculate statistical measures like contrast, correlation, and energy, which were crucial for identifying diseased leaves. I also trained and optimized a multi-class Support Vector Machine (SVM) using MATLAB's machine learning toolbox and implemented pre-trained Convolutional Neural Networks (CNNs) for deep learning. I fine-tuned these models and applied data augmentation techniques to improve accuracy and robustness.

Additionally, I developed an interactive MATLAB GUI to streamline the process, enabling users to upload leaf images, analyse them, and view results in real time. This project not only enhanced my technical skills but also demonstrated my ability to integrate simulation outputs into practical applications, making me confident in using MATLAB to address complex technical challenges effectively.