**Plant Disease Detection Using Image Processing and Machine Learning**

**NAREDLA NAGANNA DORA SASTRY (S20220020298) GUDLA VENKATA SAI KRISHNA (S20220020273) ALLAMRAJU KARTHIKEYA (S20220020252)**

**SIYYADRI BHARGAV (S20220020313)**

**Department of Electrical and Electronic Engineering, IIIT CHITTOOR**

**Abstract:** One of the important and tedious task in agricultural practices is detection of disease on crops. It requires huge time as well as skilled labor. This paper proposes a smart and efficient technique for detection of crop disease which uses computer vision and machine learning techniques. The proposed system is able to detect 20 different diseases of 5 common plants with 93% accuracy. Keywords: Digital image processing, Foreground detection, Machine learning.

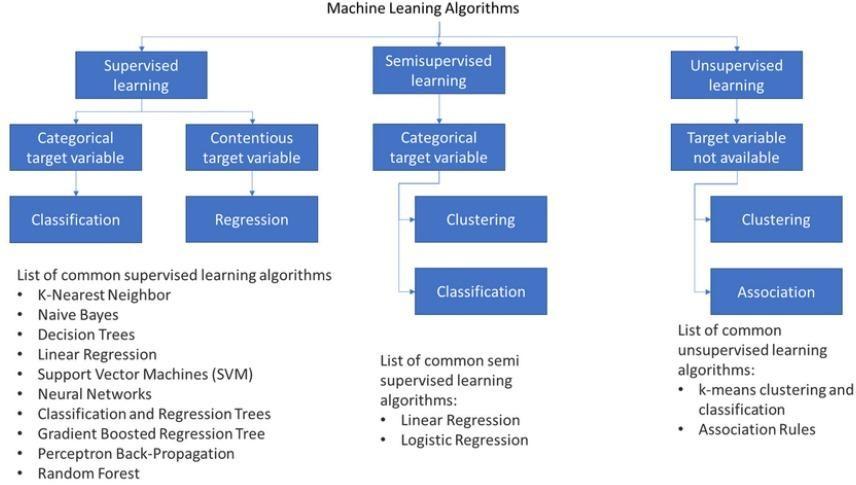
Our study evaluates the ability of the model to generalize across various plant species. The findings indicate encouraging accuracy levels, showcasing the effectiveness of machine learning in creating automated and scalable solutions for plant disease detection, which are vital for promoting sustainable agricultural practices.

**Keywords:** Digital image processing, Foreground detection, Machine learning, Plant disease detection.

# Introduction

In India about 70% of the populace relies on agriculture. Identification of the plant diseases is important in order to prevent the losses within the yield. It's terribly troublesome to observe the plant diseases manually. It needs tremendous quantity of labor, expertize within the plant diseases, and conjointly need the excessive time interval. Hence, image processing and machine learning models can be employed for the detection of plant diseases. In this project, we have described the technique for the detection of plant diseases with the help of their leaves pictures. Image processing is a branch of signal processing which can extract the image properties or useful information from the image. Machine learning is a sub part of artificial intelligence which works automatically or give instructions to do a particular task. The main aim of machine learning is to understand the training data and fit that training data into models that should be useful to the people. So it can assist in good decisions making and predicting the correct output using the large amount of training data. The color of leaves, amount of damage to leaves, area of the leaf, texture parameters are used for classification. In this project we have analyzed different image parameters or features to identifying different plant leaves diseases to achieve the best accuracy. Previously plant disease detection is done by visual inspection of the leaves or some chemical processes by experts. For doing so, a large team of experts as well as continuous observation of plant is needed, which costs high when we do with large farms. In such conditions, the recommended system proves to be helpful in monitoring large fields of crops. Automatic detection of the diseases by simply seeing the symptoms on the plant leaves makes it easier as well as cheaper. The proposed solution for plant disease detection is

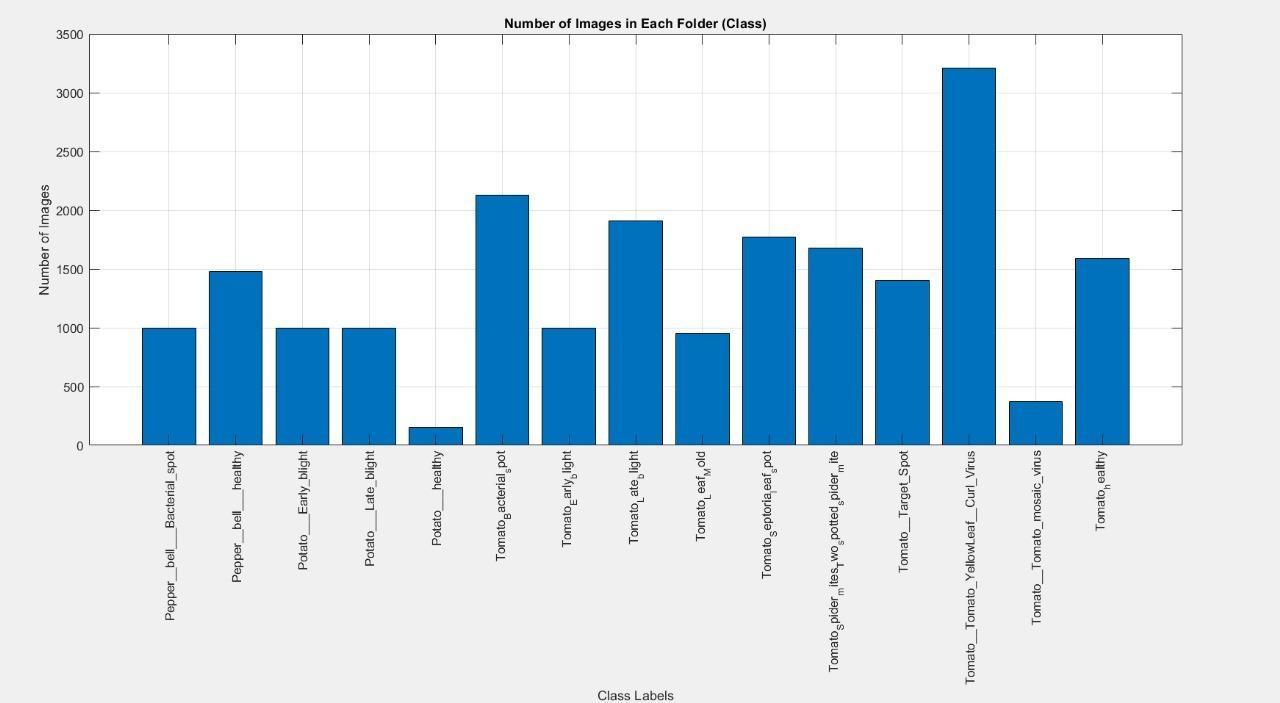
computationally less expensive and requires less time for prediction than other deep learning based approaches since it uses statistical machine learning and image processing algorithm.



# Methodology

* 1. ***Dataset***

*The bar chart shows the number of images available for each plant species or disease class in the dataset. Each bar represents a different class, with the y-axis indicating the image count. There is a noticeable imbalance, as some classes, like "Tomato\_Yellow\_Leaf\_Curl\_Virus," have many images, while others, like "Potato\_healthy," have significantly fewer. This imbalance may affect model accuracy, as classes with fewer images may be harder for the model to recognize. Applying data augmentation techniques to underrepresented classes could help improve the dataset’s balance and enhance the model's performance.*





Some samples from the datasets are shown in Fig.

**2. 2 Data preprocessing and feature extraction.**

**Data preprocessing**

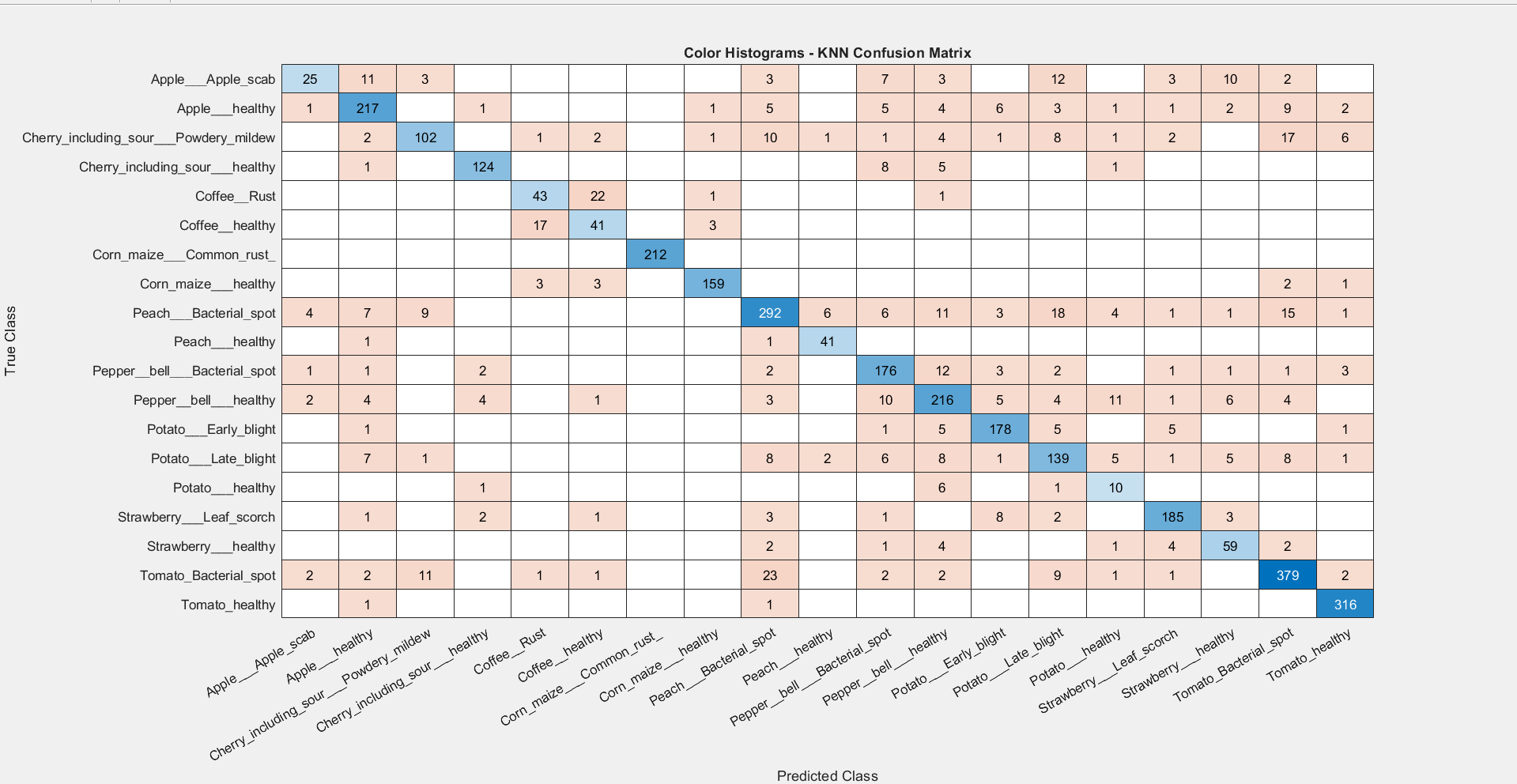
This step focuses on gathering and organizing the dataset to ensure it is ready for analysis. The dataset comprises images, each categorized into folders representing distinct classes (e.g., plant diseases). The process begins by counting the total number of images in each folder to analyze the dataset's class distribution, which is visualized using bar charts for better understanding. This visualization helps identify class

imbalances, if any. Following this, the images are loaded into the workspace, and their labels are extracted based on the folder names. This structured approach ensures that the data is well-prepared for further processing.

**Feature Extraction**

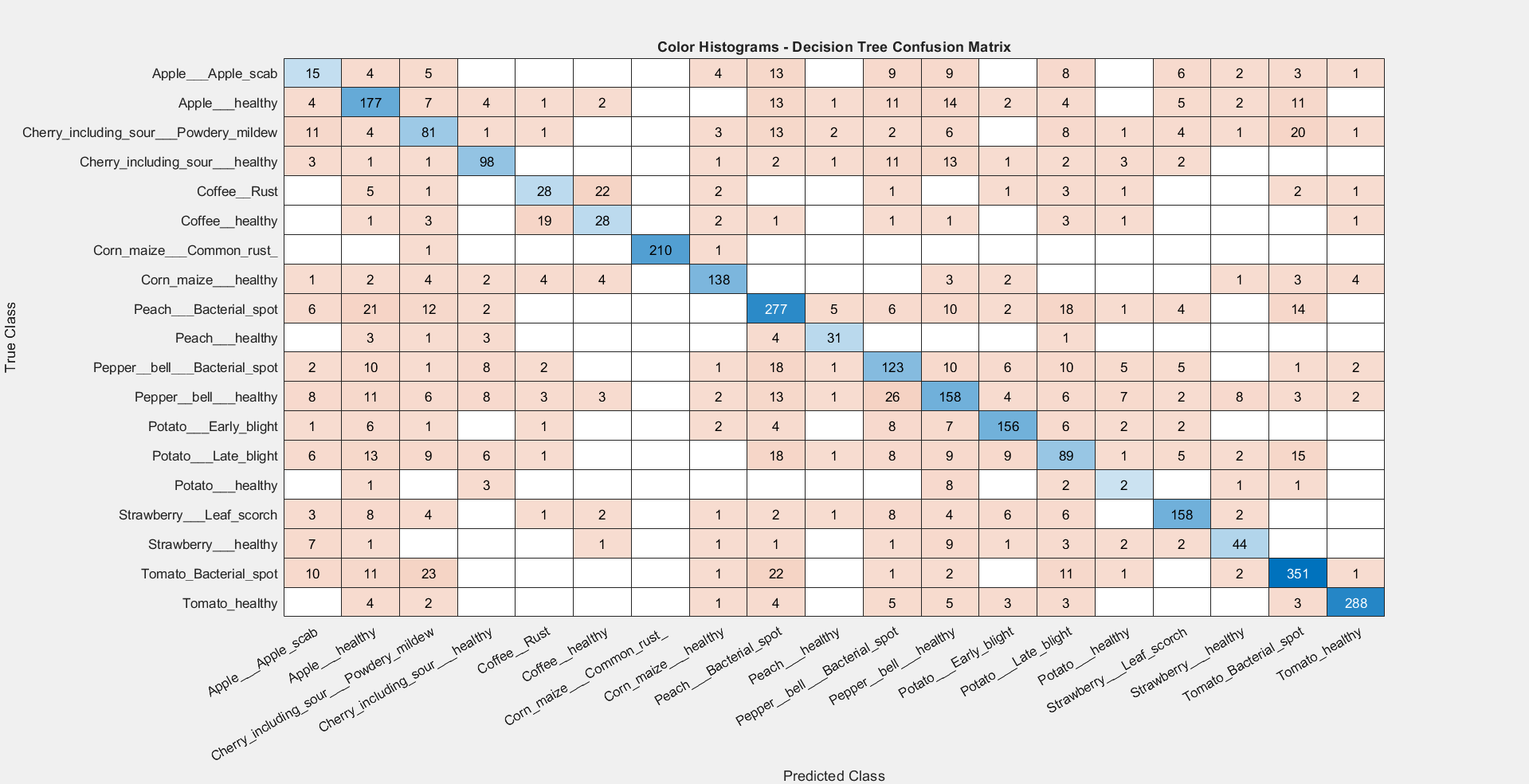
## Feature extraction is a critical step that transforms raw image data into meaningful numerical representations. In this project, RGB histograms are computed for each image, capturing the distribution of pixel intensities across the Red, Green, and Blue channels. These histograms are normalized to ensure consistency, preventing large images from disproportionately influencing the results. Finally, the normalized histograms for all three channels are concatenated into a single feature vector. This vector serves as the input for the classification models, enabling them to learn patterns and make accurate predictions.

* 1. **Updated Classification Algorithms**
     1. **Updated K-Nearest Neighbors (KNN)**



KNN is a simple yet powerful algorithm that classifies data points based on the majority label of their k nearest neighbors in the feature space. The choice of k significantly influences its performance: smaller values make the model sensitive to noise, while larger values can oversmooth boundaries. It’s particularly effective for small datasets where patterns are distinct, as it requires no explicit training phase. However, for large datasets, the model can become computationally expensive since it calculates distances for all training points. Proper feature scaling is essential for KNN to ensure meaningful distance calculations.

* + 1. **Updated Decision Tree**

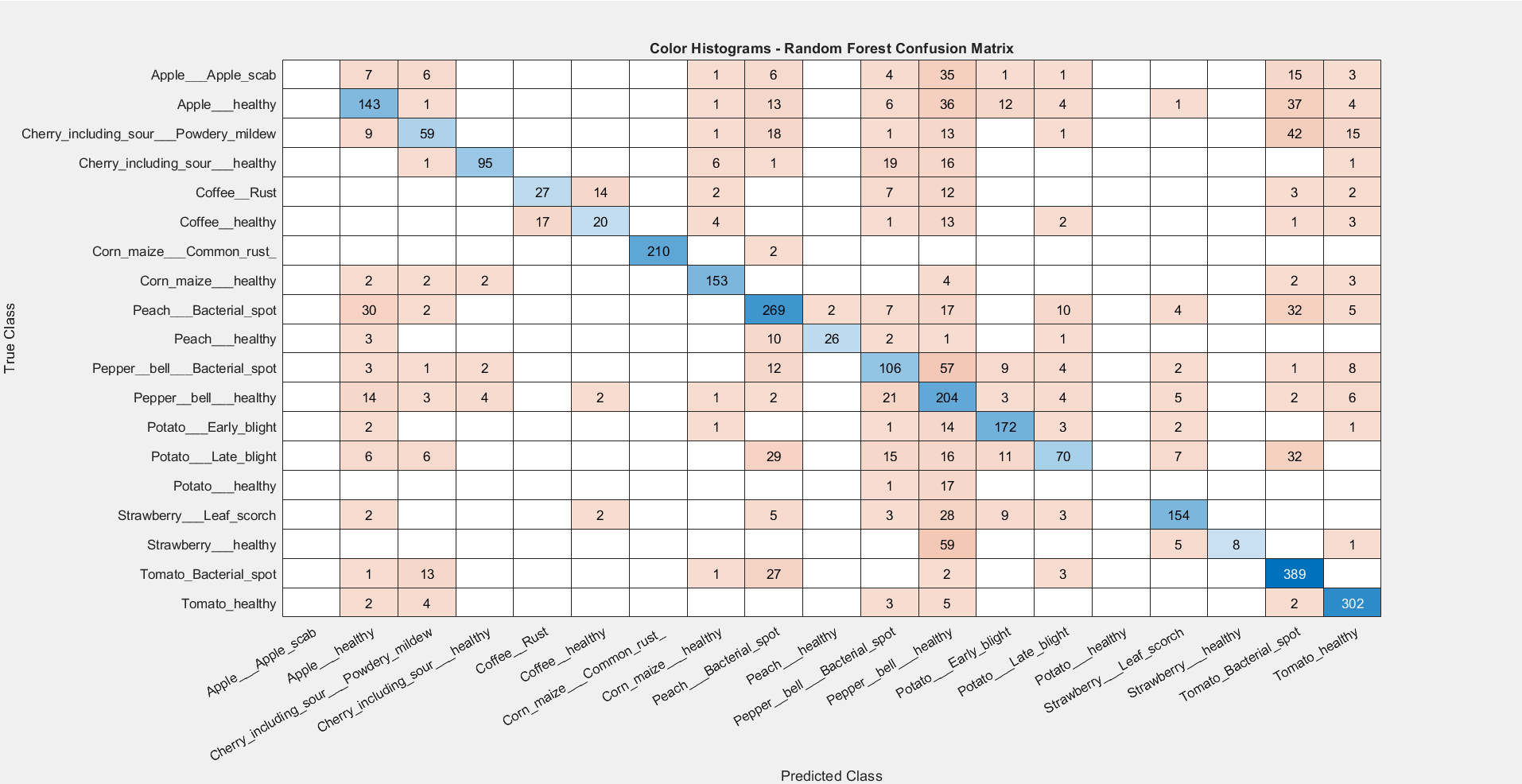


* + 1. **Updated Random Forest**

Decision Tree models split

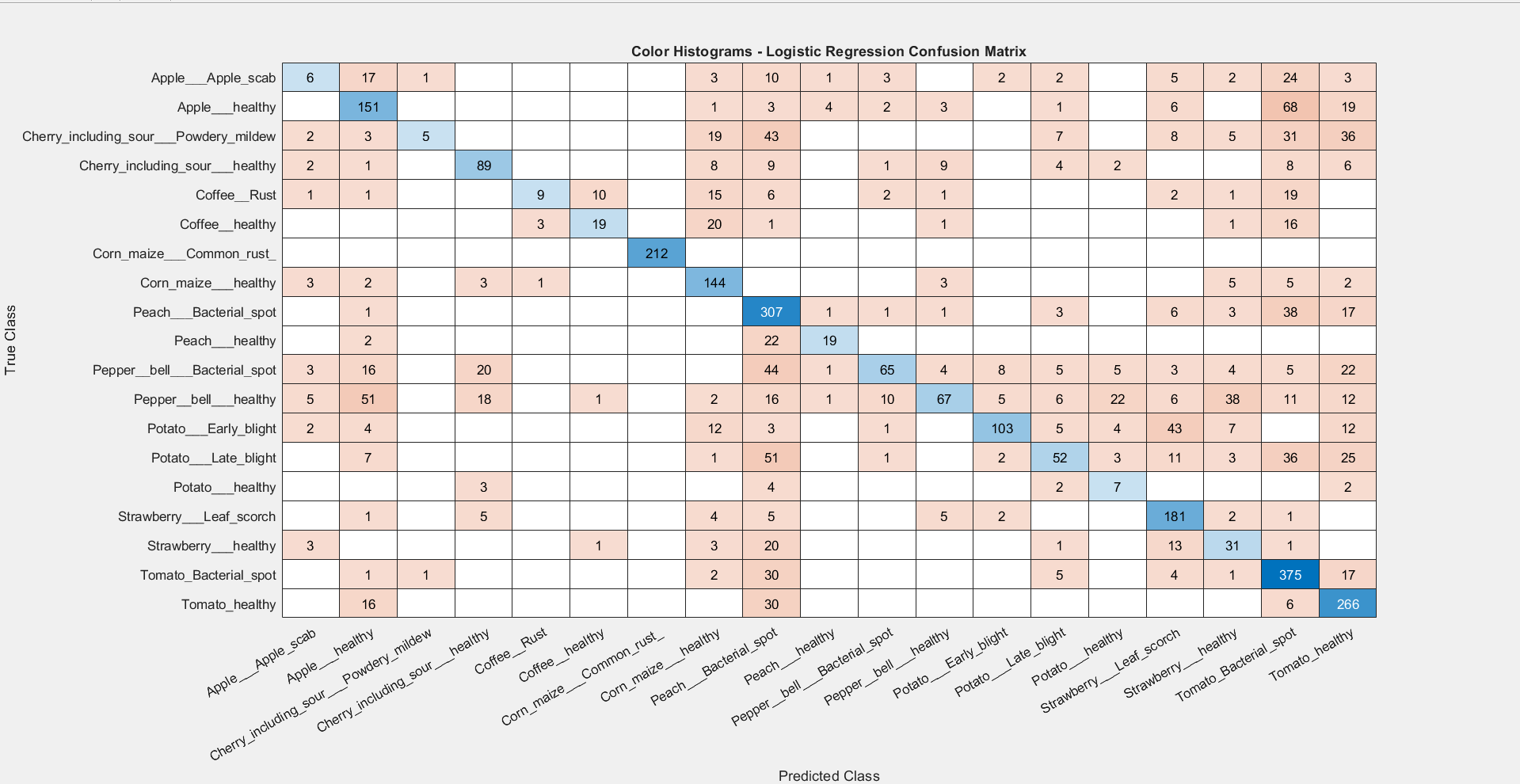
the dataset into subsets based on decision rules derived from the features. The tree structure consists of internal nodes (decision points), branches (decision outcomes), and leaf nodes (predicted classes). Decision Trees are easy to interpret, making them valuable for understanding data relationships.

They can capture non-linear decision boundaries and handle categorical or numerical data effectively. However, they are prone to overfitting when the tree grows too complex. Techniques like pruning or limiting the maximum tree depth are often applied to improve generalization and reduce overfitting.



Random Forest is an ensemble learning method that builds multiple decision trees using random subsets of data and features. The final prediction is made by aggregating (e.g., voting or averaging) the outputs of individual trees, enhancing the model's robustness and accuracy. This approach reduces overfitting andincreases resilience tonoise compared to single decision trees.By combining the predictions of multiple diverse trees, Random Forest achieves better generalization, making it particularly effective for complex datasets with nonlinear relationships.

* + 1. **Updated Logistic Regression (Multiclass)**

****

# 3.Results

Logistic Regression predicts the probability of a data point belonging to a class using a logistic (sigmoid) function. For multiclass classification, strategies like "one-vs-all" are employed, where a separate binary classifier is trained for each class. It is computationally efficient and works well for linearly separable data, providing interpretable results with clear probabilities. However, Logistic Regression struggles with non-linear decision boundaries unless extended with feature transformations. It assumes independence among features, which may limit its performance in cases of highly correlated inputs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Accuracy** | **Precision** | **Recall** | **F1 Score** | **MODEL** |
| **0.8499** | **0.7878** | **0.7905** | **0.7818** | **KNN** |
| **0.7048** | **0.6301** | **0.6246** | **0.6271** | **Decision Tree** |
| **0.6919** | **0.6620** | **0.5667** | **0.5821** | **Random Forest** |
| **0.6059** | **0.6009** | **0.5073** | **0.4971** | **Logistic**  **Regression** |

# Team Contribution

**Kartikeya Task:** Data Collection and Preprocessing

Karthikeya played a crucial role in ensuring the dataset was ready for analysis and feature extraction. He gathered images from the dataset and meticulously preprocessed them. His work included extracting RGB histograms to capture color distribution and Local Binary Patterns (LBP) to capture texture information from each image. Both types of features were then normalized to ensure uniform scaling, which is essential for effective model training and evaluation.

Additionally, Karthikeya ensured that the dataset was organized and formatted correctly to streamline its use in machine learning workflows. This involved verifying data integrity, handling any missing or corrupted entries, and labeling the dataset appropriately. His efforts provided a clean, standardized dataset that served as the foundation for subsequent feature extraction, model training, and clustering.

**NagannaTask:** Feature Extraction, Code Implementation, and Model Training

Naganna was responsible for implementing and training multiple classification models, including KNN, Decision Trees, Random Forest, Logistic Regression, and Naive Bayes. His work started with feature extraction, where he used both color histograms and LBP features generated during preprocessing to create input data for the classifiers.

He also ensured the project was accessible and user-friendly by writing modular code with well-defined paths. Naganna designed the codebase so that it could run seamlessly on any system, regardless of setup, by using a universal directory structure (./foldername). This portability enabled easy replication and testing of the project on multiple devices.

After implementing the models, Naganna evaluated their performance using metrics such as accuracy, precision, recall, and F1-score, comparing the strengths and weaknesses of each classifier.

**Sai krishna Task:** Clustering Algorithms and Code Implementation

Sai Krishna was in charge of implementing clustering algorithms, focusing on K-means and DBSCAN. He applied similar logic as Naganna for coding, ensuring that the clustering scripts were modular and portable. By setting up paths with a ./foldername structure, Sai Krishna made sure the clustering code could be executed seamlessly on any laptop without requiring reconfiguration.

To evaluate clustering quality, Sai Krishna used metrics like the Adjusted Rand Index (ARI), ensuring a thorough assessment of the models' ability to group data effectively. His work contributed to understanding the structure and patterns within the dataset, complementing the classification models.

**Bhargav Task:** Bhargav played a critical role in visualizing the project’s results and ensuring they were communicated effectively. He designed detailed graphs and plots, including precision-recall curves, bar charts, and confusion matrices, to highlight model performance metrics like accuracy, precision, and recall. His focus was not just on accuracy but also on making these visualizations intuitive and engaging for both technical and non-technical audiences.

In addition to visualization, Bhargav collaborated closely with Sai Krishna to prepare the final report and presentation. They worked together to create a cohesive narrative, ensuring the document flowed logically from problem statement to conclusions. The presentation was crafted with an emphasis on clarity, professionalism, and visual appeal, using concise points and engaging visuals to captivate the audience. Bhargav’s efforts in visualization and documentation were instrumental in effectively conveying the project’s key findings and outcomes.

**Regular team gatherings have been organized to facilitate in-depth discussions on our project's advancements, tackle any challenges encountered, and explore opportunities for potential improvements.**

**4.Conclusion:**

## In this project, we implemented and evaluated multiple machine learning models for image-based classification using the PlantVillage dataset. Key models like K-Nearest Neighbors, Decision Trees, Random Forests, Logistic Regression, and Naive Bayes were

employed, demonstrating varying performance in terms of accuracy and precision. Clustering techniques such as K-Means and DBSCAN were also explored for unsupervised learning.

## Among the methods, K-Nearest Neighbors showed the best results, achieving high accuracy and F1-scores. The project highlights the importance of feature extraction and data preprocessing in achieving reliable outcomes. This comprehensive approach provides insights into the strengths and limitations of various algorithms for agricultural disease detection tasks.