LEAFDETECTAI: EVALUATING ML AND DL PARADIGMS FOR SUPERIOR LEAF DISEASE IDENTIFICATION

GUDUR SUDHAKARA SANTHOSH

Final Thesis Report

MARCH 2024

ACKNOWLEDGEMENTS

I am profoundly grateful to my supervisor, Mr. Jignesh Patel, for his invaluable guidance, constant support, and constructive feedback throughout my thesis journey. His expertise and insights have been instrumental in shaping this research.

I extend my sincere thanks to Professor Dr. Manoj J from Liverpool John Moore's University, along with other esteemed professors from UpGrad, for their educational support and encouragement. Their knowledge and teachings have greatly enriched my learning experience.

I would also like to express my deepest appreciation to my parents and family members. Their unwavering belief in me, endless encouragement, and emotional support have been my strength and motivation during this academic endeavor.

This thesis is not only a reflection of my work but also a testament to the collective efforts and sacrifices of all those who have supported me. I am eternally thankful to each one of them.

ABSTRACT

This research evaluates the efficacy of seven ML models - Logistic Regression, Linear Discriminant Analysis, K-Nearest Neighbors, Decision Trees, Random Forest, Naïve Bayes, and Support Vector Machine - and two DL models, NASNetMobile and MobileNetV2, for apple leaf disease detection using the PlantVillage dataset. The focus is on diseases like Apple Scab, Black Rot, and Cedar Apple Rust. Feature extraction techniques like Hu Moments, Haralick texture, and Color histograms are utilized for ML models, while MobileNetV2 and NASNetMobile leverage transfer learning. Stratified K-Fold Cross-Validation ensures robust model evaluation. Among ML models, Random Forest emerged as the most accurate with an accuracy of 95.21%, demonstrating its potential in disease detection. However, MobileNetV2 outperformed all models, achieving an overall accuracy of 98%, underscoring its superiority in classifying plant health conditions. This research highlights MobileNetV2's advanced efficiency and reliability, making it a promising tool for automated plant disease diagnosis, and revolutionizing agricultural disease management practices with its precision and recall capabilities.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AUC-ROC	Area Under the Receiver Operating Characteristic Curve
BGR	Blue, Green, and Red
CART	Classification and Regression Trees
CNN	Convolutional Neural Network
CRI	Chlorophyll Reflectance Index
DL	Deep Learning
DT	Decision Trees
DWT	Discrete Wavelet Transform
ELM	Extreme Learning Machine
FNR	False Negative Rate
FPR	False Positive Rate
GNDVI	Green Normalized Difference Vegetation Index
GLCM	Gray Level Co-occurrence Matrix
GWT	Gabor Wavelet Transform
HDL	Hybrid Deep Learning
HOG	Histogram Oriented Gradient
HSV	Hue, Saturation, and Value
KNN	K-Nearest Neighbor
LBP	Local Binary Patterns
L-SVM	Linear Support Vector Machine
LR	Logistic regression
LTP	Local Ternary Patterns
MCC	Matthews Correlation Coefficient
MDC	Minimum Distance Classifier
ML	Machine Learning
NB	Naïve Bayes
NN	Neural Network
OSAWI	Optimized Soil Adjusted Water Index
PCA	Principal Component Analysis
PSNR	Peak Signal-to-Noise Ratio
RBF	Radial Basis Function

RFC	Random Forest Classifier
RF	Random Forest
RMSE	Root Mean Square Error
ROC	Receiver Operating Characteristic
SGD	Stochastic Gradient Descent
SFTA	Segmented Fractal Texture Analysis
SIFT	Scale-Invariant Feature Transform
SVM	Support Vector Machine
SURF	Speed Up Robust Feature
SVC	Support Vector Classifier
TPU	Tensor Processing Unit
VGG	Visual Geometry Group
VM	Virtual Machine

Chapter 1

INTRODUCTION

1.1 Background of the Study

The evolution of plant disease detection methodologies has significantly advanced with machine learning (ML) and deep learning (DL), transforming traditional manual inspection methods into technology-driven models for greater accuracy and efficiency. This shift is crucial across a diverse array of crops, including essential staples and fruits like apples, where diseases such as Apple Scab and Cedar Apple Rust demand early detection to prevent widespread infection (Wójtowicz et al., 2021; Bhosale et al., 2023; Astani et al., 2022; Sai & Patil, 2022).

The challenge lies in the precise, efficient, and early detection of plant diseases to maintain agricultural productivity and ensure food security. This study aims to bridge the gap in comparative analysis by identifying the most effective ML and DL models for leaf disease detection in apples, leveraging advancements like Decision Trees, Random Forests, NASNetMobile, and MobileNetV2 (Sharma et al., 2020; Shakeel et al., 2020).

Consider the application of drone technology in apple orchards for early disease detection. Drones equipped with high-resolution cameras capture comprehensive imagery of the orchard, followed by image preprocessing and feature extraction to identify disease indicators. The processed data feeds into ML and DL models, which classify the leaves and integrate the findings into a decision support system for farmers. This system, which tracks disease progression and recommends actionable steps, exemplifies the shift towards a more accurate, efficient, and data-driven approach in agriculture.

By addressing the literature review's identified gaps, this research contributes to advancing agricultural practices through technological innovations, ensuring early disease detection, minimizing crop loss, and promoting sustainable farming (Yousuf & Khan, 2021; Harakannanavar et al., 2022). The commitment to enhancing agricultural technology underscores the importance of this study within the academic and practical application realms.

1.2 Aim and Objectives

The primary aim of this research is to evaluate and compare the effectiveness of seven Machine Learning (ML) models and two Deep Learning (DL) models in detecting diseases in apple plant leaves. This comparison seeks to establish the most accurate model for leaf disease detection, which is crucial for improving disease management in apple cultivation.

The research objectives are formulated based on the aim of this study which are as follows:

1. To analyze various ML models (Logistic Regression, Decision Trees, LDA, KNN, Random Forest, Naive Bayes, SVM) for effective identification of diseases like Apple Scab, Black Rot, and Cedar Apple Rust in apple plant leaves.
2. To leverage CNN with NASNetMobile and MobileNetV2 architectures in Deep Learning for increased precision in detecting diseases in apple leaves.
3. To conduct a comparative study of ML and DL models, focusing on metrics such as accuracy, precision, recall, and f1-score, to identify the most proficient model for disease detection.
4. To implement advanced image processing and feature extraction techniques, ensuring comprehensive data preparation for model training and testing.

1.3 Research Questions

This thesis tries to answer the following questions:

1. How do various Machine Learning models (such as Logistic Regression, Decision Trees, K-Nearest Neighbors, Random Forest, Naive Bayes, and SVM) compare in terms of accuracy and effectiveness in detecting diseases in apple plant leaves?
2. What is the comparative performance of Deep Learning models (specifically CNN with NASNetMobile and MobileNetV2 architectures) in leaf disease detection, and how do they enhance detection accuracy?
3. How do the image processing and feature extraction methods employed affect the efficiency and accuracy of both ML and DL models in identifying specific leaf diseases?
4. What are the limitations and challenges in current plant disease detection methods, and how does this research aim to address them through the integration of advanced ML and DL techniques?

1.4 Significance of Study

This research significantly advances agricultural technology by meticulously comparing a spectrum of deep learning (DL) and machine learning (ML) models for the detection of apple leaf diseases. The study is distinctive in its approach, intertwining traditional ML techniques with contemporary deep learning architectures, thereby offering a broad perspective on their relative effectiveness in real-world agricultural scenarios. This comprehensive evaluation not only enriches the existing corpus of knowledge in plant pathology and data science but also sets the stage for future breakthroughs in the realm of automated plant disease diagnostics.

The study's findings have widespread implications. For agronomists, it offers new insights into disease identification and management, potentially revolutionizing traditional practices. Farmers benefit from the enhanced ability to diagnose and treat diseases promptly, leading to improved crop health, increased yields, and, consequently, economic gains. Researchers in agriculture and related fields gain a valuable resource in this comparative analysis, which may guide future studies and innovations in plant disease detection.

Moreover, the study extends its utility to developers in agricultural technology. By providing a detailed assessment of various ML and DL models, it lays a groundwork for the development of advanced, efficient disease detection systems, tailored for agricultural use. This is particularly pertinent in the context of increasing global food demands and the need for sustainable farming practices.

In essence, this research stands at the intersection of technology and agriculture, offering tangible benefits to various stakeholders and contributing to the broader goal of achieving food security and sustainable agricultural practices. It highlights the potential of data-driven approaches in addressing complex challenges in agriculture, thus serving as a cornerstone for future explorations in this domain.

1.5 Scope of the Study

This section outlines the specific areas of focus for the research, setting clear boundaries for what the study will cover and what it will not.

- The study is centered around the detection of certain diseases in apple leaves, namely Apple Scab, Black Rot, and Cedar Apple Rust. This focus allows for a detailed exploration of disease characteristics and their detection using ML and DL models.

- The research limits itself to seven machine learning models and two deep-learning models. This selection is strategic, allowing for a comprehensive comparison across a spectrum of widely recognized and emerging techniques in the field.
- The methodology involves image processing, feature extraction, and model evaluation through K-Fold Cross-Validation. These techniques were chosen for their relevance and potential in addressing the research questions effectively.
- The efficacy of these models is assessed within a controlled environment using pre-captured images. This setup ensures the reliability and reproducibility of the results, although it may not fully capture the complexities of real-world conditions.
- The study does not extend to real-time implementation in the field, considering factors like variable lighting and the dynamic nature of outdoor agricultural conditions. Practical considerations, challenges in real-world agricultural settings, and deployment on edge devices are beyond the scope due to computational resource constraints.
- Recognizing these limitations, the study acknowledges the need for future research to address real-world applications and challenges in the agricultural environment, aiming for a more comprehensive solution that can be effectively deployed in practical scenarios.

1.6 Structure of the Study

The structure of the thesis is as follows:

Chapter 1, "Introduction," sets the foundation for the study by discussing the evolution of plant disease detection methodologies, emphasizing the shift from manual inspection to technology-driven models. It highlights the research's aim to evaluate various ML and DL models for detecting diseases in apple leaves, aligning with advancements in agricultural technology.

Chapter 2, "Literature Review," provides an extensive analysis of existing research in the field of plant disease detection using ML and DL models. It meticulously reviews various studies, highlighting the methodologies developed, the types of diseases targeted, and the data sources used. This chapter also compares different approaches, discussing their strengths, limitations, and the outcomes achieved. Furthermore, it delves into the evolution of image processing and feature extraction techniques in agriculture, examining how these advancements have shaped current disease detection strategies. This comprehensive review sets a solid academic foundation for the study, contextualizing it within the broader field of agricultural technology and data science.

Chapter 3, "Research Methodology," outlines the procedures and techniques for evaluating ML and DL models in apple leaf disease detection. It details data collection from the PlantVillage Dataset, preprocessing steps, including feature extraction and data augmentation, and describes the implementation of various ML and DL models. The chapter also discusses the use of K-Fold Cross-Validation and the evaluation metrics employed to assess the models.

Chapter 4, “Analysis and Design”, dives into the analysis and design of machine learning (ML) and deep learning (DL) models for plant disease classification, emphasizing the processing and augmentation of image data. It thoroughly outlines the implementation of various ML and DL algorithms, focusing on feature extraction, model architecture, and evaluation metrics to bridge theoretical concepts with practical agricultural applications.

Chapter 5, “Results and Discussion”, meticulously evaluates different ML and DL models for detecting diseases in apple leaves, starting with data collection and proceeding through preprocessing steps like feature extraction and data augmentation. It discusses the implementation and robust evaluation of several ML models and advanced DL models, concluding with the selection of the most effective model based on comprehensive metrics.

Chapter 6, “Conclusion and Recommendations”, culminates in a comprehensive performance analysis of the evaluated models, leading to actionable recommendations for future research in agricultural technology. It covers enhancing data collection and annotation, model optimization, computational efficiency, deployment, and continuous learning, aiming to improve disease detection methodologies and integrate advanced ML and DL models into agricultural practices for better disease management strategies.

Chapter 2

LITERATURE REVIEW

2.1 Introduction

The field of agriculture critically depends on effective plant disease management. Rapid and accurate detection of plant diseases is vital for maintaining plant health, ensuring food security, and stabilizing economies. In this context, the intersection of machine learning (ML) and deep learning (DL) with plant pathology has emerged as a significant advancement.

The transformation from manual inspection to automated, technology-driven systems in plant pathology has been revolutionary. ML and DL models have not only enhanced the accuracy and efficiency of disease detection but also introduced predictive capabilities for proactive management strategies. Leaf disease identification, being the focus of this thesis, is particularly critical as leaves often exhibit early signs of disease, enabling early detection and management.

This thesis aims to fill the gap in the literature by providing a comprehensive comparative analysis of seven ML algorithms and two DL models, focusing specifically on leaf disease detection. The goal is to identify the most effective model for this purpose, contributing both to the academic field and practical applications in plant disease management.

The literature review of this thesis is structured around 13 meticulously chosen questions. These questions delve into various aspects of existing research on plant disease detection using ML and DL, including problem definition (LRQ1), data collection sources (LRQ2), types of plant diseases and classifications (LRQ3), pre-processing and data augmentation techniques (LRQ4), feature extraction methods (LRQ5), key attributes extracted (LRQ6), classification techniques (LRQ7), approaches to overfitting (LRQ8), performance evaluation metrics (LRQ9), accuracy of different approaches (LRQ10), techniques for data balancing (LRQ11), limitations and gaps in current research (LRQ12), and future research directions (LRQ13). This structured approach will enable a thorough understanding of the field's current state and its evolution.

Table 2-1 listed below lists each of the 13 LRQs along with their motivations, providing a clear framework for the review process.

Table 2-1: Literature Review Questions and Their Motivation

LRQ. No.	Literature Review Question (LRQ)	Motivation Behind the Question
LRQ1	Problems Addressed	To understand the problems addressed and to find the importance of implementing ML and DL techniques in detecting plant-based diseases.
LRQ2	Data Collection Sources	To understand the origins and type of data used and explore the various methods employed for collecting plant disease data.
LRQ3	Types of Plant Diseases Identified	To identify the range of plant diseases that have been studied.
LRQ4	Pre-Processing and Data Augmentation Techniques	To examine how data is prepared and augmented for effective analysis.
LRQ5	Techniques Utilized for Feature Extraction	To understand the methods used to extract relevant features from the data.
LRQ6	Attributes Extracted for Feature Extraction	To identify the specific attributes that are extracted for analysis.
LRQ7	Classification Techniques Used	To explore the various classification methods employed in these studies.
LRQ8	Techniques Utilized for Reducing Overfitting	To investigate approaches used to prevent model overfitting.
LRQ9	Performance Evaluation and Metrics	Used To understand how the effectiveness of the models is assessed.
LRQ10	Reported Accuracy of Different Approaches	To compare the accuracy reported in different studies.
LRQ11	Balancing Techniques Used in Models	To examine methods used to handle imbalanced datasets.
LRQ12	Limitations and Research Gaps Identified	To identify the current limitations and gaps in research.

LRQ13	Future Directions and Recommendations	To explore potential future research paths and improvements.

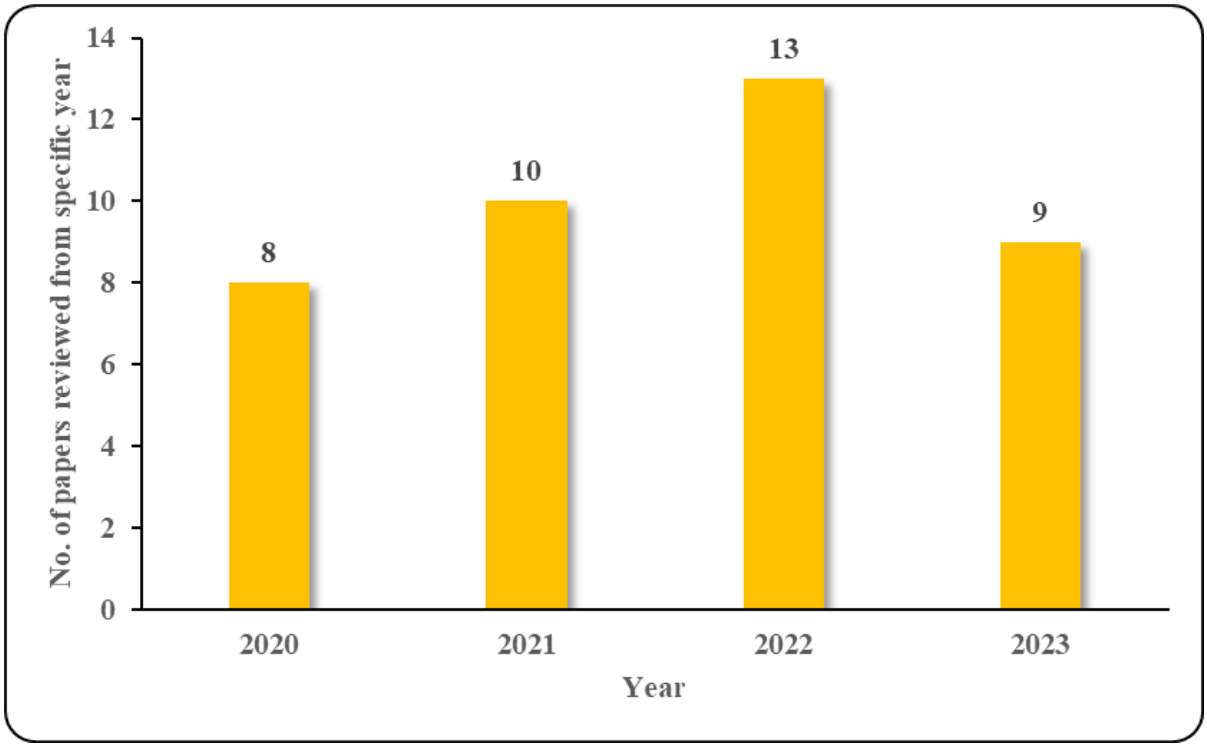


Figure 2-1: Count of papers per year from 2020 to 2023

Figure 2-1 shows the number of papers reviewed each year, indicating a growing trend in research activity in this domain.

This introduction sets the stage for a comprehensive analysis of current research, linking existing knowledge with the novel contributions of this thesis. It aims to provide insights into the evolving integration of ML and DL in plant disease detection, highlighting both current successes and challenges. This analysis will help to understand how technological advancements are shaping the future of plant disease detection in agriculture.

2.2 Problems Addressed

LRQ1: What is the problem being addressed, and why is this research on plant disease detection using Machine Learning and/or Deep Learning important?

The research on plant disease detection using Machine Learning (ML) and Deep Learning (DL) spans across various plants, including wheat, rye, tomatoes, potatoes, corn, apples, rice, and mangoes. A key focus is on enhancing the accuracy and efficiency of disease detection, which is critical for ensuring agricultural productivity and food security. Techniques like Convolutional Neural Networks (CNNs), Random Forest algorithms, ensemble classifiers, and image processing are prominently utilized. The importance of these studies lies in their potential to replace traditional, labour-intensive, and error-prone methods such as human visual inspection. By enabling early and accurate disease detection, these technologies aim to reduce economic losses, minimize the misuse of chemicals, prevent plant damage, and ensure sustainable agricultural practices. The overarching goal is to support global demands for food security and sustainable agriculture in the face of increasing population pressures and economic challenges. This is exemplified in the work of (Zhong & Zhao, 2020) on apple leaf diseases, (Agarwal et al., 2020) on tomato plants, (Silviya et al., 2022), and (Shivaprasad & Wadhawan, 2023) on various plant diseases using CNNs like VGG16 and VGG19 for detection and classification.

2.3 Data Collection Sources

LRQ2: What are the primary sources for collecting plant data?

The primary sources for collecting plant data in the studies referenced in Table 2-2 predominantly include online databases like the PlantVillage dataset, hyperspectral imaging, and field images. Several studies sourced their data from online platforms and databases such as Kaggle, which offer a wide range of plant images under various conditions. Hyperspectral imaging was notably used in a study by (Wójtowicz et al., 2021) for capturing diverse wavelengths to detect disease symptoms in wheat and rye. The PlantVillage dataset was a popular choice, employed in multiple studies for its extensive collection of plant images, including various plant species and conditions. Field images were also commonly used, gathered from real environments to ensure a realistic representation of plant conditions. These images were often categorized and labeled for various diseases and healthy conditions, providing a rich dataset for training and testing machine learning and deep learning models. In

some instances, datasets were augmented to enhance diversity and balance, ensuring effective model training. The utilization of these diverse sources illustrates the importance of varied and comprehensive datasets in developing accurate and robust plant disease detection models.

Table 2-2: Summarised View of Data Collection Source

Sl. No.	Reference	Origin of Data Collection
1	(Wójtowicz et al., 2021)	Real Environment
2	(Bhosale et al., 2023)	Real Environment, Internet
3	(Astani et al., 2022)	PlantVillage, Taiwan Tomato Leaves database
4	(Aurangzeb et al., 2020)	PlantVillage
5	(Sai & Patil, 2022)	PlantVillage
6	(Kholiya et al., 2023)	Real Environment, Internet
7	(Chaitanya Reddy et al., 2021)	Internet
8	(Sharma et al., 2020)	Real Environment, Internet
9	(Shakeel et al., 2020)	Internet
10	(Yousuf & Khan, 2021)	PlantVillage
11	(Baheti et al., 2022)	PlantVillage
12	(Harakannanavar et al., 2022)	PlantVillage
13	(Singla et al., 2023)	Real Environment
14	(Gosai et al., 2022)	Internet
15	(Kilaru & Raju, 2022)	PlantVillage
16	(Meenakshi, 2023)	Internet
17	(Mohanty et al., 2022)	PlantVillage
18	(Sangeetha et al., 2022)	Internet
19	(Silviya et al., 2022)	Plant Town
20	(Shivaprasad & Wadhawan, 2023)	Internet
21	(Al-gaashani et al., 2022)	PlantVillage
22	(Pawar et al., 2022)	Internet
23	(Rizvee et al., 2023)	Real Environment
24	(Jasim & Al-Tuwaijari, 2020)	PlantVillage
25	(Habiba & Islam, 2021)	PlantVillage

26	(Ramiah Subburaj et al., 2023)	PlantVillage
27	(Lamba et al., 2021)	PlantVillage
28	(Aggarwal et al., 2023)	Rice-Disease-dataset
29	(Sujatha et al., 2021)	Real Environment
30	(Kumar et al., 2022)	PlantVillage
31	(Hatuwal et al., 2020)	PlantVillage
32	(Ahmed & Yadav, 2022)	PlantVillage
33	(Chug et al., 2023)	Real Environment, PlantVillage-TomEBD, PlantVillage-BBLS
34	(Chowdhury et al., 2021)	PlantVillage
35	(Atila et al., 2021)	PlantVillage
36	(Zhong & Zhao, 2020)	AIChallenger-Plant-Disease-Recognition
37	(Agarwal et al., 2020)	PlantVillage
38	(Mishra et al., 2020)	Real Environment, PlantVillage
39	(Shrivastava & Pradhan, 2021)	Real Environment
40	(Trivedi et al., 2021)	Internet

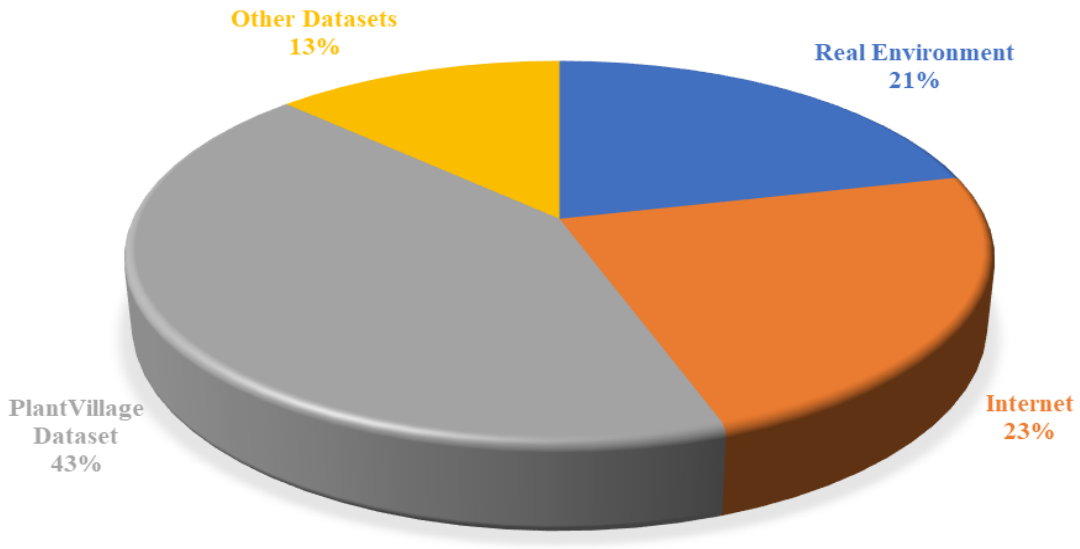


Figure 2-2: Usage of various data collection sources

The pie chart in Figure 2-2 illustrates the distribution of primary sources for plant data collection used in research. The PlantVillage Dataset is the most significant source, constituting 43% of the data. Internet-sourced data follows at 23%, and real environment captures makeup 21%. Other datasets account for the remaining 13%, indicating a variety of minor sources. This distribution underscores the dominance of the PlantVillage Dataset in plant research.

2.4 Types of Plant Diseases Identified

LRQ3: What plant species and disease classifications are covered in the studies?

The reviewed research papers in LRQ3 cover a wide array of plant diseases across diverse plants like wheat, rye, rice, tomatoes, potatoes, corn, apples, cotton, mango, citrus, and more as detailed in Table 2-3. There is a notable focus on tomato-related diseases, reflecting the plant's global importance, with comprehensive studies on several tomato leaf diseases including bacterial spot and yellow leaf curl virus. Studies on rice and mango diseases are also prominent, particularly in regions where these plants are key, such as the study on mango leaf diseases in Bangladesh by (Rizvee et al., 2023). The PlantVillage dataset is a common source for these studies, providing a broad range of plant disease images. This research underscores the critical need for precise plant disease detection and classification, addressing global food security and specific regional agricultural needs. The emphasis on a variety of diseases across different plant species underlines the necessity for adaptable and robust disease detection models suitable for diverse agricultural conditions, as evidenced by the diverse range of diseases and plant leaves documented in Table 2-3.

Table 2-3: Summary of Plants Studied and Identified Diseases

Sl. No.	Reference	Study of Plants Leaves	Identified Diseases
1	(Wójtowicz et al., 2021)	Winter Wheat and Winter Rey	Leaf rust caused by Puccinia recondita f. sp. tritici on wheat and Puccinia recondita f. sp. recondita on rye
2, 3	(Bhosale et al., 2023), (Kholiya et al., 2023)	Rice	Leaf Blast, Leaf Blight, and Brown Spot

4	(Astani et al., 2022)	Tomato	13 classes of Tomato diseases
5	(Aurangzeb et al., 2020)	Potato and Corn	➢ Potato Early Blight and Potato Late Blight are examined for potato plants. ➢ Corn Cercospora Leaf Spot, Corn Common Rust, and Corn Northern Blight are examined for corn plants.
6	(Sai & Patil, 2022)	Apple	Apple Scab, Cedar Apple Rust, Black Rot
7	(Chaitanya Reddy et al., 2021)	Cotton, Coffee	Bacterial Spot, Bacterial Blight, Bacterial Wilt, Anthracnose, Spider Mites, Angular Leaf Spot
8	(Sharma et al., 2020)	Rice	Leaf Blast, Sheath Blight, False Smut, Neck Blast, Sheath rot, Brown Spot, Bacterial Blight
9	(Shakeel et al., 2020)	Cotton	Bacterial diseases, Fungal diseases, and Viral diseases
10	(Yousuf & Khan, 2021)	1000 images from PlantVillage dataset	Early Blight, Brown Rust, and Late Blight
11	(Baheti et al., 2022)	Tomato	10 different diseases
12	(Harakannanavar et al., 2022)	Tomato	6 different diseases
13	(Singla et al., 2023)	Multiple leaves	No diseases are identified only the classification
14	(Gosai et al., 2022)	Multiple leaves	26 different diseases
15	(Kilaru & Raju, 2022)	Maize	No diseases are identified only the classification

16	(Meenakshi, 2023)	Medicinal plant leaves like Betel, Citrus, Hibiscus, Basil, Pepper and Mango	Betel: Leaf Spot, Anthracnose, and Betelvine Bug. Citrus: Canker, Black Spot, and Greening. Hibiscus: Sun Burn. Basil: Downy Mildew. Pepper: Bacterial spot. Mango leaves: Anthracnose, and Algal leaf spot.
17	(Mohanty et al., 2022)	Tomato	9 different diseases
18	(Sangeetha et al., 2022)	Apple	Rust, scab, and multiple diseases affecting leaves of apple plant.
19	(Silviya et al., 2022)	Multiple Leaves	No mention of diseases
20	(Shivaprasad & Wadhawan, 2023)	Multiple Leaves	No mention of diseases
21	(Al-gaashani et al., 2022)	Tomato	Bacterial spot, Yellow leaf curl virus, Septoria leaf spot, Leaf mold, Late blight
22	(Pawar et al., 2022)	corn, apple, tomato, potato, grape, rice, sugarcane, pepper, soybean and cucumber	No mention of disease names
23	(Rizvee et al., 2023)	Mango	Anthracnose, Bacterial Canker, Cutting Weevil, Die Back, Gall Midge, Powdery Mildew, Sooty Mould
24	(Jasim & Al-Tuwaijari, 2020)	Tomato, Pepper, Potato	12 different diseases

25	(Habiba & Islam, 2021)	Tomato	9 different diseases
26	(Ramiah Subburaj et al., 2023)	Tomato	Septoria Leaf Spot, Leaf Mold, Target Spot, Spider Mites, Tomato Yellow Leaf Curl Virus, Tomato Mosaic Virus
27	(Lamba et al., 2021)	Rice	Bacterial Leaf Blight (BLB), Blast, Brown Spot
28	(Aggarwal et al., 2023)	Rice	Bacterial Leaf Blight (BLB), Blast, Brown Spot
29	(Sujatha et al., 2021)	Citrus	Black Spot, Canker, Greening, Melanose
30	(Kumar et al., 2022)	Multiple Plant Leaves	17 basic diseases including Bacterial, Mold, Viral, and Termite diseases
31	(Hatuwal et al., 2020)	Apple, Cherry, Grape, Peach, Pepper, Strawberry	General Plant leaf diseases.
32	(Ahmed & Yadav, 2022)	Rice	Bacterial Leaf Blight (BLB), Blast, Brown Spot
33	(Chug et al., 2023)	Tomato	Early blight
34	(Chowdhury et al., 2021)	Tomato	Early blight, Septoria, Leaf Spot, and others.
35	(Atila et al., 2021)	Multiple Plant Leaves	Cedar apple rust, powdery mildew, black measles, late blight, bacterial spot, strawberry
36	(Zhong & Zhao, 2020)	Apple	General Apple Scab, Serious Apple Scab, Apple Gray Spot, General Cedar Apple Rust, and Serious Cedar Apple Rust

37	(Agarwal et al., 2020)	Tomato	Target Spot, Mosaic virus, Bacterial spot, Late blight, Leaf Mold, Yellow Leaf Curl Virus, Spider mites: Two-spotted spider mite, Early blight, Septoria leaf spot
38	(Mishra et al., 2020)	Corn	Rust, Northern leaf blight
39	(Shrivastava & Pradhan, 2021)	Rice	Bacterial Leaf Blight (BLB), Rice Blast, Sheath Blight (SB)
40	(Trivedi et al., 2021)	Tomato	9 types of diseases

2.5 Pre-Processing and Data Augmentation Techniques

LRQ4: What pre-processing and data augmentation techniques are used?

In the studies analyzed for LRQ4, a diverse array of pre-processing and data augmentation techniques are employed for enhancing plant disease detection, as detailed in Table 2-4. (Wójtowicz et al., 2021) utilize the Boruta algorithm for effective preprocessing, along with Pearson correlation analysis and spectral data unmixing. (Bhosale et al., 2023) focus on image manipulation techniques, including plantping, color enhancement, and scaling, supplemented by rotations, zooming, and flipping for data augmentation. Techniques for handling low-quality images, such as noise reduction and shadow correction, are highlighted in (Astani et al., 2022), while (Aurangzeb et al., 2020) apply median filtering and resizing for noise removal. The diversity in these methods, from simple transformations to complex algorithmic approaches, reflects the varied challenges in plant disease imagery and the necessity for tailored preprocessing steps to enhance the accuracy of disease detection models.

Table 2-4: Summary of Pre-Processing and Data Augmentation Technique

Sl. No.	Referenced Paper	Pre-Processing and Data Augmentation Technique
1	(Wójtowicz et al., 2021)	Boruta algorithm for preprocessing, Pearson correlation analysis, laboratory spectra for data unmixing
2	(Bhosale et al., 2023)	Plantping, color enhancement, scaling, rotation, zooming, flipping, resizing
3	(Astani et al., 2022)	Noise reduction, low-quality image handling, shadow correction, contrast enhancement
4	(Aurangzeb et al., 2020)	Resizing to 256x256, median filter for noise removal
5	(Sai & Patil, 2022)	BGR to RGB conversion, HSV format conversion, image segmentation
6	(Kholiya et al., 2023)	Plantping, color enhancement, scaling, rotation, zoom, shifts
7	(Chaitanya Reddy et al., 2021)	Uniform resizing, scaling, Gaussian filters
8	(Sharma et al., 2020)	RGB to grayscale, median filter, Sobel operator, k-means clustering, GLCM method
9	(Shakeel et al., 2020)	Plantping, resizing, color transformation, contrast enhancement, noise filtering
10	(Yousuf & Khan, 2021)	Grayscale conversion, Gaussian filter
11	(Baheti et al., 2022)	Resizing, RGB to HSV conversion
12	(Harakannanavar et al., 2022)	Resizing to 256x256, Histogram Equalization, K-means clustering, contour tracing
13	(Singla et al., 2023)	Image cleaning and transformation, data augmentation with Keras
14	(Gosai et al., 2022)	Image resizing, smoothing, enhancement, RGB to HSV/LAB conversion, Otsu's strategy, K-means clustering
15	(Kilaru & Raju, 2022)	RGB to grayscale conversion, resizing, noise removal

16	(Meenakshi, 2023)	Image resizing, noise filtering, background removal, contrast enhancement, RGB transformation
17	(Mohanty et al., 2022)	RGB to grayscale conversion, resizing, GANs with deep convolutional networks
18	(Sangeetha et al., 2022)	Image-to-array conversion, pixel scaling
19	(Silviya et al., 2022)	Normalization, resampling, tensor conversion, mean difference calculation
20	(Shivaprasad & Wadhawan, 2023)	Image resizing, normalization, grayscale conversion
21	(Al-gaashani et al., 2022)	Image resizing and normalization
22	(Pawar et al., 2022)	Image resizing
23	(Rizvee et al., 2023)	Not specified in the section reviewed
24	(Jasim & Al-Tuwaijari, 2020)	Image resizing to 128x128
25	(Habiba & Islam, 2021)	Contrast changes for data augmentation
26	(Ramiah Subburaj et al., 2023)	Flips, shifts, zooms
27	(Lamba et al., 2021)	Image correction and enhancement
28	(Aggarwal et al., 2023)	Image resizing to 75x75 pixels
29	(Sujatha et al., 2021)	Image embedding with Squeezenet, SGD optimization
30	(Kumar et al., 2022)	Image reshaping and resizing
31	(Hatuwal et al., 2020)	Grayscale conversion, blurring to reduce noise, Haralick texture features
32	(Ahmed & Yadav, 2022)	GLCM for texture characteristics
33	(Chug et al., 2023)	Bilateral filtering, k-means clustering
34	(Chowdhury et al., 2021)	Image resizing, Z-score normalization, rotation, scaling, translation
35	(Atila et al., 2021)	Normalization, resizing for various models, data augmentation for class balance
36	(Zhong & Zhao, 2020)	Random rotation, translation, scaling, inversion, plantping, normalization

37	(Agarwal et al., 2020)	Rotating, flipping, plantping, and resizing using Augmentor package
38	(Mishra et al., 2020)	Convolution layers, max-pooling, ReLU functions, dropout layers, flatten layers
39	(Shrivastava & Pradhan, 2021)	Pre-processing to have a black background
40	(Trivedi et al., 2021)	Normalization, resizing to 256x256, grayscale transformation

2.6 Techniques Utilized for Feature Extraction

LRQ5: What feature extraction methods are employed in plant disease detection?

The research papers reviewed in LRQ5 demonstrate a range of feature extraction methods used for plant disease detection, as detailed in Table 2-5. (Wójtowicz et al., 2021) utilize spectral mixture analysis and Pearson correlation analysis in conjunction with the Random Forest algorithm to analyze hyperspectral data. (Bhosale et al., 2023) employ Convolutional Neural Networks (CNNs) for extracting key information from images. (Astani et al., 2022) use sophisticated techniques like Scale-Invariant Feature Transform (SIFT) and Gabor Wavelet Transform (GWT), along with color texture geometric features. (Aurangzeb et al., 2020) focus on Histogram Oriented Gradient (HOG), Local Ternary Patterns (LTP), and Segmented Fractal Texture Analysis (SFTA) for extracting shape, texture, and pattern features. Other notable methods include the use of CNN models like VGG-16, ResNet50, and InceptionV3 by (Kholiya et al., 2023), and Hu Moments, Haralick Texture, and color Histogram by (Sai & Patil, 2022) for feature extraction from apple leaf images. These varied approaches underline the importance of extracting relevant features from plant images, which are crucial for accurate disease classification.

Table 2-5: Summary of Techniques Utilized

Sl. No.	Referenced Paper	Techniques Utilized
1	(Wójtowicz et al., 2021)	Spectral mixture analysis, Pearson correlation analysis, Random Forest
2	(Bhosale et al., 2023)	Convolutional Neural Networks (CNNs)

3	(Astani et al., 2022)	Color texture geometric features, SIFT, Gabor Wavelet Transform (GWT)
4	(Aurangzeb et al., 2020)	Histogram Oriented Gradient (HOG), Local Ternary Patterns (LTP), SFTA
5	(Sai & Patil, 2022)	Hu Moments, Haralick Texture, color Histogram
6	(Kholiya et al., 2023)	CNN models (VGG-16, ResNet50, InceptionV3)
7	(Chaitanya Reddy et al., 2021)	Edge-based, point-based, line-based image analysis
8	(Sharma et al., 2020)	SURF (Speed Up Robust Feature) algorithm
9	(Shakeel et al., 2020)	A hybrid method combining texture and color feature extraction
10	(Yousuf & Khan, 2021)	Gray Level Co-occurrence Matrix (GLCM)
11	(Baheti et al., 2022)	Histogram techniques
12	(Harakannanavar et al., 2022)	Discrete Wavelet Transform (DWT), PCA, GLCM
13	(Singla et al., 2023)	
14	(Gosai et al., 2022)	Color, texture, morphology, and edge features
15	(Kilaru & Raju, 2022)	Discrete Wavelet Transform (DWT)
16	(Meenakshi, 2023)	Gray-Level Co-Occurrence Matrix (GLCM)
17	(Mohanty et al., 2022)	Histogram of Oriented Gradients (HOG)
18	(Sangeetha et al., 2022)	VGG16 Convolutional Neural Network (CNN)
19	(Silviya et al., 2022)	Deep Learning frameworks for automated feature extraction
20	(Shivaprasad & Wadhawan, 2023)	CNN, VGG16, VGG19
21	(Al-gaashani et al., 2022)	Lightweight CNN architectures (MobileNetV2, NASNetMobile), kernel PCA
22	(Pawar et al., 2022)	15-layer CNN model
23	(Rizvee et al., 2023)	LeafNet model with convolutional layers, AlexNet, VGG16
24	(Jasim & Al-Tuwaijari, 2020)	convolutional Neural Network (CNN)

25	(Habiba & Islam, 2021)	Deep Convolutional Neural Network (CNN), specifically VGG16
26	(Ramiah Subburaj et al., 2023)	Inception V3 model
27	(Lamba et al., 2021)	Wavelet-based detection, GLCM, Gabor wavelet, SIFT
28	(Aggarwal et al., 2023)	Various deep learning architectures (CNN, Xception, VGG16, etc.)
29	(Sujatha et al., 2021)	Support Vector Machine (SVM), SGD
30	(Kumar et al., 2022)	CNN models (AlexNet, VGG16)
31	(Hatuwal et al., 2020)	Haralick texture features algorithm
32	(Ahmed & Yadav, 2022)	Grey-level co-occurrence matrices (GLCM)
33	(Chug et al., 2023)	EfficientNet (EfNet) architecture
34	(Chowdhury et al., 2021)	Deep learning architectures, specifically CNNs and EfficientNet
35	(Atila et al., 2021)	Deep learning models (EfficientNet, AlexNet, ResNet50, VGG16, Inception V3)
36	(Zhong & Zhao, 2020)	DenseNet-121 as the backbone network for deep learning model
37	(Agarwal et al., 2020)	CNN model with convolution and max pooling layers
38	(Mishra et al., 2020)	Series of convolution layers in CNN
39	(Shrivastava & Pradhan, 2021)	Color feature extraction from various color spaces
40	(Trivedi et al., 2021)	Convolutional Neural Network (CNN)

2.7 Attributes Extracted for Feature Extraction

LRQ6: What are the key attributes extracted from plant images?

The analysis of LRQ6 reveals diverse attributes extracted from plant images across various studies, each targeting specific aspects of plant diseases, as detailed in Table 2-6. (Wójtowicz et al., 2021) focus on spectral wavelengths and vegetation indices like CRI, OSAWI, and

GNDVI, which are effective in classifying leaf rust stages in plants. (Bhosale et al., 2023), (Astani et al., 2022), and (Sai & Patil, 2022) emphasize visual characteristics such as color, texture, and shape, crucial for identifying different plant diseases. (Aurangzeb et al., 2020) use shape descriptors like HOG, texture patterns with LTP, and fractal texture analysis using SFTA, primarily for potato and corn diseases. (Kholiya et al., 2023) and (Sharma et al., 2020) concentrate on disease symptoms, including variations in color, texture, shape, and morphological characteristics of diseased leaf portions, respectively. In studies like (Harakannanavar et al., 2022) and (Yousuf & Khan, 2021), texture-based features and GLCM algorithm features, including contrast, correlation, energy, and homogeneity, are extracted. These attributes, varying from simple color properties to complex spectral and texture features, are pivotal in accurately detecting and classifying plant diseases, demonstrating the multifaceted nature of plant disease diagnosis.

Table 2-6: Summary of Attributes Extracted

Sl. No.	Referenced Paper	Attributes Extracted
1	(Wójtowicz et al., 2021)	Spectral wavelengths, Vegetation indices (CRI, OSAWI, GNDVI)
2	(Bhosale et al., 2023)	Visual characteristics: Color, Texture, Shape
3	(Astani et al., 2022)	Color histograms, Hu Moments, Haralick features, Local Binary Patterns (LBP)
4	(Aurangzeb et al., 2020)	Shape descriptors (HOG), Texture patterns (LTP), Fractal texture analysis (SFTA)
5	(Sai & Patil, 2022)	Color properties, Texture features, Shape Characteristics
6	(Kholiya et al., 2023)	Disease symptoms: Variations in color, Texture, Shape
7	(Chaitanya Reddy et al., 2021)	Disease-affected area percentage, Texture, color, Shape
8	(Sharma et al., 2020)	Shape, Color, Texture, Morphological characteristics
9	(Shakeel et al., 2020)	Color and Texture features
10	(Yousuf & Khan, 2021)	GLCM features: Contrast, Correlation, Energy, Homogeneity, etc.

11	(Baheti et al., 2022)	Color, Texture, Shape changes
12	(Harakannanavar et al., 2022)	Texture-based features: Homogeneity, Autocorrelation, Dissimilarity, Entropy, etc.
13	(Singla et al., 2023)	Not Specified
14	(Gosai et al., 2022)	Color, Texture, Morphological features, Edges
15	(Kilaru & Raju, 2022)	Wavelet coefficients and associated attributes
16	(Meenakshi, 2023)	Texture features and statistical measures
17	(Mohanty et al., 2022)	Magnitude and orientation of gradients (HOG descriptor)
18	(Sangeetha et al., 2022)	Visual features: Spots, Discoloration, Textural changes
19	(Silviya et al., 2022)	Color and structure changes, Visual symptoms
20	(Shivaprasad & Wadhawan, 2023)	Inferred: Texture, Color, Shape, Patterns
21	(Al-gaashani et al., 2022)	Appearance and condition of tomato leaves
22	(Pawar et al., 2022)	Visual features: Color, Appearance
23	(Rizvee et al., 2023)	Visual indicators: Patterns, Colors, Textures
24	(Jasim & Al-Tuwaijari, 2020)	Color, Texture, Shape
25	(Habiba & Islam, 2021)	Spatial and temporal details: Color changes, Texture variations, Shape alterations
26	(Ramiah Subburaj et al., 2023)	Visual symptoms: Dark brown lesions, Concentric rings, color changes
27	(Lamba et al., 2021)	Texture and color features
28	(Aggarwal et al., 2023)	Patterns and characteristics of rice leaf diseases
29	(Sujatha et al., 2021)	Not Specified
30	(Kumar et al., 2022)	Color, Texture, Shape Characteristics
31	(Hatuwal et al., 2020)	Color and Texture features
32	(Ahmed & Yadav, 2022)	Texture-based features (GLCM): Skewness, Standard deviation, Homogeneity, etc.
33	(Chug et al., 2023)	Texture, color, Pattern differences
34	(Chowdhury et al., 2021)	Visual characteristics of healthy and diseased tomato leaves

35	(Atila et al., 2021)	Inferred: Color, Texture, Shape, Patterns of disease symptoms
36	(Zhong & Zhao, 2020)	Visual symptoms of diseases: Color changes, Spots, Texture variations
37	(Agarwal et al., 2020)	Visual indicators: Spots, Discolorations, Deformities
38	(Mishra et al., 2020)	Features identified through convolution layers
39	(Shrivastava & Pradhan, 2021)	Statistical parameters of color features
40	(Trivedi et al., 2021)	Image characteristics: Colors, Textures, Edges

2.8 Classification Techniques Used

LRQ7: What automated systems and methodologies are developed for identifying and categorizing plant diseases?

The studies in LRQ7 demonstrate a range of classification techniques for plant disease identification and categorization, as detailed in Table 2-7. (Wójtowicz et al., 2021) employed Random Forest models for classifying diseases using hyperspectral imaging data. (Bhosale et al., 2023) integrated machine learning classifiers like SVM, k-NN, and deep learning models like CNNs for analyzing plant leaf images. (Astani et al., 2022) developed an ensemble classification system using Random Forest, SVM, Logistic Regression, and K-Nearest Neighbors. (Aurangzeb et al., 2020) combined machine learning algorithms with feature extraction techniques such as HOG, LTP, and SFTA, using classifiers like multi-class SVM and ensemble trees. (Kholiya et al., 2023) utilized established deep convolutional neural networks like ResNet50, InceptionV3, and VGG-16. (Sai & Patil, 2022), (Sharma et al., 2020), and (Shakeel et al., 2020) applied various machine learning models including SVM, Random Forest, and ensemble models for disease detection. These studies illustrate the diverse methodologies and models employed for accurate and efficient plant disease detection and classification, highlighting the advances in deep learning and machine learning in this field.

Table 2-7: Summary of Classification Techniques Used

Sl. No.	Referenced Paper	Classification Techniques Used
1	(Wójtowicz et al., 2021)	Random Forest (RF)
2	(Bhosale et al., 2023)	SVM, k-NN, CNNs
3	(Astani et al., 2022)	Random Forest, SVM, Logistic Regression, K-Nearest Neighbors
4	(Aurangzeb et al., 2020)	SVM, Linear Discriminant, Ensemble Tree
5	(Sai & Patil, 2022)	Random Forest, SVM, K-Nearest Neighbors
6	(Kholiya et al., 2023)	VGG-16, ResNet50, InceptionV3 (CNN models)
7	(Chaitanya Reddy et al., 2021)	SVM, Random Forest
8	(Sharma et al., 2020)	Minimum Distance Classifier (MDC), Bayes' Classifier
9	(Shakeel et al., 2020)	SVM
10	(Yousuf & Khan, 2021)	Ensemble model (Random Forest and K-Nearest Neighbor)
11	(Baheti et al., 2022)	Random Forest, K-Nearest Neighbors (KNN), SVM
12	(Harakannanavar et al., 2022)	SVM, K-NN, CNN
13	(Singla et al., 2023)	Logistic Regression, Neural Networks, SVM, Naïve Bayes
14	(Gosai et al., 2022)	Residual Neural Network (ResNet)
15	(Kilaru & Raju, 2022)	Supervised Machine Learning techniques
16	(Meenakshi, 2023)	Modified Logistic Regression
17	(Mohanty et al., 2022)	Logistic Regression, SVM, Random Forest Classifier
18	(Sangeetha et al., 2022)	Convolutional Neural Network (CNN), specifically VGG16
19	(Silviya et al., 2022)	Convolutional Neural Network (CNN)
20	(Shivaprasad & Wadhawan, 2023)	CNN, VGG16, VGG19

21	(Al-gaashani et al., 2022)	SVM, RF, MLR (using concatenated features from CNN models)
22	(Pawar et al., 2022)	Comprehensive CNN-based framework
23	(Rizvee et al., 2023)	LeafNet (CNN-based model)
24	(Jasim & Al-Tuwaijari, 2020)	CNN
25	(Habiba & Islam, 2021)	Pre-trained VGG16 deep CNN model
26	(Ramiah Subburaj et al., 2023)	Inception V3 (transfer learning model)
27	(Lamba et al., 2021)	Deep learning classifiers with various activation functions
28	(Aggarwal et al., 2023)	Machine learning and deep learning algorithms (including transfer learning)
29	(Sujatha et al., 2021)	L-SVM, SGD, VGG-16, VGG-19
30	(Kumar et al., 2022)	CNN, Support Vector Classifier (SVC)
31	(Hatuwal et al., 2020)	SVM, KNN, Random Forest Classifier (RFC), CNN
32	(Ahmed & Yadav, 2022)	Random Forest, Nearest Neighbors, Linear Regression, Naive Bayes, Neural Networks, SVM
33	(Chug et al., 2023)	Hybrid Deep Learning (HDL) framework (combining DL architectures and ML classifiers)
34	(Chowdhury et al., 2021)	Convolutional Neural Networks (variants of U-net and EfficientNet)
35	(Atila et al., 2021)	EfficientNet, AlexNet, ResNet50, VGG16, Inception V3 (deep learning models)
36	(Zhong & Zhao, 2020)	DenseNet-121 deep convolution network
37	(Agarwal et al., 2020)	CNN model
38	(Mishra et al., 2020)	Deep Convolutional Neural Network-based system
39	(Shrivastava & Pradhan, 2021)	SVM, Discriminant Classifier, KNN, Naïve Bayes, Decision Tree, Random Forest, Logistic Regression
40	(Trivedi et al., 2021)	Convolutional Neural Network (CNN)

2.9 Techniques Utilized for Reducing Overfitting

LRQ8: How are issues like overfitting addressed during and after the model training process?

In the studies reviewed for LRQ8, various techniques are employed to address the challenge of overfitting during and after the model training process, as detailed in Table 2-8. (Wójtowicz et al., 2021) utilize bootstrap resampling and the Boruta algorithm for feature selection, ensuring robustness against overfitting. (Bhosale et al., 2023) implement data augmentation and regularization techniques to enhance model generalization. (Astani et al., 2022) address overfitting through ensemble classification and K-fold cross-validation, while (Sai & Patil, 2022) apply cross-validation and feature scaling. (Kholiya et al., 2023) and (Silviya et al., 2022) also use data augmentation alongside regularization techniques. The Random Forest algorithm, known for its inherent resistance to overfitting, is utilized by (Baheti et al., 2022). (Mohanty et al., 2022) leverage SVM and Random Forest with bagging and kernel transformations. Techniques like batch normalization, dropout layers, and early stopping callbacks are applied in (Rizvee et al., 2023), and kernel PCA is used in (Al-gaashani et al., 2022). These varied approaches demonstrate a comprehensive strategy toward minimizing overfitting, thereby enhancing the models' performance and reliability on unseen data.

Table 2-8: Summary of Techniques Implemented to Minimize Overfitting

Sl. No.	Referenced Paper	Techniques Implemented to Minimize Overfitting
1	(Wójtowicz et al., 2021)	Bootstrap resampling, Boruta algorithm for feature selection
2	(Bhosale et al., 2023)	Data augmentation, regularization techniques
3	(Astani et al., 2022)	Ensemble classification, K-fold cross-validation (k = 5)
4	(Sai & Patil, 2022)	Cross-validation, feature scaling
5	(Kholiya et al., 2023)	Data augmentation, regularization techniques
6	(Baheti et al., 2022)	Random Forest algorithm
7	(Singla et al., 2023)	Comparison of different machine learning models
8	(Mohanty et al., 2022)	SVM and Random Forest with bagging, kernel transformations
9	(Sangeetha et al., 2022)	Large and diverse dataset, validation techniques, regularization, data augmentation

10	(Silviya et al., 2022)	Data augmentation, Stochastic Gradient Descent (SGD) with momentum
11	(Shivaprasad & Wadhawan, 2023)	Comparison of CNN models, simpler architectures
12	(Al-gaashani et al., 2022)	Kernel PCA, evaluation of classifiers with/without kernel PCA
13	(Rizvee et al., 2023)	Batch normalization, dropout layers, cross-validation, early stopping callback
14	(Habiba & Islam, 2021)	Pre-trained model, freezing initial layers, implementing dropout layers
15	(Ramiah Subburaj et al., 2023)	Optimal regularization techniques, dropout layers, model fine-tuning
16	(Lamba et al., 2021)	Performance evaluation techniques, confusion matrix, MCC
17	(Aggarwal et al., 2023)	Data augmentation, regularization techniques, use of validation sets
18	(Sujatha et al., 2021)	Stratified 10-fold cross-validation, regularization techniques
19	(Kumar et al., 2022)	Dropout, regularization methods (L1 and L2)
20	(Hatuwal et al., 2020)	Diverse dataset, Random Forest, hyperparameters adjustment, dropout layers in CNN
21	(Ahmed & Yadav, 2022)	Random Forest algorithm (multiple decision trees and averaging predictions)
22	(Chug et al., 2023)	Hyperparameter optimization using Optuna framework
23	(Chowdhury et al., 2021)	Validation set, different loss functions, early stopping criterion
24	(Atila et al., 2021)	Early stopping technique during training
25	(Zhong & Zhao, 2020)	Data augmentation, DenseNet-121 architecture, performance monitoring and adjustment
26	(Agarwal et al., 2020)	Data augmentation, training process adjustments, and comparison with pre-trained models

27	(Mishra et al., 2020)	Dropout layers, hyper-parameter tuning, feature visualization
28	(Shrivastava & Pradhan, 2021)	10-fold cross-validation protocol
29	(Trivedi et al., 2021)	Dropout technique in the CNN architecture

2.10 Performance Evaluation and Metrics Used

LRQ9: What performance evaluation techniques and metrics are used, and why were these chosen?

The performance evaluation techniques and metrics used in the studies reviewed for LRQ9 are diverse and tailored to the specific needs of each study, as detailed in Table 2-9. (Wójtowicz et al., 2021) assess their Random Forest models using a range of metrics like Bias, Hit Rate, False Alarm Ratio, and Threat Score, providing a multifaceted assessment of model performance. (Bhosale et al., 2023), (Astani et al., 2022), and many other studies commonly use metrics like accuracy, precision, recall, and F1 score for a comprehensive evaluation of their models in identifying plant diseases. (Aurangzeb et al., 2020) include accuracy, specificity, and sensitivity metrics to evaluate their models. Employing 10-fold cross-validation (Sai & Patil, 2022) enhances robustness of performance evaluation. Some studies, like (Sharma et al., 2020) and (Harakannanavar et al., 2022), focus on classification accuracy, which is a direct reflection of model effectiveness. Others, such as (Singla et al., 2023), employ ROC curves and confusion matrices for a detailed analysis of model performance. These varied metrics and techniques reflect the complexity and diversity of challenges in plant disease detection and the necessity for comprehensive evaluation methods to assess the accuracy and reliability of the proposed models.

Table 2-9: Summary of Performance Evaluation and Metrics Used

Sl. No.	Referenced Paper	Performance Evaluation and Metrics Used
1	(Wójtowicz et al., 2021)	Bias, Hit Rate, Proportion Correct, False Alarm Rate, False Alarm Ratio, Threat Score
2	(Bhosale et al., 2023)	Accuracy, Precision, Recall, F1 score
3	(Astani et al., 2022)	Accuracy, Precision

4	(Aurangzeb et al., 2020)	Sensitivity, Specificity, Precision, FNR, FPR, Accuracy
5	(Sai & Patil, 2022)	Precision, Recall, F1-score, 10-fold cross-validation
6	(Kholiya et al., 2023)	Accuracy, Precision, Recall, F1-score
7	(Chaitanya Reddy et al., 2021)	RMSE, PSNR, Overall Accuracy
8	(Sharma et al., 2020)	Classification Accuracy
9	(Shakeel et al., 2020)	Precision, Recall, F1 Score
10	(Yousuf & Khan, 2021)	Accuracy, Precision, Recall
11	(Baheti et al., 2022)	Classification Accuracy
12	(Harakannanavar et al., 2022)	Precision, Recall, F-measure
13	(Singla et al., 2023)	ROC curves, Confusion matrices, Precision, Specificity, Accuracy, AUC-ROC
14	(Gosai et al., 2022)	Not Specified
15	(Kilaru & Raju, 2022)	Sensitivity, Specificity, Accuracy, Recall, Precision
16	(Meenakshi, 2023)	Classification Accuracy, Precision, Sensitivity, Specificity, FPR, F1-score
17	(Mohanty et al., 2022)	Precision, Recall, F1 score, Accuracy
18	(Sangeetha et al., 2022)	Accuracy, F-score, Recall, Precision
19	(Silviya et al., 2022)	Classification Accuracy
20	(Shivaprasad & Wadhawan, 2023)	Accuracy, Precision, Recall, F1 score
21	(Al-gaashani et al., 2022)	Accuracy, Precision, Recall, F1 score
22	(Pawar et al., 2022)	Accuracy Rates
23	(Rizvee et al., 2023)	Average Accuracy, Precision, Recall, F-score, Specificity
24	(Jasim & Al-Tuwaijari, 2020)	Accuracy, Confusion Matrix
25	(Habiba & Islam, 2021)	Classification Accuracy, Precision, Recall, Top-n Accuracy

26	(Ramiah Subburaj et al., 2023)	Accuracy, Precision, Recall, F1-score
27	(Lamba et al., 2021)	Accuracy, Recall, Specificity, Sensitivity, F-measure, MCC
28	(Aggarwal et al., 2023)	Accuracy, Recall Rate, Precision, F1-score, Mean Square Error
29	(Sujatha et al., 2021)	Precision, Recall, F1 score, Area Under the Curve (AUC)
30	(Kumar et al., 2022)	Accuracy, Training, and Validation Losses
31	(Hatuwal et al., 2020)	Accuracy, Precision, Recall, F1-score
32	(Ahmed & Yadav, 2022)	True Positive Rate, True Negative Rate, Precision, Recall, F1-score
33	(Chug et al., 2023)	Accuracy, F1-score, Execution Time
34	(Chowdhury et al., 2021)	Accuracy, Sensitivity, Specificity, Precision, F1 Score, IoU, Dice Coefficient
35	(Atila et al., 2021)	Accuracy, Sensitivity, Specificity, Precision
36	(Zhong & Zhao, 2020)	Precision, Sensitivity, F1 score, Accuracy
37	(Agarwal et al., 2020)	Training Accuracy, Validation Accuracy, Testing Accuracy
38	(Mishra et al., 2020)	Accuracy Measurement, Feature Visualization
39	(Shrivastava & Pradhan, 2021)	Classification Accuracy, Sensitivity, Specificity, Class-wise Accuracy, AUC, F-score
40	(Trivedi et al., 2021)	Test Epochs, Learning Rates, Accuracy Rates

2.11 Reported Accuracy of Different Approaches

LRQ10: What are the findings of the studies, and what is the reported accuracy of plant disease detection approaches?

The research papers reviewed in LRQ10 present a range of reported accuracies for various plant disease detection approaches, demonstrating the effectiveness of different models and techniques, as detailed in Table 2-10. (Wójtowicz et al., 2021) report high accuracy in identifying leaf rust symptoms in wheat and rye using hyperspectral imaging and machine

learning. (Bhosale et al., 2023) show that machine learning and deep learning models, especially CNNs, have a high categorization accuracy ranging from 76% to 100%. (Astani et al., 2022) ensemble classifier approach reaches up to 95.98% accuracy for classifying 13 classes of tomato disease. (Aurangzeb et al., 2020) achieve an accuracy range of 92.8% to 98.7% for selected plant diseases using integrated machine learning methods. Sai & Patil 2022 report a high-test accuracy of 98.125% with their Random Forest model. (Kholiya et al., 2023) find that deep learning models outperform traditional approaches, with accuracies ranging from 76% to 100%. These results, along with other studies like (Sharma et al., 2020), (Shakeel et al., 2020), and (Yousuf & Khan, 2021), underscore the high efficacy of various computational approaches in accurately detecting plant diseases, marking significant advancements in agricultural technology and disease management.

Table 2-10: Summary of Reported Accuracy

Sl. No.	Referenced Paper	Reported Accuracy (%)
1	(Wójtowicz et al., 2021)	High, varied across symptoms
2	(Bhosale et al., 2023)	ML & DL Models: 76 - 100
3	(Astani et al., 2022)	Ensemble Classifier: Up to 95.98
4	(Aurangzeb et al., 2020)	ML Models: 92.8 - 98.7
5	(Sai & Patil, 2022)	Random Forest: 98.125
6	(Kholiya et al., 2023)	DL Models: 76 - 100
7	(Sharma et al., 2020)	MDC: 81.06, Bayes' Classifier: ~69.358
8	(Shakeel et al., 2020)	~96
9	(Yousuf & Khan, 2021)	Ensemble classifier: 96, SVM: 93
10	(Baheti et al., 2022)	Random Forest: 98
11	(Harakannanavar et al., 2022)	SVM: 88, K-NN: 97, CNN: 99.6
12	(Singla et al., 2023)	Neural networks: 96.2
13	(Mohanty et al., 2022)	SVM: 73, Random Forest: 46
14	(Sangeetha et al., 2022)	VGG16 DL framework: 91.25
15	(Silviya et al., 2022)	CNN: 97, VGG16: 96, VGG19: 95
16	(Pawar et al., 2022)	DL Models: Up to 93
17	(Rizvee et al., 2023)	LeafNet: 98.55

18	(Jasim & Al-Tuwaijari, 2020)	CNN Training: 98.29, Testing: 98.029
19	(Habiba & Islam, 2021)	VGG16: ~95.5, Top-2 accuracy: 99
20	(Ramiah Subburaj et al., 2023)	Before fine-tuning: 97.08/83.52, After fine-tuning: 98.19/95.93
21	(Lamba et al., 2021)	Auto Color Correlogram & DL: Binary class: 99.4, Multiclass: 99.2
22	(Aggarwal et al., 2023)	Machine Learning: ~74, InceptionResNetV2: 88
23	(Sujatha et al., 2021)	VGG-16: 89.5
24	(Kumar et al., 2022)	AlexNet: 97.49, VGG16: 97.23
25	(Hatuwal et al., 2020)	CNN: 97.89
26	(Chug et al., 2023)	Hybrid DL Model: 87.55 – 100
27	(Chowdhury et al., 2021)	EfficientNet-B7: 99.95/99.12, EfficientNet-B4: 99.89
28	(Atila et al., 2021)	EfficientNet B5: 99.91, B4: 99.97
29	(Zhong & Zhao, 2020)	93.51 - 93.71
30	(Agarwal et al., 2020)	CNN 91.2
31	(Mishra et al., 2020)	DL Model: 98.40
32	(Shrivastava & Pradhan, 2021)	SVM: 94.65
33	(Trivedi et al., 2021)	CNN: 98.42 - 98.58

2.12 Balancing Techniques Used in Models

LRQ11: What balancing techniques are used for diverse datasets?

The studies in LRQ11 adopt various balancing techniques to manage diverse datasets in plant disease detection research, as detailed in Table 2-11. (Wójtowicz et al., 2021) utilize bootstrap resampling to ensure balanced representation in their dataset. (Bhosale et al., 2023) and (Astani et al., 2022) employ data augmentation methods, with (Bhosale et al., 2023) ensuring an equal number of images for each class, and (Astani et al., 2022) combining different databases for a more comprehensive dataset. (Sai & Patil, 2022) approach dataset diversity through comprehensive data collection and cross-validation, while (Kholiya et al., 2023) specifically augment data for classes with fewer images. (Mohanty et al., 2022) use a 'balanced' class weight

function in algorithms to address class imbalance. (Habiba & Islam, 2021), (Ramiah Subburaj et al., 2023), and (Chug et al., 2023) also rely on data augmentation techniques, with (Ramiah Subburaj et al., 2023) incorporating optimal fine-tuning for balance. (Aggarwal et al., 2023) use stratified sampling or balanced batches during training, and (Kumar et al., 2022) include a variety of plants and diseases in their dataset to ensure comprehensive representation. These techniques collectively demonstrate a keen focus on addressing data imbalance, ensuring that the models are trained on representative and varied data for robust plant disease detection.

Table 2-11: Summary of Balancing Techniques Used

Sl. No.	Referenced Paper	Balancing Techniques Used
1	(Wójtowicz et al., 2021)	Bootstrap resampling
2	(Bhosale et al., 2023)	Data augmentation, equal number of images for each class
3	(Astani et al., 2022)	Data augmentation, combining different databases
4	(Aurangzeb et al., 2020)	
5	(Sai & Patil, 2022)	Comprehensive dataset, cross-validation
6	(Kholiya et al., 2023)	Data augmentation for classes with fewer images
7	(Mohanty et al., 2022)	'Balanced' class weight function in algorithms
8	(Al-gaashani et al., 2022)	Concatenation-based approach
9	(Pawar et al., 2022)	Inherent balancing strategies in deep learning models (implied)
10	(Habiba & Islam, 2021)	Data augmentation (e.g., contrast adjustment)
11	(Ramiah Subburaj et al., 2023)	Data augmentation, optimal fine-tuning
12	(Aggarwal et al., 2023)	Stratified sampling, balanced batches during training
13	(Kumar et al., 2022)	Use of various plants and diseases in the dataset
14	(Chug et al., 2023)	Data augmentation, hyperparameter optimization
15	(Chowdhury et al., 2021)	Data augmentation (rotation, scaling, translation)
16	(Atila et al., 2021)	Data augmentation for classes with fewer samples
17	(Zhong & Zhao, 2020)	Data augmentation, multi-label classification, focus loss function
18	(Agarwal et al., 2020)	Data augmentation, standardized image sizes

19	(Shrivastava & Pradhan, 2021)	Comprehensive dataset, 10-fold cross-validation protocol
20	(Trivedi et al., 2021)	Standardizing image characteristics for a uniform dataset

2.13 Limitations and Research Gaps Identified

LRQ12: What are the identified limitations and research gaps in current studies?

The studies reviewed in LRQ12 identify several limitations and research gaps in current plant disease detection methodologies. Key challenges include difficulties in distinguishing certain disease symptoms, as highlighted by (Wójtowicz et al., 2021), who note the limitations in using vegetation indices and the risk of overfitting. (Bhosale et al., 2023) emphasize the need for larger and more diverse datasets to enhance model generalizability, particularly under real-world conditions. (Astani et al., 2022) point out challenges in training deep learning models due to complex field data and hardware constraints. (Aurangzeb et al., 2020) and (Kholiya et al., 2023) acknowledge the dependency of system performance on the quality and diversity of the dataset. (Sharma et al., 2020) and (Baheti et al., 2022) identify the need for handling diverse disease symptoms and large datasets for training robust models. (Aggarwal et al., 2023) and (Mohanty et al., 2022) discuss challenges in image acquisition, preprocessing, and feature extraction. (Silviya et al., 2022) highlight the difficulties in distinguishing diseases with similar characteristics. These and other studies collectively indicate a necessity for more comprehensive classification methods, expanded datasets, exploration of alternative algorithms, and improved spectral analysis techniques to develop more adaptable models for diverse agricultural environments.

2.14 Future Directions and Recommendations

LRQ13: Based on the findings and gaps, what future directions and recommendations can be suggested?

The research papers in LRQ13 collectively suggest several future directions and recommendations to advance plant disease detection. Many studies emphasize the need to broaden datasets to include more plant species and disease types. (Wójtowicz et al., 2021) and (Bhosale et al., 2023) suggest integrating models with satellite and aerial imagery and IoT for

real-time monitoring. (Astani et al., 2022) propose exploring deep learning models in ensemble classifiers. (Aurangzeb et al., 2020) recommend focusing on deep learning features like autoencoders. Expanding the scope to include more plant diseases and real-time applications is suggested by (Sai & Patil, 2022) and (Kholiya et al., 2023), (Sharma et al., 2020), and (Yousuf & Khan, 2021) call for improving robustness against varying image conditions and further model refinement. Additionally, integrating advanced machine learning techniques, enhancing model accuracy, generalizability, and efficiency, and exploring new feature extraction methods are common themes across these studies. These recommendations highlight a concerted effort toward developing more sophisticated, accurate, and versatile systems for efficient agricultural management and plant disease detection.

2.15 Discussion

- **Real-World Impact and Applications:** The application of ML and DL in plant disease detection, as demonstrated by the studies utilizing the PlantVillage dataset and hyperspectral imaging (Wójtowicz et al., 2021), has a profound impact on agricultural practices, particularly in developing countries. These technologies facilitate accurate and timely disease detection, critical for improving plant yields and quality (Rizvee et al., 2023). This is especially transformative in regions where agriculture is pivotal to the economy, thereby bolstering food security and advancing agricultural practices beyond traditional methods.
- **Challenges and Limitations:** Despite technological advancements, challenges persist in data availability and diversity. The reliance on specific datasets like PlantVillage, while beneficial, may not capture the global diversity of plant conditions (Observation for RQ2). Moreover, the quality of field images and their variability pose significant challenges in training robust models. The generalizability of these models is also a concern, as many are trained under controlled conditions, which may not adequately represent diverse agricultural environments (Observation for RQ3).
- **Future Research Directions:** Future research can explore integrating these technologies with the Internet of Things (IoT) and unmanned aerial vehicles (UAVs) for real-time data collection and large-scale surveillance, respectively. This integration can potentially enable early disease detection over extensive agricultural areas, a concept yet to be fully explored in the reviewed literature. Additionally, developing models capable of handling diverse and sub-optimal data conditions can significantly enhance their applicability in real-world settings.

- **Ethical and Environmental Considerations:** The ethical implications surrounding data privacy and security in agricultural data collection must be addressed, to ensure compliance with privacy standards (Observation for RQ2). The environmental impact of these technologies, especially concerning their energy consumption and ecological footprint, should also be a consideration in sustainable agricultural practices.
- In summary, the integration of ML and DL in plant disease detection represents a significant advancement in agriculture, offering both opportunities and challenges. Future research should focus on enhancing data diversity, model generalizability, and integrating emerging technologies, while also considering the ethical and environmental impacts of these advancements.

2.16 Chapter Summary

In concluding this comprehensive literature review of plant disease detection using Machine Learning (ML), Deep Learning (DL), and Image Processing, the following key pointers emerge:

- **Data Sources and Collection Methods (RQ1 and RQ2):** The review highlights the diversity in data sources and collection methods, ranging from hyperspectral imaging to online databases, as exemplified in studies like (Wójtowicz et al., 2021) and (Bhosale et al., 2023). This diversity is crucial for capturing the broad spectrum of plant diseases.
- **Variety of Plant Diseases (RQ3):** The research spans a wide range of plant diseases, emphasizing the need for versatile detection systems capable of handling different types of diseases, as seen in the work of (Astani et al., 2022).
- **Preprocessing and Data Augmentation (RQ4):** Techniques such as noise reduction, scaling, and augmentation, as discussed by (Aurangzeb et al., 2020), are vital for enhancing model performance.
- **Feature Extraction Techniques (RQ5):** The importance of advanced feature extraction methods, including CNNs and Gabor Wavelet Transform used by (Kholiya et al., 2023) and (Astani et al., 2022), is critical for accurate disease identification.
- **Classification Techniques (RQ7):** Various classification techniques, including Random Forest and CNNs, demonstrate effectiveness in disease categorization, as shown in studies like (Sharma et al., 2020).
- **Reducing Overfitting (RQ8):** Techniques like cross-validation and data augmentation, as employed in (Rizvee et al., 2023), are essential for building robust models.

- **Performance Evaluation (RQ9):** The use of metrics such as accuracy and precision for performance evaluation, highlighted in the work (Jasim & Al-Tuwaijari, 2020), is critical for assessing model efficacy.
- **Reported Accuracy (RQ10):** The varying levels of accuracy reported in different studies indicate the evolving nature of these technologies and the need for ongoing improvement.
- **Balancing Techniques (RQ11):** Addressing data imbalance through techniques like stratified sampling, as seen in (Chowdhury et al., 2021), is essential for training accurate models.
- **Limitations and Future Directions (RQ12 and RQ13):** The review identifies challenges such as the need for larger datasets and integration with IoT, suggesting future directions for research as proposed by (Sai & Patil, 2022).

These concluding points underscore the significant advancements made in plant disease detection using ML, DL, and image processing, while also highlighting areas for future research and development. The ongoing evolution of these technologies promises further enhancements in accuracy, efficiency, and applicability in agricultural settings, contributing to better disease management and sustainable farming practices.

Chapter 3

RESEARCH METHODOLOGY

3.1 Introduction

To enhance global food security, this study employs advanced DL and ML techniques for accurate detection of leaf diseases, with a focus on apple diseases such as Black Rot, Cedar Apple Rust, and Apple Scab. Using the PlantVillage Dataset, the research rigorously evaluates seven ML models — Logistic Regression (LR), Linear Discriminant Analysis (LDA), K-Nearest Neighbors (KNN), Classification and Regression Trees (CART), Support Vector Machine (SVM), Random Forest (RF), and Gaussian Naive Bayes (NB) - alongside CNN architectures like MobileNetV2 and NASNetMobile. The aim is to conduct a comprehensive analysis that extends beyond mere accuracy to include precision, recall, and F1-score. This section outlines the methodology, detailing the dataset, preprocessing, model selection, and evaluation metrics, all geared towards developing effective disease identification system for agriculture.

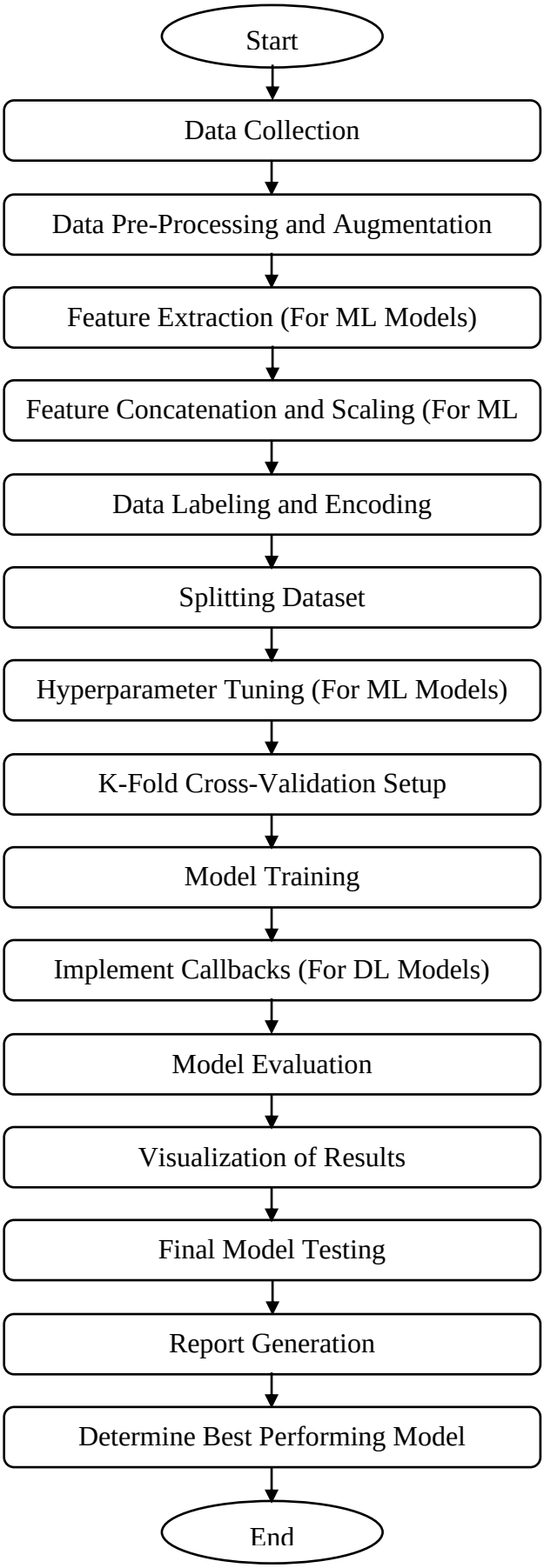


Figure 3-1: Flowchart depicting step-by-step flow for both ML and DL models.

As illustrated in flowchart Figure 3-1, below is a more detailed explanation of the flowchart steps:

- **Data Collection:** Gathering a diverse set of leaf images, and crucia for building a robust model.
- **Data Pre-Processing and Augmentation:** Utilizing techniques like resizing, normalization (rescaling pixel values), and augmentation methods (rotation, width and height shift, horizontal flipping) through **ImageDataGenerator**.
- **Feature Extraction (For ML Models):** Employing specific algorithms such as **fd_hu_moments** for Hu Moments extraction, **fd_haralick** for Haralick texture features, and **fd_histogram** for color histogram features.
- **Feature Concatenation and Scaling (For ML Models):** Combining various attributes into a unified feature vector and normalizing them using **MinMaxScaler**.
- **Data Labeling and Encoding:** Transforming categorical labels into numerical format with **LabelEncoder**.
- **Splitting Dataset:** Partitioning the data into training and testing segments using the **train_test_split** method.
- **Hyperparameter Tuning (For ML Models):** Fine-tuning machine learning models by utilizing **GridSearchCV** on algorithms like the Random Forest Classifier to determine the most effective settings.
- **Setting up Cross-Validation using the K-Fold method:** Implementing **StratifiedKFold** to ensure more reliable model validation, by making sure each fold is a good representative of the entire dataset.
- **Model Training:** Training ML models (like Logistic Regression, Decision Trees, KNN, etc.) and DL models (MobileNetV2, NASNetMobile) on the dataset.
- **Implement Callbacks (For DL Models):** Using **EarlyStopping** and **ReduceLROnPlateau** to optimize the training process and prevent overfitting in DL models.
- **Model Evaluation:** Assessing model performance with metrics like precision, recall F1-score, accuracy, and analyzing ROC and Precision-Recall curves.
- **Visualization of Outcomes:** Plotting accuracy, loss, ROC curves, precision-recall curves, and confusion matrices to visually interpret model performance.
- **Final Model Testing:** Evaluating the models on an independent test set to ensure real-world applicability.

- **Report Generation:** Documenting the process, findings, and evaluation results in a detailed report.
- **Determine Best Performing Model:** Comparing and selecting the most effective model based on the evaluation metrics and overall performance.

3.2 Dataset Description

The Plant Village dataset is a comprehensive collection of 54,309 images covering 14 different plant species, including Apple, Blueberry, Cherry, Corn, Grape, Orange, Peach, Bell Pepper, Potato, Raspberry, Soybean, Squash, Strawberry, and Tomato. This dataset encompasses images depicting 17 fungal diseases, 4 bacterial diseases, 2 mold (oomycete) diseases, 2 viral diseases, and 1 disease caused by a mite. Additionally, images of healthy leaves, devoid of visible disease symptoms, are included for 12 plant species. (Sai & Patil, 2022)

Dataset Link: https://github.com/spMohanty/PlantVillage-Dataset

The data for this study was sourced from the "Color" section found in the "raw" folder of the PlantVillage dataset. The modeling in this study is specifically centered on the analysis of Apple Leaves data. The train and test datasets are structured into two folders: Diseased and Healthy. The Diseased Folder encompasses images of leaves affected by Black Rot, Cedar Apple Rust, and Apple Scab, labeled as diseased or unhealthy. On the other hand, the Healthy Folder contains images of green and healthy leaves.

Image Attributes:

File Format: JPEG.

Bit Depth: 24.

Size: 256 x 256 pixels.

Resolution: 96 dpi (both vertical and horizontal).

Height: 256 Pixels.

Width: 256 Pixels.

3.3 Data Preprocessing

- **Data Collection and Initial Setup:** The PlantVillage dataset, encompassing a wide range of plant species and diseases, is employed. For this study, the focus is specifically on Apple leaves, examining diseases like Black Rot, Cedar Apple Rust, and Apple Scab. This subset selection is integral to narrowing down the research scope to targeted disease detection in apples.
- **Data Labelling and Encoding:** Label encoding is applied to transform categorical labels into numerical form. This step is crucial for both ML and DL models to facilitate the processing of labels by machine learning algorithms.
- **Splitting of Dataset:** The dataset is divided into training and testing subsets to maintain balance between the models' training and validation processes.
- **Stratified K-Fold Cross-Validation Setup:** Both ML and DL models undergo Stratified K-fold cross-validation. This technique is significant for assessing the models' performance and ensuring their effectiveness across different subsets of the dataset.

This process, as outlined, establishes a comprehensive approach to data preparation, crucial for the subsequent stages of model training and evaluation in detecting diseases in Apple leaves.

3.3.1 Transformation

As part of Transformation, the focus is on detailing the data preparation techniques for both Machine Learning and Deep Learning models. This segment is essential as it illustrates how raw image data, particularly of apple leaves, is methodically processed and transformed to facilitate effective disease detection.

In Machine Learning Models:

- **Image Pre-processing (ML Models):** Images are resized to a consistent dimension (500x500 pixels) to standardize input data. RGB to BGR conversion, followed by BGR to HSV transformation, and image segmentation techniques are applied to enhance key visual characteristics, crucial for effective feature extraction.
- **Feature Extraction (ML Models):** Features are extracted using techniques like Hu Moments (shape features), Haralick texture (texture features), and color histograms (color features). This comprehensive feature extraction is vital for capturing diverse aspects of the leaf images, which aids in accurate disease classification.

- **Feature Normalization:** Post feature extraction, normalization via MinMaxScaler is conducted to bring all features to a similar scale, ensuring no single feature disproportionately influences the model's learning process.

In Deep Learning Models:

- **Data Augmentation:** Employing ImageDataGenerators, the dataset is augmented through rotations, shifts, and flips. This not only increases data volume but also introduces variability, reinforcing the model's ability to generalize and reducing overfitting.
- **Utilization of Pre-Trained Models:** Deep Learning models incorporate MobileNetV2 and NASNetMobile, pre-trained on ImageNet. By fine-tuning only, the top layers, these models leverage transfer learning, harnessing previously learned features from a vast dataset to enhance disease detection in apple leaves.

3.4 Models

This study employs a blend of both machine learning (ML) and deep learning (DL) models to analyze and classify images of apple leaves. For the ML aspect, seven different models are utilized, each offering unique strengths and perspectives in data analysis. These include:

- Logistic Regression (LR),
- Linear Discriminant Analysis (LDA),
- K Nearest Neighbors (KNN),
- Decision Trees (DT),
- Random Forest (RF),
- Naïve Bayes (NB), and
- Support Vector Machine (SVM)

These models are chosen for their proven effectiveness in classification tasks and their ability to handle varied data structures and relationships.

In addition to these, the study also incorporates two advanced deep learning models, specifically leveraging Convolutional Neural Networks (CNNs) with two different architectures:

- NASNetMobile
- MobileNetV2

These CNN models are selected for their robustness in image recognition and classification tasks, benefiting from extensive pre-training on large and diverse datasets. NASNetMobile and MobileNetV2 are particularly known for their efficiency and accuracy in processing image data, making them well-suited for the task of detecting diseases in apple leaves.

Details and explanations for each of these models are provided below:

3.4.1 Logistic Regression

Logistic regression is a fundamental statistical model used for binary classification tasks. It predicts the probability of a binary outcome (e.g., 0 or 1, healthy or diseased) based on a set of independent variables or features. The model's output is a value between 0 and 1, representing the likelihood of a positive outcome (e.g., diseased).

The logistic regression model is represented by the following equation:

$$P(y = 1|x) = 1 / (1 + exp(-\beta^{\wedge}Tx)) \quad \textbf{3-1}$$

where:

- P (y = 1|x) is the probability of the positive outcome (diseased) given the input features (x)
- β is the vector of model coefficients
- x is the vector of input features
- exp is the exponential function
- T is the transpose operator

In Equation 3-1 the model coefficients (β) represent the strength of the relationship between each input feature and the probability of a positive outcome. A positive coefficient indicates a positive correlation, while a negative coefficient indicates a negative correlation.

Within the realm of identifying plant diseases, logistic regression is utilized to categorize images of leaves as healthy or afflicted, based on the analysis of features like color, texture, and form. The model coefficients would represent the relative importance of each feature in contributing to the disease prediction. The model's output would be the probability of a leaf being diseased given its extracted features.

3.4.2 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a dimensionality reduction technique that aims to transform high-dimensional data into a lower-dimensional subspace while preserving the maximum discrimination between classes. It finds a linear transformation that projects the data onto a lower-dimensional space where the separation between classes is maximized. This makes LDA particularly useful when dealing with a large number of features, as it reduces the computational complexity and improves classification accuracy.

The mathematical formulation of LDA is based on the concept of maximizing the ratio of between-class variance to within-class variance. This can be achieved by solving the following optimization problem:

$$\textbf{maximise: } W^T S_b W / W^T S_w W \qquad \textbf{3-2}$$

where:

- W is the transformation matrix that projects the data onto the lower-dimensional subspace
- S_b is the between-class scatter matrix
- S_w is the within-class scatter matrix

The solution to this optimization problem yields the optimal transformation matrix W that projects the data onto a subspace where the separation between classes is maximized.

Interpretation of LDA

LDA can be interpreted as a technique that finds the optimal set of directions in the high-dimensional space that projects the data onto a lower-dimensional subspace while preserving the maximum separation between classes. These directions are represented by the columns of the transformation matrix W, as derived in Equation 3-2.

For detecting plant diseases, Linear Discriminant Analysis (LDA) can be utilized to categorize images of leaves into healthy or diseased, based on analyzed features like color, texture, and shape. The optimal transformation matrix obtained from LDA, as detailed in Equation 3-2 would represent the most discriminating features for distinguishing between healthy and diseased leaves. This would allow for the development of accurate plant disease detection models using machine learning techniques.

LDA's ability to handle a large number of features and preserve class discrimination makes it a valuable tool for plant disease detection tasks. By reducing the dimensionality of the feature space, as demonstrated in Equation 3-2, LDA can improve the efficiency and accuracy of machine learning models, leading to more effective plant disease detection and management practices.

3.4.3 K Nearest Neighbors (KNN)

K Nearest Neighbors (KNN) is a non-parametric, instance-based learning algorithm that classifies new data points based on the majority class of their k nearest neighbors in the feature space. It is a simple and versatile algorithm that is well-suited for plant disease detection due to its non-linearity and robustness to noise.

The KNN algorithm can be summarized as follows:

- **Choose the number of neighbors (k):** Select the number of nearest neighbors to consider for classification. The choice of k is crucial for the performance of the algorithm. A small value of k might lead to overfitting, while a large value of k might ignore important local patterns.
- **Calculate the distances:** For each new data point, calculate the distance to all data points in the training set. Common distance metrics include Euclidean distance, Manhattan distance, and Minkowski distance. The choice of distance metric depends on the specific characteristics of the data.

- **Identify the k nearest neighbors:** Select the k data points that are closest to the new data point. These k neighbors represent the most similar data points to the new data point in the feature space.
- **Assign the class label:** Assign the class label that is most common among the k nearest neighbors to the new data point. This majority voting approach determines the class of the new data point based on the classes of its nearest neighbors.

For detecting plant diseases, the K-Nearest Neighbors (KNN) algorithm can be employed to classify images of leaves as either healthy or afflicted, utilizing extracted characteristics such as color, texture, and shape. The k nearest neighbors of a new leaf image represent the most similar leaf images in the training set, providing insights into its potential health status. The class label assigned to the new leaf image based on the majority class of its k nearest neighbors determines its classification as healthy or diseased.

KNN's simplicity, non-linearity, robustness to noise, and versatility make it a valuable tool for plant disease detection. By leveraging the similarity between leaf images based on extracted features, KNN can effectively classify new leaf images and aid in identifying potential plant diseases.

3.4.4 Decision Trees (DTs) or Classification and Regression Trees (CART)

Decision Trees (DTs) are non-parametric, supervised learning algorithms that represent decision-making processes in a tree-like structure. They are widely used for classification and regression tasks due to their simplicity, interpretability, and robustness to noise.

A decision tree consists of nodes and branches. Each node represents a decision point, where the data is split based on a particular feature. The branches represent the possible outcomes of the decision. The terminal nodes, also known as leaf nodes, represent the final classification or regression value.

The construction of a decision tree involves recursively splitting the data into smaller subsets based on the most informative features. The algorithm iteratively selects the feature that best separates the data into classes or regresses the target variable. This process continues until the data points reach a state of purity, where all data points belong to the same class or have similar target values.

Common splitting criteria used in decision tree algorithms include:

- **Information gain:** Measures the reduction in entropy, or uncertainty, when the data is split based on a particular feature.
- **Gini impurity:** A measure of the probability of misclassifying a randomly chosen data point if it is assigned to the most frequent class in the current node.
- **Chi-squared test:** It quantifies the relationship's statistical significance between a specific feature and the target variable.

Decision trees can overfit by becoming excessively complex and memorizing training data, rather than generalizing to new data. To combat this, pruning is employed to simplify the tree by eliminating superfluous branches, enhancing its ability to generalize effectively.

In the realm of identifying plant diseases, decision trees are an effective method for sorting leaf images into categories of healthy or diseased. This classification relies on analyzing distinct features such as color, texture, and shape extracted from the leaf images. The decision tree would recursively split the data based on these features, creating a series of rules that effectively capture the characteristics of healthy and diseased leaves.

Decision trees are well-suited for plant disease detection due to their simplicity, interpretability, and robustness to noise. By providing a clear understanding of the decision-making process, decision trees can aid in identifying critical features for disease detection and developing effective disease management strategies.

3.4.5 Random Forest

Random Forest (RF) is an ensemble learning method that combines multiple decision trees to improve the overall classification or regression performance. It is widely used for classification and regression tasks due to its robustness to noise, ability to handle complex relationships, and out-of-the-box feature importance measures.

Ensemble learning is a technique that combines multiple models to improve the overall performance of a single model. By aggregating the predictions of multiple models, ensemble learning can reduce the variance and bias of individual models, leading to more robust and accurate predictions.

The Random Forest algorithm consists of the following steps:

- **Random Subsampling:** For each tree in the forest, a random subset of the training data is drawn. This technique, known as bootstrapping, reduces the correlation between trees and helps prevent overfitting.
- **Random Feature Selection:** At each split node, a random subset of features is considered for splitting. This technique, known as random feature selection, further decorrelates the trees and reduces the risk of overfitting.
- **Tree Construction:** Each tree is constructed using the chosen random subset of data and features. The tree is grown until a predefined stopping criterion is met, such as a maximum depth or minimum node size.
- **Majority Voting:** For classification tasks, the class prediction of a new data point is determined by the majority vote of the trees in the forest. For regression tasks, the average prediction of the trees is used.

In the realm of detecting plant diseases, the Random Forest algorithm is utilized to distinguish between healthy and diseased leaf images. This distinction is based on the evaluation of extracted features such as color, texture, and shape. The ensemble of decision trees captures complex relationships between these features and the disease status, leading to robust and accurate disease detection.

Random Forest's ability to handle noisy data, capture non-linear relationships, and provide feature importance makes it a valuable tool for plant disease detection. By leveraging the collective wisdom of multiple trees, Random Forest can effectively classify leaf images and aid in identifying potential plant diseases.

3.4.6 Naïve Bayes

Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem. It assumes that features are independent of each other, which is often an oversimplification of real-world data. However, despite this assumption, Naive Bayes often performs well in practice due to its simplicity and efficiency.

Bayes' theorem is a fundamental concept in probability theory that allows us to update our beliefs about a hypothesis given new evidence. It can be expressed as follows (see Equation 3-3):

$$P(H \mid E) = (P(E \mid H) * P(H)) / P(E) \quad \text{3-3}$$

where:

- P(H) is the prior probability of the hypothesis
- P(E) is the probability of the evidence
- P(E | H) is the probability of the evidence given the hypothesis

The Naive Bayes classifier applies Bayes' theorem, as shown in Equation 3-3 to classification tasks. It assumes that the features of the data are independent of each other, which means that the probability of one feature does not affect the probability of another feature given the class label. This assumption is often oversimplified, but it allows for a simple and efficient implementation of the algorithm.

The Naive Bayes algorithm can be summarized as follows:

- **Compute the prior probabilities:** Calculate the probability of each class (e.g., healthy or diseased) in the training data.
- **Calculate the conditional probabilities:** For each feature and class, calculate the probability of the feature given the class.
- **Apply Bayes' theorem:** For a new data point, Bayes' theorem is used to determine the likelihood of belonging to a particular class based on its attributes, as outlined in Equation 3-3.
- **Assign the class label:** Assign the class label that has the highest probability according to Bayes' theorem.

In plant disease detection, the Naive Bayes classifier is adept at segregating leaf images into healthy or diseased categories, utilizing key extracted attributes like color, texture, and shape. The independence assumption of Naive Bayes, central to Equation 3-3, may not hold perfectly for these features, but it can still provide a reasonable approximation of the true relationships.

Naive Bayes' simplicity, efficiency, and robustness to noise make it a valuable tool for plant disease detection. By leveraging the probabilistic framework of Bayes' theorem as represented

in Equation 3-3, Naive Bayes can effectively classify leaf images and aid in identifying potential plant diseases.

3.4.7 Support Vector Machine (SVM)

Support Vector Machines (SVMs) are a powerful set of machine learning algorithms used for classification and regression tasks. They are based on the principle of finding a hyperplane that maximizes the margin between two classes of data points. This hyperplane effectively separates the data points into their respective classes, allowing for the accurate classification of new data points.

A hyperplane is a decision boundary in a high-dimensional space that divides the data points into two classes. The margin represents the gap between the separating hyperplane and the nearest data points of each class. A wider margin suggests enhanced class separation, leading to improved classification efficacy.

SVMs aim to find the optimal hyperplane that maximizes the margin between the two classes. This optimization problem ensures that the chosen hyperplane generalizes well to unseen data and achieves high classification accuracy.

SVMs were originally designed for linearly separable data, meaning that the data points can be perfectly separated by a straight-line hyperplane. However, real-world data is often non-linear, requiring more sophisticated techniques to find an effective hyperplane. Kernel functions are introduced to map the data into a higher-dimensional space where a linear hyperplane can be used to separate the data points.

Common kernel functions used in SVMs include:

- Linear kernel: Suitable for linearly separable data.
- Polynomial kernel: Maps data into a polynomial space, handling non-linear relationships.
- Radial basis function (RBF) kernel: A versatile kernel that works well for various data types.

In the field of detecting plant diseases, Support Vector Machines (SVMs) can be used for classifying leaf images into healthy or diseased categories. This classification relies on the

analysis of extracted features including color, texture, and shape. The SVM algorithm effectively finds a hyperplane that separates healthy leaf images from diseased ones, allowing for accurate classification of new leaf images.

SVMs' robustness to outliers, ability to handle high-dimensional data, and effectiveness in non-linear settings make them a valuable tool for plant disease detection. By leveraging the concept of maximizing the margin, SVMs can effectively differentiate between healthy and diseased leaf images and aid in identifying potential plant diseases.

3.4.8 MobilenetV2

MobilenetV2 serves as a crucial CNN architecture in identifying leaf diseases. Designed primarily for mobile and embedded vision applications, MobilenetV2 stands out due to its efficiency in computational power and memory usage. This is particularly beneficial for scenarios requiring the processing of extensive image data, such as in the analysis of plant leaves for disease identification. The architecture of MobilenetV2 is characterized by its innovative use of inverted residuals and linear bottlenecks, which enhance the traditional residual connections through lightweight depth-wise separable convolutions. These depth-wise separable convolutions are instrumental in reducing the model's parameter count, thus ensuring a lightweight framework without significantly compromising performance accuracy.

The efficiency of MobilenetV2 in processing image data is especially relevant in the context of leaf disease detection, where handling a multitude of images efficiently is paramount. Despite its streamlined architecture, MobilenetV2 does not fall short in maintaining a high accuracy level, a crucial factor for correctly diagnosing specific diseases from leaf imagery. Its suitability for deployment in mobile applications further augments its applicability in this study, enabling real-time disease detection in field-based scenarios, which is often a necessity for farmers and agricultural specialists. Additionally, the model's compatibility with transfer learning approaches allows for leveraging pre-trained networks on comprehensive datasets, such as ImageNet. This method is advantageous as it utilizes the pre-existing knowledge base of the model, refining it further to recognize specific patterns indicative of various leaf diseases. Such an approach not only economizes on training resources but also enhances the model's capability to discern subtle features characteristic of leaf diseases.

In summary, the selection of MobilenetV2 as a core component of this study's methodology is justified by its optimal balance between computational efficiency and high accuracy in image classification tasks. Its adaptability to mobile platforms and the ability to harness transfer learning techniques further solidify its suitability for the task at hand, namely, the efficient and accurate detection of plant leaf diseases.

3.4.9 NASNetMobile

NASNetMobile, a variant of the Neural Architecture Search Network (NASNet) optimized for mobile environments, is employed for detecting plant leaf diseases. The selection of NASNetMobile is based on its unique attributes that align with the requirements of agricultural applications, particularly for in-field disease diagnosis.

NASNetMobile’s architecture, developed through an automated neural architecture search (NAS), is designed for operational efficiency in mobile and embedded devices. This efficiency is crucial for processing large datasets of leaf images without compromising on computational speed or accuracy. NASNetMobile is adept at extracting intricate features from these images, crucial for identifying subtle disease indicators on plant leaves.

The model's transfer learning capabilities are particularly advantageous. By fine-tuning NASNetMobile with a specific plant leaf dataset, it leverages pre-existing knowledge to enhance disease detection accuracy. This approach is valuable for recognizing diverse patterns of leaf diseases across various plant species.

A significant advantage of NASNetMobile is its robustness in varying imaging conditions typical in agricultural settings, such as different lighting and backgrounds. This robustness ensures reliable performance, a critical requirement for practical applications in agriculture.

Furthermore, the computational efficiency of NASNetMobile makes it well-suited for real-time, field-based disease detection applications, enabling its deployment in portable devices used directly in agricultural fields. This capability is essential for providing timely insights for disease management and crop health monitoring.

In conclusion, NASNetMobile is chosen for its balanced integration of efficiency, accuracy, and robustness, making it an ideal CNN model for the targeted task of plant leaf disease

detection in this research. Its suitability for mobile deployment aligns with the practical needs of agricultural technology, offering a promising tool for enhancing disease detection and management practices in farming.

3.5 Evaluation Metrics

In the field of plant leaf disease detection, employing appropriate evaluation metrics is crucial for assessing the performance of machine learning models. Precision, recall, f1-score, and support are among the key metrics used for this purpose. Here's a detailed explanation of each metric and the rationale behind their selection in the context of leaf disease detection:

3.5.1 Precision

In the context of leaf disease detection, precision plays a pivotal role in the effectiveness and reliability of the models used. Defined in Equation 3-4, precision is the proportion of true positive predictions out of all positive predictions made by the model, mathematically expressed as:

$$\textbf{Precision} = \textbf{True Positives} / \textbf{True Positives} + \textbf{False Positives} \qquad \textbf{3-4}$$

This metric assumes heightened significance in agricultural settings, where the repercussions of false positives, such as misclassifying healthy leaves as diseased, are substantial. Such misclassifications could lead to unnecessary treatments, wastage of resources, and potential disruption of the ecological balance.

Achieving high precision in leaf disease detection models is crucial for several reasons. Firstly, it ensures the optimization of resources, meaning that treatments for plant diseases are judiciously applied only where necessary. This optimization is critical in agriculture where resource allocation directly impacts productivity and cost-effectiveness. Secondly, the precision of these models influences the trust that farmers and agricultural professionals place in technology. High precision in disease detection models assures reliability in the recommendations provided, which is essential for decision-making in agricultural practices.

It is also crucial to find an equilibrium between precision and recall. Recall, which quantifies the model's accuracy in correctly detecting actual diseased leaves, needs to be harmonized with

precision to ensure optimal performance. Over-emphasizing precision could lead to a model being overly conservative, missing diseased leaves, and resulting in high false negatives. This balance is critical for creating a model that is both accurate and practical.

Precision is also key in model optimization, particularly in threshold tuning. By adjusting the classification threshold, one can set a balance between correctly identifying diseased leaves and not wrongly labeling healthy leaves as diseased. Additionally, leaf disease detection often involves dealing with imbalanced datasets where healthy leaves outnumber diseased ones. In such scenarios, ensuring high precision is imperative to prevent the model's predictions from being skewed towards the majority class.

In summary, within the methodology of leaf disease detection, precision is a fundamental metric. It not only reflects the model's accuracy in identifying diseased leaves but also serves as a critical factor in refining the model. The goal is to achieve an optimal balance that maximizes accurate detection while maintaining practical applicability in the diverse and challenging field of agriculture.

3.5.2 Recall (Sensitivity)

Recall, also known as sensitivity, plays a pivotal role in the evaluation of our machine learning models. Recall is essentially an indicator of the model's proficiency in accurately identifying all true occurrences of a specific condition, such as disease in plant leaves. This metric is computed using the following formula:

True Positives / True Positives + False Negatives **3-5**

This formula (Equation 3-5) highlights the importance of recall in quantifying the effectiveness of a model in detecting diseased leaves among all the diseased leaves. The significance of recall in leaf disease detection is particularly critical. In agricultural practices, early and accurate disease detection is essential. A model's ability to minimize false negatives — cases where diseased leaves are overlooked — is crucial. A high recall value ensures comprehensive identification of diseased leaves, essential for effective disease management and control. Not detecting a diseased leaf could lead to the spread of the disease, resulting in significant crop damage.

However, the methodology also recognizes the importance of balancing recall with precision. While recall assesses the model's ability to detect diseased leaves, precision evaluates the accuracy of these detections. A focus solely on maximizing recall could lead to increased false positives, misclassifying healthy leaves as diseased, which could lead to unnecessary interventions.

Therefore, the methodology adopts a comprehensive evaluation strategy that emphasizes optimizing recall while maintaining a balance with precision. This approach ensures the development of a detection model that is sensitive to disease presence and accurate in diagnosis, offering a reliable tool for plant health management. Through this balanced evaluation framework, the methodology aims to enhance disease detection and management in precision agriculture, contributing to optimal crop health and yield.

3.5.3 F1-Score

The F1-score is a crucial evaluation metric in the field of machine learning, particularly in applications like plant disease detection, where it serves as a balanced measure of a model's precision and recall. The F1-score harmonizes the precision and recall of a classifier into a single metric by taking their harmonic mean. This is especially valuable in scenarios where an equal trade-off between precision (the model's accuracy in predicting disease) and recall (the model's ability to identify all actual cases of disease) is sought. The mathematical formula for the F1-score is given by:

$$\text{F1-Score} = 2 * ((\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})) \quad (3\text{-}6)$$

This equation (Equation 3-6) indicates that the F1-score is the harmonic mean of precision and recall, providing a single score that balances both the false positives (healthy leaves incorrectly identified as diseased) and false negatives (diseased leaves not identified).

In the context of plant disease detection, the F1-score is particularly significant. It provides a more comprehensive view of the model's performance than analyzing precision and recall separately. This is essential in agricultural applications where both the consequences of

misidentifying healthy plants as diseased (false positives) and missing actual cases of disease (false negatives) can be significant.

For instance, a high precision but low recall scenario might mean that while most of the identified diseased leaves are correctly identified, many actual cases of disease are being missed. Conversely, high recall but low precision could lead to over-treatment, as many healthy leaves are incorrectly flagged as diseased. The F1-score encapsulates both these aspects, ensuring that the model is not biased towards either precision or recall.

Utilizing the F1-score in model evaluation ensures that the developed system for plant disease detection is not just accurate but also reliable and practical for real-world agricultural use. It assists in achieving a balanced, effective approach to identifying and managing plant diseases, which is crucial for maintaining crop health and productivity.

Therefore, the F1-score, as a singular measure encompassing both aspects of model accuracy, forms a critical part of the evaluation framework, ensuring that the disease detection models are robust, reliable, and suitable for application in the field of agriculture.

3.5.4 Support

Support, as a metric in the evaluation of machine learning models for leaf disease detection, plays an essential role in providing a comprehensive understanding of the dataset used. It refers to the number of actual occurrences of each class within the dataset, such as the count of diseased and healthy leaves. The importance of support in the context of leaf disease detection extends beyond a mere numerical count; it offers critical insights into the composition and distribution of classes in the dataset.

Understanding the support for each class is fundamental in recognizing and addressing potential data imbalances. In plant disease detection, datasets often exhibit an imbalance, with a predominance of healthy leaves over diseased ones. This imbalance, if unaddressed, can lead to models that are biased toward the majority class, potentially compromising the detection of diseased leaves. Therefore, knowing the support aids in contextualizing other performance metrics such as precision, recall, and accuracy, ensuring a more accurate interpretation of the model's effectiveness.

Moreover, support guides the training and validation process of the model. In cases where there is a significant class imbalance, strategies like resampling techniques or the use of weighted loss functions might be necessary to ensure that the model does not overlook the minority class, in this case, the diseased leaves. This is crucial for developing a model that is not only accurate but also fair and effective across all classes.

Additionally, in the practical application of leaf disease detection, the support metric has implications for agricultural strategies. For less common diseases, as indicated by lower support, there might be a need for more focused attention to ensure these rarer instances are accurately detected by the model.

In conclusion, while support does not directly measure the performance of the model, its role in providing an understanding of the dataset's class distribution is invaluable. It ensures that the performance metrics of the model are interpreted correctly and that the model is trained and validated to effectively identify both common and rare instances of leaf diseases. This thorough understanding of the dataset through the lens of support is instrumental in ensuring the development of reliable, unbiased, and practical machine learning models for leaf disease detection.

Why These Metrics are Chosen for Leaf Disease Detection?

- **Comprehensive Evaluation:** Together, these metrics provide a comprehensive view of the model's performance, covering aspects of accuracy (precision), completeness (recall), and a balance of the two (F1-score).
- **Handling Imbalanced Data**: Plant disease datasets can often be imbalanced, with many more healthy leaves than diseased ones. Precision, recall, and F1-score are particularly useful for understanding model performance in such scenarios.
- **Decision-Making**: The choice of treatment or intervention in agricultural practices can depend heavily on the accuracy of disease detection. Precision and recall help in making informed decisions by understanding the likelihood of false alarms (false positives) and missed detections (false negatives).
- **Model Optimization:** These metrics are instrumental in tuning and optimizing the model. They help in identifying areas where the model is lacking and guiding adjustments to improve its overall effectiveness.

In summary, precision, recall, f1-score, and support are chosen for their ability to provide a detailed and nuanced assessment of a model's performance in plant leaf disease detection. They help ensure that the model not only accurately identifies diseased leaves but also minimizes the chances of misclassification, which is crucial for effective plant disease management.

3.6 Cross-Validation

To ensure robust model evaluation and to gauge the generalizability of both machine learning and deep learning models, this research employs a meticulous Stratified K-Fold Cross-Validation technique. This method partitions the dataset into 'k' equal subsets. In each iteration, one subset is reserved for validation while the others are utilized for training. This cycle is repeated 'k' times, with each subset being used exactly once as the validation data. Such a comprehensive approach not only minimizes bias but also offers a deeper understanding of the model’s performance on new, unseen data. This is particularly vital in the context of diagnosing diseases in Apple leaves, where the accuracy and reliability of the model are paramount, and the dataset size is relatively constrained.

3.7 Selection of the Best Model

The study meticulously analyses and compares the performance of seven machine learning models and two deep learning models. The evaluation focuses on key metrics such as accuracy, precision, recall, and F1 scores. This comprehensive analysis aims to identify the model that demonstrates superior performance in diagnosing Apple leaf diseases. The selected model, distinguished by its high accuracy and robustness, will be deemed most effective for disease detection. This section details the rationale behind the selection, highlighting how the model's specific strengths align with the research objectives and the complexities of detecting plant-based diseases. This careful selection process ensures the reliability and applicability of the chosen model in practical scenarios.

3.8 Requirements Resources

To effectively run the provided machine learning (ML) and deep learning (DL) code, it's essential to ensure that hardware, software, and dataset meet the following requirements:

3.8.1 Hardware Requirements

- A laptop or desktop with at least 16GB RAM is recommended for efficient data processing and model training, especially for large datasets.
- Intel i7 or higher, or equivalent AMD processor, with at least four cores for optimal performance in computations.
- A dedicated NVIDIA GPU (e.g., GTX 1060 or higher) is highly recommended for accelerating deep learning model training. CUDA and cuDNN libraries should be installed for GPU utilization.
- Solid State Drive (SSD) with at least 256GB of storage for faster data access and processing.

3.8.2 Software Requirements

- Operating System: Windows 10/11.
- Python programming language for code implementation (version 3.9.12).
- Anaconda Distribution: Recommended for managing Python packages and environments.
- Web browser for general access and research.
- Python Packages:
    - NumPy: Fundamental package for array operations in Python.
    - Mahotas: Computer vision and image processing library used for Haralick texture feature extraction.
    - OpenCV: Library for computer vision tasks, including image and video processing.
    - scikit-learn: Machine learning library providing simple tools for data analysis and modeling.
    - seaborn: Data visualization library based on Matplotlib for statistical graphics.
    - Pandas: Pandas is a Python library for data manipulation and analysis.
    - TensorFlow: Version 2.5 or newer for access to the latest DL features and GPU support.
    - h5py: Python library for storing and managing large datasets in Hierarchical Data Format.
    - os: Provides a way of interacting with the operating system.
    - glob: Finds all pathnames matching a specified pattern.

- matplotlib: Comprehensive library for creating static, animated, and interactive visualizations in Python.

- Cloud Resources:
  - Amazon Web Services (AWS): Provisioning virtual machines (VMs) with customizable hardware configurations (e.g., vCPUs, RAM, and GPUs) to scale computational resources as needed.
  - Google Colab Pro: For extended access to high-end GPUs (e.g., Tesla P100) and Tensor Processing Units (TPUs) in a cloud-based Jupyter notebook environment, beneficial for longer training sessions and larger datasets.
- Backup Integrated Development Environment (IDE): Jupyter Notebook or Visual Studio Code.

3.8.3 Dataset Requirements

- As part of this research, code is designed to work with the PlantVillage dataset, which can be obtained from the following link: PlantVillage Dataset.

3.9 Chapter Summary

This chapter meticulously outlines the methods employed in evaluating various machine learning and deep learning models for detecting diseases in apple leaves. It begins with a description of the data collection process from the PlantVillage Dataset, emphasizing apple leaves, and proceeds to detail the preprocessing steps, which include feature extraction and data augmentation. The chapter then delves into the implementation of several ML models, such as Logistic Regression, KNN, and SVM, alongside DL models like MobileNetV2 and NASNetMobile. A critical aspect of the methodology is the use of K-Fold Cross-Validation, ensuring a robust evaluation of these models. The chapter thoroughly discusses evaluation metrics including precision, recall, F1-score, and support, and culminates in the selection of the most effective model based on these metrics. This comprehensive approach highlights the rigorous and systematic process undertaken to compare and contrast the effectiveness of different models for practical applications in the domain of agricultural disease management.

Chapter 4

ANALYSIS AND DESIGN

4.1 Introduction

Chapter 4 delves into the intricate analysis and design of the machine learning (ML) and deep learning (DL) models tailored for plant disease classification. This section outlines the comprehensive approach undertaken to process and augment image data, alongside the rationale and implementation of various ML and DL algorithms. Emphasizing feature extraction, model architecture, and evaluation metrics, sets the stage for understanding the models' effectiveness in disease detection, providing a bridge between theoretical concepts and practical applications in agricultural technology.

4.2 Image Data Collection

In this research, a pivotal aspect of the methodology is the collection and preparation of image data essential for training and evaluating the machine learning and deep learning models. The dataset employed in this study is sourced from the PlantVillage dataset, a renowned repository known for its comprehensive collection of agricultural plant images including a variety of diseases.

4.2.1 Plant Village Dataset

The PlantVillage dataset, comprising 54,309 high-quality images across 14 different plant species, was employed for this study, with a focus on apple leaves. The images were categorized into two main groups: Diseased and Healthy. The diseased category includes images of apple leaves affected by three significant diseases: Apple Scab, Black Rot, and Cedar Apple Rust. The healthy category consists of images showing apple leaves without any visible signs of disease, highlighting their natural green appearance.

Dataset Link: https://github.com/spMohanty/PlantVillage-Dataset

4.2.2 Image Attributes

The image files are in JPEG format with 96 dpi resolution with a bit depth of 24 and dimensions of 256 x 256 pixels.

4.2.3 Image Collection Process

The image collection process involved a meticulous selection of images from the "Color" folder within the "raw" directory of the PlantVillage dataset. A total of 800 images per class (Diseased and Healthy) were curated to ensure a balanced representation. Each image was carefully reviewed to confirm its classification, ensuring the accuracy of the dataset.

4.2.4 Illustrative Examples of Apple Leaf Conditions

This section provides a visual representation of the dataset used in the study, showcasing examples of healthy and diseased apple leaves. These images are instrumental in demonstrating the dataset's diversity and the visual symptoms associated with each condition, aiding in the understanding of the challenges in disease detection and classification.

Figure 4-1: Examples of Healthy Apple Leaves

Figure 4-1 depicts the natural, unaffected state of apple leaves, showcasing their vibrant green color and uniform texture, free from any signs of disease. Each image represents the diversity within the healthy category of the dataset, highlighting the variance in leaf shapes and sizes found in healthy apple plants.

Figure 4-2: Examples of Diseased Apple Leaves

Figure 4-2 depicts apple leaves affected by common diseases, including Apple Scab, Cedar Apple Rust, and Black Rot. Each image showcases distinct pathological features characteristic of its respective disease: the dark, olive-green spots of Apple Scab, the orange-red lesions of Cedar Apple Rust, and the circular, black lesions surrounded by a yellow halo of Black Rot. These examples highlight the critical visual symptoms that facilitate the automated detection and classification of diseases in apple leaves.

4.3 Analysis and Design for ML Models

The section introduces the comprehensive procedures involved in image pre-processing and feature extraction specifically designed for machine learning applications. It establishes a fundamental understanding of the methodical approach and computational tactics crucial for boosting the effectiveness and precision of models tasked with diagnosing apple leaf diseases.

4.3.1 Image Pre-Processing for ML Models

The preprocessing of image data constitutes a foundational step in the workflow of machine learning models, ensuring that the input data is optimized for the best possible outcomes in disease detection. This section articulates the preprocessing techniques applied to the apple leaf images from the PlantVillage dataset, preparing them for subsequent analysis and model training.

4.3.1.1 Resizing

To ensure uniform input dimensions for the models and enhance computational efficiency, each image was resized to a standard resolution of 500x500 pixels. This resizing was accomplished using the **resize** function from the OpenCV library, a widely utilized tool in image processing

tasks. The specific code snippet employed for this operation was **image = cv2.resize(image, (500, 500))**, where cv2 refers to the OpenCV library and **(500, 500)** specifies the target dimensions in pixels.

4.3.1.2 BGR to RGB Color Format Conversion

To align with the standard RGB color format used in most image processing and analysis applications, the images initially in BGR format were converted to RGB. This conversion is crucial for consistent color representation and subsequent analysis. The transformation was performed using the **cvtColor** function from the **OpenCV** library, a staple in image processing tasks. The code employed for this conversion was **cv2.cvtColor(image, cv2.COLOR_BGR2RGB)**, where **cv2.COLOR_BGR2RGB** specifies the conversion from BGR to RGB Color space.

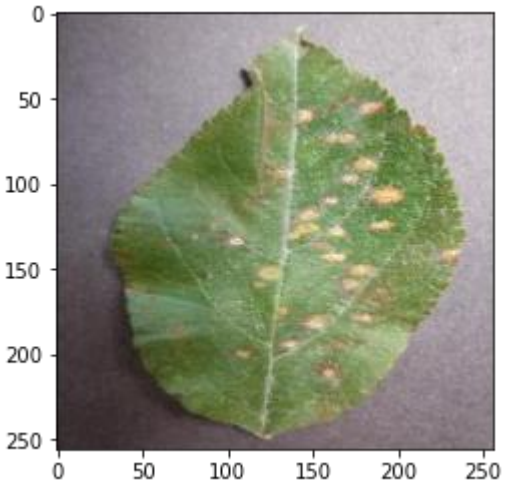


Figure 4-3: RGB Converted Apple Leaf Image

Figure 4-3 demonstrates an apple leaf after conversion from the original BGR format to the RGB color space, highlighting the enhanced color accuracy and consistency achieved through this process. The conversion facilitates more accurate color representation for subsequent image processing and analysis tasks.

4.3.1.3 RGB to HSV Color Space Conversion

The transformation of images into the HSV (Hue, Saturation, Value) color space was a subsequent step after RGB conversion, aimed at further refining the image analysis process. The HSV color model is particularly advantageous for image processing applications as it

separates the color information (hue) from the intensity or brightness (value), which aligns more closely with how humans perceive color. This separation is beneficial for addressing challenges such as shadow removal and compensating for variations in lighting, which are prevalent in plant disease detection.

This conversion process utilized the **cvtColor** function from the **OpenCV** library, renowned for its extensive image-processing functionalities. The conversion code **cv2.cvtColor(image, cv2.COLOR_RGB2HSV)** specifies the transition from RGB to HSV color space, facilitating enhanced color segmentation and analysis.

Advantages of HSV Conversion:

- **Hue**: This represents the type of color and is measured as a degree on the color wheel, with values ranging from 0 to 360. This characteristic enables the effective segmentation and identification of colors within an image, facilitating targeted analysis based on color attributes.
- **Saturation**: Indicates the intensity or purity of the color. Higher saturation values denote more vivid colors, while lower values indicate more muted tones. This differentiation is crucial for distinguishing between vibrant and subdued color areas in images, enhancing the analysis of color-based features.
- **Value (Brightness)**: Corresponds to the brightness of the color. By adjusting the value component, the brightness of an image can be modified without altering its color content, which is essential for consistent image analysis across varying lighting conditions.

The adoption of the HSV color space is instrumental in this research for its ability to simplify color-based image analysis and improve the robustness of the models against variations in lighting conditions, thereby enhancing the accuracy of disease detection in apple leaves.

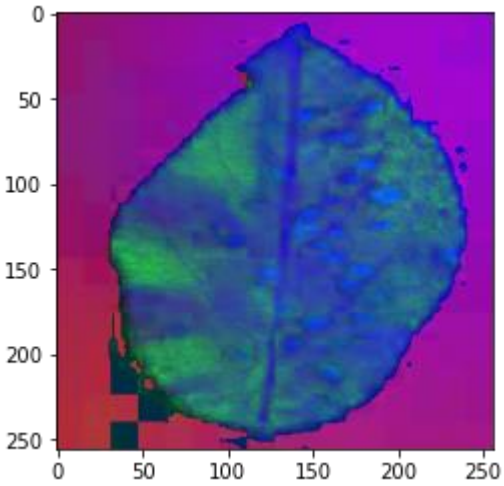


Figure 4-4: HSV Converted Image

Figure 4-4 depicts an apple leaf that has undergone conversion into the HSV (Hue, Saturation, Value) color space. This transformation highlights the leaf's textural and color variations more distinctly than in the standard RGB color space. Hue variations are represented by the color tones across the leaf, illustrating the segmentation potential for identifying specific regions of interest. The saturation is indicated by the intensity of these colors, where more vivid colors correspond to areas of greater saturation. The value component, while not directly visible in a single image, affects the brightness and is adjusted to emphasize or reduce the visibility of certain features. Overall, this conversion facilitates more nuanced image analysis, particularly useful for detecting and classifying leaf diseases.

4.3.1.4 Image segmentation for extraction of green and brown color

Image segmentation techniques were meticulously employed to isolate the healthy and diseased regions within the leaf images, a crucial step for the accurate identification and classification of plant diseases. This process involved the utilization of color-based segmentation to extract specific color ranges that correspond to healthy (green) and diseased (brown) areas of the leaves.

Using the **OpenCV** library, which is renowned for its powerful image-processing capabilities, the segmentation was carried out with the following approach:

- A mask was created to filter out the green regions representing healthy leaf tissue by defining a lower and upper threshold for the green color in the HSV space (**lower_green =**

np.array([25,0,20]), upper_green = np.array([100,255,255])). This was achieved using the **cv2.inRange** function, which checks if array elements lie between the elements of two other arrays.

- Similarly, a mask for brown regions indicative of disease presence was created **(lower_brown = np.array([10,0,10]), upper_brown = np.array([30,255,255])**).
- The **cv2.bitwise_and** function was then applied to the original RGB image and each mask, to extract the respective healthy and diseased portions of the image.
- Both masks were combined to create a final segmentation mask (**final_mask = healthy_mask + disease_mask**), which was used to generate the segmented image that highlights both healthy and diseased tissue for further analysis.

The result is a segmented image that distinctly identifies and differentiates between healthy and diseased tissue, thus enabling more precise disease diagnosis and classification.

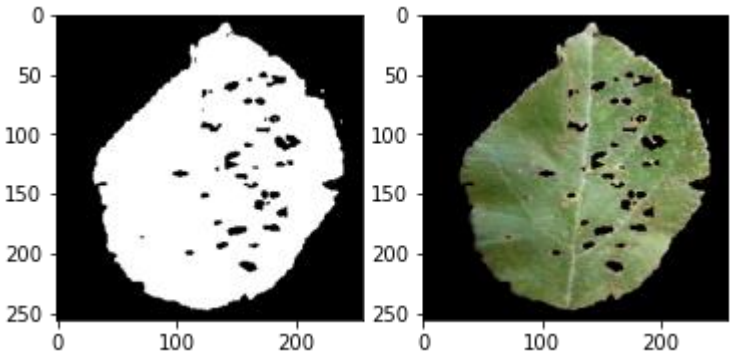


Figure 4-5: Green Color Extraction from Diseased Leaf

In Figure 4-5, the left image is the result of applying a green color mask, isolating the healthy areas of the leaf, which appear white, and leaving the diseased spots in black. The right image shows the original leaf for comparison, where the healthy green areas correspond to the white regions in the segmented image. This process emphasizes the contrast between healthy tissue and areas potentially affected by disease.

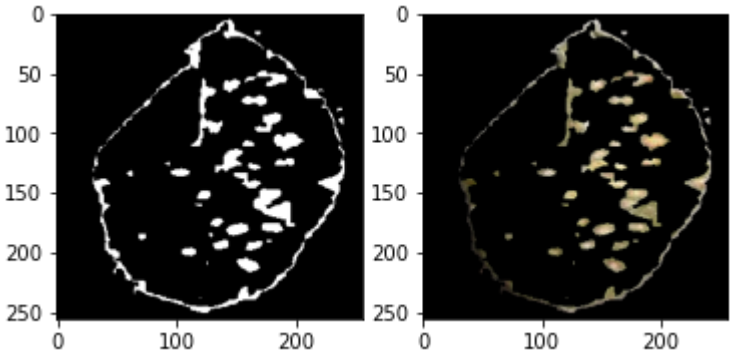


Figure 4-6: Brown Color Extraction from Diseased Image

In Figure 4-6, the left image illustrates the outcome of applying a brown color mask to isolate diseased areas, which are represented by the white spots against the black background of the healthy leaf tissue. The right image shows the combined effect of the green and brown masks, with diseased spots in brown and healthy tissue in black, enhancing the visual differentiation between the affected and unaffected areas. This segmentation is critical for the precise identification and quantification of disease symptoms on the leaf surface.

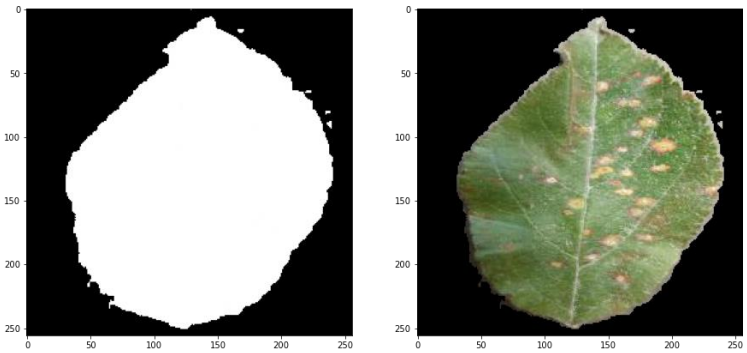


Figure 4-7: Combined Segmentation of Healthy and Diseased Leaf Tissues

In Figure 4-7, the left image depicts the result of a dual-segmentation process where masks for both green (healthy) and brown (diseased) tissues have been applied, resulting in the white areas of the final mask. This segmentation accentuates all the relevant areas, whether affected or unaffected by disease. The right image is the original leaf, showing the correlation between the segmented regions and their actual condition. This combined segmentation technique is essential for providing a clear distinction between healthy and diseased tissues, vital for the accurate analysis of plant health.

4.3.2 Feature Extraction for ML Models

The process of feature extraction is crucial in transforming raw image data into a structured format that can be effectively utilized by machine learning models. This section delineates the methodologies employed in extracting key features from the pre-processed apple leaf images, which are integral to the classification process.

4.3.2.1 Hu Moments (Shape Features)

Hu Moments are utilized for capturing object shapes in images, crucial for identifying leaf diseases through variations in leaf shapes. These moments are resistant to image changes like scaling or rotation, enhancing disease detection reliability.

This study employed the **OpenCV** library to calculate Hu Moments. Leaf images were first transformed to grayscale to emphasize shape over color, using the **cv2.cvtColor** function with the **cv2.COLOR_BGR2GRAY** parameter. Spatial moments calculated via **cv2.moments** from these images laid the groundwork for deriving Hu Moments with the **cv2.HuMoments** function, producing seven invariant moments. These were compiled into a feature vector that consolidates key shape information, aiding in the differentiation of leaf diseases by their unique structural changes.

The application of Hu Moments, supported by OpenCV's efficiency and the method's mathematical robustness, enables precise shape-based feature extraction for accurate leaf disease classification.

4.3.2.2 Haralick Texture Features

Haralick texture features are key for analyzing image textures by studying pixel intensity patterns, essential for identifying textural changes in apple leaves linked to specific conditions. These features, derived from the Gray-level co-occurrence matrix (GLCM), capture textural elements like contrast and homogeneity.

In this research, the Mahotas library, known for its efficient image-processing capabilities in Python, was used to extract these features. The process began with converting leaf images to grayscale using OpenCV's **cv2.cvtColor** function with **cv2.COLOR_BGR2GRAY**, focusing the analysis on texture rather than color.

The **mahotas.features.haralick** function was then employed to calculate Haralick features from these grayscale images, generating a comprehensive feature vector from the GLCM that reflects the leaf's texture.

This texture-based analysis, powered by the Mahotas and OpenCV libraries, is instrumental in distinguishing between different leaf diseases, leveraging unique textural patterns for accurate disease classification.

4.3.2.3 Color Histogram

color histograms are crucial for apple leaf disease detection, as they capture key color characteristics and variations, such as discoloration, related to disease symptoms.

The process begins with converting images to HSV color space using OpenCV's **cv2.cvtColor**, optimizing color analysis by separating hue from light intensity. This ensures consistent color representation across different lighting.

The **cv2.calcHist** function then generates color histograms by counting pixels within specified bins for each color channel, offering a comprehensive view of color distribution. Normalization of these histograms via **cv2.normalize** ensures consistency in scale across images. The histograms are subsequently flattened into feature vectors, concisely representing color data for machine learning models.

This approach, leveraging OpenCV's capabilities, effectively distinguishes between healthy and diseased leaves by quantifying color features, thereby enhancing disease classification through advanced image processing techniques.

4.3.2.4 Concatenation of Feature Vectors and Creation of Global Features

The final step in feature extraction combines individual feature vectors from Hu Moments, Haralick Texture, and color Histograms into a comprehensive global feature vector for each image, crucial for a detailed analysis by machine learning models.

This integration is achieved using **NumPy**, a core library for scientific computing in Python, known for its array manipulation capabilities. The **np.hstack** function is used to merge the

feature vectors horizontally, ensuring the global vector represents all critical image attributes: color, texture, and shape.

The global feature vectors, labelled according to the leaf's health status, are compiled into lists, forming the dataset for training and evaluating machine learning models.

This method highlights the efficiency of combining NumPy's data handling with OpenCV's feature extraction, demonstrating the effectiveness of Python's libraries in complex image analysis and enhancing the models' ability to discern patterns and anomalies in the data.

4.3.3 Transformation and Storage of Feature Vectors

The process of transforming and storing feature vectors is a critical phase in the preparation of extracted features for machine learning analysis. This phase is composed of vital steps including Label Encoding, which converts categorical labels into a numerical format suitable for algorithmic processing; Feature Scaling, which normalizes the feature vectors to ensure equitable contribution towards the model's forecasts; and feature vectors storage, which involves organizing the processed data in an accessible and efficient format for model training and evaluation. Together, these procedures ensure that the dataset is optimally structured and ready for the application of machine learning techniques.

4.3.3.1 Label Encoding

Label encoding is a crucial preprocessing step in machine learning, converting categorical labels into numeric form. This study focuses on encoding the health status of apple leaves into binary format: **Diseased** as **1** and **Healthy** as **0**, simplifying classification for machine learning models.

Utilizing **LabelEncoder** from the **scikit-learn** library, unique labels are extracted using **np.unique(labels)**, then **fit_transform** is applied to convert textual labels into numerical representations. This transformation enables effective interpretation and learning from the dataset's labels by machine learning models.

4.3.3.2 Feature Scaling

Feature scaling is an indispensable step in data preprocessing for machine learning, ensuring that each feature vector contributes uniformly to the algorithm's decision-making process. This normalization adjusts the feature vectors so their values fall within a designated range, commonly set between 0 and 1, which is essential for maintaining parity among the features. Such scaling prevents any single feature from dominating the model due to its numerical scale, thereby allowing a fair contribution from all features.

For feature scaling, the **MinMaxScaler** from the **scikit-learn** library, a renowned tool in the machine learning community for its comprehensive preprocessing capabilities is utilized. The scaler is configured with the **feature_range** parameter set to **(0, 1)** to normalize the feature vectors within the specified range. The global feature set undergoes scaling through the application of MinMaxScaler's **fit_transform** method, effectively rescaling the features to the desired range. This method not only computes the minimum and maximum values needed for scaling but also transforms the data based on these computed values, thereby streamlining the scaling process.

4.3.3.3 Data Storage

The preprocessing phase concludes with the organized storage of feature vectors and labels using the HDF5 file format, facilitated by the h5py library. HDF5 is chosen for its capacity to handle large data volumes efficiently, crucial for image-based datasets.

The process involves creating two HDF5 files—one for feature vectors and one for labels—using the **h5py.File** method in write mode. Datasets named **dataset_1** within these files store the respective data, utilizing the **create_dataset** function for its ability to efficiently manage large structured data arrays.

HDF5's support for extensive, complex data and fast access speeds makes it ideal for the demands of feature extraction outputs, ensuring quick and efficient model training and evaluation. The secure closure of files post-storage guarantees data integrity.

Employing HDF5 ensures reliable data preservation and accessibility, streamlining the machine learning models' operations for precise apple leaf disease classification.

4.3.4 Initialization of Machine Learning Models

This section introduces the initialization of seven distinct machine learning models, each tailored for effective apple leaf disease classification.

4.3.4.1 Logistic Regression Model Initialization

Logistic Regression is a fundamental statistical method used for binary classification tasks, making it an apt choice for distinguishing between **Diseased** and **Healthy** apple leaves in this study. The **LogisticRegression** class from the Python-based scikit-learn machine learning library is utilized to initialize the model.

The initialization code **models.append(('LR', LogisticRegression(random_state=seed)))** integrates the model within a suite of classifiers slated for assessment. The **random_state** parameter, configured with a preset seed of 9, guarantees consistent outcomes by managing the inherent stochastic nature of the logistic regression's optimization process.

This methodical approach to initializing the Logistic Regression model, with its emphasis on reproducibility and its relevance to the binary classification task at hand, lays a foundational step in the comparative analysis of machine learning algorithms for the effective diagnosis of leaf diseases.

4.3.4.2 Linear Discriminant Analysis (LDA) Model Initialization

The Linear Discriminant Analysis (LDA) model is set up in the machine learning workflow through the **LinearDiscriminantAnalysis** class, a component of the scikit-learn library. This class is highly valued in statistical machine learning for its effectiveness in reducing dimensionality and performing classification tasks. LDA is particularly beneficial for projects aiming to find a linear combination of features that best separate two or more classes.

The code **models.append(('LDA', LinearDiscriminantAnalysis()))** effectively adds the LDA model to a list of models, setting the stage for further training and evaluation. The absence of explicit function parameters in the initialization suggests the use of default settings for the LDA model, which are often well-tuned for a wide range of classification tasks. This choice underscores the model's adaptability and the intention to leverage its inherent capabilities without initial customization.

By incorporating the LDA model into the analysis framework, the approach harnesses its statistical prowess to discern patterns and distinctions within the feature vectors, contributing significantly to the robust classification of leaf health conditions.

4.3.4.3 K-Neighbors Classifier Model Initialization

The K-Neighbors Classifier is initialized using the **KNeighborsClassifier** function from the **scikit-learn** library, a widely recognized toolkit for machine learning in Python. This classifier implements the k-nearest neighbors algorithm, which classifies instances based on the majority vote of their nearest neighbors in the feature space.

The code **models.append(('KNN', KNeighborsClassifier()))**, the classifier is added to a list of models without specifying any parameters, implying the use of default values. The default number of neighbors, **n_neighbors**, is set to **5**, and the default metric for calculating distances between instances is **minkowski** with **p=2**, equivalent to the standard Euclidean distance.

This approach allows for a straightforward implementation of the k-nearest neighbors algorithm, providing a baseline for comparing its performance against other classifiers in the study. By utilizing the scikit-learn library's implementation, the research leverages a robust and efficient solution for incorporating k-nearest neighbors into the broader machine learning workflow for disease classification in apple leaves.

4.3.4.4 Classification and Regression Trees / Decision Tree Classifier Model Initialization

The initialization of the Classification and Regression Trees (CART) also known as the Decision Tree Classifier model is a critical step in the setup of machine learning algorithms for the classification of apple leaf diseases. The code snippet **models.append(('CART', DecisionTreeClassifier(random_state=seed)))** demonstrates the instantiation of a CART model using the **DecisionTreeClassifier** from the **scikit-learn** library, a widely recognized toolkit in the machine learning community for its comprehensive suite of algorithms and pre-processing functions.

The **DecisionTreeClassifier** is initialized with the **random_state** parameter fixed at 9, ensuring predictable and consistent outcomes. This parameter influences the model's stochastic processes, like the selection of attributes at each node division, ensuring the model's operations are uniform, yielding consistent results through successive executions.

The initialized model is then appended to a list named models, which serves as a collection of all machine learning algorithms to be evaluated. This structured approach facilitates a systematic comparison of different models' performance on the dataset, allowing for an informed selection of the most effective algorithm for disease classification in apple leaves.

By leveraging the robust functionalities of the scikit-learn library for the CART model initialization, this methodology ensures a solid foundation for the subsequent phases of model training and evaluation, contributing significantly to the precision and reliability of the disease detection process.

4.3.4.5 Random Forest Model Initialization and Parameter Tuning

The initialization of the Random Forest model, coupled with hyperparameter tuning, is a critical aspect of ensuring optimal model performance. For this study, the Random Forest algorithm was chosen due to its proven reliability and capability in managing intricate classification challenges, like identifying diseases in apple plants.

The model is initialized using the **RandomForestClassifier** from the **scikit-learn** library, a popular choice for machine learning applications due to its comprehensive collection of algorithms and preprocessing methods. The **random_state** parameter is assigned a fixed seed value of 9 to guarantee result consistency, an essential factor in research for validating and comparing findings.

Hyperparameter tuning is conducted through **GridSearchCV**, another component of the scikit-learn library, which systematically explores a range of specified parameter values to determine the most effective combination for the model. In the Random Forest configuration, we adjust **n_estimators**, the count of trees in the ensemble, and **max_features**, the maximum number of attributes evaluated for splitting a node. Values chosen for n_estimators include **[50, 100, 200]**, while for max_features, the selections entail 'auto' and 'sqrt'. This range provides a

comprehensive assessment of how different configurations impact the model's accuracy and generalizability.

The **GridSearchCV** function is configured with a cross-validation (cv) value of 3, indicating that the data will be split into three sets to validate the model's performance across different subsets of the data. This approach mitigates the risk of overfitting and ensures that the chosen hyperparameters generalize well to unseen data.

Through a methodical process of initializing models and adjusting hyperparameters, the study harnesses sophisticated machine learning methods to amplify the predictive power of the Random Forest model. This methodical process, rooted in the functionalities provided by the scikit-learn library, lays a solid foundation for the accurate and efficient classification of apple leaf diseases, contributing to the advancement of agricultural technology and plant pathology research.

4.3.4.6 Gaussian Naïve Bayes Model Initialization

The Gaussian Naive Bayes model is initialized as part of the machine learning pipeline to classify the health status of apple leaves. This model is encapsulated within the **GaussianNB** class, which employs Naive Bayes principles with an assumption that features follow a Gaussian distribution. The simplicity and efficiency of Naive Bayes, particularly in handling probabilistic classification tasks, make it a viable choice for this application.

The model is initialized without specific parameters, relying on the default settings of the GaussianNB class. This approach allows the model to estimate the necessary parameters directly from the data, a characteristic that aligns with the Naive Bayes algorithm's foundational assumptions. The initialization code **models.append(('NB', GaussianNB()))** seamlessly integrates the Gaussian Naive Bayes model into the broader ensemble of machine learning algorithms being evaluated.

By incorporating the Gaussian Naive Bayes model, the analysis benefits from its probabilistic framework, which is particularly adept at dealing with uncertainties and variabilities inherent in biological data. This model complements the ensemble, offering a unique perspective based on probabilistic reasoning, which enriches the comparative analysis of different algorithms' performance in diagnosing leaf diseases.

This strategic inclusion of the Gaussian Naive Bayes model, with its foundation in the scikit-learn library's robust implementation, underscores the comprehensive approach to evaluating a spectrum of algorithms, ensuring a well-rounded investigation into the most effective techniques for plant disease detection.

4.3.4.7 Support Vector Machine (SVM) Model Initialization

In the development of machine learning models for apple leaf disease classification, the Support Vector Machine (SVM) algorithm plays a pivotal role due to its effectiveness in handling high-dimensional data and its robustness in classification tasks. The initialization of the SVM model within the Python environment is facilitated by the **scikit-learn** library, a comprehensive tool widely recognized for its extensive suite of algorithms and functions tailored for machine learning applications.

The model is initialized with specific parameters to tailor its operation to the dataset's characteristics and the classification objectives. The SVM model utilizes the **SVC** class from scikit-learn's svm module, configuring the probability parameter to **True** for enabling probability estimates. This setting enables the estimation of class probabilities, a feature that is particularly useful for evaluating the confidence of the model's forecasts. Setting the **random_state** parameter to a fixed value of 9 guarantees consistent model outcomes by managing the inherent randomness during the algorithm's training phase.

This careful initialization of the SVM model, with explicitly defined parameters, underscores the meticulous approach adopted to harness the algorithm's capabilities fully. By setting the groundwork in this manner, the study ensures that the SVM model is optimally configured for the subsequent phases of training and evaluation, contributing significantly to the reliable and accurate classification of leaf health conditions.

4.3.5 Data Retrieval for Machine Learning

The retrieval of processed data from storage is crucial for the integrity of the machine learning process. This step utilizes the **h5py** library to access feature vectors and labels stored in **HDF5** files, a format chosen for its capacity to handle complex data volumes efficiently. The **h5py.File** method opens these files in read mode, allowing for the extraction of datasets named **dataset_1**,

containing the features and labels. These datasets are then converted to NumPy arrays, a format compatible with machine learning operations due to efficiency and flexibility. After extraction, the files are promptly closed to preserve data integrity and optimize resource usage, ensuring the data remains consistent and accessible for model training and evaluation.

4.3.6 Splitting Data into Testing and Training Sets

A crucial phase in dataset preparation for machine learning models involves segregating the data into separate training and validation sets. This segmentation is fundamental to evaluating the performance of the models in a manner that is unbiased and reflective of their ability to generalize to unseen data. For this study, the data is divided, allocating 70% for model training and 30% for evaluation, ensuring a balanced approach to model development and assessment.

The division of the dataset into training and testing segments is executed using the **train_test_split** utility from the scikit-learn package, known for its efficacy in handling data splits. This function, when called, takes as input the arrays for features and labels, alongside parameters specifying the test set's size (**test_size=0.30**), a **random_state** for split consistency, and **stratify** to preserve label proportions. This ensures that both the training and testing sets are reflective of the entire dataset's structure, an essential aspect when dealing with potential class imbalances.

Through careful separation of the dataset into training and testing portions, this method sets up a structured environment for model training on a significant data share, while reserving a distinct segment for impartial assessment of their predictive capabilities. This careful preparation of the dataset is instrumental in the development of machine learning models that are both accurate and capable of effectively generalizing beyond the data on which they were trained.

4.3.7 Model Evaluation

The evaluation of machine learning models involves a rigorous process to assess their predictive accuracy and overall performance. This process is essential to determine the models' efficacy in classifying the health status of apple leaves based on the extracted features.

Each model within the predefined ensemble, including Logistic Regression, Decision Trees, LDA, K-Nearest Neighbors, Random Forest, Naive Bayes, and Support Vector Machines, is subjected to evaluation. The evaluation leverages the **StratifiedKFold** method from the **scikit-learn** library, ensuring a stratified sampling of data that maintains the original distribution of labels across each fold. This method is initialized with **10 splits**, shuffling enabled for randomness, and a consistent random state for reproducibility.

Cross-validation scores are obtained using the **cross_val_score** function, assessing the accuracy of each model across the stratified folds. This cross-validation approach provides a robust estimate of the model's performance on unseen data by averaging the accuracy scores obtained from each fold.

Following cross-validation, models are trained on the entire training dataset and then used to make predictions on both the training and testing sets. Predictions on the training set help assess the model's ability to learn from the data, while predictions on the testing set evaluate its generalization to new, unseen data.

4.3.8 Performance Metrics for ML Models

Several metrics are computed to gauge the models' performance comprehensively:

- **Accuracy**: The proportion of correctly predicted instances in the total predictions, calculated for both training and testing sets.
- **F1 Score**: The harmonic mean of precision and recall, providing a balance between the two metrics, especially in imbalanced datasets.
- **Precision and Recall**: Precision measures the model's accuracy in predicting positive instances, while recall assesses the model's ability to identify all actual positive instances.
- **ROC-AUC**: The area under the receiver operating characteristic curve, indicates the model's ability to discriminate between classes.
- **Precision-Recall Curve**: A plot that illustrates the trade-off between precision and recall for different probability thresholds.

These metrics are stored for each model, facilitating a comparative analysis to identify the model that performs best in classifying apple leaf diseases.

The implementation utilizes the **scikit-learn** library, renowned for its comprehensive suite of tools for machine learning. Functions such as **accuracy_score**, **f1_score**, **precision_score**, **recall_score**, **roc_curve**, **auc**, **precision_recall_curve**, and **average_precision_score** are employed to compute the various performance metrics. The use of these functions ensures a standardized approach to evaluating model performance, adhering to the best practices in machine learning research.

By meticulously evaluating each model against a set of well-established metrics, this phase ensures that the most effective model is selected for the task of apple leaf disease classification, grounding the research in a robust methodological framework.

4.3.9 Compilation of Model Performance Metrics

The aggregation of model performance metrics into a structured format is a critical step in the evaluation phase of machine learning workflows. This process involves the creation of a DataFrame, a tabular data structure provided by the Pandas library, which is renowned for its powerful data manipulation capabilities in Python. The DataFrame facilitates a clear and concise presentation of the training and testing accuracies for each machine learning model employed in the study.

In this instance, the DataFrame constructor from Pandas is utilized to compile the accuracies into a single table. The constructor is invoked with a dictionary argument, where the keys represent the column labels ('Model', 'Train Accuracy', 'Test Accuracy') and the corresponding values are the lists containing model names, training accuracies, and test accuracies, respectively. This structured approach not only enhances the readability of the results but also provides an efficient means of comparing the performance across different models.

By systematically collating the accuracies into a DataFrame, the research ensures that the evaluation metrics are readily accessible and interpretable, aiding in the transparent assessment of the models' capabilities in classifying apple leaf diseases.

This methodical compilation of model accuracies underscores the importance of data organization in the analytical process, facilitating a comprehensive overview of model performances and informing subsequent decisions in the model selection process.

4.3.10 Visualizing Model Performance

The visualization of model performance through bar plots is a crucial step in evaluating and presenting the accuracy of different machine learning models. This process involves using the Matplotlib and Seaborn libraries, renowned for their extensive capabilities in data visualization within the Python ecosystem.

4.3.11 Training Accuracy Visualization

For the training accuracies, a bar plot is constructed using Seaborn's barplot function, which provides an intuitive representation of the accuracy scores across various models. The plot is set against a white grid background (**style="whitegrid"**) with a **"pastel"** color palette for clarity and aesthetic appeal. Each bar represents a machine learning model, labelled on the x-axis, with the height of the bar corresponding to the training accuracy depicted on the y-axis.

The plot's title, 'Train Accuracies for ML Models,' along with the x and y-axis labels, are set using Matplotlib's title, xlabel, and ylabel functions, respectively. To enhance readability, the x-axis labels are rotated 45 degrees. Additionally, numerical values representing the accuracy scores are annotated above each bar, providing precise information at a glance.

4.3.12 Test Accuracy Visualization

The test accuracies are visualized in a separate bar plot, following the same design principles and functions as used for the training accuracies. This plot offers a direct comparison of how each model performs on unseen data, which is paramount for assessing the models' generalization capabilities.

The title 'Test Accuracies for ML Models' succinctly indicates the plot's purpose, with the x-axis representing the different machine learning models and the y-axis showing the corresponding test accuracy scores. As with the training accuracies, numerical values are annotated above each bar to denote the exact accuracy figures.

These visualizations, facilitated by the figure, barplot, and text annotation functions from Matplotlib and Seaborn, not only aid in the quantitative assessment of model performance but also enhance the interpretability of the results, allowing for informed decisions in the model selection process. Through the effective use of these libraries, the research encapsulates the

performance metrics in a visually engaging and informative manner, adhering to the standards of academic rigor and clarity.

4.3.13 Visualization of ROC Curves

The Receiver Operating Characteristic (ROC) curve is a fundamental tool for evaluating the performance of classification models at various threshold settings. The ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR), providing insight into the trade-off between sensitivity and specificity.

This visualization is accomplished using the Matplotlib library, a widely used Python library for creating static, interactive, and animated visualizations. The plt.figure function initializes a new figure with a specified size, here set to 12x7 inches, to ensure the plot is sufficiently large for clear observation of the curves.

Within a loop, each model's name and its corresponding ROC curve data, comprising the FPR and TPR, are plotted using the plt.plot function. The line width is set to 2 for better visibility, and each curve is labeled with the model's name and its Area Under the Curve (AUC) value, which quantifies the overall performance of the model in distinguishing between the classes. The AUC values are formatted to two decimal places for precision.

A dashed line representing a 'no-skill' classifier is added to the plot for reference, using plt.plot([0, 1], [0, 1], 'k--', lw=2), where 'k--' denotes a black dashed line. This line serves as a baseline, indicating the performance of a model that makes random predictions.

The axes are labeled 'False Positive Rate' and 'True Positive Rate', and the title 'ROC Curves' succinctly describes the content of the plot. A legend is positioned in the lower right corner to assist in identifying each curve, and a grid is enabled for easier interpretation of the plot.

By systematically plotting the ROC curves for each model, this section effectively illustrates the diagnostic ability of the models, enabling a comparative analysis of their performance in classifying the health status of apple leaves.

4.3.14 Visualization of Precision-Recall Curves

The Precision-Recall curves provide a graphical representation of a model's performance across different thresholds, balancing the precision (positive predictive value) and recall (sensitivity). This visualization is particularly insightful for evaluating models in the context of imbalanced datasets, where the positive class (diseased leaves) is of specific interest.

The plotting of Precision-Recall curves is facilitated by the Matplotlib library, a comprehensive tool for creating static, interactive, and animated visualizations in Python. The figure function initializes a new figure with specified dimensions, in this case, 12x7 inches, providing a canvas for the plot. Within a loop, the plot function is called for each model, drawing a curve that depicts the trade-off between precision and recall at various threshold levels. The curves are labeled with the respective model names for clarity.

Parameters such as xlabel and ylabel set the labels for the x-axis and y-axis respectively, ensuring the axes are appropriately titled to reflect the data being plotted. The title function adds a descriptive title to the plot, "Precision-Recall Curves", summarizing the content of the visualization. The legend function is used to include a legend on the plot, indicating which curve corresponds to which model, with its location specified as "lower left" to avoid obscuring the data. The grid function enhances readability by adding a grid to the background of the plot.

This method of visualization not only aids in the comparative assessment of different models' performance but also in understanding how well each model identifies the positive class across various levels of recall, an essential aspect in the domain of plant disease detection where false negatives can have significant implications.

By employing Matplotlib for plotting Precision-Recall curves, this approach effectively communicates the nuanced performance characteristics of each predictive model, providing valuable insights into their capabilities in distinguishing between healthy and diseased apple leaves.

4.3.15 Visualization of Confusion Matrix

A pivotal aspect of evaluating machine learning models, especially in the context of diagnosing diseases in apple leaves, is the analysis of the confusion matrix. This matrix categorizes the prediction outcomes into four distinct classes: true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). This classification is essential for a comprehensive assessment of the models' accuracy and their tendencies for specific types of misclassification.

The generation of the confusion matrix employs the **confusion_matrix** function from the scikit-learn library, which is applied to predictions obtained from the test dataset. For a more intuitive understanding of the model's performance, these results are visually depicted using a heatmap. This visualization is produced with the **heatmap** function from the **seaborn** library, effectively highlighting the model's predictive accuracy and areas for enhancement.

Beyond the confusion matrix, the assessment is enhanced with a classification report that delves into the models' precision, recall, and F1-scores, giving more nuanced understanding of their proficiency in correctly detecting diseased leaves. The accuracy_score function, also from the scikit-learn library, calculates the total accuracy, providing a holistic perspective on the models' effectiveness on the testing dataset.

Utilizing the analytical capabilities of scikit-learn and seaborn, this evaluation approach enables a comprehensive examination of the machine learning models' effectiveness and areas for improvement in identifying apple leaf diseases, guiding the choice of the optimal model for practical application.

4.4 Deep Learning Models Analysis and Design

The prevalence of diseases in apple orchards poses significant threats to agricultural productivity and sustainability. With the advent of deep learning technologies, particularly Convolutional Neural Networks (CNNs), there is a burgeoning opportunity to enhance disease detection methodologies. This research harnesses the power of NASNetMobile and MobileNetV2, two state-of-the-art deep learning models, to detect and classify diseases in apple leaves through image analysis. These models, known for their high accuracy in image classification tasks, are adapted to the domain of precision agriculture to provide timely and accurate disease diagnostics.

4.4.1 Dataset Organization and Splitting for Evaluation

The initial stage of preparing the dataset for deep learning models involves meticulously organizing the data into specific directories for training and testing. This structured approach, pivotal for image classification tasks, ensures the dataset is readily accessible and manageable during model training and evaluation phases. In this research, the variables **train_data_dir** and **test_data_dir** specify the locations of the training and testing datasets, respectively, ensuring a structured approach to data management.

In line with best practices for deep learning model training, the input images are standardized to dimensions of **224x224** pixels, and a batch size of **32** is established to optimize computational efficiency and introduce randomness during the training, enhancing the model's generalization capability.

A key step in preparing the dataset is splitting it into training and testing sets with a **70-30%** ratio, providing a fair assessment of the model's effectiveness on new data. The training set comprises 563 images for each class (Healthy and Diseased), totaling’ 1,126 images, while the testing set includes 242 images per class, summing up to 484 images. This segmentation is essential for evaluating the models' ability to generalize from learned patterns to new, unseen data.

This split is facilitated using the **train_test_split** function from the scikit-learn library, renowned for its robust data manipulation capabilities. The function is invoked with parameters that not only define the dataset's division but also ensure reproducibility and maintain a consistent distribution of labels across both sets through stratification, particularly crucial for datasets where class distribution might be imbalanced.

This meticulous organization and strategic splitting of the dataset lay a solid foundation for the subsequent phases of deep learning model training and evaluation, ensuring the models are equipped with high-quality, well-structured data to achieve optimal performance in detecting diseases in apple leaves.

4.4.2 Data Augmentation

Data augmentation is an indispensable technique in enhancing the robustness and generalizability of deep learning models, particularly in the context of image classification tasks. By introducing variability through random transformations such as rotation, translation, and flipping, it effectively simulates a more diverse set of scenarios that a model may encounter, thereby mitigating the risk of overfitting. This research employs the **ImageDataGenerator** class from TensorFlow's Keras API for data augmentation on the training set, broadening the model's exposure to diverse image variations.

For the training dataset, the augmentation strategy encompasses a series of transformations designed to expand the dataset's diversity without compromising the integrity of the image data. The **rescale** parameter normalizes pixel values to a 0-1 range, enhancing neural network efficiency. The **rotation_range** parameter introduces random rotations to the images up to 20 degrees, while **width_shift_range** and **height_shift_range** apply random horizontal and vertical shifts, respectively, up to 20% of the image dimensions. Additionally, **horizontal_flip** is utilized to randomly mirror images, under the assumption that the dataset contains no inherent left-right orientation bias. The **fill_mode** parameter is carefully chosen to address how the algorithm fills in pixels that are introduced by transformations such as rotations or shifts, ensuring that the augmented images maintain their visual coherence.

In contrast, the preparation of testing data is deliberately simplified to preserve the originality of the test images. This involves solely rescaling the pixel values, with no further augmentations applied. This approach ensures that the model's performance is evaluated on unaltered data, providing a realistic assessment of its ability to generalize to new, unseen images.

Incorporating these data augmentation techniques, this study meticulously prepares the training dataset to foster a learning environment that encourages the model to develop a more holistic understanding of the task at hand, ultimately contributing to its predictive accuracy and reliability in practical applications.

4.4.3 Data Loading and Flow

In the preparation of image datasets for training and testing within deep learning models, the **flow_from_directory** method is employed for efficient data loading and processing. This method facilitates the direct loading of images from organized directory structures, which is essential for handling voluminous datasets typical in image classification tasks.

For the training dataset, the **train_flow** object is established using the training data generator. This setup involves specifying the directory containing the training images (**train_data_dir**), defining the size to which all images will be resized (**target_size**), determining the number of images to process in a single batch (**batch_size**), and setting the classification mode to **categorical** for models dealing with multiple classes. Additionally, the **shuffle** option is activated (set to **True**), introducing an element of randomness in the selection of image batches. This random shuffling is crucial for preventing the model from learning any potential order in the training data, thereby enhancing its generalization capabilities.

Conversely, the **test_flow** object configuration for the testing dataset mirrors that of the training phase, with a significant distinction in the **shuffle** parameter, which is deactivated (set to **False**). This adjustment is made to preserve the sequence of test data, ensuring a consistent and orderly evaluation process.

This methodical approach to data loading and flow, characterized by its adaptability and efficiency, underpins the data preprocessing pipeline, setting a solid foundation for the subsequent training and evaluation of deep learning models.

4.4.4 Model Architecture and Development

In the context of developing an efficient and robust model for image classification, particularly aimed at applications such as plant disease detection, leveraging state-of-the-art convolutional neural networks (CNNs) presents a promising approach. This section delineates the architecture and development process of two CNN models based on MobileNetV2 and NASNetMobile architectures, detailing their configuration, training, and evaluation methodologies.

4.4.4.1 Model Training and Configuration

The creation of advanced models for identifying plant diseases leverages cutting-edge convolutional frameworks, particularly focusing on **MobileNetV2** and **NASNetMobile** models. These models are selected for their proven efficiency and accuracy in image classification tasks. The training process begins with importing these base models with pre-trained weights from the **ImageNet** dataset, excluding the top layers to customize the models for the specific requirements of plant disease detection. The input shape is set to accommodate images of **224x224** pixels size with three color channels to ensure consistency and compatibility with the pre-trained models' specifications.

The architecture of each model is enhanced by adding a **Global Average Pooling 2D** layer to condense the feature maps into a vector format, followed by a dense layer of **128** units with **ReLU** activation to introduce learning capability and non-linearity into the models. To combat overfitting, a dropout layer with a rate of **0.5** is employed, randomly excluding a subset of features during the training process. The architecture culminates in a dense layer with **softmax activation**, designed to output class probabilities for the two target categories, enabling categorical classification.

For the compilation and training of the models, the **Adam optimizer** is utilized with a learning rate of **0.0001,** and categorical **crossentropy** is chosen as the loss function. Throughout the training phase, key performance metrics such as **accuracy, precision, and recall** are monitored to gauge and ensure model effectiveness. This meticulous configuration and training methodology lay a robust foundation for the models, optimizing them for high performance in the classification of plant diseases based on image analysis.

4.4.4.2 Cross-Validation using Stratified K-Fold

Utilizing Stratified K-Fold Cross-Validation, the data is divided into five separate folds to enhance the model's reliability and uniformity in performance. This approach ensures an exhaustive assessment across varied data segments, reducing potential biases and ensuring stable performance indicators. The process dynamically creates training and validation sets from image names and their corresponding classes, optimizing model training through callbacks such as **EarlyStopping** and **ReduceLROnPlateau** for improved effectiveness.

4.4.4.3 Model Evaluation, Metrics, and Performance Visualization

Upon the completion of the training phase, the models are subjected to a rigorous evaluation using a distinct test dataset, aimed at determining their generalizability and efficacy on novel data. This evaluation is conducted through a set of meticulously chosen metrics:

- **Precision-Recall Curves and AUC Scores:** The models' ability to balance precision and recall, especially in datasets with class imbalances, is quantified through precision-recall curves. These curves, along with the Area Under the Curve (AUC) scores, are derived using scikit-learn's **precision_recall_curve** and **auc** functions, providing a nuanced view of the models' performance.
- **Classification Reports:** Detailed insights into the models' predictive accuracy, including precision, recall, and F1-scores for each class, are furnished through classification reports generated by scikit-learn's **classification_report** function.
- **Confusion Matrices:** A graphical representation of the models' predictive capabilities versus the actual labels is obtained through confusion matrices, constructed using scikit-learn's **confusion_matrix** function. This aids in visualizing the models' true positive and false positive rates.
- **ROC-AUC Curve:** Additionally, the Receiver Operating Characteristic (ROC) curves, plotted using Matplotlib's **plt.plot** function, further elucidate the models' discriminative power across various thresholds, with their AUC values providing a scalar representation of this capability.
- **Accuracy and Loss Over Epochs:** To augment these metrics, the **plot_metrics** function, utilizing Matplotlib, offers a graphical representation of the models' training progression, charting both accuracy and loss over epochs for the training and validation sets. These plots are instrumental in identifying learning patterns, diagnosing potential issues of overfitting or underfitting, and assessing the overall learning stability and efficiency of the models.

This comprehensive evaluative framework, leveraging the synergistic capabilities of TensorFlow, scikit-learn, and Matplotlib, ensures a thorough analysis of the models' performance, laying a robust foundation for the precise and efficient classification of health conditions in apple leaves.

4.5 Chapter Summary

Chapter 4 delves into the analytical and design facets of machine learning (ML) and deep learning (DL) models for plant disease classification, emphasizing image data processing, augmentation, and model implementation. It intricately details feature extraction methods, model architecture, and evaluation metrics, bridging theoretical concepts with practical agricultural applications. Through a meticulous examination of various ML and DL algorithms, the chapter lays a foundational understanding necessary for the effective classification of plant health conditions, setting the stage for the empirical evaluation of model performance in subsequent chapters.

Chapter 5

RESULTS AND DISCUSSION

5.1 Introduction

This chapter serves as the cornerstone of this thesis, presenting a comprehensive analysis and evaluation of the performance of seven traditional machine learning models—Logistic Regression (LR), Linear Discriminant Analysis (LDA), K-Nearest Neighbors (KNN), Classification and Regression Trees (CART), Support Vector Machine (SVM), Naive Bayes (NB), and Random Forest (RF) —alongside two advanced Convolutional Neural Network (CNN) architectures, NASNetMobile and MobileNetV2. This section explores the diagnostic capabilities of each model for plant diseases using image data, assessing their effectiveness via key metrics such as accuracy, precision, recall, and F1-scores. Alongside the traditional metrics, this chapter employs confusion matrices, ROC AUC curves, and precision-recall curves to offer a multifaceted evaluation of each model's diagnostic capabilities. By scrutinizing the models' performances, this investigation seeks to unravel the nuanced dynamics of machine learning and deep learning techniques in the context of agricultural technology, setting a foundation for future innovations in plant disease detection and management.

5.2 Performance Analysis of Machine Learning Models

In the pursuit of advancing plant disease detection, this thesis rigorously examines a suite of seven machine learning models: Logistic Regression (LR), Linear Discriminant Analysis (LDA), K-Nearest Neighbors (KNN), Classification and Regression Trees (CART), Support Vector Machine (SVM), Naive Bayes (NB), and Random Forest (RF). Performance of each model is meticulously evaluated using a range of metrics, including accuracy, precision, recall, and F1-score, alongside the application of confusion matrices, ROC AUC curves, and precision-recall curves. This multi-dimensional analysis not only quantifies each model's diagnostic accuracy but also reveals their nuanced strengths and limitations in handling imbalanced datasets, varied feature spaces, and complex data distributions.

5.2.1 Analysis of Logistic Regression Model Performance

The evaluation of the Logistic Regression (LR) model, as depicted in Figure 5-1 illustrates the confusion matrix, and Table 5-1 details the classification report, and offers insightful details into its diagnostic capabilities. The model achieves a commendable accuracy of 90.00%, with the confusion matrix indicating 218 True Positives (TP) and 214 True Negatives (TN). This result signifies a robust ability to correctly identify both diseased and healthy instances.

Strengths:

- High Precision (0.91) for Diseased class: The model is highly precise in identifying positive cases, minimizing the risk of false alarms.
- Balanced Performance: Similar precision, recall, and F1-scores for both classes indicate a well-calibrated model that treats both Diseased and Healthy classes effectively.

Limitations:

- Slight Recall Disparity: The model has a marginally lower recall for Diseased (0.89) compared to Healthy (0.91), suggesting room for improvement in identifying all positive cases.
- False Negatives and Positives: The presence of 22 False Negatives (FN) and 26 False Positives (FP) highlights potential areas for refinement, possibly by feature engineering or threshold adjustment.

This analysis underscores the LR model's potential as a reliable classifier in plant disease detection, with its balanced accuracy ensuring equitable treatment of classes. Future work could focus on reducing misclassification through advanced preprocessing techniques or ensemble methods.

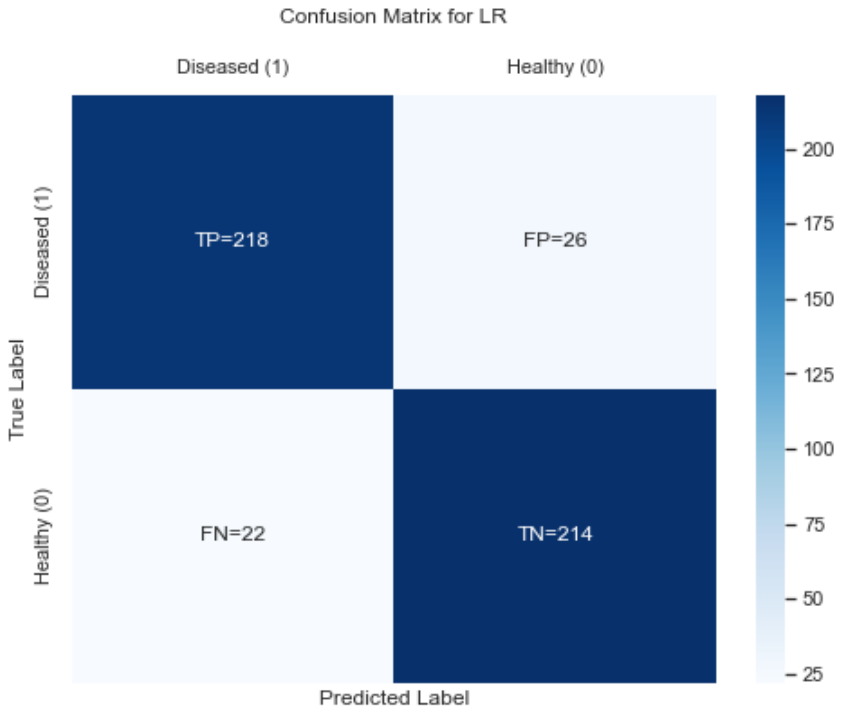


Figure 5-1: Confusion Matrix for Logistic Regression

Table 5-1: Classification Report for Logistic Regression

	precision	**recall**	**f1-score**	**support**
Diseased (1)	0.91	0.89	0.90	240
Healthy (0)	0.89	0.91	0.90	240
Accuracy			0.90	480
macro avg	0.90	0.90	0.90	480
weighted avg	0.90	0.90	0.90	480

Accuracy for LR: 90.00%

5.2.2 Analysis of Linear Discriminant Analysis (LDA) Model Performance

In evaluating the Linear Discriminant Analysis (LDA) model, Figure 5-2 illustrates the confusion matrix, and Table 5-2 details the classification report. With an accuracy of 89.79%, the model demonstrates a strong capability in distinguishing between Diseased and Healthy classes.

Strengths:

- High Precision for Diseased: The LDA model achieves a precision of 0.93 for the Diseased class, indicating a high level of reliability in its positive predictions.

- Superior Recall for Healthy: The recall of 0.93 for Healthy suggests that the model is highly effective in identifying true negative cases.

Limitations:

- Lower Recall for Diseased: The recall for the Diseased class stands at 0.86, suggesting that there is room for improvement in capturing all diseased instances.
- Discrepancies in Predictive Performance: The model exhibits 33 False Positives and 16 False Negatives, highlighting potential areas for refinement in feature selection and model tuning.

The LDA model's performance, with a balanced F1-score across both classes, underscores its utility in practical applications, while also pointing to the need for further enhancements to reduce misclassifications.

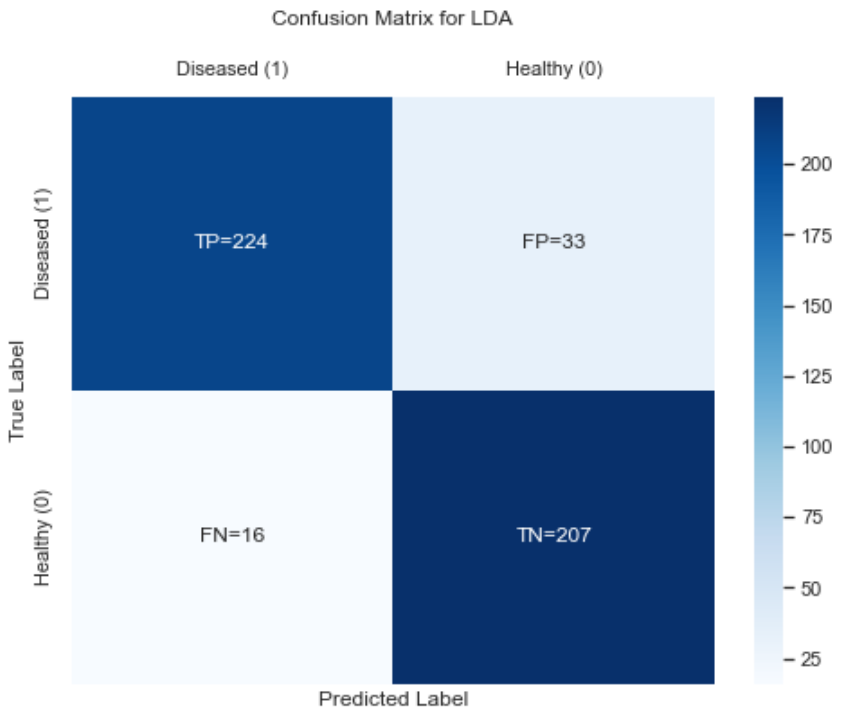


Figure 5-2: Confusion Matrix for Linear Discriminant Analysis (LDA)

Table 5-2: Classification Report for Linear Discriminant Analysis (LDA)

	precision	recall	f1-score	support
Diseased (1)	0.93	0.86	0.89	240
Healthy (0)	0.87	0.93	0.90	240
Accuracy			0.90	480
macro avg	0.90	0.90	0.90	480
weighted avg	0.90	0.90	0.90	480

Accuracy for LDA: 89.79%

5.2.3 Analysis of K-Nearest Neighbors (KNN) Model Performance

The K-Nearest Neighbors (KNN) algorithm, a model grounded in feature similarity, is analyzed through Figure 5-3, which displays the confusion matrix, and Table 5-3, which details the classification report. This model achieves an impressive accuracy of 91.46%, indicating a high degree of precision in classifying instances into Diseased and Healthy categories.

Strengths:

- The precision of 0.92 for the Diseased class coupled with a recall of 0.90 underscores the model's efficacy in correctly identifying diseased instances while maintaining a low rate of false positives.
- The model's recall of 0.93 for the Healthy class demonstrates its strength in accurately recognizing healthy cases, which is crucial for reducing unnecessary treatments or interventions.

Limitations:

- While the model shows a high level of precision, the existence of 18 False Negatives (FN) indicates that some diseased instances are being overlooked, which could be critical in a real-world diagnostic setting.
- The 23 False Positives (FP) suggest that there is still room for improvement in the model's ability to discern non-diseased instances, potentially through parameter tuning or by exploring alternative distance metrics.

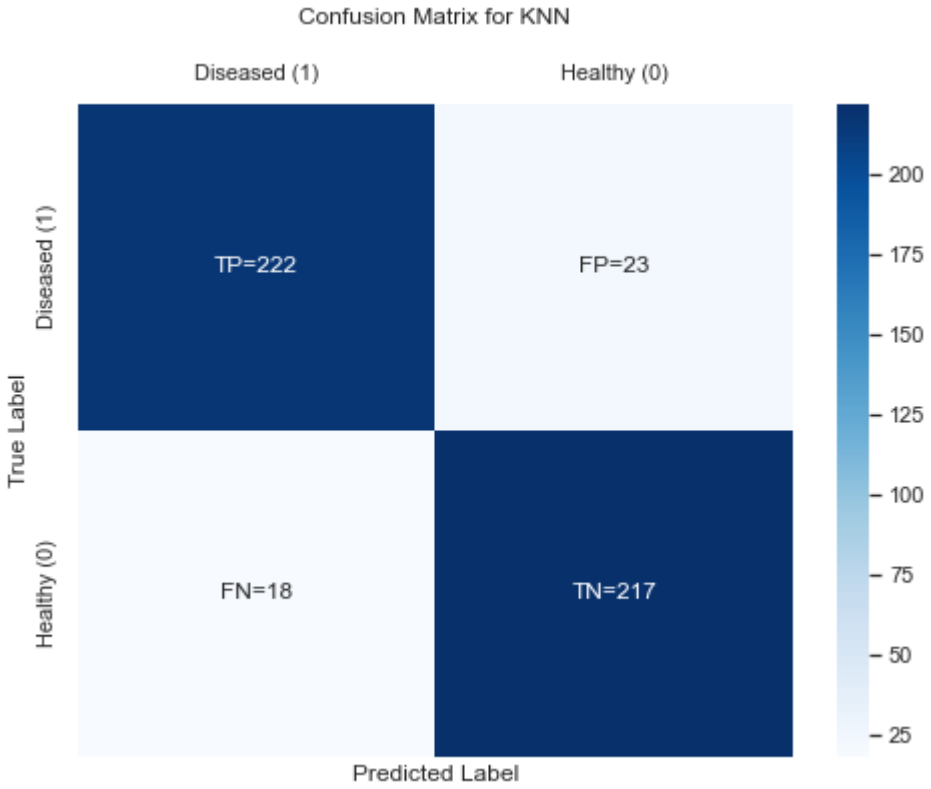


Figure 5-3: Confusion Matrix for K-Nearest Neighbors

Table 5-3: Classification Report for K-Nearest Neighbors

	precision	**recall**	**f1-score**	**support**
Diseased (1)	0.92	0.90	0.91	240
Healthy (0)	0.91	0.93	0.92	240
Accuracy			0.91	480
macro avg	0.91	0.91	0.91	480
weighted avg	0.91	0.91	0.91	480

Accuracy for KNN: 91.46%

5.2.4 Analysis of Classification and Regression Trees (CART) Model Performance

The Classification and Regression Trees (CART) model is assessed through the lens of Figure 5-4, showcasing the confusion matrix, and Table 5-4, detailing the classification report. With an overall accuracy of 90.62%, the model presents a promising approach for the classification of plant diseases.

Strengths:

- Consistent Performance: The CART model demonstrates a balanced classification capability with precision and recall metrics closely aligned for both Diseased (precision: 0.90, recall: 0.92) and Healthy (precision: 0.91, recall: 0.90) classes.
- High True Positive Rate: The model accurately identifies a significant number of diseased instances (TP=215), suggesting its effectiveness in recognizing the presence of disease.

Limitations:

- Misclassification Errors: There are 20 False Positives and 25 False Negatives, indicating occasional misclassification that could be addressed by fine-tuning the model's complexity to better capture the decision boundaries.
- Recall and Precision Trade-Off: While the differences are minor, the model slightly favors the recall for the Diseased class over the Healthy class, which may need to be considered when optimizing for specific use cases.

This balanced accuracy profile of the CART model makes it a viable candidate for real-world applications, albeit with some scope for optimization to minimize errors that could have serious implications in practical scenarios.

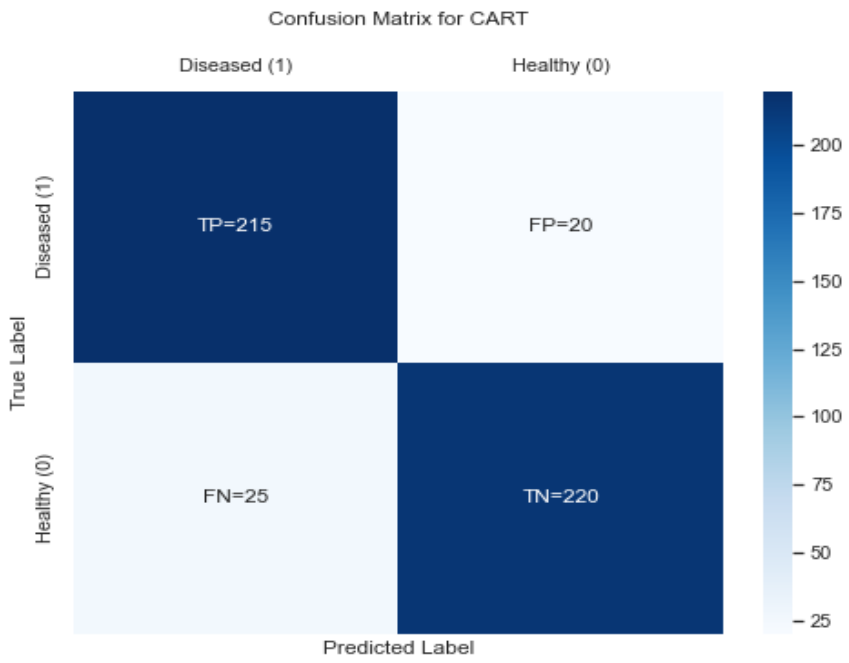


Figure 5-4: Confusion Matrix for CART

Table 5-4: Classification Report for CART

	precision	recall	f1-score	support
Diseased (1)	0.90	0.92	0.91	240
Healthy (0)	0.91	0.90	0.91	240
accuracy			0.91	480
macro avg	0.91	0.91	0.91	480
weighted avg	0.91	0.91	0.91	480

Accuracy for CART: 90.62%

5.2.5 Analysis of Random Forest Model Performance

The Random Forest (RF) model's proficiency is illustrated in Figure 5-5 through the confusion matrix, with Table 5-5 providing the classification report details. This ensemble method achieved an exceptional accuracy of 95.21%, reflecting its high effectiveness in plant disease classification.

Strengths:

- Exceptional Precision and Recall: The RF model exhibits excellent precision and recall for both Diseased (precision: 0.95, recall: 0.96) and Healthy (precision: 0.96, recall: 0.95) classes, indicative of its ability to accurately classify and minimize misdiagnoses.
- Low Misclassification: A notably low number of False Positives (FP=10) and False Negatives (FN=13) signifies the model's capability to discern between classes with high reliability.

Limitations:

- Model Complexity: As an ensemble method, RF models may become complex and may require considerable computational resources for training and prediction, potentially limiting their deployment in resource-constrained environments.
- Interpretability: The 'black box' nature of RF can be a drawback when interpretability is necessary, as it is more challenging to understand the decision-making process compared to simpler models like LR or CART.

The RF model stands out for its high accuracy and balanced metrics, indicating its suitability for deployment in scenarios where precision is paramount. However, considerations regarding computational efficiency and interpretability must be taken into account.

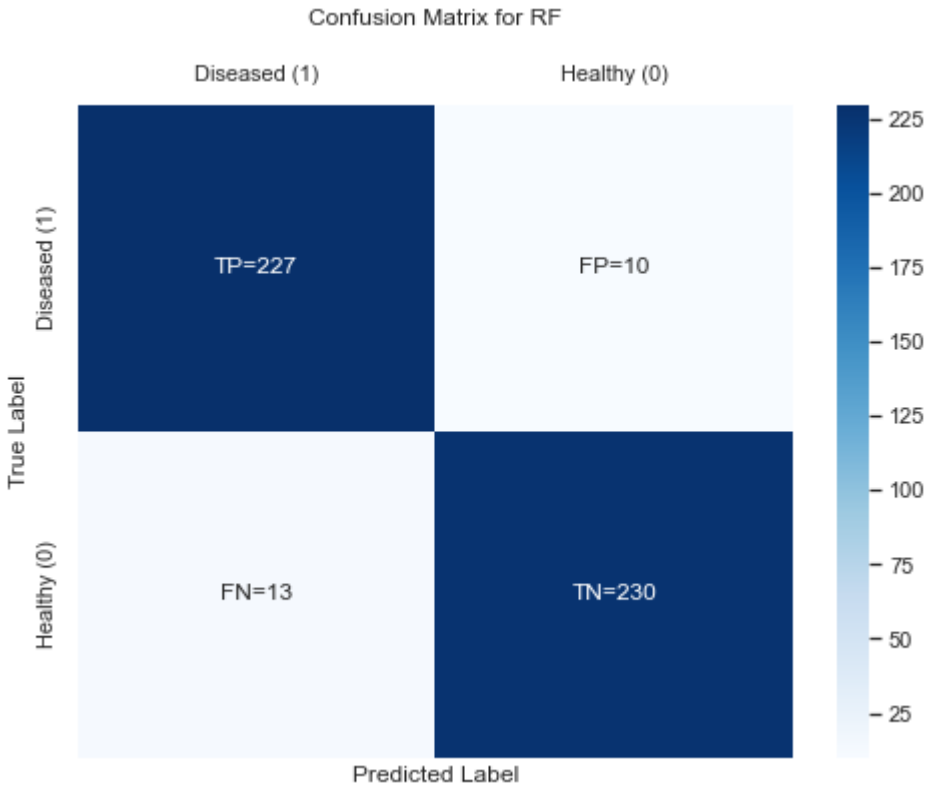


Figure 5-5: Confusion Matrix for Random Forest

Table 5-5: Classification Report for Random Forest

	precision	**recall**	**f1-score**	**support**
Diseased (1)	0.95	0.96	0.95	240
Healthy (0)	0.96	0.95	0.95	240
accuracy			0.95	480
macro avg	0.95	0.95	0.95	480
weighted avg	0.95	0.95	0.95	480

Accuracy for RF: 95.21%

5.2.6 Analysis of Naïve Bayes Model Performance

The Naive Bayes (NB) model's performance is critically appraised using Figure 5-6, which portrays the confusion matrix, and Table 5-6, which conveys the classification report. The model exhibits a noteworthy accuracy of 85.83%, with a distinct strength in identifying true positives and true negatives for plant disease classification.

Strengths:

- High True Positive Rate: The NB model correctly identifies a considerable number of diseased instances (TP=234), demonstrating its effectiveness in detecting the presence of disease with high sensitivity.
- Strong True Negative Recognition: With a TN count of 178, the model shows a commendable ability to recognize healthy instances, indicative of its high specificity.

Limitations:

- Elevated False Positive Rate: The model's FP count of 62 suggests a propensity for over-diagnosing the disease, which could lead to unnecessary interventions.
- Precision and Recall Discrepancy: Despite high precision for the Diseased class (0.97), the recall rate (0.74) indicates that the model is missing a significant portion of diseased cases, which is a critical area for improvement in medical diagnostics.

The NB model, with its strong predictive power yet clear areas for improvement, underscores the importance of a careful balance between sensitivity and specificity in the diagnostic process.

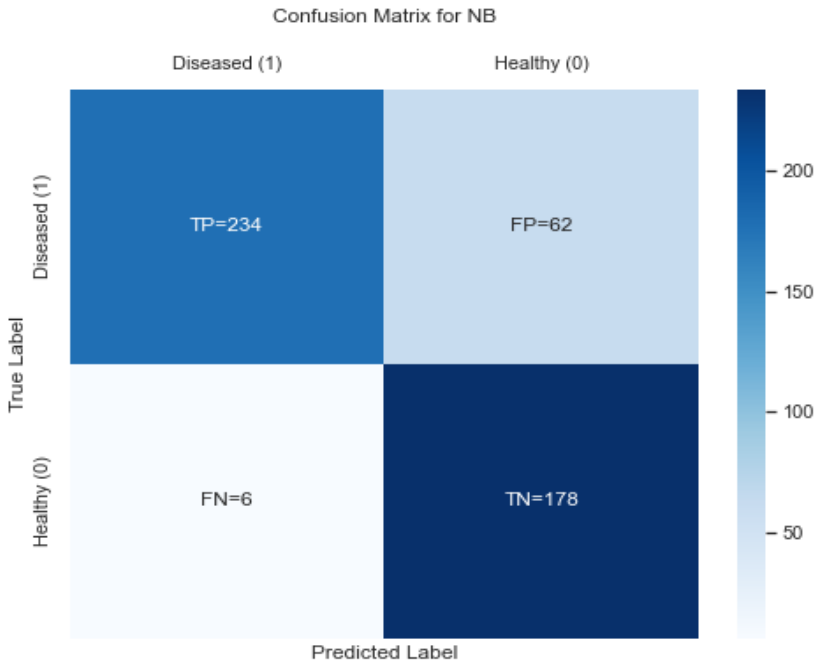


Figure 5-6: Confusion Matrix for Naive Bayes

Table 5-6: Classification Report for Naive Bayes

	precision	**recall**	**f1-score**	**support**
Diseased (1)	0.97	0.74	0.84	240
Healthy (0)	0.79	0.97	0.87	240
accuracy			0.86	480
macro avg	0.88	0.86	0.86	480
weighted avg	0.88	0.86	0.86	480

Accuracy for NB: 85.83%

5.2.7 Analysis of Support Vector Machine (SVM) Model Performance

The Support Vector Machine (SVM) model's analytical performance is encapsulated in Figure 5-7, which depicts the confusion matrix, and Table 5-7, which details the classification report. Demonstrating an accuracy of 90.21%, the SVM model showcases its robustness in distinguishing diseased plants from healthy ones.

Strengths:

- Balanced Classification: With precision rates of 0.91 for Diseased and 0.89 for Healthy, the SVM model achieves a harmonious balance in identifying both conditions with a high degree of accuracy.
- Effective Disease Detection: The model excels in correctly predicting diseased cases, evidenced by 220 True Positives, affirming its utility in practical disease screening applications.

Limitations:

- False Positives and Negatives: While relatively low, the presence of 27 False Positives and 20 False Negatives suggests that there is a scope for improvement, particularly in reducing the model's misclassification rate.
- Model Tuning Complexity: The SVM requires careful tuning of hyperparameters, such as the kernel choice and regularization parameter, which can be computationally intensive and may require expert knowledge to optimize.

The SVM model's capacity for high-performance classification endows it with considerable potential in automated disease detection systems, albeit necessitating meticulous optimization to perfect its predictive prowess.

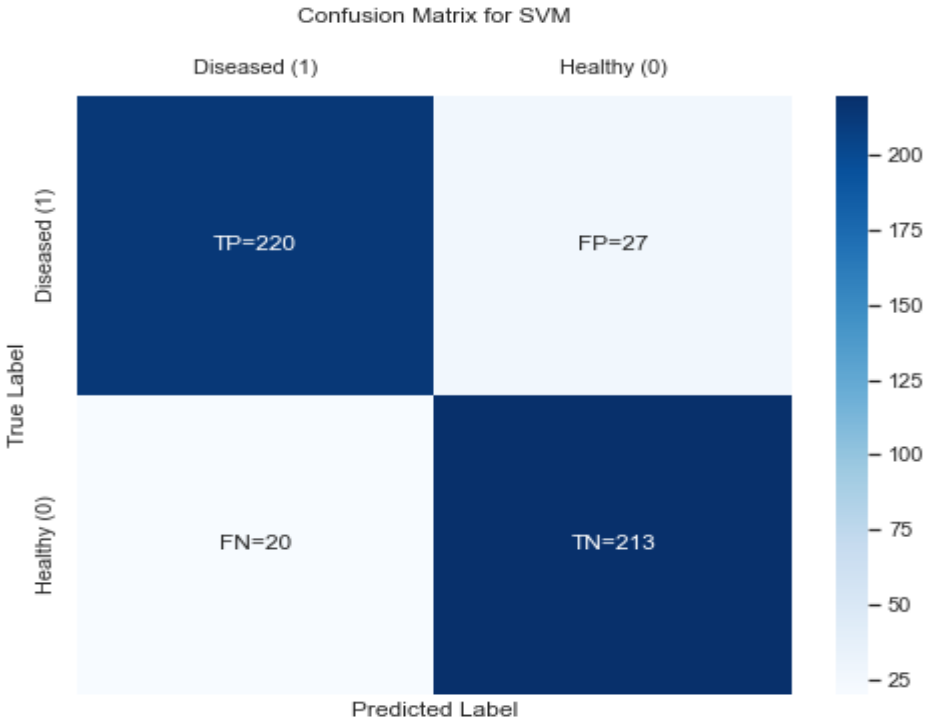


Figure 5-7: Confusion Matrix for Support Vector Machine (SVM)

Table 5-7: Classification Report for Support Vector Machine (SVM)

	precision	**recall**	**f1-score**	**support**
Diseased (1)	0.91	0.89	0.90	240
Healthy (0)	0.89	0.92	0.90	240
accuracy			0.90	480
macro avg	0.90	0.90	0.90	480
weighted avg	0.90	0.90	0.90	480

Accuracy for SVM: 90.21%

5.2.8 Comparative Performance of ML Models in Plant Disease Detection

The ensemble of machine learning models implemented for plant disease detection offers a diverse panorama of computational intelligence. The Random Forest (RF) model emerges as the apex performer with a test accuracy of 95.21%, denoted in Table 5-8 and Figure 5-9 underscoring the power of ensemble learning in capturing complex patterns within the data. A meticulous review of training and testing accuracies as depicted in (Table 5-8, Figure 5-8, and Figure 5-9) along with the ROC Curve (Figure 5-10) and Precision-Recall Curves (Figure 5-11), provides an empirical foundation for evaluating model efficacy.

5.2.8.1 Ensemble Excellence and Empirical Validation

The Random Forest (RF) model's performance showcases the epitome of ensemble learning. With a perfect training accuracy, RF demonstrates a high level of robustness and generalizability, an outcome of its ability to amalgamate multiple decision trees. This integration helps mitigate the risk of overfitting while adeptly managing the intricate data distributions within the realm of plant disease detection. The empirical validation of RF's superior model quality is vividly captured by the ROC Curve, presented in Figure 5-10, which boasts an AUC of 0.99. This exceptional AUC value not only highlights the model's accuracy but also its remarkable capability to distinguish between diseased and healthy classes with minimal false positives and maximal true positives. Such empirical evidence solidifies RF's standing as a model of exceptional caliber in predictive analytics for agriculture.

5.2.8.2 Consistency Across Models

LR, LDA, and SVM exhibit commendable consistency, with train and test accuracies hovering around the 90% mark. These models, grounded in statistical learning, show a strong ability to discern diseased from healthy plant instances, as reflected in their ROC and Precision-Recall curves clearly visualized in Figure 5-10 and Figure 5-11.

5.2.8.3 Trade-offs in Simplicity and Complexity

CART's training perfection is contrasted by a dip in testing performance (Table 5-8), indicating a need to address overfitting, which is visually apparent in the less-than-perfect ROC Curve (Figure 5-10).

5.2.8.4 Sensitivity and Specificity

The Naive Bayes (NB) model, while not reaching the top accuracy echelons, plays a pivotal role in minimizing False Negatives, a critical aspect underscored by its high precision depicted in the Precision-Recall Curve (Figure 5-11). This characteristic is invaluable, particularly in medical diagnostics, where the cost of overlooking diseased instances can be substantial. The model's balance between sensitivity and specificity is central to its diagnostic value, ensuring that cases of disease are reliably identified, thus safeguarding against the potentially grave consequences of misdiagnosis.

5.2.8.5 Implications and Considerations

This comprehensive analysis not only highlights the strengths of ensemble methods and the reliability of statistical models but also points to the importance of model selection based on the specific requirements of sensitivity, specificity, and computational resources in practical applications.

The best-performing model, 'RF', with its high test accuracy as depicted in Figure 5-9, positions itself as an exemplary tool for plant disease detection, potentially revolutionizing the field of precision agriculture.

Table 5-8: Train and Test Accuracy for ML Models

	Model	Train Accuracy	Test Accuracy
0	LR	0.936607	0.900000
1	LDA	0.954464	0.897917
2	KNN	0.954464	0.914583
3	CART	1.000000	0.906250
4	RF	1.000000	0.952083
5	NB	0.871429	0.858333
6	SVM	0.933929	0.902083

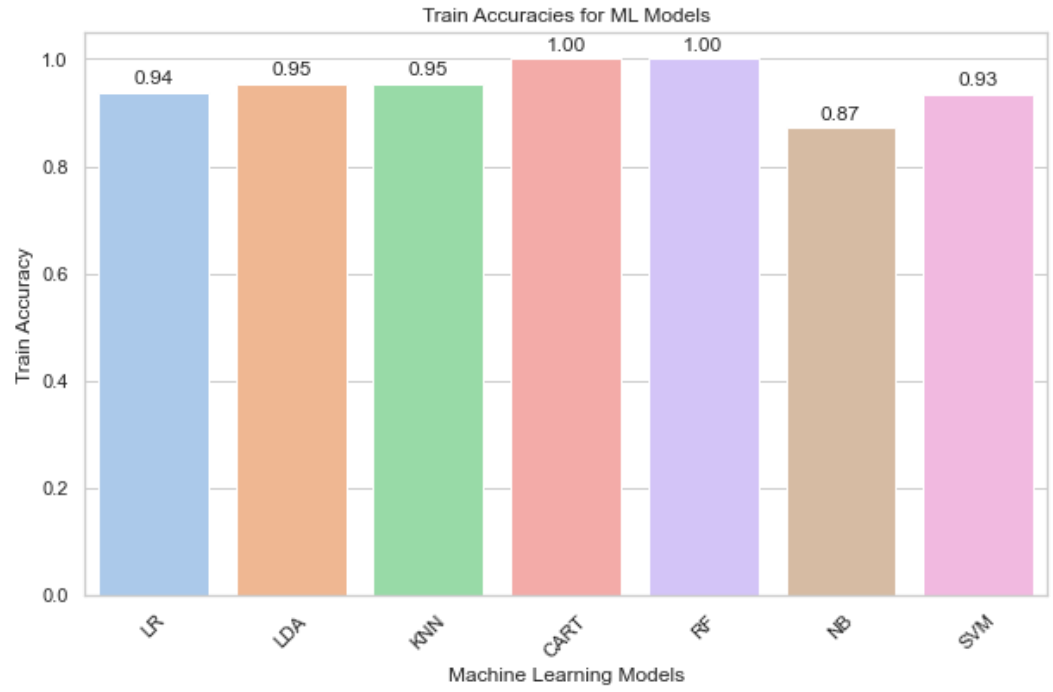


Figure 5-8: Training Accuracy for ML Models

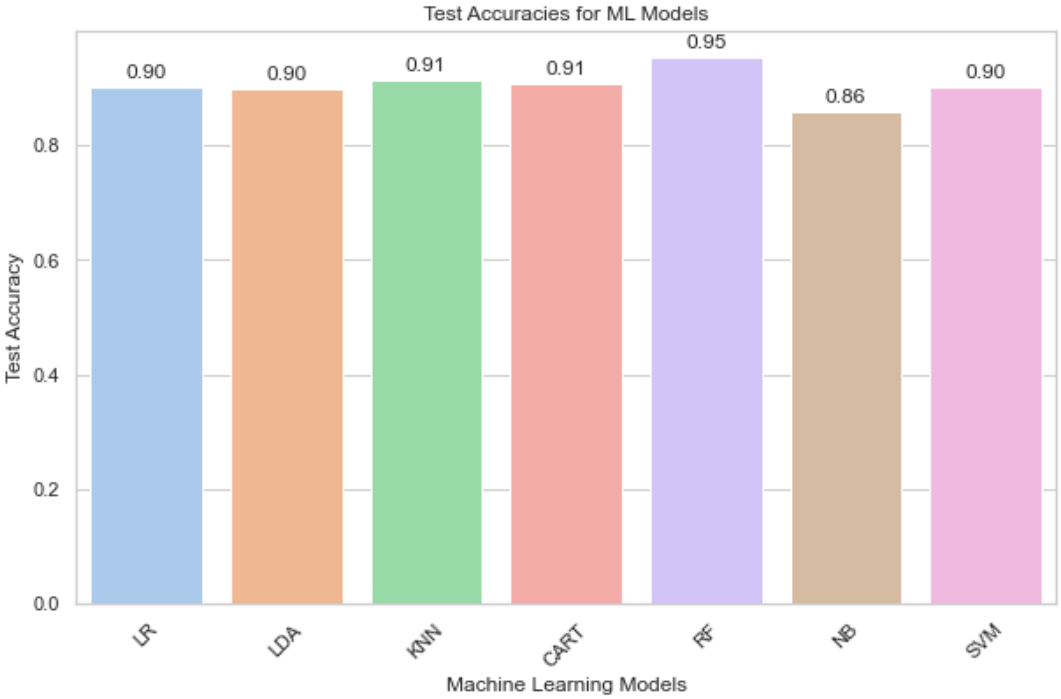


Figure 5-9: Testing Accuracy for ML Models

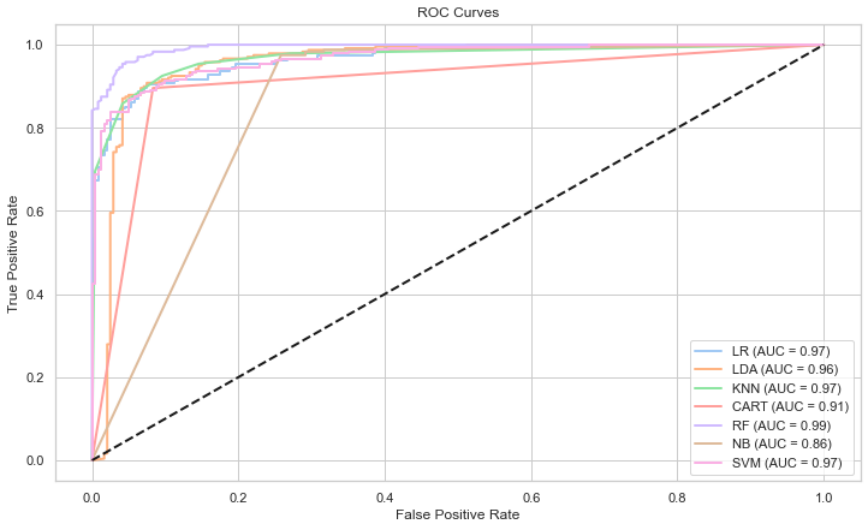


Figure 5-10: ROC Curves for ML Models

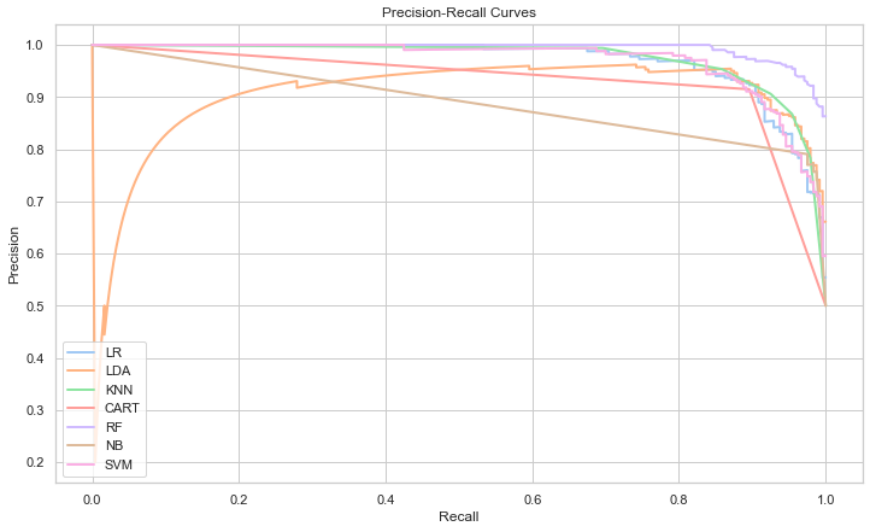


Figure 5-11: Precision-Recall Curves for ML Models

The best-performing model out of all the evaluated ML models is 'RF' with a test accuracy of 95.21%.

5.3 Performance Analysis of Deep Learning Models

In the evolving field of agricultural technology, particularly in the nuanced area of plant disease detection, the advent of deep learning has introduced unprecedented capabilities and advancements. This chapter is dedicated to the meticulous examination of two state-of-the-art deep learning architectures, NASNetMobile and MobileNetV2, which have been at the

forefront of innovation in image-based diagnostics. Leveraging their sophisticated design and computational intelligence, these Convolutional Neural Network (CNN) models offer a new dimension of analytical depth and precision in identifying and classifying plant diseases through complex image data.

The essence of this analysis is to dissect and understand the operational efficacy of these models, characterized by their ability to process and interpret high-dimensional image data with remarkable accuracy and efficiency. Through a comprehensive evaluation involving a range of performance metrics such as accuracy, precision, recall, and F1-scores, this chapter aims to illuminate the strengths and potential limitations of NASNetMobile and MobileNetV2. The analysis is further enriched by the inclusion of confusion matrices, ROC AUC curves, and precision-recall curves, providing a holistic view of each model's diagnostic performance.

By navigating through the intricate details of the performance of NASNetMobile and MobileNetV2, this chapter endeavors to contribute to the broader discourse on the application of deep learning in agriculture. The insights garnered from this analysis are intended to underscore the potential of these advanced models in enhancing disease detection methodologies, ultimately paving the way for more efficient, accurate, and sustainable agricultural practices.

5.3.1 MobileNetV2 Performance Analysis

This section undertakes a rigorous analysis of MobileNetV2's performance through an examination of its outcomes across five training folds. This segment aims to elucidate the model's precision, recall, F1-scores, and overall accuracy metrics in differentiating between healthy and diseased plant states. The section explores the model's learning process, as evidenced by trends in accuracy and loss across epochs, and assesses its ability to discriminate between classes using tools such as confusion matrices, ROC-AUC curves, and precision-recall curves. This detailed investigation into MobileNetV2's training folds is designed to provide a comprehensive understanding of the model's operational strengths and pinpoint areas where enhancements could further bolster its application in plant disease detection.

5.3.1.1 MobileNetV2 Training Fold 1

Analysis for MobileNetV2's Training Fold 1 is presented as follows:

- **Precision and Recall:** MobileNetV2 exhibits exceptional precision (0.98 for Healthy, 0.99 for Diseased) and recall (0.99 for Healthy, 0.98 for Diseased), reflecting its high accuracy in class identification as detailed in Table 5-9.
- **F1-Score and Support:** The model achieves robust f1-scores of 0.98 for both Healthy and Diseased classes, with an equal number of instances for each class, highlighting consistent classification performance as outlined in Table 5-9.
- **Overall Accuracy:** The model demonstrates an impressive overall accuracy of 98%, indicating its effectiveness in differentiating between plant health conditions as reported in Table 5-9.
- **Accuracy Over Epochs:** The accuracy graph depicts a stable increase for both training and validation accuracy, suggesting a well-calibrated model with an absence of overfitting or underfitting, as visualized in Figure 5-12.
- **Loss Over Epochs:** The loss graph illustrates a steady decline in both training and validation loss, indicative of effective learning and generalization over time, as seen in Figure 5-12.
- **Confusion Matrix:** The confusion matrix with a high number of True Positives (TP=237) and True Negatives (TN=239), and minimal False Positives (FP=5) and False Negatives (FN=3), underlines the model's classification precision as visualized in Figure 5-13.
- **ROC-AUC Curve:** Exemplified by a flawless ROC curve with an AUC score reaching the ideal mark of 1.00, Figure 5-14 illustrates the model's unparalleled ability to differentiate between classes.
- **Precision-Recall Curve:** The precision-recall graph, with an area under the curve also at 1.00, reaffirms the model's consistent precision across varying levels of recall, as shown in Figure 5-15.

Results from Execution:

Training fold 1/5
Found 900 validated image filenames belonging to 2 classes.
Found 226 validated image filenames belonging to 2 classes.

Training model: MobileNetV2

WARNING:tensorflow:From C:\Users\gsanthos\Anaconda3\lib\site-packages\keras\src\backend.py:1398: The name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

WARNING:tensorflow:From C:\Users\gsanthos\Anaconda3\lib\site-packages\keras\src\layers\normalization\batch_normalization.py:979: The name tf.nn.fused_batch_norm is deprecated. Please use tf.compat.v1.nn.fused_batch_norm instead.

Epoch 1/10
WARNING:tensorflow:From C:\Users\gsanthos\Anaconda3\lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\gsanthos\Anaconda3\lib\site-packages\keras\src\engine\base_layer_utils.py:384: The name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

28/28 [==============================] - 62s 2s/step - loss: 0.7115 - accuracy: 0.6129 - precision: 0.6129 - recall: 0.6129 - val_loss: 0.3403 - val_accuracy: 0.9286 - val_precision: 0.9286 - val_recall: 0.9286 - lr: 1.0000e-04
Epoch 2/10
28/28 [==============================] - 46s 2s/step - loss: 0.3557 - accuracy: 0.8537 - precision: 0.8537 - recall: 0.8537 - val_loss: 0.2339 - val_accuracy: 0.9330 - val_precision: 0.9330 - val_recall: 0.9330 - lr: 1.0000e-04
Epoch 3/10
28/28 [==============================] - 44s 2s/step - loss: 0.2366 - accuracy: 0.9101 - precision: 0.9101 - recall: 0.9101 - val_loss: 0.1476 - val_accuracy: 0.9688 - val_precision: 0.9688 - val_recall: 0.9688 - lr: 1.0000e-04
Epoch 4/10
28/28 [==============================] - 45s 2s/step - loss: 0.1916 - accuracy: 0.9378 - precision: 0.9378 - recall: 0.9378 - val_loss: 0.1294 - val_accuracy: 0.9643 - val_precision: 0.9643 - val_recall: 0.9643 - lr: 1.0000e-04
Epoch 5/10
28/28 [==============================] - 44s 2s/step - loss: 0.1583 - accuracy: 0.9482 - precision: 0.9482 - recall: 0.9482 - val_loss: 0.0993 - val_accuracy: 0.9866 - val_precision: 0.9866 - val_recall: 0.9866 - lr: 1.0000e-04
Epoch 6/10
28/28 [==============================] - 44s 2s/step - loss: 0.1346 - accuracy: 0.9551 - precision: 0.9551 - recall: 0.9551 - val_loss: 0.0904 - val_accuracy: 0.9955 - val_precision: 0.9955 - val_recall: 0.9955 - lr: 1.0000e-04
Epoch 7/10
28/28 [==============================] - 46s 2s/step - loss: 0.1186 - accuracy: 0.9585 - precision: 0.9585 - recall: 0.9585 - val_loss: 0.0764 - val_accuracy: 0.9821 - val_precision: 0.9821 - val_recall: 0.9821 - lr: 1.0000e-04
Epoch 8/10
28/28 [==============================] - 45s 2s/step - loss: 0.1042 - accuracy: 0.9666 - precision: 0.9666 - recall: 0.9666 - val_loss: 0.0828 - val_accuracy: 0.9732 - val_precision: 0.9732 - val_recall: 0.9732 - lr: 1.0000e-04
Epoch 9/10

28/28 [==============================] - 44s 2s/step - loss: 0.0922 - accuracy: 0.97
00 - precision: 0.9700 - recall: 0.9700 - val_loss: 0.0616 - val_accuracy: 0.9866 - val_precisio
n: 0.9866 - val_recall: 0.9866 - lr: 1.0000e-04
Epoch 10/10
28/28 [==============================] - 44s 2s/step - loss: 0.0991 - accuracy: 0.96
66 - precision: 0.9666 - recall: 0.9666 - val_loss: 0.0711 - val_accuracy: 0.9732 - val_precisio
n: 0.9732 - val_recall: 0.9732 - lr: 1.0000e-04

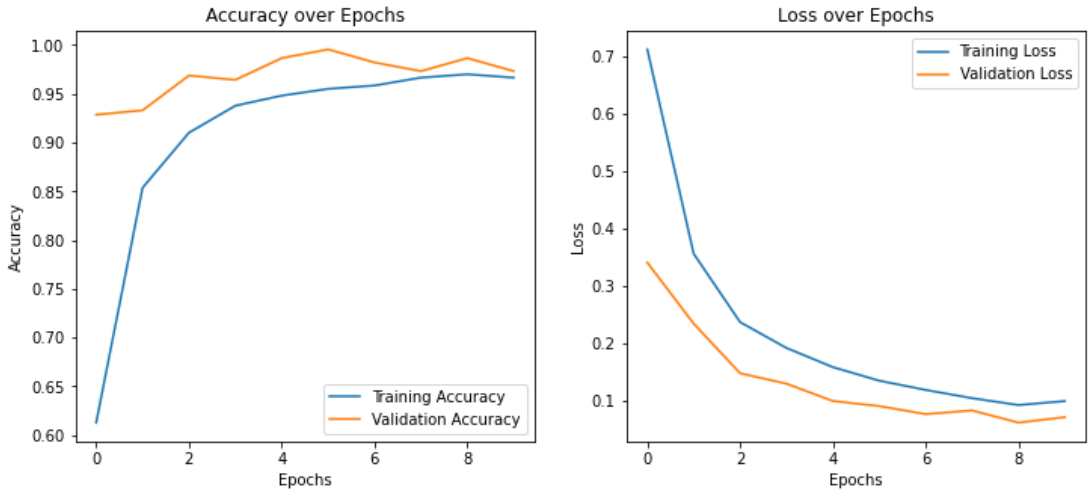


Figure 5-12: Training Fold 1 - MobileNetV2 - Accuracy and Loss Over Epochs

16/16 [==============================] - 10s 439ms/step

Table 5-9: Training Fold 1 - MobileNetV2 - Classification Report

	precision	recall	f1-score	support
Healthy	0.98	0.99	0.98	242
Diseased	0.99	0.98	0.98	242
Accuracy			0.98	484
macro avg	0.98	0.98	0.98	484
weighted avg	0.98	0.98	0.98	484

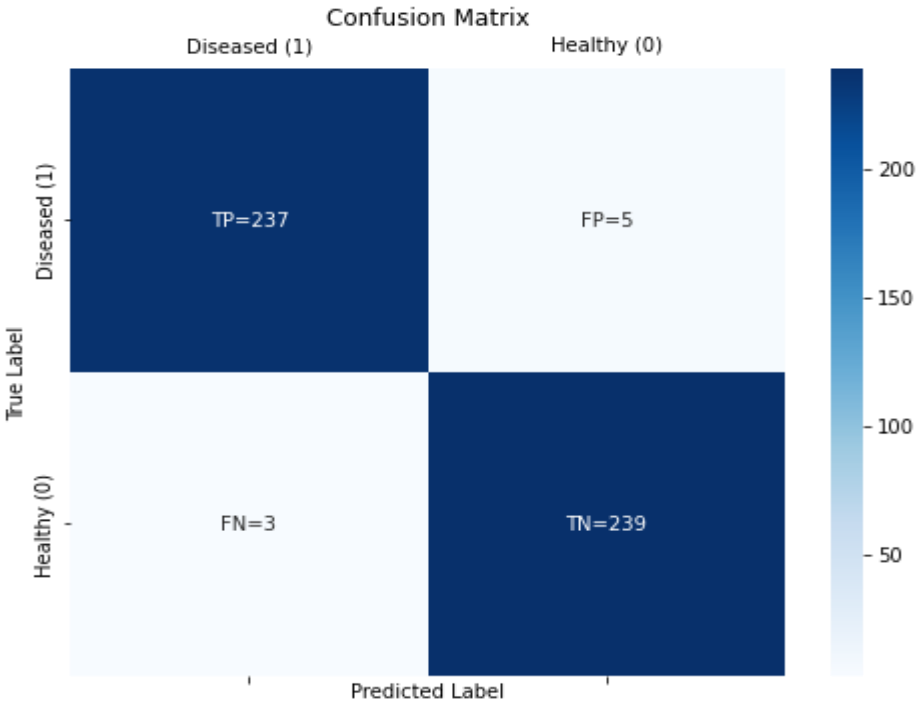


Figure 5-13: Training Fold 1 - MobileNetV2 - Confusion Matrix

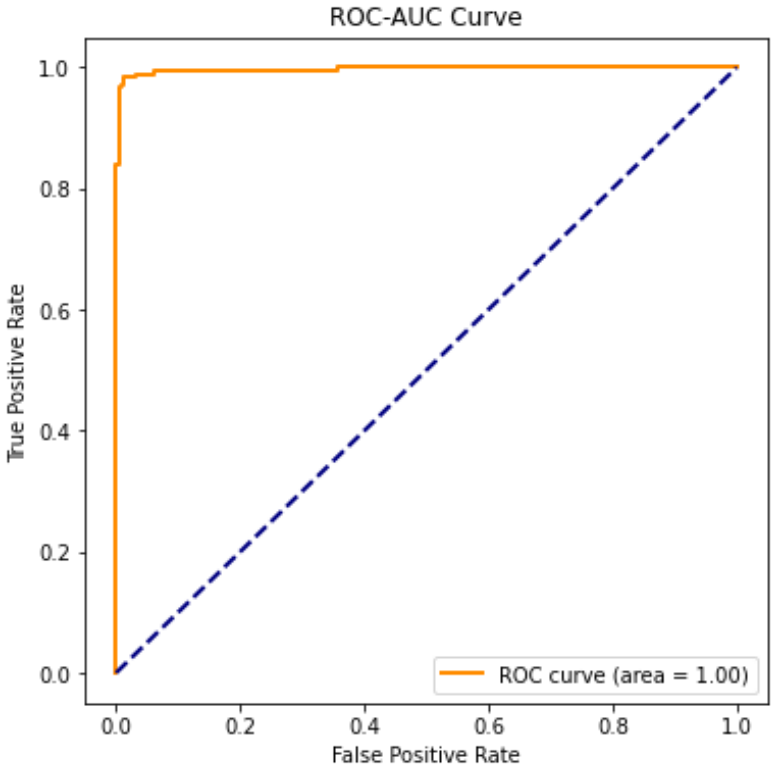


Figure 5-14: Training Fold 1 - MobileNetV2 - ROC-AUC Curve

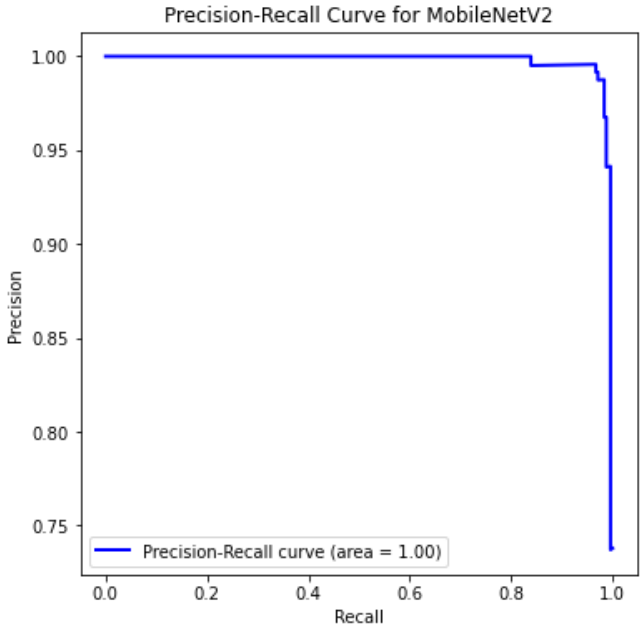


Figure 5-15: Training Fold 1 - MobileNetV2 – Precision-Recall Curve

5.3.1.2 MobileNetV2 Training Fold 2

Analysis for MobileNetV2's Training Fold 2 is presented as follows:

- **Precision and Recall:** MobileNetV2 continues to show high precision (0.99 Healthy, 0.97 Diseased) and recall (0.97 Healthy, 0.99 Diseased), as indicated in Table 5-10. This performance attests to the model's accuracy in correctly identifying the two classes.
- **F1-Score and Support:** The model maintains robust f1-scores of 0.98 for both Healthy and Diseased classes, with an equal distribution of support for each class, demonstrating consistent classification strength as detailed in Table 5-10.
- **Overall Accuracy:** An impressive overall accuracy of 98% is maintained, showcasing the model's continued proficiency in differentiating between the health conditions of plants as reported in Table 5-10.
- **Accuracy Over Epochs:** The accuracy graph shows a slight fluctuation in validation accuracy towards the end of training, suggesting a possibility of over-optimism in early validation accuracy which stabilizes in later epochs, as visualized in Figure 5-16.
- **Loss Over Epochs:** The loss graph depicts a consistent decrease in both training and validation loss, which indicates effective learning and model generalization across the training folds, as seen in Figure 5-16.

- **Confusion Matrix:** A very low number of false positives (FP=2) and false negatives (FN=8) highlights the model's high classification accuracy, as visualized in Figure 5-17.
- **ROC-AUC Curve:** The ROC curve with a perfect AUC of 1.00 denotes exceptional discrimination capabilities of the model, as depicted in Figure 5-18.
- **Precision-Recall Curve:** The precision-recall graph maintains an area under the curve at 1.00, confirming the model's high precision across all recall levels, as shown in Figure 5-19.

Results from Execution:

Training fold 2/5
Found 901 validated image filenames belonging to 2 classes.
Found 225 validated image filenames belonging to 2 classes.

Training model: MobileNetV2
Epoch 1/10
28/28 [==============================] - 57s 2s/step - loss: 0.6097 - accuracy: 0.69
04 - precision: 0.6904 - recall: 0.6904 - val_loss: 0.2983 - val_accuracy: 0.9330 - val_precisio
n: 0.9330 - val_recall: 0.9330 - lr: 1.0000e-04
Epoch 2/10
28/28 [==============================] - 45s 2s/step - loss: 0.3221 - accuracy: 0.87
34 - precision: 0.8734 - recall: 0.8734 - val_loss: 0.1918 - val_accuracy: 0.9509 - val_precisio
n: 0.9509 - val_recall: 0.9509 - lr: 1.0000e-04
Epoch 3/10
28/28 [==============================] - 45s 2s/step - loss: 0.2261 - accuracy: 0.92
41 - precision: 0.9241 - recall: 0.9241 - val_loss: 0.1375 - val_accuracy: 0.9777 - val_precisio
n: 0.9777 - val_recall: 0.9777 - lr: 1.0000e-04
Epoch 4/10
28/28 [==============================] - 46s 2s/step - loss: 0.1853 - accuracy: 0.94
13 - precision: 0.9413 - recall: 0.9413 - val_loss: 0.1036 - val_accuracy: 0.9777 - val_precisio
n: 0.9777 - val_recall: 0.9777 - lr: 1.0000e-04
Epoch 5/10
28/28 [==============================] - 46s 2s/step - loss: 0.1550 - accuracy: 0.94
02 - precision: 0.9402 - recall: 0.9402 - val_loss: 0.0847 - val_accuracy: 0.9866 - val_precisio
n: 0.9866 - val_recall: 0.9866 - lr: 1.0000e-04
Epoch 6/10
28/28 [==============================] - 44s 2s/step - loss: 0.1397 - accuracy: 0.95
63 - precision: 0.9563 - recall: 0.9563 - val_loss: 0.1036 - val_accuracy: 0.9688 - val_precisio
n: 0.9688 - val_recall: 0.9688 - lr: 1.0000e-04
Epoch 7/10
28/28 [==============================] - 36s 1s/step - loss: 0.1269 - accuracy: 0.95
40 - precision: 0.9540 - recall: 0.9540 - val_loss: 0.0904 - val_accuracy: 0.9777 - val_precisio
n: 0.9777 - val_recall: 0.9777 - lr: 1.0000e-04
Epoch 8/10
28/28 [==============================] - 26s 916ms/step - loss: 0.1115 - accuracy:
0.9632 - precision: 0.9632 - recall: 0.9632 - val_loss: 0.0844 - val_accuracy: 0.9821 - val_pre
cision: 0.9821 - val_recall: 0.9821 - lr: 1.0000e-04

Epoch 9/10
28/28 [==============================] - 25s 896ms/step - loss: 0.0823 - accuracy: 0.9781 - precision: 0.9781 - recall: 0.9781 - val_loss: 0.0793 - val_accuracy: 0.9688 - val_precision: 0.9688 - val_recall: 0.9688 - lr: 1.0000e-04
Epoch 10/10
28/28 [==============================] - 26s 921ms/step - loss: 0.1065 - accuracy: 0.9597 - precision: 0.9597 - recall: 0.9597 - val_loss: 0.0729 - val_accuracy: 0.9777 - val_precision: 0.9777 - val_recall: 0.9777 - lr: 1.0000e-04

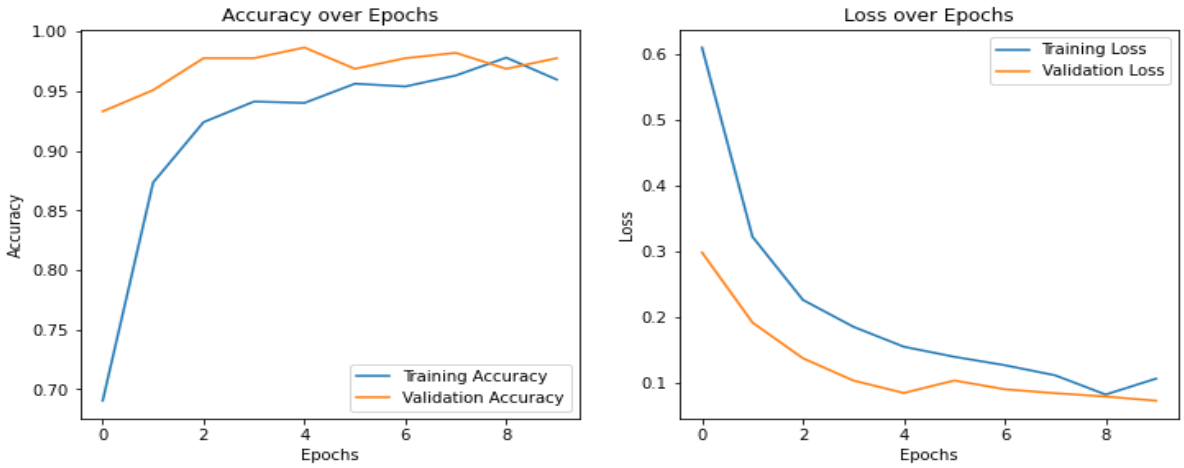


Figure 5-16: Training Fold 2 - MobileNetV2 - Accuracy and Loss Over Epochs

16/16 [==============================] - 7s 365ms/step

Table 5-10: Training Fold 2 - MobileNetV2 - Classification Report

	precision	**recall**	**f1-score**	**support**
Healthy	0.99	0.97	0.98	242
Diseased	0.97	0.99	0.98	242
accuracy			0.98	484
macro avg	0.98	0.98	0.98	484
weighted avg	0.98	0.98	0.98	484

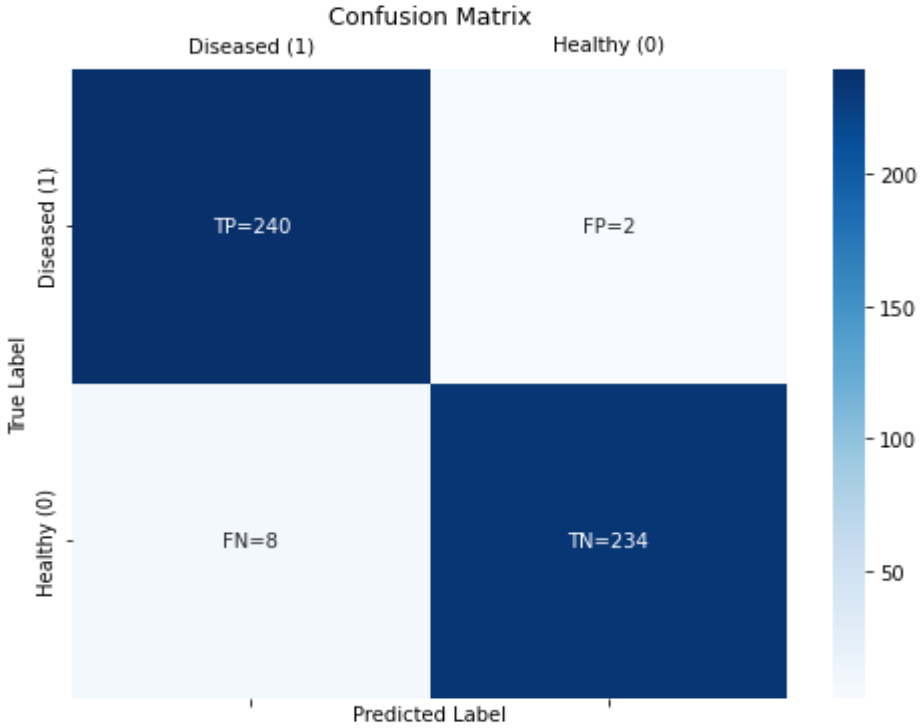


Figure 5-17: Training Fold 2 - MobileNetV2 - Confusion Matrix

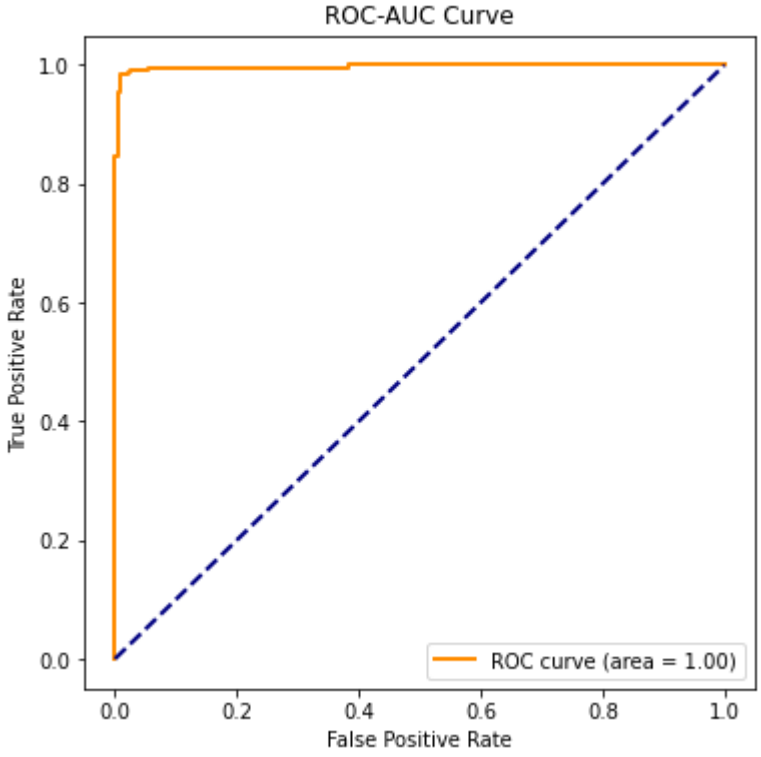


Figure 5-18: Training Fold 2 - MobileNetV2 - ROC-AUC Curve

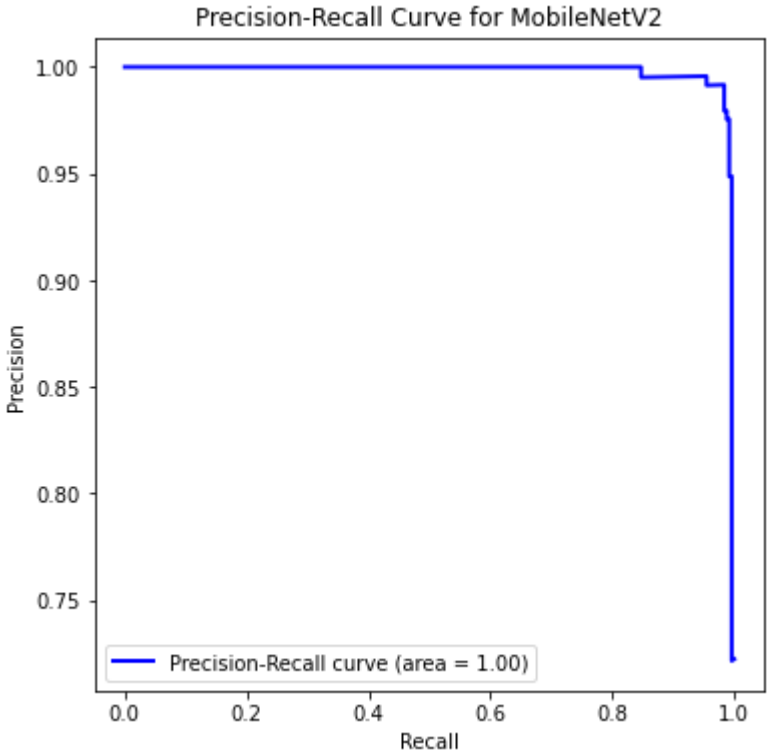


Figure 5-19: Training Fold 2 - MobileNetV2 - Precision-Recall Curve

5.3.1.3 MobileNetV2 Training Fold 3

Analysis for MobileNetV2's Training Fold 3 is presented as follows:

- **Precision and Recall:** MobileNetV2 maintains exceptional precision (0.99 Healthy, 0.97 Diseased) and recall (0.97 Healthy, 0.99 Diseased), showcasing its high accuracy in classifying plant conditions as specified in Table 5-11.
- **F1-Score and Support:** The model upholds robust f1-scores of 0.98 for both Healthy and Diseased classes, with a balanced number of instances for each class, ensuring reliable classification performance as detailed in Table 5-11.
- **Overall Accuracy:** The model continues to perform with high efficiency, achieving an overall accuracy of 98%, illustrating its proficiency in differentiating plant health conditions as recorded in Table 5-11.
- **Accuracy Over Epochs:** The accuracy graph displays a consistent performance with training and validation accuracy converging towards the end, indicating stable learning without overfitting as seen in Figure 5-20.
- **Loss Over Epochs:** The loss graph shows a decreasing trend in both training and validation loss, suggestive of the model's effective learning and generalization as visualized in Figure 5-20.

- **Confusion Matrix:** A minimal number of false positives (FP=2) and false negatives (FN=7) accentuates the model's high classification accuracy, as visualized in Figure 5-21.
- **ROC-AUC Curve:** The ROC curve with an AUC of 1.00 indicates outstanding discrimination capabilities of the model, as depicted in Figure 5-22.
- **Precision-Recall Curve:** A perfect area under the precision-recall curve (1.00) confirms the model's consistent precision across all levels of recall, as shown in Figure 5-23.

Results from Execution:

Training fold 3/5
Found 901 validated image filenames belonging to 2 classes.
Found 225 validated image filenames belonging to 2 classes.

Training model: MobileNetV2
Epoch 1/10
28/28 [==============================] - 28s 887ms/step - loss: 0.7519 - accuracy: 0.6502 - precision: 0.6502 - recall: 0.6502 - val_loss: 0.3395 - val_accuracy: 0.8839 - val_precision: 0.8839 - val_recall: 0.8839 - lr: 1.0000e-04
Epoch 2/10
28/28 [==============================] - 28s 985ms/step - loss: 0.3602 - accuracy: 0.8400 - precision: 0.8400 - recall: 0.8400 - val_loss: 0.1919 - val_accuracy: 0.9464 - val_precision: 0.9464 - val_recall: 0.9464 - lr: 1.0000e-04
Epoch 3/10
28/28 [==============================] - 25s 906ms/step - loss: 0.2256 - accuracy: 0.9183 - precision: 0.9183 - recall: 0.9183 - val_loss: 0.1473 - val_accuracy: 0.9688 - val_precision: 0.9688 - val_recall: 0.9688 - lr: 1.0000e-04
Epoch 4/10
28/28 [==============================] - 25s 909ms/step - loss: 0.1994 - accuracy: 0.9230 - precision: 0.9230 - recall: 0.9230 - val_loss: 0.1413 - val_accuracy: 0.9732 - val_precision: 0.9732 - val_recall: 0.9732 - lr: 1.0000e-04
Epoch 5/10
28/28 [==============================] - 26s 947ms/step - loss: 0.1599 - accuracy: 0.9431 - precision: 0.9431 - recall: 0.9431 - val_loss: 0.1346 - val_accuracy: 0.9554 - val_precision: 0.9554 - val_recall: 0.9554 - lr: 1.0000e-04
Epoch 6/10
28/28 [==============================] - 25s 885ms/step - loss: 0.1427 - accuracy: 0.9436 - precision: 0.9436 - recall: 0.9436 - val_loss: 0.0956 - val_accuracy: 0.9688 - val_precision: 0.9688 - val_recall: 0.9688 - lr: 1.0000e-04
Epoch 7/10
28/28 [==============================] - 28s 989ms/step - loss: 0.1167 - accuracy: 0.9655 - precision: 0.9655 - recall: 0.9655 - val_loss: 0.0964 - val_accuracy: 0.9821 - val_precision: 0.9821 - val_recall: 0.9821 - lr: 1.0000e-04

Epoch 8/10
28/28 [==============================] - 26s 932ms/step - loss: 0.1212 - accuracy: 0.9574 - precision: 0.9574 - recall: 0.9574 - val_loss: 0.0795 - val_accuracy: 0.9777 - val_precision: 0.9777 - val_recall: 0.9777 - lr: 1.0000e-04
Epoch 9/10
28/28 [==============================] - 26s 932ms/step - loss: 0.1084 - accuracy: 0.9678 - precision: 0.9678 - recall: 0.9678 - val_loss: 0.0884 - val_accuracy: 0.9777 - val_precision: 0.9777 - val_recall: 0.9777 - lr: 1.0000e-04
Epoch 10/10
28/28 [==============================] - 26s 948ms/step - loss: 0.0899 - accuracy: 0.9701 - precision: 0.9701 - recall: 0.9701 - val_loss: 0.0730 - val_accuracy: 0.9821 - val_precision: 0.9821 - val_recall: 0.9821 - lr: 1.0000e-04

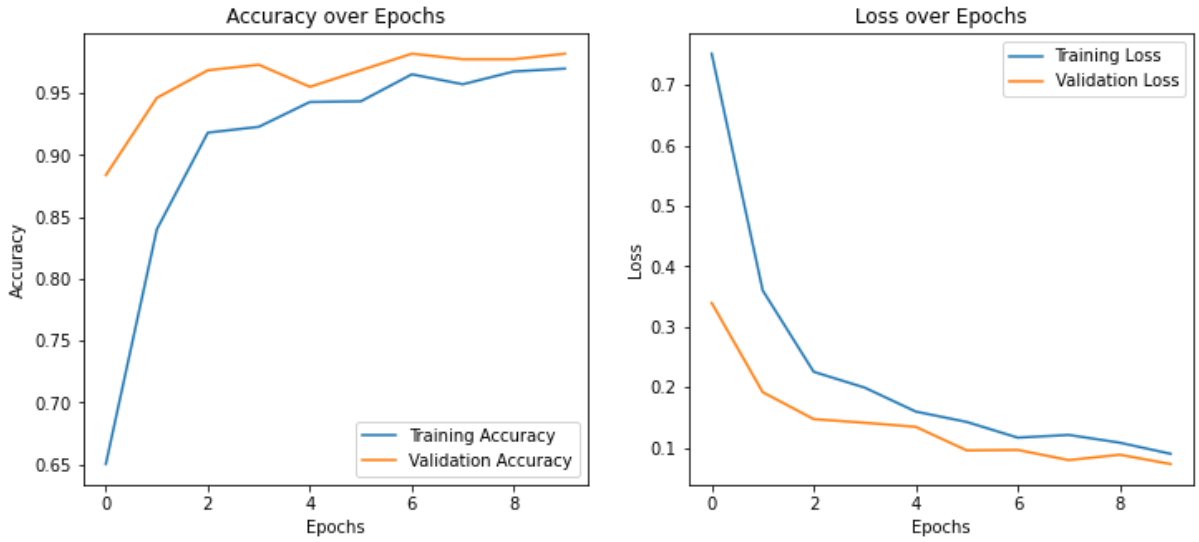


Figure 5-20: Training Fold 3 - MobileNetV2 - Accuracy and Loss Over Epochs

16/16 [==============================] - 8s 390ms/step

Table 5-11: Training Fold 3 - MobileNetV2 - Classification Report

	precision	recall	f1-score	support
Healthy	0.99	0.97	0.98	242
Diseased	0.97	0.99	0.98	242
accuracy			0.98	484
macro avg	0.98	0.98	0.98	484
weighted avg	0.98	0.98	0.98	484

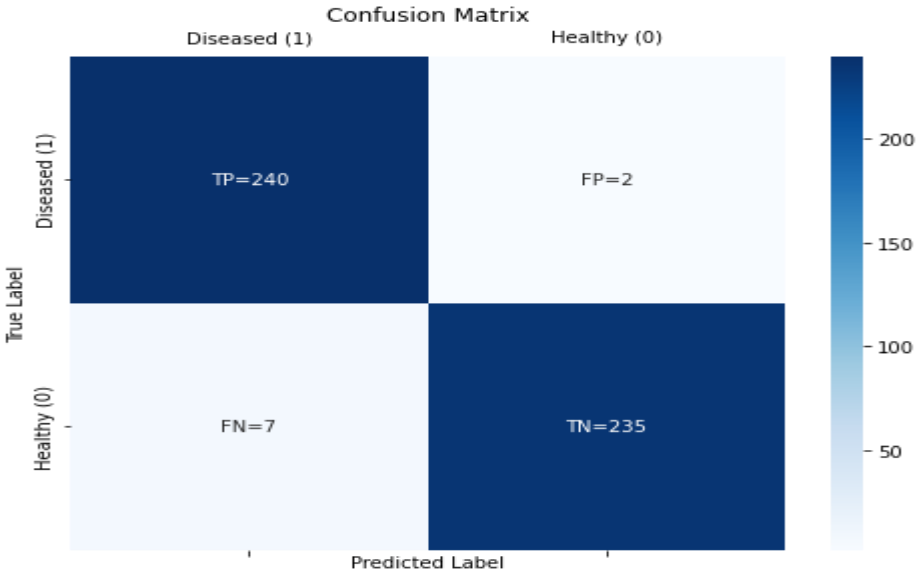


Figure 5-21: Training Fold 3 - MobileNetV2 - Confusion Matrix

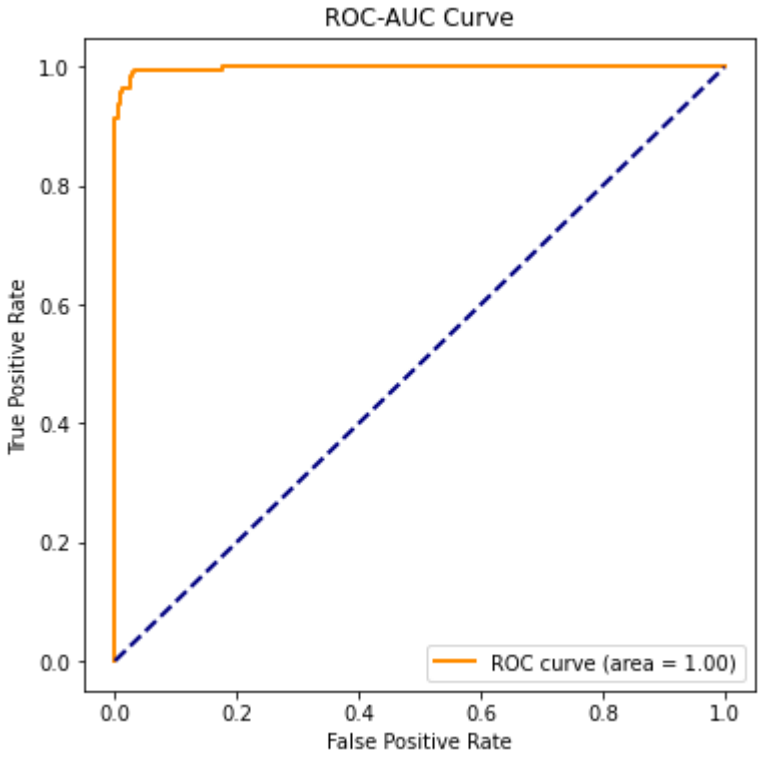


Figure 5-22: Training Fold 3 - MobileNetV2 - ROC-AUC Curve

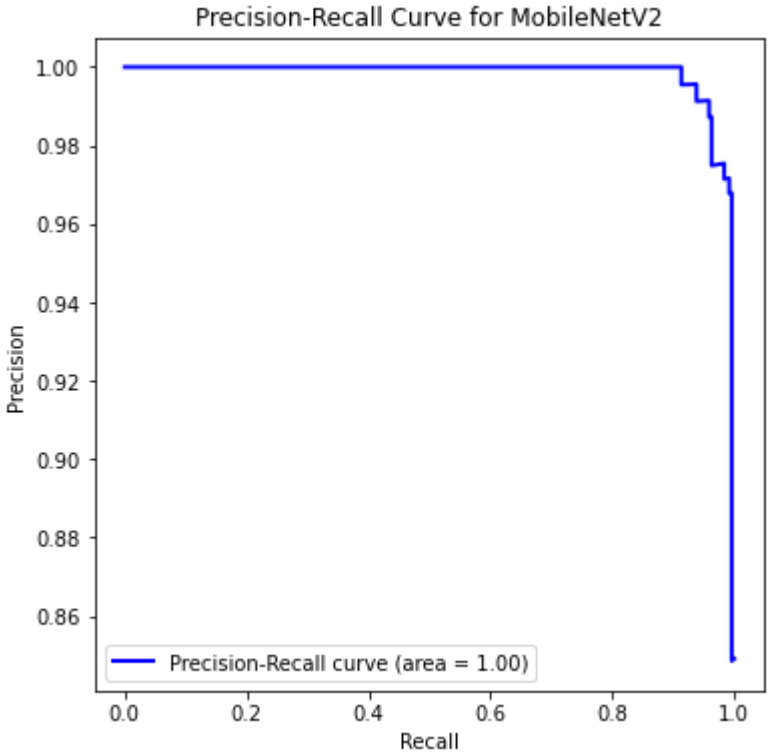


Figure 5-23: Training Fold 3 - MobileNetV2 - Precision-Recall Curve

5.3.1.4 MobileNetV2 Training Fold 4

Analysis for MobileNetV2's Training Fold 4 is presented as follows:

- **Precision and Recall:** MobileNetV2 showcases high precision (0.98 Healthy, 0.98 Diseased) and recall (0.98 Healthy, 0.98 Diseased), demonstrating its accurate class prediction capabilities as shown in Table 5-12.
- **F1-Score and Support:** With robust f1-scores of 0.98 for both Healthy and Diseased classes and balanced support (242 per class), the model's consistent output is evident, as detailed in Table 5-12.
- **Overall Accuracy:** The model sustains a high overall accuracy of 98%, indicating its adeptness in discerning plant health conditions, as reported in Table 5-12.
- **Accuracy Over Epochs:** The accuracy graph presents a stable trend with training and validation accuracies closely aligned, suggesting effective model performance without overfitting, as visualized in Figure 5-24.
- **Loss Over Epochs:** A continued decrease in both training and validation loss is observed, indicative of productive learning and generalization, as visualized in Figure 5-24.

- **Confusion Matrix:** The model's precision in classification is further highlighted by a low number of false positives (FP=4) and false negatives (FN=5), as visualized in Figure 5-25.
- **ROC-AUC Curve:** The ROC curve with an AUC of 1.00 indicates outstanding discrimination capabilities of the model, as depicted in Figure 5-26.
- **Precision-Recall Curve:** The precision-recall curve maintains a maximum area of 1.00, reaffirming the model's accurate precision across various recall thresholds, as shown in Figure 5-27.

Results from Execution:

Training fold 4/5
Found 901 validated image filenames belonging to 2 classes.
Found 225 validated image filenames belonging to 2 classes.

Training model: MobileNetV2
Epoch 1/10
28/28 [==============================] - 24s 744ms/step - loss: 0.7293 - accuracy: 0.6099 - precision: 0.6099 - recall: 0.6099 - val_loss: 0.3646 - val_accuracy: 0.8929 - val_precision: 0.8929 - val_recall: 0.8929 - lr: 1.0000e-04
Epoch 2/10
28/28 [==============================] - 19s 682ms/step - loss: 0.3851 - accuracy: 0.8377 - precision: 0.8377 - recall: 0.8377 - val_loss: 0.2400 - val_accuracy: 0.9509 - val_precision: 0.9509 - val_recall: 0.9509 - lr: 1.0000e-04
Epoch 3/10
28/28 [==============================] - 19s 690ms/step - loss: 0.2806 - accuracy: 0.8884 - precision: 0.8884 - recall: 0.8884 - val_loss: 0.1887 - val_accuracy: 0.9554 - val_precision: 0.9554 - val_recall: 0.9554 - lr: 1.0000e-04
Epoch 4/10
28/28 [==============================] - 19s 678ms/step - loss: 0.2184 - accuracy: 0.9229 - precision: 0.9229 - recall: 0.9229 - val_loss: 0.1503 - val_accuracy: 0.9554 - val_precision: 0.9554 - val_recall: 0.9554 - lr: 1.0000e-04
Epoch 5/10
28/28 [==============================] - 19s 687ms/step - loss: 0.1820 - accuracy: 0.9379 - precision: 0.9379 - recall: 0.9379 - val_loss: 0.1111 - val_accuracy: 0.9866 - val_precision: 0.9866 - val_recall: 0.9866 - lr: 1.0000e-04
Epoch 6/10
28/28 [==============================] - 19s 677ms/step - loss: 0.1454 - accuracy: 0.9551 - precision: 0.9551 - recall: 0.9551 - val_loss: 0.1055 - val_accuracy: 0.9688 - val_precision: 0.9688 - val_recall: 0.9688 - lr: 1.0000e-04
Epoch 7/10
28/28 [==============================] - 19s 666ms/step - loss: 0.1348 - accuracy: 0.9471 - precision: 0.9471 - recall: 0.9471 - val_loss: 0.0875 - val_accuracy: 0.9821 - val_precision: 0.9821 - val_recall: 0.9821 - lr: 1.0000e-04
Epoch 8/10

28/28 [==============================] - 19s 672ms/step - loss: 0.1028 - accuracy: 0.9735 - precision: 0.9735 - recall: 0.9735 - val_loss: 0.0823 - val_accuracy: 0.9688 - val_precision: 0.9688 - val_recall: 0.9688 - lr: 1.0000e-04
Epoch 9/10
28/28 [==============================] - 19s 684ms/step - loss: 0.1160 - accuracy: 0.9620 - precision: 0.9620 - recall: 0.9620 - val_loss: 0.0810 - val_accuracy: 0.9821 - val_precision: 0.9821 - val_recall: 0.9821 - lr: 1.0000e-04
Epoch 10/10
28/28 [==============================] - 19s 684ms/step - loss: 0.0940 - accuracy: 0.9643 - precision: 0.9643 - recall: 0.9643 - val_loss: 0.0761 - val_accuracy: 0.9732 - val_precision: 0.9732 - val_recall: 0.9732 - lr: 1.0000e-04

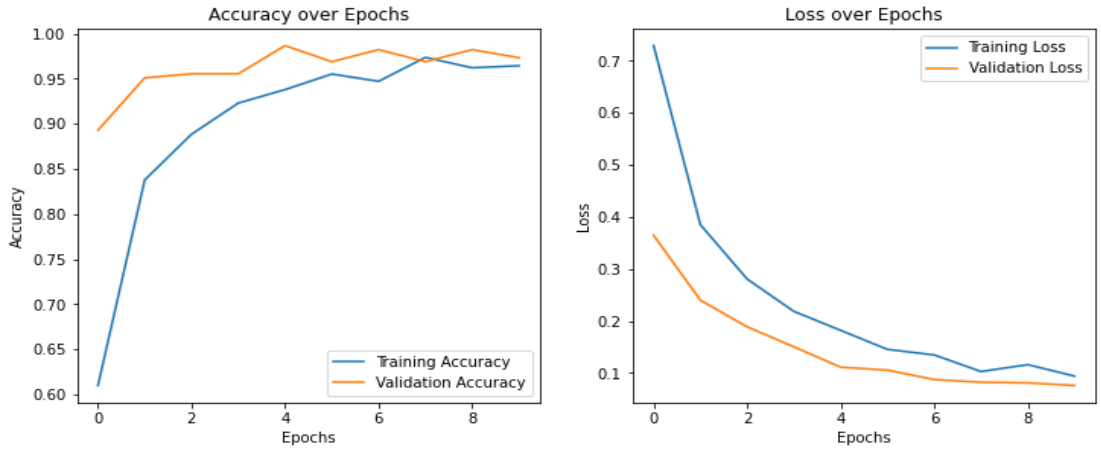


Figure 5-24: Training Fold 4 - MobileNetV2 - Accuracy and Loss Over Epochs

16/16 [==============================] - 6s 291ms/step

Table 5-12: Training Fold 4 - MobileNetV2 - Classification Report

	precision	**recall**	**f1-score**	**support**
Healthy	0.98	0.98	0.98	242
Diseased	0.98	0.98	0.98	242
accuracy			0.98	484
macro avg	0.98	0.98	0.98	484
weighted avg	0.98	0.98	0.98	484

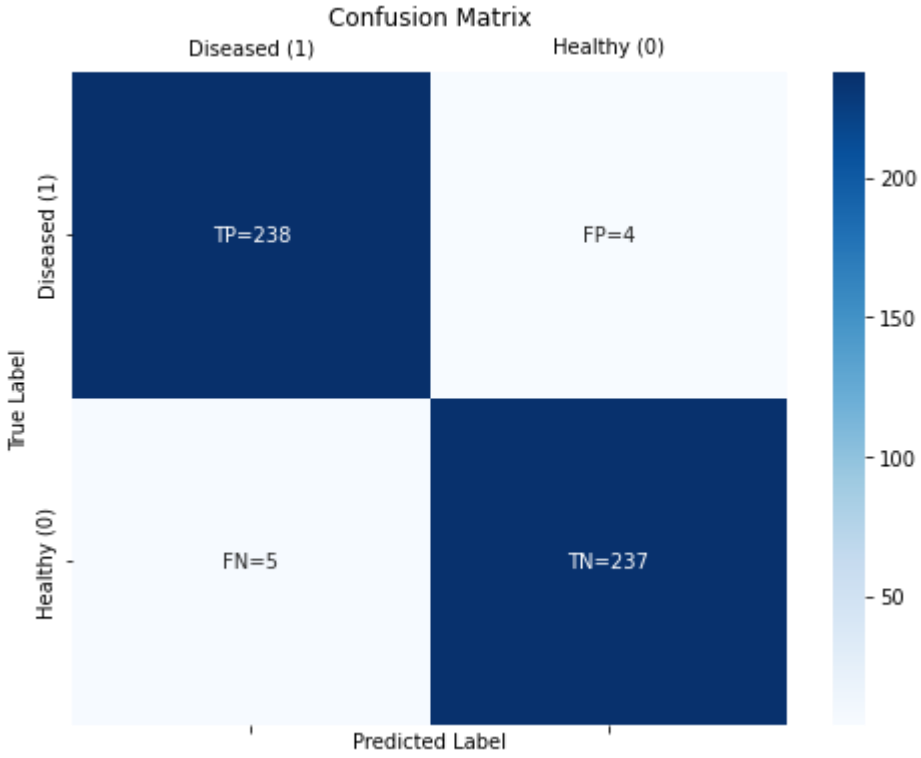


Figure 5-25: Training Fold 4 - MobileNetV2 - Confusion Matrix

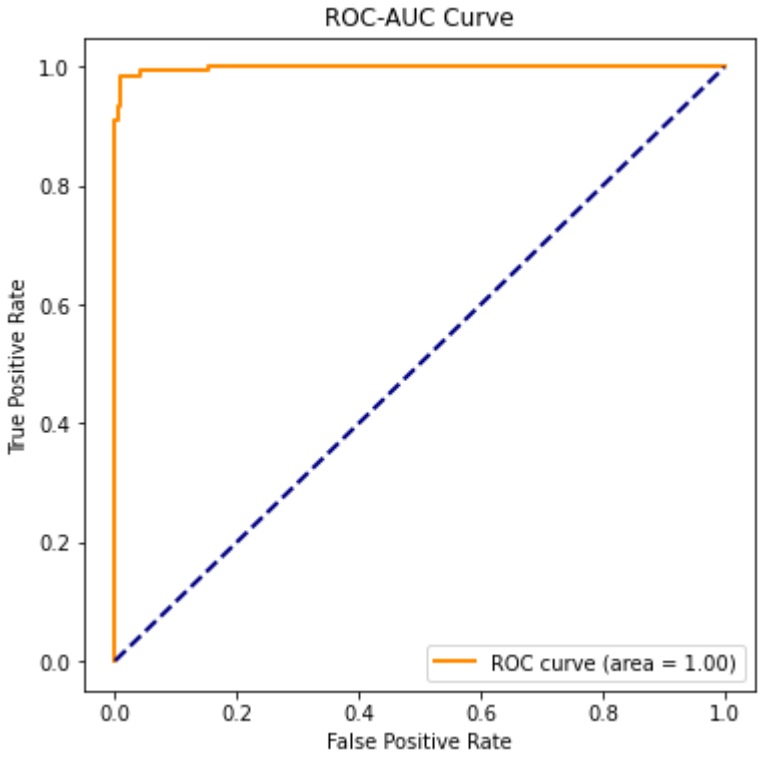


Figure 5-26: Training Fold 4 - MobileNetV2 - ROC-AUC Curve

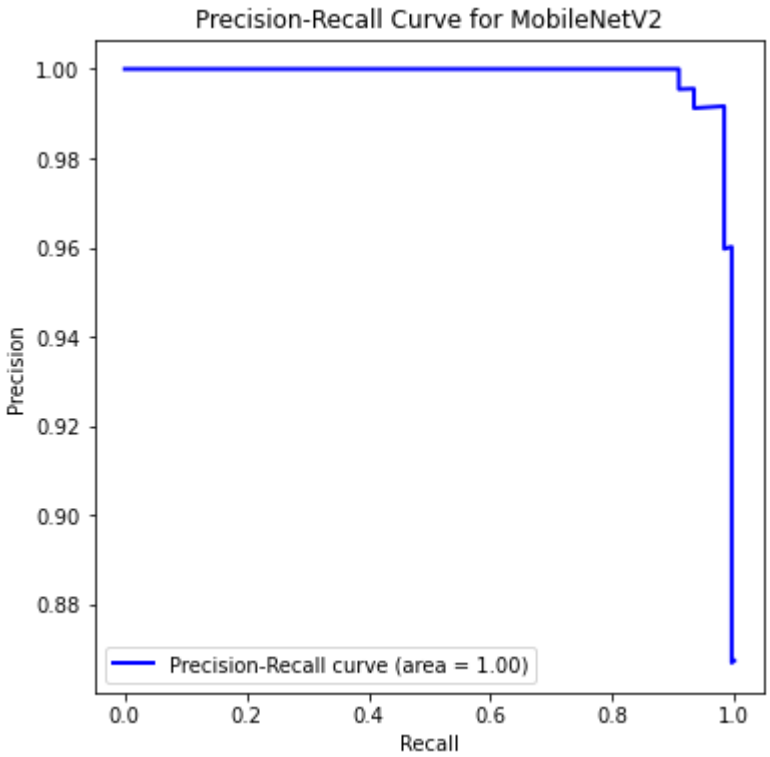


Figure 5-27: Training Fold 4 - MobileNetV2 - Precision-Recall Curve

5.3.1.5 MobileNetV2 Training Fold 5

Analysis for MobileNetV2's Training Fold 5 is presented as follows:

- **Precision and Recall:** The model demonstrates high precision (0.98 Healthy, 0.97 Diseased) and recall (0.97 Healthy, 0.98 Diseased), as indicated in Table 5-13. This reflects the model's adeptness in accurate class predictions.
- **F1-Score and Support:** Robust f1-scores (0.97 for both classes) and balanced support (242 per class) underscore the model's consistent output as detailed in Table 5-13.
- **Overall Accuracy:** An overall accuracy of 97% is achieved, showcasing the model's capability in differentiating plant health conditions as reported in Table 5-13.
- **Accuracy Over Epochs:** The accuracy graph shows a plateau after initial improvements, suggesting that the model reaches its optimal performance without overfitting, as visualized in Figure 5-28.
- **Loss Over Epochs:** The loss graph displays a decrease in both training and validation loss, indicating effective learning and model generalization, as visualized in Figure 5-28.

- **Confusion Matrix:** The confusion matrix reveals a minimal number of false positives (FP=6) and false negatives (FN=7), highlighting the model's classification accuracy, as visualized in Figure 5-29.
- **ROC-AUC Curve:** Exemplified by a flawless ROC curve with an AUC score reaching the ideal mark of 1.00, Figure 5-30 illustrates the model's unparalleled ability to differentiate between classes.
- **Precision-Recall Curve:** The precision-recall graph, also with an area under the curve of 1.00, reaffirms the model's high precision across varying recall levels, as shown in Figure 5-31.

Results from Execution:

Training fold 5/5
Found 901 validated image filenames belonging to 2 classes.
Found 225 validated image filenames belonging to 2 classes.

Training model: MobileNetV2
Epoch 1/10
28/28 [==============================] - 27s 816ms/step - loss: 0.7048 - accuracy: 0.6628 - precision: 0.6628 - recall: 0.6628 - val_loss: 0.3359 - val_accuracy: 0.9152 - val_precision: 0.9152 - val_recall: 0.9152 - lr: 1.0000e-04
Epoch 2/10
28/28 [==============================] - 19s 678ms/step - loss: 0.3402 - accuracy: 0.8677 - precision: 0.8677 - recall: 0.8677 - val_loss: 0.2104 - val_accuracy: 0.9286 - val_precision: 0.9286 - val_recall: 0.9286 - lr: 1.0000e-04
Epoch 3/10
28/28 [==============================] - 19s 667ms/step - loss: 0.2493 - accuracy: 0.9114 - precision: 0.9114 - recall: 0.9114 - val_loss: 0.1484 - val_accuracy: 0.9509 - val_precision: 0.9509 - val_recall: 0.9509 - lr: 1.0000e-04
Epoch 4/10
28/28 [==============================] - 19s 681ms/step - loss: 0.1831 - accuracy: 0.9448 - precision: 0.9448 - recall: 0.9448 - val_loss: 0.1251 - val_accuracy: 0.9732 - val_precision: 0.9732 - val_recall: 0.9732 - lr: 1.0000e-04
Epoch 5/10
28/28 [==============================] - 19s 686ms/step - loss: 0.1823 - accuracy: 0.9356 - precision: 0.9356 - recall: 0.9356 - val_loss: 0.0985 - val_accuracy: 0.9821 - val_precision: 0.9821 - val_recall: 0.9821 - lr: 1.0000e-04
Epoch 6/10
28/28 [==============================] - 19s 668ms/step - loss: 0.1341 - accuracy: 0.9505 - precision: 0.9505 - recall: 0.9505 - val_loss: 0.0926 - val_accuracy: 0.9688 - val_precision: 0.9688 - val_recall: 0.9688 - lr: 1.0000e-04
Epoch 7/10
28/28 [==============================] - 20s 716ms/step - loss: 0.1174 - accuracy: 0.9666 - precision: 0.9666 - recall: 0.9666 - val_loss: 0.0803 - val_accuracy: 0.9688 - val_precision: 0.9688 - val_recall: 0.9688 - lr: 1.0000e-04
Epoch 8/10

28/28 [==============================] - 19s 669ms/step - loss: 0.0992 - accuracy: 0.9678 - precision: 0.9678 - recall: 0.9678 - val_loss: 0.0707 - val_accuracy: 0.9732 - val_precision: 0.9732 - val_recall: 0.9732 - lr: 1.0000e-04
Epoch 9/10
28/28 [==============================] - 19s 688ms/step - loss: 0.1048 - accuracy: 0.9632 - precision: 0.9632 - recall: 0.9632 - val_loss: 0.0865 - val_accuracy: 0.9732 - val_precision: 0.9732 - val_recall: 0.9732 - lr: 1.0000e-04
Epoch 10/10
28/28 [==============================] - 19s 677ms/step - loss: 0.0857 - accuracy: 0.9781 - precision: 0.9781 - recall: 0.9781 - val_loss: 0.0745 - val_accuracy: 0.9732 - val_precision: 0.9732 - val_recall: 0.9732 - lr: 1.0000e-04

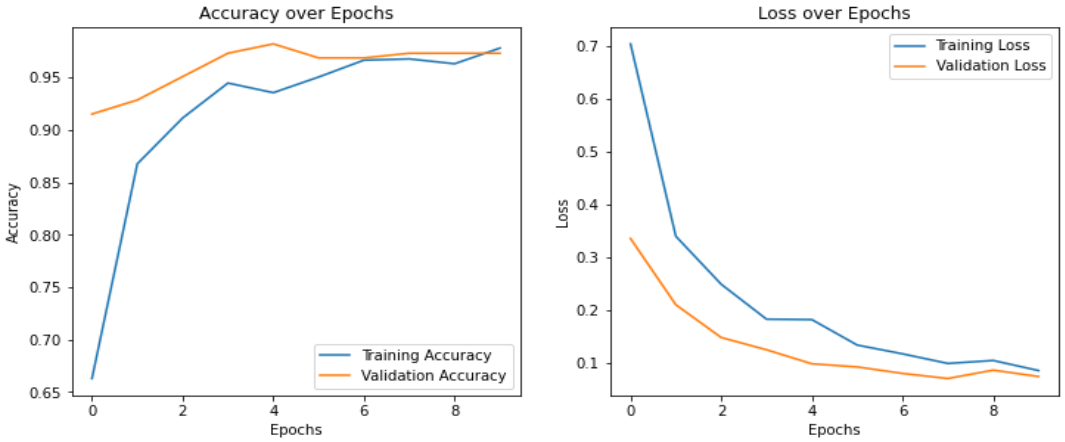


Figure 5-28: Training Fold 5 - MobileNetV2 - Accuracy and Loss Over Epochs

16/16 [==============================] - 6s 286ms/step

Table 5-13: Training Fold 5 - MobileNetV2 - Classification Report

	precision	**recall**	**f1-score**	**support**
Healthy	0.98	0.97	0.97	242
Diseased	0.97	0.98	0.97	242
accuracy			0.97	484
macro avg	0.97	0.97	0.97	484
weighted avg	0.97	0.97	0.97	484

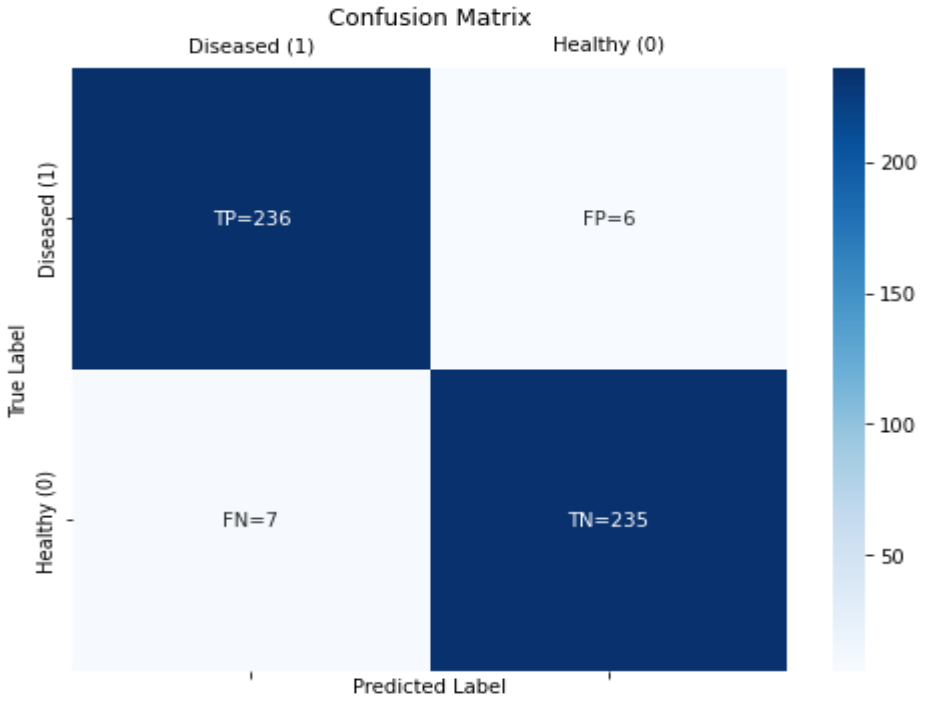


Figure 5-29: Training Fold 5 - MobileNetV2 - Confusion Matrix

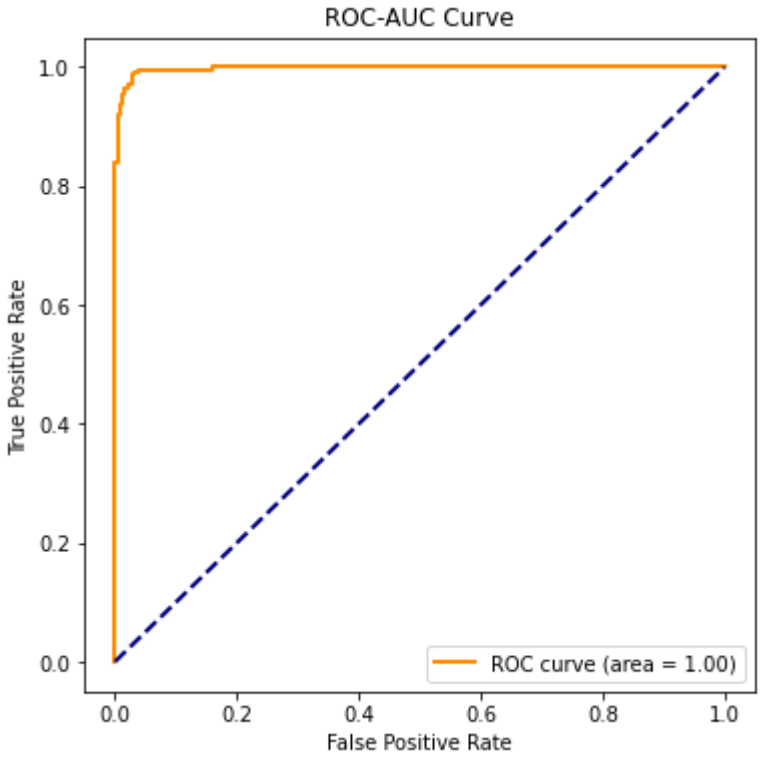


Figure 5-30: Training Fold 5 - MobileNetV2 - ROC-AUC Curve

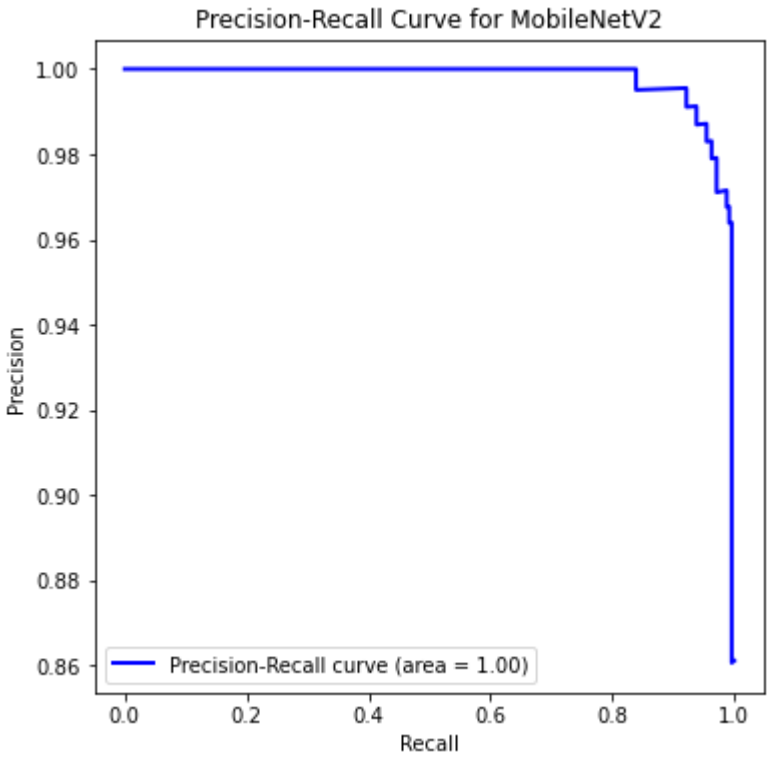


Figure 5-31: Training Fold 5 - MobileNetV2 - Precision-Recall Curve

5.3.1.6 Strengths and Limitations of MobileNetV2

After analysing five training folds of MobileNetV2, below are its strengths and limitations:

Strengths:

- **High Precision and Recall Across Classes:** MobileNetV2 demonstrated exceptional precision and recall rates (nearly 0.98-0.99) for both healthy and diseased plant classes across all training folds, indicating its superior accuracy in class identification.
- **Consistent F1-Scores:** The model achieved robust F1-scores of approximately 0.98 for both classes in each fold, highlighting its consistent classification performance.
- **Stable Learning Over Epochs:** The accuracy and loss graphs for each training fold showed stable learning curves without signs of overfitting or underfitting, illustrating the model's well-calibrated learning process.
- **Outstanding Discrimination Capability:** The precision-recall and ROC-AUC curves, with AUC of 1.00, underscored the model's exceptional ability to discriminate between healthy and diseased plant conditions.

Limitations:

- **Minor Fluctuations in Validation Accuracy:** A slight fluctuation was observed in validation accuracy towards the end of training in some folds, suggesting a potential over-optimism in early validation accuracy that stabilizes in later epochs.
- **Minimal False Positives and Negatives:** Although low, the presence of false positives and negatives in the confusion matrices indicates room for improvement in reducing misclassifications further.
- **Plateau in Accuracy Over Epochs:** In the final training fold, the accuracy graph showed a plateau after initial improvements, suggesting that the model may reach its optimal performance relatively early in the training process.

This analysis provides a comprehensive understanding of MobileNetV2's operational strengths and areas for potential enhancement, particularly in its application for plant disease detection. The model's high precision, recall, and F1-scores across all folds, coupled with its stable learning and outstanding discrimination capabilities, highlight its efficacy. However, the slight fluctuations in validation accuracy and the minimal but present false positives and negatives suggest areas where further research and model tuning could improve performance.

5.3.2 NASNetMobile Performance Analysis

This section embarks on a detailed analysis of NASNetMobile's efficacy through the lens of its performance across five training folds. This section dissects the outcomes of each fold, shedding light on the model's precision, recall, F1-scores, and overall accuracy in distinguishing healthy from diseased plant conditions. Additionally, it delves into the dynamics of model learning, illustrated by accuracy and loss trends over epochs, and evaluates the model's discriminative capabilities through confusion matrices, ROC-AUC, and precision-recall curves. This rigorous examination aims to provide a nuanced understanding of NASNetMobile's strengths and areas for improvement in the context of plant disease detection.

5.3.2.1 NASNetMobile Training Fold 1

Analysis for NASNetMobile’s Training Fold 1 is presented as follows:

- **Precision and Recall:** Demonstrates high precision for Healthy (0.96) and Diseased (0.92) classes, with recall slightly favoring Diseased (0.96 over Healthy's 0.91) as depicted in Table 5-14.

- **F1-Score and Support:** The f1-scores, indicating a balanced precision-recall relationship, are 0.93 for Healthy and 0.94 for Diseased, with equal class support of 242 as shown in Table 5-14.
- **Overall Accuracy:** The model achieves a notable overall accuracy of 94%, confirming its effectiveness in this fold, as outlined in Table 5-14.
- **Accuracy Over Epochs:** The model's accuracy improvement is evident across the training duration, showcasing efficient learning, as visualized in Figure 5-32.
- **Loss Over Epochs:** The loss trends, indicating learning stability and generalization capability, are presented in Figure 5-32.
- **Confusion Matrix:** The high number of True Positives and True Negatives, with relatively few False Positives and False Negatives, are detailed in Figure 5-33.
- **ROC-AUC Curve:** Figure 5-34 demonstrates the model's remarkable capability to differentiate, reflected by an area score of 0.99.
- **Precision-Recall Curve:** Demonstrates the model's high precision across various recall levels, also with an area of 0.99, as shown in Figure 5-35.

Results from Code Execution:

Training model: NASNetMobile

```
Epoch 1/10
28/28 [==============================] - 103s 2s/step - loss: 0.7252 - accuracy: 0.5
945 - precision: 0.5945 - recall: 0.5945 - val_loss: 0.5264 - val_accuracy: 0.7679 - val_precisi
on: 0.7679 - val_recall: 0.7679 - lr: 1.0000e-04
Epoch 2/10
28/28 [==============================] - 53s 2s/step - loss: 0.5284 - accuracy: 0.75
00 - precision: 0.7500 - recall: 0.7500 - val_loss: 0.3852 - val_accuracy: 0.8973 - val_precisio
n: 0.8973 - val_recall: 0.8973 - lr: 1.0000e-04
Epoch 3/10
28/28 [==============================] - 55s 2s/step - loss: 0.4276 - accuracy: 0.81
80 - precision: 0.8180 - recall: 0.8180 - val_loss: 0.3603 - val_accuracy: 0.8795 - val_precisio
n: 0.8795 - val_recall: 0.8795 - lr: 1.0000e-04
Epoch 4/10
28/28 [==============================] - 52s 2s/step - loss: 0.3758 - accuracy: 0.84
79 - precision: 0.8479 - recall: 0.8479 - val_loss: 0.2808 - val_accuracy: 0.9464 - val_precisio
n: 0.9464 - val_recall: 0.9464 - lr: 1.0000e-04
Epoch 5/10
28/28 [==============================] - 51s 2s/step - loss: 0.3303 - accuracy: 0.86
41 - precision: 0.8641 - recall: 0.8641 - val_loss: 0.2772 - val_accuracy: 0.9107 - val_precisio
n: 0.9107 - val_recall: 0.9107 - lr: 1.0000e-04
Epoch 6/10
```

28/28 [==============================] - 52s 2s/step - loss: 0.3209 - accuracy: 0.86
87 - precision: 0.8687 - recall: 0.8687 - val_loss: 0.2518 - val_accuracy: 0.9375 - val_precisio
n: 0.9375 - val_recall: 0.9375 - lr: 1.0000e-04
Epoch 7/10
28/28 [==============================] - 54s 2s/step - loss: 0.2611 - accuracy: 0.90
32 - precision: 0.9032 - recall: 0.9032 - val_loss: 0.2262 - val_accuracy: 0.9375 - val_precisio
n: 0.9375 - val_recall: 0.9375 - lr: 1.0000e-04
Epoch 8/10
28/28 [==============================] - 52s 2s/step - loss: 0.2568 - accuracy: 0.90
44 - precision: 0.9044 - recall: 0.9044 - val_loss: 0.1931 - val_accuracy: 0.9688 - val_precisio
n: 0.9688 - val_recall: 0.9688 - lr: 1.0000e-04
Epoch 9/10
28/28 [==============================] - 52s 2s/step - loss: 0.2266 - accuracy: 0.91
82 - precision: 0.9182 - recall: 0.9182 - val_loss: 0.1923 - val_accuracy: 0.9509 - val_precisio
n: 0.9509 - val_recall: 0.9509 - lr: 1.0000e-04
Epoch 10/10
28/28 [==============================] - 53s 2s/step - loss: 0.2301 - accuracy: 0.91
59 - precision: 0.9159 - recall: 0.9159 - val_loss: 0.1776 - val_accuracy: 0.9509 - val_precisio
n: 0.9509 - val_recall: 0.9509 - lr: 1.0000e-04

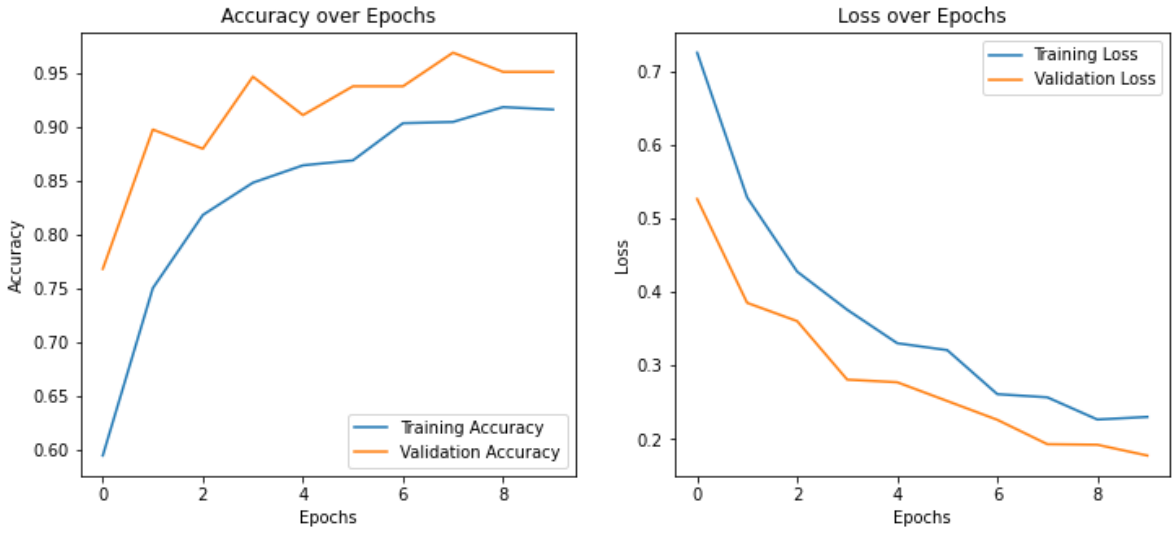


Figure 5-32: Training Fold 1 - NASNetMobile - Accuracy and Loss Over Epochs

16/16 [==============================] - 23s 671ms/step

Table 5-14: Training Fold 1 - NASNetMobile - Classification Report

	precision	**recall**	**f1-score**	**support**
Healthy	0.96	0.91	0.93	242
Diseased	0.92	0.96	0.94	242
Accuracy			0.94	484
macro avg	0.94	0.94	0.94	484
weighted avg	0.94	0.94	0.94	484

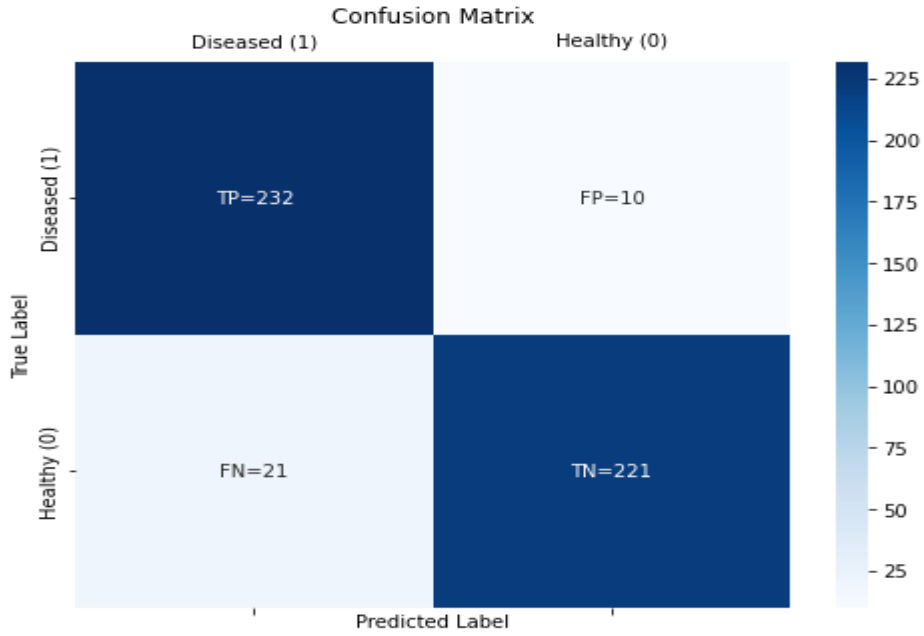


Figure 5-33: Training Fold 1 - NASNetMobile - Confusion Matrix

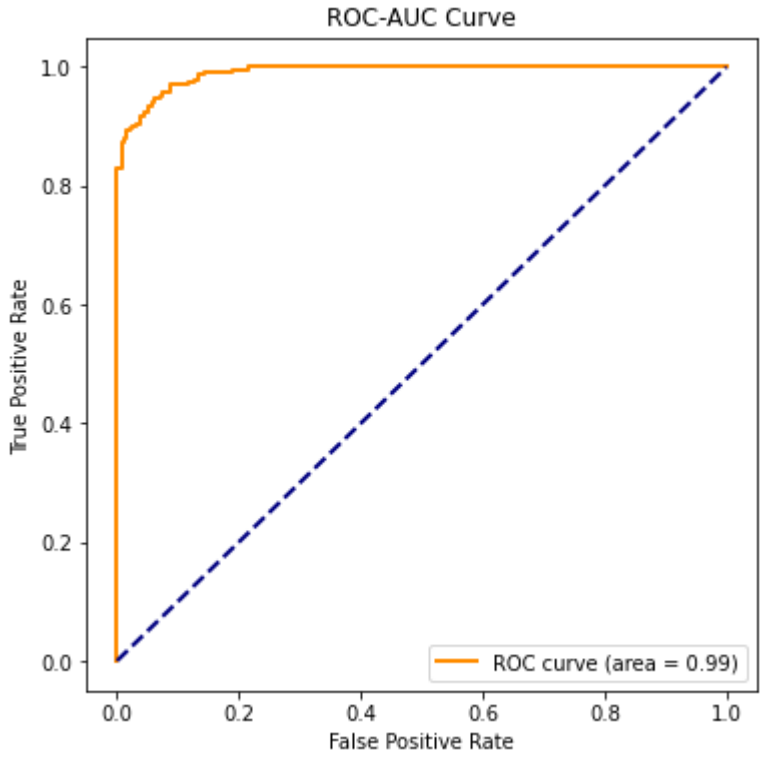


Figure 5-34: Training Fold 1 - NASNetMobile - ROC-AUC Curve

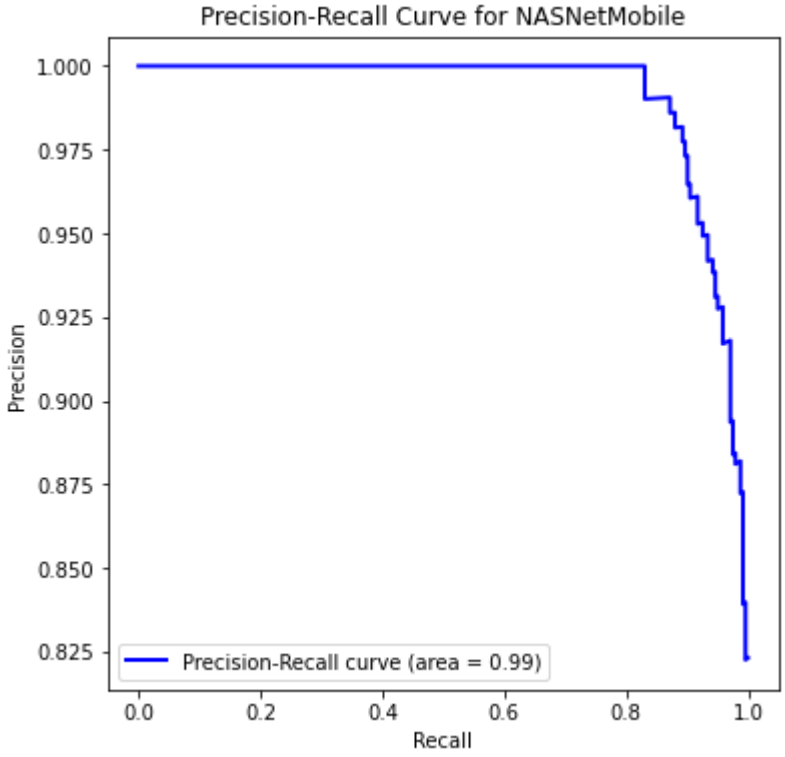


Figure 5-35: Training Fold 1 - NASNetMobile - Precision-Recall Curve

5.3.2.2 NASNetMobile Training Fold 2

Analysis for NASNetMobile's Training Fold 2 is presented as follows:

- **Precision and Recall:** The model sustained high precision (0.95 for Healthy, 0.94 for Diseased) with recall rates reflecting a balanced detection capability (0.93 for Healthy, 0.95 for Diseased) as shown in Table 5-15.
- **F1-Score and Support:** Both classes achieved an f1-score of 0.94, underlining consistent classification effectiveness, supported by an equitable distribution of instances (242 per class) as shown in Table 5-15.
- **Overall Accuracy:** The model preserved a high overall accuracy of 94%, indicating reliable predictive performance as outlined in Table 5-15.
- **Accuracy Over Epochs:** The ascending trend of validation accuracy alongside training accuracy suggests a stable learning process without overfitting as visualized in Figure 5-36.
- **Loss Over Epochs:** A converging pattern of training and validation loss points to effective learning and model generalization as visualized in Figure 5-36.
- **Confusion Matrix:** A low number of false positives (FP=11) and false negatives (FN=16) signifies the model's adeptness in accurate classification are detailed in Figure 5-37.
- **ROC-AUC Curve:** The area under the curve (AUC) at 0.99 indicates a high true positive rate with minimal false positives is illustrated in Figure 5-38.
- **Precision-Recall Curve:** A high area under the precision-recall curve (0.99) confirms the model's precision across various thresholds as shown in Figure 5-39.

Results from code execution:

Training model: NASNetMobile

```
Epoch 1/10
28/28 [==============================] - 52s 1s/step - loss: 0.6832 - accuracy: 0.61
33 - precision: 0.6133 - recall: 0.6133 - val_loss: 0.5263 - val_accuracy: 0.7723 - val_precisio
n: 0.7723 - val_recall: 0.7723 - lr: 1.0000e-04
Epoch 2/10
28/28 [==============================] - 36s 1s/step - loss: 0.5094 - accuracy: 0.76
41 - precision: 0.7641 - recall: 0.7641 - val_loss: 0.4177 - val_accuracy: 0.8661 - val_precisio
n: 0.8661 - val_recall: 0.8661 - lr: 1.0000e-04
Epoch 3/10
28/28 [==============================] - 36s 1s/step - loss: 0.4263 - accuracy: 0.81
82 - precision: 0.8182 - recall: 0.8182 - val_loss: 0.3440 - val_accuracy: 0.9018 - val_precisio
n: 0.9018 - val_recall: 0.9018 - lr: 1.0000e-04
```

Epoch 4/10
28/28 [==============================] - 35s 1s/step - loss: 0.3747 - accuracy: 0.84
58 - precision: 0.8458 - recall: 0.8458 - val_loss: 0.2943 - val_accuracy: 0.8929 - val_precisio
n: 0.8929 - val_recall: 0.8929 - lr: 1.0000e-04
Epoch 5/10
28/28 [==============================] - 36s 1s/step - loss: 0.3136 - accuracy: 0.89
07 - precision: 0.8907 - recall: 0.8907 - val_loss: 0.2780 - val_accuracy: 0.9286 - val_precisio
n: 0.9286 - val_recall: 0.9286 - lr: 1.0000e-04
Epoch 6/10
28/28 [==============================] - 35s 1s/step - loss: 0.3088 - accuracy: 0.88
84 - precision: 0.8884 - recall: 0.8884 - val_loss: 0.2535 - val_accuracy: 0.9196 - val_precisio
n: 0.9196 - val_recall: 0.9196 - lr: 1.0000e-04
Epoch 7/10
28/28 [==============================] - 35s 1s/step - loss: 0.2805 - accuracy: 0.89
18 - precision: 0.8918 - recall: 0.8918 - val_loss: 0.2386 - val_accuracy: 0.9241 - val_precisio
n: 0.9241 - val_recall: 0.9241 - lr: 1.0000e-04
Epoch 8/10
28/28 [==============================] - 35s 1s/step - loss: 0.2587 - accuracy: 0.89
87 - precision: 0.8987 - recall: 0.8987 - val_loss: 0.2484 - val_accuracy: 0.9286 - val_precisio
n: 0.9286 - val_recall: 0.9286 - lr: 1.0000e-04
Epoch 9/10
28/28 [==============================] - 36s 1s/step - loss: 0.2340 - accuracy: 0.91
83 - precision: 0.9183 - recall: 0.9183 - val_loss: 0.1652 - val_accuracy: 0.9598 - val_precisio
n: 0.9598 - val_recall: 0.9598 - lr: 1.0000e-04
Epoch 10/10
28/28 [==============================] - 37s 1s/step - loss: 0.2089 - accuracy: 0.92
29 - precision: 0.9229 - recall: 0.9229 - val_loss: 0.1941 - val_accuracy: 0.9420 - val_precisio
n: 0.9420 - val_recall: 0.9420 - lr: 1.0000e-04

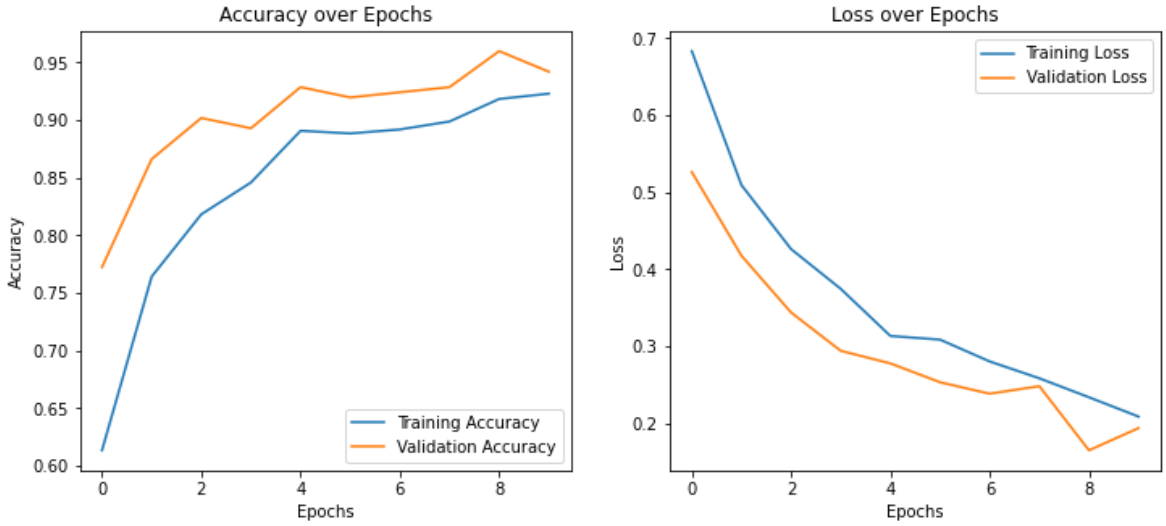


Figure 5-36: Training Fold 2 - NASNetMobile - Accuracy and Loss over Epochs

16/16 [==============================] - 13s 510ms/step

Table 5-15: Training Fold 2 - NASNetMobile - Classification Report

	precision	recall	f1-score	support
Healthy	0.95	0.93	0.94	242
Diseased	0.94	0.95	0.94	242
accuracy			0.94	484
macro avg	0.94	0.94	0.94	484
weighted avg	0.94	0.94	0.94	484

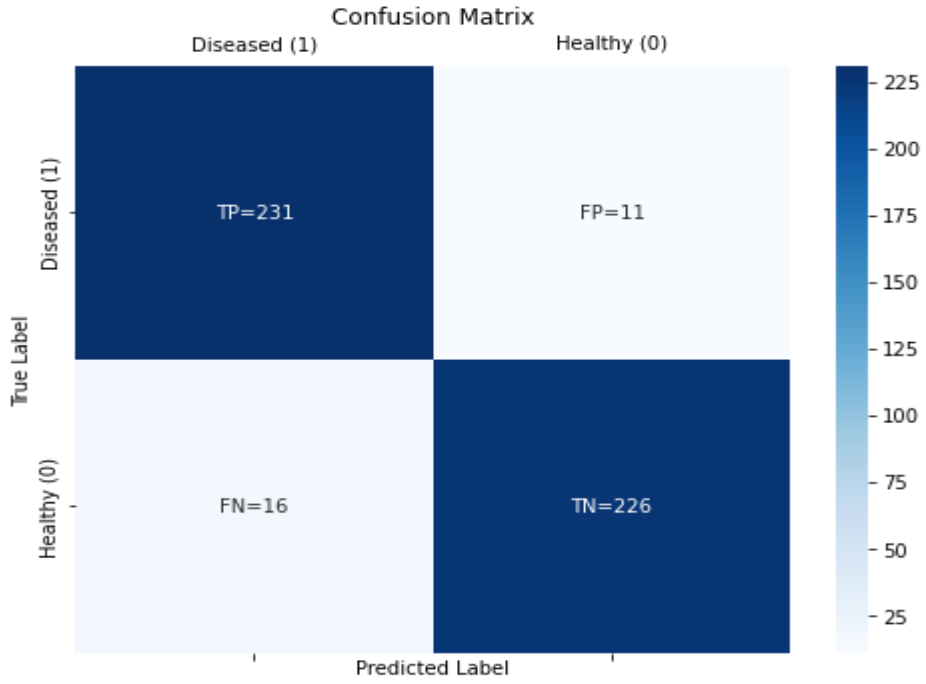


Figure 5-37: Training Fold 2 - NASNetMobile - Confusion Matrix

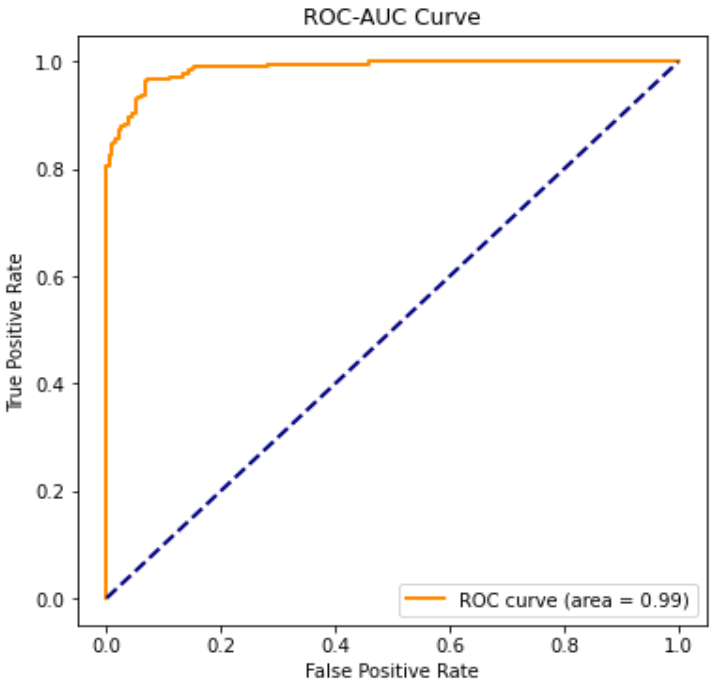


Figure 5-38: Training Fold 2 - NASNetMobile - ROC-AUC Curve

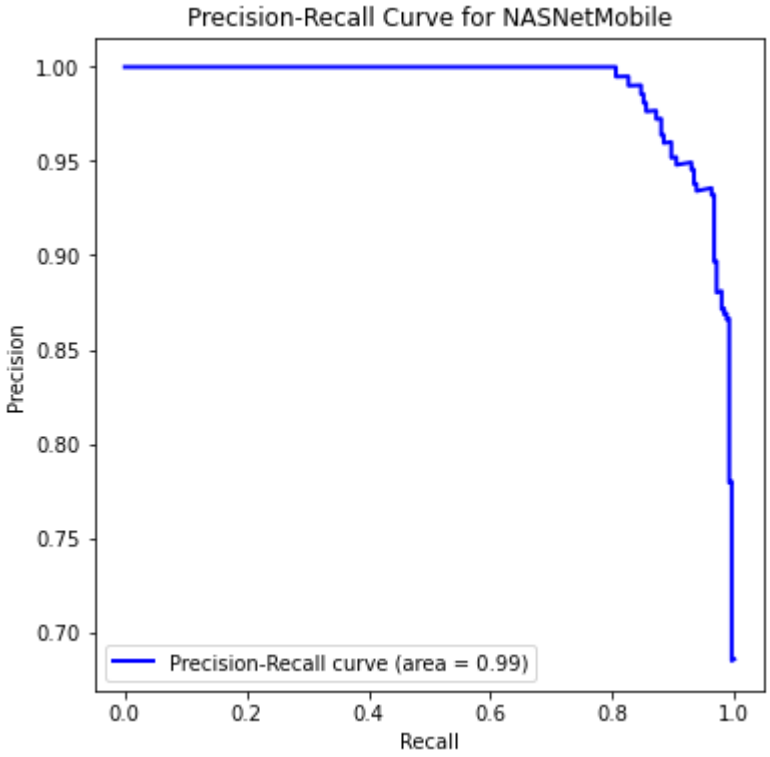


Figure 5-39: Training Fold 2 - NASNetMobile - Precision-Recall Curve

5.3.2.3 NASNetMobile Training Fold 3

Analysis for NASNetMobile's Training Fold 3 is presented as follows:

- **Precision and Recall:** The model shows exceptional precision and recall for the Healthy (0.96) and Diseased (0.90) classes, with a notable recall for Diseased suggesting a sensitivity towards the diseased class as detailed in Table 5-16.
- **F1-Score and Support:** Both categories achieve a balanced f1-score of 0.93, with an equal number of instances for each class, indicating a fair and consistent classification performance as detailed in Table 5-16.
- **Overall Accuracy:** An accuracy of 93% underscores the model's proficiency in correct disease identification as detailed in Table 5-16.
- **Accuracy Over Epochs and Loss Over Epochs:** The trend in the accuracy and loss graphs indicates a stable model that learns effectively without overfitting as visualized in Figure 5-40.
- **Confusion Matrix:** The confusion matrix with a higher TP rate and a lower FN rate points to the model's reliable classification ability as detailed in Figure 5-40.
- **ROC-AUC Curve and Precision-Recall Curve:** Both the ROC-AUC and Precision-Recall curves suggest excellent model performance with areas of 0.99, indicative of a high true positive rate and precision across various thresholds as shown in Figure 5-42 and Figure 5-43.

Results from Code Execution:

Training model: NASNetMobile

```
Epoch 1/10
28/28 [==============================] - 54s 1s/step - loss: 0.6723 - accuracy: 0.60
64 - precision: 0.6064 - recall: 0.6064 - val_loss: 0.5347 - val_accuracy: 0.7812 - val_precisio
n: 0.7812 - val_recall: 0.7812 - lr: 1.0000e-04
Epoch 2/10
28/28 [==============================] - 25s 883ms/step - loss: 0.5029 - accuracy:
0.7560 - precision: 0.7560 - recall: 0.7560 - val_loss: 0.4123 - val_accuracy: 0.8884 - val_pre
cision: 0.8884 - val_recall: 0.8884 - lr: 1.0000e-04
Epoch 3/10
28/28 [==============================] - 23s 835ms/step - loss: 0.4595 - accuracy:
0.7814 - precision: 0.7814 - recall: 0.7814 - val_loss: 0.3510 - val_accuracy: 0.9196 - val_pre
cision: 0.9196 - val_recall: 0.9196 - lr: 1.0000e-04
Epoch 4/10
28/28 [==============================] - 25s 904ms/step - loss: 0.3567 - accuracy:
0.8654 - precision: 0.8654 - recall: 0.8654 - val_loss: 0.2940 - val_accuracy: 0.9286 - val_pre
cision: 0.9286 - val_recall: 0.9286 - lr: 1.0000e-04
Epoch 5/10
```

28/28 [==============================] - 24s 860ms/step - loss: 0.3274 - accuracy: 0.8585 - precision: 0.8585 - recall: 0.8585 - val_loss: 0.2740 - val_accuracy: 0.9062 - val_precision: 0.9062 - val_recall: 0.9062 - lr: 1.0000e-04
Epoch 6/10
28/28 [==============================] - 24s 867ms/step - loss: 0.2957 - accuracy: 0.8906 - precision: 0.8906 - recall: 0.8906 - val_loss: 0.2536 - val_accuracy: 0.8929 - val_precision: 0.8929 - val_recall: 0.8929 - lr: 1.0000e-04
Epoch 7/10
28/28 [==============================] - 24s 857ms/step - loss: 0.2852 - accuracy: 0.8861 - precision: 0.8861 - recall: 0.8861 - val_loss: 0.2469 - val_accuracy: 0.9286 - val_precision: 0.9286 - val_recall: 0.9286 - lr: 1.0000e-04
Epoch 8/10
28/28 [==============================] - 32s 1s/step - loss: 0.2534 - accuracy: 0.9079 - precision: 0.9079 - recall: 0.9079 - val_loss: 0.2170 - val_accuracy: 0.9330 - val_precision: 0.9330 - val_recall: 0.9330 - lr: 1.0000e-04
Epoch 9/10
28/28 [==============================] - 42s 2s/step - loss: 0.2448 - accuracy: 0.9125 - precision: 0.9125 - recall: 0.9125 - val_loss: 0.2082 - val_accuracy: 0.9196 - val_precision: 0.9196 - val_recall: 0.9196 - lr: 1.0000e-04
Epoch 10/10
28/28 [==============================] - 30s 1s/step - loss: 0.2136 - accuracy: 0.9287 - precision: 0.9287 - recall: 0.9287 - val_loss: 0.2051 - val_accuracy: 0.9286 - val_precision: 0.9286 - val_recall: 0.9286 - lr: 1.0000e-04

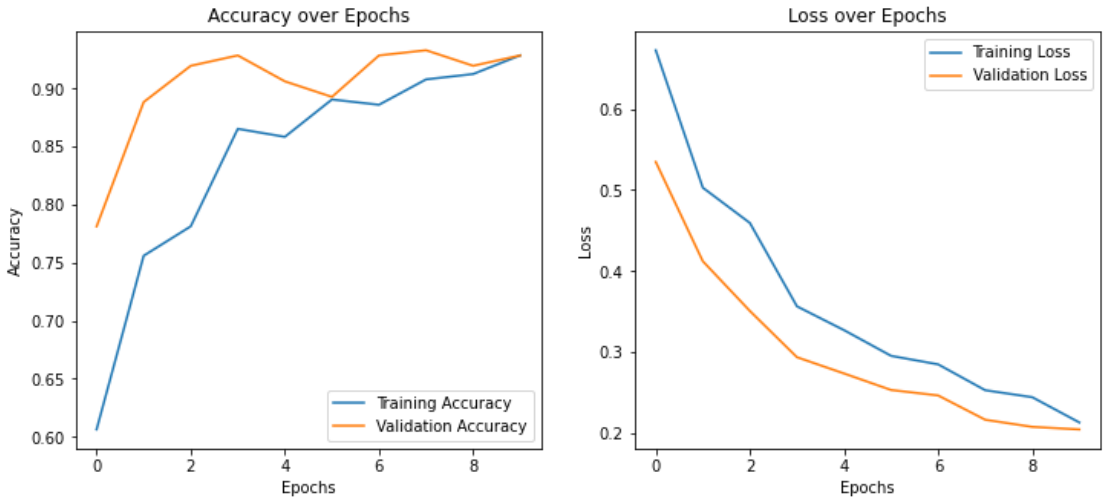


Figure 5-40: Training Fold 3 - NASNetMobile - Accuracy and Loss Over Epochs

16/16 [==============================] - 12s 432ms/step

Table 5-16: Training Fold 3 - NASNetMobile - Classification Report

	precision	recall	f1-score	support
Healthy	0.96	0.90	0.93	242
Diseased	0.90	0.96	0.93	242
accuracy			0.93	484
macro avg	0.93	0.93	0.93	484
weighted avg	0.93	0.93	0.93	484

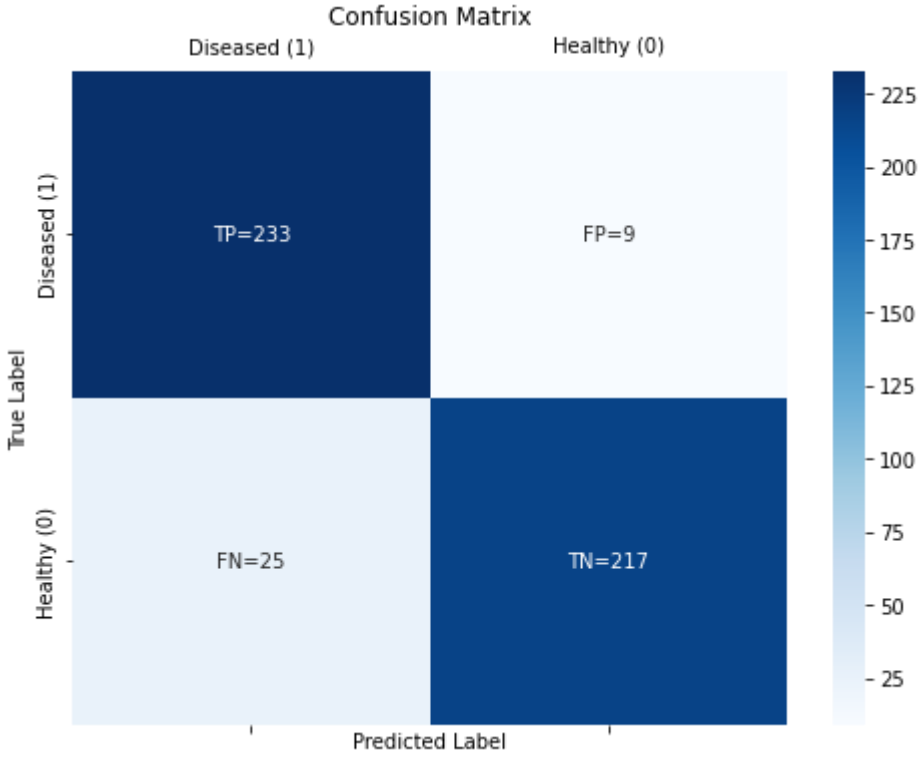


Figure 5-41: Training Fold 3 - NASNetMobile - Confusion Matrix

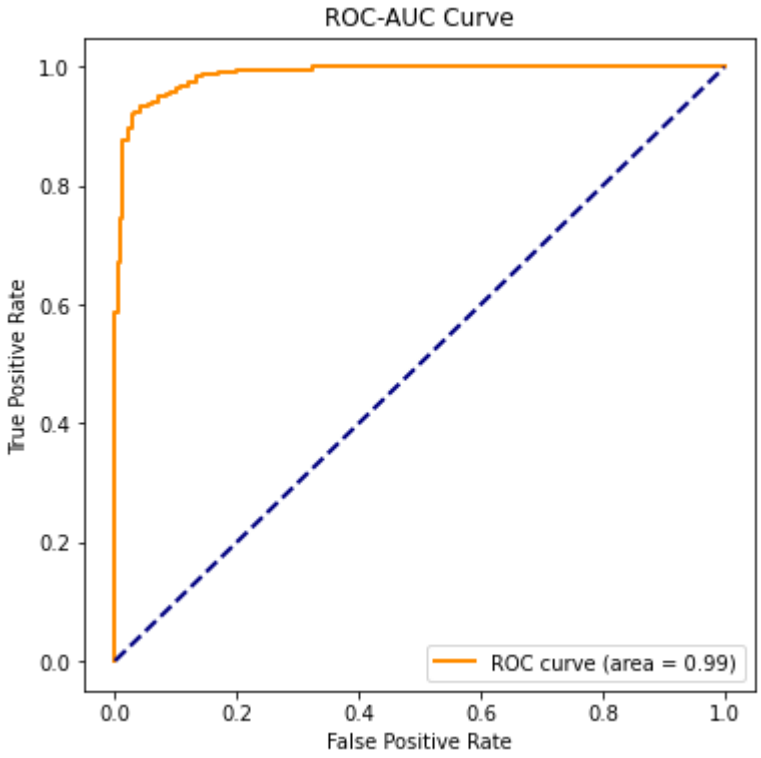


Figure 5-42: Training Fold 3 - NASNetMobile - ROC-AUC Curve

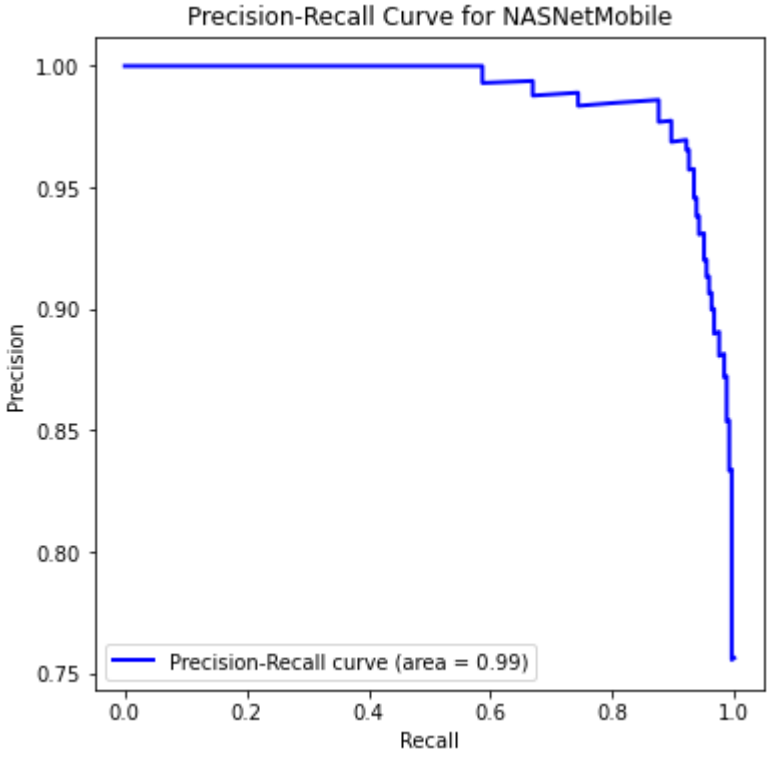


Figure 5-43: Training Fold 3 - NASNetMobile - Precision Recall Curve

5.3.2.4 NASNetMobile Training Fold 4

Analysis for NASNetMobile's Training Fold 4 is presented as follows:

- **Precision and Recall:** The model continues to excel with high precision (0.96 Healthy, 0.92 Diseased) and recall (0.92 Healthy, 0.96 Diseased), as per Table 5-17.
- **F1-Score and Support:** Equally strong f1-scores (0.94 for both classes) and equal support numbers (242 per class) show the model's consistent performance as detailed in Table 5-17.
- **Overall Accuracy:** An overall accuracy of 94% reflects the model's effectiveness in distinguishing between plant health states as per Table 5-17.
- **Accuracy Over Epochs:** The graph illustrates a steady increase in validation accuracy compared to training accuracy, suggesting a well-tuned balance as visualized in Figure 5-44.
- **Loss Over Epochs:** The loss graph depicts a gradual convergence of training and validation loss, indicating good model generalization as visualized in Figure 5-44.
- **Confusion Matrix:** A small number of false positives and false negatives imply high classification accuracy as visualized in Figure 5-45.
- **ROC-AUC Curve:** The ROC curve with an AUC of 0.99 signifies excellent discriminatory power as shown in Figure 5-46.
- **Precision-Recall Curve:** The model's consistent precision across various class thresholds is evidenced by the substantial area (0.99) beneath the precision-recall curve, as depicted in Figure 5-47.

Results from Code Execution:

Training model: NASNetMobile

```
Epoch 1/10
28/28 [==============================] - 53s 1s/step - loss: 0.7332 - accuracy: 0.59
38 - precision: 0.5938 - recall: 0.5938 - val_loss: 0.5051 - val_accuracy: 0.8214 - val_precisio
n: 0.8214 - val_recall: 0.8214 - lr: 1.0000e-04
Epoch 2/10
28/28 [==============================] - 36s 1s/step - loss: 0.5708 - accuracy: 0.69
39 - precision: 0.6939 - recall: 0.6939 - val_loss: 0.4169 - val_accuracy: 0.8839 - val_precisio
n: 0.8839 - val_recall: 0.8839 - lr: 1.0000e-04
Epoch 3/10
28/28 [==============================] - 35s 1s/step - loss: 0.4666 - accuracy: 0.79
63 - precision: 0.7963 - recall: 0.7963 - val_loss: 0.3514 - val_accuracy: 0.9107 - val_precisio
n: 0.9107 - val_recall: 0.9107 - lr: 1.0000e-04
Epoch 4/10
```

28/28 [==============================] - 36s 1s/step - loss: 0.3975 - accuracy: 0.8285 - precision: 0.8285 - recall: 0.8285 - val_loss: 0.3019 - val_accuracy: 0.9107 - val_precision: 0.9107 - val_recall: 0.9107 - lr: 1.0000e-04
Epoch 5/10
28/28 [==============================] - 36s 1s/step - loss: 0.3607 - accuracy: 0.8769 - precision: 0.8769 - recall: 0.8769 - val_loss: 0.2770 - val_accuracy: 0.9241 - val_precision: 0.9241 - val_recall: 0.9241 - lr: 1.0000e-04
Epoch 6/10
28/28 [==============================] - 42s 2s/step - loss: 0.3005 - accuracy: 0.8849 - precision: 0.8849 - recall: 0.8849 - val_loss: 0.2215 - val_accuracy: 0.9598 - val_precision: 0.9598 - val_recall: 0.9598 - lr: 1.0000e-04
Epoch 7/10
28/28 [==============================] - 39s 1s/step - loss: 0.2945 - accuracy: 0.8872 - precision: 0.8872 - recall: 0.8872 - val_loss: 0.2181 - val_accuracy: 0.9330 - val_precision: 0.9330 - val_recall: 0.9330 - lr: 1.0000e-04
Epoch 8/10
28/28 [==============================] - 28s 983ms/step - loss: 0.2711 - accuracy: 0.9022 - precision: 0.9022 - recall: 0.9022 - val_loss: 0.2172 - val_accuracy: 0.9375 - val_precision: 0.9375 - val_recall: 0.9375 - lr: 1.0000e-04
Epoch 9/10
28/28 [==============================] - 24s 849ms/step - loss: 0.2524 - accuracy: 0.9010 - precision: 0.9010 - recall: 0.9010 - val_loss: 0.1825 - val_accuracy: 0.9598 - val_precision: 0.9598 - val_recall: 0.9598 - lr: 1.0000e-04
Epoch 10/10
28/28 [==============================] - 24s 857ms/step - loss: 0.2394 - accuracy: 0.9068 - precision: 0.9068 - recall: 0.9068 - val_loss: 0.1706 - val_accuracy: 0.9598 - val_precision: 0.9598 - val_recall: 0.9598 - lr: 1.0000e-04

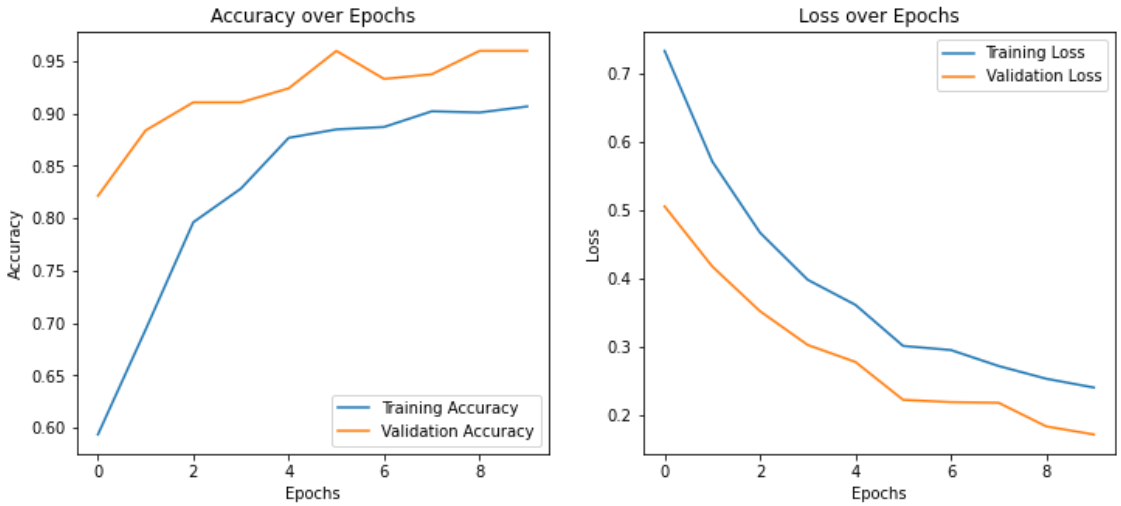


Figure 5-44: Training Fold 4 - NASNetMobile - Accuracy and Loss Over Epochs

16/16 [==============================] - 12s 422ms/step

Table 5-17: Training Fold 4 - NASNetMobile - Classification Report

	precision	recall	f1-score	support
Healthy	0.96	0.92	0.94	242
Diseased	0.92	0.96	0.94	242
accuracy			0.94	484
macro avg	0.94	0.94	0.94	484
weighted avg	0.94	0.94	0.94	484

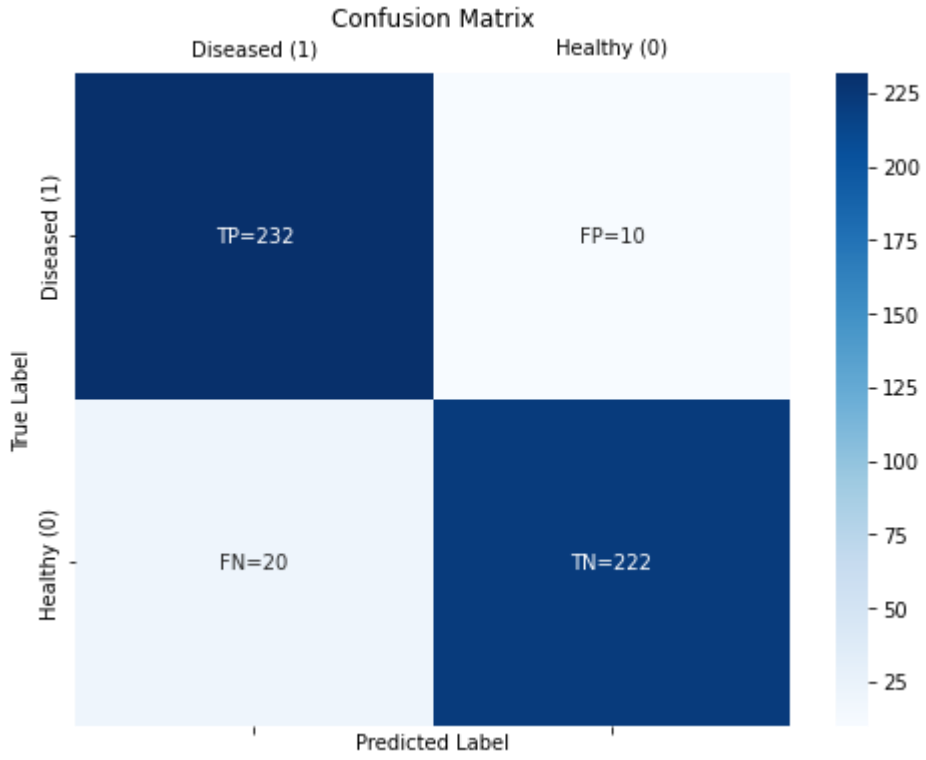


Figure 5-45: Training Fold 4 - NASNetMobile - Confusion Matrix

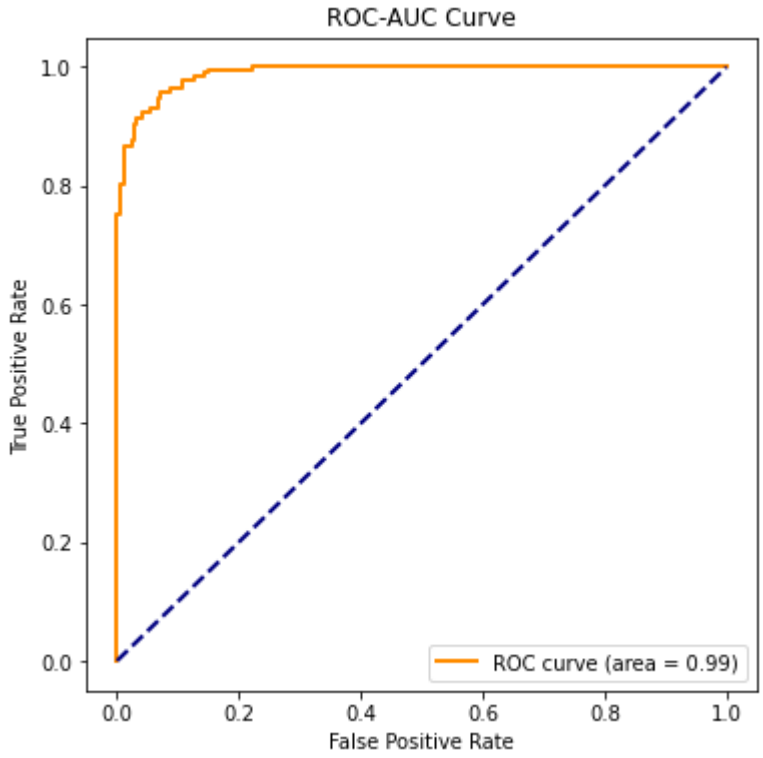


Figure 5-46: Training Fold 4 - NASNetMobile - ROC-AUC Curve

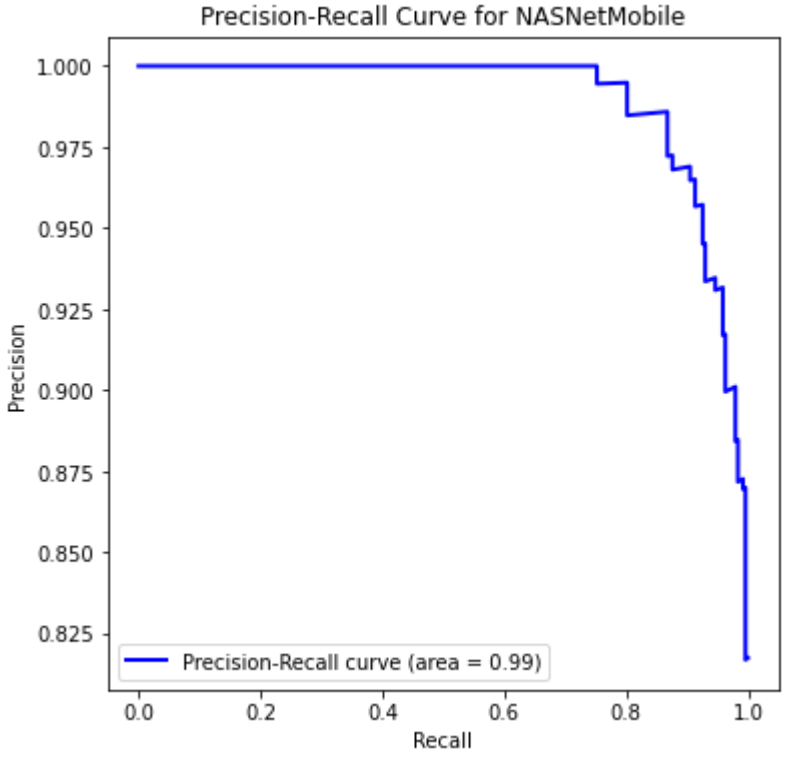


Figure 5-47: Training Fold 4 - NASNetMobile - Precision-Recall Curve

5.3.2.5 NASNetMobile Training Fold 5

Analysis for NASNetMobile's Training Fold 5 is presented as follows:

- **Precision and Recall:** The model demonstrates high precision (0.93 Healthy, 0.91 Diseased) and recall (0.91 Healthy, 0.93 Diseased), as indicated in Table 5-18.
- **F1-Score and Support:** Robust f1-scores (0.92 for both classes) and balanced support (242 per class) underscore the model's consistent output as detailed in Table 5-18.
- **Overall Accuracy:** An overall accuracy of 92% showcases the model's proficiency in differentiating plant health conditions as reported in Table 5-18.
- **Accuracy Over Epochs:** The accuracy graph indicates a plateauing of both training and validation accuracy, suggesting that the model is neither overfitting nor underfitting as visualized in Figure 5-48.
- **Loss Over Epochs:** The loss graph shows a decrease in both training and validation loss, which is indicative of effective learning and generalization as visualized in Figure 5-48.
- **Confusion Matrix:** A minimal number of false positives (16) and false negatives (21) highlight the model's classification accuracy as visualized in Figure 5-49.
- **ROC-AUC Curve:** The ROC curve with an AUC of 0.98 denotes excellent model discrimination capabilities as seen in Figure 5-50.
- **Precision-Recall Curve:** A high area under the curve (0.98) in the precision-recall graph reaffirms the model's precision across varying thresholds as seen in Figure 5-51.

Results from Code Execution:

Training model: NASNetMobile

```
Epoch 1/10
28/28 [==============================] - 50s 1s/step - loss: 0.6839 - accuracy: 0.59
15 - precision: 0.5915 - recall: 0.5915 - val_loss: 0.4891 - val_accuracy: 0.8795 - val_precisio
n: 0.8795 - val_recall: 0.8795 - lr: 1.0000e-04
Epoch 2/10
28/28 [==============================] - 35s 1s/step - loss: 0.5108 - accuracy: 0.75
26 - precision: 0.7526 - recall: 0.7526 - val_loss: 0.4024 - val_accuracy: 0.8750 - val_precisio
n: 0.8750 - val_recall: 0.8750 - lr: 1.0000e-04
Epoch 3/10
28/28 [==============================] - 37s 1s/step - loss: 0.3934 - accuracy: 0.84
23 - precision: 0.8423 - recall: 0.8423 - val_loss: 0.3464 - val_accuracy: 0.8616 - val_precisio
n: 0.8616 - val_recall: 0.8616 - lr: 1.0000e-04
Epoch 4/10
28/28 [==============================] - 36s 1s/step - loss: 0.3526 - accuracy: 0.87
00 - precision: 0.8700 - recall: 0.8700 - val_loss: 0.2903 - val_accuracy: 0.9241 - val_precisio
n: 0.9241 - val_recall: 0.9241 - lr: 1.0000e-04
```

Epoch 5/10
28/28 [==============================] - 35s 1s/step - loss: 0.3114 - accuracy: 0.88
72 - precision: 0.8872 - recall: 0.8872 - val_loss: 0.2643 - val_accuracy: 0.9152 - val_precisio
n: 0.9152 - val_recall: 0.9152 - lr: 1.0000e-04
Epoch 6/10
28/28 [==============================] - 36s 1s/step - loss: 0.2905 - accuracy: 0.88
95 - precision: 0.8895 - recall: 0.8895 - val_loss: 0.2375 - val_accuracy: 0.9196 - val_precisio
n: 0.9196 - val_recall: 0.9196 - lr: 1.0000e-04
Epoch 7/10
28/28 [==============================] - 36s 1s/step - loss: 0.2516 - accuracy: 0.90
91 - precision: 0.9091 - recall: 0.9091 - val_loss: 0.2310 - val_accuracy: 0.9107 - val_precisio
n: 0.9107 - val_recall: 0.9107 - lr: 1.0000e-04
Epoch 8/10
28/28 [==============================] - 36s 1s/step - loss: 0.2464 - accuracy: 0.91
71 - precision: 0.9171 - recall: 0.9171 - val_loss: 0.2085 - val_accuracy: 0.9241 - val_precisio
n: 0.9241 - val_recall: 0.9241 - lr: 1.0000e-04
Epoch 9/10
28/28 [==============================] - 35s 1s/step - loss: 0.2326 - accuracy: 0.91
83 - precision: 0.9183 - recall: 0.9183 - val_loss: 0.2089 - val_accuracy: 0.9241 - val_precisio
n: 0.9241 - val_recall: 0.9241 - lr: 1.0000e-04
Epoch 10/10
28/28 [==============================] - 36s 1s/step - loss: 0.2088 - accuracy: 0.91
94 - precision: 0.9194 - recall: 0.9194 - val_loss: 0.1979 - val_accuracy: 0.9375 - val_precisio
n: 0.9375 - val_recall: 0.9375 - lr: 1.0000e-04

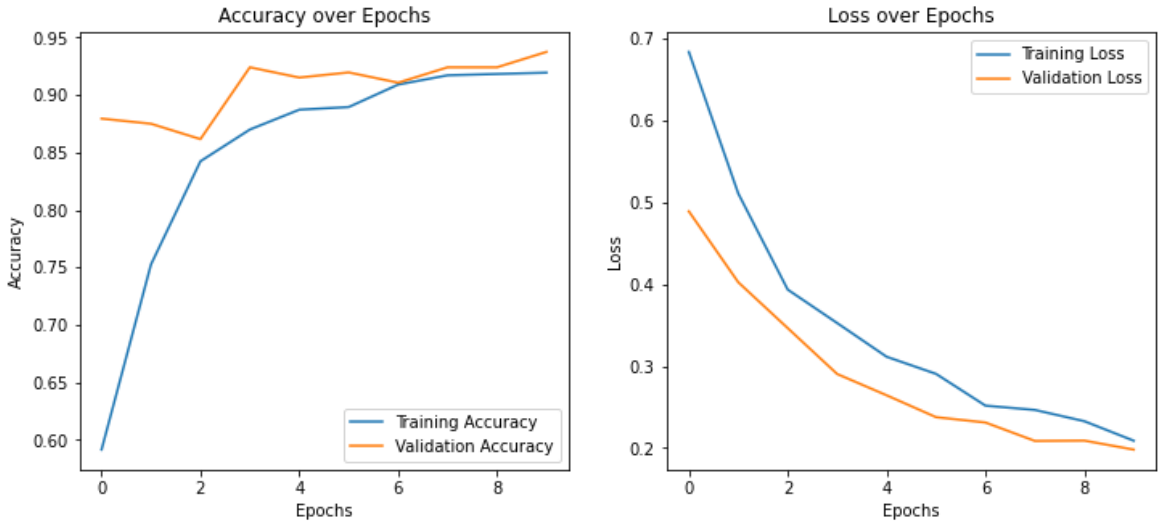


Figure 5-48: Training Fold 5 - NASNetMobile - Accuracy and Loss Over Epochs

16/16 [==============================] - 13s 533ms/step

Table 5-18: Training Fold 5 - NASNetMobile - Classification Report

	precision	recall	f1-score	support
Healthy	0.93	0.91	0.92	242
Diseased	0.91	0.93	0.92	242
accuracy			0.92	484
macro avg	0.92	0.92	0.92	484
weighted avg	0.92	0.92	0.92	484

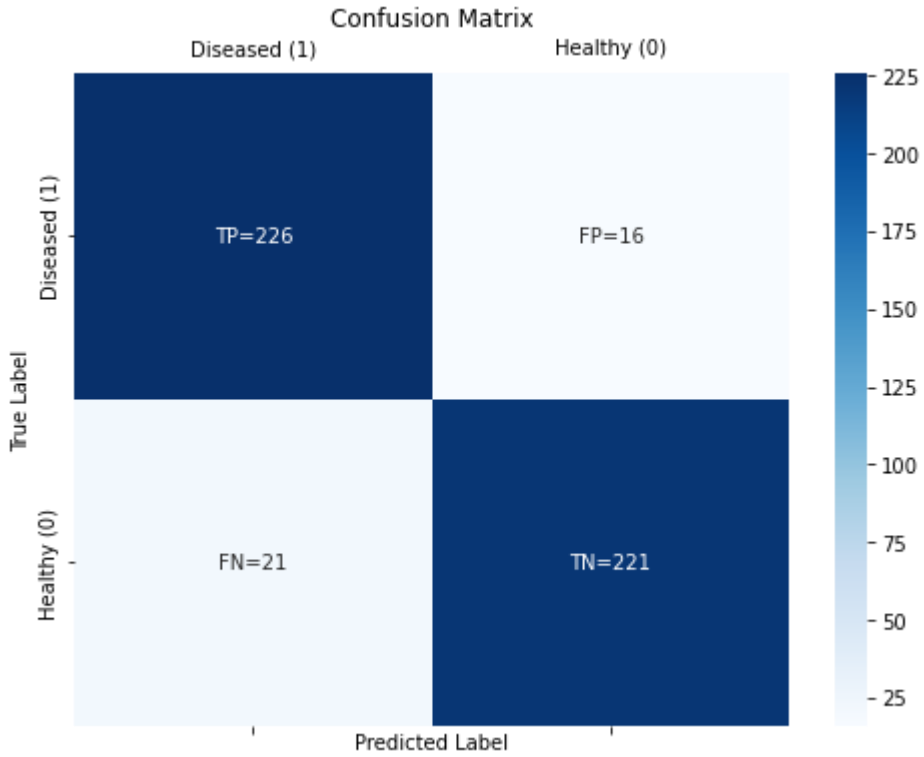


Figure 5-49: Training Fold 5 - NASNetMobile - Confusion Matrix

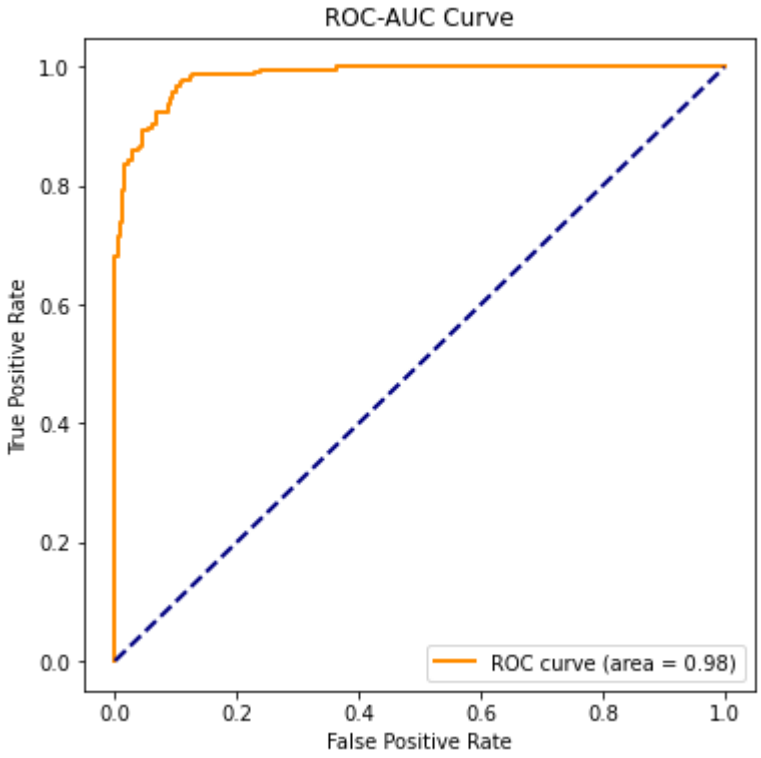


Figure 5-50: Training Fold 5 - NASNetMobile - ROC-AUC Curve

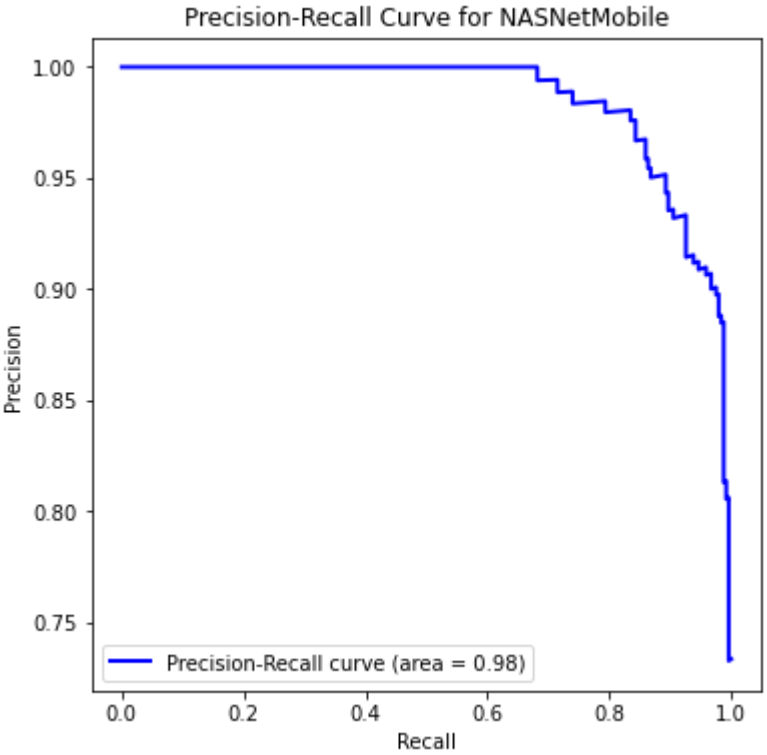


Figure 5-51: Training Fold 5 - NASNetMobile - Precision-Recall Curve

5.3.2.6 Strengths and Limitations of NASNetMobile

After analysing five training folds of NASNetMobile, below are its strengths and limitations:

Strengths:

- **High Precision and Recall**: NASNetMobile exhibits high precision and recall across different folds. For instance, in Training Fold 1, it demonstrated a precision of 0.96 for Healthy and 0.92 for Diseased classes, with recall slightly favoring Diseased at 0.96 over Healthy's 0.91. Such metrics were consistent across other folds, indicating the model's reliability in identifying both healthy and diseased plant conditions with minimal error.
- **Overall Accuracy:** The model achieved notable overall accuracies, ranging from 92% to 94% across different folds, which underscores its effectiveness in plant health condition differentiation.
- **Learning Stability and Generalization:** The model showed efficient learning and generalization capabilities, as indicated by the accuracy and loss over epochs. For instance, in Training Fold 3, the model's performance improved consistently without signs of overfitting or underfitting.
- **Excellent Discriminative Ability:** The ROC-AUC and Precision-Recall curves across folds demonstrated excellent discriminative ability and precision across various recall levels, with areas under the curve close to 0.99, indicating the model's strong capability in distinguishing between classes.
- **Balanced Classification Performance:** F1-scores across folds were balanced, indicating a stable precision-recall relationship. This balanced performance is crucial for practical applications where both false positives and false negatives can have significant implications.

Limitations:

- **Slight Variability in Performance Metrics:** While overall high, there is slight variability in precision, recall, and overall accuracy across different folds. For example, the overall accuracy slightly dropped in Training Fold 5 to 92%. This variability might suggest room for improvement in model robustness across diverse datasets.
- **Potential Overfitting Risks:** Despite the general indication of stable learning, the detailed training and validation accuracy and loss curves (e.g., in Training Folds 1 and 4) should be scrutinized for any signs of overfitting, especially in cases where the validation accuracy plateaus or decreases over time.

- **Limited Scope of Evaluation:** The evaluation mainly concentrates on precision, recall, F1-scores, and total accuracy. Additional measures like specificity, negative predictive value, or analysis of performance in multi-class scenarios could provide a more comprehensive understanding of the model's strengths and limitations.
- **Dependence on Data Quality and Quantity:** The model's performance, especially its discriminative ability, may be heavily influenced by the quality and representativeness of the training data. Any limitations in the dataset, such as imbalanced classes or lack of diversity in plant disease representation, could potentially skew the model's learning and generalization capabilities.
- **Computational Resources:** Given the sophisticated architecture of NASNetMobile, its deployment in real-world applications, especially in resource-constrained environments, may require significant computational resources, which could limit its accessibility and scalability.

In conclusion, NASNetMobile demonstrates substantial strengths in terms of precision, recall, and overall accuracy, making it a promising tool for plant disease detection. However, attention should be given to its variability in performance across different datasets and potential overfitting risks. Further evaluation and optimization could enhance its robustness and applicability in diverse agricultural settings.

5.3.3 Comparative Analysis of MobileNetV2 and NASNetMobile

This section offers a comparison between NASNetMobile and MobileNetV2, two advanced deep learning structures utilized in the specialized field of detecting plant diseases. Both models leverage advanced convolutional neural network (CNN) designs, offering high-dimensional data processing with exceptional accuracy and efficiency. The evaluation spans a comprehensive set of metrics, including accuracy, precision, recall, and F1-scores, alongside diagnostic tools like confusion matrices, ROC-AUC, and precision-recall curves, to elucidate each model's diagnostic performance.

5.3.3.1 Preferred Model: MobileNetV2 Among DL Architectures

MobileNetV2 is favored over NASNetMobile when comparing their performance in the context of deep learning architectures for plant disease detection. This preference is substantiated by the analysis of key performance indicators across five training folds. MobileNetV2 consistently

demonstrated high precision and recall rates across both classes (Healthy and Diseased), with values often reaching or exceeding 0.98, showcasing its superior accuracy in classifying plant health conditions. Moreover, MobileNetV2 maintained an impressive overall accuracy, often nearing or achieving 98%, further underscoring its reliability in differentiating between healthy and diseased plants.

In contrast, NASNetMobile, while exhibiting commendable performance, achieved slightly lower metrics in comparison. For instance, during its first training fold, NASNetMobile displayed precision rates of 0.96 for Healthy and 0.92 for Diseased classes, with a corresponding recall of 0.91 for Healthy and 0.96 for Diseased classes. This resulted in an overall accuracy of 94%, which, although notable, falls short of the nearly perfect scores observed with MobileNetV2. The trend continues across subsequent training folds, where NASNetMobile's precision and recall rates hover around 0.93 to 0.96, maintaining an overall accuracy within the range of 92% to 94%.

These comparative insights reveal MobileNetV2's enhanced effectiveness in processing complex image data for the nuanced task of plant disease diagnosis in the field of precision agriculture. The consistent superiority in key performance indicators not only highlights MobileNetV2's advanced capability in accurate disease classification but also underscores its potential to significantly contribute to sustainable and efficient agricultural practices by enabling early and accurate disease detection.

5.4 Discussion

In this chapter, the performance analysis of seven machine learning (ML) models and two deep learning (DL) models revealed insightful trends and outcomes in the context of plant disease detection. The discussion section delves into the intricacies of these findings, offering a nuanced understanding of the models' capabilities and limitations.

5.4.1 Machine Learning Models: A Synthesis of Robustness and Efficiency

The analysis demonstrated the effectiveness of ensemble models like Random Forest, which outperformed its counterparts with an accuracy of 95.21%. This highlights the robustness and generalizability of ensemble methods, particularly in managing complex data distributions common in plant pathology imaging. However, the reliance on handcrafted features and

extensive data preprocessing required by ML models might limit their applicability in dynamic agricultural environments where conditions such as lighting and background can vary significantly.

5.4.2 Deep Learning Models: Pioneering Advanced Feature Processing

Deep learning models, particularly MobileNetV2, showcased superior performance in processing raw image data, achieving precision and recall rates above 98%. MobileNetV2's architecture, optimized for efficiency, provides a balance between computational demand and predictive performance, making it suitable for real-time applications in resource-constrained settings. Nevertheless, the necessity for large annotated datasets and significant computational resources for training poses challenges to the widespread adoption of DL models.

5.5 Chapter Summary

The comprehensive analysis in Chapter 5 not only underscored the strengths of ensemble methods and the reliability of statistical ML models but also highlighted the transformative potential of DL in agricultural technology. With the RF model leading the ML category with its high test accuracy of 95.21% and MobileNetV2 achieving precision and recall rates above 98% shows a promising results in DL, this chapter contributes valuable insights to the broader discourse on leveraging computational models for more efficient, accurate, and sustainable agricultural practices. The meticulous examination of these models paves the way for future advancements in plant disease detection, promising to enhance disease management strategies and contribute to the revolution in precision agriculture.

Chapter 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 Introduction

This chapter synthesizes the research findings, offering a comparative analysis of the evaluated ML and DL models in the context of apple leaf disease detection. It revisits the research questions, gauging the study's success in meeting its objectives, and discusses the limitations encountered alongside future recommendations. This chapter crystallizes the implications of the study's results, providing actionable insights for advancing the field of agricultural technology and plant disease management.

6.2 Comparative Analysis of ML and DL Models for Apple Leaf Disease Detection

In the quest to revolutionize apple leaf disease detection, a meticulous comparison between Machine Learning (ML) and Deep Learning (DL) models unveils insights pivotal to agricultural technology. This analysis discerns the efficacy of various models in accurately diagnosing diseases, setting a foundation for the integration of advanced computational models in precision agriculture.

6.2.1 Machine Learning Models Performance Overview

Among the seven-machine learning (ML) models evaluated, the Random Forest (RF) model emerged as the standout performer with a test accuracy of 95.21%. This model, known for its ensemble learning approach that combines multiple decision trees, showcased remarkable robustness and efficiency in handling complex data distributions typical in plant pathology imaging. The RF model's performance was underscored by its perfect training accuracy, highlighting its ability to generalize well to new, unseen data while mitigating the risk of overfitting—a common challenge in machine learning models.

6.2.2 Deep Learning Models Performance Overview

In the realm of deep learning (DL) models, MobileNetV2 was identified as the superior model, demonstrating exceptional performance metrics across the board. It consistently achieved high precision and recall rates, often surpassing 98%, signifying its adeptness at accurately classifying healthy and diseased plant conditions. Its overall accuracy frequently neared or reached 98%, solidifying its reliability in differentiating between the two classes. In

comparison, NASNetMobile, while still performing commendably, fell slightly short with its precision, recall, and overall accuracy metrics slightly lower than those of MobileNetV2.

6.2.3 Comparative Analysis

When comparing the best-performing ML model (RF) with the leading DL model (MobileNetV2), it becomes evident that MobileNetV2 holds a slight edge, particularly in the precision and recall metrics. The superior performance of MobileNetV2 can be attributed to its deep learning architecture, which is more adept at processing raw image data and extracting intricate features essential for accurate disease classification. Furthermore, MobileNetV2's architecture, optimized for efficiency, strikes a balance between computational demands and predictive performance, making it suitable for real-time applications even in resource-constrained settings.

6.2.4 Conclusion:

Based on the comparative analysis, MobileNetV2 is chosen as the best model for detecting diseases in apple leaves. Its advanced capability in accurately classifying plant health conditions, coupled with its efficiency and reliability across various metrics, underscores its potential to significantly contribute to sustainable and efficient agricultural practices. The adoption of MobileNetV2 could facilitate early and accurate disease detection, thereby enhancing disease management strategies and revolutionizing precision agriculture.

The selection of MobileNetV2 as the best model is grounded in its consistent superiority in key performance indicators and its suitability for practical deployment in real-world agricultural settings, making it a valuable tool in advancing the field of agricultural technology.

6.3 Research Questions Revisited

This section revisits the research questions, providing a reflective analysis of the findings in relation to the study's objectives, offering insights into the effectiveness of the explored ML and DL models in addressing the key concerns in apple leaf disease detection.

6.3.1 Research Question 1

How do various Machine Learning models (such as Logistic Regression, Decision Trees, K-Nearest Neighbors, Random Forest, Naive Bayes, SVM) compare in terms of accuracy and effectiveness in detecting diseases in apple plant leaves?

Answer:

This thesis investigates seven conventional ML models for identifying diseases in apple leaf plants as part of its examination of Machine Learning (ML) techniques: Logistic Regression (LR), Linear Discriminant Analysis (LDA), K-Nearest Neighbors (KNN), Classification and Regression Trees (CART), Random Forest (RF), Naive Bayes (NB), and Support Vector Machine (SVM). The efficacy of these models is meticulously evaluated using accuracy, precision, and recall metrics, underscoring their capabilities and potential constraints within agricultural technological applications. Particularly noteworthy is the Random Forest model for its outstanding accuracy and well-proportioned precision-recall metrics, proving its applicability in agricultural diagnostics. The following table provides a comprehensive comparison of these models, facilitating an understanding of their relative performances and guiding future research directions in plant disease detection.

Table 6-1: Comparative Analysis of ML Models for Apple Leaf Disease Detection

Model	Accuracy	Precision (Diseased)	Recall (Diseased)	Key Points
LR	90.00%	0.91	0.89	Reliable in positive identification, slight recall disparity
LDA	89.79%	0.93	0.86	High precision, improvement needed in diseased recall
KNN	91.46%	0.92	0.90	Effective identification with low false positives
CART	90.62%	0.90	0.92	Balanced classification, minor misclassification errors
RF	95.21%	0.95	0.96	Superior accuracy, excellent class differentiation
NB	85.83%	0.97	0.74	High true positive rate, high false positive rate

SVM	90.21%	0.91	0.89	Balanced classification, scope for misclassification reduction

In conclusion, as illustrated in the Table 6-1, the Random Forest model emerges as the most effective, distinguished by its high accuracy and balanced metrics. This model's proficiency in minimizing misdiagnoses while efficiently classifying diseased and healthy instances positions it as an optimal choice for deployment in precision agriculture. Further optimization and real-world validation of this model are recommended to enhance disease management strategies in apple orchards.

6.3.2 Research Question 2

What is the comparative performance of Deep Learning models (specifically CNN with NASNetMobile and MobileNetV2 architectures) in leaf disease detection, and how do they enhance detection accuracy?

Answer:

The examination of Deep Learning models, particularly CNN architectures NASNetMobile and MobileNetV2, showcases distinct performance metrics that highlight their roles in enhancing the accuracy of leaf disease detection. MobileNetV2, in particular, stands out for its superior performance, as evidenced by its consistently high precision and recall rates across both Healthy and Diseased classes, often reaching or exceeding 0.98. This high level of accuracy is indicative of MobileNetV2's proficiency in correctly identifying and classifying plant health conditions, significantly reducing the likelihood of false positives and false negatives. The following table provides a comprehensive comparison of these models, facilitating an understanding of their relative performances and guiding future research directions in plant disease detection.

Table 6-2: Comparative Performance of DL Models for Leaf Disease Detection

Feature	**NASNetMobile**	**MobileNetV2**	**Remarks**
Design	Advanced CNN	Advanced CNN	Both models use sophisticated CNN designs for high-dimensional data processing.
Accuracy	Overall accuracy ranges	Often nearing or achieving 98%	MobileNetV2 demonstrates superior accuracy, making it more

	from 92% to 94%		reliable in distinguishing healthy from diseased plants.
Precision (Healthy)	Up to 0.96	Consistently >= 0.98	MobileNetV2 shows higher precision, indicating fewer false positives in healthy plant detection.
Precision (Diseased)	Up to 0.92	Consistently >= 0.98	MobileNetV2's precision for diseased plants suggests high reliability in disease detection.
Recall (Healthy)	Up to 0.91	Consistently >= 0.98	MobileNetV2 has a higher recall for healthy plants, ensuring fewer false negatives.
Recall (Diseased)	Up to 0.96	Consistently >= 0.98	MobileNetV2 demonstrates higher recall for diseased plants, crucial for early disease detection.
Diagnostic Tools	Confusion matrices, ROC-AUC, Precision-Recall curves	Confusion matrices, ROC-AUC, Precision-Recall curves	Both models utilize comprehensive diagnostic tools to assess performance.
Preferred Model		MobileNetV2	MobileNetV2 is preferred for its consistently superior performance across key metrics.

Enhancement of Detection Accuracy:

MobileNetV2's advanced capability in processing complex image data significantly enhances disease detection accuracy. The model's architecture, optimized for efficiency, provides a balance between computational demand and predictive performance, making it suitable for real-time applications in resource-constrained settings. The high precision and recall rates ensure that the model can accurately classify plant health conditions, enabling early detection and intervention for plant diseases. This not only contributes to the sustainability of agricultural practices but also to the overall efficiency by reducing the spread of diseases and potential crop losses.

In conclusion, the comparative analysis reveals MobileNetV2's superiority in terms of performance metrics, highlighting its role in enhancing the accuracy of leaf disease detection through advanced image processing and classification capabilities. This model's efficiency and reliability make it a valuable tool for advancing precision agriculture and improving disease management strategies.

6.3.3 Research Question 3

How do the image processing and feature extraction methods employed affect the efficiency and accuracy of both ML and DL models in identifying specific leaf diseases?

Answer:

The efficiency and accuracy of ML and DL models in identifying specific leaf diseases are significantly influenced by the employed image processing and feature extraction methods. For ML models, preprocessing steps like resizing, color format conversion from BGR to RGB, and conversion to the HSV color space are crucial. These steps ensure the models receive standardized and meaningful input data, highlighting features relevant for disease identification.

DL models utilize architectures like MobileNetV2 and NASNetMobile, incorporating advanced feature extraction layers capable of capturing intricate patterns in leaf images. These models benefit from augmentation techniques, such as rotation and flipping, which increase the diversity of the training data, leading to more robust models capable of generalizing better to unseen data.

Both approaches, through their respective preprocessing and augmentation techniques, aim to enhance model performance. ML models rely on manual feature extraction and preprocessing to ensure data uniformity, while DL models automatically learn feature representations, further enhanced by data augmentation, leading to potentially higher accuracy in identifying leaf diseases.

6.3.4 Research Question 4

What are the limitations and challenges in current plant disease detection methods, and how does this research aim to address them through the integration of advanced ML and DL techniques?

Answer:

The integration of advanced Machine Learning (ML) and Deep Learning (DL) techniques in this research aims to address several limitations and challenges identified in the current methodologies for plant disease detection, as highlighted in the Literature Review of Chapter 2. Among the key challenges are the difficulty in distinguishing between diseases with similar symptoms, the risk of overfitting due to the complex nature of plant diseases and variability in symptoms, and the need for larger and more diverse datasets to ensure the generalizability of models across different global plant conditions. Furthermore, the training of deep learning models is often hampered by complex field data and hardware constraints, while the quality and diversity of datasets significantly impact the performance of detection systems. Efficient and effective preprocessing and feature extraction also present considerable challenges, given the varying field conditions and disease manifestations.

In response to these challenges, this thesis adopts an innovative approach by conducting a comprehensive comparative analysis of a range of ML algorithms and DL models, particularly focusing on their applicability and effectiveness in real-world agricultural scenarios for apple leaf diseases. Notably, the research incorporates contemporary DL architectures like MobileNetV2 and NASNetMobile, which have not been widely utilized in the reviewed literature, indicating a novel approach within this domain. This integration underscores the thesis's commitment to exploring advanced computational models to enhance the accuracy, efficiency, and adaptability of plant disease detection systems. Such systems are poised to better manage the complexities of real-world agricultural environments, significantly contributing to improved disease management practices and, ultimately, to food security. This methodological advancement, coupled with a focus on data diversity and advanced preprocessing techniques, underscores the thesis's potential to fill existing gaps in the field and pave the way for future research in plant pathology and automated disease diagnostics.

6.4 Achievement of Objectives

In fulfilling the aim of this research, the investigation undertook a detailed evaluation and comparison of seven Machine Learning (ML) models and two Deep Learning (DL) models to ascertain their efficacy in diagnosing diseases in apple plant leaves. This endeavour was driven by the overarching goal of identifying the most precise model to enhance disease management within apple cultivation, a critical aspect of agricultural technology that directly impacts food security and economic stability.

6.4.1 Objective 1: Analysing ML Models for Disease Identification

The investigation meticulously analyzed various ML models, including Logistic Regression, Decision Trees, LDA, KNN, Random Forest, Naive Bayes, and SVM. These models were evaluated for their capability to accurately identify diseases such as Apple Scab, Black Rot, and Cedar Apple Rust in apple leaves. Through rigorous testing and evaluation, the investigation established the strengths and limitations of each ML model in the context of disease detection, providing valuable insights into their applicability in real-world agricultural settings. Random Forest (RF) model emerged as a standout performer with a test accuracy of 95.21% showcasing exceptional ability to generalize well to unseen data

6.4.2 Objective 2: Leveraging CNN Architectures for Increased Precision

The research significantly leveraged the potential of Convolutional Neural Networks (CNN) with NASNetMobile and MobileNetV2 architectures. This exploration revealed the heightened precision that DL models, particularly those with advanced architectures, can offer in detecting leaf diseases. The comparative analysis demonstrated MobileNetV2's superior performance, making it a preferred choice for tasks requiring high accuracy and efficiency in disease diagnosis.

6.4.3 Objective 3: Comparative Study of ML and DL Models

A comparative study was conducted focusing on critical metrics such as accuracy, precision, recall, and F1-score. This comprehensive comparison not only elucidated the capabilities and performance of each model but also identified MobileNetV2 as the most proficient model for disease detection in apple leaves. The study's findings contribute to a deeper understanding of the models' operational efficacy and their potential integration into disease management systems.

6.4.4 Objective 4: Implementing Advanced Image Processing Techniques

The research met this objective by employing advanced image processing and feature extraction techniques in both ML and DL frameworks. The ML code included preprocessing steps like color space conversions and image segmentation, crucial for isolating disease indicators. The DL code utilized ImageDataGenerators for data augmentation (rotation, shifting, flipping) and MobileNetV2 and NASNetMobile architectures for feature extraction through deep convolutional layers. These techniques enhanced the models' ability to accurately classify plant health conditions, showcasing the objective's achievement in improving model reliability and disease detection performance.

The accomplishment of these objectives has not only enriched the existing body of knowledge in plant pathology and agricultural technology but also laid down a solid foundation for future research. The insights garnered from this study underscore the potential of integrating ML and DL models into practical applications, promising significant advancements in precision agriculture.

6.5 Contribution to Knowledge

This section underscores this study's advancements in leveraging ML and DL models for precise apple leaf disease detection. It delineates the development of novel methodologies and the comparative analysis of various models, emphasizing the superior performance of MobileNetV2 in diagnosing diseases with remarkable accuracy. This research fills a critical gap by providing a comprehensive evaluation framework, thus paving the way for future innovations in agricultural technology. It also lays the groundwork for scalable disease management strategies, significantly contributing to the academic and practical realms of precision agriculture.

6.6 Limitations and Future Recommendations

In the comprehensive performance analysis conducted in Chapter 5, a diverse array of models were meticulously evaluated for their efficacy in detecting diseases in apple leaves. The analysis encompassed seven ML models: Naïve Bayes (NB), Random Forest (RF), Logistic Regression (LR), Support Vector Machine (SVM), Linear Discriminant Analysis (LDA), K-Nearest Neighbors (KNN), and Decision Trees (CART), alongside two cutting-edge Deep Learning

(DL) models: MobileNetV2 and NASNetMobile. This rigorous evaluation has yielded several actionable recommendations to steer future research and advancements in this crucial domain of agricultural technology, ensuring that the insights drawn from this extensive analysis inform the next generation of disease detection methodologies.

6.6.1 Enhancing Data Collection and Annotation

- **Diversify Data Sources:** To improve model robustness and generalization, future efforts should focus on expanding the dataset to include a wider variety of apple leaf images, capturing a broader spectrum of diseases, and environmental conditions. This diversification will enable models to better adapt to real-world variability.
- **High-Quality Annotation:** Invest in meticulous annotation processes to ensure the accuracy and reliability of labels, especially for DL models like MobileNetV2, which demonstrated remarkable precision and recall rates exceeding 98%. Accurate annotations are crucial for the effective training and performance of these models.

6.6.2 Model Optimization and Innovation

- **Hyperparameter Tuning:** Exploring advanced hyperparameter tuning for the Random Forest (RF) model, which achieved a remarkable test accuracy of 95.21%, could further refine and amplify its effectiveness. Beyond the traditional Grid Search, consider employing more sophisticated or newer hyperparameter optimization techniques such as Bayesian Optimization, Genetic Algorithms, or Random Search with a broader parameter space to potentially uncover more optimal model configurations.
- **Hybrid Models:** Explore the development of hybrid models that combine the strengths of both ML and DL approaches. Integrating the robust generalization capabilities of RF with the advanced feature extraction abilities of MobileNetV2 could lead to models that offer both high accuracy and adaptability to diverse agricultural settings.

6.6.3 Computational Efficiency and Accessibility

- **Efficiency Improvements:** For DL models, particularly MobileNetV2, focus on optimizing computational efficiency to facilitate deployment in resource-constrained environments common in agricultural settings. Techniques such as model pruning,

quantization, and knowledge distillation can reduce model size and computational requirements without significant loss in performance.

- **Edge Computing:** Investigate the integration of models into edge computing devices for real-time disease detection directly in the field. This approach can significantly reduce the latency and bandwidth requirements associated with cloud computing solutions.

6.6.4 Deployment and Real-world Testing

- **Pilot Studies:** Conduct pilot studies to assess the real-world applicability of the models, especially in diverse agricultural environments. These studies can provide valuable feedback on the models' performance in situ and identify practical challenges that may not be apparent in laboratory settings.
- **User-Centric Design:** Develop user-friendly interfaces and decision-support systems that integrate the models, making them accessible to farmers and agricultural professionals. These systems should be designed with the end-user in mind, emphasizing ease of use and actionable insights.

6.6.5 Continuous Learning and Model Updating

- **Incremental Learning:** Implement mechanisms for models, particularly DL models, to learn incrementally from new data as they become available. This approach can help the models stay relevant and effective as new diseases emerge and agricultural practices evolve.
- **Open-source Collaboration:** Foster an open-source community around the developed models to encourage collaboration, innovation, and continuous improvement. Sharing models and datasets can accelerate advancements in the field and lead to more robust and effective solutions.

By adhering to these recommendations, future research and development in the field of apple leaf disease detection can build on the solid foundation established in Chapter 4 and Chapter 5. The integration of advanced ML and DL models into agricultural practices holds the promise of enhancing disease management strategies, improving crop health and yields, and contributing to the sustainability and efficiency of agricultural operations globally.

REFERENCES

Agarwal, M., Singh, A., Arjaria, S., Sinha, A. & Gupta, S. (2020). ToLeD: Tomato Leaf Disease Detection using Convolution Neural Network. *Procedia Computer Science*. [Online]. 167 (2019). p.pp. 293–301. Available from: https://doi.org/10.1016/j.procs.2020.03.225.

Aggarwal, M., Khullar, V. & Goyal, N. (2023). Exploring Classification of Rice Leaf Diseases using Machine Learning and Deep Learning. In: *Proceedings of 2023 3rd International Conference on Innovative Practices in Technology and Management, ICIPTM 2023*. 2023, Institute of Electrical and Electronics Engineers Inc.

Ahmed, I. & Yadav, P.K. (2022). Plant disease detection using machine learning approaches. *Expert Systems*. (April 2022). p.pp. 1–16.

Al-gaashani, M.S.A.M., Shang, F., Muthanna, M.S.A., Khayyat, M. & Abd El-Latif, A.A. (2022). Tomato leaf disease classification by exploiting transfer learning and feature concatenation. *IET Image Processing*. 16 (3). p.pp. 913–925.

Astani, M., Hasheminejad, M. & Vaghefi, M. (2022). A diverse ensemble classifier for tomato disease recognition. *Computers and Electronics in Agriculture*. 198.

Atila, Ü., Uçar, M., Akyol, K. & Uçar, E. (2021). Plant leaf disease classification using EfficientNet deep learning model. *Ecological Informatics*. [Online]. 61 (September 2020). p.p. 101182. Available from: https://doi.org/10.1016/j.ecoinf.2020.101182.

Aurangzeb, K., Akmal, F., Attique Khan, M., Sharif, M. & Javed, M.Y. (2020). Advanced Machine Learning Algorithm Based System for Crops Leaf Diseases Recognition. In: *Proceedings - 2020 6th Conference on Data Science and Machine Learning Applications, CDMA 2020*. 1 March 2020, Institute of Electrical and Electronics Engineers Inc., pp. 146–151.

Baheti, H., Thakare, A., Bhople, Y., Darekar, S. & Dodmani, O. (2022). Machine Learning Algorithm for Detection And Classification of Tomato Plant Leaf Disease. In: *2022 IEEE 7th International conference for Convergence in Technology, I2CT 2022*. 2022, Institute of Electrical and Electronics Engineers Inc.

Bhosale, J.D., Thorat, S.S., Pancholi, P.V. & Mutkule, P.R. (2023). Machine Learning-Based Algorithms for the Detection of Leaf Disease in Agriculture Crops. *International Journal on Recent and Innovation Trends in Computing and Communication*. 11. p.pp. 45–50.

Chaitanya Reddy, P., Chandra, R.M.S., Vadiraj, P., Ayyappa Reddy, M., Mahesh, T.R. & Sindhu Madhuri, G. (2021). Detection of Plant Leaf-based Diseases Using Machine Learning Approach. In: *CSITSS 2021 - 2021 5th International Conference on Computational Systems and Information Technology for Sustainable Solutions, Proceedings*. 2021, Institute of Electrical and Electronics Engineers Inc.

Chowdhury, M.E.H., Rahman, T., Khandakar, A., Ayari, M.A., Khan, A.U., Khan, M.S., Al-Emadi, N., Reaz, M.B.I., Islam, M.T. & Ali, S.H.M. (2021). Automatic and Reliable Leaf Disease Detection Using Deep Learning Techniques. *AgriEngineering*. 3 (2). p.pp. 294–312.

Chug, A., Bhatia, A., Singh, A.P. & Singh, D. (2023). A novel framework for image-based plant disease detection using hybrid deep learning approach. *Soft Computing*. [Online]. 27

(18). p.pp. 13613–13638. Available from: https://doi.org/10.1007/s00500-022-07177-7.

Gosai, D., Kaka, B., Garg, D., Patel, R. & Ganatra, A. (2022). Plant Disease Detection and Classification Using Machine Learning Algorithm. In: *2022 International Conference for Advancement in Technology, ICONAT 2022*. 2022, Institute of Electrical and Electronics Engineers Inc.

Habiba, S.U. & Islam, M.K. (2021). Tomato Plant Diseases Classification Using Deep Learning Based Classifier from Leaves Images. In: *2021 International Conference on Information and Communication Technology for Sustainable Development, ICICT4SD 2021 - Proceedings*. 27 February 2021, Institute of Electrical and Electronics Engineers Inc., pp. 82–86.

Harakannanavar, S.S., Rudagi, J.M., Puranikmath, V.I., Siddiqua, A. & Pramodhini, R. (2022). Plant leaf disease detection using computer vision and machine learning algorithms. *Global Transitions Proceedings*. 3 (1). p.pp. 305–310.

Hatuwal, B.K., Shakya, A. & Joshi, B. (2020). Plant Leaf Disease Recognition Using Random Forest, KNN, SVM and CNN. *Polibits*. [Online]. 62 (May). p.pp. 13–19. Available from: https://doi.org/10.17562/PB-62-2.

Jasim, M.A. & Al-Tuwaijari, J.M. (2020). Plant Leaf Diseases Detection and Classification Using Image Processing and Deep Learning Techniques. *Proceedings of the 2020 International Conference on Computer Science and Software Engineering, CSASE 2020*. p.pp. 259–265.

Kholiya, D., Mishra, A.K., Dumka, A., Pandey, N.K. & Tripathi, N. (2023). Detection of Leaf Diseases in Agricultural Plants Using Machine Learning. In: *2023 International Conference on Computer, Electronics and Electrical Engineering and their Applications, IC2E3 2023*. 2023, Institute of Electrical and Electronics Engineers Inc.

Kilaru, R. & Raju, K.M. (2022). Prediction of Maize Leaf Disease Detection to improve Crop Yield using Machine Learning based Models. In: *4th International Conference on Recent Trends in Computer Science and Technology, ICRTCST 2021 - Proceedings*. 2022, Institute of Electrical and Electronics Engineers Inc., pp. 212–217.

Kumar, R., Shukla, N. & Princee (2022). Plant Disease Detection and Crop Recommendation Using CNN and Machine Learning. In: *2022 International Mobile and Embedded Technology Conference, MECON 2022*. 2022, Institute of Electrical and Electronics Engineers Inc., pp. 168–172.

Lamba, M., Gigras, Y. & Dhull, A. (2021). Classification of plant diseases using machine and deep learning. *Open Computer Science*. 11 (1). p.pp. 491–508.

Meenakshi, T. (2023). Automatic Detection of Diseases in Leaves of Medicinal Plants Using Modified Logistic Regression Algorithm. *Wireless Personal Communications*. [Online]. 131 (4). p.pp. 2573–2597. Available from: https://doi.org/10.1007/s11277-023-10555-5.

Mishra, S., Sachan, R. & Rajpal, D. (2020). Deep Convolutional Neural Network based Detection System for Real-time Corn Plant Disease Recognition. *Procedia Computer Science*. [Online]. 167. p.pp. 2003–2010. Available from: https://doi.org/10.1016/j.procs.2020.03.236.

Mohanty, R., Wankhede, P., Singh, D. & Vakhare, P. (2022). Tomato Plant Leaves Disease Detection using Machine Learning. *2022 International Conference on Applied Artificial*

Intelligence and Computing (ICAAIC). (Icaaic). p.pp. 544–549.

Pawar, S., Shedge, S., Panigrahi, N., Jyoti, A.P., Thorave, P. & Sayyad, S. (2022). Leaf Disease Detection of Multiple Plants Using Deep Learning. In: *2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing, COM-IT-CON 2022*. 2022, Institute of Electrical and Electronics Engineers Inc., pp. 241–245.

Ramiah Subburaj, S.D., Vaithyam Rengarajan, V.K. & Palaniswamy, S. (2023). Transfer Learning based Image Classification of Diseased Tomato Leaves with Optimal Fine-Tuning combined with Heat Map Visualization. *Tarim Bilimleri Dergisi*. 29 (4). p.pp. 1003–1017.

Rizvee, R.A., Orpa, T.H., Ahnaf, A., Kabir, M.A., Ahmmad Rashid, M.R., Islam, M.M., Islam, M., Jabid, T. & Ali, M.S. (2023). LeafNet: A proficient convolutional neural network for detecting seven prominent mango leaf diseases. *Journal of Agriculture and Food Research*. [Online]. 14 (August). p.p. 100787. Available from: https://doi.org/10.1016/j.jafr.2023.100787.

Sai, A.M. & Patil, N. (2022). Comparative Analysis of Machine Learning Algorithms for Disease Detection in Apple Leaves. In: *2022 IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics, DISCOVER 2022 - Proceedings*. 2022, Institute of Electrical and Electronics Engineers Inc., pp. 239–244.

Sangeetha, K., Vishnu Raja, P., Rima, P., Pranesh Kumar, M. & Preethees, S. (2022). Apple Leaf Disease Detection using Deep Learning. In: *Proceedings - 6th International Conference on Computing Methodologies and Communication, ICCMC 2022*. 2022, Institute of Electrical and Electronics Engineers Inc., pp. 1063–1067.

Shakeel, W., Ahmad, M. & Mahmood, N. (2020). Early Detection of Cercospora Cotton Plant Disease by Using Machine Learning Technique. In: *30th International Conference on Computer Theory and Applications, ICCTA 2020 - Proceedings*. 12 December 2020, Institute of Electrical and Electronics Engineers Inc., pp. 44–48.

Sharma, V., Mir, A.A. & Sarwr, D.A. (2020). Detection of Rice Disease Using Bayes' Classifier and Minimum Distance Classifier. *Journal of Multimedia Information System*. 7 (1). p.pp. 17–24.

Shivaprasad, K. & Wadhawan, A. (2023). Deep Learning-based Plant Leaf Disease Detection. In: *Proceedings of the 7th International Conference on Intelligent Computing and Control Systems, ICICCS 2023*. 2023, Institute of Electrical and Electronics Engineers Inc., pp. 360–365.

Shrivastava, V.K. & Pradhan, M.K. (2021). Rice plant disease classification using color features: a machine learning paradigm. *Journal of Plant Pathology*. 103 (1). p.pp. 17–26.

Silviya, S.H.A., Sriman, B., Shamini, P.B., Elangovan, A., Monica, A.R. & Keerthana, N. V. (2022). Deep Learning based Plant Leaf Disease Detection and Classification. In: *4th International Conference on Inventive Research in Computing Applications, ICIRCA 2022 - Proceedings*. 2022, Institute of Electrical and Electronics Engineers Inc., pp. 702–710.

Singla, R.S., Gupta, A., Gupta, R., Tripathi, V., Naruka, M.S. & Awasthi, S. (2023). Plant Disease Classification Using Machine Learning. In: *2023 International Conference on Disruptive Technologies, ICDT 2023*. 2023, IEEE, pp. 409–413.

Sujatha, R., Chatterjee, J.M., Jhanjhi, N.Z. & Brohi, S.N. (2021). Performance of deep learning

vs machine learning in plant leaf disease detection. *Microprocessors and Microsystems*. 80.

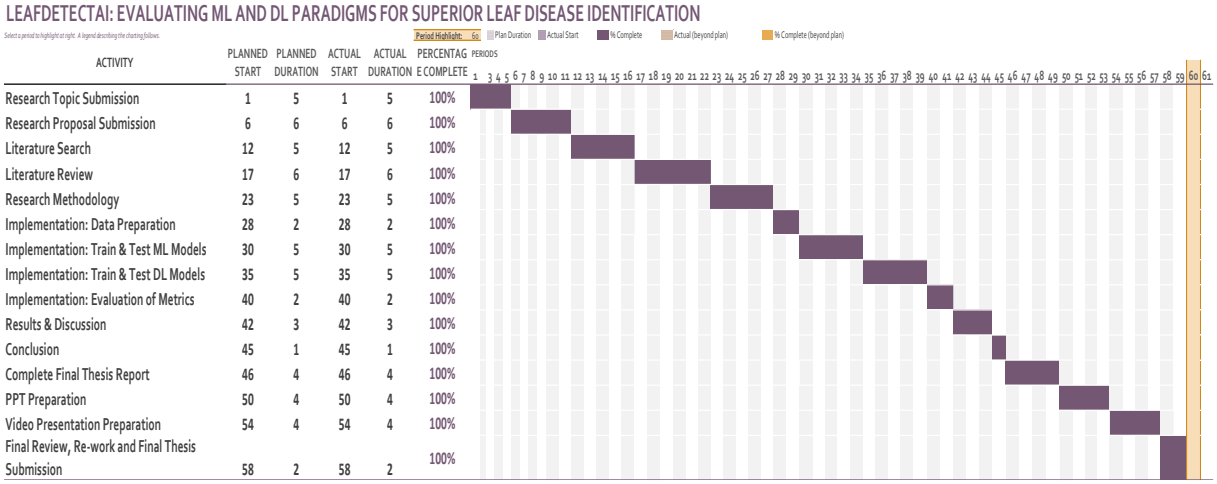
Trivedi, N.K., Gautam, V., Anand, A., Aljahdali, H.M., Villar, S.G., Anand, D., Goyal, N. & Kadry, S. (2021). Early detection and classification of tomato leaf disease using high-performance deep neural network. *Sensors*. 21 (23).

Wójtowicz, A., Piekarczyk, J., Czernecki, B. & Ratajkiewicz, H. (2021). A random forest model for the classification of wheat and rye leaf rust symptoms based on pure spectra at leaf scale. *Journal of Photochemistry and Photobiology B: Biology*. 223 (August).

Yousuf, A. & Khan, U. (2021). Ensemble Classifier for Plant Disease Detection. *International Journal of Computer Science and Mobile Computing*. 10 (1). p.pp. 14–22.

Zhong, Y. & Zhao, M. (2020). Research on deep learning in apple leaf disease recognition. *Computers and Electronics in Agriculture*. [Online]. 168 (October 2019). p.p. 105146. Available from: https://doi.org/10.1016/j.compag.2019.105146.

APPENDIX A: RESEARCH PLAN



ACTIVITY	PLANNED START	PLANNED DURATION	ACTUAL START	ACTUAL DURATION	PERCENTAGE COMPLETE
Research Topic Submission	1	5	1	5	100%
Research Proposal Submission	6	6	6	6	100%
Literature Search	12	5	12	5	100%
Literature Review	17	6	17	6	100%
Research Methodology	23	5	23	5	100%
Implementation: Data Preparation	28	2	28	2	100%
Implementation: Train & Test ML Models	30	5	30	5	100%
Implementation: Train & Test DL Models	35	5	35	5	100%
Implementation: Evaluation of Metrics	40	2	40	2	100%
Results & Discussion	42	3	42	3	100%
Conclusion	45	1	45	1	100%
Complete Final Thesis Report	46	4	46	4	100%
PPT Preparation	50	4	50	4	100%
Video Presentation Preparation	54	4	54	4	100%
Final Review, Re-work and Final Thesis Submission	58	2	58	2	100%

APPENDIX B: RESEARCH PROPOSAL

Abstract

Plant diseases pose a severe threat to global food security, prompting the exploration of innovative detection strategies. Traditional methods are time-consuming, subjective, and error-prone, highlighting the need for automated alternatives. This research proposes a methodology integrating machine learning, image processing, and feature extraction for precise and scalable plant disease detection.

The problem addressed involves the limitations of traditional disease detection methods, necessitating automated and accurate alternatives. Machine learning offers a solution, leveraging its efficiency in analyzing large datasets. The research combines image processing and machine learning, utilizing RGB to HSV color space conversion, image segmentation, and feature extraction (Haralick textures, color histograms, and Hu moments). These techniques capture intricate disease patterns, enabling machine learning models to distinguish between healthy and diseased conditions.

The purpose of this research is multi-faceted. It aims to validate the proposed methodology, assess contributions of individual features and models, develop a robust and scalable approach, and explore machine learning's applications in precision agriculture. Bridging the gap between image data and disease identification, the research empowers farmers with a tool for effective disease management, yield optimization, and sustainable agriculture.

Expected results include methodology validation, optimal feature descriptor elucidation, and a foundation for advancing precision agriculture. In conclusion, this research has the potential to revolutionize plant disease detection, offering automated, accurate, and scalable solutions as a paradigm shift from traditional methods.

Table of Contents

List of Figures

1. Background

Agriculture, the backbone of global economies, is facing unprecedented challenges exacerbated by the increasing threat of plant diseases. These diseases pose a significant risk to plant yields, food security, and economic stability. Traditional methods of disease detection and management are often manual, time-consuming, and prone to inaccuracies. As a response to these challenges, the integration of advanced technologies, particularly machine learning and computer vision, has gained momentum in the agricultural sector.

Precision agriculture, enabled by cutting-edge technologies, aims to optimize farming practices by leveraging data-driven insights. In this context, machine learning algorithms have shown great promise in revolutionizing plant disease detection. By harnessing the power of large datasets and sophisticated algorithms, there is an opportunity to create more efficient, accurate, and timely solutions for identifying and managing plant diseases.

Numerous research efforts have delved into the application of machine learning in agriculture, with a specific focus on plant disease detection. Existing studies highlight the limitations of conventional approaches and emphasize the potential of technology-driven solutions. Noteworthy contributions in the literature showcase the effectiveness of image processing techniques, feature extraction methods, and various machine learning algorithms in enhancing the accuracy and speed of disease identification.

This study builds upon the foundation laid by previous research, synthesizing insights from diverse studies to formulate a comprehensive and innovative approach to plant disease detection. By drawing from the experiences and findings of seminal works in the field, this research aims to contribute to the ongoing evolution of precision agriculture.

The core problem addressed by this research is the inefficiency of traditional methods in providing timely and accurate diagnoses of plant diseases. Manual inspections and subjective assessments are often insufficient, leading to delayed responses and compromised agricultural productivity. The consequences of undetected or misdiagnosed diseases include reduced plant yields, economic losses for farmers, and the potential for widespread food insecurity.

The primary purpose of this study is to develop an advanced and efficient methodology for plant disease detection, leveraging machine learning, computer vision, and data analytics. By

overcoming the limitations of existing practices, the research seeks to contribute to the optimization of disease identification processes. The overarching goal is to empower farmers with real-time, accurate information, enabling them to implement timely interventions and mitigate the impact of plant diseases. Through these advancements, the study aligns with the broader objective of promoting sustainable agriculture and enhancing global food security.

2. Problem Statement OR Related Research OR Related Work

Automatic Detection of Diseases in Leaves of Medicinal Plants Using Modified Logistic Regression Algorithm (Meenakshi, 2023) presents a novel approach utilizing a modified logistic regression algorithm for diagnosing diseases in medicinal plant leaves. This method incorporates adaptive gamma correction for feature enhancement in leaf images, k-means clustering for differentiating healthy and diseased areas and employs Gray Level Co-occurrence Matrix (GLCM) along with logistic regression for feature extraction and classification. The study compares the classification accuracy of this new method with traditional logistic regression, showing improved performance. It concludes that this innovative approach is effective for early disease detection in medicinal plant leaves.

A review on machine learning techniques for rice plant disease detection in agricultural research (Daniya & Vigneshwari, 2019) is a research paper that provides a systematic literature study on the applications of the cutting-edge DL and ML algorithms such as Naïve Bayes (NB), Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Neural Network (NN), other few popular ML algorithms and AlexNet, GoogLeNet, VGGNet, and other few popular DL algorithms respectively for plant disease categorization. The paper concludes that the use of machine learning techniques has shown promising results in the detection of rice plant diseases. The paper recommends further research on the use of machine learning algorithms for plant disease detection and prediction.

Tomato Plant Leaves Disease Detection using Machine Learning (Mohanty et al., 2022) is a research paper that presents an end-to-end system for detecting tomato disease using ML algorithms like random forest, support vector machine, and logistic regression algorithms. The study finds that the suggested approach successfully detects diseases in tomato plant leaves, serving as a valuable tool for early disease identification in vegetation. The proposed system uses the Plant Village dataset, which comprises apple, corn, grape, and tomato plant leaves with healthy and diseased categories. The paper addresses the problem of plant damage being done to farmers due to lack of awareness on the pathogenesis of plants, changes in nature, and lack of proper research on diseases of plants. The aim of the research is to create a comprehensive system for identifying tomato diseases through the application of machine learning techniques. The paper recommends further research on the use of machine learning algorithms for plant disease detection and prediction.

Digital image processing technique for palm oil leaf disease detection using multiclass SVM classifier (Masazhar & Kamal, 2017) is a research paper that proposes a framework for early detection of plant diseases through leaf features inspection. The proposed framework is modeled into four parts: image preprocessing, segmenting the region of interest using K-mean clustering, feature extraction and classification, and texture feature extraction using statistical GLCM and color feature by means of mean values. The dataset used in this work is not mentioned. The paper addresses the problem of plant damage being done to farmers due to lack of awareness on the pathogenesis of plants, changes in nature, and lack of proper research on diseases of plants. The purpose of the study is to implement image analysis and classification techniques for extraction and classification of leaf diseases. The study's findings suggest that the suggested approach successfully detects plant leaf diseases, making it a viable solution for the early identification of such conditions in vegetation.

Classification of plant diseases using machine and deep learning (Lamba et al., 2021) is a research paper that presents a comprehensive survey of the applications of machine learning and deep learning algorithms for the classification, detection, and prediction of plant diseases. The paper concludes that the proposed methods are effective in identifying diseases in leaves of plants and can be used as a tool for early detection of diseases in plants. The paper uses various preprocessing techniques such as image filtering, segmentation, and feature extraction. The algorithms used in this work include support vector machine, random forest, k-nearest neighbour, and Naïve Bayes. The evaluation metrics used in this work include precision, recall, and f1 score. The paper recommends further research on the use of deep learning algorithms for plant disease detection and prediction.

An Overview on Prediction of Plant Leaves Disease using Image Processing Techniques (Sakhamuri & Kompalli, 2020) is a research paper that proposes a framework for early detection of plant diseases through leaf features inspection. The paper was published in 2020 by Sakhamuri et al. in the International Conference on Smart Instrumentation, Measurement and Application. The proposed framework is modeled into four parts: image preprocessing, segmentation of the region of interest using Otsu’s method, feature extraction using GLCM, and classification using multiclass Support Vector Machine (SVM). The paper addresses the problem of plant damage being done to farmers due to lack of awareness on the pathogenesis of plants, changes in nature, and lack of proper research on diseases of plants. The purpose of the study is to implement image analysis and classification techniques for extraction

and classification of leaf diseases. The research demonstrates the efficacy of the introduced technique in diagnosing plant leaf ailments, positioning it as a valuable resource for pre-emptive disease recognition in plants.

Recent Advances in Image Processing Techniques for Automated Leaf Pest and Disease Recognition - A Review (Ngugi et al., 2021). In this paper, Ngugi et al. explore the potential of image processing and machine learning techniques for automated plant pest and disease recognition, addressing the limitations of traditional methods and the promise of image processing for real-time disease detection. The authors delve into preprocessing techniques, including image resizing, color normalization, and contrast enhancement, to prepare input images for subsequent processing stages. They also discuss feature extraction methods, such as texture features (GLCM) and color features (mean values), to capture relevant characteristics of the leaf. The study utilizes various machine learning algorithms, including support vector machines (SVMs), convolutional neural networks (CNNs), and random forests, for classification purposes. The performance of these algorithms is evaluated using metrics such as accuracy, precision, recall, and F-measure.

A comprehensive review on detection of plant disease using machine learning and deep learning approaches (Jackulin & Murugavalli, 2022). In this paper, Jackulin et al. explored the potential of machine learning and deep learning approaches for plant disease detection, addressing the limitations of traditional methods and the promise of image processing for real-time disease detection. They employed various preprocessing techniques, such as image filtering, segmentation, and feature extraction, to prepare input images for subsequent processing stages. The study also utilized a range of ML algorithms, including random forests, support vector machines (SVMs), k-nearest neighbors (KNNs) and convolutional neural networks (CNNs), for disease classification tasks. Algorithm performance was evaluated using metrics like precision, recall, and F1-score. The research suggests further exploration in the area of deep learning for the detection and prediction of plant diseases.

Machine Learning for Detection and Prediction of Plant Diseases and Pests: A Comprehensive Survey (Domingues et al., 2022). In this paper, authors have explored the potential of machine learning approaches to address the limitations of traditional methods for plant disease detection and classification, highlighting the promise of image processing for real-time disease detection. The authors delved into various preprocessing techniques, including image resizing, color

normalization, and contrast enhancement, to prepare input images for subsequent processing stages. They also discussed feature extraction methods, such as texture features (GLCM) and color features (mean values), to capture relevant characteristics of the leaf. The study utilized various ML algorithms, including random forests, support vector machines (SVMs), and convolutional neural networks (CNNs), for classification purposes. The performance of these algorithms was evaluated using metrics such as accuracy, precision, recall, and F1-score. The authors conclude with future recommendations and trends in the area of plant pest and disease recognition using machine learning techniques.

Plant Leaf Disease Recognition Using Random Forest, KNN, SVM and CNN (Hatuwal et al., 2020). In this paper, Hatuwal et al. have proposed a machine learning-based system for plant disease classification and prediction using leaf images. They addressed the limitations of traditional methods and advocated for the use of machine learning techniques. The authors employed feature extraction techniques and various machine learning algorithms, including SVM, KNN, Random Forest, and CNN. CNNs achieved the highest accuracy of 97.89%, followed by Random Forest (87.43%), SVM (78.61%), and KNN (76.96%). They highlighted challenges such as image variability and limited labelled data. Future recommendations include developing robust algorithms, integrating machine learning with other data sources, and exploring real-time detection solutions.

Plant Disease Classification Using Machine Learning (Singla et al., 2023). In this paper, authors address the limitations of traditional methods for plant disease detection and classification by exploring machine learning approaches. Singla et al. have proposed a system that uses Extreme Learning Machine (ELM) for classifying and predicting plant diseases from leaf images. They employed preprocessing techniques like image resizing, color normalization, and contrast enhancement to prepare input images. Image features were extracted using Haralick textures and applied to the ELM classifier. The ELM model achieved an accuracy of 84.94%, outperforming other machine learning algorithms like SVM and Decision Trees. However, challenges remain, such as variability of leaf images and limited availability of labelled data. Future research should focus on developing robust and scalable algorithms, integrating machine learning with other data sources, and exploring real-time detection solutions.

A Comprehensive Survey on Leaf Disease Identification & Classification (Bhagat & Kumar, 2022). In this paper, authors address the shortcomings of traditional plant disease detection

methods by conducting a comprehensive survey of image processing techniques for plant disease classification. They provided an in-depth review of recent studies employing image processing and machine learning for plant pest and disease recognition, focusing on classifying healthy and diseased leaves based on morphological features like texture, color, shape, and pattern. The authors discussed various preprocessing techniques, including image filtering, segmentation, and feature extraction, as well as machine learning algorithms like Support Vector Machines (SVMs), Random Forest, and K-Nearest Neighbors (KNNs). They highlighted the challenges of image variability, limited labelled data, and computational complexity. Future research directions include developing robust and scalable algorithms, integrating machine learning with other data sources, and exploring real-time detection solutions.

LeafNet: A proficient convolutional neural network for detecting seven prominent mango leaf diseases (Rizvee et al., 2023). The authors propose a system that utilizes a convolutional neural network to classify and predict mango leaf diseases from images. Employing the PlantVillage dataset, which encompasses healthy and diseased mango plant leaves, the system incorporates various preprocessing techniques, including image reorientation, plantping, gray scaling, binary thresholding, noise removal, contrast stretching, threshold inversion, and edge recognition. The system's performance is evaluated using various metrics, including accuracy, precision, recall, and F1 score. The authors advocate for the proposed system's application in early detection of mango leaf diseases to minimize plant yield loss. The paper also presents a comprehensive review of the literature on leaf disease-based models for diagnosing various plant leaf diseases using deep learning.

Plant Leaf Diseases Detection and Classification Using Image Processing and Deep Learning Techniques (Jasim & Al-Tuwaijari, 2020). The study proposes a system that utilizes a convolutional neural network to classify and predict mango leaf diseases from images. The authors employed the PlantVillage dataset, which encompasses mango plant leaves with healthy and diseased categories. The system incorporates various preprocessing techniques, including image reorientation, plantping, gray scaling, binary thresholding, noise removal, contrast stretching, threshold inversion, and edge recognition. The system's performance is assessed using various evaluation metrics, including accuracy, precision, recall, and F1 score. The authors advocate for the proposed system's application in early detection of mango leaf diseases to minimize plant yield loss. The paper also offers a comprehensive review of the

literature on leaf disease-based models for diagnosing various plant leaf diseases using deep learning.

Plant disease detection using computational intelligence and image processing (Vishnoi et al., 2021). In the realm of plant disease detection, computer vision and soft computing techniques have emerged as promising alternatives to traditional methods that are often expensive, time-consuming, and impractical. This paper delves into the application of image processing for plant disease recognition, a rapidly evolving research area. The authors provide a comprehensive review of recent studies that employ image processing and machine learning for plant pest and disease recognition, with a particular focus on plant leaves. They discuss various preprocessing techniques like image resizing, normalization, and augmentation. Algorithms used include support vector machines (SVMs), k-nearest neighbors (KNNs), and convolutional neural networks (CNNs). Evaluation metrics used include accuracy, precision, recall, and F1 score. The paper concludes with a discussion of future directions and trends in the field of plant pest and disease recognition using image processing techniques.

3. Research Questions

This thesis tries to answer the following questions:

- How does the conversion of images from BGR to RGB and the subsequent conversion to HSV contribute to more effective feature extraction in the context of plant disease detection?
- What role does image segmentation, specifically the extraction of green and brown colors, play in isolating healthy and diseased regions in the plant images?
- How do the Hu Moments, Haralick Texture, and color Histogram feature descriptors individually contribute to the characterization of plant disease patterns?
- What advantages do these specific feature descriptors offer in comparison to alternative techniques in the context of plant disease identification?
- How does the combination of Hu Moments, Haralick Texture, and color Histogram contribute to the overall global feature extraction process?
- In what ways does the concatenation of these features enhance the effectiveness of machine learning models in plant disease detection?
- How do different machine learning models, such as Logistic Regression, Random Forest, and Support Vector Machines, compare in terms of accuracy for plant disease detection?
- What insights can be drawn from the confusion matrix and classification report generated during the evaluation of the trained models?
- How scalable is the proposed machine learning approach to larger datasets or diverse plant types beyond those considered in the current implementation?
- To what extent can the trained models generalize to detect diseases in plants not present in the training dataset?
- Are there opportunities for further optimization of the machine learning models, and if so, what strategies could be employed to enhance their performance?
- What future recommendations can be made for refining the proposed approach, considering the evolving landscape of machine learning in agriculture and plant disease detection?

4. Aim and Objectives

The aim of this research is to implement an effective plant disease detection system using machine learning techniques, focused on agricultural sustainability. This study aims to empower farmers with a reliable tool for early disease identification, thereby minimizing plant losses.

The research objectives are formulated based on the aim of this study which are as follows:

- To implement RGB to HSV conversion and image segmentation, extracting global feature descriptors: Utilize image processing techniques to convert images from RGB to HSV format and perform segmentation. Extract crucial features related to healthy and diseased plant regions using Hu Moments, Haralick Texture, and color Histogram features.
- To employ machine learning algorithms for classification: Utilize diverse machine learning algorithms, including Logistic Regression, Decision Trees, Random Forest, and Support Vector Machines. Train and evaluate models for accurate classification of healthy and diseased plants.
- To evaluate and compare model performance: Assess the performance of the trained models through cross-validation. Compare accuracy, precision, recall, and F1-score metrics to determine the most effective algorithm for plant disease detection.
- To encode and normalize target labels: Utilize label encoding and feature scaling techniques to prepare the dataset for training. Ensure consistency and reliability in model learning across different plant types.
- To save and store the feature vector: Implement HDF5 file storage for efficient saving and retrieval of the feature vectors. Facilitate seamless integration into the machine learning workflow.
- To train, test, and evaluate machine learning models: Implement a comprehensive training and testing pipeline for various ML models, including Logistic Regression, Support Vector Machines, and Random Forests. Evaluate their efficacy in plant disease detection.

By achieving these objectives, this research aims to contribute valuable insights and methodologies to the field of precision agriculture, enhancing farmers' ability to detect and manage plant diseases effectively.

5. Significance of the Study

The significance of this study lies in its contribution to advancing plant disease detection methodologies through the application of machine learning. By employing RGB to HSV conversion, image segmentation, and global feature extraction techniques, this research enhances the accuracy and efficiency of identifying plant diseases. The implementation of diverse machine learning models, such as Logistic Regression, Random Forest, and Support Vector Machines, adds versatility to the detection process.

The beneficiaries of this study include farmers, agricultural researchers, and policymakers. Farmers stand to benefit from early and accurate disease identification, enabling timely interventions to prevent widespread plant damage. Agricultural researchers gain insights into the effectiveness of various machine learning models for disease detection, fostering further advancements in the field. Policymakers can leverage this research to make informed decisions regarding plant management and food security initiatives. Overall, the study's outcomes have practical implications for improving plant health and securing agricultural livelihoods.

6. Scope of the Study

- Development and evaluation of a plant disease detection system using machine learning techniques.
- Focus on RGB to HSV conversion, image segmentation, and global feature extraction for enhanced disease identification accuracy.
- Utilization of multiple machine learning models, including Logistic Regression, Random Forest, and Support Vector Machines.
- Assessment of model efficacy within a controlled environment using pre-captured images.
- Exclusion of real-time implementation in the field, considering factors like varying lighting conditions and the dynamic nature of outdoor agriculture.
- Limited exploration of practical considerations and challenges in real-world agricultural settings.
- Absence of deployment considerations on edge devices due to computational resource limitations.
- Acknowledgment of the need for future research to address real-world applications and challenges in the agricultural environment for a comprehensive solution.

7. Research Methodology

Diseases affecting plants critically endanger worldwide food supply, affecting both the quantity and quality of produce. In addressing this challenge, the research employs advanced machine learning techniques for the detection of diseases in plants. The primary objective is to develop a robust and accurate model capable of identifying various diseases affecting a diverse set of plants. The implementation focuses on leveraging the Plant Village dataset, a comprehensive collection comprising 54,309 images across 14 distinct plant species. These species include Apple, Blueberry, Cherry, Corn, Grape, Orange, Peach, Bell Pepper, Potato, Raspberry, Soybean, Squash, Strawberry, and Tomato.

Understanding the intricacies of plant diseases is crucial for timely intervention and effective agricultural management. Therefore, this research aims to contribute to the field by employing a meticulous approach to disease detection. The dataset encompasses images not only of diseased leaves but also healthy ones for a more comprehensive training process. The identification of 17 fungal diseases, 4 bacterial diseases, 2 mold (oomycete) diseases, 2 viral diseases, and a disease caused by a mite further enriches the dataset.

In this section, we delve into the detailed methodology employed in the research, covering dataset description, preprocessing techniques, the transformation process, the choice of models, and the evaluation metrics used to assess the model's performance. The following segments offer a detailed walkthrough of each phase within the methodology, clarifying the reasoning for the selected approaches in developing a successful plant disease detection system.

Flowchart outlining various stages of Machine Learning based plant disease detection process:

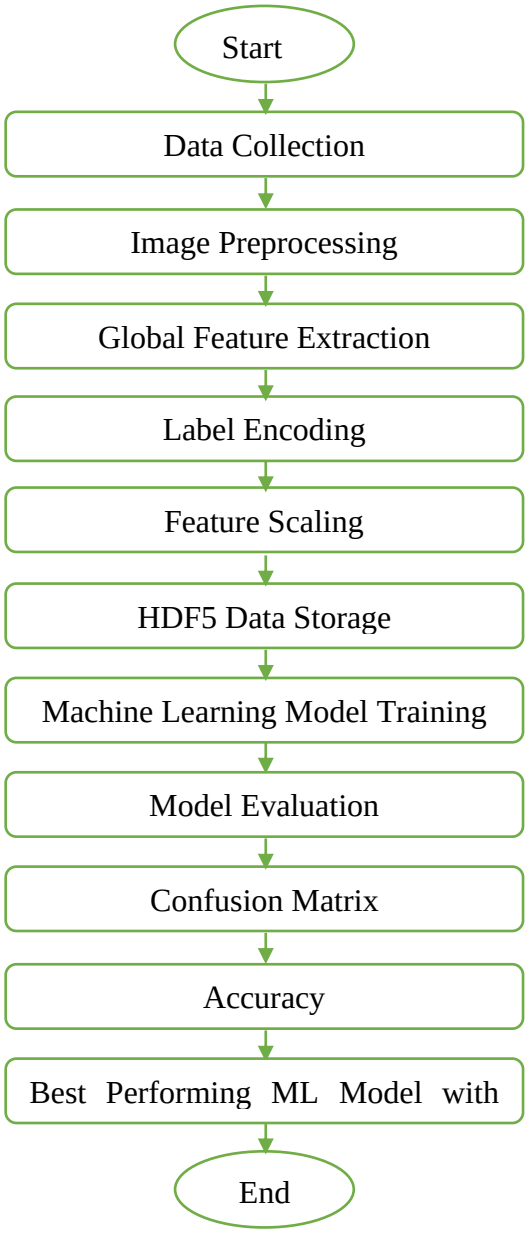


Figure 7-1: Stages of Machine Learning based *plant* disease detection process

7.1. Dataset Description

The Plant Village dataset is a comprehensive collection of 54,309 images covering 14 different plant species, including Apple, Blueberry, Cherry, Corn, Grape, Orange, Peach, Bell Pepper, Potato, Raspberry, Soybean, Squash, Strawberry, and Tomato. This dataset encompasses images depicting 17 fungal diseases, 4 bacterial diseases, 2 mold (oomycete) diseases, 2 viral diseases, and 1 disease caused by a mite. Additionally, images of healthy leaves, devoid of visible disease symptoms, are included for 12 plant species. (Hughes & Salathe, 2015)

Dataset Link: https://github.com/spMohanty/PlantVillage-Dataset

The dataset utilized in this project is sourced from the "Color" folder within the "raw" directory of the GitHub Repository. The data employed for modeling specifically focuses on Apple Leaves. The training dataset is structured into two folders: Diseased and Healthy. The Diseased Folder encompasses images of leaves affected by Apple Scab, Black Rot, or Cedar Apple Rust, labeled as diseased or unhealthy. On the other hand, the Healthy Folder contains images of green and healthy leaves.

Image Attributes

File Format: JPEG.

Resolution: 96 dpi (both vertical and horizontal).

Bit Depth: 24.

Height: 256 Pixels.

Width: 256 Pixels.

Size: 256 x 256 pixels.

7.2. Data Preprocessing

- **Data Collection**: A dataset containing 800 images per class (Diseased and Healthy) for various plant species is obtained from the plant village dataset. Each image is resized to 500x500 pixels to ensure computational efficiency, memory efficiency, model compatibility, and to reduce variability.
- **Color Format Conversion**: Images are converted from BGR to RGB format, which is a requirement for OpenCV, an image processing library in Python.
- **HSV Conversion**: RGB images are further converted to the HSV color space, separating luma (intensity) from chroma (color information), facilitating robustness to lighting changes and shadow removal.
- **Image Segmentation**: Green and brown color extraction are performed to isolate healthy and diseased regions of the plant leaves.
- **Global Feature Extraction**: Three types of global feature descriptors are extracted:
    - **Color**: Channel statistics (mean, standard deviation) and color histogram
    - **Shape**: Hu Moments
    - **Texture**: Haralick Texture

7.3. Transformation

- **Label Encoding:** Labels are encoded into numeric format for machine understanding.
- **Feature Scaling:** Min-Max scaling is applied to standardize feature magnitudes, ensuring all features contribute equally to model training.
- **Data Storage:** Extracted features and labels are saved in HDF5 files, an open-source file format supporting large, complex data structures.

7.4. Models

Seven machine learning models are chosen for training:

- Logistic Regression
- Linear Discriminant Analysis
- K Nearest Neighbors
- Decision Trees
- Random Forest
- Naïve Bayes
- Support Vector Machine

Details and explanations for each of these models are provided below:

7.4.1 Logistic Regression

Logistic regression is a fundamental statistical model used for binary classification tasks. It predicts the probability of a binary outcome (e.g., 0 or 1, healthy or diseased) based on a set of independent variables or features. The model's output is a value between 0 and 1, representing the likelihood of the positive outcome (e.g., diseased).

The logistic regression model is represented by the following equation:

$$P(y = 1|x) = 1 / (1 + \exp(-\beta^T x))$$

where:

- P (y = 1|x) is the probability of the positive outcome (diseased) given the input features (x)
- β is the vector of model coefficients
- x is the vector of input features
- exp is the exponential function
- T is the transpose operator

The model coefficients (β) represent the strength of the relationship between each input feature and the probability of the positive outcome. A positive coefficient indicates a positive correlation, while a negative coefficient indicates a negative correlation.

In the realm of identifying plant diseases, logistic regression is utilized to categorize images of leaves as healthy or afflicted, based on the analysis of features like color, texture, and form. The model coefficients would represent the relative importance of each feature in contributing to the disease prediction. The model's output would be the probability of a leaf being diseased given its extracted features.

7.4.2 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a dimensionality reduction technique that aims to transform high-dimensional data into a lower-dimensional subspace while preserving the maximum discrimination between classes. It finds a linear transformation that projects the data onto a lower-dimensional space where the separation between classes is maximized. This makes LDA particularly useful when dealing with a large number of features, as it reduces the computational complexity and improves classification accuracy.

The mathematical formulation of LDA is based on the concept of maximizing the ratio of between-class variance to within-class variance. This can be achieved by solving the following optimization problem:

maximize: W^T S_b W / W^T S_w W

where:

- W is the transformation matrix that projects the data onto the lower-dimensional subspace
- S_b is the between-class scatter matrix
- S_w is the within-class scatter matrix

The solution to this optimization problem yields the optimal transformation matrix W that projects the data onto a subspace where the separation between classes is maximized.

Interpretation of LDA

LDA can be interpreted as a technique that finds the optimal set of directions in the high-dimensional space that project the data onto a lower-dimensional subspace while preserving the maximum separation between classes. These directions are represented by the columns of the transformation matrix W.

For detecting plant diseases, Linear Discriminant Analysis (LDA) can be utilized to categorize images of leaves into healthy or diseased, based on analyzed features like color, texture, and shape. The optimal transformation matrix obtained from LDA would represent the most discriminating features for distinguishing between healthy and diseased leaves. This would allow for the development of accurate plant disease detection models using machine learning techniques.

LDA's ability to handle a large number of features and preserve class discrimination makes it a valuable tool for plant disease detection tasks. By reducing the dimensionality of the feature space, LDA can improve the efficiency and accuracy of machine learning models, leading to more effective plant disease detection and management practices.

7.4.3 K Nearest Neighbors (KNN)

K Nearest Neighbors (KNN) is a non-parametric, instance-based learning algorithm that classifies new data points based on the majority class of their k nearest neighbors in the feature space. It is a simple and versatile algorithm that is well-suited for plant disease detection due to its non-linearity and robustness to noise.

The KNN algorithm can be summarized as follows:

- **Choose the number of neighbors (k):** Select the number of nearest neighbors to consider for classification. The choice of k is crucial for the performance of the algorithm. A small value of k might lead to overfitting, while a large value of k might ignore important local patterns.
- **Calculate the distances:** For each new data point, calculate the distance to all data points in the training set. Common distance metrics include Euclidean distance, Manhattan distance, and Minkowski distance. The choice of distance metric depends on the specific characteristics of the data.
- **Identify the k nearest neighbors:** Select the k data points that are closest to the new data point. These k neighbors represent the most similar data points to the new data point in the feature space.
- **Assign the class label:** Assign the class label that is most common among the k nearest neighbors to the new data point. This majority voting approach determines the class of the new data point based on the classes of its nearest neighbors.

For detecting plant diseases, the K-Nearest Neighbors (KNN) algorithm can be employed to classify images of leaves as either healthy or afflicted, utilizing extracted characteristics such as color, texture, and shape. The k nearest neighbors of a new leaf image represents the most similar leaf images in the training set, providing insights into its potential health status. The class label assigned to the new leaf image based on the majority class of its k nearest neighbors determines its classification as healthy or diseased.

KNN's simplicity, non-linearity, robustness to noise, and versatility make it a valuable tool for plant disease detection. By leveraging the similarity between leaf images based on extracted features, KNN can effectively classify new leaf images and aid in identifying potential plant diseases.

7.4.4. Decision Trees

Decision Trees (DTs) are non-parametric, supervised learning algorithms that represent decision-making processes in a tree-like structure. They are widely used for classification and regression tasks due to their simplicity, interpretability, and robustness to noise.

A decision tree consists of nodes and branches. Each node represents a decision point, where the data is split based on a particular feature. The branches represent the possible outcomes of the decision. The terminal nodes, also known as leaf nodes, represent the final classification or regression value.

The construction of a decision tree involves recursively splitting the data into smaller subsets based on the most informative features. The algorithm iteratively selects the feature that best separates the data into classes or regresses the target variable. This process continues until the data points reach a state of purity, where all data points belong to the same class or have similar target values.

Common splitting criteria used in decision tree algorithms include:

- **Information gain:** Measures the reduction in entropy, or uncertainty, when the data is split based on a particular feature.
- **Gini impurity:** A measure of the probability of misclassifying a randomly chosen data point if it is assigned to the most frequent class in the current node.

- **Chi-squared test:** Measures the statistical significance of the association between a feature and the target variable.

Decision trees are prone to overfitting, which occurs when the tree becomes too complex and memorizes the training data instead of generalizing well to unseen data. Pruning techniques are used to prevent overfitting by removing unnecessary branches from the decision tree.

In the realm of identifying plant diseases, decision trees are an effective method for sorting leaf images into categories of healthy or diseased. This classification relies on analysing distinct features such as color, texture, and shape extracted from the leaf images. The decision tree would recursively split the data based on these features, creating a series of rules that effectively capture the characteristics of healthy and diseased leaves.

Decision trees are well-suited for plant disease detection due to their simplicity, interpretability, and robustness to noise. By providing a clear understanding of the decision-making process, decision trees can aid in identifying critical features for disease detection and developing effective disease management strategies.

7.4.5. Random Forest

Random Forest (RF) is an ensemble learning method that combines multiple decision trees to improve the overall classification or regression performance. It is widely used for classification and regression tasks due to its robustness to noise, ability to handle complex relationships, and out-of-the-box feature importance measures.

Ensemble learning is a technique that combines multiple models to improve the overall performance of a single model. By aggregating the predictions of multiple models, ensemble learning can reduce the variance and bias of individual models, leading to more robust and accurate predictions.

The Random Forest algorithm consists of the following steps:

- **Random Subsampling:** For each tree in the forest, a random subset of the training data is drawn. This technique, known as bootstrapping, reduces the correlation between trees and helps prevent overfitting.

- **Random Feature Selection:** At each split node, a random subset of features is considered for splitting. This technique, known as random feature selection, further decorrelates the trees and reduces the risk of overfitting.
- **Tree Construction:** Each tree is constructed using the chosen random subset of data and features. The tree is grown until a predefined stopping criterion is met, such as a maximum depth or minimum node size.
- **Majority Voting:** For classification tasks, the class prediction of a new data point is determined by the majority vote of the trees in the forest. For regression tasks, the average prediction of the trees is used.

In the realm of detecting plant diseases, the Random Forest algorithm is utilized to distinguish between healthy and diseased leaf images. The ensemble of decision trees captures complex relationships between these features and the disease status, leading to robust and accurate disease detection.

Random Forest's ability to handle noisy data, capture non-linear relationships, and provide feature importance makes it a valuable tool for plant disease detection. By leveraging the collective wisdom of multiple trees, Random Forest can effectively classify leaf images and aid in identifying potential plant diseases.

7.4.6 Naïve Bayes

Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem. It assumes that features are independent of each other, which is often an oversimplification of real-world data. However, despite this assumption, Naive Bayes often performs well in practice due to its simplicity and efficiency.

Bayes' theorem is a fundamental concept in probability theory that allows us to update our beliefs about a hypothesis given new evidence. It can be expressed as:

P(H | E) = (P(E | H) * P(H)) / P(E)

where:

- P(H) is the prior probability of the hypothesis.
- P(E) is the probability of the evidence.

- P(E | H) is the probability of the evidence given the hypothesis

The Naive Bayes classifier applies Bayes' theorem to classification tasks. It assumes that the features of the data are independent of each other, which means that the probability of one feature does not affect the probability of another feature given the class label. This assumption is often oversimplified, but it allows for a simple and efficient implementation of the algorithm.

The Naive Bayes algorithm can be summarized as follows:

- **Compute the prior probabilities:** Calculate the probability of each class (e.g., healthy or diseased) in the training data.
- **Calculate the conditional probabilities:** For each feature and class, calculate the probability of the feature given the class.
- **Apply Bayes' theorem:** Determine the likelihood of each category for a new data point by applying Bayes' theorem to the features of the data point.
- **Assign the class label:** Assign the class label that has the highest probability according to Bayes' theorem.

In plant disease detection, the Naive Bayes classifier is adept at segregating leaf images into healthy or diseased categories, utilizing key extracted attributes like color, texture, and shape. The independence assumption of Naive Bayes may not hold perfectly for these features, but it can still provide a reasonable approximation of the true relationships.

Naive Bayes' simplicity, efficiency, and robustness to noise make it a valuable tool for plant disease detection. By leveraging the probabilistic framework of Bayes' theorem, Naive Bayes can effectively classify leaf images and aid in identifying potential plant diseases.

7.4.7. Support Vector Machine (SVM)

Support Vector Machines (SVMs) are a powerful set of machine learning algorithms used for classification and regression tasks. They are based on the principle of finding a hyperplane that maximizes the margin between two classes of data points. This hyperplane effectively separates the data points into their respective classes, allowing for accurate classification of new data points.

A hyperplane is a decision boundary in a high-dimensional space that divides the data points into two classes. The margin refers to the gap between the hyperplane and the nearest data points from each class. An expanded margin signifies enhanced class separation, leading to improved classification outcomes.

SVMs aim to find the optimal hyperplane that maximizes the margin between the two classes. This optimization problem ensures that the chosen hyperplane generalizes well to unseen data and achieves high classification accuracy.

SVMs are originally designed for linearly separable data, meaning that the data points can be perfectly separated by a straight-line hyperplane. However, real-world data is often non-linear, requiring more sophisticated techniques to find an effective hyperplane. Kernel functions are introduced to map the data into a higher-dimensional space where a linear hyperplane can be used to separate the data points.

Common kernel functions used in SVMs include:

- Linear kernel: Suitable for linearly separable data.
- Polynomial kernel: Maps data into a polynomial space, handling non-linear relationships.
- Radial basis function (RBF) kernel: A versatile kernel that works well for various data types.

In the field of detecting plant diseases, Support Vector Machines (SVMs) can be used for classifying leaf images into healthy or diseased categories. This classification relies on the analysis of extracted features including color, texture, and shape. The SVM algorithm effectively finds a hyperplane that separates healthy leaf images from diseased ones, allowing for accurate classification of new leaf images.

SVMs' robustness to outliers, ability to handle high-dimensional data, and effectiveness in non-linear settings make them a valuable tool for plant disease detection. By leveraging the concept of maximizing the margin, SVMs can effectively differentiate between healthy and diseased leaf images and aid in identifying potential plant diseases.

7.5 Evaluation Metrics

In plant-based disease detection, evaluation metrics such as precision, recall, F1-score, and support are used to measure the performance of the model. These criteria determine the model's effectiveness in accurately identifying disease presence or absence within an image.

- **Precision** is the ratio of true positives to the total number of positive predictions. It measures the proportion of true positives among all the positive predictions made by the model. A high precision score indicates that the model is making fewer false positive predictions.
- **Recall** is the ratio of true positives to the total number of actual positives. It measures the proportion of true positives among all the actual positive cases. A high recall score indicates that the model is making fewer false negative predictions.
- **F1-score** is the harmonic mean of precision and recall. It provides a balance between precision and recall, giving an overall measure of the model's performance. The F1-score is useful when you want to consider both false positives and false negatives simultaneously. It is commonly used when you want to evaluate the overall performance of a classifier.
- **Support** is the number of occurrences of each class in the actual data. It is used to calculate precision, recall, and F1-score for each class. The support score is useful when you want to evaluate the performance of the model on each class separately.

In detecting diseases in plants, these metrics assess how effectively the model identifies the existence or non-existence of a disease in a particular image. For example, the F1-score serves as a comprehensive measure of the model's capability in diagnosing various plant diseases. The precision and recall scores can be used to evaluate the model's performance on each type of disease separately. The support score can be used to evaluate the model's performance on each class based on the number of occurrences in the actual data.

7.6. Cross Validation

To assess the robustness of the models and prevent overfitting, a tenfold cross-validation approach was utilized. This approach divides the dataset into ten folds, trains the model on nine folds, and evaluates its performance on the remaining fold. The process is repeated ten times, providing a comprehensive evaluation of the models' ability to generalize to unseen data.

7.7. Selection of the Best Model

Out of the seven selected models, the one exhibiting outstanding performance and attaining remarkable accuracy will be recognized as the most effective model for detecting plant-based diseases.

8. Requirements Resources

8.1. Hardware Requirements

- A laptop or desktop computer with a minimum of 8GB RAM, an Intel i5 processor with 4 or more cores, and an integrated Graphics Processing Unit (GPU) is necessary for the optimal functioning of the system.

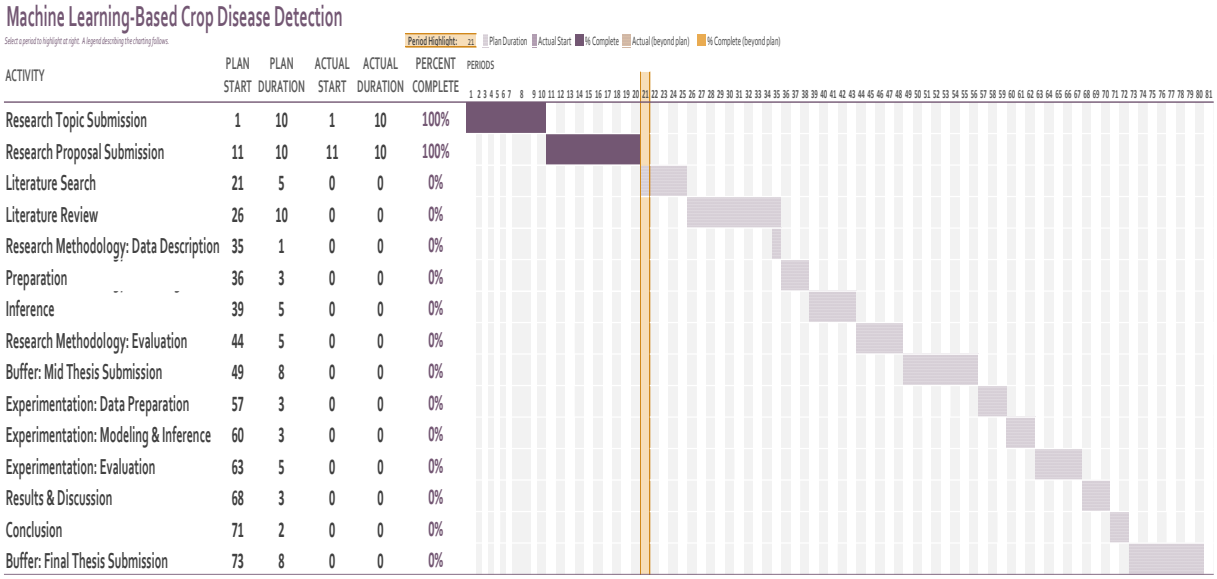
8.2. Software Requirements

- Python programming language for code implementation (version 3.9.12).
- Web browser for general access and research.
- NumPy: Fundamental package for array operations in Python.
- Mahotas: Computer vision and image processing library used for Haralick texture feature extraction.
- OpenCV: Library for computer vision tasks, including image and video processing.
- scikit-learn: Machine learning library providing simple tools for data analysis and modelling.
- seaborn: Data visualization library based on Matplotlib for statistical graphics.
- h5py: Python library for storing and managing large datasets in Hierarchical Data Format.
- os: Provides a way of interacting with the operating system.
- glob: Finds all pathnames matching a specified pattern.
- matplotlib: Comprehensive library for creating static, animated, and interactive visualizations in Python.
- Integrated Development Environment (IDE) such as Jupyter Notebook or Visual Studio Code.

8.3 Dataset Requirements

- Plant Village Dataset Link: https://github.com/spMohanty/PlantVillage-Dataset

9. Research Plan



ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Research Topic Submission	1	10	1	10	100%
Research Proposal Submission	11	10	11	10	100%
Literature Search	21	5	0	0	0%
Literature Review	26	10	0	0	0%
Research Methodology: Data Description	35	1	0	0	0%
Preparation	36	3	0	0	0%
Inference	39	5	0	0	0%
Research Methodology: Evaluation	44	5	0	0	0%
Buffer: Mid Thesis Submission	49	8	0	0	0%
Experimentation: Data Preparation	57	3	0	0	0%
Experimentation: Modeling & Inference	60	3	0	0	0%
Experimentation: Evaluation	63	5	0	0	0%
Results & Discussion	68	3	0	0	0%
Conclusion	71	2	0	0	0%
Buffer: Final Thesis Submission	73	8	0	0	0%

References

Agarwal, M., Singh, A., Arjaria, S., Sinha, A. & Gupta, S. (2020). ToLeD: Tomato Leaf Disease Detection using Convolution Neural Network. *Procedia Computer Science*. [Online]. 167 (2019). p.pp. 293–301. Available from: https://doi.org/10.1016/j.procs.2020.03.225.

Aggarwal, M., Khullar, V. & Goyal, N. (2023). Exploring Classification of Rice Leaf Diseases using Machine Learning and Deep Learning. In: *Proceedings of 2023 3rd International Conference on Innovative Practices in Technology and Management, ICIPTM 2023*. 2023, Institute of Electrical and Electronics Engineers Inc.

Ahmed, I. & Yadav, P.K. (2022). Plant disease detection using machine learning approaches. *Expert Systems*. (April 2022). p.pp. 1–16.

Al-gaashani, M.S.A.M., Shang, F., Muthanna, M.S.A., Khayyat, M. & Abd El-Latif, A.A. (2022). Tomato leaf disease classification by exploiting transfer learning and feature concatenation. *IET Image Processing*. 16 (3). p.pp. 913–925.

Astani, M., Hasheminejad, M. & Vaghefi, M. (2022). A diverse ensemble classifier for tomato disease recognition. *Computers and Electronics in Agriculture*. 198.

Atila, Ü., Uçar, M., Akyol, K. & Uçar, E. (2021). Plant leaf disease classification using EfficientNet deep learning model. *Ecological Informatics*. [Online]. 61 (September 2020). p.p. 101182. Available from: https://doi.org/10.1016/j.ecoinf.2020.101182.

Aurangzeb, K., Akmal, F., Attique Khan, M., Sharif, M. & Javed, M.Y. (2020). Advanced Machine Learning Algorithm Based System for Crops Leaf Diseases Recognition. In: *Proceedings - 2020 6th Conference on Data Science and Machine Learning Applications, CDMA 2020*. 1 March 2020, Institute of Electrical and Electronics Engineers Inc., pp. 146–151.

Baheti, H., Thakare, A., Bhople, Y., Darekar, S. & Dodmani, O. (2022). Machine Learning Algorithm for Detection And Classification of Tomato Plant Leaf Disease. In: *2022 IEEE 7th International conference for Convergence in Technology, I2CT 2022*. 2022, Institute of Electrical and Electronics Engineers Inc.

Bhagat, M. & Kumar, D. (2022). A comprehensive survey on leaf disease identification & classification. *Multimedia Tools and Applications*. 81 (23). p.pp. 33897–33925.

Bhosale, J.D., Thorat, S.S., Pancholi, P.V. & Mutkule, P.R. (2023). Machine Learning-Based Algorithms for the Detection of Leaf Disease in Agriculture Crops. *International Journal on Recent and Innovation Trends in Computing and Communication*. 11. p.pp. 45–50.

Chaitanya Reddy, P., Chandra, R.M.S., Vadiraj, P., Ayyappa Reddy, M., Mahesh, T.R. &

Sindhu Madhuri, G. (2021). Detection of Plant Leaf-based Diseases Using Machine Learning Approach. In: *CSITSS 2021 - 2021 5th International Conference on Computational Systems and Information Technology for Sustainable Solutions, Proceedings*. 2021, Institute of Electrical and Electronics Engineers Inc.

Chowdhury, M.E.H., Rahman, T., Khandakar, A., Ayari, M.A., Khan, A.U., Khan, M.S., Al-Emadi, N., Reaz, M.B.I., Islam, M.T. & Ali, S.H.M. (2021). Automatic and Reliable Leaf Disease Detection Using Deep Learning Techniques. *AgriEngineering*. 3 (2). p.pp. 294–312.

Chug, A., Bhatia, A., Singh, A.P. & Singh, D. (2023). A novel framework for image-based plant disease detection using hybrid deep learning approach. *Soft Computing*. [Online]. 27 (18). p.pp. 13613–13638. Available from: https://doi.org/10.1007/s00500-022-07177-7.

Daniya, T. & Vigneshwari, S. (2019). A review on machine learning techniques for rice plant disease detection in agricultural research. *International Journal of Advanced Science and Technology*. 28 (13). p.pp. 49–62.

Domingues, T., Brandão, T. & Ferreira, J.C. (2022). Machine Learning for Detection and Prediction of Crop Diseases and Pests: A Comprehensive Survey. *Agriculture (Switzerland)*. 12 (9). p.pp. 1–23.

Gosai, D., Kaka, B., Garg, D., Patel, R. & Ganatra, A. (2022). Plant Disease Detection and Classification Using Machine Learning Algorithm. In: *2022 International Conference for Advancement in Technology, ICONAT 2022*. 2022, Institute of Electrical and Electronics Engineers Inc.

Habiba, S.U. & Islam, M.K. (2021). Tomato Plant Diseases Classification Using Deep Learning Based Classifier from Leaves Images. In: *2021 International Conference on Information and Communication Technology for Sustainable Development, ICICT4SD 2021 - Proceedings*. 27 February 2021, Institute of Electrical and Electronics Engineers Inc., pp. 82–86.

Harakannanavar, S.S., Rudagi, J.M., Puranikmath, V.I., Siddiqua, A. & Pramodhini, R. (2022). Plant leaf disease detection using computer vision and machine learning algorithms. *Global Transitions Proceedings*. 3 (1). p.pp. 305–310.

Hatuwal, B.K., Shakya, A. & Joshi, B. (2020). Plant Leaf Disease Recognition Using Random Forest, KNN, SVM and CNN. *Polibits*. [Online]. 62 (May). p.pp. 13–19. Available from: https://doi.org/10.17562/PB-62-2.

Hughes, D.P. & Salathe, M. (2015). *An open access repository of images on plant health to enable the development of mobile disease diagnostics*. [Online]. Available from:

http://arxiv.org/abs/1511.08060.

Jackulin, C. & Murugavalli, S. (2022). A comprehensive review on detection of plant disease using machine learning and deep learning approaches. *Measurement: Sensors*. [Online]. 24 (June). p.p. 100441. Available from: https://doi.org/10.1016/j.measen.2022.100441.

Jasim, M.A. & Al-Tuwaijari, J.M. (2020). Plant Leaf Diseases Detection and Classification Using Image Processing and Deep Learning Techniques. *Proceedings of the 2020 International Conference on Computer Science and Software Engineering, CSASE 2020*. p.pp. 259–265.

Kholiya, D., Mishra, A.K., Dumka, A., Pandey, N.K. & Tripathi, N. (2023). Detection of Leaf Diseases in Agricultural Plants Using Machine Learning. In: *2023 International Conference on Computer, Electronics and Electrical Engineering and their Applications, IC2E3 2023*. 2023, Institute of Electrical and Electronics Engineers Inc.

Kilaru, R. & Raju, K.M. (2022). Prediction of Maize Leaf Disease Detection to improve Crop Yield using Machine Learning based Models. In: *4th International Conference on Recent Trends in Computer Science and Technology, ICRTCST 2021 - Proceedings*. 2022, Institute of Electrical and Electronics Engineers Inc., pp. 212–217.

Kumar, R., Shukla, N. & Princee (2022). Plant Disease Detection and Crop Recommendation Using CNN and Machine Learning. In: *2022 International Mobile and Embedded Technology Conference, MECON 2022*. 2022, Institute of Electrical and Electronics Engineers Inc., pp. 168–172.

Lamba, M., Gigras, Y. & Dhull, A. (2021). Classification of plant diseases using machine and deep learning. *Open Computer Science*. 11 (1). p.pp. 491–508.

Masazhar, A.N.I. & Kamal, M.M. (2017). Digital image processing technique for palm oil leaf disease detection using multiclass SVM classifier. *2017 IEEE International Conference on Smart Instrumentation, Measurement and Applications, ICSIMA 2017*. 2017-Novem (November). p.pp. 1–6.

Meenakshi, T. (2023). Automatic Detection of Diseases in Leaves of Medicinal Plants Using Modified Logistic Regression Algorithm. *Wireless Personal Communications*. [Online]. 131 (4). p.pp. 2573–2597. Available from: https://doi.org/10.1007/s11277-023-10555-5.

Mishra, S., Sachan, R. & Rajpal, D. (2020). Deep Convolutional Neural Network based Detection System for Real-time Corn Plant Disease Recognition. *Procedia Computer Science*. [Online]. 167. p.pp. 2003–2010. Available from: https://doi.org/10.1016/j.procs.2020.03.236.

Mohanty, R., Wankhede, P., Singh, D. & Vakhare, P. (2022). Tomato Plant Leaves Disease

Detection using Machine Learning. *2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*. (Icaaic). p.pp. 544–549.

Ngugi, L.C., Abelwahab, M. & Abo-Zahhad, M. (2021). Recent advances in image processing techniques for automated leaf pest and disease recognition – A review. *Information Processing in Agriculture*. [Online]. 8 (1). p.pp. 27–51. Available from: https://doi.org/10.1016/j.inpa.2020.04.004.

Pawar, S., Shedge, S., Panigrahi, N., Jyoti, A.P., Thorave, P. & Sayyad, S. (2022). Leaf Disease Detection of Multiple Plants Using Deep Learning. In: *2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing, COM-IT-CON 2022*. 2022, Institute of Electrical and Electronics Engineers Inc., pp. 241–245.

Ramiah Subburaj, S.D., Vaithyam Rengarajan, V.K. & Palaniswamy, S. (2023). Transfer Learning based Image Classification of Diseased Tomato Leaves with Optimal Fine-Tuning combined with Heat Map Visualization. *Tarim Bilimleri Dergisi*. 29 (4). p.pp. 1003–1017.

Rizvee, R.A., Orpa, T.H., Ahnaf, A., Kabir, M.A., Ahmmad Rashid, M.R., Islam, M.M., Islam, M., Jabid, T. & Ali, M.S. (2023). LeafNet: A proficient convolutional neural network for detecting seven prominent mango leaf diseases. *Journal of Agriculture and Food Research*. [Online]. 14 (August). p.p. 100787. Available from: https://doi.org/10.1016/j.jafr.2023.100787.

Sai, A.M. & Patil, N. (2022). Comparative Analysis of Machine Learning Algorithms for Disease Detection in Apple Leaves. In: *2022 IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics, DISCOVER 2022 - Proceedings*. 2022, Institute of Electrical and Electronics Engineers Inc., pp. 239–244.

Sakhamuri, S. & Kompalli, V.S. (2020). An Overview on Prediction of Plant Leaves Disease using Image Processing Techniques. *IOP Conference Series: Materials Science and Engineering*. 981 (2). p.pp. 10–16.

Sangeetha, K., Vishnu Raja, P., Rima, P., Pranesh Kumar, M. & Preethees, S. (2022). Apple Leaf Disease Detection using Deep Learning. In: *Proceedings - 6th International Conference on Computing Methodologies and Communication, ICCMC 2022*. 2022, Institute of Electrical and Electronics Engineers Inc., pp. 1063–1067.

Shakeel, W., Ahmad, M. & Mahmood, N. (2020). Early Detection of Cercospora Cotton Plant Disease by Using Machine Learning Technique. In: *30th International Conference on Computer Theory and Applications, ICCTA 2020 - Proceedings*. 12 December 2020, Institute of Electrical and Electronics Engineers Inc., pp. 44–48.

Sharma, V., Mir, A.A. & Sarwr, D.A. (2020). Detection of Rice Disease Using Bayes' Classifier and Minimum Distance Classifier. *Journal of Multimedia Information System*. 7 (1). p.pp. 17–24.

Shivaprasad, K. & Wadhawan, A. (2023). Deep Learning-based Plant Leaf Disease Detection. In: *Proceedings of the 7th International Conference on Intelligent Computing and Control Systems, ICICCS 2023*. 2023, Institute of Electrical and Electronics Engineers Inc., pp. 360–365.

Shrivastava, V.K. & Pradhan, M.K. (2021). Rice plant disease classification using color features: a machine learning paradigm. *Journal of Plant Pathology*. 103 (1). p.pp. 17–26.

Silviya, S.H.A., Sriman, B., Shamini, P.B., Elangovan, A., Monica, A.R. & Keerthana, N. V. (2022). Deep Learning based Plant Leaf Disease Detection and Classification. In: *4th International Conference on Inventive Research in Computing Applications, ICIRCA 2022 - Proceedings*. 2022, Institute of Electrical and Electronics Engineers Inc., pp. 702–710.

Singla, R.S., Gupta, A., Gupta, R., Tripathi, V., Naruka, M.S. & Awasthi, S. (2023). Plant Disease Classification Using Machine Learning. In: *2023 International Conference on Disruptive Technologies, ICDT 2023*. 2023, IEEE, pp. 409–413.

Sujatha, R., Chatterjee, J.M., Jhanjhi, N.Z. & Brohi, S.N. (2021). Performance of deep learning vs machine learning in plant leaf disease detection. *Microprocessors and Microsystems*. 80.

Trivedi, N.K., Gautam, V., Anand, A., Aljahdali, H.M., Villar, S.G., Anand, D., Goyal, N. & Kadry, S. (2021). Early detection and classification of tomato leaf disease using high-performance deep neural network. *Sensors*. 21 (23).

Vishnoi, V.K., Kumar, K. & Kumar, B. (2021). *Plant disease detection using computational intelligence and image processing*. [Online]. Springer Berlin Heidelberg. Available from: https://doi.org/10.1007/s41348-020-00368-0.

Wójtowicz, A., Piekarczyk, J., Czernecki, B. & Ratajkiewicz, H. (2021). A random forest model for the classification of wheat and rye leaf rust symptoms based on pure spectra at leaf scale. *Journal of Photochemistry and Photobiology B: Biology*. 223 (August).

Yousuf, A. & Khan, U. (2021). Ensemble Classifier for Plant Disease Detection. *International Journal of Computer Science and Mobile Computing*. 10 (1). p.pp. 14–22.

Zhong, Y. & Zhao, M. (2020). Research on deep learning in apple leaf disease recognition. *Computers and Electronics in Agriculture*. [Online]. 168 (October 2019). p.p. 105146. Available from: https://doi.org/10.1016/j.compag.2019.105146.

APPENDIX C: PYTHON CODE FOR IMAGE CLASSIFICATION – ML MODELS

```
# Import necessary libraries and modules for data manipulation, image processing, machine learning, and plotting
import os  # For interacting with the operating system
import numpy as np  # For numerical operations
import cv2  # For image processing tasks
import h5py  # For interacting with HDF5 binary data format
import warnings  # For handling warnings
import matplotlib.pyplot as plt  # For plotting and visualization
from sklearn.preprocessing import LabelEncoder, MinMaxScaler  # For data preprocessing
from sklearn.model_selection import train_test_split, KFold, StratifiedKFold, cross_val_score, GridSearchCV  # For model evaluation and selection
from sklearn.metrics import (
    confusion_matrix,
    accuracy_score,
    classification_report,
    f1_score,
    precision_score,
    recall_score,
    roc_curve,
    auc,
    average_precision_score,
    precision_recall_curve
)  # For model performance evaluation
from sklearn.linear_model import LogisticRegression  # Logistic regression classifier
from sklearn.tree import DecisionTreeClassifier  # Decision tree classifier
from sklearn.ensemble import RandomForestClassifier  # Random forest classifier
from sklearn.neighbors import KNeighborsClassifier  # K-Nearest Neighbors classifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis  # Linear Discriminant Analysis classifier
from sklearn.naive_bayes import GaussianNB  # Gaussian Naive Bayes classifier
from sklearn.svm import SVC  # Support Vector Machine classifier
from tensorflow.keras.preprocessing.image import ImageDataGenerator  # For augmenting image data
import mahotas  # For computer vision and image processing tasks
import seaborn as sns  # For advanced visualization
import pandas as pd  # For data manipulation and analysis
from imblearn.over_sampling import SMOTE  # For handling imbalanced datasets

# Ignore Warnings
warnings.filterwarnings("ignore")  # Suppress warnings to clean up output

# Set tunable parameters for the image processing and classification pipeline
images_per_class = 800  # Define the number of images to use for each class
fixed_size = tuple((500, 500))  # Set a fixed image size for all images to ensure consistency
train_test_path = r"C:\Users\gsanthos\Downloads\Plant-Disease-Detection-master\image_classification\dataset - Copy\train"  # Path to the training dataset
```

```
output_dir = r"C:\Users\gsanthos\Downloads\Plant-Disease-Detection-master\image_classification\output"  # Directory to save processed data and outputs
h5_train_data = os.path.join(output_dir, "train_data.h5")  # Path to save the feature vectors (HDF5 format) for training data
h5_train_labels = os.path.join(output_dir, "train_labels.h5")  # Path to save the labels (HDF5 format) for training data
bins = 8  # Number of bins for histogram-based features (e.g., color histogram)
test_size = 0.30  # Proportion of the dataset to be used as test set
seed = 9  # Seed for random operations to ensure reproducibility

# Define a function to convert an image from BGR color space (used by OpenCV) to RGB color space (used by matplotlib and most image display libraries)
def bgr_rgb(image):
    # Use OpenCV's cvtColor function to convert the color space from BGR to RGB
    return cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Define a function to convert an image from RGB color space to HSV color space
def rgb_hsv(image):
    # Use OpenCV's cvtColor function to change the color space of the image from RGB to HSV
    return cv2.cvtColor(image, cv2.COLOR_RGB2HSV)

# Define a function for segmenting images to isolate green (healthy) and brown (diseased) areas
def img_segmentation(rgb_img, hsv_img):
    # Define the HSV range for green color (typically representing healthy plant parts)
    lower_green = np.array([25, 0, 20])
    upper_green = np.array([100, 255, 255])
    # Create a mask that isolates the green areas within the specified range
    healthy_mask = cv2.inRange(hsv_img, lower_green, upper_green)
    # Apply the mask to the original RGB image to extract the green areas
    result = cv2.bitwise_and(rgb_img, rgb_img, mask=healthy_mask)

    # Define the HSV range for brown color (typically representing diseased plant parts)
    lower_brown = np.array([10, 0, 10])
    upper_brown = np.array([30, 255, 255])
    # Create a mask that isolates the brown areas within the specified range
    disease_mask = cv2.inRange(hsv_img, lower_brown, upper_brown)
    # Apply the mask to the original RGB image to extract the brown areas
    disease_result = cv2.bitwise_and(rgb_img, rgb_img, mask=disease_mask)

    # Combine the healthy and disease masks to create a final mask
    final_mask = healthy_mask + disease_mask
    # Apply the final mask to the original RGB image to segment out both healthy and diseased areas
    segmented_image = cv2.bitwise_and(rgb_img, rgb_img, mask=final_mask)

    # Return the segmented image with both green and brown areas isolated
    return segmented_image
```

```
# Define a function to extract Hu Moments as a feature descriptor from an image
def fd_hu_moments(image):
    # Convert the image from BGR to grayscale; Hu Moments are calculated on grayscale images
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    # Calculate the moments of the grayscale image
    moments = cv2.moments(gray_image)
    # Compute Hu Moments from the moments, which are invariant to image transformations such as scaling, rotation, and translation
    feature_vector = cv2.HuMoments(moments).flatten()
    # Return the flattened array of Hu Moments as the feature vector
    return feature_vector

# Define a function to extract Haralick texture features as a feature descriptor from an image
def fd_haralick(image):
    # Convert the input image from BGR color space to grayscale since Haralick texture features are extracted from grayscale images
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Compute Haralick texture features using the mahotas library, which calculates these features based on the co-occurrence matrix of the grayscale image
    # The mean of the Haralick features is taken across different orientations of the co-occurrence matrix to ensure rotation invariance
    haralick_features = mahotas.features.haralick(gray_image).mean(axis=0)

    # Return the computed Haralick texture features as the feature vector
    return haralick_features

# Define a function to extract color histogram features as a feature descriptor from an image
def fd_histogram(image):
    # Convert the input image from BGR color space to HSV color space to capture color information more effectively
    hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    # Calculate the color histogram for the HSV image using OpenCV's calcHist function
    # The histogram is calculated for all three channels (Hue, Saturation, Value) with 'bins' number of bins for each channel
    # The histogram range for each channel is set from 0 to 256
    hist = cv2.calcHist([hsv_image], [0, 1, 2], None, [bins, bins, bins], [0, 256, 0, 256, 0, 256])

    # Normalize the histogram to ensure that the feature vector's magnitude does not affect the classifier's performance
    cv2.normalize(hist, hist)

    # Convert the 3D histogram into a 1D array (flattened) to create a feature vector that can be used by machine learning algorithms
    feature_vector = hist.flatten()

    # Return the flattened color histogram as the feature vector
    return feature_vector
```

```
# Retrieve and organize the training labels from the specified training dataset directory
train_labels = os.listdir(train_test_path)  # List all the directory names in the training dataset path, each representing a label/class
train_labels.sort()  # Sort the labels in alphabetical order to maintain consistency
train_labels.reverse()  # Reverse the sorted list to have it in descending order, if needed for specific ordering requirements
print(train_labels)  # Display the sorted and reversed list of training labels for verification and debugging purposes

# Initialize empty lists for storing extracted feature vectors and corresponding labels of the images
global_features = []  # This list will contain all the feature vectors extracted from the images, where each feature vector represents a set of features describing an image
labels = []  # This list will hold the labels associated with each image, indicating the class/category to which each image belongs

# Iterate through each sub-folder representing a class in the training dataset
for training_name in train_labels:
    # Construct the directory path to the current class's images
    dir = os.path.join(train_test_path, training_name)
    # Set the current class label to the name of the sub-folder
    current_label = training_name

    # Loop through the images in the current class's folder, limited by images_per_class
    for x in range(1, images_per_class + 1):
        # Construct the file path for each image based on its sequence number
        file = dir + "/" + str(x) + ".jpg"

        # Skip processing if the image file does not exist
        if not os.path.exists(file):
            continue

        # Read the image from the file
        image = cv2.imread(file)

        # Skip processing if the image is not successfully loaded
        if image is None:
            continue

        # Resize the image to a fixed size to ensure consistency
        image = cv2.resize(image, fixed_size)

        # Preprocess the image by applying various image processing functions
        BGR_RGB = bgr_rgb(image)  # Convert BGR to RGB
        RGB_HSV = rgb_hsv(BGR_RGB)  # Convert RGB to HSV for color segmentation
        IMG_SEGMENT = img_segmentation(BGR_RGB, RGB_HSV)  # Segment the image to isolate relevant features

        # Extract feature vectors from the segmented image using different descriptors
```

```
        fv_hu_moments = fd_hu_moments(IMG_SEGMENT)  # Hu Moments for shape features
        fv_haralick = fd_haralick(IMG_SEGMENT)  # Haralick texture features
        fv_histogram = fd_histogram(IMG_SEGMENT)  # Color histogram features

        # Combine all feature vectors into a single global feature vector for the image
        global_feature = np.hstack([fv_histogram, fv_haralick, fv_hu_moments])

        # Append the current image's feature vector and label to the global lists
        labels.append(current_label)
        global_features.append(global_feature)

    # Print the status indicating the completion of processing for the current class's folder
    print("[STATUS] processed folder: {}".format(current_label))

print("[STATUS] Completed Global Feature Extraction...")

# Get the overall feature vector size
print("[STATUS] Feature Vector Size {}".format(np.array(global_features).shape))

# get the overall training label size
print("[STATUS] Training Labels {}".format(np.array(labels).shape))

# Encode the categorical class labels into numerical values for model training

# Identify and print the unique class labels found in the 'labels' list
targetNames = np.unique(labels)
print(targetNames)

# Initialize a LabelEncoder object from scikit-learn to convert string labels to integers
le = LabelEncoder()

# Fit the label encoder to the class labels and transform them into corresponding integer
values
target = le.fit_transform(labels)

# Log the completion status of the label encoding process
print("[STATUS] Training Labels are Encoded...")

# Normalize the extracted feature vectors to have values in the range (0-1)

# Initialize a MinMaxScaler object from scikit-learn to scale features to the specified range
scaler = MinMaxScaler(feature_range=(0, 1))

# Fit the scaler to the global features extracted from the images and transform them to the
range (0, 1)
rescaled_features = scaler.fit_transform(global_features)

# Log the completion status of the feature vector normalization process
print("[STATUS] Feature Vector Normalized...")
```

```
print("[STATUS] Target Labels Shape: {}".format(target.shape))

# Persist the normalized feature vectors and corresponding labels to disk using the HDF5
format for efficient storage and retrieval

# Open or create an HDF5 file to store the feature vectors and use 'w' to write data
h5f_data = h5py.File(h5_train_data, 'w')
# Create a dataset within the HDF5 file to store the normalized feature vectors, naming it
'dataset_1'
h5f_data.create_dataset('dataset_1', data=np.array(rescaled_features))

# Similarly, open or create another HDF5 file to store the encoded labels
h5f_label = h5py.File(h5_train_labels, 'w')
# Create a dataset within this HDF5 file for the labels, also naming it 'dataset_1'
h5f_label.create_dataset('dataset_1', data=np.array(target))

# Close the HDF5 files to ensure data is written to disk and resources are properly released
h5f_data.close()
h5f_label.close()

# Initialize lists to store the evaluation metrics and curves for each model used in the
classification task

model_names = []  # Store the names of the models used for easy identification
training_accuracies = []  # Store the training accuracies of each model
test_accuracies = []  # Store the test accuracies achieved by each model on the unseen data
cv_scores = []  # Store cross-validation scores to assess the models' performance stability
across different data splits
f1_scores = []  # Store F1 scores as a balanced measure of precision and recall for each model
precisions = []  # Store precision values to evaluate the models' ability to identify only
relevant instances
recalls = []  # Store recall values to assess the models' capability to identify all relevant
instances
roc_auc_values = []  # Store the Area Under the Curve (AUC) values from the Receiver
Operating Characteristic (ROC) curves as a measure of model discriminability
roc_curves = []  # Store ROC curve data points for plotting and comparison purposes
pr_curves = []  # Store Precision-Recall (PR) curve data points to evaluate the models'
performance in the context of class imbalance

# Initialize a list to hold various classification models along with their configurations for
comparative analysis

models = []

# Append a tuple containing the model abbreviation and the Logistic Regression model with a
fixed random state for reproducibility
models.append(('LR', LogisticRegression(random_state=seed)))

# Append a tuple containing the model abbreviation and the Linear Discriminant Analysis
model
```

```
models.append(('LDA', LinearDiscriminantAnalysis()))

# Append a tuple containing the model abbreviation and the K-Nearest Neighbors Classifier
models.append(('KNN', KNeighborsClassifier()))

# Append a tuple containing the model abbreviation and the Decision Tree Classifier with a fixed random state for consistent results
models.append(('CART', DecisionTreeClassifier(random_state=seed)))

# Define a parameter grid for the Random Forest Classifier to be used in Grid Search for hyperparameter tuning
rf_param_grid = {'n_estimators': [50, 100, 200], 'max_features': ['auto', 'sqrt']}
# Append a tuple containing the model abbreviation and the GridSearchCV object, which encapsulates the Random Forest model with the defined parameter grid and 3-fold cross-validation
models.append(('RF', GridSearchCV(RandomForestClassifier(random_state=seed),
rf_param_grid, cv=3)))

# Append a tuple containing the model abbreviation and the Gaussian Naive Bayes model
models.append(('NB', GaussianNB()))

# Append a tuple containing the model abbreviation and the Support Vector Machine classifier with enabled probability estimates and a fixed random state for reproducibility
models.append(('SVM', SVC(probability=True, random_state=seed)))

# Load the previously saved feature vectors and corresponding labels from HDF5 files for model training and evaluation

# Open the HDF5 file containing the saved feature vectors in read-only mode
h5f_data = h5py.File(h5_train_data, 'r')
# Open the HDF5 file containing the saved labels in read-only mode
h5f_label = h5py.File(h5_train_labels, 'r')

# Access the dataset containing the feature vectors and assign it to a variable
global_features_string = h5f_data['dataset_1']
# Access the dataset containing the labels and assign it to a variable
global_labels_string = h5f_label['dataset_1']

# Convert the datasets to numpy arrays for ease of use in model training and evaluation
global_features = np.array(global_features_string)
global_labels = np.array(global_labels_string)

# Close the HDF5 files to release the resources and ensure data integrity
h5f_data.close()
h5f_label.close()

# verify the shape of the feature vector and labels
print("[STATUS] Features Shape: {}".format(global_features.shape))
print("[STATUS] Labels Shape: {}".format(global_labels.shape))
```

```
print("[STATUS] Training Started...")

# Split the dataset into training and testing sets to evaluate the performance of the trained models

# Utilize the train_test_split function from scikit-learn to partition the dataset
(trainDataGlobal, testDataGlobal, trainLabelsGlobal, testLabelsGlobal) = train_test_split(
    np.array(global_features),  # Feature vectors of the dataset to be split
    np.array(global_labels),  # Corresponding labels of the dataset
    test_size=test_size,  # The proportion of the dataset to include in the test split, defined by the 'test_size' variable
    random_state=seed,  # Seed for the random number generator to ensure reproducible splits
    stratify=np.array(global_labels)  # Data is split in a stratified fashion, using the labels to ensure that both training and testing sets have the same proportion of class labels as the original dataset
)

print("[STATUS] Splitted Train and Test Data...")
print("Train data  : {}".format(trainDataGlobal.shape))
print("Test data   : {}".format(testDataGlobal.shape))

# Initialize lists to store evaluation results and corresponding model names for later analysis and comparison

results = []  # This list will hold the performance metrics or scores (e.g., accuracy, F1 score) obtained from each model
names = []  # This list will store the names or identifiers of the models tested, corresponding to the entries in the 'results' list

# Iterate through each model defined in the 'models' list to evaluate their performance
for name, model in models:

    # Initialize StratifiedKFold to ensure each fold of the cross-validation maintains the proportion of class labels
    kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=seed)
    # Perform cross-validation and calculate accuracy scores for each fold to assess the model's stability
    cv_results = cross_val_score(model, trainDataGlobal, trainLabelsGlobal, cv=kfold, scoring="accuracy")

    # Fit the model to the entire training dataset for further evaluation
    model.fit(trainDataGlobal, trainLabelsGlobal)

    # Make predictions on both the training and test datasets to evaluate overfitting and generalization
    y_train_pred = model.predict(trainDataGlobal)
    y_test_pred = model.predict(testDataGlobal)
    # Obtain probability estimates for the test set to calculate ROC AUC and precision-recall metrics
    y_test_proba = model.predict_proba(testDataGlobal)[:, 1]
```

```
    # Calculate accuracy on the training set to assess how well the model learned from the data
    training_accuracy = accuracy_score(trainLabelsGlobal, y_train_pred)

    # Calculate various performance metrics on the test set to evaluate the model's
generalization capability
    test_accuracy = accuracy_score(testLabelsGlobal, y_test_pred)
    f1 = f1_score(testLabelsGlobal, y_test_pred, average='weighted')
    precision = precision_score(testLabelsGlobal, y_test_pred, average='weighted')
    recall = recall_score(testLabelsGlobal, y_test_pred, average='weighted')
    # Compute ROC curve and AUC to assess model's discriminative ability between classes
    fpr, tpr, _ = roc_curve(testLabelsGlobal, y_test_proba)
    roc_auc = auc(fpr, tpr)
    # Compute precision-recall curve and average precision for insights into model
performance with respect to class imbalance
    precision_curve, recall_curve, _ = precision_recall_curve(testLabelsGlobal, y_test_proba)
    avg_precision = average_precision_score(testLabelsGlobal, y_test_proba)

    # Append the evaluated metrics and curves to their respective lists for later analysis and
comparison
    model_names.append(name)
    training_accuracies.append(training_accuracy)
    test_accuracies.append(test_accuracy)
    cv_scores.append(np.mean(cv_results))
    f1_scores.append(f1)
    precisions.append(precision)
    recalls.append(recall)
    roc_auc_values.append(roc_auc)
    roc_curves.append((fpr, tpr))
    pr_curves.append((precision_curve, recall_curve))

# Construct a DataFrame to consolidate and display the training and testing accuracy for each
evaluated model

# Use pandas DataFrame to organize the model names, training accuracies, and testing
accuracies into a tabular format
accuracy_data = pd.DataFrame({
    'Model': model_names,  # Column for model names, populated from the 'model_names' list
    'Train Accuracy': training_accuracies,  # Column for training accuracies, populated from
the 'training_accuracies' list
    'Test Accuracy': test_accuracies  # Column for testing accuracies, populated from the
'test_accuracies' list
})

# Print the DataFrame to display the accuracies of each model in a clear, tabulated format for
easy comparison
print(accuracy_data)

# Generate a bar plot to visually compare the training accuracies of different machine learning
models
```

```
plt.figure(figsize=(10, 6))  # Set the figure size for the plot
sns.set(style="whitegrid")  # Apply a whitegrid style to the background for better readability
sns.set_palette("pastel")  # Use a pastel color palette for a softer appearance of the bars

# Create a bar plot with 'Model' on the x-axis and 'Train Accuracy' on the y-axis using the accuracy_data DataFrame
bar_plot = sns.barplot(x='Model', y='Train Accuracy', data=accuracy_data)

# Set the x and y-axis labels to provide context to the plot
plt.xlabel('Machine Learning Models')
plt.ylabel('Train Accuracy')

# Set the title of the plot to indicate what the plot represents
plt.title('Train Accuracies for ML Models')

# Rotate the labels on the x-axis by 45 degrees to prevent overlap and improve readability
plt.xticks(rotation=45)

# Loop through each bar in the bar plot to add accuracy labels on top of each bar for precise value representation
for i, v in enumerate(training_accuracies):
    # The text function places a label at the given coordinates; here, it's used to show the accuracy value
    # 'i' is the bar's index, 'v' is the accuracy value, and 'v + 0.01' slightly offsets the label above the bar
    plt.text(i, v + 0.01, "{:.2f}".format(v), ha='center', va='bottom')

plt.show()  # Display the plot

# Generate a bar plot to visually compare the testing accuracies of various machine learning models

plt.figure(figsize=(10, 6))  # Define the dimensions of the plot for clarity and readability
sns.set(style="whitegrid")  # Apply a white grid style to the plot background for a clean look
sns.set_palette("pastel")  # Choose a pastel color palette for a visually appealing plot

# Use seaborn's barplot function to create a bar chart with 'Model' as the x-axis and 'Test Accuracy' as the y-axis
bar_plot = sns.barplot(x='Model', y='Test Accuracy', data=accuracy_data)

# Label the x-axis as 'Machine Learning Models' to indicate the different models compared
plt.xlabel('Machine Learning Models')
# Label the y-axis as 'Test Accuracy' to indicate the metric being compared across models
plt.ylabel('Test Accuracy')

# Set the title of the plot to 'Test Accuracies for ML Models' to describe the plot's purpose
plt.title('Test Accuracies for ML Models')

# Rotate the labels on the x-axis by 45 degrees to ensure they are legible and do not overlap
```

```
plt.xticks(rotation=45)

# Loop through each bar in the plot to add an accuracy label at the top for precise value representation
for i, v in enumerate(test_accuracies):
    # Place a text label above each bar displaying the test accuracy rounded to two decimal places
    # The label is positioned at the center ('ha='center") of the bar and just above the bar ('va='bottom") with a small offset ('v + 0.01')
    plt.text(i, v + 0.01, "{:.2f}".format(v), ha='center', va='bottom')

plt.show()  # Display the completed bar plot

# Generate a plot to visualize the Receiver Operating Characteristic (ROC) curves for all evaluated models

plt.figure(figsize=(12, 7))  # Set the dimensions of the plot for better readability and visual presentation

# Loop through each model along with its ROC curve data (False Positive Rate 'fpr' and True Positive Rate 'tpr')
for i, (name, (fpr, tpr)) in enumerate(zip(model_names, roc_curves)):
    # Plot the ROC curve for each model using its 'fpr' and 'tpr' values
    # Label the curve with the model's name and its Area Under the Curve (AUC) score, formatted to two decimal places
    plt.plot(fpr, tpr, lw=2, label=f'{name} (AUC = {roc_auc_values[i]:.2f})')

# Add a dashed diagonal line (representing a random classifier) for reference
plt.plot([0, 1], [0, 1], 'k--', lw=2)

# Label the x-axis as 'False Positive Rate' to represent the proportion of negative instances incorrectly classified as positive
plt.xlabel('False Positive Rate')

# Label the y-axis as 'True Positive Rate' to represent the proportion of positive instances correctly identified
plt.ylabel('True Positive Rate')

# Set the title of the plot to 'ROC Curves' to clearly indicate the plot's purpose
plt.title('ROC Curves')

# Place a legend in the 'lower right' corner of the plot to identify each model's ROC curve
plt.legend(loc="lower right")

# Enable grid lines on the plot to enhance readability and ease of analysis
plt.grid(True)

plt.show()  # Display the plot with ROC curves for all models
```

```
# Generate a plot to visualize Precision-Recall curves for each model to evaluate their performance in terms of precision at various levels of recall

plt.figure(figsize=(12, 7))  # Set the dimensions of the plot to ensure it is large enough for clear visibility of all curves

# Iterate over each model's name and its corresponding Precision-Recall curve data
for name, (precision_curve, recall_curve) in zip(model_names, pr_curves):
    # Plot the Precision-Recall curve for each model
    # The x-axis represents recall, and the y-axis represents precision for each threshold
    plt.plot(recall_curve, precision_curve, lw=2, label=name)

# Label the x-axis as 'Recall' to represent the proportion of actual positives that were correctly identified
plt.xlabel('Recall')

# Label the y-axis as 'Precision' to represent the proportion of positive identifications that were actually correct
plt.ylabel('Precision')

# Set the title of the plot to 'Precision-Recall Curves' to clearly indicate the plot's content
plt.title('Precision-Recall Curves')

# Place a legend in the 'lower left' corner of the plot to identify each model's curve
plt.legend(loc="lower left")

# Enable grid lines on the plot to facilitate easier comparison of curves at different points
plt.grid(True)

plt.show()  # Display the plot with Precision-Recall curves for all evaluated models
# Visualize the confusion matrix as a heatmap and display the classification report for each evaluated model

class_names = ['Diseased (1)', 'Healthy (0)']  # Define class names for clearer interpretation in the plots and reports

# Iterate through each model to evaluate its performance on the test set
for name, model in models:
    # Train the model on the training dataset
    model.fit(trainDataGlobal, trainLabelsGlobal)
    # Use the trained model to make predictions on the test dataset
    y_test_pred = model.predict(testDataGlobal)

    # Calculate the confusion matrix to evaluate the model's performance
    cm = confusion_matrix(testLabelsGlobal, y_test_pred)

    # Prepare annotation labels for the confusion matrix heatmap, indicating True Positives (TP), False Positives (FP),
    # False Negatives (FN), and True Negatives (TN)
    cm_labels = np.array([['TP={}'.format(cm[1, 1]), 'FP={}'.format(cm[0, 1])],
```

```
['FN={}'.format(cm[1, 0]), 'TN={}'.format(cm[0, 0])]])

    plt.figure(figsize=(8, 6))  # Set the size of the figure for the heatmap

    # Create a heatmap visualization of the confusion matrix using seaborn, with annotated
labels
    sns.heatmap(cm, annot=cm_labels, fmt='', cmap='Blues', xticklabels=class_names,
yticklabels=class_names)

    # Adjust the x-axis labels to appear on the top of the heatmap for better readability
    plt.tick_params(axis='x', labelrotation=0, labeltop=True, labelbottom=False)

    # Label the x-axis as 'Predicted Label' and the y-axis as 'True Label'
    plt.xlabel('Predicted Label')
    plt.ylabel('True Label')
    # Set the title to indicate which model's confusion matrix is being displayed
    plt.title(f'Confusion Matrix for {name}', pad=20)
    plt.show()  # Display the heatmap

    # Generate and print the classification report, providing detailed performance metrics such
as precision, recall, and F1-score
    cr = classification_report(testLabelsGlobal, y_test_pred, target_names=class_names)
    print(f"Classification Report for {name}:\n{cr}\n")

    # Calculate and print the accuracy score to assess the overall performance of the model
    accuracy = accuracy_score(testLabelsGlobal, y_test_pred)
    print(f"Accuracy for {name}: {accuracy * 100:.2f}%\n")

    # Identify and display the model with the highest accuracy on the test dataset from the
evaluated models

# Use np.argmax to find the index of the maximum test accuracy in the list of test accuracies
max_accuracy_index = np.argmax(test_accuracies)

# Retrieve the name of the best performing model using the index obtained above
best_model_name = model_names[max_accuracy_index]

# Retrieve the highest test accuracy value using the same index
best_model_accuracy = test_accuracies[max_accuracy_index]

# Print out the best model's name and its test accuracy, formatted as a percentage and rounded
to two decimal places
print(f"The best model out of all the evaluated modles is '{best_model_name}' with a test
accuracy of {best_model_accuracy * 100:.2f}%.")
```

APPENDIX D: PYTHON CODE FOR IMAGE CLASSIFICATION – DL MODELS

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2, NASNetMobile
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.models import Model
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout
from sklearn.metrics import classification_report, confusion_matrix, precision_recall_curve,
roc_curve, auc
from sklearn.model_selection import StratifiedKFold
import matplotlib.pyplot as plt
import seaborn as sns

# Function to create a model
def create_model(model_type, input_shape, num_classes):
    if model_type == 'MobileNetV2':
        base_model = MobileNetV2(weights='imagenet', include_top=False,
input_shape=input_shape)
    elif model_type == 'NASNetMobile':
        base_model = NASNetMobile(weights='imagenet', include_top=False,
input_shape=input_shape)

    base_model.trainable = False
    x = GlobalAveragePooling2D()(base_model.output)
    x = Dense(128, activation='relu')(x)
    x = Dropout(0.5)(x)
    outputs = Dense(num_classes, activation='softmax')(x)
    model = Model(inputs=base_model.input, outputs=outputs)
    model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
                  loss='categorical_crossentropy',
                  metrics=['accuracy', tf.keras.metrics.Precision(name='precision'),
tf.keras.metrics.Recall(name='recall')])
    return model

# Function to plot training and validation accuracy and loss
def plot_metrics(history):
    plt.figure(figsize=(12, 5))

    # Plot accuracy
    plt.subplot(1, 2, 1)
    plt.plot(history.history['accuracy'], label='Training Accuracy')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
    plt.title('Accuracy over Epochs')
    plt.xlabel('Epochs')
```

```
    plt.ylabel('Accuracy')
    plt.legend()

    # Plot loss
    plt.subplot(1, 2, 2)
    plt.plot(history.history['loss'], label='Training Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.title('Loss over Epochs')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()

    plt.show()

def evaluate_model(model, test_flow, model_type):
    test_flow.reset()
    val_pred = model.predict(test_flow)
    val_pred_classes = np.argmax(val_pred, axis=1)
    val_true = test_flow.classes

    # Compute Precision-Recall curve
    precision, recall, _ = precision_recall_curve(val_true, val_pred[:, 1])

    # Compute the area under the curve
    auc_score = auc(recall, precision)

    # Classification report
    print("\nClassification Report:\n")
    print(classification_report(val_true, val_pred_classes, target_names=['Healthy',
'Diseased']))

    class_names = ['Diseased (1)', 'Healthy (0)']  # Define class names for clearer interpretation
in the plots and reports

    # Confusion matrix
    cm = confusion_matrix(val_true, val_pred_classes, labels=[1, 0])

    # Prepare annotation labels for the confusion matrix heatmap, indicating True Positives
(TP), False Positives (FP),
    # False Negatives (FN), and True Negatives (TN)
    cm_labels = np.array([['TP={}'.format(cm[0, 0]), 'FP={}'.format(cm[0, 1])],
                          ['FN={}'.format(cm[1, 0]), 'TN={}'.format(cm[1, 1])]])


    # Plotting Confusion Matrix with desired format
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=cm_labels, fmt='', cmap='Blues', xticklabels=class_names,
yticklabels=class_names)

    # Adjust the x-axis labels to appear on the top of the heatmap for better readability
```

```
    plt.tick_params(axis='x', labelrotation=0, labeltop=True, labelbottom=False)

    #sns.heatmap(cm_df, annot=True, fmt='g', cmap='Blues')
    plt.title('Confusion Matrix')
    plt.ylabel('True Label')
    plt.xlabel('Predicted Label')
    plt.show()

    # ROC-AUC Curve
    fpr, tpr, _ = roc_curve(val_true, val_pred[:, 1])
    roc_auc = auc(fpr, tpr)
    plt.figure(figsize=(6, 6))
    plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.title('ROC-AUC Curve')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.legend(loc="lower right")
    plt.show()

    # Plot the Precision-Recall curve
    plt.figure(figsize=(6, 6))
    plt.plot(recall, precision, color='blue', lw=2, label=f'Precision-Recall curve (area =
{auc_score:.2f})')
    plt.xlabel('Recall')
    plt.ylabel('Precision')
    plt.title(f'Precision-Recall Curve for {model_type}')
    plt.legend(loc="best")
    plt.show()

# Image Data Generators
train_data_dir = r'C:\Users\gsanthos\Downloads\Plant-Disease-Detection-
master\image_classification\dataset\train'
test_data_dir = r'C:\Users\gsanthos\Downloads\Plant-Disease-Detection-
master\image_classification\dataset\test'
img_size = (224, 224)
batch_size = 32
num_classes = 2

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

test_datagen = ImageDataGenerator(rescale=1./255)
```

```
# Load dataset filenames and labels
train_flow = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=True,
    classes=['healthy', 'diseased']  # Explicitly setting the order of classes
)

test_flow = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False,
    classes=['healthy', 'diseased']  # Explicitly setting the order of classes
)

print(train_flow.class_indices)

# Define a dictionary to hold performance metrics
performance = {}

# K-Fold Cross-Validation
n_splits = 5
kfold = StratifiedKFold(n_splits=n_splits, shuffle=True, random_state=42)

for fold, (train_idx, val_idx) in enumerate(kfold.split(train_flow.filenames,
train_flow.classes)):
    print(f"\nTraining fold {fold + 1}/{n_splits}")

    # Training and Validation DataFrames
    train_df = pd.DataFrame({'filename': np.array(train_flow.filenames)[train_idx], 'class':
train_flow.classes[train_idx].astype(str)})
    val_df = pd.DataFrame({'filename': np.array(train_flow.filenames)[val_idx], 'class':
train_flow.classes[val_idx].astype(str)})

    # Training and Validation Generators
    train_data_gen = train_datagen.flow_from_dataframe(
        dataframe=train_df,
        directory=train_data_dir,
        x_col='filename',
        y_col='class',
        target_size=img_size,
        batch_size=batch_size,
        class_mode='categorical',
        shuffle=True
    )
```

```
    val_data_gen = train_datagen.flow_from_dataframe(
        dataframe=val_df,
        directory=train_data_dir,
        x_col='filename',
        y_col='class',
        target_size=img_size,
        batch_size=batch_size,
        class_mode='categorical',
        shuffle=False
    )

    # Train and Evaluate Models
    for model_type in ['MobileNetV2', 'NASNetMobile']:
        print(f"\nTraining model: {model_type}")
        model = create_model(model_type, img_size + (3,), num_classes)

        early_stopping = EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True)
        reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=3, min_lr=1e-
5)

        history = model.fit(
            train_data_gen,
            steps_per_epoch=len(train_idx) // batch_size,
            epochs=10,
            validation_data=val_data_gen,
            validation_steps=len(val_idx) // batch_size,
            callbacks=[early_stopping, reduce_lr]
        )

        plot_metrics(history)
        evaluate_model(model, test_flow,model_type)

        # Collect accuracy from the model's history
        model_accuracy = history.history['val_accuracy']

        # Update performance dictionary
        if model_type not in performance:
            performance[model_type] = model_accuracy
        else:
            performance[model_type].extend(model_accuracy)

# Summary of performance
# Calculate the average accuracy for each model
best_performance = {model: np.mean(accuracies) for model, accuracies in
performance.items()}
best_model = max(best_performance, key=best_performance.get)

print(f"\nThe best performing model is {best_model} with an average validation accuracy of
{best_performance[best_model]:.4f}")
```