Machine Learning and IoT-based Plant Disease Prediction

[1]Mrs. Mamatha H. S
Assistant Professor
Department of Mathematics
University BDT College of Engineering
Davangere, Karnataka, 577004, India

[2]Mrs. Kavitha G
Assistant Professor
Department of Computer Science and Engineering
University BDT College of Engineering
Davangere, Karnataka, 577004, India

Abstract— Agriculture is the most important sector; every human on earth depends on it. The quality and quantity of food production may be reduced by the diseases affecting the plants. The plant disease has to be identified as early as possible to stop spreading to other plants and treat the affected plant for better yield. The manual inspection of plant diseases is highly impractical due to several factors like the requirement for highly skilled pathologists, time consumption, and high cost. But these issues can be solved in this research by using advanced technologies like Machine Learning (ML) and Internet of Things (IoT). A large plant disease database is taken, which contains 38 classes. It includes more than 9 types of plants with healthy and various disease-affected images. The identification of the correct type of disease is highly challenging because of its similar features. To extract the features highly correlated to the disease, Swin Transformer (ST) is used, and the most important features from this are reduced by two methods: Linear Discriminant Analysis (LDA) and Principal Component Analysis (PCA). The reduced features are given to the ML algorithms, such as Support Vector Machine (SVM), Random Forest (RF), and XGBoost for classification. The six combinations of models, such as ST-LDA-SVM, ST-LDA-RF, ST-LDA-XGBoost, ST-PCA-SVM, ST-PCA-RF, and ST-PCA-XGBoost, are evaluated using standard metrics. The ST-PCA-XGBoost gives excellent performance of 98.16% Accuracy, 98.19% Recall, 98.21% Precision, and 98.20% F1-Score in classifying 38 categories from the database. The best-performing algorithm is stored in the Firebase cloud. A user-friendly mobile application has been developed for plant disease detection. In the designed mobile app, the user can simply upload the plant images and identify the plant's health status. By using IoT technology, the images are sent to the cloud, and the model in the cloud is used to detect the plant disease and return the result to the mobile. This helps the farmers to detect the plant disease accurately without spending much time or money.

Keywords— PlantVillage Dataset, Mobile Application, Machine Learning, Confusion Matrix, ShuffleNet, Feature Reduction, Accuracy, Google Colab.

I. INTRODUCTION

Plant diseases have a significant impact on crop quality and yield. Early detection and prevention of these diseases are critical for crop protection and increased production. Agriculture accounts for around 17% of India's GDP [1]. Important crops, including pepper, potatoes, and tomatoes, rank among the top in India. A variety of factors contribute to the development and transmission of diseases in agricultural regions, including environmental conditions and cross-contamination. A wide range of crops exhibiting strong productivity in agricultural sectors can be studied. Pest infestations reduce crop productivity by 30–40% per year [2]. Virus, fungi, and bacteria

cause infectious diseases in crops. Because of the numerous diseases and the diverse factors, agricultural professionals often struggle to effectively control infections. As a result, this situation directly affects the overall quantity and quality of agricultural production. Instead of relying on modern specialized devices, farmers often manually and visually inspect crops for symptoms of disease. Traditional techniques of visual inspection, which rely solely on a professional's knowledge, present several obstacles and constraints in the agricultural field [3]. In the worst case, an untreated plant disease can lead to complete crop failure and significantly reduce productivity. Certain agricultural diseases may not show evident signs, making it difficult to determine the best course of treatment. In such instances, identifying the correct diagnosis, nature, and method of intervention can be challenging. Therefore, advanced and in-depth research is required.

To solve the problems mentioned above, computer-aided automated systems such as IoT, and ML can enable accurate, fast, and early disease diagnosis [4]. The advantages of these technologies stem from their capacity to generate quick and consistent results using computerized recognition and image processing approaches. ML and IoT technologies in agriculture can help minimize labor costs, time inefficiencies, and improve crop yields and quality. Implementing suitable management strategies enables more effective disease control programs by using the most recent data on crop health and disease locations.

Some notable research works on plant disease detection are analyzed to identify the research gap. The research [5] utilized K-means clustering (KMC) and SVM to identify turmeric leaf disease. KMC was applied to extract the damaged area from turmeric leaf images. Color, texture, and shape were used to visually differentiate various diseases. An SVM classifier was used to assess the type of disease affecting turmeric leaves. Performance measurement criteria were utilized to evaluate the effectiveness of the experimental technique on turmeric images. According to the experimental outcome, the proposed strategy outperformed the existing backpropagation method. The study [6] explored the use of ML-based texture analysis techniques to predict the severity of chickpea disease. Several techniques were employed during model development, including augmentation, annotation, scaling, and color conversion. Texture features were retrieved. Bayesian optimization was applied to avoid local minima. Multi-class algorithms for classification were created using image classification approaches, including Convolutional Neural Networks (CNN), SVM, and K-Nearest Neighbors (KNN). The proposed texture feature based-KNN network was outstanding in predicting the severity of chickpea disease. This study [7] compared KNN, SVM, and CNN models. Three unique models were presented and evaluated for identifying eight major leaf diseases. When trained with images from the soybean leaf disease dataset, the CNN model outperformed the KNN and SVM models, which achieved accuracies of 64% and 76%, respectively. The study aims to [8] was to develop a classification framework to predict the probability of downy mildew on melon leaves by applying integrated features, along with comparative classification methods such as LGBM, RF, and XGBoost. The fused features included texture, Canny edge, color, and Shannon entropy. These features were fed into a classification algorithm to estimate the probability of downy mildew in melon crop. The model was evaluated using the confusion matrix, and the RF algorithm achieved the highest accuracy. The work [9] focused on classifying mango leaf diseases by analyzing leaf images using the XGBoost algorithm. Additionally, it investigated an XGBoost-based classification approach that used Histogram of Oriented Gradients (HoG) for extracting features. The study successfully diagnosed

mango leaf problems by analyzing key features in leaf images, demonstrating the efficiency of the XGBoost model combined with HoG. A high level of classification accuracy was achieved.

Some researchers have proven that hybrid models perform better on image classification tasks than single models. The paper [10] proposed automated leaf disease classification using ML. The approach has three stages: processing, feature extraction, and categorization. The leaf image was separated into green, blue, and red bands, and noise was eliminated by a median filter. Key components of the images were then extracted. To improve SVM performance, its parameters were optimally tuned by the Sunflower Optimization technique. The affected region was isolated using a level-set segmentation technique. The method's effectiveness was evaluated for achieving the highest accuracy for plant disease prediction. The proposed plant disease detection models in research [11] were developed using ML techniques such as Naive Bayes, RF, Linear Regression, CNN, KNN, and SVM. Unbiased metrics were used to evaluate the performance of the generated plant disease risk assessment model. Results showed that the ensemble plant disease model outperformed the other proposed detection approaches. In study [12], RF and CNN algorithms were integrated to propose a new approach for diagnosing bacterial disease. The model extracted features from input images using three convolution and pooling, and a fully connected layer before classifying the images using RF into bacterial disease categories. Experimental outcome showed that the CNN-RF method classified images with high precision compared to other existing methods. In article [13], DL techniques were investigated, and a convolutional ensemble model was designed to increase the model's capacity to recognize the plant lesions. The ensemble combined three MobileNet variants into a new model, which distinguished between different types of plant diseases. Experimental outcome validated the effectiveness of the proposed model and confirmed that it outperformed other advanced methods. Study [14] introduced a hybrid RF–Multiclass SVM strategy for foliar disease identification in plants. To enhance classification accuracy, images were processed and segmented by Spatial Fuzzy C-Means before classification. The PlantVillage dataset, consisting of more than 50,000 healthy and affected leaf images, was used. Evaluation metrics were assessed to find the system's effectiveness. The proposed strategy was compared with recent models to assess its superiority.

After obtaining better results, they moved to deploy the model on a common platform so that anyone can access it and benefit from it. In experiment [15], CNN, ResNet-50, VGG-16, and 19 networks were analyzed for crop disease detection using a 10,000-image PlantVillage dataset. The experimental outcome highlights that the ResNet-50 performs better than others. Consequently, the ResNet-50 network was chosen for deployment as a web application for plant disease identification in the real world. This online tool assists farmers by analyzing plant leaf images for early disease identification. The automated disease diagnosis system described in research [16] detects diseases rapidly using advanced image processing, Raspberry Pi, and KMC. Upon disease detection, the system activates an alert and sends SMS notifications containing disease details, causes, and solutions, delivered through a voice assistant in the local language. Simultaneously, the disease information is shown on a display and emailed to the user. This approach ensures effective communication, supports timely decision-making, and promotes sustainable agriculture.

From the literature survey, it is clear that individual models perform well, but hybrid models work excellently for image classification. The hybrid model takes advantage of each model's strengths and gives better results. The focus is now on making these solutions usable in real life. Many

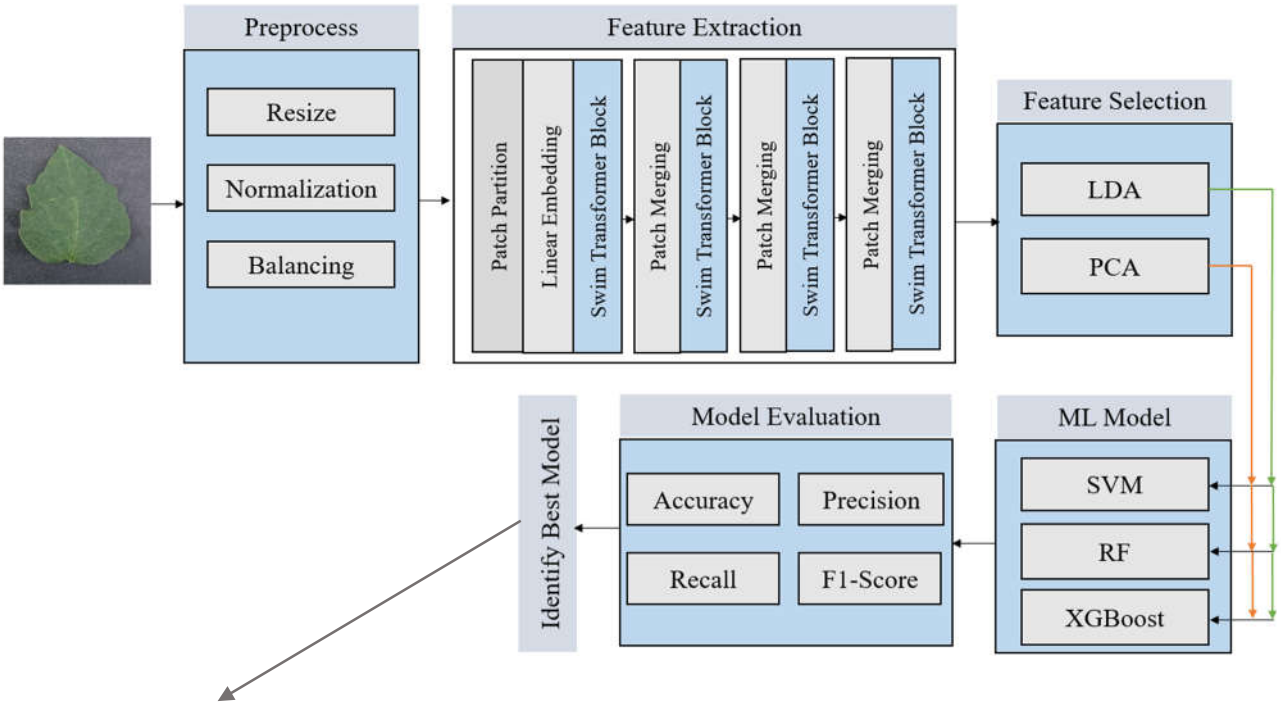
researchers focus on a limited number of crop types, concentrating on a particular crop and its diseases, which is not enough.

The research tries to build a reliable model for detecting diseases in multiple crops. The research proposes an effective feature extraction technique using the ST network, and the most important features are identified by PCA and LDA techniques. For classification, SVM, RF, and XGBoost are employed. Various combinations of feature selection and ML models are evaluated to identify the best one for multi-class crop disease prediction. After finding the best model, the research uses IoT to develop a user-friendly mobile app for detecting plant diseases anywhere in the world without requiring knowledge of technologies.

Structure of the research paper: Section 1: Importance of plant disease detection and research gap identified by the literature survey. Section 2: Research methodology from data collection to model design. Section 3: Identifying the best model through experiments and deployment, and the working of the mobile app. Section 4: Conclusion, limitations, and future work.

II. MATERIALS AND METHODS

The complete research on ML and IoT-based plant disease detection is illustrated in Figure 1. First, the images are downloaded from the PlantVillage dataset and processed. Features are extracted from the ST network, and further, the features are selected using PCA and LDA methods. The selected features are given to the ML models: SVM, RF, and XGBoost. The ST-PCA and ST-LDA features, along with the three ML models, are evaluated to identify the best one. The best model is deployed in the Firebase cloud. Parallelly, the mobile app is designed in such a way that the user can upload leaf images. Once the user uploads the images, they are sent to the model in Firebase, and the plant health status is detected. The result from Firebase is then transferred to the mobile app to display the result. The transfer of images and output is achieved through IoT.



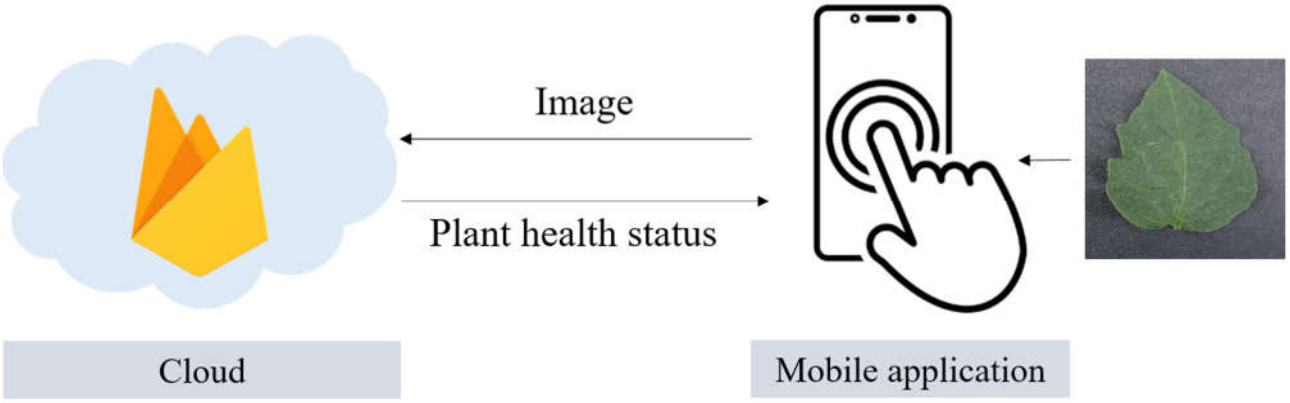


Fig. 1. Proposed Methodology

A. Data Collection and Processing

The PlantVillage dataset [17] is an open, accessible dataset composed of 54,305 images of healthy and diseased leaves taken under perfect conditions. The images depict 14 different plant species, such as blueberry, tomato, pepper, grape, squash, cherry, potato, strawberry, raspberry, soy, peach, orange, and apple. It includes images of 17 basic, 2 mold (oomycete), 2 viral, 4 bacterial, and 1 mite disease. Additionally, 12 plants exhibit healthy leaves with no signs of disease.

The collected images are first resized to 224×224. This helps to make the image ready to be fed into ST for feature extraction. Next, the target class images are highly imbalanced. A total of 500 images from each class are taken—if a class has more than 500 images, it will be reduced by randomly eliminating images; if the category has fewer than 500 images, it will be increased using augmentation methods. The images are finally normalized to reduce complexity in training the model. The images are then split in the ratio of 7:2:1 for training, validation, and testing.

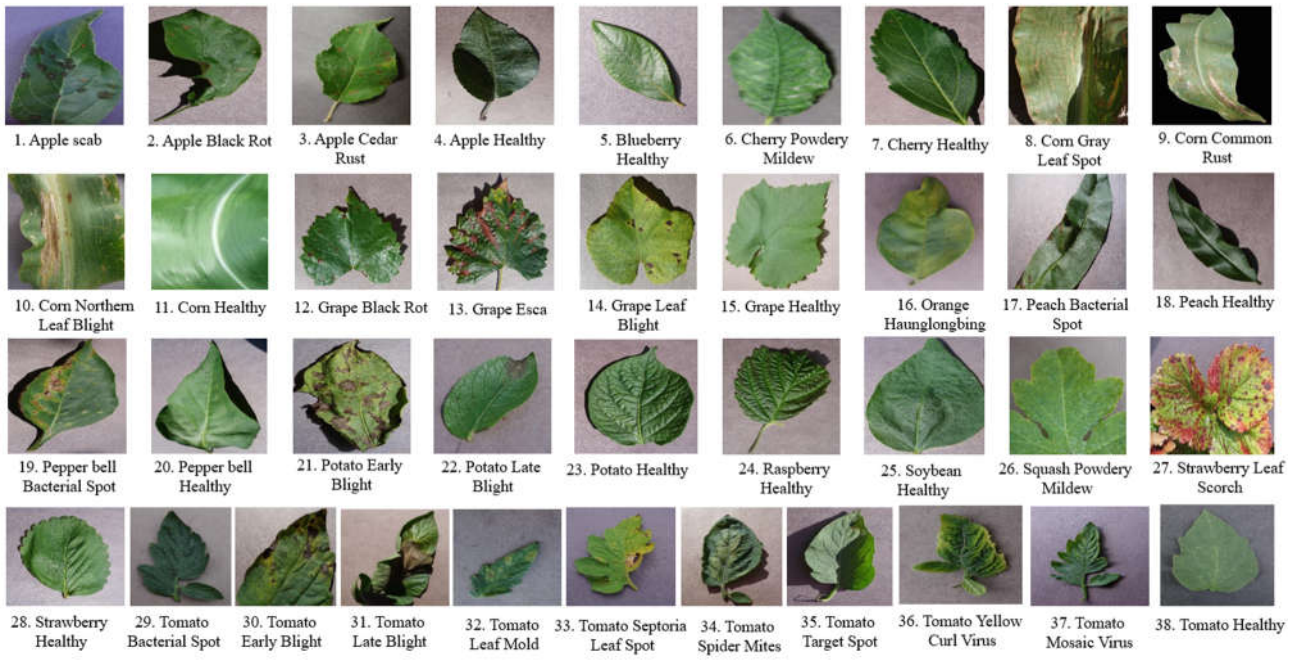


Fig. 2. Sample images from the PlantVillage Dataset

B. Feature Extraction

The ST is used as the backbone for the plant disease detection research. The model is chosen because of its lightweight and reliable output. The ST is a visual model built on the Transformer architecture that enhances its applicability and generalizability by introducing concepts such as hierarchical structure, blockwise self-attention, and multiscale fusion [18]. Figure 2 illustrates the architecture of the ST network. Its distinguishing feature is the use of a hierarchical Transformer structure, which creates feature maps at multiple scales by combining image blocks. Furthermore, it employs a shifted window approach to limit the computing scope of self-attention, thereby promoting cross-window connectivity. This hierarchical structure enables the ST to adapt to various scale visual tasks, while the shifted window technique maintains linear computational complexity proportional to image size [19]. In the initial stage of this technique, a $224 \times 224 \times 3$ image is input into the patch partition and splitted into multiple non-overlapping patches of size 4×4, each with dimensions $4 \times 4 \times 3$. In Stage 1, the partitioned patches pass through a linear embedding layer that changes the feature dimension to C before being delivered to an ST block. The ST block extracts image features using self-attention computations before passing them to the next level. Patch merging is used in Stages 2–4 to connect adjacent windows by combining feature maps from the previous stage into 2×2 sizes. These combined feature maps are then passed through the ST block, resulting in hierarchical feature representations. During this process, the patch merging layer performs downsampling and dimensionality augmentation, while the ST block extracts features from images.

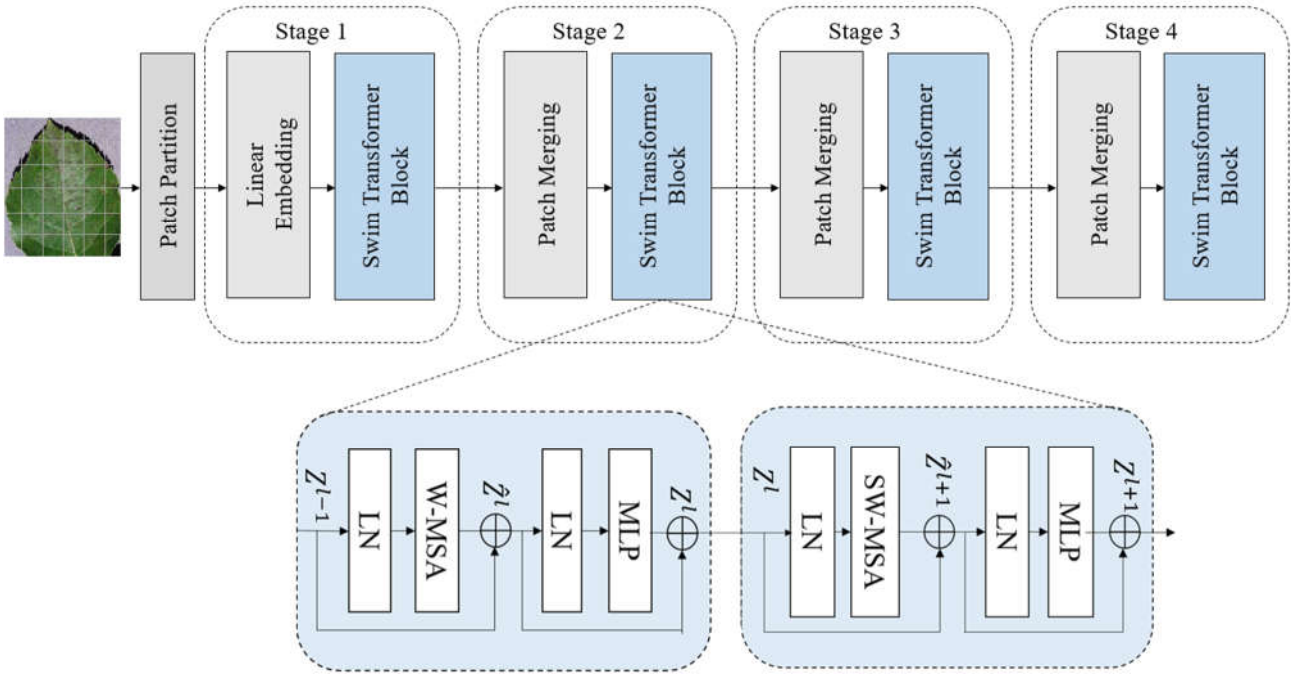


Fig. 3. Swin-Transformer Architecture

The ST block module consists of window-based multi-head self-attention (W-MSA), a multilayer perceptron (MLP), shifted window-based multi-head self-attention (SW-MSA), normalization, a residual connectivity layer, and the ST block itself [20]. The SW-MSA is the central component of the ST block, evaluating self-attention by alternating between conventional and shifted windows at specific stages. This approach allows the model to effectively incorporate global information while remaining computationally efficient. In the self-attention computation, the model calculates similarity using relative positional encoding and restricts attention computation to each

window, reducing processing cost. The formula for calculating self-attention using relative positional encoding is given in Equation (1):

$$Attention(Q,K,V) = Softmax\left(\frac{Q*K^T}{\sqrt{d_k}} + B\right)V \quad (1)$$

Where B represents the relative position code $B \in R^{M^2*M^2}$. M^2 refers to the number of patches in the window. $Q,K,V \in R^{M^2*d}$, where d represents the matrix's dimension. The scaling factor $\sqrt{d_k}$ is used to reduce the impact of dot product fluctuation, where k represents the number of queries. The formula for the entire block is given in Equation (2-5):

$$\hat{Z}^l = W - MSA\left(LN(Z^{l-1})\right) + Z^{l-1} \quad (2)$$

$$Z^l = MLP\left(LN\left(\hat{Z}^l\right)\right) + \hat{Z}^l \quad (3)$$

$$\hat{Z}^{l+1} = SW - MSA\left(LN(Z^l)\right) + Z^l \quad (4)$$

$$Z^{l+1} = MLP\left(LN\left(\hat{Z}^{l+1}\right)\right) + \hat{Z}^{l+1} \quad (5)$$

where $\hat{Z}^l$ represents the features of the W-MSA output, and Z^l represents the features of the MLP output.

C. *Feature Selection*

The extracted features from the ST model are of high dimensionality, with a dimension of 768, which is quite large. For reducing the dimensionality and selecting the most vital features, the PCA and LDA methods are applied. The working of both algorithms is discussed in this subsection.

Linear Discriminant Analysis (LDA): LDA is a popularly employed approach for reducing feature dimension [21]. It is particularly useful for identifying linear combinations of features that successfully distinguish two or more classes within a dataset. Essentially, LDA maximizes separation between classes while decreasing variance within each. According to [22], the basic premise in determining the discriminant component features, denoted by W, is to solve the generalized Rayleigh quotient, and the computation is given in Equation (6):

$$W = arg\ max_w \frac{W^T S_B W}{W^T S_W W} \quad (6)$$

For a four-class classification, the scatter matrices between and among classes are constructed using the mathematical formula shown in Equations (7-8):

$$S_B = \frac{1}{4}\sum_{i=0}^{3}(\mu - \mu_i)(\mu - \mu_i)^T \quad (7)$$

$$S_W = \sum_{i=0}^{4}\sum_{j=0}^{n}\left(x_{i,j} - \overline{x_{i,j}}\right)\left(x_{i,j} - \overline{x_{i,j}}\right)^T \quad (8)$$

In this equation, μ represents the overall mean vector, μ_i represents the mean vector of category i, n represents the sample dimension, $x_{i,j}$ represents the j[th] sample in the i[th] category, and $\overline{x_{i,j}}$ represents the corresponding mean. An eigenvalue decomposition of the specified solution is performed, as outlined in Equation (9):

$$S_B W = \lambda S_W W \tag{9}$$

Where λ represents the eigenvalue. Assuming S_W is a non-singular matrix, the within-class variance matrix is transposed, simplifying Equation (10).

$$S_W^{-1} S_B W = \lambda W \tag{10}$$

The $S_W^{-1} S_B$ matrix has no more than $n-1$ non-zero eigenvalues, resulting in three unique eigenvectors that are used to project the dataset.

Principal Component Analysis (PCA): PCA employs orthogonal transformations to find the correlated variables and transform them into uncorrelated variables [23]. The fused feature vector dimension is reduced to enable effective categorization via PCA. As a result, the most discriminative features are obtained and used to construct the model. Fused feature has n dimensions $FF = x_1, x_2, \dots, x_n$, which must be reduced to $k \ll n$ dimensions. PCA uses the following processes to obtain reduced fused features:

Data Scaling given in the Equation (11)

$$x_j^i = \frac{x_j^i - \overline{x_j}}{\sigma_j} \tag{11}$$

Co-variance matrix computation is given in the Equation (12):

$$\sum = \frac{1}{m} \sum_i^m (x_i)\,(x_i)^T, \sum \in R^{n*n} \tag{12}$$

Eigenvector and Eigenvalue calculations are given in Equation (13):

$$u^T \sum = \lambda \mu U = \begin{bmatrix} | & | & | \\ u_1 & u_2 \dots & u_n \\ | & | & | \end{bmatrix}, u_i \in R^n \tag{13}$$

The first 50 eigenvalues from the N-dimensional space are selected for use in the ML model, and it is given in Equation (14):

$$x_i^{new} = \begin{bmatrix} u_1^T x^i \\ u_2^T x^i \\ \cdots \\ \cdots \\ u_k^T x^i \end{bmatrix} \in R^N \tag{14}$$

D. Classification

The selected features are given to the ML models for categorization. In this research, SVM, RF, and XGBoost are used. Each model is discussed in detail in this subsection.

Support Vector Machine (SVM): An SVM is a supervised technique that categorizes data by selecting the best line or hyperplane in an N-dimensional space that maximizes classification distance [24]. It distinguishes between categories by computing the ideal hyperplane, which maximizes the margin that separates the closest data samples from opposing groups. The total features in the input samples defines whether the hyperplane is two-dimensional or N-dimensional. Because multiple hyperplanes are utilized to differentiate groups, maximizing the margin between data samples allows

the algorithm to choose the optimal decision boundary. This improves generalization to new data and results in accurate classification predictions. When data is not linearly separable, the kernel is used [25]. SVMs operate on linearly separable data, meaning no adjustments are needed to divide the data into distinct groups. The separating hyperplane can be expressed mathematically as shown in Equation (15):

$$wx + b = 0 \tag{15}$$

Where w denotes the weight, x the input, and b the bias. For calculating the maximum distance, the hard-margin technique is used. In this technique, the data samples are completely separated from the support vectors. This is expressed by the formula in Equation (16):

$$\left(wx_j + b\right)Y_j \geq a \tag{16}$$

To maximize the margin, use the formula $max\ \gamma = a/||w||$, where a indicates the projected margin onto w.

Random Forest (RF): An RF is an combination of decision trees, each built with a subset of randomly chosen variables [26]. For a p-dimensional random vector $X = (X_1, \ldots, X_P)^T$, an unknown joint distribution $P_{XY}(X, Y)$ is assumed, where X represents the input, and Y represents the output. The objective is to identify the function $f(X)$ that correctly estimates Y. This function is calculated by reducing the expected score of a specified loss $L(Y, f(X))$ as shown in Equation (17).

$$E_{XY}\left(L\left(Y, f(X)\right)\right) \tag{17}$$

Conceptually, $L(Y, f(X))$ is a measure of how close $f(X)$ is to Y; it penalizes $f(X)$ values that are too far away. Typical possibilities of L include squared error loss. For regression, $L(Y, f(X)) = (Y - f(X))^2$; for classification, zero-one loss is used.

$$L\left(Y, f(X)\right) = I(Y \neq f(X)) = \begin{cases} 0\ if\ Y = f(X) \\ 1\ otherwise \end{cases} \tag{18}$$

Minimizing $E_{XY}(L(Y, f(X)))$ for squared error loss provides the conditional expectation.

$$f(x) = E(Y|X = x) \tag{19}$$

In categorization, if the set of potential values of Y is represented by $\mathcal{y}$, minimizing $E_{XY}(L(Y, f(X)))$ with zero-one loss:

$$f(x) = \arg max_{y \in \mathcal{y}}\, P(Y = y|X = x) \tag{20}$$

Ensembles use "base learners" $h_1(x), \ldots, h_J(x)$ to create the ensemble predictor $f(x)$. For classification, $f(x)$ is the most commonly predicted class ("voting").

$$f(x) = \arg max_{y \in \mathcal{y}} \sum_{j=1}^{J} I\left(y = h_j(x)\right) \tag{21}$$

RF's j[th] base learner is a tree named $h_j\left(X, \Theta_j\right)$, where Θ_j is a set of random variables that are independent for $j = 1, \ldots, J$.

XGBoost: XGBoost is a powerful regression and classification algorithm developed in research [27] XGBoost, which relies on the gradient boosting structure, continuously adds new decision trees that

match values with residuals over multiple iterations, thereby increasing learner effectiveness and speed [28]. Details about XGBoost are presented below. If a sample set comprises n samples and m features, it can be written as $D = \{(x_i, y_i)\}(|D| = nx_i \in R^m, y_i \in R)$, where x and y represent the eigen and true value. The method incorporates the outcomes of K trees into the final predicted output, which is stated in Equation (22):

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in F \tag{22}$$

F denotes the set of decision trees as given in Equation (23).

$$F = \{f(x) = w_{q(x)}\}(q: R^m \to T, w \in R^T) \tag{23}$$

Suppose $f(x)$ represents one of the trees, T represents the total leaf nodes, $w_{q(x)}$ represents the leaf node's weight, and q represents the tree structure, which maps the sample to the appropriate leaf node. Consequently, the predicted output of XGBoost is obtained by summing the values from the leaf nodes of all trees. The primary objective of the model is to learn these k trees by minimizing the following objective function.

$$L^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{24}$$

The loss l represents the difference between $\hat{y}_i$ and y_i. The regularization term Ω is employed to determine the decision tree's penalty, which helps to prevent overfitting. Ω is represented in Equation (25):

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|\omega\|^2 \tag{25}$$

In the regular term, γ represents the hyperparameter controlling the model's complexity, and T represents the total leaf nodes. λ represents the penalty coefficient. Then, XGBoost performs the Taylor expansion, taking the first few parameters, removing the high-order small infinitesimal parameters, and finally converting the objective function as given in Equation (26):

$$L^{(t)} \approx \sum_{i=1}^{n} \left[l\left(y_i, \hat{y}_i^{(t-1)}\right) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_k) \tag{26}$$

The optimal leaf node score, $w_j^* = -\frac{G_j}{H_j + \lambda}$, is derived by iterating the tree model and its leaf nodes. Where, G_j is $\sum_{i=1}^{I_j} g_j$ and H_j is $\sum_{i=1}^{I_j} h_j$, the optimal value is substituted to obtain the final objective function:

$$Obj = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T \tag{27}$$

III. RESULT AND DISCUSSION

The identification of plant diseases has gained significance in recent years due to the decreasing number of farmers with in-depth knowledge. Automating plant disease detection helps improve plant growth and productivity. Many plants are cultivated in the agricultural field, and they are also affected by various diseases. To address this, the largest publicly available dataset was selected. It contains around 9 types of plants and their possible diseases. In total, the number of categories is 38. To perform this high-level multiclass classification, a highly efficient model is required. Meanwhile, to deploy the model in a mobile application, it should be lightweight. The ST

model is used to retrieve leaf features. The extracted features are then reduced in dimensionality using PCA and LDA. The reduced features are given to ML models such as SVM, RF, and XGBoost.

The experiment was conducted on Google Colab using the available GPU. Python programming and the TensorFlow library were used. Six different models — ST-PCA-SVM, ST-PCA-RF, ST-PCA-XGBoost, ST-LDA-SVM, ST-LDA-RF, and ST-LDA-XGBoost — were evaluated under the same conditions: same number of epochs, batch size, and optimizer. After training and validation, the models were tested. To evaluate the models, performance metrics such as accuracy, recall, precision, and F1-score (F1) were used. The formulas used to calculate these metrics are given in Equations (28-31). In the equation, TP, TN, FP, and FN represent the True Positives, True Negatives, False Positives, and False Negatives, respectively.

$$Accuracy = (TP + TN)/(TP + FP + TN + FN) \tag{28}$$

$$Precision = TP/(TP + FP) \tag{29}$$

$$Recall = TP/(TP + FN) \tag{30}$$

$$F1\ score = 2 * \big((Precision * Recall)\big)/\big((Precision + Recall)\big) \tag{31}$$

First, the results of ST feature extraction combined with LDA feature reduction across different ML models are analyzed, as shown in Table I. For the ST-LDA features, XGBoost performs the best, achieving an accuracy of 97.58%, recall of 97.49%, precision of 97.67%, and an F1 of 97.57%. Next to XGBoost, the SVM model performs well with 95.63% accuracy, 95.77% recall, 95.46% precision, and 95.62% F1. Finally, the RF model gives poor performance with an accuracy of 92.63%, a recall of 92.85%, a precision of 92.41%, and an F1-score of 92.61%.

TABLE I. PERFORMANCE ANALYSIS OF ST-PCA-ML APPROACHES ON PLANT DISEASE DETECTION

Feature Extraction	Feature Selection	Classification	Accuracy	Recall	Precision	F1-Score
ST	LDA	SVM	95.63	95.77	95.46	95.62
		RF	92.63	92.85	92.41	92.61
		XGBoost	97.58	97.49	97.67	97.57

Next, the results of ST feature extraction combined with PCA feature reduction across various ML models are analyzed, as shown in Table II. Using ST-PCA features, XGBoost gives excellent results with an accuracy, recall, precision, and F1 of 98.16%, 98.19%, 98.21%, and 98.20%, respectively. The SVM model performs around the 96% range, and RF around the 94% range.

TABLE II. PERFORMANCE ANALYSIS OF ST-LDA-ML APPROACHES ON PLANT DISEASE DETECTION

Feature Extraction	Feature Selection	Classification	Accuracy	Recall	Precision	F1-Score
ST	PCA	SVM	96.58	96.81	96.30	96.56
		RF	94.63	94.31	94.91	94.61
		XGBoost	98.16	98.19	98.21	98.20

All six ML model combinations are compared using a bar graph in Figure 3. The figure clearly shows that XGBoost with ST and PCA features outperforms other models in plant leaf disease detection. The excellent performance is attained by the proposed ST-PCA-XGBoost network.

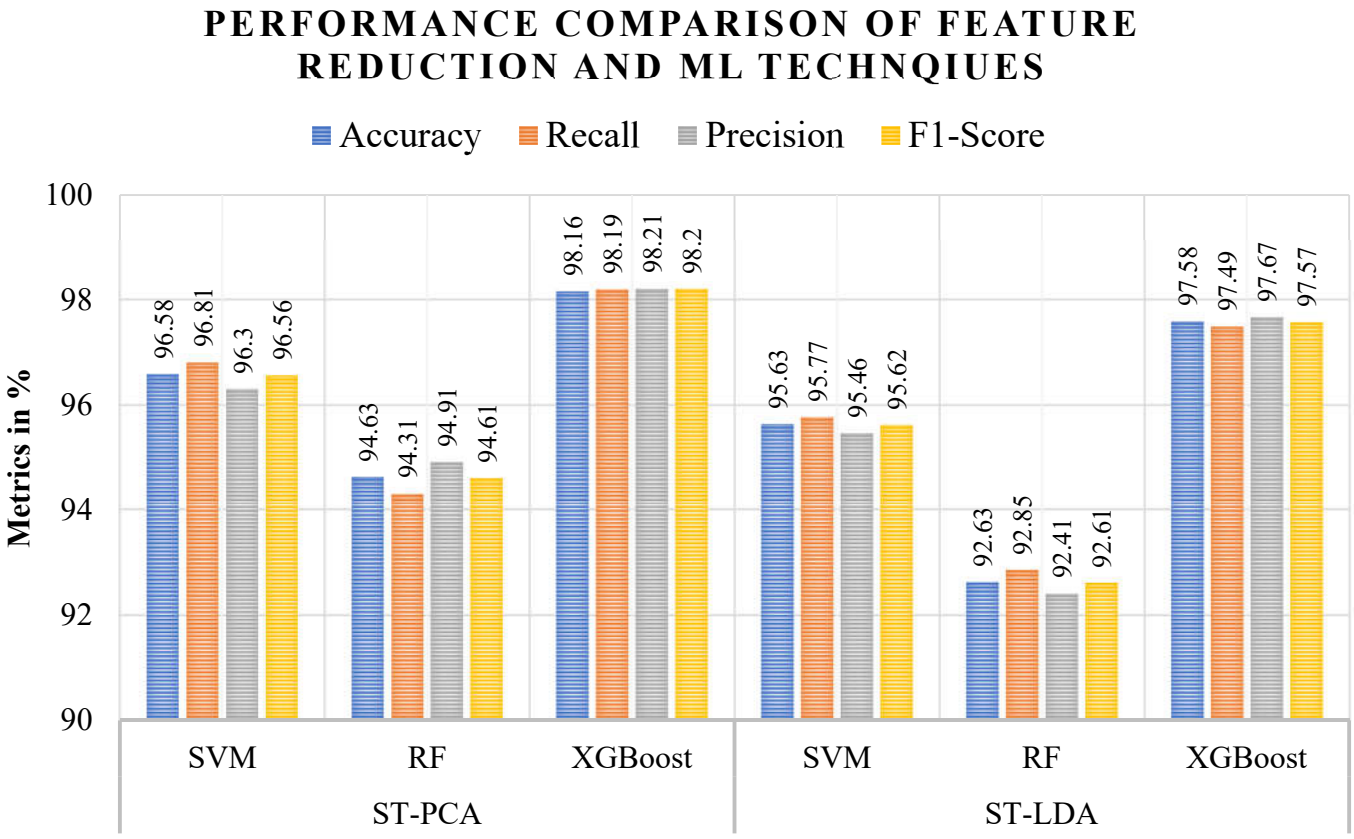


Fig. 4. Performance comparison of different combinations of feature selection and ML techniques

The confusion matrix for the proposed network is illustrated in Figure 4. The matrix confirms accurate identification of diseases such as Apple Black Rot, Apple Healthy, Cherry Powdery Mildew, Cherry Healthy, Corn Rust, Grape Black Rot, Pepper Bell Healthy, Potato Healthy, Pepper Bell Bacterial Spot, Soybean Healthy, Tomato Late Blight, Early Blight, Target Spot, Yellow Leaf Curl Virus, and Mosaic Virus. Table III provides the detailed classification report for each category in the dataset.

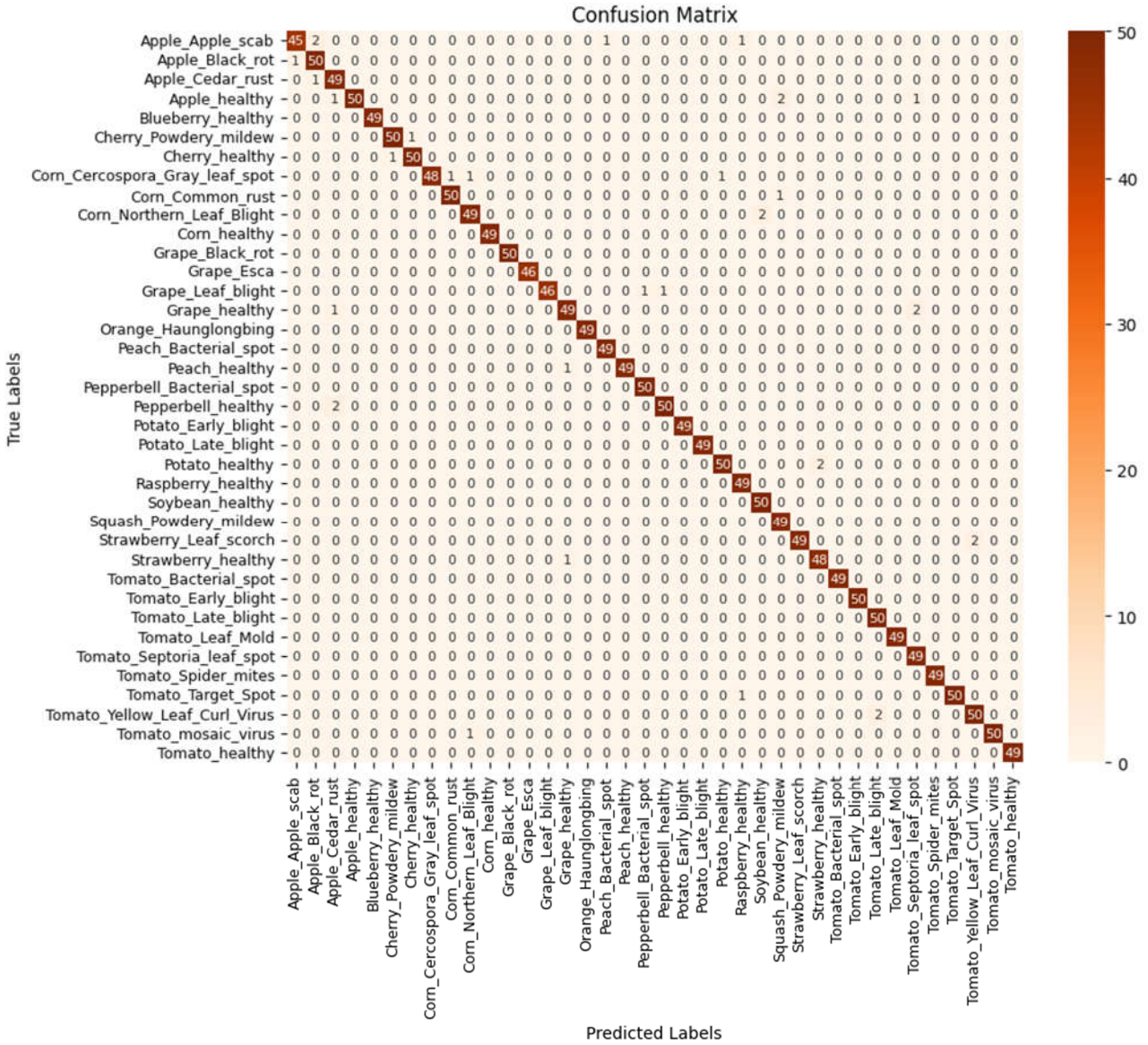


Fig. 5. Confusion Matrix of ST-PCA-XGBoost on Plantvillage Dataset

TABLE III. CLASSIFICATION REPORT OF ST-PCA-XGBOOST ON PLANTVILLAGE DATABASE

Class	Precision	Recall	F1-Score	Class	Precision	Recall	F1-Score
Apple Scab	0.9783	0.9184	0.9474	Healthy Pepperbell	0.9804	0.9615	0.9709
Apple Black Rot	0.9434	0.9804	0.9615	Potato Early Blight	1.0000	1.0000	1.0000
Apple Cedar Rust	0.9245	0.9800	0.9515	Potato Late Blight	1.0000	1.0000	1.0000
Healthy Apple	1.0000	0.9259	0.9615	Healthy Potato	0.9804	0.9615	0.9709
Healthy Blueberry	1.0000	1.0000	1.0000	Healthy Raspberry	0.9608	1.0000	0.9800
Cherry Powdery Mildew	0.9804	0.9804	0.9804	Healthy Soybean	0.9615	1.0000	0.9804
Healthy Cherry	0.9804	0.9804	0.9804	Squash Powdery Mildew	0.9423	1.0000	0.9703
Corn Gray Leaf Spot	1.0000	0.9412	0.9697	Strawberry Leaf Scorch	1.0000	0.9608	0.9800
Corn Common Rust	0.9804	0.9804	0.9804	Healthy Strawberry	0.9600	0.9796	0.9697
Corn Northern Leaf Blight	0.9608	0.9608	0.9608	Tomato Bacterial Spot	1.0000	1.0000	1.0000
Healthy Corn	1.0000	1.0000	1.0000	Tomato Early Blight	1.0000	1.0000	1.0000
Grape Black Rot	1.0000	1.0000	1.0000	Tomato Late Blight	0.9615	1.0000	0.9804
Grape Esca	1.0000	1.0000	1.0000	Tomato Leaf Mold	1.0000	1.0000	1.0000

Grape Leaf Blight	1.0000	0.9583	0.9787	Tomato Septoria Leaf Spot	0.9423	1.0000	0.9703
Healthy Grape	0.9608	0.9423	0.9515	Tomato Spider Mites	1.0000	1.0000	1.0000
Orange Haunglongbing	1.0000	1.0000	1.0000	Tomato Target Spot	1.0000	0.9804	0.9901
Peach Bacterial Spot	0.9800	1.0000	0.9899	Tomato Yellow Curl Virus	0.9615	0.9615	0.9615
Healthy Peach	1.0000	0.9800	0.9899	Tomato Mosaic Virus	1.0000	0.9804	0.9901
Pepperbell Bacterial Spot	0.9804	1.0000	0.9901	Healthy Tomato	1.0000	1.0000	1.0000

After evaluation, it is confirmed that the ST-PCA-XGBoost model performs well in plant disease detection and is deployed in a mobile application. The mobile app is designed using MIT App Inventor. The user interface is designed using the Designer screen, and the logic is implemented using blocks. A sample program created in MIT App Inventor is shown in Figure 5.

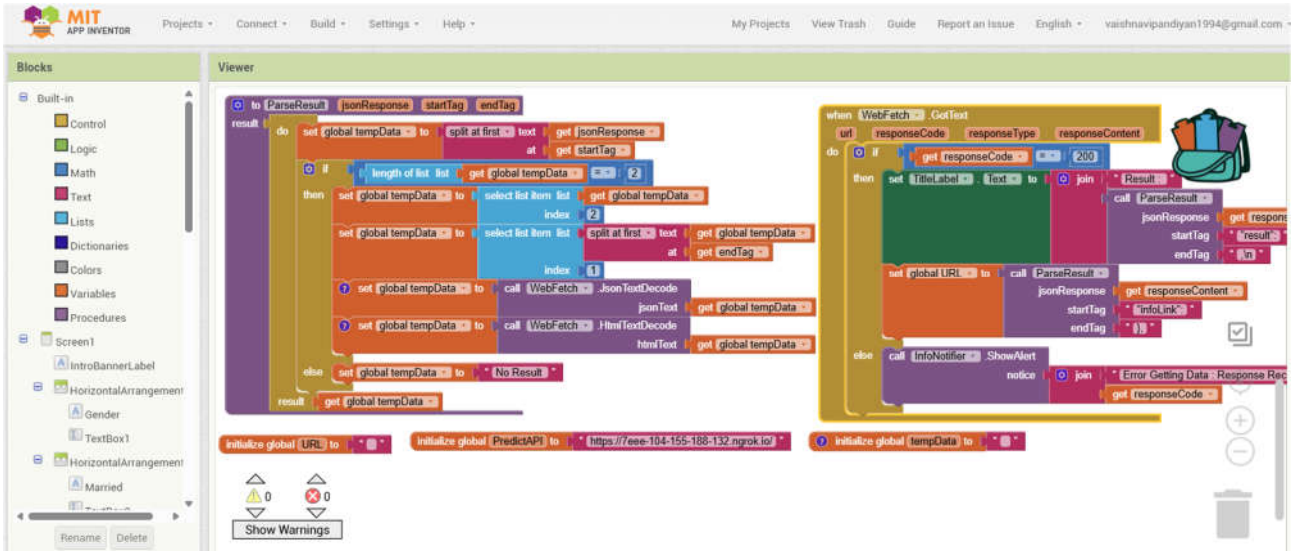


Fig. 6. Mobile Application Programming using MIT App Inventor

The ST-PCA-XGBoost model is stored in Firebase in .h5 format, and MIT App Inventor is linked with Firebase to access the model. The functionality of the mobile app is tested by uploading leaf images. On the home page, an upload option is available. The user can upload the leaf image with a simple click. After uploading, a preview of the image is displayed, and a submit button becomes visible. When the submit button is clicked, the image is sent to Firebase. The model stored in Firebase analyzes the image and identifies the plant leaf's health status. The result is sent back to the mobile app.

The app displays the health status of the leaf, the detected disease, its probability, and recommended future actions. A back button is also available for the user to return to the home screen and test other images. The step-by-step working of the designed mobile app is shown in Figure 6. The working of the mobile application proves that the proposed research framework is highly reliable for predicting leaf diseases from images. It is suitable for real-time use, and even users with zero knowledge of AI can access the mobile app and benefit from it.

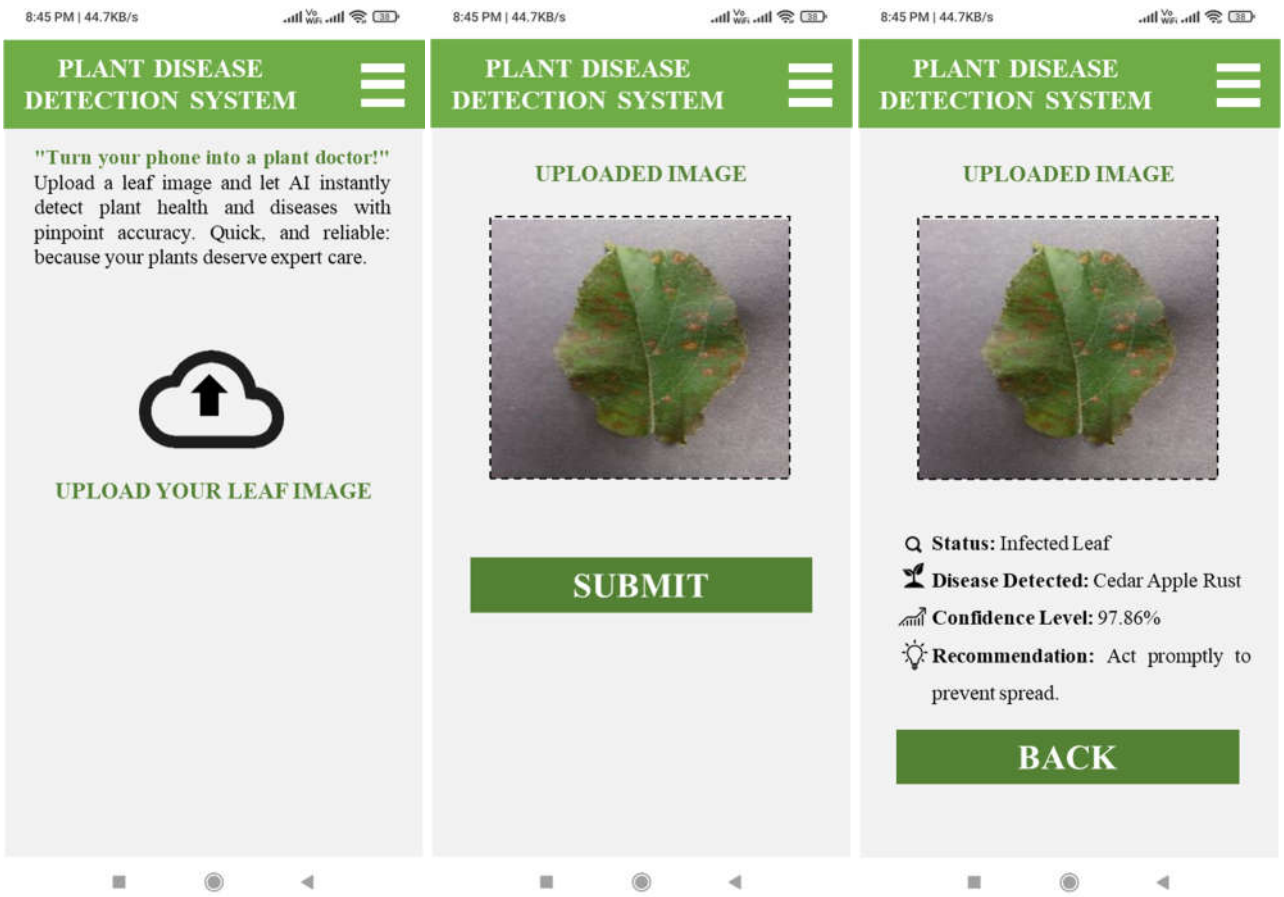


Fig. 7. Working of the designed mobile app for plant disease detection

IV. CONCLUSION

The research aims to provide a solution for farmers to identify plant diseases using advanced technologies like ML and IoT. To conduct the research, a PlantVillage dataset of 38 categories of affected and healthy leaf images was taken from various crops. To classify the images with high accuracy, the features taken for classification are very important. In this research, the ST is used to extract the features from the leaves. Further, the most important features are selected by LDA and PCA techniques. The outcomes of both techniques are given to the ML models of SVM, RF, and XGBoost separately for classification. The ST-PCA-based SVM, RF, and XGBoost, and the ST-LDA-based SVM, RF, and XGBoost are evaluated using standard metrics. In the performance comparison, the ST-PCA-XGBoost gives the highest accuracy of 98.16%. The model is finalized and saved in the Firebase cloud for easy access from the mobile application. The mobile app is designed using MIT App Inventor software. The designed mobile application is tested by uploading the leaf images and predicting the output correctly, displaying it within a few seconds. This shows that the designed framework will be helpful for society in detecting plant diseases. It is believed that this will pave the path for improving farmers' living standards as well as help in increasing the country's economy.

The research has several limitations, such as using only a few crops for detecting plant diseases. But this is not enough—future research should expand to detecting diseases in at least 50 crops. Next, the model is deployed in the cloud and can be accessed easily, but security is not

considered in this research. In the future, the research will improve the model's security by using advanced technologies like blockchain.

REFERENCE

1. Nichit, Savita Vitthalrao. "India's Economic Performance of Agriculture, Industry & Services." *Indian Economy Indian Economy–Prospects and Future* (2017).
2. Sharma, Smriti, Rubaljot Kooner, and Ramesh Arora. "Insect pests and crop losses." *Breeding insect resistant crops for sustainable agriculture* (2017): 45-66.
3. Oliver, Y. M., M. J. Robertson, and M. T. F. Wong. "Integrating farmer knowledge, precision agriculture tools, and crop simulation modelling to evaluate management options for poor-performing patches in cropping fields." *European Journal of Agronomy* 32, no. 1 (2010): 40-50.
4. Jafar, Abbas, Nabila Bibi, Rizwan Ali Naqvi, Abolghasem Sadeghi-Niaraki, and Daesik Jeong. "Revolutionizing agriculture with artificial intelligence: plant disease detection methods, applications, and their limitations." *Frontiers in Plant Science* 15 (2024): 1356260.
5. Kumar, P., and Salman Latheef TA. "Projection of Plant Leaf Disease Using Support Vector Machine Algorithm." In *2023 International Conference on Recent Advances in Science and Engineering Technology (ICRASET)*, pp. 1-6. IEEE, 2023.
6. Hayit, Tolga, Ali Endes, and Fatma Hayit. "KNN-based approach for the classification of fusarium wilt disease in chickpea based on color and texture features." *European Journal of Plant Pathology* 168, no. 4 (2024): 665-681.
7. Nikith, B. V., N. K. S. Keerthan, M. S. Praneeth, and T. Amrita. "Leaf disease detection and classification." *Procedia Computer Science* 218 (2023): 291-300.
8. Rozikin, Chaerur, Agus Buono, Sri Wahjuni, and Chusnul Arif. "Benchmarking the LGBM, random forest, and XGBoost models based on accuracy in classifying melon leaf disease." *International Journal of Advanced Computer Science and Applications* 14, no. 10 (2023).
9. Suprayoga, Rizky, Suparsianto Zega, and Siti Mardiana. "Classification of Mango Leaf Diseases Using XGBoost Method and HoG Feature Extraction." In *2023 International Conference on Modeling & E-Information Research, Artificial Learning and Digital Applications (ICMERALDA)*, pp. 197-202. IEEE, 2023.
10. Dubey, Ratnesh Kumar, and Dilip Kumar Choubey. "Efficient prediction of blast disease in paddy plant using optimized support vector machine." *IETE Journal of Research* 70, no. 4 (2024): 3679-3689.
11. Ahmed, Imtiaz, and Pramod Kumar Yadav. "Plant disease detection using machine learning approaches." *Expert Systems* 40, no. 5 (2023): e13136.
12. Banerjee, Deepak, Vinay Kukreja, Shanmugasundaram Hariharan, Vishal Jain, and Varun Jindal. "Predicting Tulip Leaf Diseases: A Integrated CNN and Random Forest Approach." In *2023 World Conference on Communication & Computing (WCONF)*, pp. 1-6. IEEE, 2023.
13. Chen, Junde, Adnan Zeb, Yaser Ahangari Nanehkaran, and Defu Zhang. "Stacking ensemble model of deep learning for plant disease recognition." *Journal of Ambient Intelligence and Humanized Computing* 14, no. 9 (2023): 12359-12372.
14. Sahu, Santosh Kumar, and Manish Pandey. "An optimal hybrid multiclass SVM for plant leaf disease detection using spatial Fuzzy C-Means model." *Expert systems with applications* 214 (2023): 118989.
15. Islam, Md Manowarul, Md Abdul Ahad Adil, Md Alamin Talukder, Md Khabir Uddin Ahamed, Md Ashraf Uddin, Md Kamran Hasan, Selina Sharmin, Md Mahbubur Rahman, and Sumon Kumar Debnath. "DeepCrop: Deep learning-based crop disease prediction with web application." *Journal of Agriculture and Food Research* 14 (2023): 100764.
16. Saranya, R., and M. S. Suvetha. "IoT-Based Leaf Disease Detection and Alerting System Using K-means Algorithm." In *2023 International Conference on Intelligent Technologies for Sustainable Electric and Communications Systems (iTech SECOM)*, pp. 201-206. IEEE, 2023.
17. Hughes, David, and Marcel Salathé. "An open access repository of images on plant health to enable the development of mobile disease diagnostics." *arXiv preprint arXiv:1511.08060* (2015).

18. Liu, Ze, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. "Swin transformer: Hierarchical vision transformer using shifted windows." In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012-10022. 2021.
19. Liu, Ze, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning et al. "Swin transformer v2: Scaling up capacity and resolution." In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12009-12019. 2022.
20. Xie, Zhenda, Yutong Lin, Zhuliang Yao, Zheng Zhang, Qi Dai, Yue Cao, and Han Hu. "Self-supervised learning with swin transformers." *arXiv preprint arXiv:2105.04553* (2021).
21. Zhao, Shuping, Bob Zhang, Jian Yang, Jianhang Zhou, and Yong Xu. "Linear discriminant analysis." *Nature Reviews Methods Primers* 4, no. 1 (2024): 70.
22. Mika, Sebastian, Gunnar Ratsch, Jason Weston, B. Scholkopf, Alex Smola, and K-R. Muller. "Constructing descriptive and discriminative nonlinear features: Rayleigh coefficients in kernel feature spaces." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25, no. 5 (2003): 623-628.
23. Mishra, Sidharth Prasad, Uttam Sarkar, Subhash Taraphder, Sanjay Datta, Devi Swain, Reshma Saikhom, Sasmita Panda, and Menalsh Laishram. "Multivariate statistical data analysis-principal component analysis (PCA)." *International Journal of Livestock Research* 7, no. 5 (2017): 60-78.
24. Sarang, Poornachandra. "Support vector machines: a supervised learning algorithm for classification and regression." In *Thinking Data Science: A Data Science Practitioner's Guide*, pp. 153-165. Cham: Springer International Publishing, 2023.
25. Negi, Harendra Singh, Sushil Chandra Dimri, Bhawnesh Kumar, and Mangey Ram. "Support vector machine and classification, kernel trick for separating of data points." *Mathematics in Engineering, Science & Aerospace (MESA)* 15, no. 2 (2024).
26. Parmar, Aakash, Rakesh Katariya, and Vatsal Patel. "A review on random forest: An ensemble classifier." In *International conference on intelligent data communication technologies and internet of things*, pp. 758-763. Cham: Springer International Publishing, 2018.
27. Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785-794. 2016.
28. Hakkal, Soukaina, and Ayoub Ait Lahcen. "XGBoost to enhance learner performance prediction." *Computers and Education: Artificial Intelligence* 7 (2024): 100254.