Machine Learning: Final Project

Machine Learning Based Leaf Disease Detection

Submitted by:

Galimuthy Elia Anusha,

Keerthana H,

Muhammad Uwais,

Tiya Rose,

Vaidhav Adit

Submitted to: Prof Devanand Thanissery

Table of Contents

1. Problem Statement and Relevance

Agriculture plays a major role in India's economy, employing over 45% of the workforce and contributing significantly to rural livelihoods [1]. Farmers often rely on traditional experience to judge crop health, but this has become harder in recent years due to climate change, irregular rainfall, soil issues, pollution and the spread of new plant diseases. Small farmers are especially affected because they do not always have access to experts or fast diagnostic services. When diseases are detected late, crops can fail, leading to financial losses.

To address this challenge, there is a growing need for simple and accessible tools that can help farmers identify plant diseases early. Machine learning (ML) offers a practical way to analyze leaf images and provide instant feedback on crop health. A quick and reliable prediction system can reduce dependency on external services, save time and help farmers make timely decisions to protect their crops.

1.1 Objectives

Our project has the following main objectives:

1. Use the curated dataset to train a machine learning model capable of making two predictions from a single leaf image:
    - The type of plant
    - Its disease status (healthy or what type of disease)
2. Build a model that works efficiently and gives consistent results, so farmers can rely on it for early disease detection.
3. Make the model transparent by explaining how it works, which includes:
    - How the model makes decisions
    - Which features or patterns in the leaf images are important
    - How the model was trained
    - Why it gives certain predictions
4. To try and make a smart interface that provides AI driven solutions to farmers based on the models prediction.

5. Present a working system that demonstrates how ML can support farmers by reducing time spent on diagnosis and improving decision making.

1.2 Expected Outcomes

- A trained ML model that can classify plant type and disease category with good accuracy.
- Evidence that the model reduces the time needed for disease identification.
- An interpretable and well documented model that shows how image features guide predictions.
- A system that can eventually be integrated into farmer friendly applications.

For the dataset, we have used the Open Leaf Image Dataset (OLID), which contains high quality images of various crops and their disease conditions, along with labels suitable for classification tasks. The dataset is supported by a peer reviewed research publication in Frontiers in Plant Science [2].

2. Data Collection

For this project we combined data from multiple sources and collected as much relevant data as possible. This approach was taken in part to address issues of data imbalance (i.e, where some plant type or disease categories are under represented). However, it is important to note that the OLID-I dataset was the most important dataset in our project and forms the base of our pipeline. Below are the five datasets we used.

2.1 OLID I (Open Leaf Image Dataset)

The OLID I dataset is a publicly released image dataset for plant stress recognition. According to the authors, the images were collected under natural field conditions in Bangladesh at three different sites.

- The dataset includes leaf images of multiple crops and under various stress conditions (both biotic: such as pests or diseases and abiotic: such as nutrient deficiency)
- The images are annotated with two levels of class labels: a “primary class” (crop/plant type) and a “secondary class” (stress/disease or healthy condition).

- Collection methodology: The authors captured images in real agricultural fields rather than only controlled lab settings. They emphasize that this helps model the real world appearance of stress in leaves.
- The dataset size is reported as 4,749 images on Kaggle covering healthy, nutritional deficient and pest infested leaves.
- The dataset enables classification of both plant type and stress status, aligning well with our project's goal of predicting these two attributes.

The list of plant types and their stress labels includes:

Plant Types	**Disease Category**
Tomato	JAS_MIT, K, LM, MIT, N, N_K, HEALTHY
Snake Gourd	K, LS, N, N_K, HEALTHY
Ash Gourd	K, K_Mg, N, N_K, N_MG, PM, HEALTHY
Bitter Gourd	DM, JAS, K, K_Mg, LS, N, N_K, N_Mg, HEALTHY
Bottle Gourd	DM, JAS, JAS_MIT, K, LS, N, N_K, N_Mg, HEALTHY
Ridge Gourd	N, N_Mg, PC, PLEI, PLEI_IEM, PLEI_MIT, HEALTHY
Cucumber	K, N, N_K, HEALTHY
Egg Plant	EB, FB, JAS, K, MIT, MIT_EB, N, N_K, HEALTHY

Table 1: Classes and Types in OLID-I Dataset

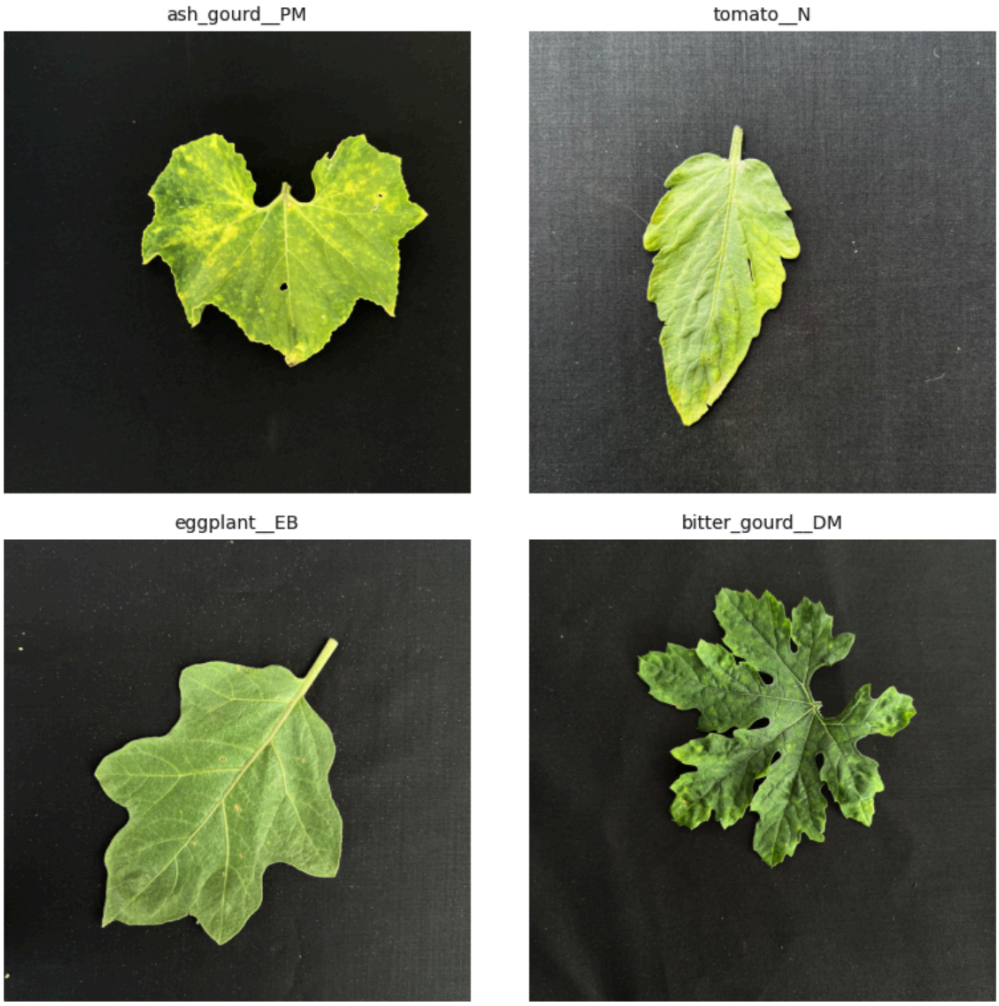


Image 1: Sample Images of Plants and their Diseases

Acronym	**Description**
K	Potassium
K_Mg	Potassium and Magnesium
N	Nitrogen
N_K	Nitrogen and Potassium
N_Mg	Nitrogen and Magnesium

PM	Powdery Mildew
DM	Downy Mildew
JAS	Jassid
LS	Leaf Spot
JAS_MIT	Jassid and Mite
EB	Epilachna Beetle
FB	Flea Beetle
MIT	Mite
MIT_EB	Mite and Epilachna Beetle
PC	Pumpkin Caterpillar
PLEI	Pumpkin Leaf Eating Insect
PLEI_IEM	Pumpkin Leaf Insect and Insect Egg Mass
PLEI_MIT	Pumpkin Leaf Eating Insect and Mite
LM	Leaf Miner

Table 2: Stress Labels and Their Full Form

2.2 Proposed Layered BG Dataset Computer Vision Dataset

This dataset was taken from Roboflow Universe, where 2430 augmented multiple Bitter Gourd images along with the nine leaf deficiency or classes were found. This dataset was designed to support deep learning tasks to detect plant disease and nutrient deficiency, hence aligning with our goal [4].

The nine classes includes Healthy- 532, Downy mildew- 171, Leaf Spot-103, Jassid-96 , Potassium and Magnesium Deficiency-126 , Nitrogen Deficiency- 491, Nitrogen and Potassium Deficiency-401 , Nitrogen and Magnesium Deficiency-371 , Potassium Deficiency-139.

2.3 Cucumber Leaf Disease Dataset

The dataset is taken from kaggle and has 3754 images with 3 classes of Healthy leaf, Powdery mildew and Downy mildew. All the images have a white background and are augmented before only [5].

2.4 Bottle Gourd

This dataset is taken from kaggle and has 1819 images with 3 classes of Healthy leaf-521 , Downy mildew-691 and Anthracnose-607. All these images have a white background and are augmented before only [6].

2.5 Tomatooo

This dataset is taken from kaggle and has 2610 images with 4 classes of tomato bacterial spot-600, tomato fresh leaf-601, tomato fresh leaf virus-754, tomato spotted wilt- 655. All these images have a white background and are augmented before only [7].

Image 2: Sample Images of Plants and their Diseases

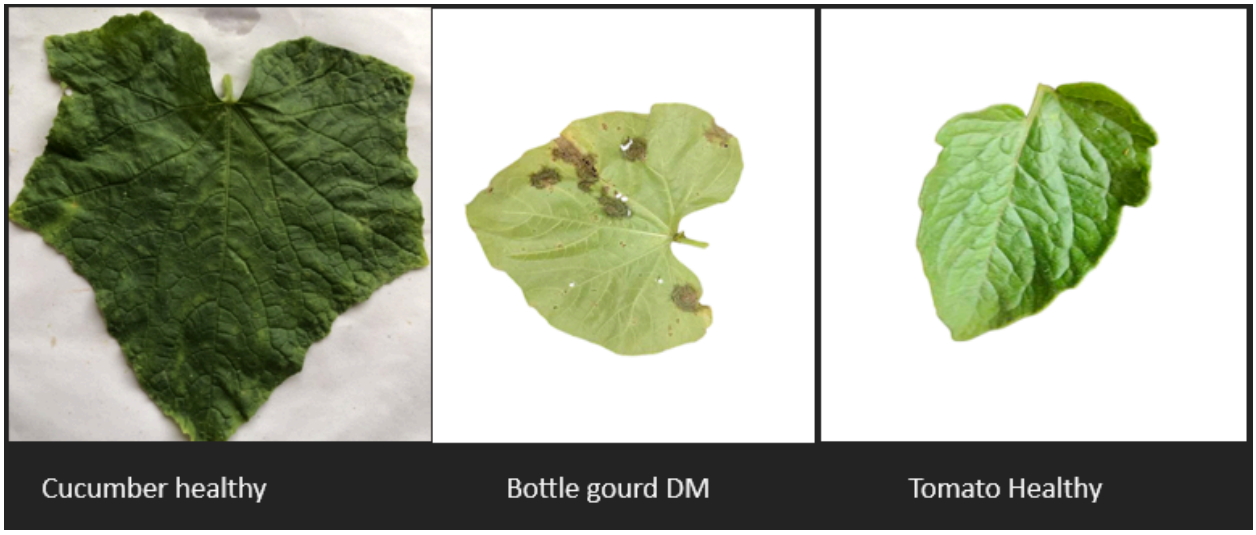


Image 3: Sample Images of Plants and their Diseases

Plant Types	Disease Category
Bitter Gourd	DM, JAS, K, K_Mg, LS, N, N_K, N_Mg, HEALTHY
Cucumber	PM, DM, HEALTHY
Bottle Gourd	DM, ANTHRACNOSE, HEALTHY
Tomato	BACTRIAL, VIRUS, SPOTTED, HEALTHY

Table 3: Classes and Types From Other Datasets

3. Exploratory Data Analysis

3.1. Initial Dataframe Formation

To build our dataset, we started by automatically scanning the project folders to locate every image file. Instead of manually labeling thousands of photos, we used a different and more efficient approach. Since the images were already organized into specific folders (A folder named “tomato_healthy” contained only healthy tomato leaves), we simply used those folder names to identify the category for each image.

For every image found during the scan, the system identified the name of the folder immediately containing it. This folder name was automatically captured and assigned as the official "Label" for that specific image.

Finally, we combined all this information into a single dataframe.The generated dataframe had two features for every sample:

- The Source Path: The precise location of the image file, allowing the system to retrieve it when needed.
- The Category: The label derived from the folder structure (e.g, "ash_gourd_K").

```
[3]: df.head()
```

[3]:

	image_path	label
0	/kaggle/input/olid-i/ash_gourd__PM/ash_gourd__...	ash_gourd__PM
1	/kaggle/input/olid-i/ash_gourd__PM/ash_gourd__...	ash_gourd__PM
2	/kaggle/input/olid-i/ash_gourd__PM/ash_gourd__...	ash_gourd__PM
3	/kaggle/input/olid-i/ash_gourd__PM/ash_gourd__...	ash_gourd__PM
4	/kaggle/input/olid-i/ash_gourd__PM/ash_gourd__...	ash_gourd__PM

Image 1: The Initial Data Frame

3.2 Class Distribution

The first thing to do before starting out any sort of pre-processing was to do an examination of the class distribution in the dataset. This was done to assess the intrinsic class imbalance present in the dataset. Since our goal was to build an accurate machine learning model, it was important to identify categories that were over represented or under represented. A model trained on an imbalanced dataset can easily become biased toward the majority classes, leading to misleadingly high accuracy while performing poorly on minority classes.

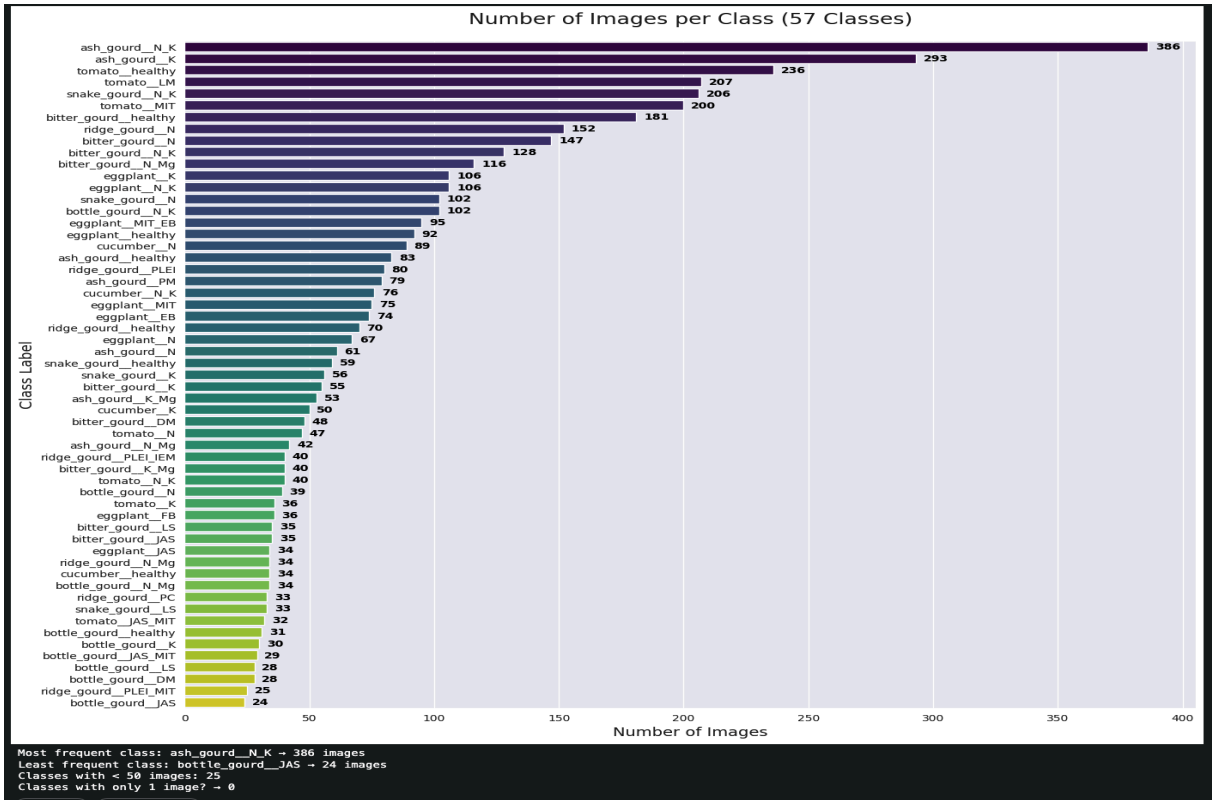


Image 2: Class Distribution of the OLID-I Dataset

From Image 2 we can infer that the dataset is highly imbalanced. Certain plant and status types have many more images than others. This suggested that the model could end up being biased towards some particular class types.

To solve this, we thought of multiple solutions; gather more data, downscaling, upscaling or using sampling methods that try and solve data imbalance. Each of the methods will be explained in detail under section 3.3.

3.3 Filtering the Data

The dataset contains multiple classes for both plant type and disease status. As shown in Table 1, a single plant type can appear under several different “status” categories. For example, Ash Gourd includes labels such as K, N, K_N, and others. These labels fall into two groups: simple and composite.

- Simple labels represent plants with a single deficiency or issue (ash_gourd_K, tomato_N).
- Composite labels represent plants showing multiple deficiencies at the same time (ash_gourd_N_Mg, snake_gourd_N_K).

For the purposes of our project, we decided to drop the composite labels. This was because composite labels create a more complex pattern, and distinguishing them from individual labels can be extremely challenging for a traditional machine learning model. For example, telling apart K vs. K_Mg requires the model to detect subtle combined symptoms that often overlap with or resemble single issue symptoms. Another reason was to try to reduce class imbalance. In section 3.2 we saw that our dataset was highly skewed towards some data points. These repetitive ones also included composite labels like ash_gourd_N_K. Since our project is restricted to classical ML models rather than deep learning, we prioritised model performance and clarity. To keep the classification more reliable and reduce unnecessary confusion for the model, we excluded composite labels from our final dataset.

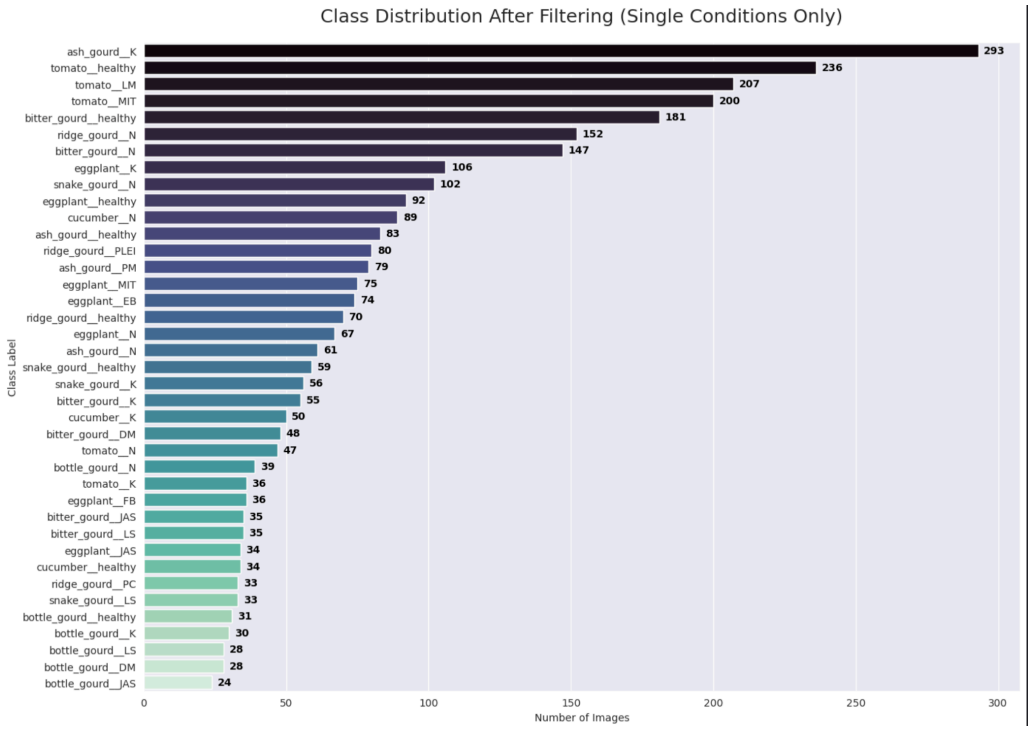


Image 3.1: New Class Distribution After Filtering Out the Composite Labels

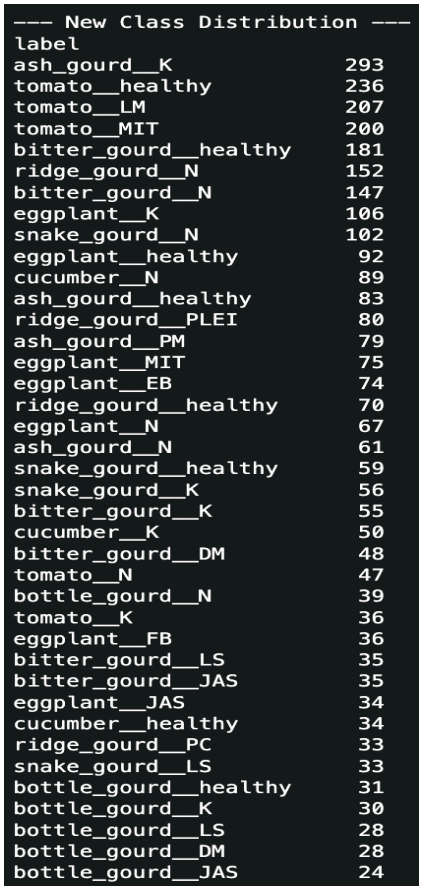


Image 3.2: New Class Distribution of 39 Classes

It's important to note that even after filtering, there is extremely high class imbalance. To solve this we need to try out some methods as mentioned in section 4.2.2.

3.4 Natural Patterns and Insights in the Dataset

3.4.1 Image Distribution

After removing composite labels, the dataset contains 3,165 images across 8 crops. Tomato is the most represented crop with 726 images (23% of the total dataset), followed by Ash Gourd (516), Bitter Gourd (501) and Eggplant (484). Ridge Gourd, Snake Gourd, Bottle Gourd and Cucumber form the tail with 335, 250, 180, and 173 images respectively. This shows a moderate crop level imbalance as we spoke about in previous sections. The top 4 crops together account for nearly 70% of all images, while the bottom 4 crops contribute only 30%. Such distribution suggests that

any model will naturally see more examples of Tomato and Ash Gourd, potentially biasing performance toward these crops if not handled properly [3].

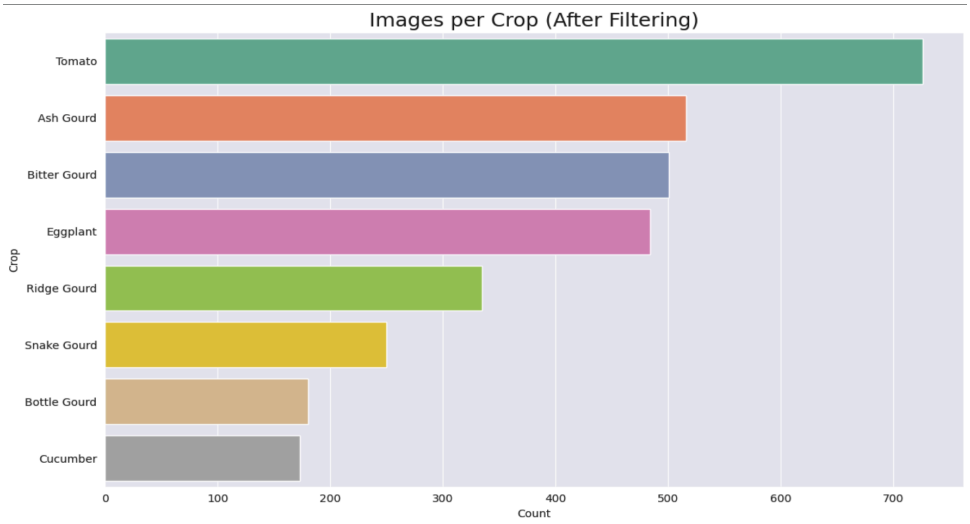


Image 3.4.1: Images Per Crop

3.4.2 Most Frequent Conditions / Diseases

Among all samples “healthy” is the most common class with 786 images, and closely followed by Nitrogen deficiency (N – 704) and Potassium deficiency (K - 626). Mite damage (MIT – 275) and Leaf Miner (LM – 207) are the next significant biotic stresses. The dominance of N and K deficiencies indicates that nutrient related problems are the primary concern in this dataset, far outweighing fungal, bacterial or insect issues. This shifts the problem from general disease classification toward primarily detecting and distinguishing nutrient deficiencies.

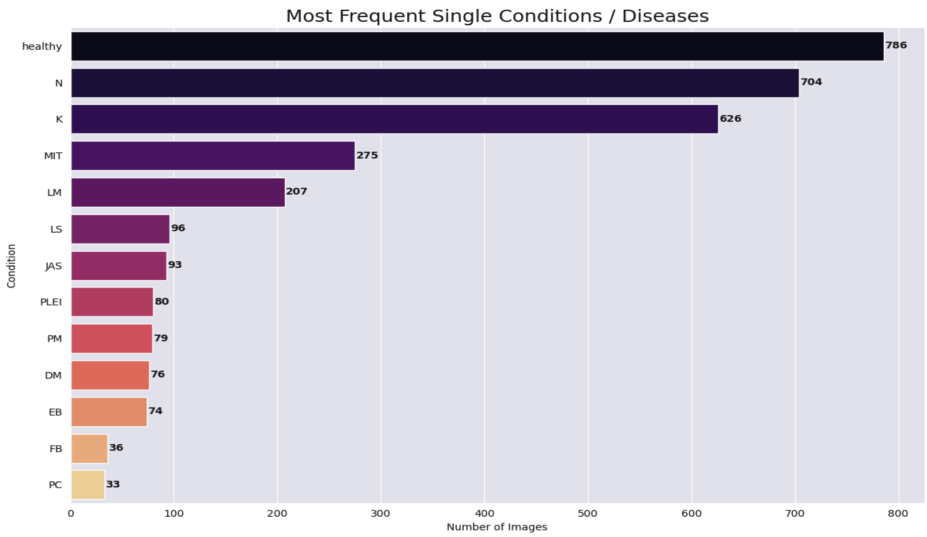


Image 3.4.2: Most Frequent Single Conditions / Diseases

3.4.3 Percentage of Healthy Images per Crop

There is visible variation in healthiness across crops. Bitter Gourd has the highest healthy proportion at 36.1%, followed by Tomato (32.5%). In contrast, Ash Gourd has only 16.1% healthy images. This means that over 83% of its samples show some stress or deficiency. Bottle Gourd (17.2%), Eggplant (19.0%) and Cucumber (19.7%) also lean heavily toward diseased/deficient states. This crop specific health imbalance is a potential issue. Models may learn that "Ash Gourd = probably deficient" by default, which could hurt generalization if not corrected.

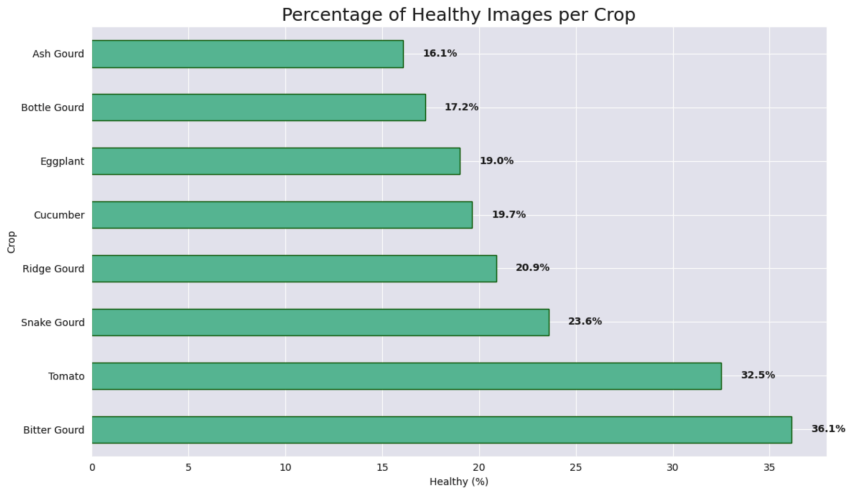


Image 3.4.3: Percentage of Healthy Images Per Crop

3.4.4 Healthy vs Diseased/Deficient Distribution per Crop (Stacked %)

Each crop exhibits a unique disease profile. Ash Gourd is overwhelmingly affected by Potassium (57%) and Nitrogen (12%) deficiencies with almost no biotic stress. Tomato suffers mainly from Leaf Miner (28%) and Mite damage (27%). Eggplant shows high diversity (Early Blight, Fruit Borer, Jasmonic acid response etc). Bitter Gourd and Snake Gourd have relatively balanced healthy to diseased ratios. This confirms that different crops have distinct "signatures" of stress, supporting the inclusion of crop type as an explicit feature.

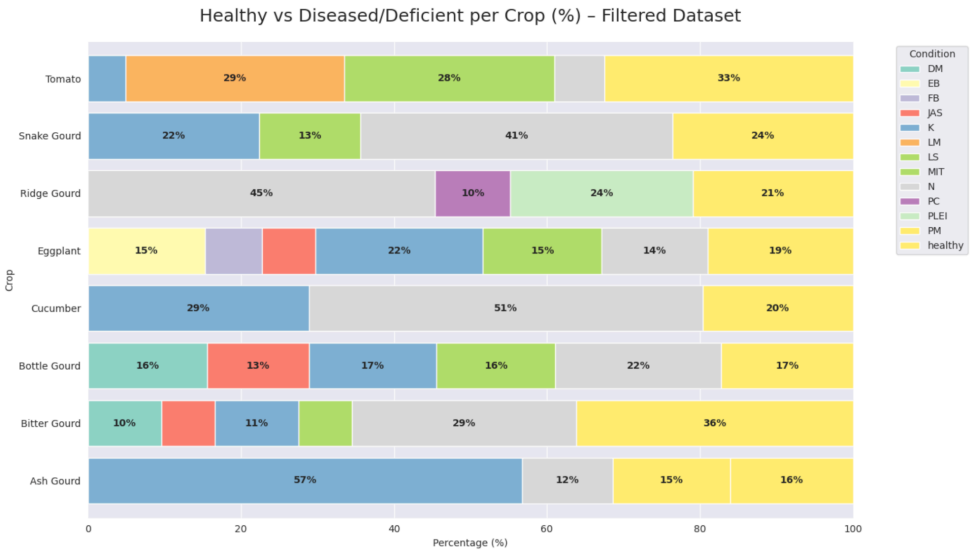


Image 3.4.4.1: Healthy vs Diseased/Deficient Distribution per Crop

Raw counts:

condition	DM	EB	FB	JAS	K	LM	LS	MIT	N	PC	PLEI	PM	healthy
crop													
Ash Gourd	0	0	0	0	293	0	0	0	61	0	0	79	83
Bitter Gourd	48	0	0	35	55	0	35	0	147	0	0	0	181
Bottle Gourd	28	0	0	24	30	0	28	0	39	0	0	0	31
Cucumber	0	0	0	0	50	0	0	0	89	0	0	0	34
Eggplant	0	74	36	34	106	0	0	75	67	0	0	0	92
Ridge Gourd	0	0	0	0	0	0	0	0	152	33	80	0	70
Snake Gourd	0	0	0	0	56	0	33	0	102	0	0	0	59
Tomato	0	0	0	0	36	207	0	200	47	0	0	0	236

Image 3.4.4.2: Crop Wise Count

3.4.5 Top 15 Most Common Single Diseases/Deficiencies (After Filtering)

By removing all composite labels, we obtained cleaner classes. The ranking remains the same as before filtering for the major categories: Nitrogen deficiency (704), Potassium deficiency (626), Mite damage (275), and Leaf Miner (207) dominate. All remaining biotic stresses (Leaf Spot, Powdery Mildew, Downy Mildew, Early Blight, etc.) fall below 100 images each. This filtering step successfully transformed a messy composite label problem into a cleaner single condition classification task dominated by two nutrient deficiencies.

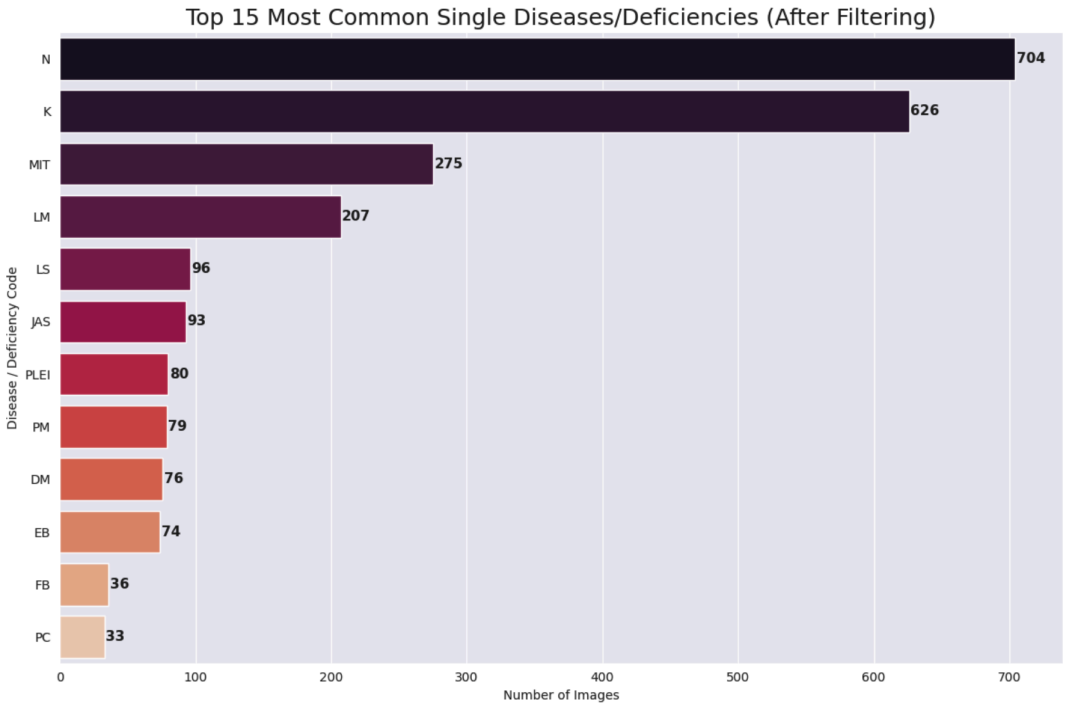


Image 3.4.5: Most Common Single Diseases/Deficiencies

3.4.6 Image Resolution Distribution

To measure this we considered a sample of images from the entire 3000+. The dataset is extremely consistent in resolution, the vast majority of images are exactly 3024 × 3024 pixels (so possibly an iPhone/Android rear camera output). Only three slightly different sizes appear in a sample of 1,000 images, with 3024 × 3024 being the clear mode. This uniformity is excellent, it eliminates variation due to resizing artifacts, ensures all extracted features that we computed are on identical grids. The features we computed are further explained in Section 3.5.3.

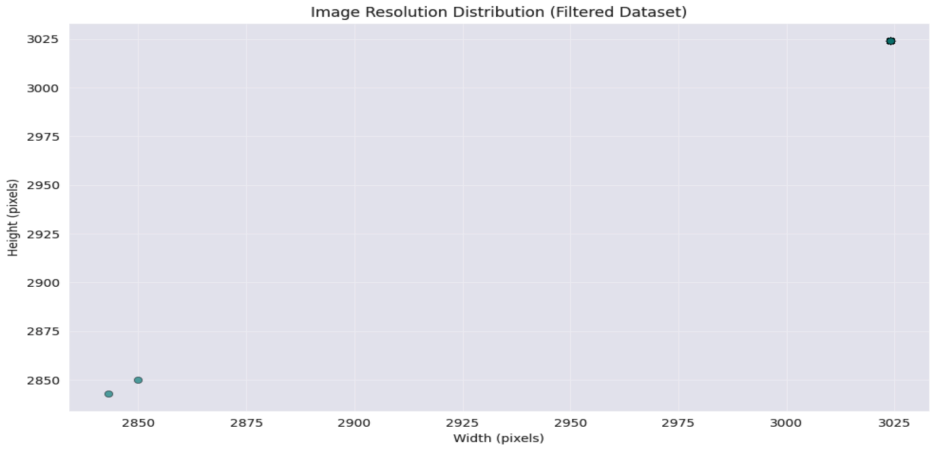


Image 3.4.6.1: Image Resolution Distribution

Image Dimension Statistics (sample):

	Height	Width
count	1000.000000	1000.000000
mean	3023.645000	3023.645000
std	7.935613	7.935613
min	2843.000000	2843.000000
25%	3024.000000	3024.000000
50%	3024.000000	3024.000000
75%	3024.000000	3024.000000
max	3024.000000	3024.000000

Image 3.4.6.2: Statistics For Image Distribution

3.4.7 Class Imbalance Summary (Filtered Dataset)

After filtering, we have 3,165 images across 39 well defined single condition classes. Average 81 images per class. The top 10 classes contain 54% of all data, while the bottom 20 classes hold 24%, giving an imbalance ratio of only 2.2 (this is much better than the original dataset). Only four classes have $\leq$ 30 images and none fall below 24. This filtered version has significantly reduced the long tail problems making it far more suitable for both classical machine learning.

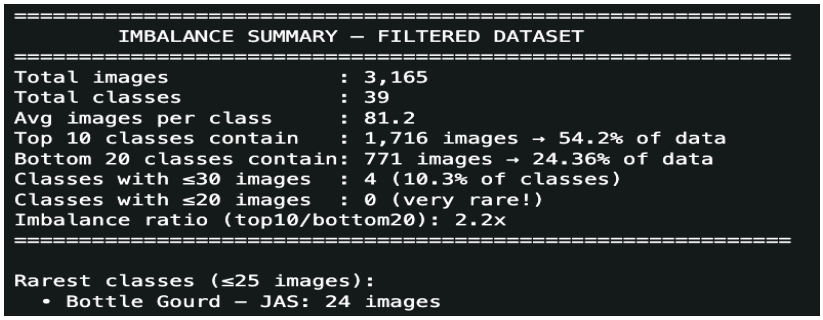


Image 3.4.7: Imbalance Summary

3.5 Pre-Processing the Data

We converted the raw leaf images into a powerful numerical representation using traditional computer vision techniques before feeding them to any machine learning model.

3.5.1 Leaf Segmentation for Background Removal

All images were first resized to 256 × 256 pixels so that every sample has the same dimensions, ensuring consistency in feature extraction. The images were then converted from RGB to HSV color space because the green shade of healthy leaves is much more stable in HSV under varying lighting conditions. Using this HSV representation, we applied simple color thresholds (Hue 25–85, Saturation > 40, Value > 40) to identify green regions. This produced a binary mask, white pixels indicated leaf tissue and black pixels represented the background such as soil, hands, pots or the surrounding environment.

However, this raw mask usually contains small errors, like tiny white specks in the background or tiny black holes inside the leaf . To clean it, we applied morphological opening to remove small white dots outside the leaf, and morphological closing to fill small dark gaps inside the leaf. These operations remove salt and pepper noise and produce a smooth, continuous leaf silhouette. Finally, we used a bitwise AND operation between the original image and this cleaned mask, resulting in two outputs; a segmented color image and a segmented grayscale image containing only the leaf. This step is crucial because removing background clutter ensures that

the texture and color features used by the model truly represent the leaf and are not distorted by irrelevant objects.

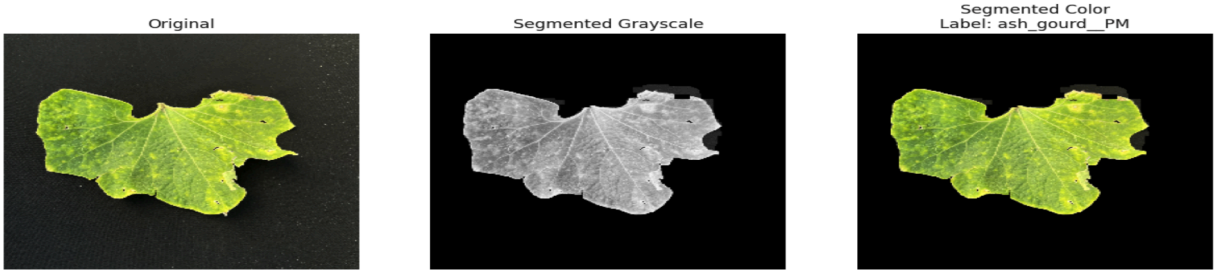


Image 3.5.1: Leaf Segmentation Levels

3.5.3 Color Feature Extraction (HSV Histogram = 288 Features)

Following segmentation, color features were computed using a 3D HSV histogram consisting of 8 bins for Hue, 12 for Saturation and 3 for Value. This resulted in a total of 288 color features per image. These features quantify the distribution of yellowing, whitening, browning or chlorosis across the leaf surface symptoms that frequently occur in nutrient deficiencies or early stage fungal infections. Since color based plant discrimination techniques are widely used in agricultural imaging and many systems rely heavily on color information due to its simplicity and efficiency, our use of HSV histograms is academically justified [8].

3.5.4 Texture Feature Extraction (GLCM + LBP = 46 Features)

Texture features were extracted using two complementary methods. The first being, the Gray Level Co-occurrence Matrix (GLCM) and the latter, Local Binary Patterns (LBP).

The whole idea of GLCM is that it answers the question, "If a pixel has a certain gray shade, what shade does the pixel next to it have? ". For each pixel it looks at another pixel a fixed distance away (like 5 pixels) in multiple directions and records their intensity combinations in a matrix. In our pipeline, GLCM was computed at four orientations (0°, 45°, 90° and 135°) and a distance of five pixels. From these matrices, contrast, dissimilarity, homogeneity, energy and correlation were extracted. Totally, 20 GLCM features that capture niche variations on the leaf surface [8].

LBP answers this question, “Is each neighbor pixel brighter or darker than the center pixel?”. LBP was used to capture very minute texture characteristics such as lesions, white patches, granular deviations and subtle roughness [8]. We generated 26 additional features through this. Together, GLCM and LBP produced 46 texture features.

3.5.5 Shape Feature Extraction (HOG = 1764 Features)

Shape features were extracted using the Histogram of Oriented Gradients (HOG). This basically captures the edge structures, contour details and gradient variations which is generally caused by disease symptoms such as tunneling, curling or boundary roughening. Using 32 x 32 cells, 2 x 2 block structures and 9 gradient orientation bins, our implementation produced 1,764 shape features per image. Previous research emphasizes the importance of combining color and shape descriptors demonstrating improved performance for plant discrimination and weed classification tasks [9].

3.5.6 Final Feature Vector

All the extracted features of 288 color features, 46 texture features, 1,764 shape features and 1 encoded plant type value were combined into a final feature vector consisting of 2,099 total features per image. The final pre-processed dataset consisted of an input feature matrix X of shape (3165, 2099) and a corresponding target vector y of length 3165.

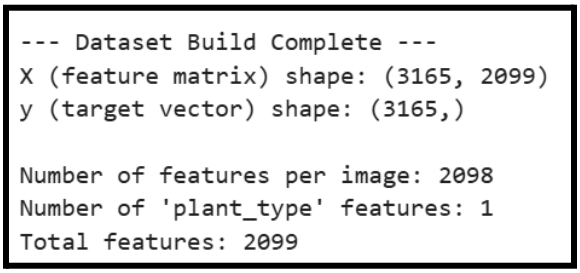


Image 3.5.6: Final Feature Matrix Shape After Pre-Processing

3.6 EDA Based on Processed Features

After constructing the 2,099D feature vector for each image, we did a thorough EDA to evaluate the feature usefulness, detect redundancy, understand the dimensionality and inspect separability among classes.

3.6.1 Ranking Features Using Mutual Information

Mutual Information (MI) is a measure of how much information one random variable contains about another random variable. It quantifies the dependency between two variables. [10]. The highest MI value observed was 1.9567 and the median MI across all features was 0.18717. This indicates that a substantial portion of the feature space carries meaningful signals. Approximately 1,980 features exhibited MI values greater than 0.01, demonstrating that the majority of features contribute in some way to class discrimination. The MI analysis also revealed that the plant type encoding had the highest single feature MI which was followed by several HOG, GLCM and LBP features. Hence why we claimed the process to be academically relevant. As highlighted by Hamuda et al., structural and texture based characteristics often provide stronger discriminatory capability than raw color information when analyzing plant images under variable outdoor lighting [8].

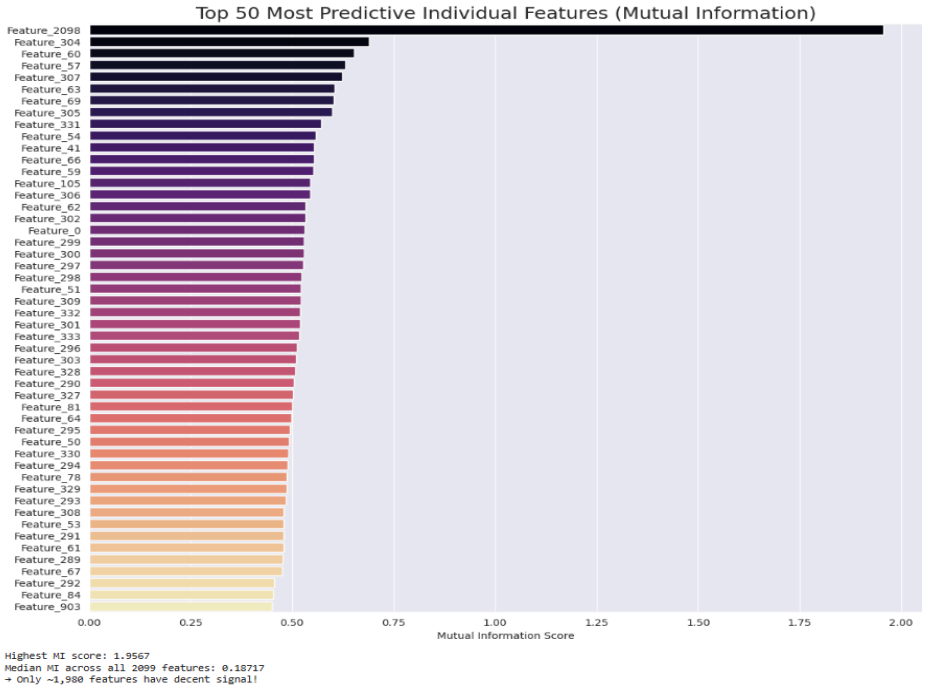


Image 3.6.1: Top 50 Most Predictive Individual Features (Mutual Information)

3.6.2 Predictive Power by Feature Group

To understand the relative importance of feature categories the MI values were aggregated for each feature group. Shape features (HOG) contributed a total of 333.0 MI units, significantly outweighing the contributions of color features (31.8), texture features (21.3) and the plant type feature (2.0). This means that more than 85% of the total predictive power comes from shape descriptors. This shows the critical role of structural and gradient based patterns in disease identification.

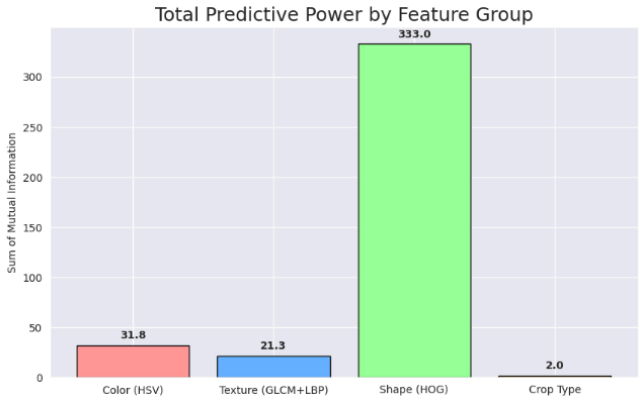


Image 3.6.2: Total Predictive Power by Feature Group

3.6.3 Correlation Analysis of Top Features

A correlation matrix of the top 36 predictive features was computed to detect redundancy. As we expected, many HOG features were highly correlated with one another due to overlapping during the gradient computation. Texture features exhibited moderate correlation and color features remained largely independent. The maximum pairwise correlation observed among top features was 1.000. Hence, it confirmed the presence of perfectly redundant descriptors. These findings validated the need for dimensionality reduction to avoid multi level collinearity.

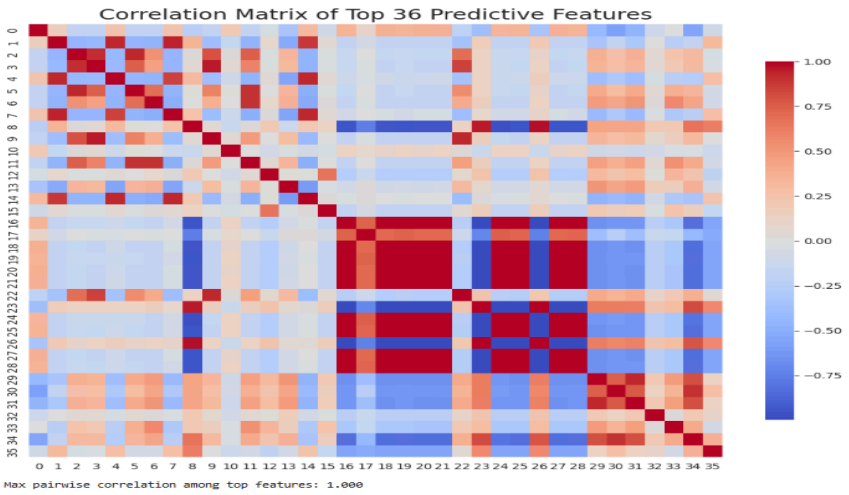


Image 3.6.3: Correlation Matrix of Top 36 Predictive Features

3.6.4 PCA Screen Plot

Principal Component Analysis (PCA) was applied to the normalized feature matrix to determine its effective dimensionality. As stated in the previous result, most of the feature space highlighted redundancy, hence, to get a clear understanding of the real number of features we would actually need, we conducted PCA. Despite the original 2,099 features, only 464 principal components were required to explain 95% of the total variance, demonstrating that approximately 78% of the original feature space was redundant. This reduction from 2,099 to 464 components corresponds to maintaining only 22.1% of the original dimensionality while still retaining nearly all meaningful information.

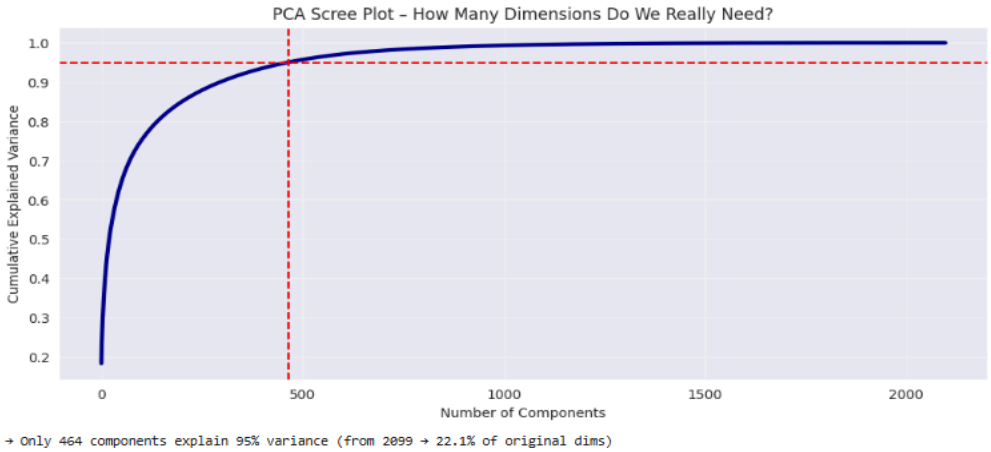


Image 3.6.4: PCA Scree Plot Showing Variance Explained by Components

3.6.5 t-SNE Visualization - Class Clustering

We created a t-SNE visualization using only the top 200 most predictive features identified by MI. The result revealed clear clustering behavior for multiple classes. Diseases with strong structural signatures such as LM, MD and PM formed tightly grouped clusters, while nutrient deficiency categories demonstrated slight overlap due to their visually similar symptoms. These moderately well defined clusters further confirm that the handcrafted features did manage to capture disease specific biological patterns. It also revealed to us that the dataset is inherently separable in high dimensional space.

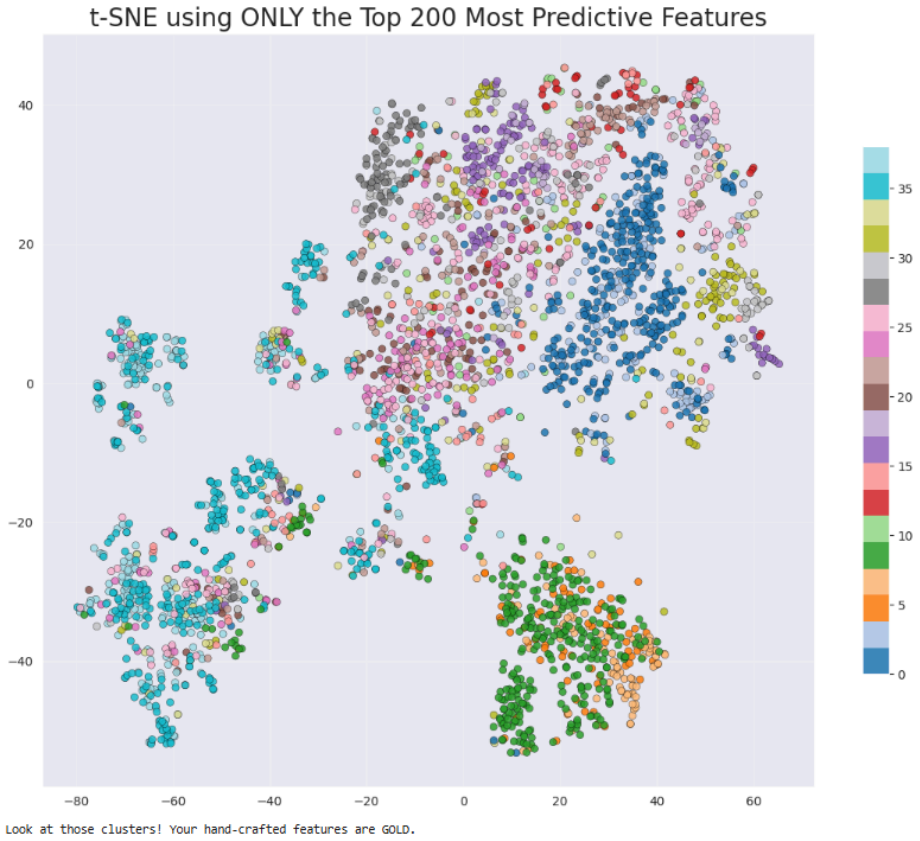


Image 3.6.5: t-SNE Visualization Using the Top 200 Most Predictive Features

4. Model Training

Since the goal of this project is to predict both the plant type and the nutrient deficiency/disease status of a leaf image, the problem naturally fits into a multi-label classification setup. Classical

machine learning models were chosen because they are effective when working with structured, tabular representations of image features, while also providing strong interpretability and lower computational cost compared to deep learning. These models learn class boundaries directly from extracted features, making them suitable for relatively small datasets and for settings where transparency and explainability are essential.

To evaluate model performance consistently across both outputs, a custom multi-output evaluation function was implemented. This function computes accuracy, macro-averaged and weighted-averaged precision, recall, and F1-score for each target, which captures both overall performance and per-class behavior. The macro F1-score was emphasized for the Status target because the dataset is highly imbalanced across deficiency labels, and macro averaging ensures that minority classes are not overshadowed by high-frequency labels. The accuracy metric used provides a general measure of how often the model predicts correctly, making it suitable for understanding high-level performance on plant type, where class distribution is fairly balanced.

However, deficiency labels in the Status target are highly imbalanced, with certain nutrient categories having very few samples. To address this, macro-averaged precision, recall, and F1-score were used because they treat each class equally and highlight how well the model performs on minority classes. Weighted averages were also included to account for class frequency and give a more realistic picture of performance on the imbalanced dataset. Using this combination of metrics ensures that the evaluation reflects both overall accuracy and per-class reliability, making the results more interpretable and fair for all label categories.

Both training and testing metrics were examined to detect overfitting and to understand hidden bottlenecks such as class imbalance, limited samples, and overlapping feature distributions.

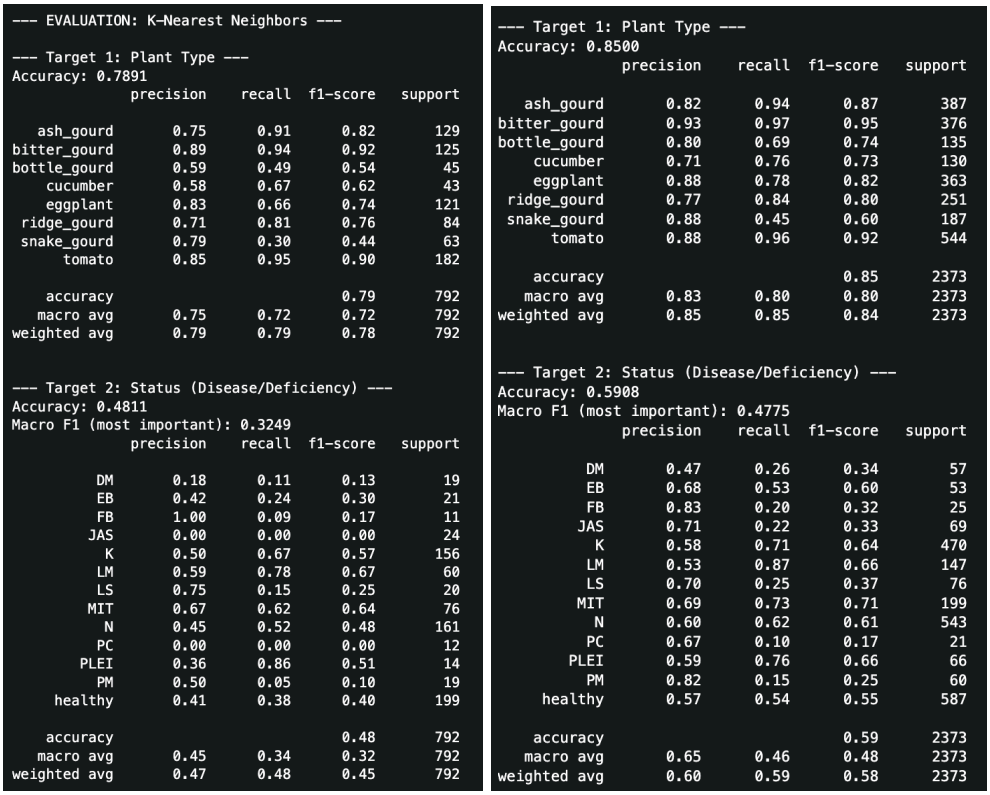
4.1 Stage 1 Models

4.1.1 K-Nearest Neighbors (KNN) - Without PCA

The baseline KNN model (k = 7) achieved a test accuracy of 78.9% for plant type prediction, with strong performance on well-represented classes such as tomato, bitter gourd, and ash gourd.

However, minority classes like bottle gourd and snake gourd showed noticeably lower F1-scores (0.54 and 0.44). This reflects KNN's sensitivity to local data density when a class has fewer neighboring points, the model defaults to predicting majority-class labels. For the Status target, performance dropped significantly; the accuracy was 48.1%, and macro F1-score was only 0.3249, mainly due to severe label imbalance and overlapping feature clusters across deficiency types.

Training performance further revealed the model's limitations: although plant type accuracy rose to 85% on the training set, the Status macro F1 only reached 0.4775, indicating that even with full data access, KNN struggles to separate visually similar nutrient deficiencies. These results highlight that while KNN handles plant type well, it is constrained by minority-class sparsity, high dimensionality, and noisy features.



```
--- EVALUATION: K-Nearest Neighbors ---

--- Target 1: Plant Type ---
Accuracy: 0.7891
              precision    recall  f1-score   support

   ash_gourd       0.75      0.91      0.82       129
bitter_gourd       0.89      0.94      0.92       125
bottle_gourd       0.59      0.49      0.54        45
    cucumber       0.58      0.67      0.62        43
    eggplant       0.83      0.66      0.74       121
 ridge_gourd       0.71      0.81      0.76        84
 snake_gourd       0.79      0.30      0.44        63
      tomato       0.85      0.95      0.90       182

    accuracy                           0.79       792
   macro avg       0.75      0.72      0.72       792
weighted avg       0.79      0.79      0.78       792


--- Target 2: Status (Disease/Deficiency) ---
Accuracy: 0.4811
Macro F1 (most important): 0.3249
              precision    recall  f1-score   support

          DM       0.18      0.11      0.13        19
          EB       0.42      0.24      0.30        21
          FB       1.00      0.09      0.17        11
         JAS       0.00      0.00      0.00        24
           K       0.50      0.67      0.57       156
          LM       0.59      0.78      0.67        60
          LS       0.75      0.15      0.25        20
         MIT       0.67      0.62      0.64        76
           N       0.45      0.52      0.48       161
          PC       0.00      0.00      0.00        12
        PLEI       0.36      0.86      0.51        14
          PM       0.50      0.05      0.10        19
     healthy       0.41      0.38      0.40       199

    accuracy                           0.48       792
   macro avg       0.45      0.34      0.32       792
weighted avg       0.47      0.48      0.45       792
```

```
--- Target 1: Plant Type ---
Accuracy: 0.8500
              precision    recall  f1-score   support

   ash_gourd       0.82      0.94      0.87       387
bitter_gourd       0.93      0.97      0.95       376
bottle_gourd       0.80      0.69      0.74       135
    cucumber       0.71      0.76      0.73       130
    eggplant       0.88      0.78      0.82       363
 ridge_gourd       0.77      0.84      0.80       251
 snake_gourd       0.88      0.45      0.60       187
      tomato       0.88      0.96      0.92       544

    accuracy                           0.85      2373
   macro avg       0.83      0.80      0.80      2373
weighted avg       0.85      0.85      0.84      2373


--- Target 2: Status (Disease/Deficiency) ---
Accuracy: 0.5908
Macro F1 (most important): 0.4775
              precision    recall  f1-score   support

          DM       0.47      0.26      0.34        57
          EB       0.68      0.53      0.60        53
          FB       0.83      0.20      0.32        25
         JAS       0.71      0.22      0.33        69
           K       0.58      0.71      0.64       470
          LM       0.53      0.87      0.66       147
          LS       0.70      0.25      0.37        76
         MIT       0.69      0.73      0.71       199
           N       0.60      0.62      0.61       543
          PC       0.67      0.10      0.17        21
        PLEI       0.59      0.76      0.66        66
          PM       0.82      0.15      0.25        60
     healthy       0.57      0.54      0.55       587

    accuracy                           0.59      2373
   macro avg       0.65      0.46      0.48      2373
weighted avg       0.60      0.59      0.58      2373
```

Image 4.1.1: KNN Results for Test v/s Train

4.1.2 K-Nearest Neighbors (KNN) - With PCA

To address high-dimensional feature noise, PCA was applied, reducing 2,099 features to 464 components while preserving 95% variance. This lowered computational burden and improved neighborhood structure by eliminating redundant dimensions. However, PCA produced only modest improvements. Plant type accuracy remained in a similar range, and Status prediction continued to suffer from low macro F1-scores. This indicates that deficiencies in the dataset, particularly extreme imbalance and overlapping visual features limit KNN's ability to form distinct clusters, even after dimensionality reduction. PCA reduced noise but did not fundamentally alter the distribution issues that affect KNN's neighborhood-based predictions.

```
--- EVALUATION: KNN + PCA ---

--- Target 1: Plant Type ---
Accuracy: 0.7904
              precision    recall  f1-score   support

   ash_gourd       0.76      0.90      0.82       129
bitter_gourd       0.88      0.95      0.92       125
bottle_gourd       0.65      0.49      0.56        45
    cucumber       0.61      0.70      0.65        43
    eggplant       0.83      0.68      0.75       121
 ridge_gourd       0.69      0.80      0.74        84
 snake_gourd       0.76      0.30      0.43        63
      tomato       0.85      0.94      0.90       182

    accuracy                           0.79       792
   macro avg       0.75      0.72      0.72       792
weighted avg       0.79      0.79      0.78       792


--- Target 2: Status (Disease/Deficiency) ---
Accuracy: 0.4735
Macro F1 (most important): 0.3209
              precision    recall  f1-score   support

          DM       0.22      0.11      0.14        19
          EB       0.36      0.24      0.29        21
          FB       1.00      0.09      0.17        11
         JAS       0.00      0.00      0.00        24
           K       0.50      0.66      0.57       156
          LM       0.57      0.77      0.65        60
          LS       0.75      0.15      0.25        20
         MIT       0.67      0.62      0.64        76
           N       0.45      0.50      0.47       161
          PC       0.00      0.00      0.00        12
        PLEI       0.36      0.86      0.51        14
          PM       0.50      0.05      0.10        19
     healthy       0.40      0.37      0.39       199

    accuracy                           0.47       792
   macro avg       0.44      0.34      0.32       792
weighted avg       0.46      0.47      0.45       792
```

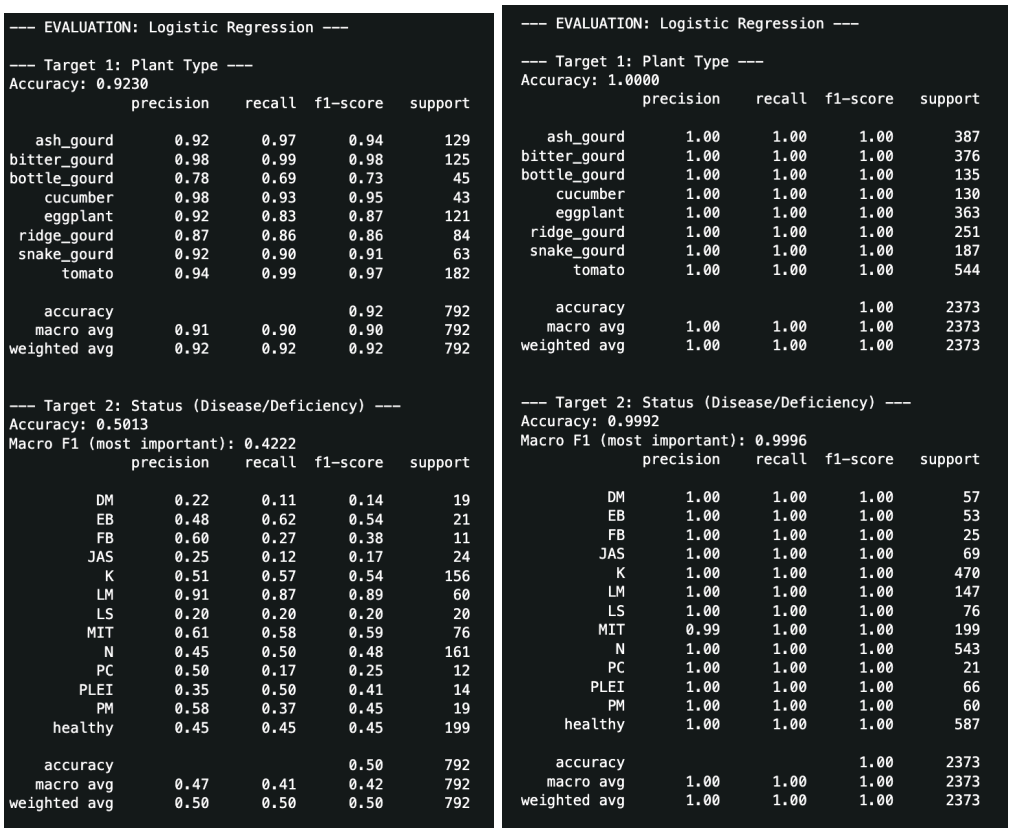
```
--- EVALUATION: K-Nearest Neighbors ---

--- Target 1: Plant Type ---
Accuracy: 0.8500
              precision    recall  f1-score   support

   ash_gourd       0.82      0.94      0.88       387
bitter_gourd       0.93      0.98      0.95       376
bottle_gourd       0.78      0.67      0.72       135
    cucumber       0.74      0.77      0.75       130
    eggplant       0.88      0.76      0.82       363
 ridge_gourd       0.77      0.84      0.80       251
 snake_gourd       0.87      0.45      0.59       187
      tomato       0.87      0.96      0.92       544

    accuracy                           0.85      2373
   macro avg       0.83      0.80      0.80      2373
weighted avg       0.85      0.85      0.84      2373


--- Target 2: Status (Disease/Deficiency) ---
Accuracy: 0.5891
Macro F1 (most important): 0.4696
              precision    recall  f1-score   support

          DM       0.42      0.23      0.30        57
          EB       0.70      0.57      0.62        53
          FB       0.83      0.20      0.32        25
         JAS       0.71      0.22      0.33        69
           K       0.58      0.71      0.64       470
          LM       0.54      0.88      0.67       147
          LS       0.62      0.24      0.34        76
         MIT       0.70      0.72      0.71       199
           N       0.60      0.62      0.61       543
          PC       0.33      0.05      0.08        21
        PLEI       0.57      0.76      0.65        66
          PM       0.83      0.17      0.28        60
     healthy       0.56      0.53      0.55       587

    accuracy                           0.59      2373
   macro avg       0.62      0.45      0.47      2373
weighted avg       0.60      0.59      0.57      2373
```

Image 4.1.2: KNN Results for Test v/s Train (PCA)

4.1.3 Logistic Regression

Logistic Regression performed surprisingly well for plant type classification, achieving a test accuracy of 92.3%, with high precision and recall across nearly all classes. This suggests that plant type features are linearly separable in the transformed feature space. However, the Status target again showed weaker performance, with 50.1% accuracy and a macro F1-score of 0.4222, demonstrating that deficiencies are not linearly separable. The model drastically overfitted the training data, where both plant type and Status achieved almost 100% accuracy, indicating that

the classifier memorized patterns rather than learning generalizable boundaries. This overfitting reflects the fact that while Logistic Regression benefits from standardized features, it struggles with complex multi-class imbalance and non-linear relationships present in deficiency labels.



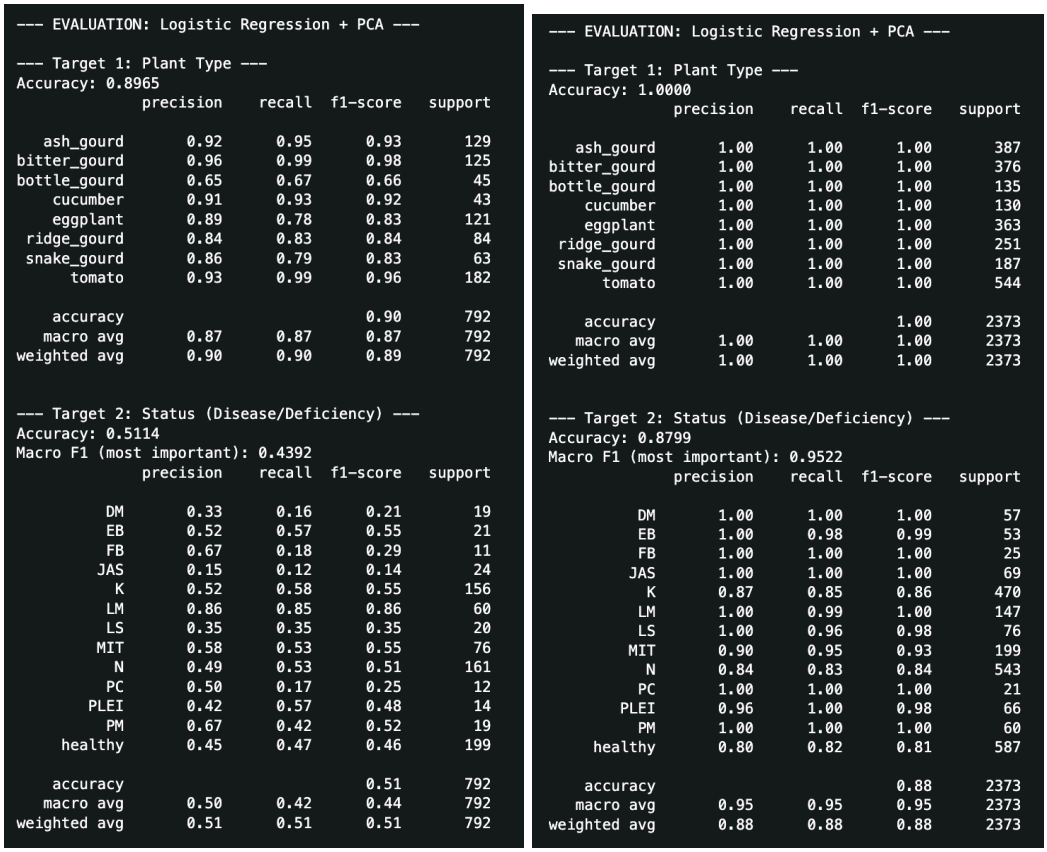
```
--- EVALUATION: Logistic Regression ---

--- Target 1: Plant Type ---
Accuracy: 0.9230
              precision    recall  f1-score   support

   ash_gourd       0.92      0.97      0.94       129
bitter_gourd       0.98      0.99      0.98       125
bottle_gourd       0.78      0.69      0.73        45
    cucumber       0.98      0.93      0.95        43
    eggplant       0.92      0.83      0.87       121
 ridge_gourd       0.87      0.86      0.86        84
 snake_gourd       0.92      0.90      0.91        63
      tomato       0.94      0.99      0.97       182

    accuracy                           0.92       792
   macro avg       0.91      0.90      0.90       792
weighted avg       0.92      0.92      0.92       792


--- Target 2: Status (Disease/Deficiency) ---
Accuracy: 0.5013
Macro F1 (most important): 0.4222
              precision    recall  f1-score   support

          DM       0.22      0.11      0.14        19
          EB       0.48      0.62      0.54        21
          FB       0.60      0.27      0.38        11
         JAS       0.25      0.12      0.17        24
           K       0.51      0.57      0.54       156
          LM       0.91      0.87      0.89        60
          LS       0.20      0.20      0.20        20
         MIT       0.61      0.58      0.59        76
           N       0.45      0.50      0.48       161
          PC       0.50      0.17      0.25        12
        PLEI       0.35      0.50      0.41        14
          PM       0.58      0.37      0.45        19
     healthy       0.45      0.45      0.45       199

    accuracy                           0.50       792
   macro avg       0.47      0.41      0.42       792
weighted avg       0.50      0.50      0.50       792
```

```
--- EVALUATION: Logistic Regression ---

--- Target 1: Plant Type ---
Accuracy: 1.0000
              precision    recall  f1-score   support

   ash_gourd       1.00      1.00      1.00       387
bitter_gourd       1.00      1.00      1.00       376
bottle_gourd       1.00      1.00      1.00       135
    cucumber       1.00      1.00      1.00       130
    eggplant       1.00      1.00      1.00       363
 ridge_gourd       1.00      1.00      1.00       251
 snake_gourd       1.00      1.00      1.00       187
      tomato       1.00      1.00      1.00       544

    accuracy                           1.00      2373
   macro avg       1.00      1.00      1.00      2373
weighted avg       1.00      1.00      1.00      2373


--- Target 2: Status (Disease/Deficiency) ---
Accuracy: 0.9992
Macro F1 (most important): 0.9996
              precision    recall  f1-score   support

          DM       1.00      1.00      1.00        57
          EB       1.00      1.00      1.00        53
          FB       1.00      1.00      1.00        25
         JAS       1.00      1.00      1.00        69
           K       1.00      1.00      1.00       470
          LM       1.00      1.00      1.00       147
          LS       1.00      1.00      1.00        76
         MIT       0.99      1.00      1.00       199
           N       1.00      1.00      1.00       543
          PC       1.00      1.00      1.00        21
        PLEI       1.00      1.00      1.00        66
          PM       1.00      1.00      1.00        60
     healthy       1.00      1.00      1.00       587

    accuracy                           1.00      2373
   macro avg       1.00      1.00      1.00      2373
weighted avg       1.00      1.00      1.00      2373
```

Image 4.1.3: Test vs train results Logistic regression

4.1.4 Logistic Regression (With PCA)

Logistic Regression with PCA performed similarly to the non-PCA version for plant type, confirming that the main structural differences among plants are preserved within the top principal components. The classifier maintained strong accuracy with reduced dimensionality while benefiting from significantly faster training. However, for the deficiency Status target, the model still struggled, showing only a modest improvement in macro-averaged metrics. This indicates that although PCA removes noise, the deficiency categories still exhibit substantial overlap in the reduced feature space, making them difficult to linearly separate.



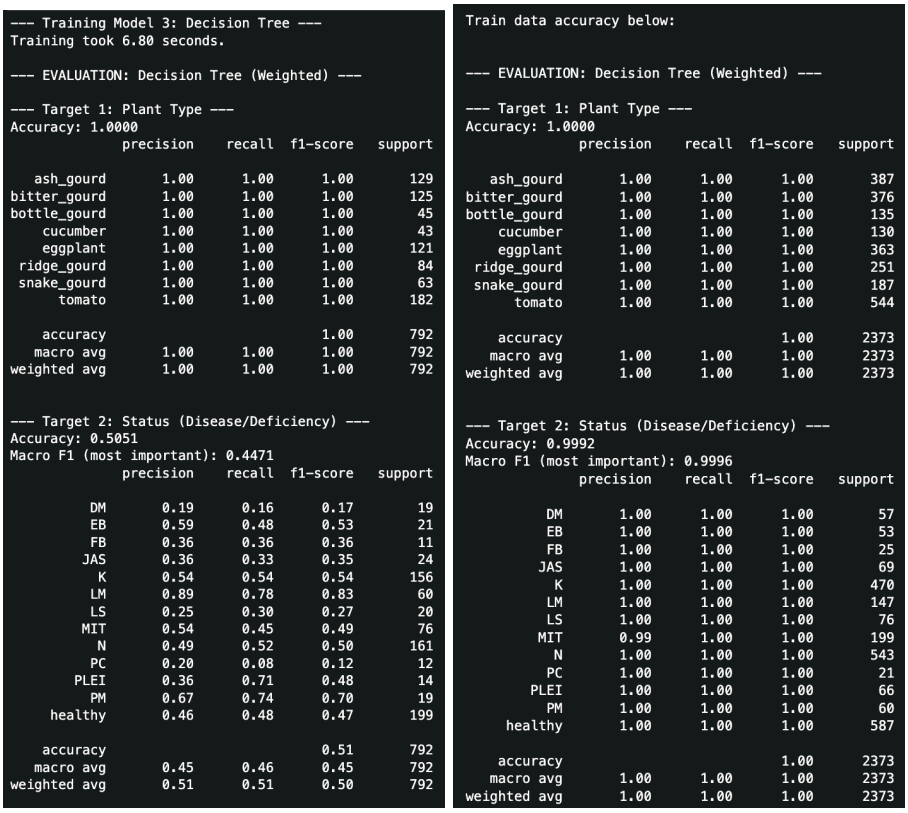
```
--- EVALUATION: Logistic Regression + PCA ---

--- Target 1: Plant Type ---
Accuracy: 0.8965
              precision    recall  f1-score   support

   ash_gourd       0.92      0.95      0.93       129
bitter_gourd       0.96      0.99      0.98       125
bottle_gourd       0.65      0.67      0.66        45
    cucumber       0.91      0.93      0.92        43
    eggplant       0.89      0.78      0.83       121
 ridge_gourd       0.84      0.83      0.84        84
 snake_gourd       0.86      0.79      0.83        63
      tomato       0.93      0.99      0.96       182

    accuracy                           0.90       792
   macro avg       0.87      0.87      0.87       792
weighted avg       0.90      0.90      0.89       792


--- Target 2: Status (Disease/Deficiency) ---
Accuracy: 0.5114
Macro F1 (most important): 0.4392
              precision    recall  f1-score   support

          DM       0.33      0.16      0.21        19
          EB       0.52      0.57      0.55        21
          FB       0.67      0.18      0.29        11
         JAS       0.15      0.12      0.14        24
           K       0.52      0.58      0.55       156
          LM       0.86      0.85      0.86        60
          LS       0.35      0.35      0.35        20
         MIT       0.58      0.53      0.55        76
           N       0.49      0.53      0.51       161
          PC       0.50      0.17      0.25        12
        PLEI       0.42      0.57      0.48        14
          PM       0.67      0.42      0.52        19
     healthy       0.45      0.47      0.46       199

    accuracy                           0.51       792
   macro avg       0.50      0.42      0.44       792
weighted avg       0.51      0.51      0.51       792
```

```
--- EVALUATION: Logistic Regression + PCA ---

--- Target 1: Plant Type ---
Accuracy: 1.0000
              precision    recall  f1-score   support

   ash_gourd       1.00      1.00      1.00       387
bitter_gourd       1.00      1.00      1.00       376
bottle_gourd       1.00      1.00      1.00       135
    cucumber       1.00      1.00      1.00       130
    eggplant       1.00      1.00      1.00       363
 ridge_gourd       1.00      1.00      1.00       251
 snake_gourd       1.00      1.00      1.00       187
      tomato       1.00      1.00      1.00       544

    accuracy                           1.00      2373
   macro avg       1.00      1.00      1.00      2373
weighted avg       1.00      1.00      1.00      2373


--- Target 2: Status (Disease/Deficiency) ---
Accuracy: 0.8799
Macro F1 (most important): 0.9522
              precision    recall  f1-score   support

          DM       1.00      1.00      1.00        57
          EB       1.00      0.98      0.99        53
          FB       1.00      1.00      1.00        25
         JAS       1.00      1.00      1.00        69
           K       0.87      0.85      0.86       470
          LM       1.00      0.99      1.00       147
          LS       1.00      0.96      0.98        76
         MIT       0.90      0.95      0.93       199
           N       0.84      0.83      0.84       543
          PC       1.00      1.00      1.00        21
        PLEI       0.96      1.00      0.98        66
          PM       1.00      1.00      1.00        60
     healthy       0.80      0.82      0.81       587

    accuracy                           0.88      2373
   macro avg       0.95      0.95      0.95      2373
weighted avg       0.88      0.88      0.88      2373
```

Image 4.1.4: Logistical Regression Results for Test v/s Train (PCA)

4.1.5 Decision Tree (Weighted)

The Decision Tree model achieved perfect accuracy (100%) on the training set for both targets, which is expected given its non-parametric nature and ability to form arbitrarily deep partitions. However, real generalization performance on the test set showed clear overfitting. Plant type accuracy remained high at 100%, but Status accuracy was only 50.5% with a macro F1 of 0.4471. Though class weights helped balance splits, the tree still captured dataset noise and created many deep, class-specific partitions. Minority deficiency classes such as PC and LS remained particularly difficult, receiving very low recall and F1-scores. This highlights the instability of standalone Decision Trees when trained on high-dimensional, imbalanced multi-output problems.



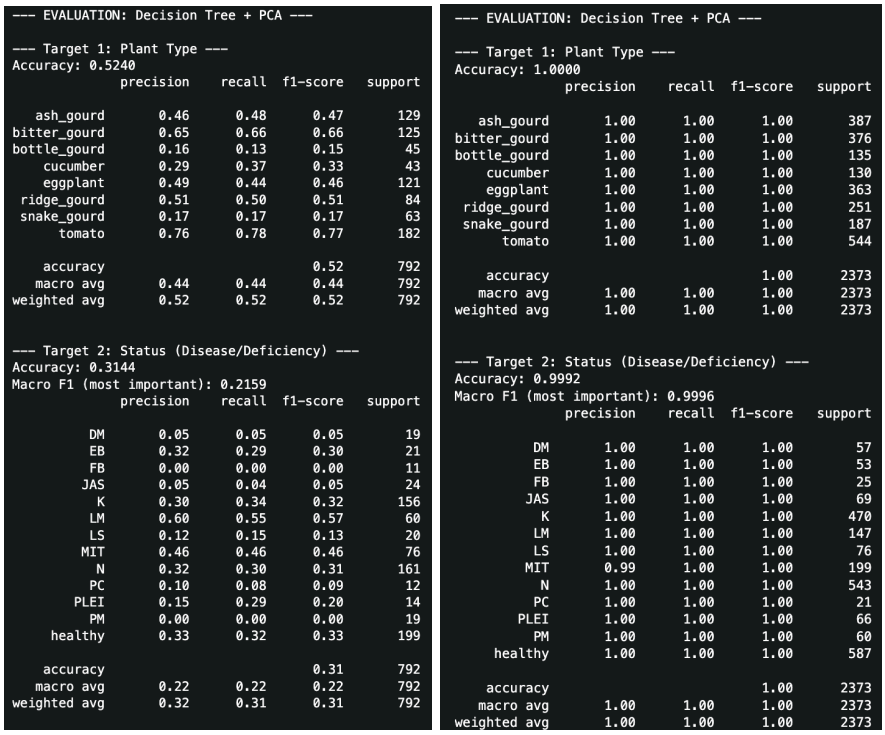
```
--- Training Model 3: Decision Tree ---
Training took 6.80 seconds.

--- EVALUATION: Decision Tree (Weighted) ---

--- Target 1: Plant Type ---
Accuracy: 1.0000
              precision    recall  f1-score   support

   ash_gourd       1.00      1.00      1.00       129
bitter_gourd       1.00      1.00      1.00       125
bottle_gourd       1.00      1.00      1.00        45
    cucumber       1.00      1.00      1.00        43
    eggplant       1.00      1.00      1.00       121
 ridge_gourd       1.00      1.00      1.00        84
 snake_gourd       1.00      1.00      1.00        63
      tomato       1.00      1.00      1.00       182

    accuracy                           1.00       792
   macro avg       1.00      1.00      1.00       792
weighted avg       1.00      1.00      1.00       792


--- Target 2: Status (Disease/Deficiency) ---
Accuracy: 0.5051
Macro F1 (most important): 0.4471
              precision    recall  f1-score   support

          DM       0.19      0.16      0.17        19
          EB       0.59      0.48      0.53        21
          FB       0.36      0.36      0.36        11
         JAS       0.36      0.33      0.35        24
           K       0.54      0.54      0.54       156
          LM       0.89      0.78      0.83        60
          LS       0.25      0.30      0.27        20
         MIT       0.54      0.45      0.49        76
           N       0.49      0.52      0.50       161
          PC       0.20      0.08      0.12        12
        PLEI       0.36      0.71      0.48        14
          PM       0.67      0.74      0.70        19
     healthy       0.46      0.48      0.47       199

    accuracy                           0.51       792
   macro avg       0.45      0.46      0.45       792
weighted avg       0.51      0.51      0.50       792
```

```
Train data accuracy below:


--- EVALUATION: Decision Tree (Weighted) ---

--- Target 1: Plant Type ---
Accuracy: 1.0000
              precision    recall  f1-score   support

   ash_gourd       1.00      1.00      1.00       387
bitter_gourd       1.00      1.00      1.00       376
bottle_gourd       1.00      1.00      1.00       135
    cucumber       1.00      1.00      1.00       130
    eggplant       1.00      1.00      1.00       363
 ridge_gourd       1.00      1.00      1.00       251
 snake_gourd       1.00      1.00      1.00       187
      tomato       1.00      1.00      1.00       544

    accuracy                           1.00      2373
   macro avg       1.00      1.00      1.00      2373
weighted avg       1.00      1.00      1.00      2373


--- Target 2: Status (Disease/Deficiency) ---
Accuracy: 0.9992
Macro F1 (most important): 0.9996
              precision    recall  f1-score   support

          DM       1.00      1.00      1.00        57
          EB       1.00      1.00      1.00        53
          FB       1.00      1.00      1.00        25
         JAS       1.00      1.00      1.00        69
           K       1.00      1.00      1.00       470
          LM       1.00      1.00      1.00       147
          LS       1.00      1.00      1.00        76
         MIT       0.99      1.00      1.00       199
           N       1.00      1.00      1.00       543
          PC       1.00      1.00      1.00        21
        PLEI       1.00      1.00      1.00        66
          PM       1.00      1.00      1.00        60
     healthy       1.00      1.00      1.00       587

    accuracy                           1.00      2373
   macro avg       1.00      1.00      1.00      2373
weighted avg       1.00      1.00      1.00      2373
```

Image 4.1.5: Test vs train results Decision Tree

4.1.6 Decision Tree (With PCA)

After PCA, the Decision Tree model became more stable and less prone to memorizing noise compared to its non-PCA version, which achieved nearly perfect training accuracy. Because PCA removes many irrelevant variations, the decision boundaries become smoother and less overfitted. However, while plant type remained relatively easy for the tree to classify, deficiency prediction saw only small improvements. The reduced space does not fully resolve the inherent ambiguity and imbalance among deficiency classes. Thus, although PCA helped regularize the tree, the model remained fundamentally limited by its tendency to create axis-aligned splits in a transformed, continuous feature space.



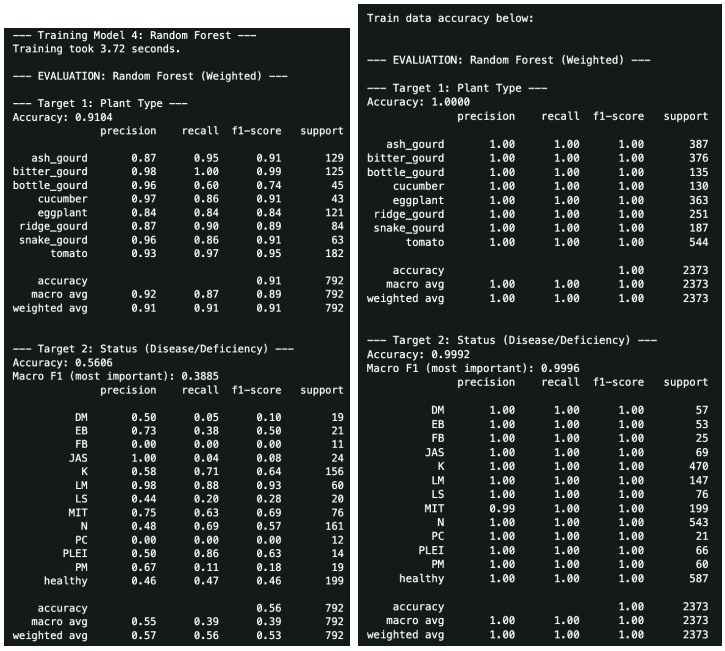
```
--- EVALUATION: Decision Tree + PCA ---

--- Target 1: Plant Type ---
Accuracy: 0.5240
              precision    recall  f1-score   support

   ash_gourd       0.46      0.48      0.47       129
bitter_gourd       0.65      0.66      0.66       125
bottle_gourd       0.16      0.13      0.15        45
    cucumber       0.29      0.37      0.33        43
    eggplant       0.49      0.44      0.46       121
 ridge_gourd       0.51      0.50      0.51        84
 snake_gourd       0.17      0.17      0.17        63
      tomato       0.76      0.78      0.77       182

    accuracy                           0.52       792
   macro avg       0.44      0.44      0.44       792
weighted avg       0.52      0.52      0.52       792


--- Target 2: Status (Disease/Deficiency) ---
Accuracy: 0.3144
Macro F1 (most important): 0.2159
              precision    recall  f1-score   support

          DM       0.05      0.05      0.05        19
          EB       0.32      0.29      0.30        21
          FB       0.00      0.00      0.00        11
         JAS       0.05      0.04      0.05        24
           K       0.30      0.34      0.32       156
          LM       0.60      0.55      0.57        60
          LS       0.12      0.15      0.13        20
         MIT       0.46      0.46      0.46        76
           N       0.32      0.30      0.31       161
          PC       0.10      0.08      0.09        12
        PLEI       0.15      0.29      0.20        14
          PM       0.00      0.00      0.00        19
     healthy       0.33      0.32      0.33       199

    accuracy                           0.31       792
   macro avg       0.22      0.22      0.22       792
weighted avg       0.32      0.31      0.31       792
```

```
--- EVALUATION: Decision Tree + PCA ---

--- Target 1: Plant Type ---
Accuracy: 1.0000
              precision    recall  f1-score   support

   ash_gourd       1.00      1.00      1.00       387
bitter_gourd       1.00      1.00      1.00       376
bottle_gourd       1.00      1.00      1.00       135
    cucumber       1.00      1.00      1.00       130
    eggplant       1.00      1.00      1.00       363
 ridge_gourd       1.00      1.00      1.00       251
 snake_gourd       1.00      1.00      1.00       187
      tomato       1.00      1.00      1.00       544

    accuracy                           1.00      2373
   macro avg       1.00      1.00      1.00      2373
weighted avg       1.00      1.00      1.00      2373


--- Target 2: Status (Disease/Deficiency) ---
Accuracy: 0.9992
Macro F1 (most important): 0.9996
              precision    recall  f1-score   support

          DM       1.00      1.00      1.00        57
          EB       1.00      1.00      1.00        53
          FB       1.00      1.00      1.00        25
         JAS       1.00      1.00      1.00        69
           K       1.00      1.00      1.00       470
          LM       1.00      1.00      1.00       147
          LS       1.00      1.00      1.00        76
         MIT       0.99      1.00      1.00       199
           N       1.00      1.00      1.00       543
          PC       1.00      1.00      1.00        21
        PLEI       1.00      1.00      1.00        66
          PM       1.00      1.00      1.00        60
     healthy       1.00      1.00      1.00       587

    accuracy                           1.00      2373
   macro avg       1.00      1.00      1.00      2373
weighted avg       1.00      1.00      1.00      2373
```

Image 4.1.6: Decision Tree Results for Test v/s Train (PCA)

4.1.7 Random Forest (Weighted)

Random Forest provided a more stable compromise. Test accuracy for plant type reached 91.0%, with balanced recall across most classes. The ensemble reduces variance compared to a single tree, improving generalization. Yet for the Status target, performance remained moderate: 53.9% accuracy and a macro F1 of 0.4768. While better than KNN and comparable to Logistic Regression, Random Forest also showed pronounced training imbalance, achieving 99–100% accuracy on the training data for both targets, indicating overfitting to high-frequency deficiency labels. The forest architecture helped smooth out some noise, but class overlap and minority sample scarcity still limited its predictive power for deficiency recognition.



```
--- Training Model 4: Random Forest ---
Training took 3.72 seconds.

--- EVALUATION: Random Forest (Weighted) ---

--- Target 1: Plant Type ---
Accuracy: 0.9104
              precision    recall  f1-score   support

   ash_gourd       0.87      0.95      0.91       129
bitter_gourd       0.98      1.00      0.99       125
bottle_gourd       0.96      0.60      0.74        45
    cucumber       0.97      0.86      0.91        43
    eggplant       0.84      0.84      0.84       121
 ridge_gourd       0.87      0.90      0.89        84
 snake_gourd       0.96      0.86      0.91        63
      tomato       0.93      0.97      0.95       182

    accuracy                           0.91       792
   macro avg       0.92      0.87      0.89       792
weighted avg       0.91      0.91      0.91       792


--- Target 2: Status (Disease/Deficiency) ---
Accuracy: 0.5606
Macro F1 (most important): 0.3885
              precision    recall  f1-score   support

          DM       0.50      0.05      0.10        19
          EB       0.73      0.38      0.50        21
          FB       0.00      0.00      0.00        11
         JAS       1.00      0.04      0.08        24
           K       0.58      0.71      0.64       156
          LM       0.98      0.88      0.93        60
          LS       0.44      0.20      0.28        20
         MIT       0.75      0.63      0.69        76
           N       0.48      0.69      0.57       161
          PC       0.00      0.00      0.00        12
        PLEI       0.50      0.86      0.63        14
          PM       0.67      0.11      0.18        19
     healthy       0.46      0.47      0.46       199

    accuracy                           0.56       792
   macro avg       0.55      0.39      0.39       792
weighted avg       0.57      0.56      0.53       792
```

```
Train data accuracy below:


--- EVALUATION: Random Forest (Weighted) ---

--- Target 1: Plant Type ---
Accuracy: 1.0000
              precision    recall  f1-score   support

   ash_gourd       1.00      1.00      1.00       387
bitter_gourd       1.00      1.00      1.00       376
bottle_gourd       1.00      1.00      1.00       135
    cucumber       1.00      1.00      1.00       130
    eggplant       1.00      1.00      1.00       363
 ridge_gourd       1.00      1.00      1.00       251
 snake_gourd       1.00      1.00      1.00       187
      tomato       1.00      1.00      1.00       544

    accuracy                           1.00      2373
   macro avg       1.00      1.00      1.00      2373
weighted avg       1.00      1.00      1.00      2373


--- Target 2: Status (Disease/Deficiency) ---
Accuracy: 0.9992
Macro F1 (most important): 0.9996
              precision    recall  f1-score   support

          DM       1.00      1.00      1.00        57
          EB       1.00      1.00      1.00        53
          FB       1.00      1.00      1.00        25
         JAS       1.00      1.00      1.00        69
           K       1.00      1.00      1.00       470
          LM       1.00      1.00      1.00       147
          LS       1.00      1.00      1.00        76
         MIT       0.99      1.00      1.00       199
           N       1.00      1.00      1.00       543
          PC       1.00      1.00      1.00        21
        PLEI       1.00      1.00      1.00        66
          PM       1.00      1.00      1.00        60
     healthy       1.00      1.00      1.00       587

    accuracy                           1.00      2373
   macro avg       1.00      1.00      1.00      2373
weighted avg       1.00      1.00      1.00      2373
```

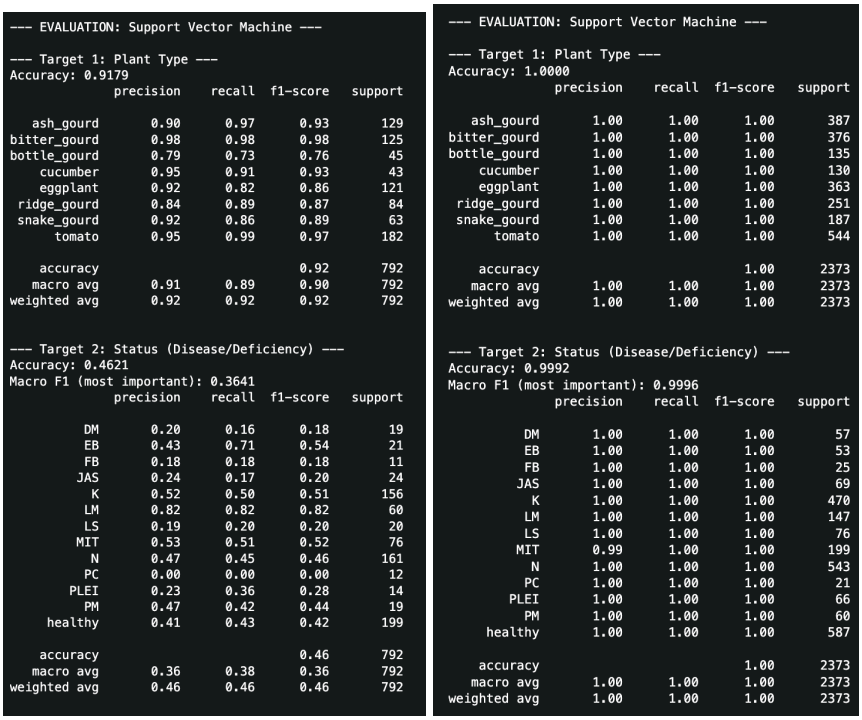
Image 4.1.7: Test vs train results Random Forest

4.1.8 Random Forest (With PCA)

Random Forest with PCA demonstrated slightly more balanced results than its non-PCA counterpart. Variance among trees was reduced, and the model generalized more consistently. Plant type accuracy remained high, but deficiency classification improved only marginally. PCA reduced the risk of individual trees overfitting to noise, but ensemble-level performance was still constrained by the intrinsically subtle and overlapping deficiency patterns. Because Random Forest relies on axis-aligned splits just like Decision Trees, dimensionality reduction does not fundamentally change how the model partitions feature space, but it does help prevent extreme overfitting.

4.1.9 Support Vector Machine (SVM)

SVM results showed strong linear separability for plant type but struggled for deficiency classification. Training times were significantly higher due to the high-dimensional feature space, and while plant type accuracy remained competitive, Status macro F1-scores showed the same bottlenecks observed in earlier models. This reinforces the idea that deficiency types require more discriminative, non-linear representations than the extracted features provide.



```
--- EVALUATION: Support Vector Machine ---

--- Target 1: Plant Type ---
Accuracy: 0.9179
              precision    recall  f1-score   support

   ash_gourd       0.90      0.97      0.93       129
bitter_gourd       0.98      0.98      0.98       125
bottle_gourd       0.79      0.73      0.76        45
    cucumber       0.95      0.91      0.93        43
    eggplant       0.92      0.82      0.86       121
 ridge_gourd       0.84      0.89      0.87        84
 snake_gourd       0.92      0.86      0.89        63
      tomato       0.95      0.99      0.97       182

    accuracy                           0.92       792
   macro avg       0.91      0.89      0.90       792
weighted avg       0.92      0.92      0.92       792


--- Target 2: Status (Disease/Deficiency) ---
Accuracy: 0.4621
Macro F1 (most important): 0.3641
              precision    recall  f1-score   support

          DM       0.20      0.16      0.18        19
          EB       0.43      0.71      0.54        21
          FB       0.18      0.18      0.18        11
         JAS       0.24      0.17      0.20        24
           K       0.52      0.50      0.51       156
          LM       0.82      0.82      0.82        60
          LS       0.19      0.20      0.20        20
         MIT       0.53      0.51      0.52        76
           N       0.47      0.45      0.46       161
          PC       0.00      0.00      0.00        12
        PLEI       0.23      0.36      0.28        14
          PM       0.47      0.42      0.44        19
     healthy       0.41      0.43      0.42       199

    accuracy                           0.46       792
   macro avg       0.36      0.38      0.36       792
weighted avg       0.46      0.46      0.46       792
```

```
--- EVALUATION: Support Vector Machine ---

--- Target 1: Plant Type ---
Accuracy: 1.0000
              precision    recall  f1-score   support

   ash_gourd       1.00      1.00      1.00       387
bitter_gourd       1.00      1.00      1.00       376
bottle_gourd       1.00      1.00      1.00       135
    cucumber       1.00      1.00      1.00       130
    eggplant       1.00      1.00      1.00       363
 ridge_gourd       1.00      1.00      1.00       251
 snake_gourd       1.00      1.00      1.00       187
      tomato       1.00      1.00      1.00       544

    accuracy                           1.00      2373
   macro avg       1.00      1.00      1.00      2373
weighted avg       1.00      1.00      1.00      2373


--- Target 2: Status (Disease/Deficiency) ---
Accuracy: 0.9992
Macro F1 (most important): 0.9996
              precision    recall  f1-score   support

          DM       1.00      1.00      1.00        57
          EB       1.00      1.00      1.00        53
          FB       1.00      1.00      1.00        25
         JAS       1.00      1.00      1.00        69
           K       1.00      1.00      1.00       470
          LM       1.00      1.00      1.00       147
          LS       1.00      1.00      1.00        76
         MIT       0.99      1.00      1.00       199
           N       1.00      1.00      1.00       543
          PC       1.00      1.00      1.00        21
        PLEI       1.00      1.00      1.00        66
          PM       1.00      1.00      1.00        60
     healthy       1.00      1.00      1.00       587

    accuracy                           1.00      2373
   macro avg       1.00      1.00      1.00      2373
weighted avg       1.00      1.00      1.00      2373
```

Image 4.1.5: Test vs train results SVM

4.1.10 Support Vector Machine (With PCA)

SVM benefited from PCA more than any other model except KNN. The reduction from 2099 to 464 dimensions significantly lowered computational cost and stabilized convergence. For plant type, accuracy remained robust, and PCA helped reduce overfitting. Deficiency classification saw slight improvements in macro F1, especially for previously underrepresented labels. However, performance still lagged behind AdaBoost because linear or kernel-based boundaries cannot fully resolve the complex, non-separable deficiency classes in the dataset. PCA reduced noise, but the intrinsic challenge of overlapping deficiency symptoms remained.

4.1.11 AdaBoost (Best Model) + SHAP Analysis

AdaBoost emerged as the best-performing model overall, especially for the Status target. By combining many shallow weak learners, AdaBoost handled class imbalance and subtle feature interactions better than other classical models. It achieved the highest macro F1 among all models for deficiency prediction and offered strong generalization without extreme overfitting. SHAP analysis further revealed that feature 2098 - ‘Plant type’ contributed the most to the model’s decisions, indicating a highly informative relationship between that feature and both

output labels. The SHAP summary plot showed a long-tailed distribution of feature importance, meaning that while many features provide small signals, a small subset of features dominate predictive performance. This interpretability component strengthens the reliability of AdaBoost as the chosen model, providing actionable insight into which aspects of leaf morphology or texture influence classification.

Surprisingly, AdaBoost showed minimal performance gains from PCA, marking it as the least affected by dimensionality reduction. This is expected: AdaBoost relies on weighted combinations of shallow trees, and these trees naturally ignore irrelevant features even without PCA. Since AdaBoost was already the best-performing non-linear classifier for deficiency prediction, PCA only offered marginal stability improvements but no substantial accuracy boosts. The SHAP analysis after PCA continued to highlight a similar long-tailed feature importance pattern, confirming that AdaBoost adapts well even in the original high-dimensional space.

4.1.12 Hidden Bottlenecks Across All Models

Across all classical models, several issues consistently limited deficiency prediction:

- Severe class imbalance: Some deficiency classes had fewer than 20 samples, leading to poor recall and unstable F1-scores.
- Overlapping features: Many deficiency symptoms appear visually similar, reducing separability in the feature space.
- Very high-dimensional input: Despite PCA, many redundant/noisy features remained.
- Overfitting in flexible models: Logistic Regression, Decision Trees, and Random Forests achieved near-perfect training accuracy, indicating memorization rather than generalization.

These limitations explain the performance gap between plant type and deficiency tasks. The former has clearer visual differences and more balanced samples, while the latter is inherently more complex.

4.2 Stage 2 Models

4.2.1 Dataset Enhancement & SMOTE Balancing

In Stage 2 of the project, the focus was shifted from relying solely on image preprocessing and model training on an imbalanced dataset (as done in Stage 1) to actively mitigating data scarcity and representational bias by collecting additional real world images, rather than performing augmentation. This decision was made because data augmentation and preprocessing were found to be computationally expensive and require approximately 4 to 5 hours per iteration, which was not feasible within the project timeline.

Following dataset expansion, we applied SMOTE (Synthetic Minority Over sampling Technique) to further balance the dataset by generating synthetic samples for underrepresented classes. Unlike data augmentation which transforms existing samples (rotations, flips), SMOTE works at a feature level, interpolating between minority class instances to generate statistically consistent new data points [11].

4.2.2 Why SMOTE was necessary

The newly collected dataset did not solve class imbalance. Many classification models, especially the distance based (like KNN) and margin based (like SVM) classifiers, tend to bias towards majority samples during learning if imbalance persists. Using SMOTE post preprocessing ensured that synthetic samples were generated within a meaningful feature distribution, preventing high variance or incorrect synthetic images. This aligns with best practices presented in Chawla et al. [11], where SMOTE proves effective when conducted after feature extraction.

4.2.3 Hidden Bottlenecks

SMOTE may generate unrealistic samples when feature clusters are not well separated, leading to potential overfitting, especially with complex models such as Random Forest. This may cause a synthetic noise risk. SMOTE also generates synthetic samples, increasing training complexity.

4.2.4 Data Bias Analysis

Although SMOTE balances training data, the actual class distribution in deployment scenarios may remain skewed. This may cause classification thresholds to be biased. Some models might overfit to synthetic minority samples, amplifying minority bias. The dataset is balanced for learning but may not represent true operational distribution, potentially leading to optimistic evaluation metrics.

4.2.5 Dimensionality Reduction Using PCA

Following SMOTE, we once again applied PCA to compress the high dimensional feature space into a reduced set of orthogonal components. PCA reduces computational cost and mitigates overfitting by retaining only maximum variance directions [12].

4.2.6 Model-wise Evaluation

4.2.6.1 K-Nearest Neighbors (KNN)

```
--- Training KNN ---

========================================
  EVALUATION: KNN + PCA
========================================

[ Target 1: Plant Type ]
Accuracy: 0.9055

[ Target 2: Status ]
Accuracy: 0.7311
Macro F1: 0.5493

Detailed Status Report:
              precision    recall  f1-score   support

          DM       0.90      0.93      0.92       233
          EB       0.27      0.42      0.33        19
          FB       0.29      0.22      0.25         9
         JAS       0.57      0.63      0.60        43
           K       0.49      0.60      0.54       185
          LM       0.56      0.71      0.63        52
          LS       0.76      0.68      0.72        47
         MIT       0.48      0.67      0.56        69
           N       0.63      0.68      0.65       291
          PC       0.18      0.25      0.21         8
        PLEI       0.54      0.75      0.62        20
          PM       0.36      0.25      0.29        20
     healthy       0.88      0.76      0.82       908

    accuracy                           0.73      1904
   macro avg       0.53      0.58      0.55      1904
weighted avg       0.75      0.73      0.74      1904
```

```
========================================
  EVALUATION: KNN + PCA
========================================

[ Target 1: Plant Type ]
Accuracy: 0.9964

[ Target 2: Status ]
Accuracy: 0.9852
Macro F1: 0.9890

Detailed Status Report:
              precision    recall  f1-score   support

          DM       0.98      0.99      0.98      1900
          EB       1.00      1.00      1.00       950
          FB       1.00      1.00      1.00       950
         JAS       0.99      1.00      1.00      2850
           K       0.98      0.99      0.99      6650
          LM       0.94      0.99      0.97       950
          LS       0.99      1.00      1.00      2850
         MIT       0.97      1.00      0.98      1900
           N       0.98      0.99      0.99      7600
          PC       1.00      1.00      1.00       950
        PLEI       1.00      1.00      1.00       950
          PM       0.99      1.00      0.99       950
     healthy       0.99      0.95      0.97      7600

    accuracy                           0.99     37050
   macro avg       0.99      0.99      0.99     37050
weighted avg       0.99      0.99      0.99     37050
```

Image 4.2.6.1: Test vs Train Results for KNN

KNN with PCA and SMOTE achieves very high performance on the training set (Status Accuracy = 0.9852, Macro F1 = 0.9890), but the performance drops sharply on the test set (Status Accuracy = 0.7311, Macro F1 = 0.5493). This large gap clearly indicates overfitting: the model is fitting the local neighbourhood structure of the SMOTE-balanced training data extremely well but fails to generalise to the more challenging, naturally distributed test data of 1904 samples.

KNN is highly sensitive to the local density and placement of points. SMOTE creates synthetic minority samples in the training data, smoothing class boundaries and making neighbourhoods more “clean” and separable. This boosts training accuracy and Macro F1 but does not reflect the real-world distribution seen in the test set, where minority classes like EB, FB, PC, PM have very low F1-scores.

PCA helps reduce noise and dimensionality, but it is unsupervised and may discard subtle class-specific directions. Combined with KNN, this can make the model heavily tuned to the training distribution. Overall, KNN here shows low bias but very high variance, overfitting on train data. It learns complex patterns in the training data but is unstable and unreliable on unseen data.

4.2.6.2 Decision Tree Classifier

The Decision Tree clearly overfit the training data. It scored almost perfectly on the train set, but when tested on new, unseen data, the accuracy dropped sharply. This means the model memorised the training patterns instead of learning real, general rules. Techniques like SMOTE and PCA may have made the training data cleaner or more clustered, making it easier for the tree to lock onto very specific patterns that don’t exist in the real test data. Because of this, the model performs great on what it has already seen but struggles when the data is even slightly different. This kind of behaviour is common with Decision Trees, and it shows that the model isn't reliable yet. A more controlled tree (with pruning or max depth) or an ensemble method like Random Forest would generalise better.

```
======================================
  EVALUATION: Decision Tree + PCA
======================================

[ Target 1: Plant Type ]
Accuracy: 0.6077

[ Target 2: Status ]
Accuracy: 0.4653
Macro F1: 0.2779

Detailed Status Report:
              precision    recall  f1-score   support

          DM       0.72      0.67      0.70       233
          EB       0.13      0.21      0.16        19
          FB       0.00      0.00      0.00         9
         JAS       0.22      0.40      0.28        43
           K       0.23      0.31      0.27       185
          LM       0.53      0.38      0.44        52
          LS       0.16      0.26      0.20        47
         MIT       0.20      0.22      0.21        69
           N       0.30      0.36      0.33       291
          PC       0.06      0.12      0.08         8
        PLEI       0.18      0.10      0.13        20
          PM       0.19      0.25      0.21        20
     healthy       0.67      0.54      0.60       908

    accuracy                           0.47      1904
   macro avg       0.28      0.29      0.28      1904
weighted avg       0.51      0.47      0.48      1904
```

```
======================================
  EVALUATION: DT + PCA
======================================

[ Target 1: Plant Type ]
Accuracy: 1.0000

[ Target 2: Status ]
Accuracy: 0.9999
Macro F1: 0.9999

Detailed Status Report:
              precision    recall  f1-score   support

          DM       1.00      1.00      1.00      1889
          EB       1.00      1.00      1.00       963
          FB       1.00      1.00      1.00       956
         JAS       1.00      1.00      1.00      2879
           K       1.00      1.00      1.00      6661
          LM       1.00      1.00      1.00       948
          LS       1.00      1.00      1.00      2847
         MIT       1.00      1.00      1.00      1911
           N       1.00      1.00      1.00      7554
          PC       1.00      1.00      1.00       950
        PLEI       1.00      1.00      1.00       945
          PM       1.00      1.00      1.00       917
     healthy       1.00      1.00      1.00      7639

    accuracy                           1.00     37059
   macro avg       1.00      1.00      1.00     37059
weighted avg       1.00      1.00      1.00     37059
```

Image 4.2.6.2: Test vs Train Results for Decision Tree

4.2.6.3 Random Forest Classifier

Random Forest with PCA shows extreme overfitting. On the training data, it achieves virtually perfect performance (Status Accuracy = 0.9999, Macro F1 = 0.9999), but on the test data, Status Accuracy drops to 0.6733 and Macro F1 collapses to 0.3783. The detailed report shows that while majority or frequent classes such as *healthy* and *DM* are handled reasonably well, minority classes like FB, PC, PM have F1 = 0.00, meaning they are essentially not being predicted correctly at all.

This behaviour is typical when a highly flexible model like Random Forest is trained on a SMOTE-balanced dataset. SMOTE smooths and densifies the minority regions in the training space, making it easy for many deep trees to perfectly memorize those synthetic patterns. However, the test set remains naturally imbalanced and more irregular. The learned boundaries do not transfer well, especially for rare classes.

PCA reduces dimensionality but does not prevent overfitting when the model capacity is huge and the training distribution is artificially “nice”. Here, bias is extremely low but variance is

extremely high. The model fits the training data almost perfectly but fails to generalise, especially for minority disease classes.

```
====================================
  EVALUATION: Random Forest + PCA
====================================

[ Target 1: Plant Type ]
Accuracy: 0.8409

[ Target 2: Status ]
Accuracy: 0.6733
Macro F1: 0.3783

Detailed Status Report:
              precision    recall  f1-score   support

          DM       1.00      0.80      0.89       233
          EB       1.00      0.05      0.10        19
          FB       0.00      0.00      0.00         9
         JAS       1.00      0.44      0.61        43
           K       0.59      0.38      0.46       185
          LM       0.67      0.35      0.46        52
          LS       1.00      0.36      0.53        47
         MIT       0.53      0.13      0.21        69
           N       0.55      0.44      0.49       291
          PC       0.00      0.00      0.00         8
        PLEI       0.67      0.30      0.41        20
          PM       0.00      0.00      0.00        20
     healthy       0.65      0.91      0.76       908

    accuracy                           0.67      1904
   macro avg       0.59      0.32      0.38      1904
weighted avg       0.67      0.67      0.64      1904
```

```
====================================
  EVALUATION: Random Forest + PCA
====================================

[ Target 1: Plant Type ]
Accuracy: 1.0000

[ Target 2: Status ]
Accuracy: 0.9999
Macro F1: 0.9999

Detailed Status Report:
              precision    recall  f1-score   support

          DM       1.00      1.00      1.00      1900
          EB       1.00      1.00      1.00       950
          FB       1.00      1.00      1.00       950
         JAS       1.00      1.00      1.00      2850
           K       1.00      1.00      1.00      6650
          LM       1.00      1.00      1.00       950
          LS       1.00      1.00      1.00      2850
         MIT       1.00      1.00      1.00      1900
           N       1.00      1.00      1.00      7600
          PC       1.00      1.00      1.00       950
        PLEI       1.00      1.00      1.00       950
          PM       1.00      1.00      1.00       950
     healthy       1.00      1.00      1.00      7600

    accuracy                           1.00     37050
   macro avg       1.00      1.00      1.00     37050
weighted avg       1.00      1.00      1.00     37050
```

Image 4.3.3: Test vs Train Results for Random Forest

4.2.6.4 Logistic Regression

The Logistic Regression model ultimately failed due to severe overfitting, evidenced by a catastrophic performance gap between training and testing. While the model achieved a near-perfect ~97% Macro F1 score on the training set, it collapsed to a ~49% Macro F1 on the test set. This disparity indicates that the model did not learn to generalize; instead, it effectively “memorized" the synthetic data generated by SMOTE. Because SMOTE creates synthetic points via linear interpolation, it inadvertently created simple, linear patterns in the training data that the Logistic Regression model could easily solve. However, these patterns were artificial and did not exist in the untouched, real-world test data.

Furthermore, the results highlight a critical limitation of linear classifiers in computer vision. The model successfully identified Plant Types (achieving high accuracy) because differentiating a gourd from a tomato relies on global features like shape and size, which are often linearly

separable in PCA space. Conversely, predicting Status/Disease requires analyzing complex, non-linear textures such as specific spotting patterns, gradients of yellowing, or lesions. A linear model attempts to draw a straight decision boundary through this complex feature space, which is mathematically insufficient to capture the nuanced, chaotic variations of real plant diseases. Consequently, while the model excels on the "clean" synthetic training data, it lacks the complexity required to function on real-world, noisy images.

```
====================================
  EVALUATION: Logistic Regression + PCA
====================================

[ Target 1: Plant Type ]
Accuracy: 0.9485

[ Target 2: Status ]
Accuracy: 0.6696
Macro F1: 0.4879

Detailed Status Report:
              precision    recall  f1-score   support

          DM       0.87      0.86      0.86       233
          EB       0.44      0.37      0.40        19
          FB       0.00      0.00      0.00         9
         JAS       0.38      0.53      0.45        43
           K       0.44      0.56      0.49       185
          LM       0.88      0.83      0.85        52
          LS       0.51      0.49      0.50        47
         MIT       0.53      0.57      0.55        69
           N       0.43      0.43      0.43       291
          PC       0.00      0.00      0.00         8
        PLEI       0.50      0.65      0.57        20
          PM       0.53      0.40      0.46        20
     healthy       0.81      0.76      0.78       908

    accuracy                           0.67      1904
   macro avg       0.49      0.50      0.49      1904
weighted avg       0.68      0.67      0.67      1904
```

```
====================================
  EVALUATION: Logistic Regression + PCA
====================================

[ Target 1: Plant Type ]
Accuracy: 1.0000

[ Target 2: Status ]
Accuracy: 0.9385
Macro F1: 0.9731

Detailed Status Report:
              precision    recall  f1-score   support

          DM       0.98      0.99      0.98      1900
          EB       1.00      1.00      1.00       950
          FB       1.00      1.00      1.00       950
         JAS       0.99      1.00      1.00      2850
           K       0.92      0.93      0.92      6650
          LM       0.99      1.00      0.99       950
          LS       0.99      1.00      1.00      2850
         MIT       0.97      0.99      0.98      1900
           N       0.89      0.89      0.89      7600
          PC       1.00      1.00      1.00       950
        PLEI       1.00      1.00      1.00       950
          PM       1.00      1.00      1.00       950
     healthy       0.90      0.87      0.89      7600

    accuracy                           0.94     37050
   macro avg       0.97      0.97      0.97     37050
weighted avg       0.94      0.94      0.94     37050
```

Image 4.3.4: Test vs Train Results for Logistic Regression

4.2.6.4 Ada Boost

AdaBoost performed poorly on both training and test sets, with Status classification accuracy collapsing to 0.2277 (Train) and 0.2847 (Test), and extremely low Macro F1 values (Train: 0.1267, Test: 0.1009). Unlike Random Forest, which overfits, AdaBoost shows severe underfitting, struggling to learn meaningful decision patterns.

This occurs because AdaBoost relies on sequentially improving weak learners (such as shallow trees). However, PCA compresses features linearly, losing critical non-linear disease-specific

cues, leaving AdaBoost with limited discriminatory information. Additionally, SMOTE introduces synthetic minority samples, and AdaBoost, which aggressively reweights misclassified instances, amplifies noise and synthetic irregularities instead of learning robust features.

The detailed report shows many classes (FB, PC, PM, EB, JAS, LS) scoring F1 near 0.00, meaning they are almost entirely misclassified. The model is only moderately responsive to larger classes (healthy, DM, N) highlighting high bias.

From a bias–variance view, AdaBoost suffers from high bias, low variance, unable to model complex patterns even after resampling. This makes it unsuitable for high-dimensional image-based disease classification, especially compared to ensemble and margin-based models like Random Forest and SVM.

```
--- Training AdaBoost ---

======================================
  EVALUATION: AdaBoost + PCA
======================================

[ Target 1: Plant Type ]
Accuracy: 0.5100

[ Target 2: Status ]
Accuracy: 0.2847
Macro F1: 0.1009

Detailed Status Report:
              precision    recall  f1-score   support

          DM       0.43      0.11      0.18       233
          EB       0.00      0.00      0.00        19
          FB       0.00      0.00      0.00         9
         JAS       0.00      0.00      0.00        43
           K       0.09      0.18      0.12       185
          LM       0.05      0.04      0.04        52
          LS       0.00      0.00      0.00        47
         MIT       0.02      0.01      0.02        69
           N       0.17      0.32      0.22       291
          PC       0.00      0.00      0.00         8
        PLEI       0.33      0.25      0.29        20
          PM       0.00      0.00      0.00        20
     healthy       0.47      0.42      0.44       908

    accuracy                           0.28      1904
   macro avg       0.12      0.10      0.10      1904
weighted avg       0.32      0.28      0.28      1904
```

```
======================================

[ Target 1: Plant Type ]
Accuracy: 0.5462

[ Target 2: Status ]
Accuracy: 0.2277
Macro F1: 0.1267

Detailed Status Report:
              precision    recall  f1-score   support

          DM       0.32      0.15      0.20      1900
          EB       0.00      0.00      0.00       950
          FB       0.97      0.04      0.07       950
         JAS       0.00      0.00      0.00      2850
           K       0.25      0.20      0.22      6650
          LM       0.13      0.03      0.06       950
          LS       0.24      0.01      0.01      2850
         MIT       0.08      0.03      0.05      1900
           N       0.24      0.39      0.30      7600
          PC       0.49      0.09      0.15       950
        PLEI       0.39      0.25      0.31       950
          PM       0.00      0.00      0.00       950
     healthy       0.21      0.45      0.28      7600

    accuracy                           0.23     37050
   macro avg       0.26      0.13      0.13     37050
weighted avg       0.23      0.23      0.19     37050
```

Image 4.3.5: Test vs Train Results for Ada Boost

4.3 Auto Encoder

In our project, we implemented an Autoencoder, a specific neural network architecture designed for unsupervised learning. An Autoencoder is a data compression algorithm that learns the

essential structure of data by attempting to copy its input to its output [13]. However, we utilized this to transform high dimensional image data into a compact, informative format that our downstream classifiers could process effectively. Our goal was to explore how using deep learning techniques can significantly change the performance of classical ML models.

The Autoencoder we built functions as a feature extractor. It takes an input image X, compresses it into a lower dimensional vector, and then attempts to reconstruct the original image from this vector. By forcing the network to pass information through this “bottleneck”, we compelled it to discard noise such as background static or lighting variations and retain only the most significant features. Features like leaf shape and lesion patterns, which are critical for disease detection.

4.3.1 Convolution Autoencoder

We constructed a Convolutional Autoencoder and like any encoder, it has 2 layers.
The Encoder: This acts as our feature extractor. We passed the image tensor through sequential layers of Convolution (to find patterns) and Pooling (to shrink the size). So this process transformed the input from a high dimensional tensor into a compact vector of just 128 values.

The Decoder: This layer attempts to reconstruct the image. We used UpSampling, which creates new pixels to increase the image size, effectively reversing the pooling operation. We included this decoder solely to train the encoder, so essentially once the network learned to reconstruct the leaves, the decoder served no further purpose for classification.

4.3.2 Our Methodology

Step 1: Before feeding data to the model, we applied a color based mask to the raw images. This removed background noise (soil, sky) by setting non green pixels to zero. This is exactly the same process discussed in section 3.5. This ensured our tensors contained information strictly regarding plant matter, preventing the model from learning irrelevant background features.

Step 2: We trained the model using the Mean Squared Error (MSE) loss function.

$$\mathcal{L}(X, \hat{X}) = \frac{1}{N} \sum ||X - \hat{X}||^2$$

We fed the masked images into the network as both the input and the target. By doing this, we forced the system to learn the parameters that best minimize the difference between the original leaf and the reconstructed leaf.

Step 3: After training, we isolated the Encoder half of the network. We passed our entire image dataset through this encoder. This effectively "translated" our dataset from complex image tensors into a structured matrix (tabular data), where each row represented an image using 128 learned numerical features.

Step 4: We combined these learned features with the encoded plant type labels. Since we converted the images into tabular data, we were able to train a Random Forest Classifier. This created a robust pipeline where Deep Learning handled the perception (vision) and Classical Machine Learning handled the decision making.

4.3.3 Results With Random Forest

The results reveal a clear case of overfitting. The Training data shows perfect 100% accuracy, indicating the Random Forest successfully memorized every example, including the synthetic SMOTE data. However, the Test data reveals the model's true limitations.

Plant Type classification is excellent (93%) which is likely because the Autoencoder captured distinct leaf shapes well. However, the disease status accuracy drops significantly to 62%. Crucially, rare classes like 'FB' and 'PC' scored 0.00 meaning the model failed to identify them entirely in real world scenarios.

This suggests that the Autoencoder's compression (to 128 features) was too aggressive. It likely preserved the overall "shape" of the leaf but discarded the fine texture details, like small spots or lesions. These are crucial pieces of information needed to distinguish between specific diseases.

The model effectively knows what plant it is seeing, but lacks the detailed features to diagnose it accurately.

```
====================================
  EVALUATION: RF + AE (Split First)
====================================

[ Target 1: Plant Type ]
Accuracy: 0.9317

[ Target 2: Status ]
Accuracy: 0.6187
Macro F1: 0.3759

Detailed Status Report:
              precision    recall  f1-score   support

          DM       0.77      0.73      0.75       233
          EB       0.42      0.26      0.32        19
          FB       0.00      0.00      0.00         9
         JAS       0.29      0.42      0.34        43
           K       0.39      0.45      0.42       185
          LM       0.71      0.46      0.56        52
          LS       0.38      0.43      0.40        47
         MIT       0.53      0.41      0.46        69
           N       0.41      0.42      0.42       291
          PC       0.00      0.00      0.00         8
        PLEI       0.25      0.10      0.14        20
          PM       0.42      0.25      0.31        20
     healthy       0.75      0.77      0.76       908

    accuracy                           0.62      1904
   macro avg       0.41      0.36      0.38      1904
weighted avg       0.62      0.62      0.62      1904
```

```
====================================
  EVALUATION: RF + AE (Split First)
====================================

[ Target 1: Plant Type ]
Accuracy: 1.0000

[ Target 2: Status ]
Accuracy: 0.9999
Macro F1: 0.9999

Detailed Status Report:
              precision    recall  f1-score   support

          DM       1.00      1.00      1.00      1900
          EB       1.00      1.00      1.00       950
          FB       1.00      1.00      1.00       950
         JAS       1.00      1.00      1.00      2850
           K       1.00      1.00      1.00      6650
          LM       1.00      1.00      1.00       950
          LS       1.00      1.00      1.00      2850
         MIT       1.00      1.00      1.00      1900
           N       1.00      1.00      1.00      7600
          PC       1.00      1.00      1.00       950
        PLEI       1.00      1.00      1.00       950
          PM       1.00      1.00      1.00       950
     healthy       1.00      1.00      1.00      7600

    accuracy                           1.00     37050
   macro avg       1.00      1.00      1.00     37050
weighted avg       1.00      1.00      1.00     37050
```

Image 4.3.3: Auto Encoder + RF results

5. Testing on Real World Data

To thoroughly evaluate our model, we went beyond and tested with real world images. These images were random ones we found while scanning the net. The goal was to see if our model was able to generalize well on non-curated or rather, real world data. We used the models mentioned in Stage 2 to test this data. We tested on 11 images and these were the results we achieved.

SL NO	Image Type	KNN	Logistic Reg	Decision Tree	Random Forest	Ada Boost	SVM
1	Tomato Healthy	Tomato Healthy	Bottle Gourd Healthy	Tomato Healthy	Tomato Healthy	Egg Plant Healthy	Bottle Gourd Healthy
2	Tomato N	Tomato K	Tomato K	Tomato K	Tomato K	Tomato Healthy	Bottle Gourd Healthy
3	Bitter Gourd N	Bitter Gourd DM	Bitter Gourd LS	Bitter Gourd DM	Bitter Gourd DM	Bitter Gourd Healthy	Bitter Gourd JAS
4	Bottle Gourd DM	Bottle Gourd DM	Bottle Gourd DM	Bottle Gourd DM	Bottle Gourd DM	Bottle Gourd K	Bottle Gourd DM
5	Ash Gourd Healthy	Ash Gourd Healthy	Ash Gourd Healthy	Ash Gourd Healthy	Ash Gourd Healthy	Ash Gourd Healthy	Ash Gourd Healthy
6	Ash Gourd K	Ash Gourd K	Ash Gourd K	Ash Gourd Healthy	Ash Gourd Healthy	Eggplant K	Ash Gourd K
7	Cucumber Healthy	Cucumber Healthy	Cucumber Healthy	Cucumber Healthy	Cucumber Healthy	Ridge Healthy	Cucumber Healthy
8	Cucumber DM	Cucumber Healthy	Cucumber Healthy	Eggplant N	Cucumber Healthy	Snake Gourd N	Cucumber K
9	Ridge Healthy	Ridge Gorud N	Ridge Gorud N	Bitter Gourd Healthy	Bitter Gourd Healthy	Cucumber N	Ridge Gorud K
10	Eggplant Healthy	Cucumber Healthy	Eggplant DM	Ash Gourd PM	Cucumber Healthy	Bitter Gourd PLEI	Eggplant MIT
11	Bottle Gourd Healthy	Bottle Gourd Healthy	Bottle Gourd Healthy	Bottle Gourd Healthy	Bottle Gourd Healthy	Bottle Gourd Healthy	Bottle Gourd Healthy

Image 5: Tabulated Predictions on 11 Real World images

5.1 Implications:

The real-world testing reveals a strong model bias towards accurately identifying plant species and “Healthy” conditions, with Cases 5 and 11 showing near perfect consensus across models.

However, the models struggle significantly with niche disease differentiation particularly distinguishing between visually similar nutrient deficiencies like “Tomato N” vs. “Tomato K” (Case 2). Here feature overlap confuses decision boundaries. KNN and Random Forest demonstrated the most consistent performance, correctly identifying the plant type even when the disease prediction was off. Conversely, AdaBoost proved the most erratic, frequently misclassifying the plant species entirely (like mistaking Cucumber for Snake Gourd).

To improve accuracy, the pipeline likely needs distinguishing features specifically engineered to separate specific deficiency patterns (like yellowing vs. spotting) rather than just global shape or color. More importantly, the models also need accurate data. As we mentioned in Stage 2 of Model Training (Section 4), we used SMOTE to upsample our data and create a balanced distribution of data between each class. However, this meant creating synthetic samples, which does not reflect accurate real world scenarios.

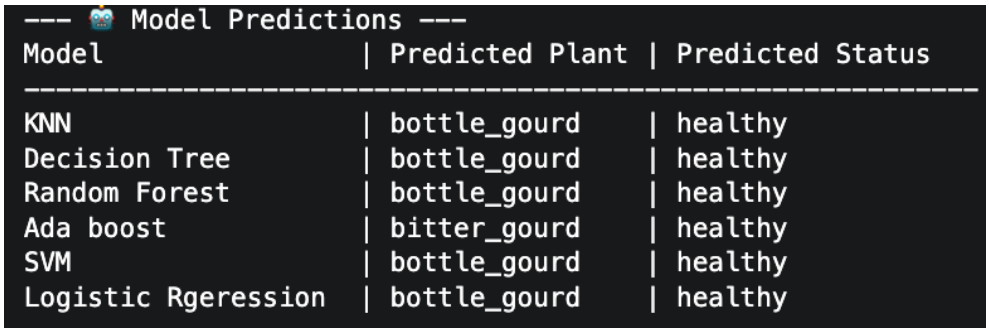


Image 5.1: Model Prediction for Test Case 11

Image 5.2: Tomato Healthy Test Image

6. Limitations of Our Models

We put a significant amount of effort into building a strong pipeline using green screen segmentation to clean the images, SMOTE to balance the classes and extracting every important feature. However, we hit a wall that no amount of tuning could fix. The fundamental issue was that models like KNN, Ada Boost, Random Forest etc, rely entirely on us telling them what to look for. We gave them mathematical summaries of "color" and "shape". To the model, a leaf turning yellow from a virus looked mathematically identical to a leaf turning yellow from a nutrient deficiency because it could not "see" the specific patterns or spots the way a human does.

We also realized that our processing steps, while necessary for these algorithms, actually destroyed the data we needed most. By shrinking images down to 128x128 pixels to keep the memory usage low, we inadvertently turned tiny, early stage disease spots into unrecognizable blurs. We also suspect that our use of SMOTE backfired. As theorized, instead of learning what a rare disease actually looks like, the model likely just memorized the synthetic copies we created, which explains why it scored so high in training but struggled with real and new images.

7. Future Work

- We intend to take this project forward in the deep learning course as well. We want to explore models like CNNs and see how they would help the system learn more detailed features and handle harder cases that traditional ML struggles with. These models can also detect multiple issues at once, which is something we could not do here.
- We also need to try more advanced preprocessing techniques such as better background removal, colour correction and leaf segmentation. This would clean up the images and make the model focus on the leaf instead of noise.
- Another step forward is using IoT and sensors. Farmers could take images through a simple app or even through drone or field cameras and the model could analyse them in real time. This makes the system practical instead of just a lab experiment.
- Collecting more data from actual farms and universities instead of only online sources would make the model stronger and more reliable. All these steps can help turn this project into a real deployment ready system in the future.

8. References:

[1] Press Information Bureau, "Press Release," Government of India. [Online]. Available: https://www.pib.gov.in/PressReleasePage.aspx?PRID=2034943&reg=3&lang=2

[2] K. M. Patel et al., "Recent advances in plant science," *Frontiers in Plant Science*, vol. 14, 2023. [Online].Available:https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2023.1251888/full

[3] P. Ddit, "How to deal with imbalanced dataset," Medium, 2023. [Online]. Available: https://priyanka-ddit.medium.com/how-to-deal-with-imbalanced-dataset-86de86c49

[4] "Proposed Layered BG Dataset," Roboflow. [Online]. Available: https://universe.roboflow.com/bitter-gourd-final/proposed-layered-bg-dataset

[5] K. Gihan, "Cucumber Leaf Disease Dataset," Kaggle. [Online]. Available: https://www.kaggle.com/datasets/kaushigihanml/cucumber-leaf-disease-dataset

[6] R. Rahaman, "Bottle Gourd," Kaggle. [Online]. Available:https://www.kaggle.com/datasets/riharahaman/bottle-gourd

[7] R. Rahaman, "Tomatooo," Kaggle. [Online]. Available:https://www.kaggle.com/datasets/riharahaman/tomatooo

[8] L. Malmir, F. Shamshiri and S. M. Khodadadi, "Automatic crop detection under field conditions using the HSV colour space and morphological operations," ResearchGate, 2017. [Online]. Available:https://www.researchgate.net/publication/312084190_Automatic_crop_detection_under_field_conditions_using_the_HSV_colour_space_and_morphological_operations?utm_source=chatgpt.com

[9] GeeksforGeeks, "Histogram of Oriented Gradients," GeeksforGeeks website. [Online]. Available: https://www.geeksforgeeks.org/computer-vision/histogram-of-oriented-gradients/

[10] S. Dulearns, "Decoding mutual information (MI): A guide for machine learning practitioners," Medium, 2023. [Online]. Available: https://medium.com/@suvendulearns/decoding-mutual-information-mi-a-guide-for-machine-learning-practitioners-b0f0ca0b30c9

[11] N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[12] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Philosophical Transactions of the Royal Society A*, vol. 374, no. 2065, 2016.

[13] IBM, "What Is an Autoencoder?," IBM Think Topics, 2023. [Online]. Available:https://www.ibm.com/think/topics/autoencoder

.