Employing Computer Vision for Disease Detection in the PlantVillage Dataset

Closson, Nikolay; Frenette, Maxence; Lin, Brian; Minazzo, Agnese; Vahid, Mana

DATASCI 281 Computer Vision - Final Project

Abstract

Tomato crops significantly contribute to agricultural revenue worldwide, yet they face persistent threats from various plant diseases that can drastically reduce yields. This paper introduces a novel application of computer vision technologies to identify and classify diseases in tomato plants using the PlantVillage dataset. By integrating traditional feature extraction techniques such as color histogram, color based segmentation, and edge detection in tandem with complex feature extraction techniques (Haralick features and ResNet50) and machine learning algorithms, our system aims to automate disease diagnosis, offering a scalable solution to enhance crop management and mitigate yield losses. Preliminary results demonstrate the potential of our model to achieve a good accuracy in disease identification, promising a substantial impact on global food security and the agricultural economy (Hughes & Salathé, 2015; Mohanty et al., 2016).

1. Introduction

Tomatoes are among the most economically significant horticultural crops globally, contributing extensively to agricultural revenues. In the United States alone, tomatoes account for over $2 billion annually in farm cash receipts (USDA, 2021), highlighting their economic importance. Internationally, tomatoes are pivotal in both developed and developing economies, underlining the critical need for effective disease management strategies to sustain production levels and market stability.

Plant diseases are a principal challenge in tomato cultivation, with the potential to cause up to 40% yield loss annually, severely impacting revenue and food supply (Oerke, 2006). Traditional disease detection methods rely on expert knowledge, which is not always accessible, especially in remote areas. Advances in digital technology and machine learning present an opportunity to revolutionize disease diagnosis through automated systems. This project leverages a comprehensive dataset of over 54,000 images from the PlantVillage platform, featuring healthy and diseased tomato leaves across multiple categories, to develop a robust image-based disease detection system (Hughes & Salathé, 2015).

Utilizing a combination of simple and complex feature extraction methods, this study explores the effectiveness of machine learning algorithms in identifying disease patterns specific to tomato plants. We assessed the performance of several classifiers, including Logistic Regression, Simple

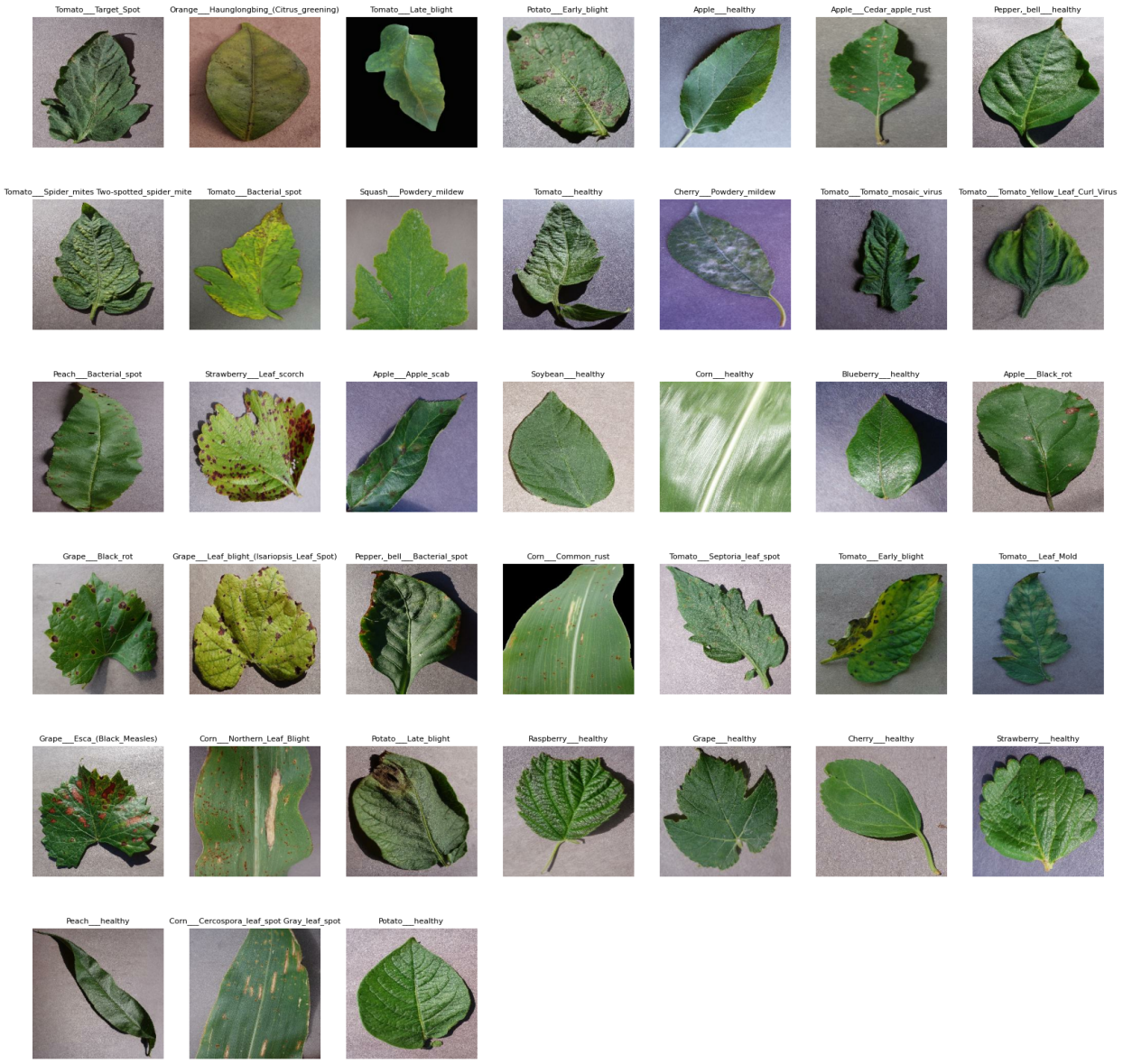
Perceptron, and Random Forest models, facilitated by deep learning techniques through large pre-trained Convolutional Neural Networks. Our approach not only aims to enhance the accuracy of plant disease diagnosis but also contributes to the scalability of solutions needed to meet the increasing global food demand projected for 2050 (FAO, 2020).

In summary, this paper presents a detailed analysis of applying computer vision to detect tomato plant diseases, thereby supporting efforts to improve yield, reduce economic losses, and increase food security worldwide.

2. Dataset

The dataset utilized in this project is the PlantVillage dataset, a comprehensive collection designed to facilitate the identification of diseases in crop leaves. It includes a total of 54,303 images, represented by 38 different categories. These categories cover a range of healthy and diseased leaf images from 14 different crops (Figure1). The dataset was imported and processed using TensorFlow Datasets (‘tfds’), ensuring efficient handling and integration within the project.

Figure 1. Random sample of crop images from the PlantVillage dataset.



2.1 Data Loading and Initial Analysis

The dataset was loaded and then analyzed to extract the original sizes of each image before any preprocessing steps. This was achieved using a custom function to fetch the image size, and the sizes were subsequently counted and displayed as 256x256 pixels for all 54,303 images. The image sizes ranged from a minimum of 5.07 KB to a maximum of 56.94 KB, with a standard deviation of 8.64 KB. All images were normalized to the [0,1] range, and no duplicates were found (see Appendix 1).

2.2 Class Distribution and Image Size Analysis

The PlantVillage dataset contains 14 crops and 38 classes. However, for some crops, healthy and unhealthy images are missing (Table 1). We focused on the tomato crop, which has one healthy class and nine unhealthy classes (Figure 2), with a total of 18,159 images. Among these, 1,591 are images of healthy leaves, and 16,568 are images of diseased leaves. Tomato images vary in size, with a minimum of 5.07 KB, a maximum of 54.05 KB, a standard deviation of 864 KB, and a mean size of 29.48 KB.

Table 1. Distribution of Healthy and Unhealthy Crop Samples in the Dataset.

Crop	Healthy	Unhealthy	Total
Tomato	1591	16568	18159
Orange	0	5507	5507
Potato	152	2000	2152
Apple	1645	1526	3171
Pepper, _bell	1477	997	2474
Squash	0	1835	1835
Cherry	854	1052	1906
Peach	360	2297	2657
Strawberry	456	1109	1565

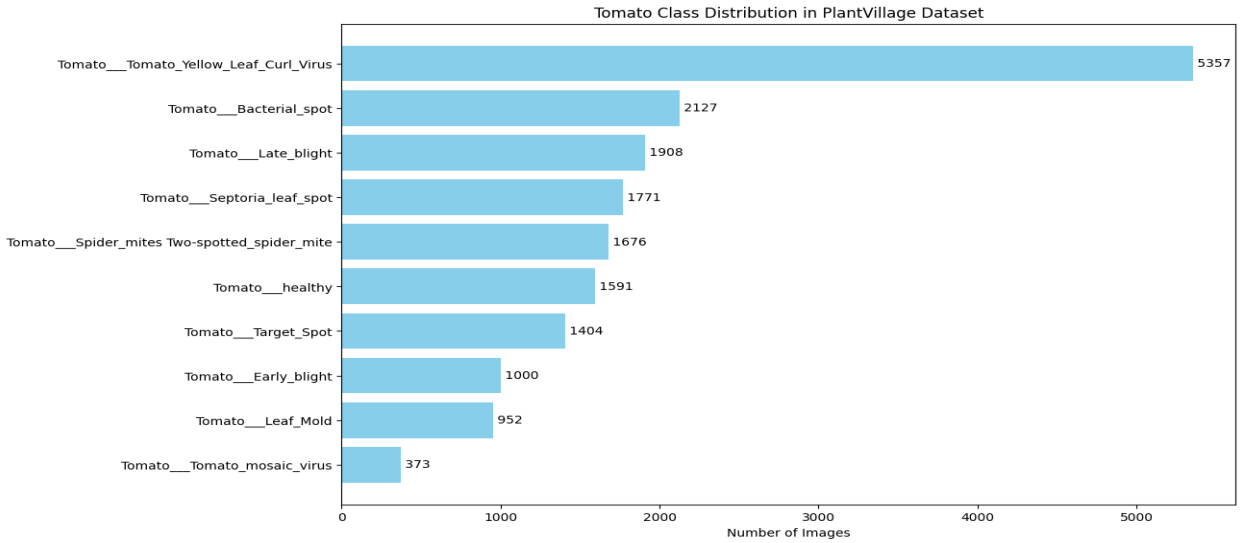
Soybean	5090	0	5909
Corn	1162	2690	3852
Blueberry	1502	0	1502
Grape	423	3639	4062
Raspberry	371	0	371

Figure 2. Ten classes of tomato images, five images per class.

The class distribution within the dataset was analyzed to ensure a balanced representation of each category. The results indicated an unbalanced dataset, with a diverse set of classes represented by varying numbers of images.

For instance, the 'Tomato___Tomato_Yellow_Leaf_Curl_Virus' class had the highest count of images (5,357), while the 'Tomato___Tomato_mosaic_virus' class had the fewest, counted at 373 images, shown in Figure 3 below.

Figure3. Tomato class distribution in PlantVillage Dataset.



3. Methodology

Our approach began with loading the PlantVillage dataset, focusing specifically on tomato images. These images underwent preprocessing, including resizing and normalization, followed by various feature extraction techniques like color histograms, color-based segmentation, edge detection, and texture extraction using Haralick descriptors. Additionally, deep features were extracted using the ResNet50 model. The extracted features were then input into classifiers - Simple Perceptron, Random Forest, and Logistic Regression - to categorize the tomato images (see Appendix 2).

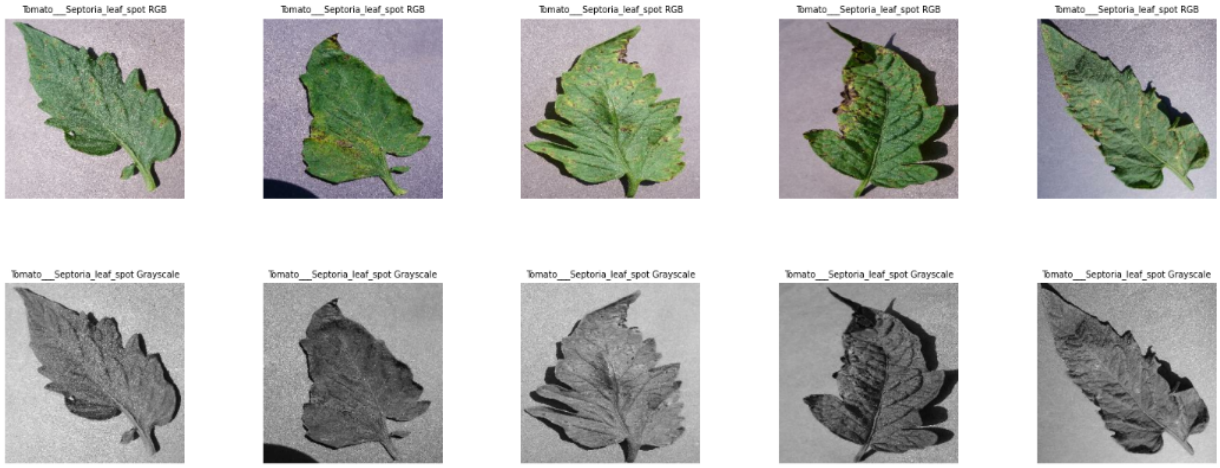
3.1 Data Preprocessing

The preprocessing steps included resizing all images to a standard dimension of 256x256 pixels and normalizing pixel values to the [0, 1] range (see Appendix 1). This normalization was essential to ensure consistent input for the subsequent machine learning models.

3.1.1 Grayscale

Grayscale conversion was considered to reduce computational complexity and focus on the shape and texture of leaves. However, it was clear that grayscale (Figure 4) was not suitable as many diseases often manifest through color changes, which are lost in grayscale.

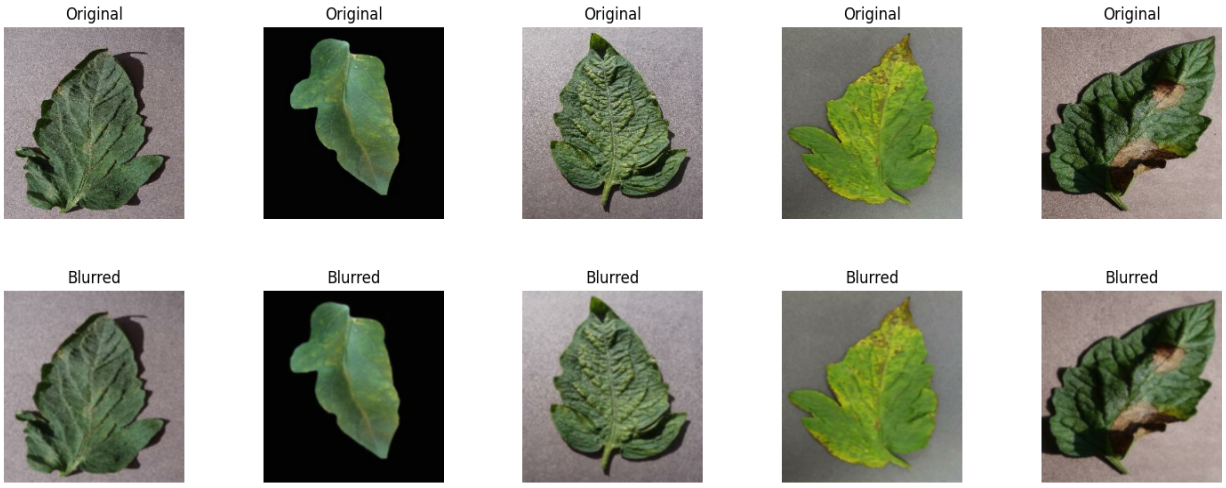
Figure 4. Original and grayscale tomato class images, showing the potential impact of preprocessing on color recognition.



3.1.2 Gaussian Filter

Gaussian filtering was applied to the colored images to test its effectiveness. This method smooths out noise and finer details, which can help in reducing the impact of minor variations and focus on larger features. However, Gaussian filtering was found to worsen visibility of disease-related features, as shown in Figure 5. Disease identification depends not only on color variations and texture details but also on shapes and structures in images. Many diseases affect the edges of leaves, causing them to curve inward, and the application of the Gaussian filter smoothed such effects, making them no longer visible. In addition, as mentioned earlier, disease alters color; some spots can be found on the edges of leaves, and the Gaussian filter could have an impact on them, thereby smoothing away valuable information for the classifier.

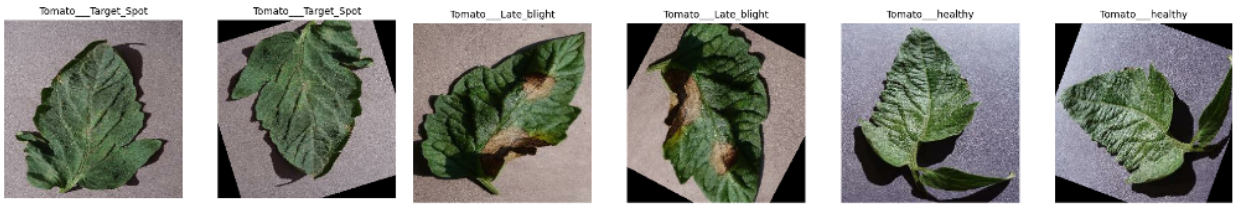
Figure 5. Gaussian filter: random sample of original and blurred tomato images.



3.1.3 Rotation

One of the preprocessing techniques we considered was rotation. This technique aimed to help the classifier learn from leaves oriented in different directions, simulating real-world scenarios where leaves are rarely perfectly vertically aligned. The rotation was intended to increase the robustness of the model by exposing it to a variety of leaf orientations and thus improving its generalization capability. However, implementing rotation introduced several complications. Many of the original images in the PlantVillage dataset were taken with the leaves placed on colored paper backgrounds. When these images were rotated, black areas appeared in the corners of the newly generated images (Figure 6). These black areas were artifacts of the rotation process, caused by the rectangular nature of the image and the absence of image data outside the original bounds. This introduced unwanted variability in the background, which could confuse the classifier and potentially degrade its performance. This challenge could have been overcome by uniformly darkening all backgrounds. Moreover, the PlantVillage dataset is already large, containing 54,303 images. Adding even a single rotation for each image would have effectively doubled the size of the dataset. This increase in data volume would have posed significant computational challenges, including the need for increased storage space, longer processing times, and greater computational power for both training and inference stages. Considering these factors, we decided not to proceed with the rotation approach. Instead, we focused on other data augmentation and preprocessing techniques that did not introduce such significant complications or computational overhead. This decision was made to ensure that our preprocessing pipeline remained efficient and that the quality and consistency of the dataset were maintained.

Figure 6. One random rotation: image depicts original and concatenated rotated image.



3.2 Simple Feature extraction

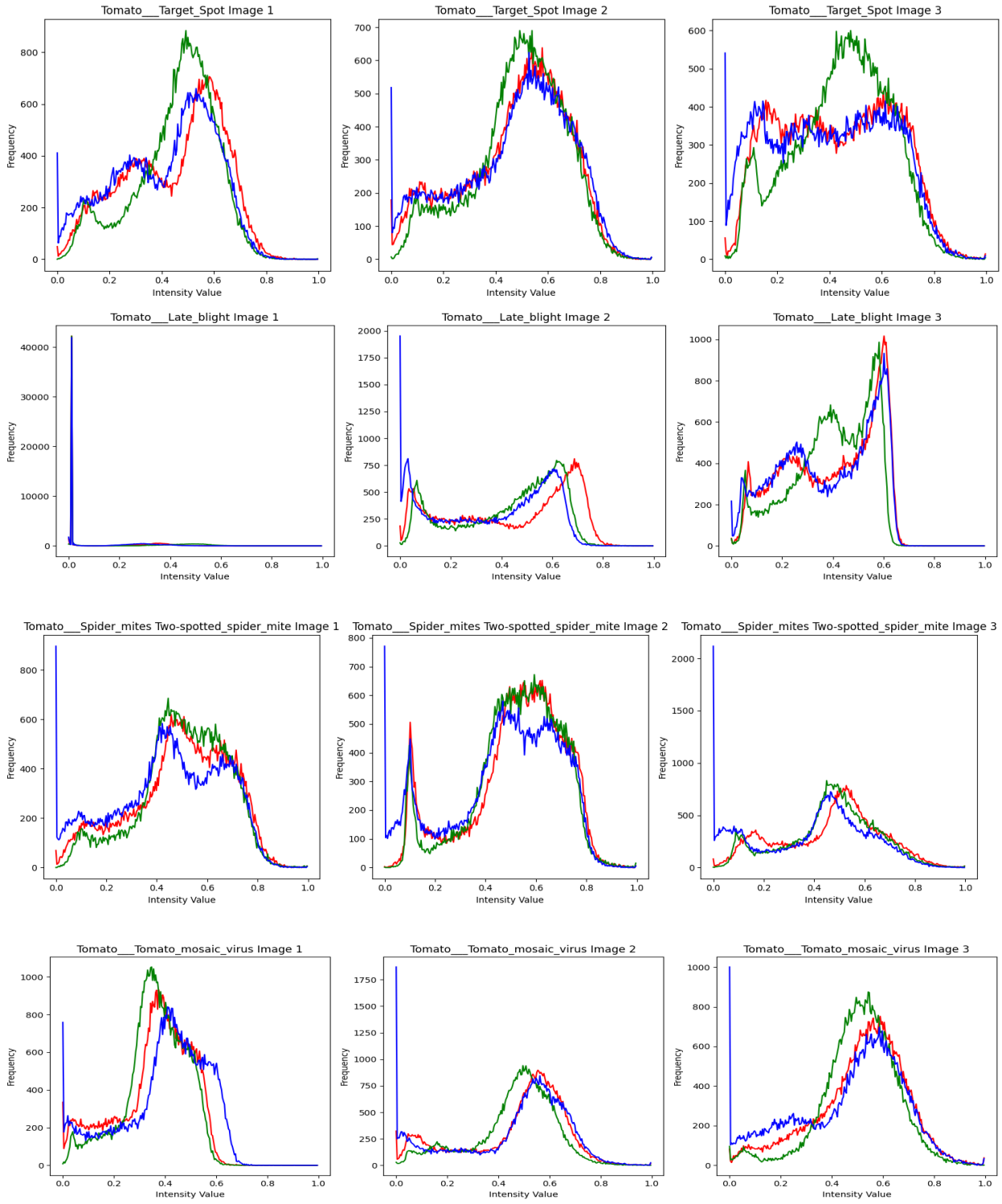
Simple feature extraction from images is essential as it transforms raw pixel data into meaningful representations that machine learning models can efficiently process. This process highlights crucial aspects like edges, textures, and shapes, enabling better pattern recognition and classification, which improves the model's ability to understand and interpret visual information accurately.

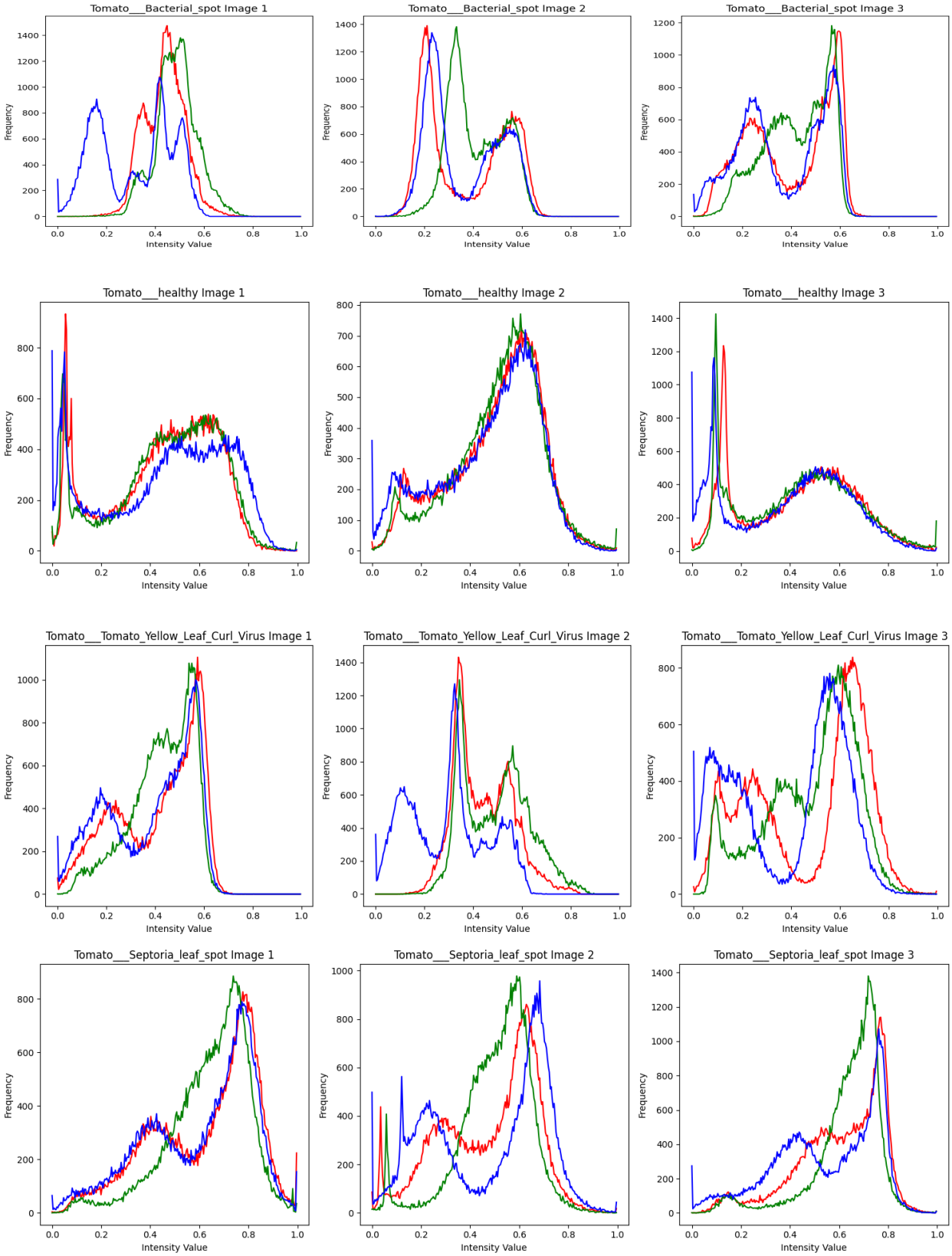
3.2.1 Color Histogram

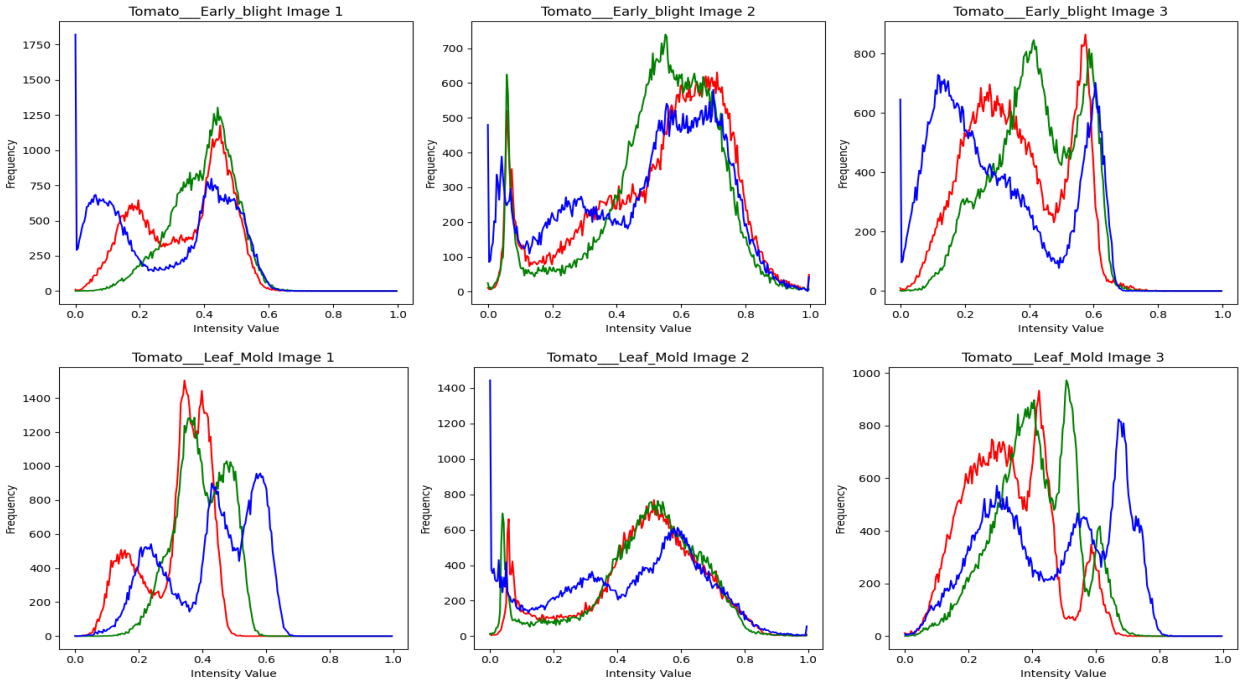
Color histograms represent the RGB (Red, Green , Blue) color distributions in images (Figure7), useful for distinguishing healthy leaves from diseased ones based on color variations indicative of different disease symptoms. The histograms for various tomato leaf diseases reveal distinct color patterns and intensity distributions, aiding in their identification based on visual features, as

shown in Figure 7. For ‘Tomato___Target_Spot’, the histograms show peaks around the mid-intensity range (0.4 to 0.6), indicating intermediate shades with a predominance of green hues due to the natural color of healthy leaves mixed with spots. ‘Tomato___Late_blight’ histograms show a significant spike at the lowest intensity value in the blue channel for image 1, suggesting a dark background, while images 2 and 3 display peaks around mid to high intensity values (0.4 to 0.6) across all channels, indicating a mix of shades typical of blight symptoms. The ‘Tomato___Spider_mites_Two_spotted_spider_mite’ histograms show peaks in the low to mid-intensity range (0.2 to 0.4) with a notable spike at low intensity in the blue channel, indicating dark regions or background, and a mix of green and red, possibly due to spider mite damage. ‘Tomato___Bacterial_spot’ histograms display distinct peaks around mid-intensity values (0.3 to 0.6) with noticeable sharp peaks in all channels, suggesting well-defined spots or lesions altering the leaf's natural color. For ‘Tomato___healthy’, the histograms show smooth distributions with less pronounced peaks, indicating a uniform color distribution typical of healthy leaves, with peaks around 0.4 to 0.6 reflecting the natural green hue. The ‘Tomato___Tomato_mosaic_virus’ histograms demonstrate peaks in the mid-intensity range (0.4 to 0.6) with broader distributions across all channels, suggesting a mix of healthy and discolored regions typical of mosaic virus infections. The ‘Tomato___Tomato_Yellow_Leaf_Curl_Virus’ histograms show peaks in the mid-intensity range (0.4 to 0.6) with sharper peaks in the green channel, indicating yellowing or curling regions mixed with healthy green areas, characteristic of leaf curl virus symptoms. ‘Tomato___Septoria_leaf_spot’ histograms display distinct peaks around mid to high intensities (0.4 to 0.8), suggesting significant leaf discoloration with well-defined spots or lesions typical of Septoria leaf spot. The ‘Tomato___Early_blight’ histograms exhibit peaks in the mid to high intensity range (0.4 to 0.6) with noticeable variations in the red and green channels, indicating characteristic blight symptoms such as lesions and discoloration. Lastly, ‘Tomato___Leaf_Mold’ histograms show peaks in the mid-intensity range (0.4 to 0.6) with higher frequencies in the red and green channels, suggesting moldy or decayed regions affecting the natural color of the leaves. These distinct color patterns and intensity distributions facilitate the identification of different tomato leaf diseases based on visual features.

Figure 7. Tomato leaves color histogram for all 10 classes.




Tomato___Bacterial_spot Image 1
Tomato___Bacterial_spot Image 2
Tomato___Bacterial_spot Image 3
Tomato___healthy Image 1
Tomato___healthy Image 2
Tomato___healthy Image 3
Tomato___Tomato_Yellow_Leaf_Curl_Virus Image 1
Tomato___Tomato_Yellow_Leaf_Curl_Virus Image 2
Tomato___Tomato_Yellow_Leaf_Curl_Virus Image 3
Tomato___Septoria_leaf_spot Image 1
Tomato___Septoria_leaf_spot Image 2
Tomato___Septoria_leaf_spot Image 3
Frequency
Intensity Value



3.2.2 Edge Detection and Color Augmentation

A color-based segmentation technique was applied to highlight regions of interest, focusing on color ranges associated with potential disease symptoms, specifically red, brown, and yellow areas. This segmentation masked out the background by retaining only pixels within the specified color bounds. Subsequently, edge detection was performed on the segmented images to emphasize the boundaries of disease-affected areas. The edge detection process involved computing Sobel gradients and determining the gradient magnitude, which highlighted significant edges within the images. The effectiveness of this preprocessing pipeline is illustrated in Figure 8, showcasing a subset of the processed images, with highlighted edges and segmented regions that aid in disease identification.

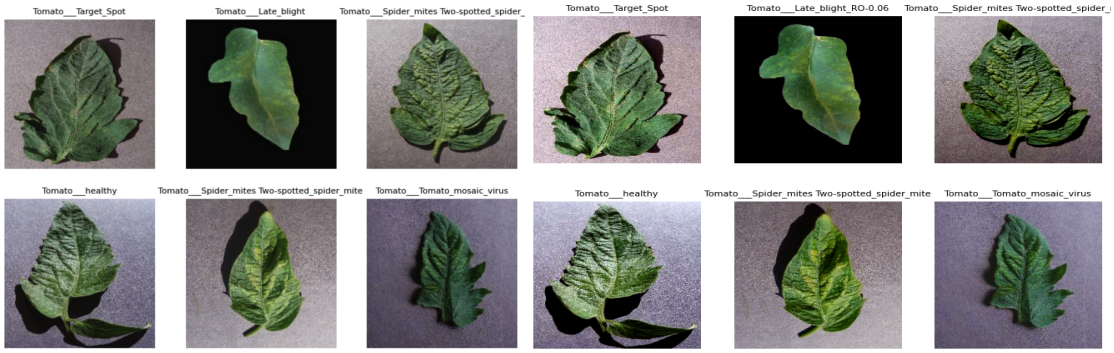
Figure 8. Tomato Leaves with Color Augmentation and Sobel Edge Detection for Disease Identification.

3.2.3 HOG and LBP

In addition to the previous simple feature extractions, we also conducted a parallel comparative analysis of HOG and LBP features, which revealed critical insights into their effectiveness for disease identification in tomato plants (Figure 9). HOG features highlighted the distribution of edges and textures, showing how pronounced edge structures could indicate significant texture changes due to disease. In contrast, low or zero HOG values suggested minimal texture variation. LBP features complemented this by providing detailed information about texture patterns, with higher values indicating more frequent and intricate textures. Together, these features offer a robust method for distinguishing between different disease patterns and healthy leaves, potentially enhancing the model's classification accuracy and reliability in identifying various tomato diseases.

Although we performed analyses using HOG and LBP features, they were not included in the final project. This exploration was conducted to investigate alternative features that could have improved the model, but the results did not warrant their integration into the final pipeline.

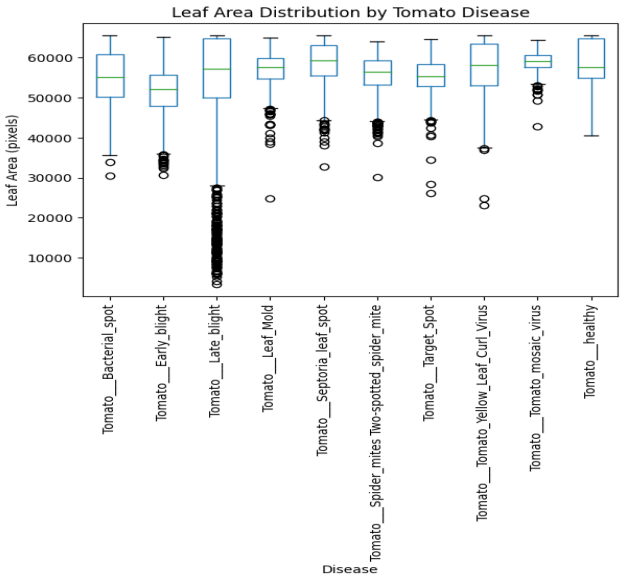
Figure 9. Comparative Analysis of LBP and HOG Features for Disease Identification in Tomato Plants.



3.2.4 Area Calculation

We calculated the area of each leaf by converting segmented images to grayscale, applying a binary threshold to distinguish leaf pixels from the background, and summing the binary mask values to quantify the leaf area in pixels. To explore the relationship between tomato diseases and leaf area, we generated a boxplot (Figure 10) displaying the distribution of leaf areas across different tomato diseases. The plot visually represents the variability in leaf size associated with each disease, providing insights into how certain diseases might affect leaf growth.

Figure 10. Leaf Area Distribution by different classes of Tomato.



We observed that for some diseases, such as 'Tomato___Early_blight' and 'Tomato___Late_blight', there is a great variability in leaf size, as indicated by the wider boxes and longer whiskers. This suggests that these diseases may cause more heterogeneous effects on leaf growth. In contrast, classes like 'Tomato___Tomato_Yellow_Leaf_Curl_Virus' and 'Tomato___Spider_mites Two-spotted_spider_mite' show narrower distributions, indicating more uniform leaf sizes within these disease categories.

The presence of outliers in many classes suggests that some leaves may exhibit variances in area, potentially due to varying stages of disease progression or differences in how the images were captured. In addition, there are more than 10,000 varieties of tomatoes (Johnny's Selected Seeds), reflecting the plant's genetic diversity and adaptability to various climates and uses. These varieties can have both similar and different leaf characteristics depending on their type and specific traits, as shown in Figure 2. We determined that calculating the leaf area of tomato plants without considering the specific variety could lead to inaccuracies in modeling. Different tomato varieties have varying leaf shapes and sizes, which can introduce noise into the data, reducing the accuracy of models. For instance, models might misinterpret leaf area differences as indicators of disease or health issues, rather than simply reflecting natural variations in leaf structure. This could cause poor generalization and inaccurate predictions. The PlantVillage dataset does not specify the tomato varieties (nor the varieties of other included crops), further complicating the analysis.

3.3 Complex Feature extraction

For our final feature set, we chose to include two complex features: Haralick Texture features, and ResNet50 features.

3.3.1 Haralick

Haralick features are statistical measures of texture that describe the arrangement and properties of pixels in an image. The texture of the leaves often changes due to the disease, so we hoped that these features would be useful in capturing patterns that may be indicative of disease.

Our first complex feature set was created through extracting the Haralick texture features from our processed images. The Haralick feature set was created by first applying the simple features to the image (color histogram, color based segmentation and edge detection) before running the Haralick extraction code on each color channel to get our feature set. For each channel, we computed 5 Haralick features: contrast, dissimilarity, homogeneity, energy, and correlation. The final set is a vector of length 15.

3.3.2 ResNet50

For our second complex feature set, we used the ResNet50 model as a feature extractor. This was done by removing the final fully connected layer of the pre-trained ResNet50 model and using the 2048-dimensional embedding vector as the features for our classifiers. The pre-trained weights of ResNet50 were learned during extensive training on the ImageNet dataset, which contains over 14 million images from a wide variety of domains. This training process allows ResNet50 to learn a wide range of features that are typically generic enough to be useful for many image classification problems, as many visual concepts are shared across different types of images, such as edges, textures, and shapes. By leveraging these pre-learned features, we can take advantage of deep learning techniques without the need to train a large CNN model from scratch. We hope that these features will translate well for classifying plant diseases as well.

4. Model

After extracting Haralick and ResNet50 features, these distinct feature sets were combined to form a comprehensive 2063-dimensional representation of each image. Haralick features provided detailed texture information joint with the information gained from the simple features applied beforehand, while ResNet50 features contributed deep, abstract visual representations. The combined feature vectors were then normalized to ensure dataset uniformity.

To prepare for classifier training, the combined feature vectors and their corresponding labels were shuffled to guarantee a representative distribution across all splits. The dataset was then divided into training (70%), validation (15%), and test (15%) sets. This stratified splitting approach aimed to optimize model performance and enhance generalization to unseen data. The training set was used for model fitting, the validation set for hyperparameter tuning and preventing overfitting, and the test set for evaluating the final model's performance.

4.1 Classifiers

4.1.1 Logistic Regression

For our first classifier, we chose the Logistic Regression. The Logistic Regression model was designed to classify tomato plant diseases based on the extracted features from the dataset. The architecture of the Logistic Regression model is straightforward, consisting of a linear combination

of the input features followed by the application of the logistic function to predict the probability of each class. This model was trained using the 'lbfgs' solver, which is well-suited for handling multiclass classification problems and can efficiently optimize the logistic regression objective function. The regularization technique applied was L2 regularization, also known as ridge regularization, which helps prevent overfitting by penalizing large coefficients in the model. The strength of this regularization was controlled by the inverse of the regularization parameter, 'C', which was tuned during the model development process. In the final model, a 'C' value was chosen that provided a good balance between model complexity and classification performance. The model was trained for 10 epochs, during which it showed gradual improvement in both training and validation accuracy over time. The multi-class classification was handled using the "one-vs-rest" approach, where a separate binary classifier is trained for each class, treating the class as positive and the others as negative. The probabilities from these binary classifiers were then combined to produce the final multiclass prediction. The model was evaluated using standard metrics, including accuracy, precision, recall, and F1-score, and it was found to provide competitive performance in identifying various tomato plant diseases.

4.1.2 Simple Perceptron

For our second classifier, we chose to use a simple perceptron model. A simple perceptron is a two layer neural network with no hidden layers, including only an input layer and output layer. This model is limited in its ability to handle complex patterns and cannot learn non-linear relationships between inputs and outputs.

After experimenting with hyperparameters, we ended up with our architecture for our final model. The model had an input layer of size 2063 and an output layer of size 10. We trained the model on the categorical cross-entropy loss, using the Adam optimizer with a learning rate of 0.025, a batch size of 32, and 20 epochs.

4.1.3 Random Forest

For our third classifier, we implemented a Random Forest model, which is an ensemble method combining multiple decision trees. The final model was composed of 100 decision trees (estimators) to aggregate predictions from various trees for more robust results. Random Forest method is particularly powerful because it reduces overfitting by averaging the predictions of the individual trees, which are themselves prone to high variance.

We trained the Random Forest model on our training set (X_{train},y_{train}), and evaluated its performance on the validation set (X_{valid}). The model was optimized using default parameters with 100 trees, and the process was parallelized to improve computational efficiency.

Model performance was evaluated using a confusion matrix and various classification metrics, including precision, recall, and F1-score for each disease class.

4.1.4 Hyperparameter tuning

In order to achieve the maximum performance possible, we performed hyperparameter tuning on our best model, the Random Forest. We used a grid search because it was the simplest method to configure. In order to make the search less computationally expensive, we used sklearn's experimental HalvingGridSearch algorithm. The search strategy starts evaluating all the candidates with a small amount of resources and iteratively selects the best candidates, using more and more resources. We used the macro F1 score computed with 5-fold cross-validation as our scoring metric.

This gave us the following hyperparameters: max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=256. We then retrained a random forest with these hyperparameters on the entire training set. This gave us a model with

This model achieves a weighted average of 0.62, which is the same as the untuned version of the model. However, it does achieve a test accuracy of 0.65 which beats the untuned random forest classifier's test accuracy of 0.64. This is not surprising since it is well-known that random forest classifiers perform very well without any hyperparameter tuning (Probst, P. et al., 2019).

5. Results & Discussion

The Logistic Regression model, despite its simplicity, provided a baseline performance with a test accuracy of approximately 49%. The training accuracy started at approximately 28.17% and increased to 35.41% by the final epoch, while the validation accuracy improved from 30.76% to 35.64%. This gradual improvement indicates that the model struggled to capture complex patterns in the data, as demonstrated by the relatively low accuracy on the test set. The overall precision, recall, and F1-score were 45.25%, 49.41%, and 44.56%, respectively. The model performed best on the 'Tomato___Yellow_Leaf_Curl_Virus" and 'Tomato___healthy' classes, achieving F1-scores of 0.74 and 0.63. However, it struggled significantly with other classes, such as 'Tomato___Leaf_Mold' and 'Tomato___Early_blight', where it achieved F1-scores of 0.00 and 0.07, respectively, failing to make any correct predictions. The overall macro average F1-score was 0.33, indicating that the model had difficulty generalizing across all classes. These results suggest that while Logistic Regression can provide a quick benchmark, it is not well-suited for this multi-class classification problem, particularly when dealing with complex patterns.

Additionally, the Receiver Operating Characteristic (ROC) analysis, as shown in Appendix 3 (Figure A3.3), further illustrates the model's limitations. The ROC curves for different classes varied significantly, with the area under the curve (AUC) ranging from 0.72 for 'Tomato__ Septoria_leaf_spot' to 0.96 for 'Tomato_healthy'. This wide range in AUC values highlights the inconsistent performance of the Logistic Regression model across different classes, reinforcing the conclusion that more sophisticated models are needed for better handling of this dataset (see Appendix 3 for confusion matrix, ROC curves, and detailed performance metrics).

The Simple Perceptron model, which is a basic neural network with no hidden layers, achieved a validation accuracy of 46% and a test accuracy of 46%. Similar to Logistic Regression, the Simple

Perceptron had varying success across different classes. The model performed reasonably well on the ‘Tomato___Yellow_Leaf_Curl_Virus’ class with an F1-score of 0.70, but its performance was notably poor on classes like ‘Tomato___Septoria_leaf_spot’ and ‘Tomato___Target_Spot’, where it failed to capture any meaningful patterns (F1-scores of 0.00). The macro average F1-score was 0.27, indicating that the model did not handle class imbalances well and was unable to learn non-linear relationships effectively. The overall weighted average F1-score of 0.39 shows that while the Simple Perceptron offers slightly better generalization than Logistic Regression, it remains inadequate for more complex multi-class problems.

The ROC analysis, illustrated in Appendix 4 (Figure A4.3), further highlights the model's weaknesses. The areas under the ROC curves (AUCs) varied significantly across the different classes, with ‘Tomato___Yellow_Leaf_Curl_Virus’ achieving an AUC of 0.91, indicating reasonable discriminatory power. However, classes like ‘Tomato___Septoria_leaf_spot’ and ‘Tomato___Leaf_Mold’ had AUCs of 0.78 and 0.74, respectively, suggesting the model's inability to separate these classes effectively. These findings confirm that the Simple Perceptron struggles to model more complex, non-linear relationships in the data.

Furthermore, the SHAP (SHapley Additive exPlanations) analysis revealed insights into feature importance, as shown in Appendix 4 (Figure A4.4). The most influential features were primarily related to the ResNet50 features, particularly ‘ResNet50_feature_2044’ and ‘ResNet50_feature_2034’, which had the highest impact on model output. However, the overall contribution of features was relatively modest. This is likely due to the large number of features (2063 in total), where the impact of individual features tends to be diluted, especially in a simple model like the Perceptron that lacks the capacity to capture complex patterns effectively. Additionally, the high dimensionality may have introduced redundancy and noise, further reducing the effectiveness of each feature. These limitations underscore the need for more sophisticated models to fully exploit the available features, handle high-dimensional data, and achieve better classification performance (see Appendix 4 for for confusion matrix, ROC curves, detailed performance metrics, and SHAP summary plot)

The Random Forest classifier demonstrated the best performance among the three models, with a validation accuracy of 71% and a test accuracy of 65%. This model leveraged its ensemble nature to handle the complexity of the dataset more effectively. The Random Forest achieved an F1-score of 0.86 on the ‘Tomato___Yellow_Leaf_Curl_Virus’ class and maintained relatively high scores across other classes, such as ‘Tomato___healthy’ (F1-score of 0.80) and ‘Tomato___Septoria_leaf_spot’ (F1-score of 0.66). The overall macro average F1-score was 0.62, which is significantly higher than those of the Logistic Regression and Simple Perceptron models. The ability of Random Forest to capture both linear and non-linear relationships, along with its robustness against overfitting, made it the most suitable model for this task.

The ROC analysis, as depicted in Appendix 5 (Figure A5.3), further underscores the Random Forest's superior performance. The ROC curves for all classes show high areas under the curve (AUC), with the ‘Tomato_healthy’ class achieving an AUC of 0.98 and ‘Tomato___Yellow_Leaf_Curl_Virus’ achieving an AUC of 0.97. These high AUC values indicate that the model is highly effective at distinguishing between the classes. Even the more challenging

classes, such as ‘Tomato___Septoria_leaf_spot’ (AUC of 0.92) and ‘Tomato____Target_Spot’ (AUC of 0.94), demonstrate the model's strong discriminatory power across the board.

The confusion matrix (Appendix 5 and 6, Figure A5.1 and A6.1) reveals that the Random Forest model has fewer misclassifications compared to the other models, particularly for the ‘Tomato___Yellow_Leaf_Curl_Virus’ and ‘Tomato___healthy’ classes, which had the highest precision. However, there were still some confusions observed, especially between diseases with similar visual symptoms, such as ‘Tomato___Spider_mites Two-spotted_spider_mite’ and ‘Tomato___Late_blight’. These instances of confusion indicate potential areas where further feature engineering or the inclusion of additional data might help improve the model's performance.

Moreover, the SHAP (SHapley Additive exPlanations) analysis (Appendix 5, Figure A5.4) provided insights into the importance of various features used by the model. The top 10 most influential features, which included texture-based features from the Haralick descriptor applied to simple feature extractions like color histograms, color-based segmentation, and edge detection, were crucial in differentiating between the tomato disease classes. Features such as ‘Haralick_R_contrast’ and ‘Haralick_R_dissimilarity’ had the most substantial impact on the model's decisions, highlighting the importance of texture in classifying leaf conditions. The SHAP analysis also confirmed that while the model used a large number of features, these top features were critical in driving accurate predictions, further validating the effectiveness of the Random Forest approach in this context (see Appendix 5 and 6 for confusion matrix, ROC curves, detailed performance metrics, and SHAP summary plot).

6. Conclusion

This study effectively applies both simple and complex feature extraction techniques to improve the identification and classification of tomato plant diseases. By combining methods such as color histogram, edge detection, and area calculation with more sophisticated techniques like Haralick texture features and ResNet50, our model achieved good accuracy and reliability. Initial challenges with preprocessing methods like grayscale conversion and Gaussian filtering underscored the need to retain color and texture details for effective disease identification. Of the classifiers evaluated, the random forest model performed the best, showing high recall, precision, and F1-scores across different disease classes. These results suggest that leveraging both simple and complex feature extraction methods can enhance the accuracy and reliability of automated disease diagnosis systems, providing a scalable solution to mitigate yield losses and enhancing crop management and contributing to global food security.

Looking ahead, future steps could involve adjusting the dataset to ensure a more balanced representation of each disease class, potentially by collecting additional images, especially from the same tomato varieties, to perform variety-specific analyses. Additionally, while we considered using Local Binary Patterns (LBP) and Histogram of Oriented Gradients (HOG), these techniques were not included in the final model. Incorporating these feature extraction methods could be explored to further enhance classification performance. Furthermore, leveraging deep learning architectures like Convolutional Neural Networks (CNNs) or Visual Transformers could provide

even more powerful tools for disease classification. These advancements would not only refine the model's accuracy but also contribute to creating a more robust and scalable solution for mitigating yield losses, thereby enhancing crop management and supporting global food security.

7. References

Oerke, E.-C. (2006). Crop losses to pests. The Journal of Agricultural Science, 144, 31-43.

Hughes, D. P., & Salathé, M. (2015). An open access repository of images on plant health to enable the development of mobile disease diagnostics. Plant Methods, 11, 51.

Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. Frontiers in Plant Science, 7, 1419.

Probst, P., Wright, M. N, Boulesteix, A. (2019). Hyperparameters and tuning strategies for random forest. Wires Data Mining and Knowledge History, Volume 9, Issue 3, May/June.

Food and Agriculture Organization of the United Nations (FAO). (2020). The State of Food and Agriculture.

United States Department of Agriculture (USDA). (2021). Economic Research Service. Vegetables and Pulses Data.

Pranesh, K. et al. (2021). Plant Disease Detection Using Image Processing and Machine Learning. Cornell University. Computer Science. Computer Vision and Pattern Recognition. arXiv: 2106:10698.

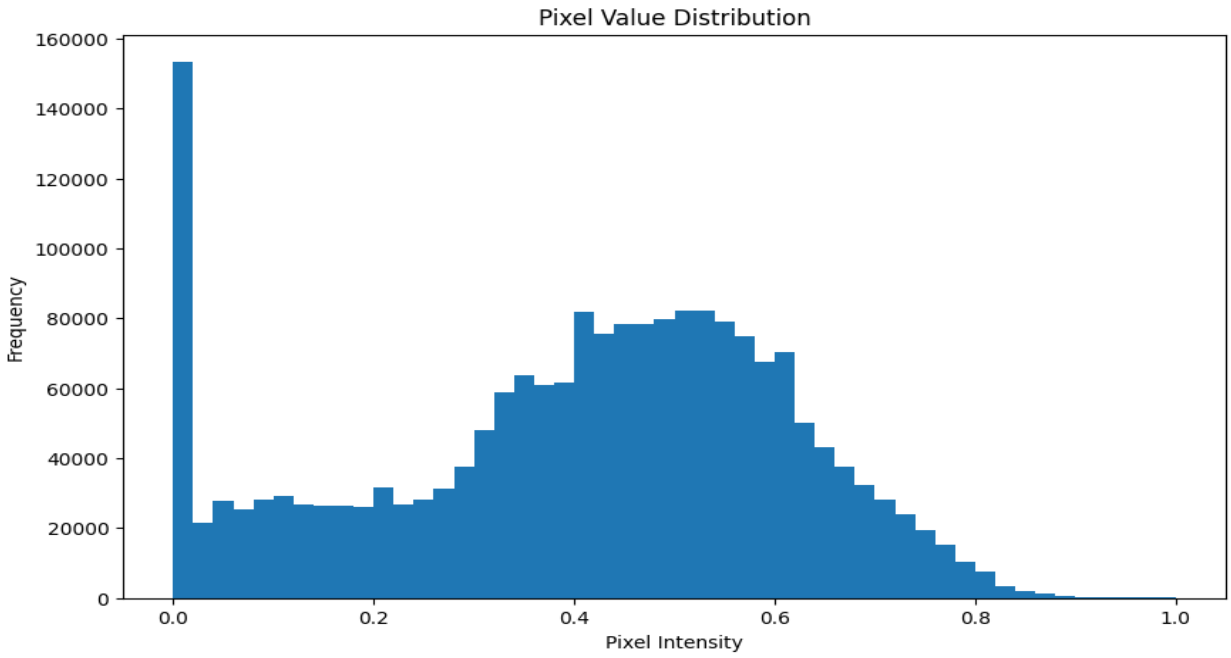
Madhu, K. et al. (2022) Plants Diseases Prediction Framework: A Image-Based System Using Deep Learning. 2022 IEEE World Conference on Applied Intelligence and Computing.

Habiba N. Ngugi, et al. (2024) Revolutionizing crop disease detection with computational deep learning: a comprehensive review. Springer Link, Volume 196, article number 302.

Probst, P. et al. (2019) Hyperparameters and tuning strategies for random forest

Appendix

Appendix 1 - Distribution of Pixel Intensity Values After Image Resizing and Normalization



Appendix 2 - Links to GitHub repository and Google drive

1. GitHub repository (LINK): it contains Notebook with features extractions and generation of X (train, valid, test) and y(train, valid, test)

2. Google Drive (LINK): it contains the (train, valid, test) and y(train, valid, test) npy file.

Appendix 3 - Logistic Regression: confusion matrix, ROC curves, and detailed performance metrics

Figure A3.1: Confusion Matrix for Logistic Regression

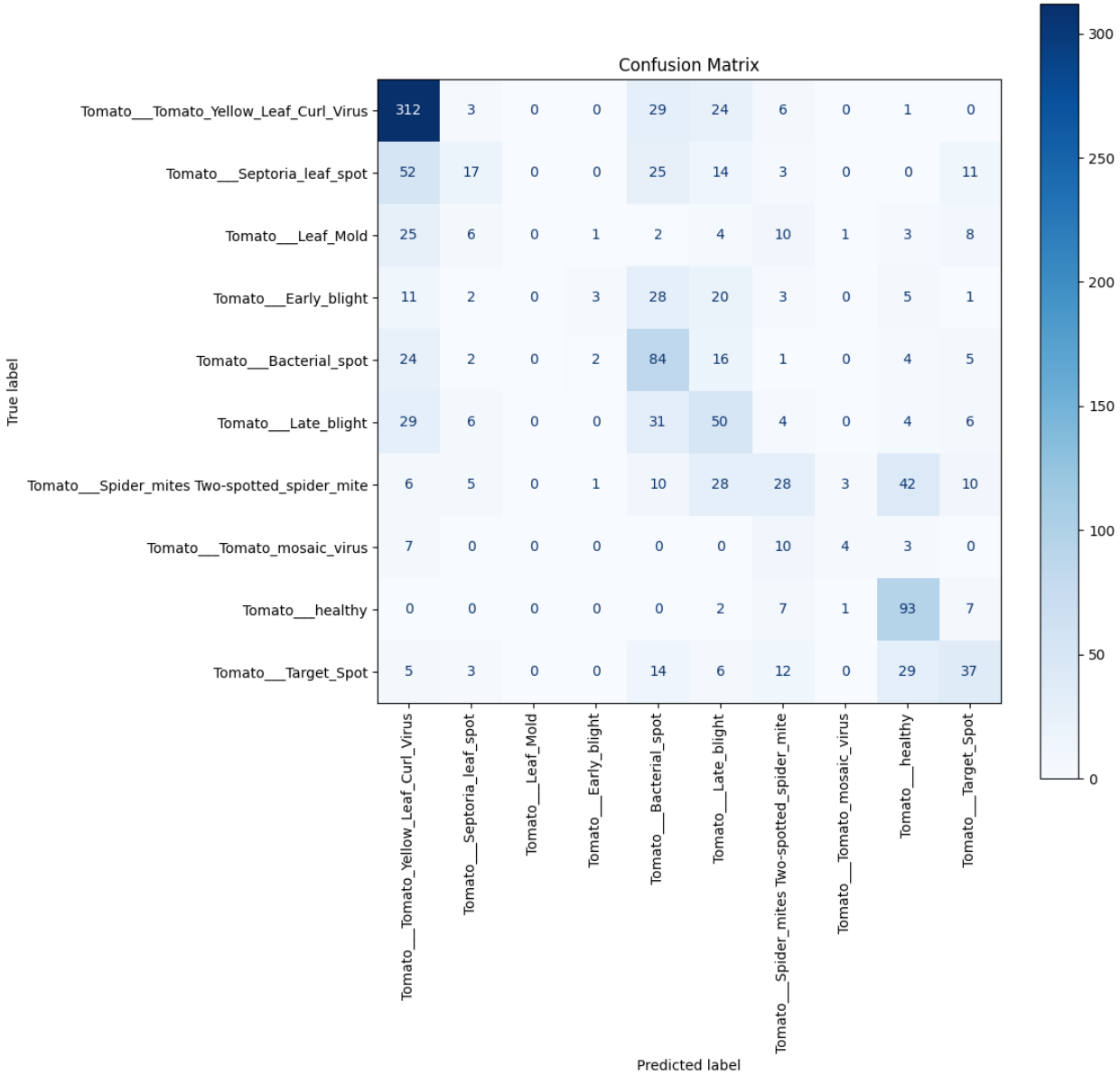


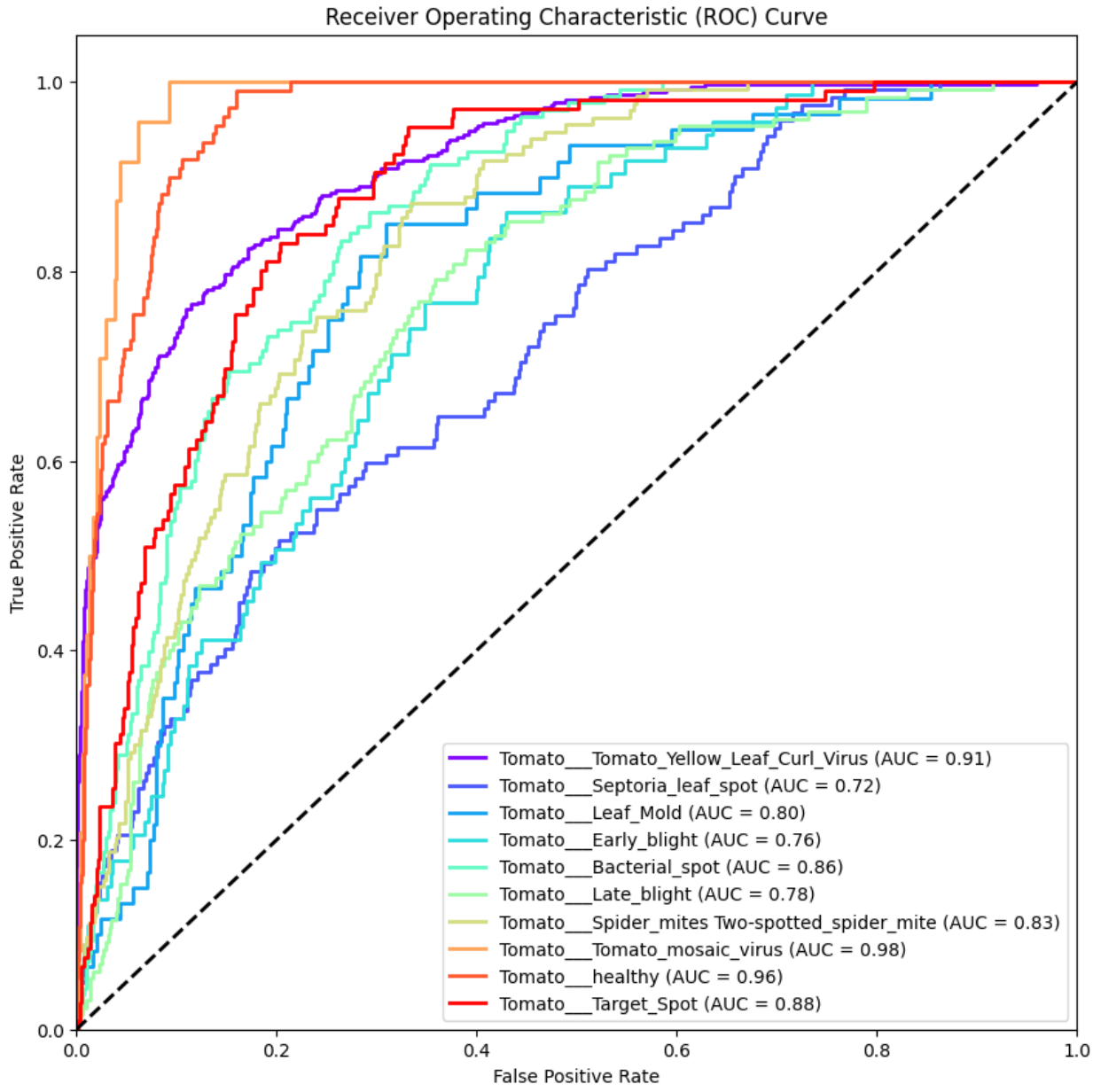
Figure A3.2: Detailed Performance Metrics for Logistic Regression

Logistic Regression - Validation Classification Report

	precision	recall	f1-score
Tomato___Tomato_Yellow_Leaf_Curl_Virus	0.66	0.73	0.84
Tomato___Septoria_leaf_spot	0.39	0.14	0.20
Tomato___Leaf_Mold	0.00	0.00	0.00
Tomato___Early_blight	0.43	0.04	0.07
Tomato___Bacterial_spot	0.38	0.61	0.47
Tomato___Late_blight	0.30	0.38	0.34
Tomato___Spider_mites Two-spotted_spider_mite	0.33	0.21	0.26
Tomato___Tomato_mosaic_virus	0.44	0.17	0.24
Tomato___healthy	0.51	0.85	0.63
Tomato___Target_Spot	0.44	0.35	0.39
accuracy			0.49
macro avg	0.39	0.36	0.33
weighted avg	0.45	0.49	0.45

Validation accuracy	0.49
Test Accuracy	0.45

Figure A3.3: ROC Curve for Logistic Regression



Appendix 4 - Simple Perceptron: confusion matrix, ROC curves, and detailed performance metrics

Figure A4.1: Confusion Matrix for Simple Perceptron

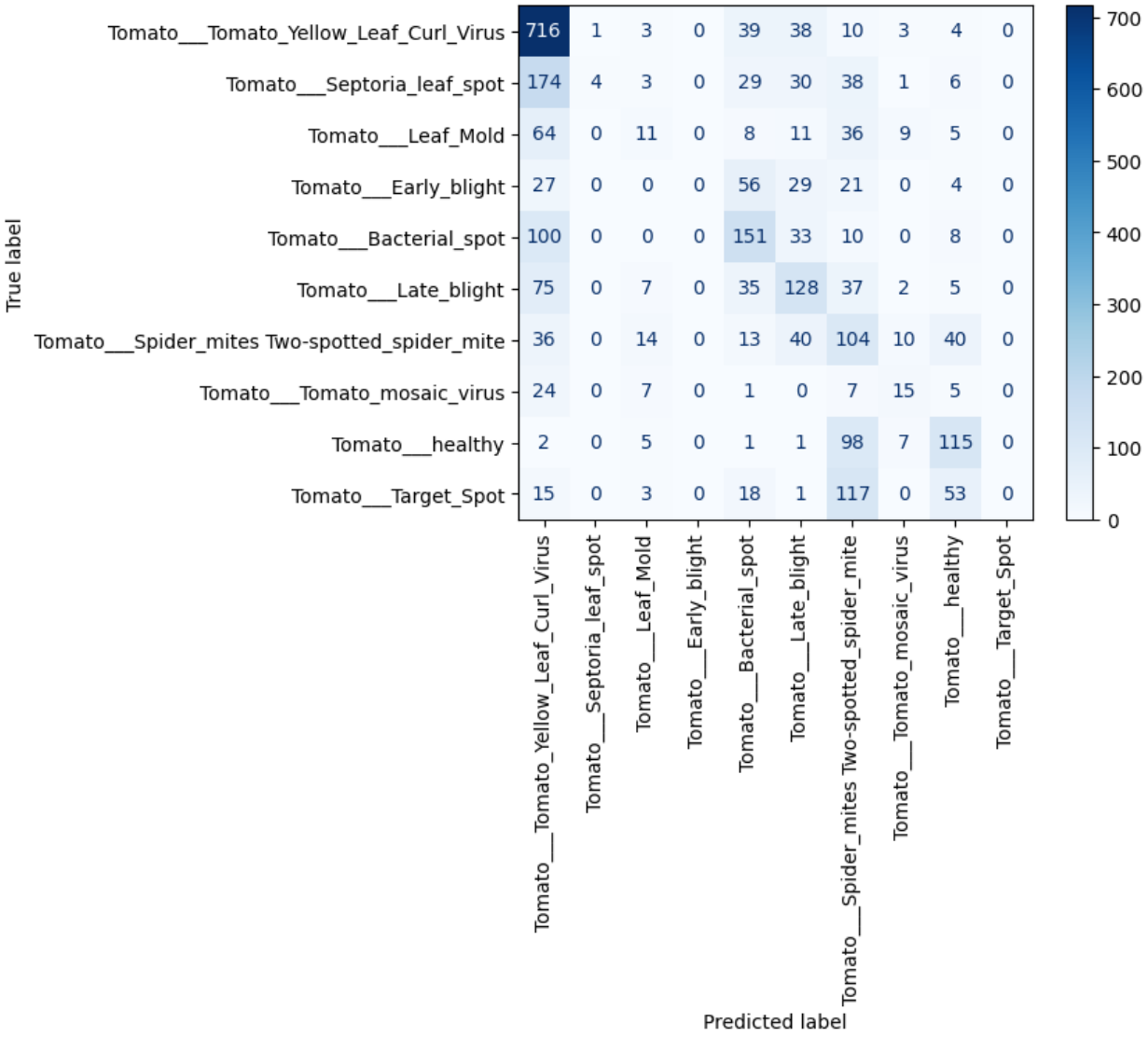


Figure A4.2: Detailed Performance Metrics for Simple Perceptron

Simple Perceptron - Validation Classification Report

	precision	recall	f1-score
Tomato___Tomato_Yellow_Leaf_Curl_Virus	0.57	0.89	0.70
Tomato___Septoria_leaf_spot	0.00	0.00	0.00
Tomato___Leaf_Mold	0.22	0.11	0.14
Tomato___Early_blight	0.32	0.02	0.03
Tomato___Bacterial_spot	0.35	0.66	0.45
Tomato___Late_blight	0.50	0.31	0.38
Tomato___Spider_mites Two-spotted_spider_mite	0.26	0.28	0.27
Tomato___Tomato_mosaic_virus	0.56	0.06	0.11
Tomato___healthy	0.50	0.71	0.58
Tomato___Target_Spot	0.00	0.00	0.00
accuracy			0.47
macro avg	0.33	0.30	0.27
weighted avg	0.37	0.47	0.39

Validation accuracy	0.46
Test Accuracy	0.46

Figure A4.3: ROC Curve for Simple Perceptron

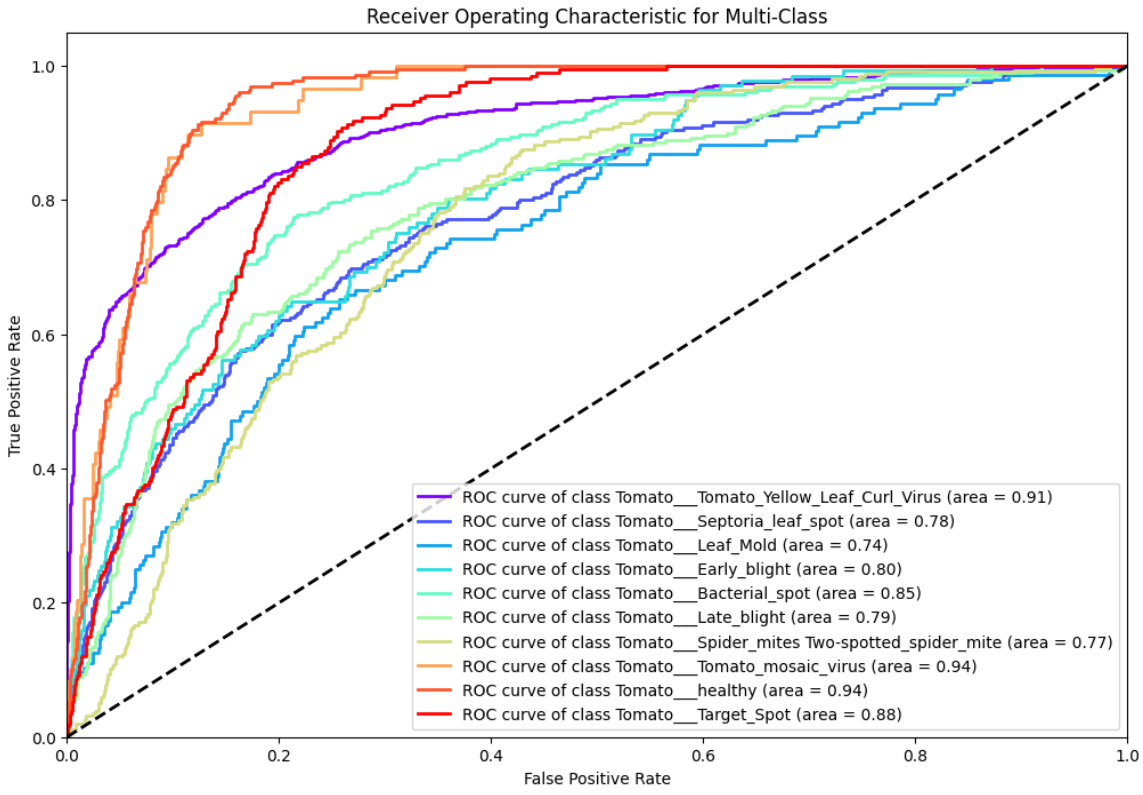
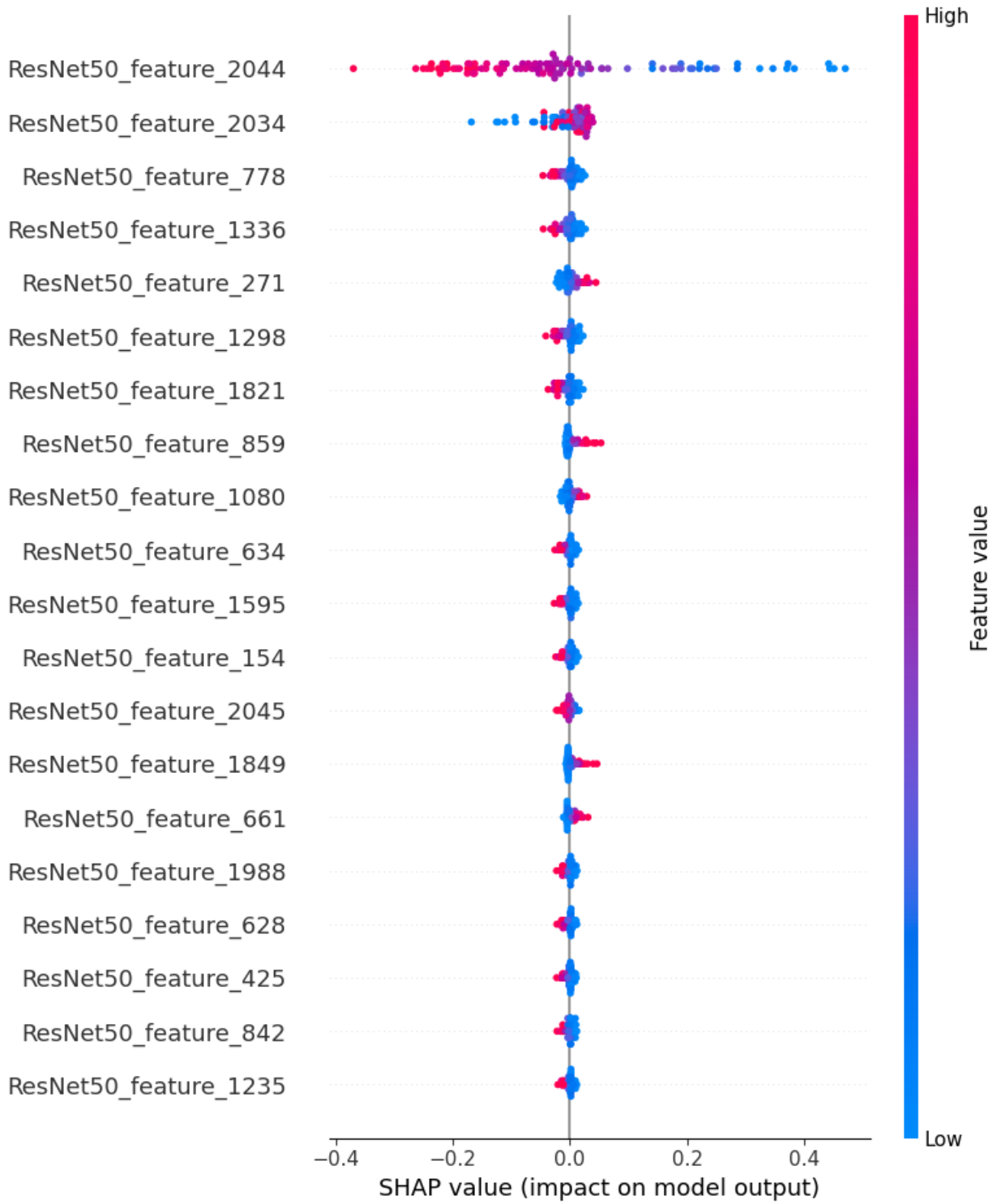


Figure A4.4: SHAP Summary Plot for Simple Perceptron



Appendix 5 - Random Forest: confusion matrix, ROC curves, and detailed performance metrics

Figure A5.1: Confusion Matrix for Random Forest

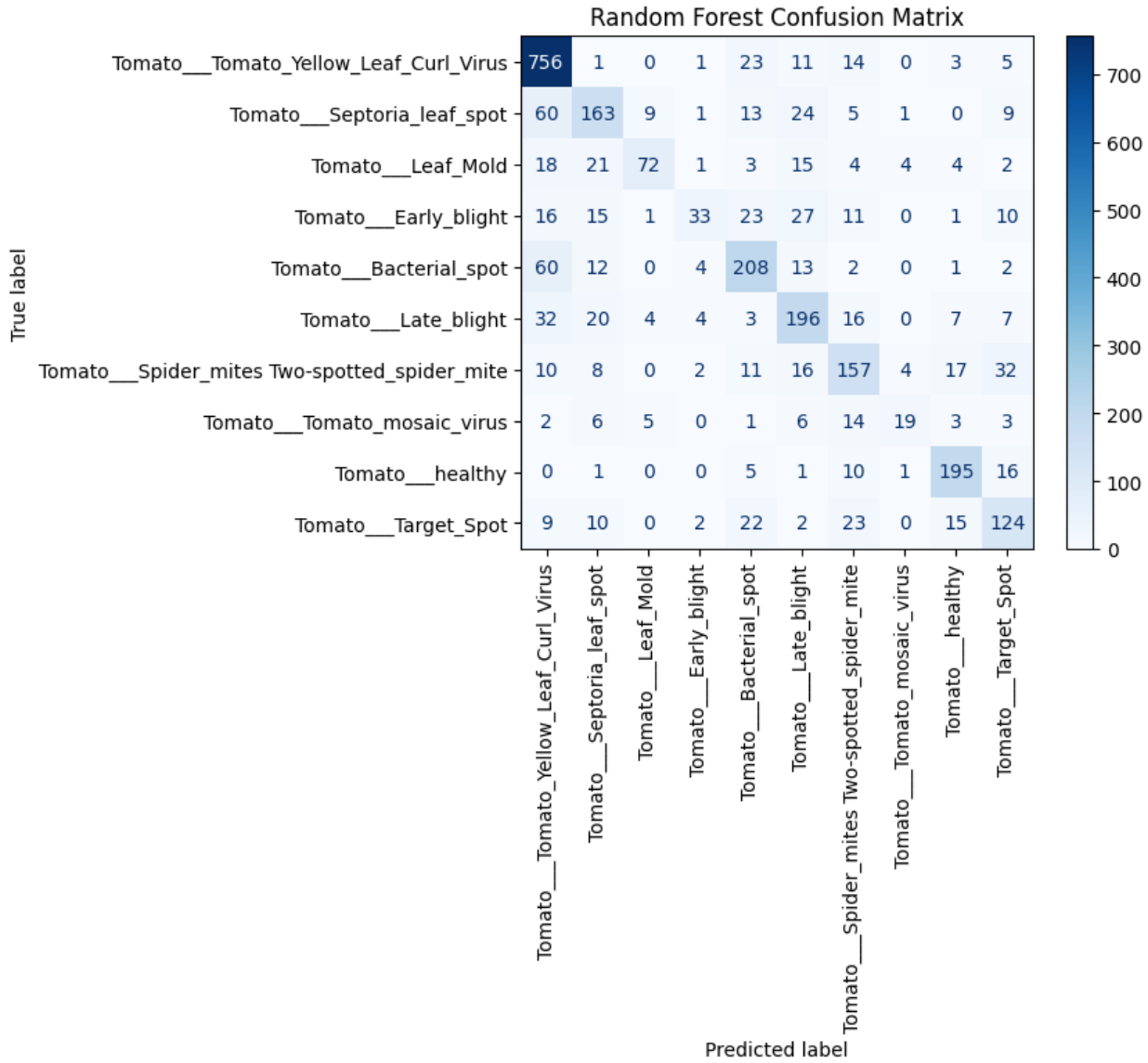


Figure A5.2: Detailed Performance Metrics for Random Forest

Random Forest - Validation Classification Report

	precision	recall	f1-score
Tomato___Tomato_Yellow_Leaf_Curl_Virus	0.79	0.93	0.85
Tomato___Septoria_leaf_spot	0.63	0.57	0.60
Tomato___Leaf_Mold	0.79	0.50	0.61
Tomato___Early_blight	0.69	0.24	0.36
Tomato___Bacterial_spot	0.67	0.69	0.68
Tomato___Late_blight	0.63	0.68	0.65
Tomato___Spider_mites Two-spotted_spider_mite	0.61	0.61	0.61
Tomato___Tomato_mosaic_virus	0.66	0.32	0.43
Tomato___healthy	0.79	0.85	0.82
Tomato___Target_Spot	0.59	0.60	0.59
accuracy			0.71
macro avg	0.68	0.60	0.62
weighted avg	0.79	0.71	0.69

Validation accuracy	0.71
Test Accuracy	0.64

Figure A5.3: ROC Curve for Random Forest

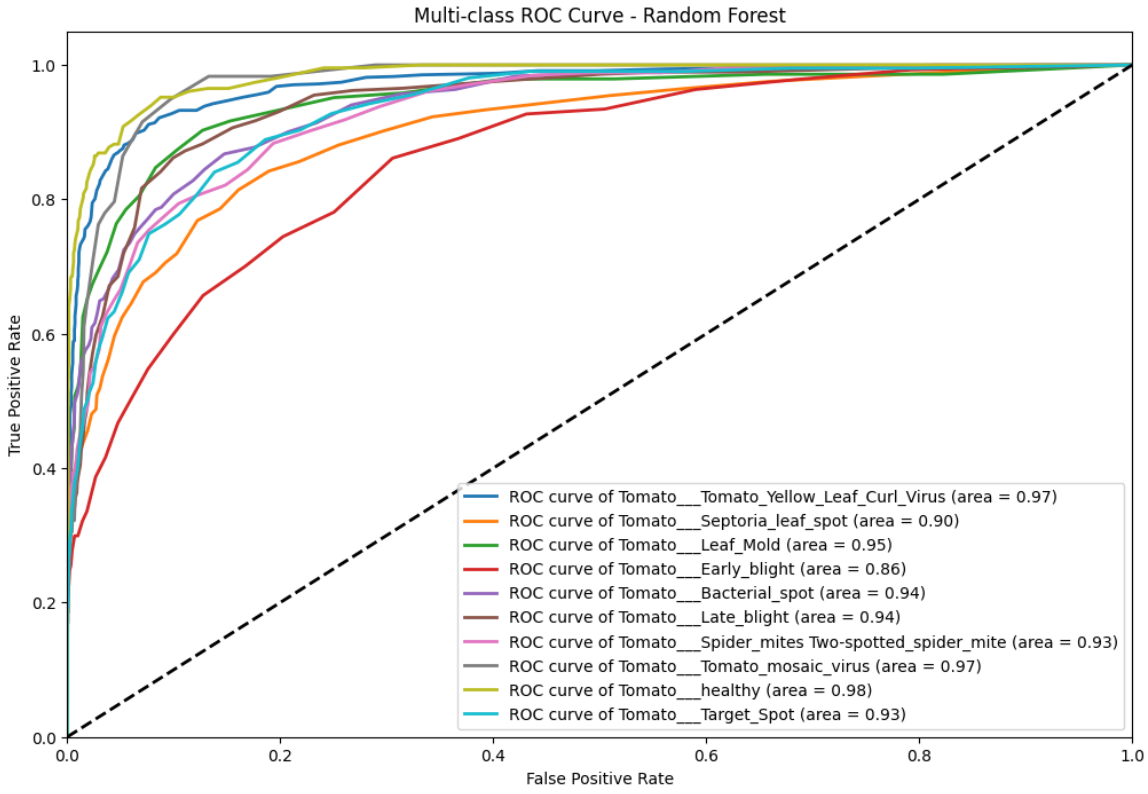
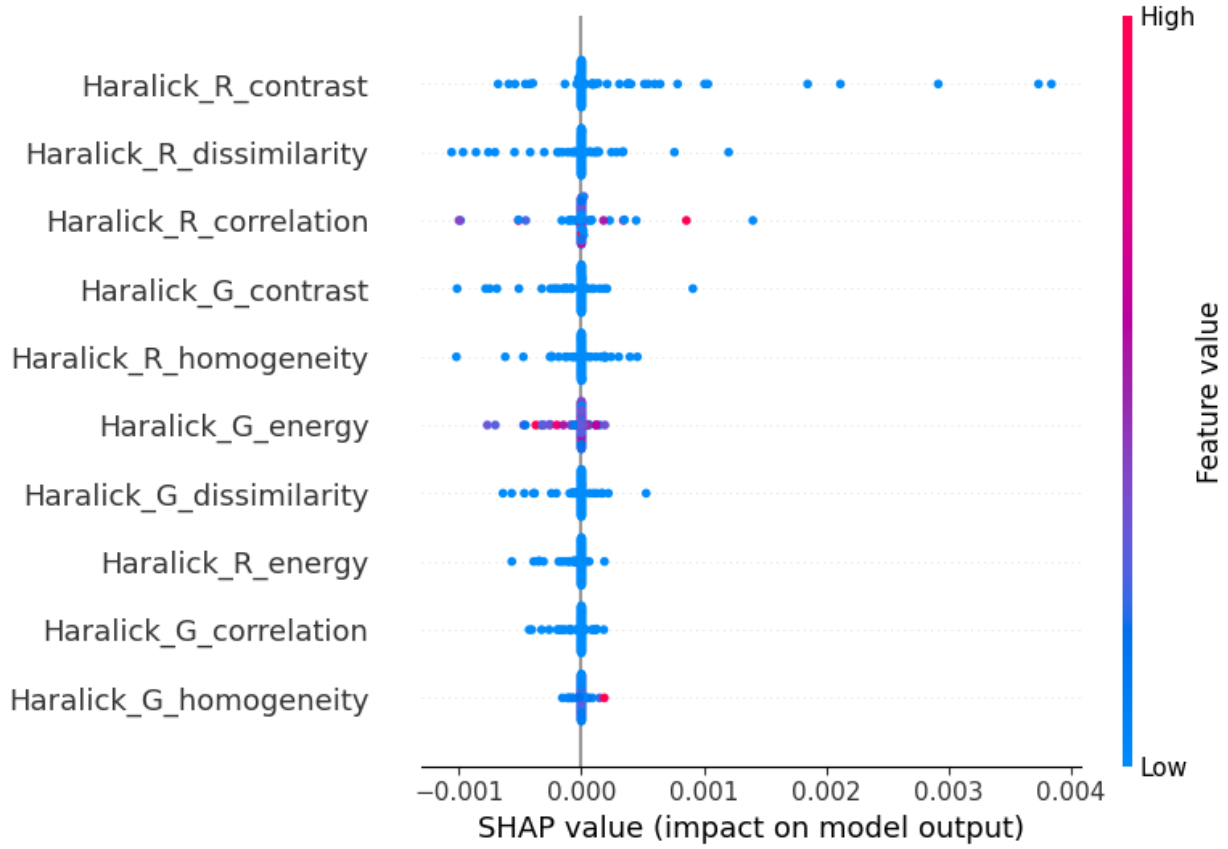


Figure A5.4: SHAP Summary Plot for Random Forest



Appendix 6 - Random Forest (with hyperparameter tuning): confusion matrix, ROC curves, and detailed performance metrics

Figure A6.1: Confusion Matrix for Random Forest

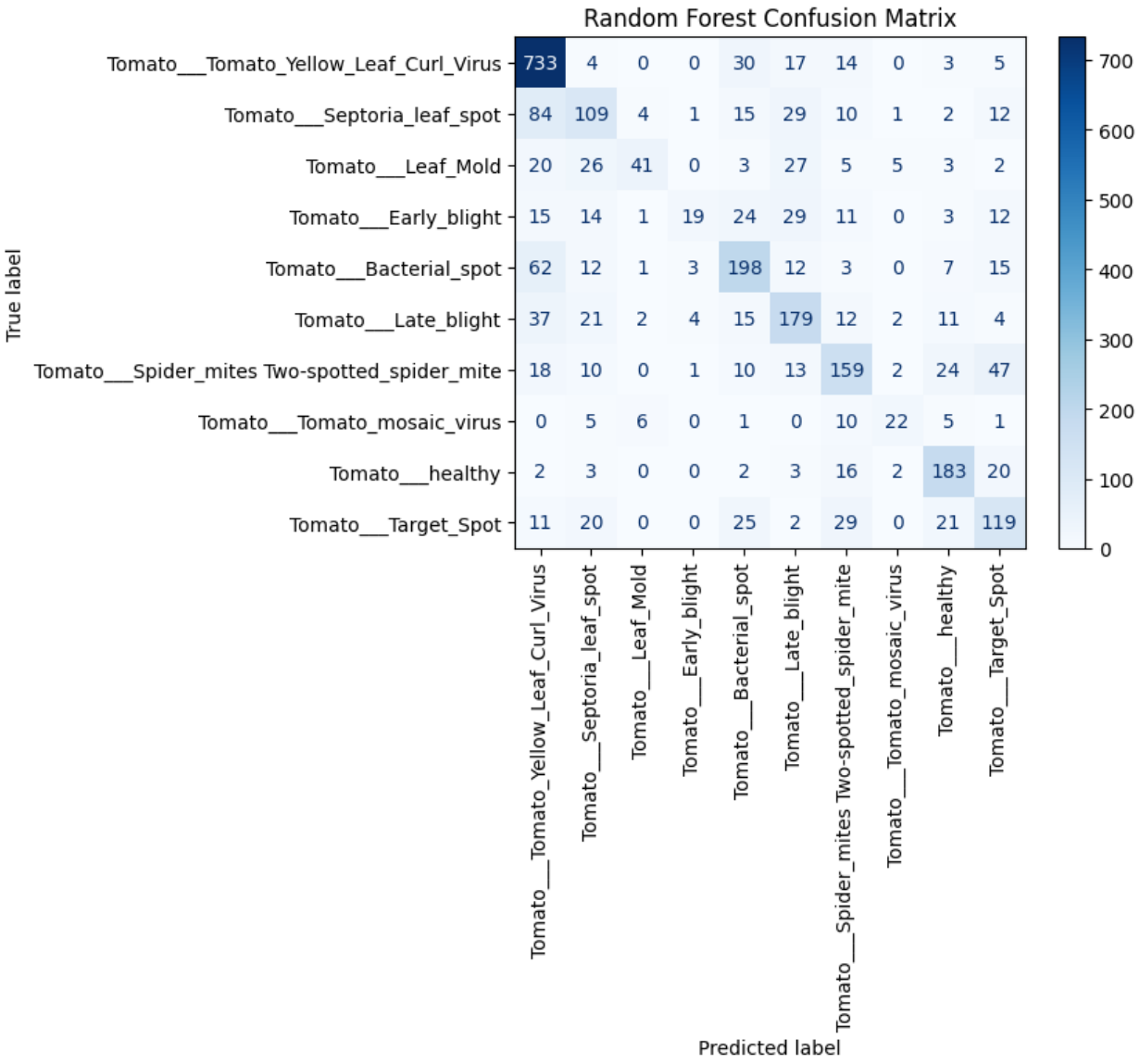


Figure A6.2: Detailed Performance Metrics for Random Forest

Random Forest - Validation Classification Report

	precision	recall	f1-score
Tomato___Tomato_Yellow_Leaf_Curl_Virus	0.79	0.93	0.86
Tomato___Septoria_leaf_spot	0.62	0.55	0.59
Tomato___Leaf_Mold	0.76	0.47	0.58
Tomato___Early_blight	0.85	0.24	0.38
Tomato___Bacterial_spot	0.65	0.68	0.66
Tomato___Late_blight	0.63	0.69	0.66
Tomato___Spider_mites Two-spotted_spider_mite	0.63	0.6	0.62
Tomato___Tomato_mosaic_virus	0.61	0.39	0.47
Tomato___healthy	0.76	0.85	0.8
Tomato___Target_Spot	0.58	0.6	0.59
accuracy			0.71
macro avg	0.69	0.60	0.62
weighted avg	0.71	0.71	0.69

Validation accuracy	0.71
Test Accuracy	0.65

Figure A6.3: ROC Curve for Random Forest

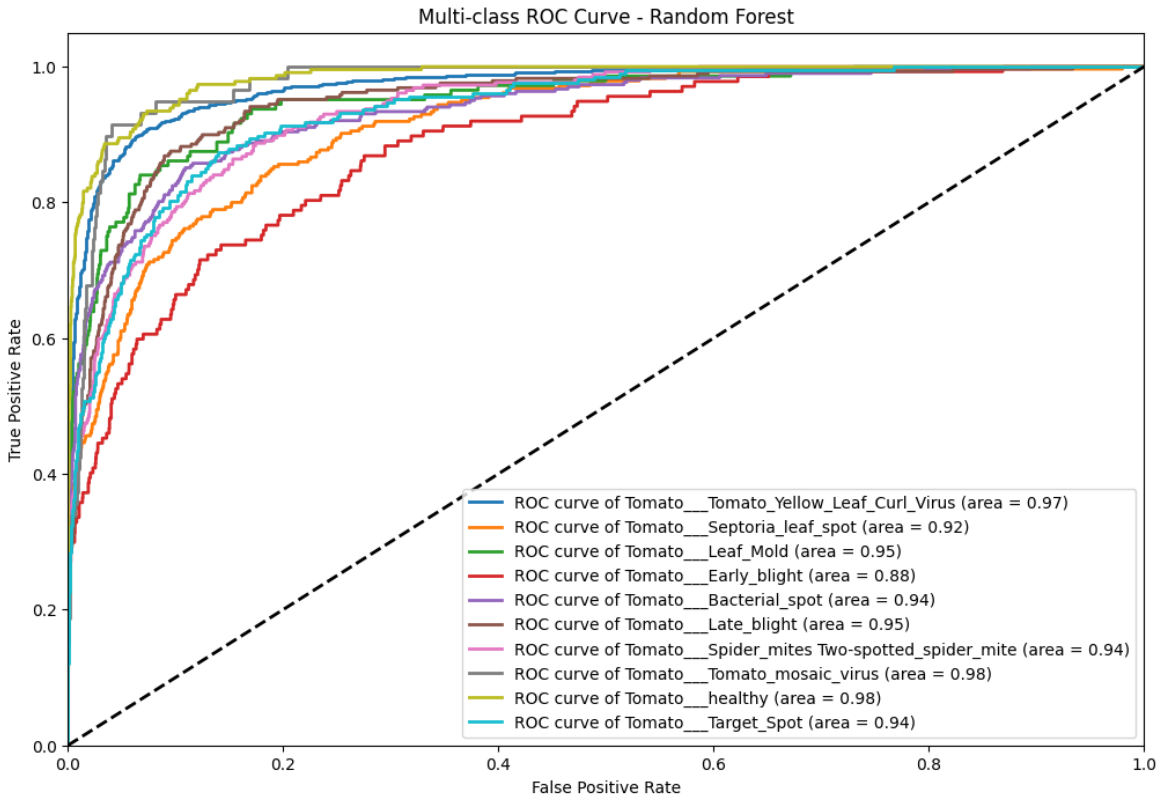


Figure A6.4: SHAP Summary Plot for Random Forest