People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University Of Science And Technology HOUARI BOUMEDIENE
Faculty of Mathematics

Final Year Project Report Presented for the Degree of **MASTER**

In : Applied Mathematics

Option : Statistics and Applied Probabilites

Presented by : FERGUOUS Wafa

Entitled

Comparative Analysis of Plants Disease Prediction and Identification Using Structured and Unstructured Data

Defended publicly , on 07/08/2023, in front of jury composed of:

Mrs.	K.DJABALLAH	Professeur	USTHB	President
Mrs.	A.BOUCHAFAA	MAA	USTHB	Supervisor
Mr.	M.AIB	Doctor	AITech	Co-Supervisor
Mrs.	L.ZEGHIDI	MAA	USTHB	Examiner

Dedication

وآخر دعواهم أن الحمد لله رب العالمين

الحمد لله الذي أنعم وأكرم وأتم، الحمد لله الذي قر الفؤاد بعد أيام التشوق والعنا.

إلى أول أسباب نجاحي، وبطل طفولتي، وأجمل نعم الله علي، إلى الطيب والدي.

إلى النور الذي يضيء عتمتي، ويقيني الكبير، إلى جنتي والدتي.

إلى أميراتي الصغيرات هناء، إيناس.

إلى صغيري عبد الرؤوف.

إلى من كانوا لي خير سند في كل خطوة أخطوها أميرة، سهيلة.

إلى خالاتي فاطمة، خيرة و جدتي اللواتي لازمني دعاؤهن طول مشواري.

إلى رفيقتي الخطوة الأولى والخطوة ما قبل الأخيرة ملك، سامية.

إلى البعيدة عن العين القريبة من القلب نرجس.

إلى الصديقات أية، نور جهان، سارة.

وإلى كل من وقف بقربي حتى أصل إلى ما أنا عليه الآن.

أنا ممتنة

Remerciement

In the name of Allah, the Most Gracious, the Most Merciful.
All praise be to Allah, the Almighty, for granting me the strength, guidance, and perseverance to complete this dissertation. Without His blessings and mercy, this achievement would not have been possible.
I would like to thank my supervisor Dr. Asma BOUCHAFAA for all her advice and remarks. It was an honor to work on my project with her.
My gratitude goes to Dr. Mabrouk AIB for the opportunity and guidance he provided.
I must thank Prof. Dr. K. DJABALLAH and Dr. L. ZEGHIDI for accepting to examine and evaluate my work.
I would also like to thank my family and friends for their love and support. They have always been there for me, even during the most challenging times. I am so blessed to have them in my life.
I want to sincerely thank all the people who took part in my journey. Their valuable insights and experiences have been extremely important for the success of my project.

Abstract

Plant disease prediction and identification are important tasks in agriculture as they can help farmers take timely action to prevent crop losses. In this thesis, we investigated the use of machine learning and deep learning techniques for these purposes. We began by reviewing the fundamentals of machine learning and deep learning, with a specific focus on their application to plant disease prediction.

The study then presents a case study where we utilized machine learning and deep learning techniques to predict the risk of plant disease in tomato crops and identify plant diseases based on leaf images. For the experiments, we used two different datasets. In the first experiment, we employed structured data to train three models: logistic regression, an artificial neural network (ANN), and a support vector machine (SVM). In the second experiment, we used unstructured data to train convolutional neural network (CNN) architectures VGG16 and MobileNetV1.

The results of the case study indicate that the ANN model achieved the best performance in terms of accuracy and F1-score. The ANN model correctly predicted the risk of disease in 97% of the test data. Additionally, the CNN architecture called MobileNetV1 demonstrated an impressive accuracy of 98% in the plant leaf disease identification task.

Based on the results of the case study, we conclude that machine learning and deep learning techniques can be effectively utilized for plant disease prediction and identification. These methods offer promising avenues for assisting farmers in safeguarding their crops and enhancing agricultural productivity.

Keywords: Machine Learning, Deep Learning, Plant Disease Prediction, Structured Data, Unstructured Data, Convolutional Neural Networks.

Résumé

La prédiction et l'identification des maladies des plantes sont des tâches importantes en agriculture car elles peuvent aider les agriculteurs à prendre des mesures en temps opportun pour prévenir les pertes de récoltes. Dans cette mémoire, nous avons étudié l'utilisation des techniques d'apprentissage automatique et d'apprentissage profond à ces fins. Nous avons commencé par passer en revue les fondamentaux de l'apprentissage automatique et de l'apprentissage profond, en mettant l'accent sur leur application à la prédiction des maladies des plantes.

L'étude présente ensuite une étude de cas dans laquelle nous avons utilisé des techniques d'apprentissage automatique et d'apprentissage profond pour prédire le risque de maladies des plantes dans les cultures de tomates et identifier les maladies des plantes sur la base d'images de feuilles. Pour les expériences, nous avons utilisé deux ensembles de données différents. Dans la première expérience, nous avons utilisé des données structurées pour entraîner trois modèles : la régression logistique, un réseau de neurones artificiels (ANN) et une machine à vecteurs de support (SVM). Dans la deuxième expérience, nous avons utilisé des données non structurées pour entraîner les architectures de réseaux de neurones convolutifs (CNN) VGG16 et MobileNetV1.

Les résultats de l'étude de cas indiquent que le modèle ANN a obtenu les meilleures performances en termes de précision et de score F1. Le modèle ANN a prédit correctement le risque de maladie dans 97% des données de test. De plus, l'architecture CNN appelée MobileNetV1 a démontré une précision impressionnante de 98% dans la tâche d'identification des maladies des feuilles de plantes.

Sur la base des résultats de l'étude de cas, nous concluons que les techniques d'apprentissage automatique et d'apprentissage profond peuvent être efficacement utilisées pour la prédiction et l'identification des maladies des plantes. Ces méthodes offrent des perspectives prometteuses pour aider les agriculteurs à protéger leurs cultures et améliorer la productivité agricole.

Mots-clés : Apprentissage automatique, Apprentissage profond, Prédiction des maladies des plantes, Données structurées, Données non structurées, Réseaux de neurones convolutifs.

Contents

List of Figures

List of Tables

GENERAL INTRODUCTION

Many crops grown in Algeria face significant challenges due to plant diseases, which harm agricultural production and farmers' livelihoods. Globally, plant diseases cause substantial losses, estimated at around 25% annually, impacting roughly 600 million people [10].

In Algeria, tomatoes are among the most important crops, with high local consumption and export potential. However, these crops are highly susceptible to fungal and bacterial infections, making it difficult to predict, diagnose, and control disease outbreaks. In particular, late blight disease is a significant threat to tomato crops, causing devastating consequences and significant economic losses.

By collecting data from sensors in the field and utilizing prediction techniques, farmers can manage diseases more effectively and reduce the cost of crop losses and pesticide use. Thus, the need for accurate disease prediction systems in Algeria is crucial to improve crop production and secure the livelihoods of farmers.

The environment plays a critical role in disease development. A suitable environment is necessary for the disease to form and spread, and the concept of the Disease Triangle (Antle et al., 2017) can be used to analyze plant diseases (shown in 0.1). The Disease Triangle consists of three essential elements: the host (the plant itself), an active pathogen (e.g., virus, bacterium, fungus, or parasite), and a favorable environment with suitable weather conditions. All three conditions must be present simultaneously for a disease to occur; otherwise, the disease will not develop.

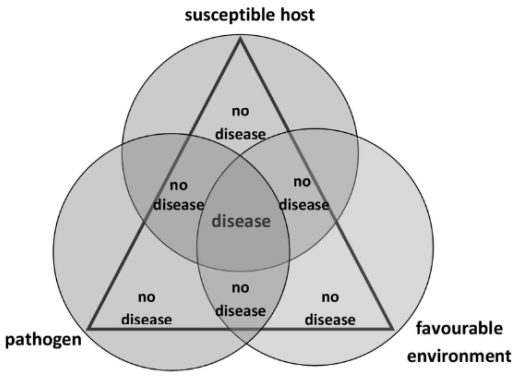

Figure 0.1: Disease triangle illustration [1]

In recent times, diseases can easily spread worldwide, causing increasingly complicated situations. New diseases may appear in regions where they have never been seen before, making it difficult to control them effectively due to limited experience.
using new technologies is essential for early disease detection. Deep Learning, a widely-used technology, shows promise in achieving close-to-human accuracy in disease identification.

Through our study, we aim to highlight the role that classification algorithms and Deep Learning architecture play in predicting and identifying plant diseases.

The study outcomes are expected to contribute valuable insights to the field of agricultural technology, supporting informed decision-making by farmers and agricultural experts to manage plant diseases effectively and ensure global food security.

Research Question

Can we accurately predict plant diseases based on weather parameters and identify diseases using convolutional neural networks, and how do different models perform in comparison?

Overview of the Chapters

Chapter 1

This chapter introduces the Machine Learning, Deep Learning, and Convolutional Neural Networks, providing a solid foundation for understanding their applications.

Chapter 2

This chapter shifts the focus to the application of Machine Learning and Deep Learning techniques in the domain of plant disease prediction.

Chapter 3

Building upon the previous chapter, this chapter addresses the application of Machine Learning and Deep Learning techniques in plant disease identification using unstructured data, specifically images.

Chapter 4

This chapter presents the experiments conducted for plant disease prediction and identification using both structured and unstructured data. It outlines the different models used for the prediction and identification , including logistic regression, artificial neural networks, support vector machines, VGG16 and MobileNet. The chapter also discusses the dataset used for the experiments and the evaluation metrics used to assess the model's performance. The results of the experiments are presented and interpreted.

Chapter 1

Machine Learning and Deep Learning Fundamentals

The definitions and explanations in this chapter are primarily sourced from the following references: An Introduction to Statistical Learning [6] , Deep Learning [11] , Hands-On Mathematics for Deep Learning [2]

1.1 Machine Learning

1.1.1 What is Machine Learning?

Machine learning (ML) is a subfield of artificial intelligence (AI) that focuses on developing algorithms and models that enable computers to learn from data and make predictions or decisions without being explicitly programmed.

1.1.2 Learning Algorithms

A machine learning algorithm is capable of learning from data. But what do we mean by learning? In his book "Machine Learning," Tom Mitchell (1997) defines learning as "the ability of a computer program to improve its performance on a task with experience."
There are many different types of tasks that can be learned by machine learning algorithms include:

- **Classification:** Categorizing data into different classes.
- **Regression:** Predicting a continuous value, such as a price or a rating.
- **Natural language processing:** Understanding and generating human language.
- **Computer vision:** Identifying and understanding objects in images and videos.

1.1.3 Capacity, Overfitting and Underfitting

Capacity

The capacity of a machine learning model refers to how well it can learn from data. A model with high capacity can understand complex relationships between features and the target, while a model with low capacity can only understand simple relationships.

Overfitting

Overfitting occurs when a model becomes overly fixated on the training data and encounters difficulties in performing well with new data. This can happen when the model has excessive complexity and assimilates the irrelevant details or noise present in the training data.

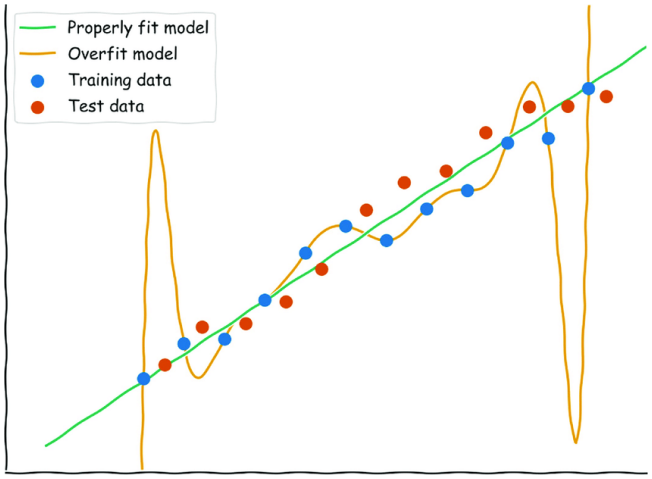

Figure 1.1: A visual example of overfitting [12]

Underfitting

Underfitting occurs when a model doesn't learn the training data well and makes inaccurate predictions. This can happen when the model has too little capacity and fails to capture the underlying relationships in the data.

There are a number of techniques that can be used to avoid overfitting and underfitting, including:

- **Regularization**: Penalizing a model for having too many parameters can prevent it from learning the noise in the training data.
- **Cross-validation**: Evaluating a model's performance on new data helps identify if it's overfitting or underfitting the training data.
- **Data augmentation**: Increasing the size of the training dataset artificially can help prevent models from overfitting.

1.1.4 Hyperparameters and Validation Sets

Hyperparameters

In machine learning, we have what we call hyperparameters(often denoted using Greek letters). These are like special settings that determine how a learning algorithm works. They are different from the model's parameters, which are learned during the training phase. The interesting thing about hyperparameters is that they are not changed by the learning algorithm itself.

Why aren't hyperparameters learned during training?

If the hyperparameters, which are responsible for regulating the complexity of the model, were determined based on the training dataset, there is a likelihood of selecting the most complex settings. This tendency could potentially result in overfitting, wherein the model fits too closely to the training data and fails to generalize well to new data.

How do we choose hyperparameters?

There are several methods to choose hyperparameters, such as:

- **Grid search**: Trying out all possible combinations of hyperparameter values.
- **Random search**: Randomly selecting hyperparameter values from a predefined range.
- **Bayesian optimization**: Using a statistical model to estimate the best hyperparameter values.

Cross-Validation

Cross-validation is a technique used to evaluate how well a model performs on new, unseen data. It helps identify whether a model is overfitting (memorizing the training data) or underfitting (not capturing the underlying patterns).

K-fold Cross-Validation

K-fold cross-validation is a commonly used approach in which the dataset is divided into k non-overlapping subsets or folds. The model's performance is evaluated by averaging the results across k trials. In each trial, one subset is used as the test set, while the rest of the data is used for training.

1.1.5 Supervised Learning

Supervised learning is a type of machine learning algorithm that involves training a model to predict an output variable based on input features and labeled examples. In supervised learning, the algorithm is fed with input data ($x_i, i = 1, \ldots, n$) and corresponding output labels (y_i), and its goal is to learn a function that maps the input data to the correct output labels.

Supervised learning is used in a variety of applications, such as image classification, speech recognition, and natural language processing.

The input data can be represented in different formats, such as text, images, or audio signals, and the output labels can be categorical, such as class labels, or continuous, such as numeric values.
There are several types of supervised learning algorithms, including regression, classification.

1.1.6 Unsupervised Learning

Unsupervised learning is a type of machine learning algorithm where the algorithm models a set of inputs without labeled examples.
In unsupervised learning, the goal is to find patterns or structures in the data without any prior knowledge of the output. This is typically used in clustering problems, where the goal is to group similar data points together based on their features.
Some examples of unsupervised learning algorithms include k-means clustering, hierarchical clustering, and principal component analysis.
Unsupervised learning is different from supervised learning, which requires labeled examples to train the model

1.2 Deep Learning

Deep learning is a subfield of machine learning (ML) that focuses on developing algorithms and models capable of learning and making predictions or decisions without explicit programming, within the broader field of artificial intelligence (AI).
Deep learning algorithms and models can be categorized into different types based on their learning approach and architectural designs, such as convolutional neural networks (CNNs) for image analysis.

1.2.1 Linear Neural Networks

linear neural networks form the essential foundation of complex neural network architectures. Linear neural networks are particularly useful for various tasks, including logistic regression. this task are common within the domain of statistical learning, providing valuable insights into the fundamental of machine learning.

1.2.1.1 Logistic Regression

Logistic regression is a statistical method used to estimate the likelihood that an event will happen. In the context of predicting plant diseases using meteorological data, logistic regression can be applied to estimate the probability of a plant being diseased based on the weather conditions.

Rather than directly predicting whether a plant is diseased or not, logistic regression predicts the probability of disease occurrence. This probability is denoted as $p(\text{weather})$, which represents the probability of disease given specific weather conditions.

To model this relationship, we need a function that can provide probabilities for all weather values. Logistic regression utilizes the logistic function, denoted as $\sigma(z)$ and defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{1.1}$$

In logistic regression, the linear combination of input variables and coefficients is represented as z. In the case of plant disease prediction, the logistic regression model equation can be expressed as follows:

$$\text{logit}(p(\text{weather})) = \beta_0 + \beta_1 \cdot \text{temp} + \beta_2 \cdot \text{humidity} + \beta_3 \cdot \text{wind speed} + \dots \tag{1.2}$$

Here, logit() is the inverse of the logistic function, β_0 represents the intercept coefficient, and $\beta_1, \beta_2, \beta_3, \dots$ represent the coefficients associated with the corresponding input variables $\text{temp}, \text{humidity}, \dots, \text{wind speed}, \dots$.

The coefficients $\beta_0, \beta_1, \beta_2, \beta_3, \dots$ in the logistic function are estimated using a method called maximum likelihood estimation (MLE). MLE finds the coefficients that maximize the likelihood of observing the actual disease occurrences in the training data. The likelihood function is defined as:

$$L(\beta_0, \beta_1, \ldots, \beta_n) = \prod \left[p(\text{weather}_i)^{\text{disease}_i}\right] \left[1 - p(\text{weather}_i)^{1-\text{disease}_i}\right] \tag{1.3}$$

Here, disease_i represents the actual disease status (0 for no disease, 1 for disease) for the i-th observation.

The odds of disease can be calculated using the logistic regression model. Odds represent the likelihood of disease occurrence divided by the likelihood of no disease. In logistic regression, the odds of disease can be obtained by taking the exponential function of the linear combination of the input variables and coefficients:

$$\text{Odds}(\text{disease} = \text{Yes}) = e^{\beta_0 + \beta_1 \cdot \text{temp} + \beta_2 \cdot \text{humidity} + \beta_3 \cdot \text{wind speed} + \ldots} \tag{1.4}$$

Here, e represents the base of the natural logarithm.

Logistic regression allows us to estimate the relationship between the input variable (weather) and the probability of disease by fitting a line to the data. The coefficients of the logistic regression model represent the impact of the input variable on the log odds of disease. By estimating these coefficients, logistic regression provides a way to make predictions and understand the influence of the input variable on the probability of disease.

1.2.2 Feedforward Neural Networks

1.2.2.1 Understanding biological neural networks

The human brain is an amazing organ that can process complex information by using interconnected neurons.These neurons communicate with each other through synapses, which can strengthen or weaken connections to facilitate learning and adaptation(as illustrated in figure1.2). With around 86 billion neurons, the brain operates in parallel, allowing us to perform various tasks. Neurons consist of a body, axon, and dendrites, and synapses connect them, determining the importance of received information. Neurons communicate through electrochemical reactions, and they become active when the combined inputs exceed a certain threshold. Artificial neural networks (ANNs) are inspired by biological neurons.

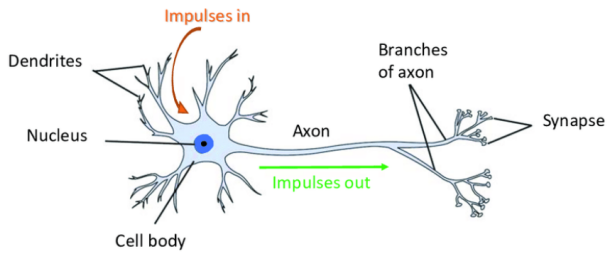

Figure 1.2: an illustration of a human neuron [13]

1.2.2.2 Perceptron

Perceptron model is a type of artificial neuron that can be used for supervised learning. It was first introduced by Frank Rosenblatt in 1958. The perceptron model is a single-layer neural network that consists of an input layer, a hidden layer, and an output layer. The input layer receives the input data, the hidden layer performs the computation, and the output layer produces the output data.

The perceptron model works by multiplying the input data by a set of weights and then adding a bias term. If the sum of the weighted inputs is greater than a threshold, then the output is 1. Otherwise, the output is 0.

Here is a diagram of a basic perceptron model

[

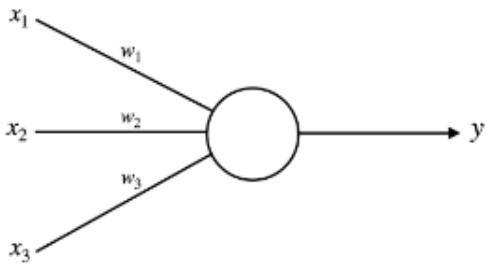

Figure 1.3: Diagram of a basic perceptron model [2]

Perceptron models were limited in their ability to handle nonlinear problems. To overcome this limitation, modern perceptrons, known as nodes, incorporate an activation function that introduces nonlinearity to the output.

The improved perceptrons have the following structure:

$$y = \varphi(w_1x_1 + w_2x_2 + \ldots + w_nx_n + b) \tag{1.5}$$

the variables represent the following:

- y: output
- φ: nonlinear activation function
- x_i: inputs
- w_i: weights
- b: bias

1.2.2.3 Multi-Layer Perceptron

An **MLP (Multi-Layer Perceptron)** is a simple type of feedforward neural network (FNN) that addresses the limitations of perceptron models. It consists of multiple nodes arranged in

layers, with computation performed sequentially. The nodes in each layer are fully connected to all the neurons in the next layer.

The MLP architecture includes an input layer, one or more hidden layers, and an output layer.The number of nodes in the output layer depends on the specific problem.

Here is a diagram of multi-layer perceptron model with 2 hidden layers

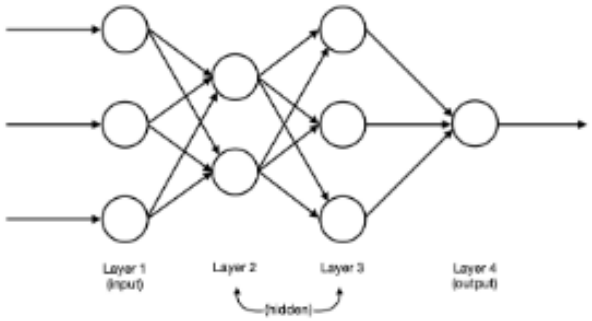

Figure 1.4: MLP Architecture [2]

1.2.2.4 Activation functions

There are many different activation functions available, each with its own advantages and disadvantages. The most common activation functions are:

Sigmoid: The sigmoid function is a special type of function that takes the output of a neuron and squeezes it to be in [0,1]. It's like compressing the value to fit within this range. By using the sigmoid function, the output of the network can be interpreted as the probability of something being true or false. So, it helps us determine the likelihood or chance of something happening.

The sigmoid function is written as follows:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1.6}$$

The function looks as follows:

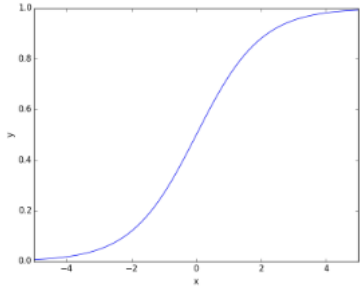

Figure 1.5: Plot of the sigmoid function [2]

Tanh: The tanh function is similar to the sigmoid function, but it has a wider range of outputs ([-1,1]). This makes it useful for regression problems, where the output of the network

is a continuous value.
The tanh function is written as follows:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{1.7}$$

The function looks as follows:

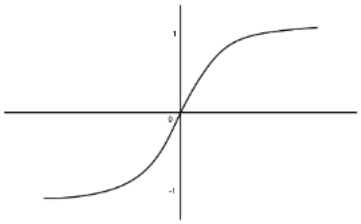

Figure 1.6: Plot of the tanh function [2]

ReLU: The ReLU function is a non-linear function that outputs 0 for negative inputs and the input value for positive inputs. This makes it a very efficient activation function, as it does not require any calculations for negative inputs.
The ReLU function is written as follows:

$$\text{ReLU}(x) = \max(0, x) \tag{1.8}$$

The function looks as follows:

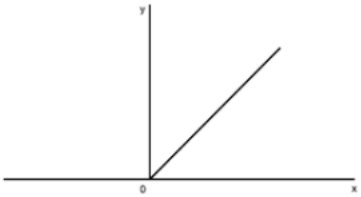

Figure 1.7: Plot of the ReLU function [2]

Softmax: The softmax function is a special type of activation function that is used in the output layer of a neural network for classification problems. The softmax function normalizes the outputs of the neurons in the output layer so that they sum to 1. This ensures that the output of the network represents a probability distribution over the possible classes.
The softmax function is written as follows:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{1.9}$$

The choice of activation function depends on the specific problem that the neural network is trying to solve.

1.2.2.5 The Loss Function

Loss functions are used to measure how well a neural network is performing. They do this by comparing the output of the network to the desired output. The network is then adjusted to reduce the loss function.

1.2.2.6 Backpropagation

Backpropagation is a technique used to train neural networks. It works by calculating the error of the network's output and then using that error to update the network's weights.

The first step in backpropagation is to **calculate the error of the network's output**. This is done by comparing the network's output to the desired output. The difference between the two is the error.

The next step is to **use the error to update the network's weights**. This is done by **taking the derivative of the loss function with respect to the weights**. The derivative tells us how much each weight needs to be changed in order to reduce the error.

The weights are then **updated using a technique called gradient descent**. Gradient descent is an iterative algorithm that repeatedly updates the weights in order to **minimize the loss function**.

the process is illustrated in 1.8

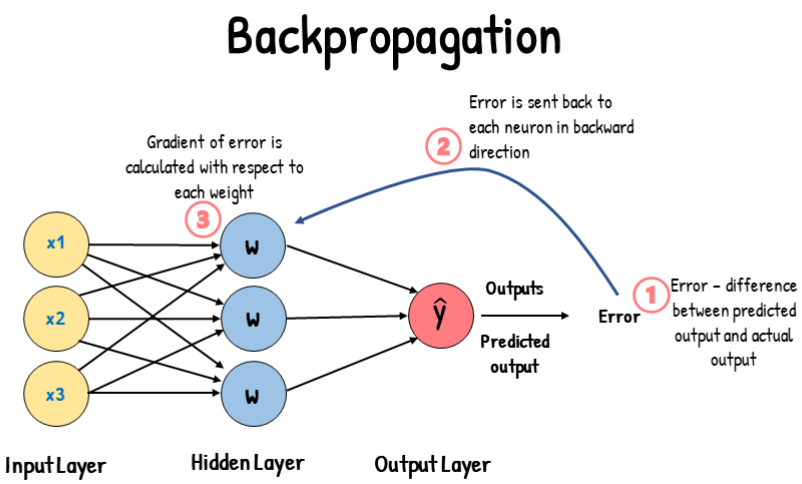

Figure 1.8: backpropagation process [3]

1.2.3 Regularization

Regularization is a technique used to prevent neural networks from overfitting to the training data. Overfitting occurs when a model learns the training data too well, and as a result, it

performs poorly on new data. Regularization helps to prevent overfitting by adding constraints to the model, which makes it less likely to learn the training data too well.

1.2.3.1 Norm Penalties

Norm penalties are a method used to control the complexity of a model and prevent overfitting.

To apply a norm penalty, we modify the objective function of the model. We add a term called the norm penalty term, which helps regulate the cost function.

The modified objective function is written as:

$$J(\theta; X, y) \equiv J(\theta; X, y) + \alpha N(\theta) \tag{1.10}$$

In this equation, α is a value called the hyperparameter. It determines how strong the regularization effect will be. If α is high, the regularization effect is stronger, and if α is low, the regularization effect is weaker.

here are two main types of norm penalties:

- **L1 regularization (Lasso):** It adds a penalty term, $\lambda \sum |w|$, to the loss function, where λ is the regularization parameter and $\sum |w|$ represents the sum of the absolute values of the weights. This encourages sparsity, as it pushes many weights towards zero.
- **L2 regularization (Ridge):** It adds a penalty term, $\lambda \sum w^2$, to the loss function, where λ is the regularization parameter and $\sum w^2$ represents the sum of the squares of the weights. This encourages smaller weights but does not enforce sparsity.

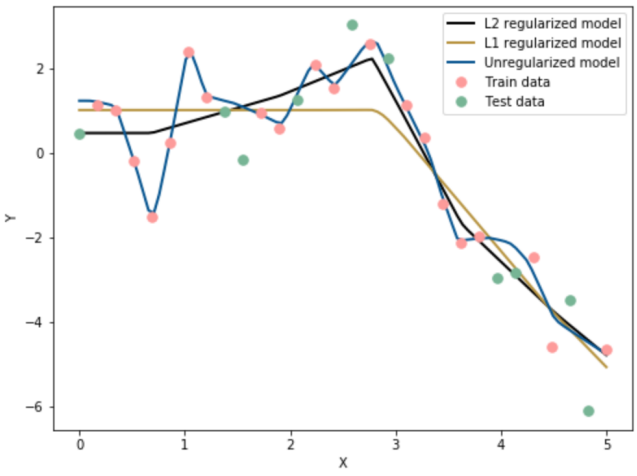

Figure 1.9: L1 and L2 regularizatio [14]

1.2.3.2 Dropout

In addition to penalties,dropout is another widely used approach for regularization in neural networks. Dropout addresses the issue of co-adaptation by randomly dropping out (setting to zero) a fraction of neurons in each iteration.

Example:
Consider a multilayer perceptron (MLP) with one hidden layer containing 10 neurons. If we apply dropout with a probability of $p = 0.5$, half of the neurons (i.e., 5 neurons) will be randomly set to 0 during each training iteration.

1.2.4 Hyperparameters

Hyperparameters are extremely important as they greatly influence how neural networks behave and how well they work. There are several hyperparameters that are frequently used in deep learning, and some examples of these include:

- Batch size (B): It represents the number of data points processed together. Using a larger batch size can improve the model's performance, but it may also require more memory and computation time.
- Learning rate (α): This parameter determines the size of the steps the model takes while learning. A larger learning rate can speed up the training process, but it may also lead to more mistakes and poor generalization to new data.
- Optimizer: It refers to the algorithm used to update the model's parameters during training. There are different optimizers available, each with their own strengths and weaknesses. Selecting the appropriate optimizer can significantly impact the model's learning performance.
- Epochs (E): is the number of times the model goes through all the data during training. Increasing the number of epochs can enhance the model's performance, but it also takes longer to train.

1.2.5 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks or CNNs, are a type of artificial neural network inspired by how our brain processes visual information. Just like our brain recognizes objects in the world, CNNs are designed to recognize patterns and objects in images.

In our brain, there are cells that perform computations similar to convolutions. These cells help us detect basic features like edges and curves. Additionally, there are more complex cells that have spatial invariance, meaning they can recognize the same features regardless of their position or orientation.

Similarly, in CNNs, convolutional layers are used to extract different features from images. These layers help the network identify basic shapes and patterns. As the information passes through deeper layers, the network learns to recognize more complex features and objects.

CNNs can learn to understand and interpret visual information, enabling them to perform tasks like image classification, object detection.

1.2.5.1 Convolutional Neural Networks Architecture

1.2.5.2 RGB Notation

RGB refers to the color channels of an image. An RGB image is composed of three separate channels: **red**, **green**, and **blue** shown in figure 1.10. Each channel represents the intensity of that specific color component in each pixel of the image. By combining these three channels, we can represent a wide range of colors and create a full-color image.

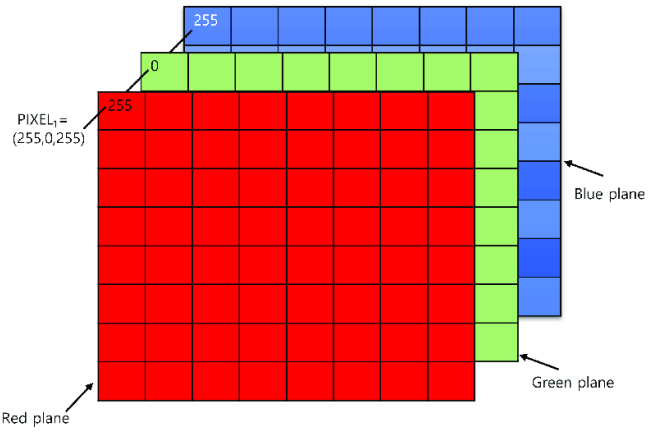

Figure 1.10: Example of a three-dimensional RGB matrix [4]

1.2.5.3 The Convolution Layer

The convolutional layer in a neural network consists of learnable filters that are small in size but extend through the full depth of the input volume. These filters slide or move across the width and height of the input, calculating dot products at each position. As a result, we get a 2-dimensional map that shows how each filter responds to different locations in the input. The network learns these filters to activate when they detect certain visual characteristics like edges or colors (The process in the convolutional layer can be visualized through the images shown in Figure 1.11).

Imagine we have a 4×4 input matrix representing an image:

$$\text{Input Matrix} : \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

Now, let's consider a convolutional layer with two 3×3 filters:

$$\text{Filter 1} : \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad \text{Filter 2} : \begin{bmatrix} -1 & 0 & -1 \\ 0 & -1 & 0 \\ -1 & 0 & -1 \end{bmatrix}$$

Applying Filter 1:

To compute the activation map using Filter 1, we slide the filter over the input matrix and perform dot products:

$$\text{Activation Map 1}: \begin{bmatrix} 1\cdot 1+2\cdot 0+3\cdot 1 & 2\cdot 0+3\cdot 1+4\cdot 0 \\ 5\cdot 0+6\cdot 1+7\cdot 0 & 6\cdot 1+7\cdot 0+8\cdot 1 \end{bmatrix}$$

Simplifying the calculations:

$$\text{Activation Map 1}: \begin{bmatrix} 4 & 3 \\ 6 & 9 \end{bmatrix}$$

Applying Filter 2:

Similarly, we slide Filter 2 over the input matrix and compute dot products:

$$\text{Activation Map 2}: \begin{bmatrix} 1\cdot(-1)+2\cdot 0+3\cdot(-1) & 2\cdot 0+3\cdot(-1)+4\cdot 0 \\ 5\cdot 0+6\cdot(-1)+7\cdot 0 & 6\cdot(-1)+7\cdot 0+8\cdot(-1) \end{bmatrix}$$

Simplifying the calculations:

$$\text{Activation Map 2}: \begin{bmatrix} -4 & 1 \\ -6 & -9 \end{bmatrix}$$

Each activation map represents the response of a filter at different spatial positions. In our example, Filter 1 may be detecting certain visual features, like edges or patterns, while Filter 2 may be detecting different features.

To create the output volume, we stack the activation maps together:

$$\text{Output Volume}: \begin{bmatrix} 4 & 3 \\ 6 & 9 \\ -4 & 1 \\ -6 & -9 \end{bmatrix}$$

The output volume captures the responses of the filters and represents the transformed information from the input. By using multiple filters, the convolutional layer can capture and detect various visual features in the data.

This stacking process allows the network to capture and represent complex patterns in the data. The network learns to recognize specific visual features like edges or colors by adjusting the filter weights during the training process.

1.2.5.4 Pooling Layer

The Pooling layer, similar to the Convolutional Layer, is a part of a Convolutional Neural Network (CNN). Its purpose is to reduce the size of the convolved features. This helps in reducing the computational power needed to process the data by reducing its dimensionality. Additionally, the Pooling layer is useful for extracting important features that are not affected by their position or rotation in the image, which helps in training the model effectively.

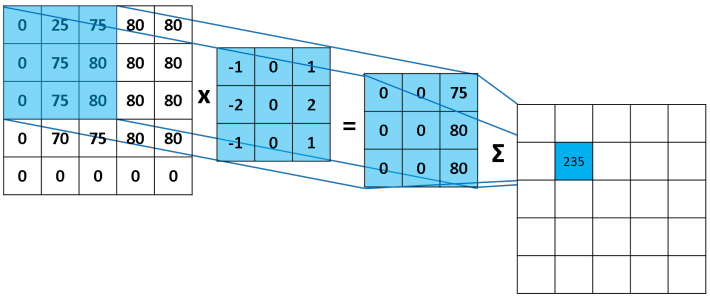

Figure 1.11: A step in the Convolution Process. [5].

There are two types of Pooling Max Pooling and Average Pooling as shown in Figure 1.12)

- **Max Pooling** selects the maximum value from a specific area of the image.
- **Average Pooling** calculates the average of all the values in that area.

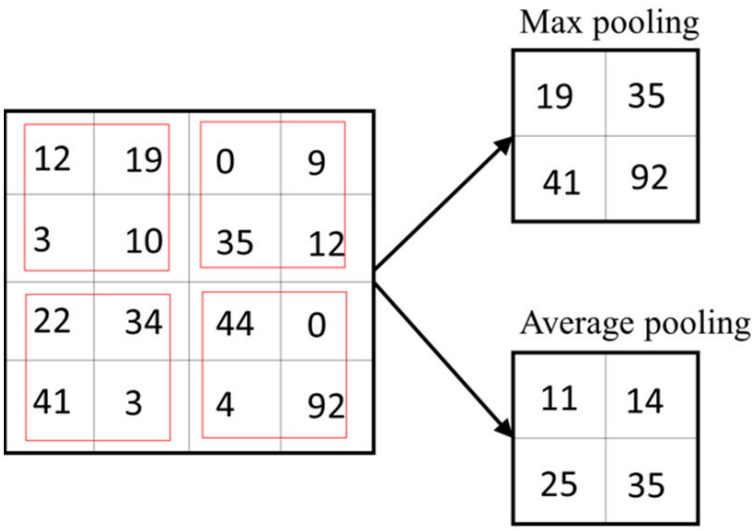

Figure 1.12: Types of pooling
[15]

Max Pooling not only reduces dimensionality but also acts as a noise suppressant. It discards noisy activations and helps in denoising the data. In contrast, Average Pooling mainly focuses on dimensionality reduction and also acts as a noise-suppressing mechanism, although not as effectively as Max Pooling.

Let's Suppose we have a 4x4 matrix representing a portion of an image:

$$\begin{bmatrix} 2 & 4 & 3 & 1 \\ 1 & 6 & 2 & 8 \\ 5 & 9 & 7 & 3 \\ 2 & 3 & 4 & 6 \end{bmatrix}$$

Now let's apply Max Pooling with a 2x2 kernel. We slide the kernel over the matrix, and at each step, we select the maximum value within the kernel's area to form the pooled output.

In this case, the pooled output would be:

$$\begin{bmatrix} 6 & 8 \\ 9 & 7 \end{bmatrix}$$

As we can see, each 2x2 kernel selects the maximum value within that region, resulting in a 2x2 matrix.

Now, let's apply Average Pooling with the same 2x2 kernel to the original matrix. We again slide the kernel over the matrix, and this time we calculate the average of all the values within the kernel's area.

The pooled output using Average Pooling would be:

$$\begin{bmatrix} 3.25 & 3.5 \\ 4.75 & 5 \end{bmatrix}$$

In this case, we take the average of the values within each 2x2 region to obtain the pooled matrix.

After applying pooling, the resulting matrix is typically downsampled and fed into the next layer, which can be a regular Neural Network for further processing and classification.

It's important to note that the **number of Convolutional and Pooling layers** in a CNN can vary depending on the complexity of the images being processed. Increasing the number of layers allows the network to capture more detailed information, but it also requires more computational power.

1.2.5.5 Flattening

After applying pooling operations to the feature map, we obtain a matrix of features. This matrix represents the important information extracted from the previous layers of the network. However, in order to process this information efficiently, we need to flatten the matrix.

Flattening the matrix involves transforming it into a single column by concatenating all the values in a sequential manner. This way, we create a long list of features that can be easily fed into the neural network for further processing.

let's say we have a 2×3 matrix:

$$\begin{bmatrix} 2 & 4 & 3 \\ 1 & 6 & 2 \end{bmatrix}$$

After flattening this matrix, we would get:

$$\begin{bmatrix} 2 & 4 & 3 & 1 & 6 & 2 \end{bmatrix}$$

Now, this flattened representation can be passed to the neural network, where each value in the column represents an input feature that the network can work with.

By flattening the pooled feature map, we ensure that the neural network can efficiently process the extracted features and make predictions or classifications based on them.

1.2.5.6 Fully Connected Layer

The Fully Connected Layer is an essential component of a neural network, particularly in the later stages of the model. Also known as the Dense Layer, it plays a crucial role in learning complex patterns and relationships from the features extracted by the earlier layers of the network. The Fully Connected Layer connects every neuron from the previous layer to every neuron in the current layer, creating a fully connected network structure.

When we apply convolutional and pooling layers to an input image, we obtain a feature map that represents the important features in the image. The Fully Connected Layer takes this flattened feature map as input and performs a series of matrix multiplications, followed by activation functions, to generate the final output or prediction.

To understand how the Fully Connected Layer let's works with this simple example. Suppose we have flattened the feature map from the previous layers, resulting in a 1-dimensional vector:

$$\begin{bmatrix} 2 & 4 & 3 & 1 & 6 & 2 \end{bmatrix}$$

Now, let's assume we have a Fully Connected Layer with 3 neurons. Each neuron in this layer has its own set of weights (represented as a vector) and a bias term. The number of neurons in the Fully Connected Layer is a hyperparameter that can be adjusted based on the complexity of the task and the desired network architecture.

For our example, let's consider the weight vectors and biases of the three neurons in the Fully Connected Layer:

$$\text{Neuron 1: } \begin{bmatrix} 0.5 & -0.2 & 0.8 & -0.6 & 0.3 & 0.1 \end{bmatrix}, \text{Bias: } 0.2$$
$$\text{Neuron 2: } \begin{bmatrix} -0.4 & 0.7 & 0.2 & -0.1 & 0.6 & 0.5 \end{bmatrix}, \text{Bias: } -0.3$$
$$\text{Neuron 3: } \begin{bmatrix} 0.1 & 0.6 & -0.3 & 0.4 & -0.5 & 0.2 \end{bmatrix}, \text{Bias: } 0.1$$

To calculate the output of each neuron, we perform a dot product between the weight vector and the input feature vector and add the bias term:

$$\text{Output of Neuron 1} = \sum_{i=1}^{6} (2 \times 0.5) + (4 \times -0.2) + (3 \times 0.8) + (1 \times -0.6) + (6 \times 0.3) + (2 \times 0.1) + 0.2$$

$$\text{Output of Neuron 2} = \sum_{i=1}^{6} (2 \times -0.4) + (4 \times 0.7) + (3 \times 0.2) + (1 \times -0.1) + (6 \times 0.6) + (2 \times 0.5) - 0.3$$

$$\text{Output of Neuron 3} = \sum_{i=1}^{6}(2\times 0.1)+(4\times 0.6)+(3\times -0.3)+(1\times 0.4)+(6\times -0.5)+(2\times 0.2)+0.1$$

After calculating the output for each neuron, we apply an activation function to introduce non-linearity to the network. Commonly used activation functions include ReLU (Rectified Linear Unit), Sigmoid, or Tanh.

Once the activation function is applied, the output of the Fully Connected Layer is obtained, and this output can be further processed by additional layers in the network or used for the final prediction.

Chapter 2

Plants Disease Prediction Using Structured Data

Predicting plant diseases is very important in farming. It helps farmers and growers be ready before a disease outbreak. The main aim is to foresee when and how a disease will happen, so that farmers can reduce the harm and take the right actions to stop the disease from spreading. Predicting plant diseases uses different methods, like weather data, crop history, and information about how often the disease has happened before.
In the prediction process, we gather and study data, find patterns of diseases, and figure out what things make the disease spread. We also make models that can tell us what might happen based on this information.
Plant disease prediction systems use technology to make this process automatic and use different methods, like machine learning and statistics, to make predictions. These systems try to give warnings early when diseases might happen, reduce the need for pesticides, and decrease the harm that plant diseases cause to the economy and environment. They are easy to use and can work with different plants and places, so they are useful tools for managing agriculture.

2.1 Introduction to Plant Disease Prediction

Plant disease prediction is a method of managing plant diseases by accurately predicting their occurrence or severity.The main idea is to figure out the chances of a disease outbreak or how much worse the disease will get. As computers get better and we have more data, disease prediction systems will become even more important in the future. Here are some examples of models we use to predict plant diseases:

- **California PestCast Disease Model Database[1]:** This is a database of plant disease models that describe the interaction between environmental, host, and pathogen variables that can result in disease. A model can be presented as a simple rule, an equation, a graph, or a table.
- **Model-Based Forecasting of Agricultural Crop Disease:** This model has been

[1] Available at `https://ipm.ucanr.edu/DISEASE/DATABASE/diseasemodeldatabase.html`

used to predict disease risk and outbreaks of Coffee Leaf Rust and wheat stripe rust. It involves using theoretical knowledge on crop growth, disease development, and environmental influences, alongside data from disease monitoring, climate/weather, and other explanatory variables for assessing and predicting disease. [16]

2.2 Structured Data in Plant Disease Prediction

2.2.1 Structured Data

Structured data is information that has been arranged and presented in a clear and organized manner so that people and machines can easily read and comprehend it. It is usually kept in databases or spreadsheets and allows for effective handling, examination, and use in applications that rely on data, such as business intelligence and machine learning.

2.2.2 Types and Sources of Structured Data

Structured data is very important when it comes to predicting plant diseases. It gives us useful information about different things that make plant diseases happen and how bad they can be. By studying and understanding this structured data, researchers and experts in this field can create models that help predict and control plant diseases.
Here are some examples of structured data types that are commonly used for predicting plant diseases:

- Environmental factors (temperature, humidity, rainfall)
- Soil properties (pH, nutrient content)
- Plant characteristics (leaf color, size, shape)
- Disease-related attributes (symptoms, severity)

the focus of this chapter will primarily be on environmental factors. These factors provide valuable insights into the conditions that contribute to disease occurrence and progression. This data can be collected from a variety of sources, such as:

- **On-site meteorological stations**:These are stations placed in or near the field that have sensors to measure different weather factors like temperature, humidity, rain, wind speed, and wind direction.
- **Remote sensing**: This process uses satellites, drones to take pictures of the field. These pictures are studied to find out weather information like temperature and moisture levels.
- **Weather forecast websites**: Many websites provide weather forecast information for specific locations, which can be used to plan agricultural activities.

2.2.3 Data Preprocessing Techniques for Structured Data

Data preprocessing is the process of getting data ready for analysis by cleaning, changing, and organizing it. It's an important step in any data science project because it makes machine learning models work better and give more accurate results.

Cleaning and andling missing values

In data preprocessing,cleaning the data is an important task . This means removing errors, duplicates, and irrelevant data.It's also really important to deal with missing information. There are different ways to do this:

- Deleting rows with missing values: This is the simplest approach but may result in data loss.
- Imputation: This involves replacing missing values with estimates. Several imputation methods exist, such as mean imputation ($\hat{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$), median imputation ($\hat{x} = \text{median}(x_1, x_2, \ldots, x_n)$), and k-nearest neighbors imputation.

Normalization and standardization

Normalization and standardization are techniques used to scale feature values(scaling feature means to have a mean of 0 (μ) and a standard deviation of 1 (σ). Typically done using min-max normalization ($x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$) or z-score normalization ($x_{\text{norm}} = \frac{x-\mu}{\sigma}$). for consistent ranges, which enhances machine learning model performance.

Feature scaling and transformation

Feature scaling and transformation are techniques used to adjust the values of different features in a way that helps machine learning models work better. By modifying the range or distribution of feature values, these methods make the features more compatible with the algorithms used in machine learning.

2.3 Feature Engineering and Selection

Feature engineering is an important step in plant disease prediction, as it can help to improve the accuracy and performance of predictive models. Here are some reasons why feature engineering is important for plant disease prediction:

- Noise reduction:. [17]
- Compatibility with machine learning algorithms [17]
- Improved interpretability . [17]

After getting the features we need or making new ones, we might have too many of them. This can be a problem because having lots of features can make it hard to see and understand the data, and it can also make the machine learning algorithms slow down.

To fix this, we can use dimensionality reduction techniques. These techniques help us decrease the number of features in a structured dataset. This makes it easier to work with and can make the machine learning algorithms run faster. Some common techniques for dimensionality reduction include:

- **Principal component analysis (PCA):** PCA is a linear dimensionality reduction technique that projects data points onto a lower-dimensional subspace that retains as much of the variance of the original data as possible. [18]
- **Linear discriminant analysis (LDA):** LDA is a linear dimensionality reduction technique that projects data points onto a lower-dimensional subspace that maximizes the separation between two or more classes. [19]
- **Feature selection using correlation analysis:** Feature selection using correlation analysis can be used to select a subset of features that are highly correlated with the target variable. This can help to reduce the dimensionality of the data and improve the performance of machine learning algorithms. [19]

2.3.1 Principal Component Analysis (PCA)

PCA, or Principal Component Analysis, is a statistical technique used for dimensionality reduction. It aims to transform the original set of variables into a new set of uncorrelated variables called principal components. These new variables are ordered in such a way that the first principal component captures the most variance, the second captures the second most, and so on.

The process of PCA involves two main steps: transformation and selection.

The Transformation Step

Given the standardized data matrix Z with n observations and p variables, where each row represents an observation, and each column represents a standardized variable, the correlation matrix A can be computed as follows:

$$A = \frac{1}{n-1} Z^T \cdot Z$$

Where Z^T is the transpose of matrix Z.

Next, we need to find the eigenvectors and eigenvalues of the correlation matrix A. Let $\mathbf{v}_i$ represent the eigenvector and λ_i represent the corresponding eigenvalue. The eigenvectors are unit vectors, which means $\mathbf{v}_i^T \cdot \mathbf{v}_i = 1$. The eigenvectors are the columns of the orthogonal matrix P, and the eigenvalues are the elements of the diagonal matrix D.

$$A\mathbf{v}_i = \lambda_i \mathbf{v}_i$$

After computing the eigenvectors and eigenvalues, we form the orthogonal matrix P by stacking the eigenvectors $\mathbf{v}_i$ as columns:

$$P = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_p]$$

Now, we can perform the transformation by multiplying the standardized data matrix Z with the orthogonal matrix P:

$$ZP = Z \cdot P$$

Where ZP is the new transformed data matrix, and each column of ZP represents a principal component.

The Selection Step

In the selection step, we need to decide how many principal components to keep in order to reduce the dimensionality of the data. This decision is usually based on the eigenvalues. The eigenvalues represent the amount of variance explained by each principal component. We sort the eigenvalues in descending order, and then we choose the first k principal components, where k is the desired reduced dimensionality.

The transformed data matrix ZP now becomes our new dataset with reduced dimensions, containing only the first k principal components.

Mathematically, let D_k be the diagonal matrix containing the first k largest eigenvalues and P_k be the orthogonal matrix containing the corresponding eigenvectors:

$$D_k = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_k \end{bmatrix}$$

$$P_k = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]$$

Then, the reduced data matrix ZP_k containing the first k principal components is given by:

$$ZP_k = Z \cdot P_k$$

The reduced data matrix ZP_k now represents the new dataset with reduced dimensions, containing only the first k principal components.

2.4 Classification Models for Plant Disease Prediction

Plant disease prediction using machine learning is becoming popular because it has the potential to help farmers save money and increase crop yields. However, it is important to note that this technology is still in its early stages of development and requires expertise in a variety of fields, including plant disease study, statistics, and algorithms.
Countries like the United States, Brazil, and China are investing a lot of money in this field because plant epidemics cause huge financial losses. According to a study by the Food and Agriculture Organization of the United Nations, plant diseases cost the global economy an estimated 220 billion each year. [20] Recent research shows that there has been a significant increase in studies using machine learning to predict agricultural diseases. This progress is because of the advancements made in artificial intelligence in the 21st century. [21] [22]

Table 2.1 presents a bibliographic synthesis of research studies conducted in the field of agricultural disease prediction using various machine-learning techniques.

Article	Plant/Crop	Diseases	Models/Algorithms
[23]	Citrus	Gummosis	SVR
[24]	Grape	Downy mildew, powdery mildew, anthracnose	RM for crop disease forecasting. ANN with K-NN for weather forecasting
[25]	Mango	Anthracnose	RF, SVM
[26]	Oil palm	Ganoderma	ANN
[27]	Potato	Late Blight	SVM
[28]	Potato	Late Blight	ANN, SVM

Tableau 2.1: Plant Disease Detection Models/Algorithms

we affirm that there are many other research studies on plant disease prediction that we was unable to review.However, from our thorough research, we noticed a big difference in Algeria compared to other advanced countries. The reason for this gap in research is the lack of cooperation between universities in different fields of study. Predicting agricultural diseases needs input from various areas of expertise.

In 1.2.2, we have already discussed artificial neural networks. In this section, the focus will be on support vector machines (SVMs)

2.4.1 Support Vectors Machines

The definitions and explanations in this section are primarily sourced from the following reference: An Introduction to Statistical Learning [6]

Maximal Margin Classifier

The maximal margin classifier is a method used to separate different classes by using a hyperplane. But what exactly is a hyperplane? In simple terms, a hyperplane is a flat surface that can be thought of as a line in a two-dimensional space or a flat plane in a three-dimensional space.

Mathematically, a hyperplane can be represented by an equation in the form:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p = 0 \tag{2.1}$$

This equation describes the relationship between the variables $X_1, X_2, \ldots, X_p$ and the coefficients $\beta_0, \beta_1, \beta_2, \ldots, \beta_p$. If a point $(X_1, X_2, \ldots, X_p)$ satisfies this equation, it means that the point lies on the hyperplane shown in Figure 2.1. If the point does not satisfy the equation, it means that it lies on one side of the hyperplane.

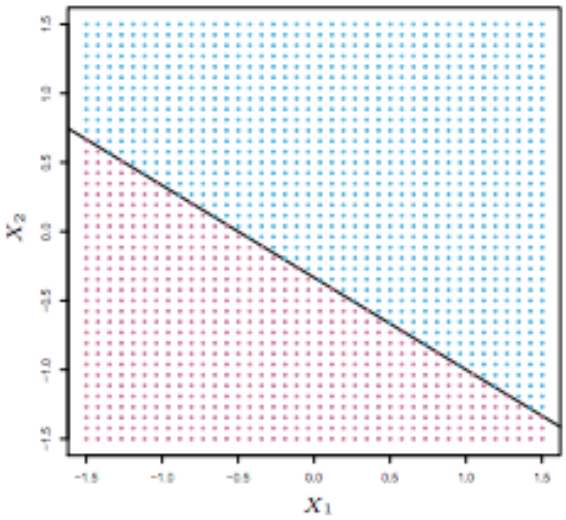

Figure 2.1: Hyperplane in a 2D space [6]

The goal of the maximal margin classifier is to find a hyperplane that maximizes the distance between the hyperplane and the nearest data points of different classes. By maximizing this distance, we can achieve better separation between the classes and improve the classifier's performance.

when separating data using a hyperplane, there may be multiple options that perfectly separate the data. To determine the best hyperplane, we use the concept of the maximal margin or optimal separating hyperplane. This hyperplane is the farthest from the data points, and its selection is based on maximizing the margin, which is the distance between the hyperplane and the data points. By choosing the hyperplane with the largest margin, We achieve improved separation between different data classes, enhancing the performance of our classification model(see Figure 2.2). What if no separating plane exists(Figure 2.3)?

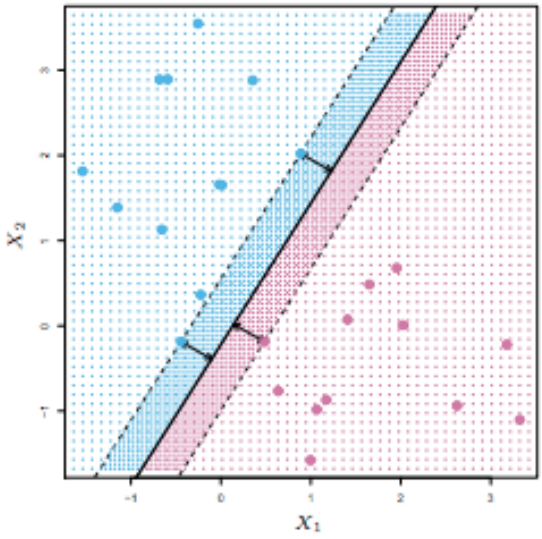

Figure 2.2: maximal margin hyperplane [6]

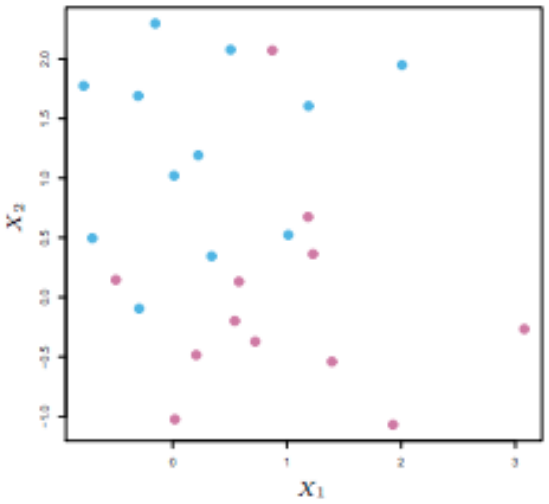

Figure 2.3: Non-Separable Classes [6]

in this situation, we don't have a maximal margin classifier. Instead, we use a support vector classifier that can somewhat separate the classes using a soft margin.

Support Vector Machine (SVM)

The Support Vector Machine (SVM) is an extension of the support vector classifier that uses kernels to enlarge the feature space in a specific way, allowing it to handle non-linear boundaries between classes. The SVM involves inner products of observations, which measure their similarity. These inner products are used to compute the coefficients of the linear support vector classifier.

The linear support vector classifier can be represented as:

$$f(x) = \beta_0 + \sum_i \alpha_i h(x, x_i) \tag{2.2}$$

where $f(x)$ is the predicted outcome, β_0 is the intercept, α_i are the parameters for each training observation, and $h(x, x_i)$ represents the inner product between the new point x and each training point x_i.

To generalize the similarity measurement, we replace the inner product with a kernel function, denoted as $K(x, x_i)$.

Different kernel functions quantify similarity in different ways. For example, a linear kernel

$$K(x, x_i) = \sum_i x_j x_{ij} \tag{2.3}$$

represents the original support vector classifier.

A polynomial kernel

$$K(x, x_i) = (1 + \sum_i x_j x_{ij})^d \tag{2.4}$$

of degree d offers a more flexible decision boundary.

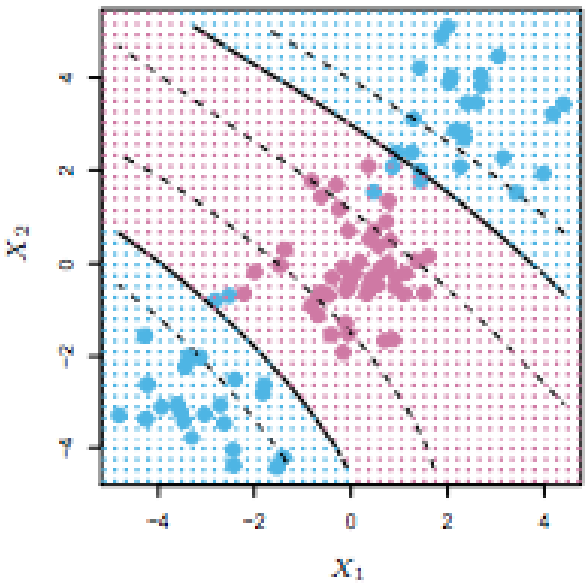

Figure 2.4: An SVM with a polynomial kernel [6]

By using a non-linear kernel, such as a polynomial kernel, the SVM fits the data in a higher-dimensional space involving polynomial terms. This allows it to handle complex patterns that cannot be captured in the original feature space.

2.5 Evaluation Metrics for Prediction Models

Evaluation metrics are essential for assessing the performance and effectiveness of prediction models for plant disease prediction. By using the right evaluation metrics, researchers and practitioners can gain insights into the accuracy, precision, and overall predictive capabilities of their models. The following are some of the most commonly used evaluation metrics in plant disease prediction:

2.5.1 Confusion Matrix

A confusion matrix is a table that shows the predicted and actual values of a classification model. It can be used to calculate accuracy, precision, recall, and F1-score.
We can represent confusion matrix as follows:

Predicted Class		
Actual Class	Yes	No
Yes	TP	FN
No	FP	TN

Tableau 2.2: Confusion Matrix

where :

- True Positives (TP): The classifier correctly predicted "Yes".
- True Negatives (TN): The classifier correctly predicted "No".
- False Positives (FP): The classifier incorrectly predicted "Yes" for samples that were actually "No".
- False Negatives (FN): The classifier incorrectly predicted "No" for samples that were actually "Yes".

2.5.2 Accuracy

Accuracy is a commonly used metric to evaluate the performance of a prediction model. It measures the percentage of correct predictions made by the model out of the total number of predictions.
To calculate accuracy, we divide the number of correct predictions by the total number of predictions and multiply the result by 100 to express it as a percentage.

accuracy can be represented as:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \times 100 \tag{2.5}$$

2.5.3 Precision, Recall, and F1-Score

Precision measures the percentage of positive predictions that are actually positive. It is calculated by dividing the number of true positives (TP) by the sum of true positives and false positives (FP). In other words, precision captures how precise or accurate the model is when it predicts positive instances.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.6}$$

Recall, also known as sensitivity or true positive rate, measures the percentage of actual positive instances that are predicted as positive by the model. It is calculated by dividing the number of true positives (TP) by the sum of true positives and false negatives (FN). Recall captures the model's ability to identify positive instances from the entire set of actual positive instances.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2.7}$$

F1-score is a harmonic mean of precision and recall, providing a balanced measure of a model's performance. It is calculated by taking twice the product of precision and recall divided by their sum. The F1-score reaches its best value at 1 (perfect precision and recall) and worst value at 0 (poor precision and recall).

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.8}$$

These metrics are often used in conjunction with the confusion matrix, which summarizes the performance of a classification model.

Chapter 3

Plants Disease Identification Using Unstructured Data

3.1 Unstructured Data in Plant Disease Prediction

Unstructured data in plant disease prediction refers to data that is not organized or structured in a specific way, such as images of diseased and healthy plant leaves collected under controlled conditions. Machine learning and deep learning approaches can be employed to analyze this unstructured data for the identification and classification of plant diseases.

Sources of Unstructured Data

unstructured data from various sources can be used to develop effective strategies for plant disease prediction and management. here are some sources of unstructured data:

- Sensor data: Sensor data from various sources, such as drones, satellites, and ground-based sensors, can be used to detect and monitor plant diseases [29].
- Machine learning models: Machine learning models can generate unstructured data that can be used to improve the accuracy and reliability of plant disease prediction models [30].

using unstructured data for plant disease prediction can provide several benefits that can help improve plant health, reduce crop losses, and increase food security

3.2 Image Processing and Computer Vision Techniques

Image processing and computer vision methods had a big influence in various fields, including plant science. As mentioned by Hannah Dee [31], these techniques use digital image analysis to get important information from plant pictures. This helps researchers and experts learn more about plants, identify problems, and make advancements in plant-related research.

What is Image Processing?

Image processing is a big area that includes many different methods for changing and studying digital pictures. Here is some of the most common techniques used in image processing :

Filtering: This technique is used to modify the pixel values in an image to improve its visual quality or to extract specific features. For example, a blur filter can be used to smooth out an image, while a sharpen filter can be used to increase the contrast between adjacent pixels.

Segmentation: This technique is used to divide an image into different regions based on their properties, such as color, texture, or intensity. This can be used to identify objects in an image, to extract specific features from an image, or to simplify an image for further processing.

Edge detection: This technique is used to identify the edges in an image. Edges are important because they can be used to identify objects in an image, to measure the shape of objects, or to track the movement of objects.

Image restoration: This technique is used to improve the quality of an image that has been degraded by noise, blur, or other artifacts. This can be done by using algorithms to remove the noise or blur.

What is Computer Vision?

Computer vision is a powerful tool that can be used to automate plant image analysis and derive valuable insights efficiently. Here is some of the ways that computer vision can be used in plant image analysis include:

Image recognition: This is the process of identifying objects in an image. Computer vision can be used to identify different plant species, as well as the presence of pests or diseases.

Object detection: This is the process of finding and locating objects in an image. Computer vision can be used to find and locate pests or diseases on plants.

Image classification: This is the process of assigning a label to an image. Computer vision can be used to classify plant images according to their species, health, or other factors.

Tracking: This is the process of following the movement of an object in an image or video. Computer vision can be used to track the growth of plants over time or to monitor the movement of pests or diseases.

Once features have been extracted, they can be used to train a machine learning model to perform a variety of tasks, such as:

- Pest and disease detection
- Plant identification
- Plant health assessment.

3.3 Convolutional Neural Networks Application in Agriculture

Advanced techniques in deep learning, especially Convolutional Neural Networks (CNN), have made important progress in analyzing images. Many research studies have been done to automatically recognize plant diseases. This increasing interest and use create a chance to create automatic image methods for diagnosing plant diseases, recognizing plants, counting fruits, and detecting weeds. These technologies can greatly help farmers by improving farming practices, enhancing agriculture methods, and increasing food security.

In the past five years, the researcher has presented multiple methods for detecting plant leaf diseases based on CNNs. Notably, most of these publications have emerged after 2016, highlighting the novelty and modernity of this approach in agriculture. Figure 3.1 illustrates the number of research articles published between 2013 and 2022 on the automatic detection of plant leaf diseases using CNNs. The graph clearly shows a peak in automatic disease detection research in 2016, confirming the emerging nature of this field. [7]

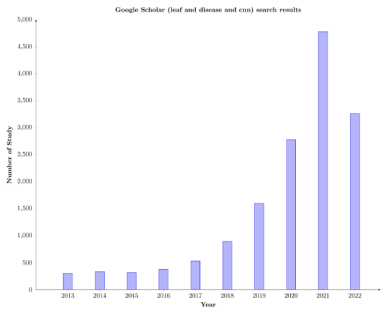

Figure 3.1: Number of articles published for the detection of plant leaf diseases using CNN from 2013 to 2022 [7]

3.3.1 VGGNet

3.3.1.1 Introduction to VGG

VGGNets are a special type of computer networks called convolutional neural networks (CNNs) that were created by the Visual Geometry Group (VGG) at the University of Oxford. These networks, known for their simplicity and effectiveness, have shown great success in tasks related to classifying images.

The original VGGNet, called VGG16, was introduced in 2014. VGG16 has 16 layers specifically designed for image analysis, along with 3 fully connected layers. The image analysis layers use 3x3 filters, while the pooling layers use 2x2 filters. VGG16 was trained on a large dataset called ImageNet, which has over 1.2 million images covering 1,000 different object categories. Remarkably, VGG16 achieved a top-5 error rate of 7.3% on the ImageNet test set, which is a significant improvement over previous state-of-the-art performance.

3.3.1.2 VGG Architectures

VGGNet is a popular neural network used for analyzing images. It is known for using small filters and pooling layers to process images effectively. The filters act like small windows that move across the image, searching for patterns. By recognizing smaller details and combining them, the network gains a better understanding of the whole picture.

Pooling layers, specifically max pooling, are important in VGGNet. They help reduce the amount of information the network needs to process. Max pooling focuses on selecting the most important features in a specific image region while discarding less important information. This makes the network faster and more efficient in analyzing images.

Another important feature of VGGNet is the presence of multiple convolutional layers. These layers work together to learn different features in an image. As the layers progress, they understand more complex patterns. Having more convolutional layers, like in VGGNet, allows the network to learn finer details and recognize more sophisticated features.

VGGNet has two main versions: VGG-16 (illustrated in figure 3.2). The numbers in their names indicate the number of convolutional layers in each design. VGG-16 has 16 convolutional layers. Generally, networks with more layers are more powerful in capturing subtle aspects of an image.

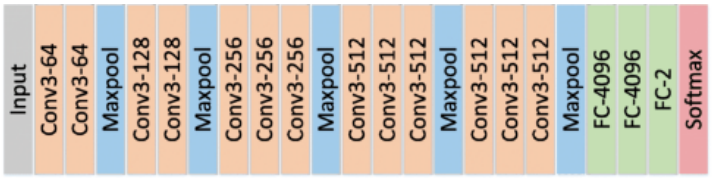

Figure 3.2: VGG-16 Architecture [8]

3.3.1.3 VGG Applications in Agriculture

The use of VGG neural networks has been widely studied in agriculture, specifically for detecting crop diseases through analyzing images. Several research papers have examined the effectiveness of VGG models in this field. For example, Mkonyi et al. (2020) utilized VGG-16 and achieved a 91.9% accuracy in early identification of Tuta absoluta, a harmful pest that affects tomato plants.

Similarly, Wang et al. (2017) conducted a study using VGG-16 to detect crop diseases and obtained a 90.4% accuracy on a test dataset of images. Their findings further support the effectiveness of VGG-16 in recognizing and categorizing crop diseases based on image analysis.

Furthermore, Darwish et al. (2020) proposed an optimized model that combines VGG-16 with the orthogonal learning particle swarm optimization algorithm for diagnosing plant diseases. Their study reported an impressive 98.2% accuracy in detecting crop diseases using VGG-16. This research showcases the potential of VGG models in achieving highly accurate disease diagnosis in agriculture.

Overall, these studies demonstrate the valuable applications of VGG neural networks in agriculture, particularly in detecting crop diseases. The high accuracy rates achieved by VGG-

16 model emphasize its potential to transform disease identification and management in the agricultural industry.

3.3.2 MobileNet

3.3.2.1 Introduction to MobileNet

MobileNets are special types of computer networks known as convolutional neural networks (CNNs). They were designed to work efficiently and be lightweight, which makes them great for mobile phones and other small devices. They were created to solve the problem of using complex networks on devices with limited resources.

The original MobileNet was made by Google researchers in 2017. Unlike regular CNNs, MobileNets use special types of convolutions called depth-wise separable convolutions. These convolutions help reduce the amount of calculations needed, so the network can process images faster and use less resources. This makes MobileNets good for analyzing images in real-time on mobile devices.

3.3.2.2 MobileNet Architecture

MobileNets are known for being simple and effective in image analysis. They are efficient because they use two important techniques: depth-wise convolutions and point-wise convolutions.

Depth-wise convolutions apply a specific filter to each part of the image, without needing to consider all the parts together. This saves a lot of calculations and helps the network work faster.

Point-wise convolutions, also known as 1x1 convolutions, combine and change the features obtained from the depth-wise convolutions. This helps MobileNets learn complex relationships between different parts of the image in an efficient way.

Using these depth-wise and point-wise convolutions makes MobileNets lightweight and perfect for devices with limited resources, like smartphones and small computers.

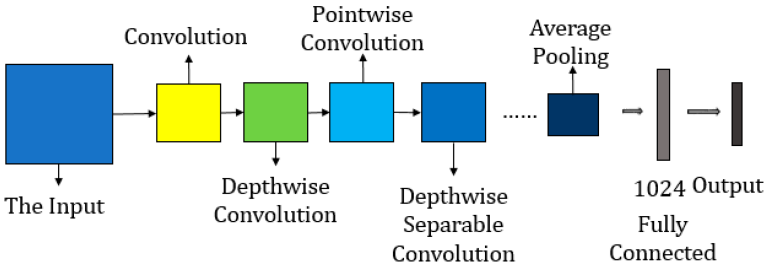

Figure 3.3: MobileNet Architecture [9]

3.3.2.3 MobileNet Applications in Agriculture

MobileNet-based Convolutional Neural Networks (CNNs) have shown promising results in various agricultural applications, especially in detecting and identifying plant diseases. Many

studies have investigated the effectiveness of MobileNet architectures in this field. In the paper "MobileNet Based Apple Leaf Diseases Identification," the authors propose a method that uses MobileNet to efficiently and accurately identify apple leaf diseases, reducing the workload for experts who would otherwise do it manually [32]. In "Automated Plant Leaf Disease Detection and Classification using Optimal MobileNet Based CNNs," MobileNet-based CNNs were used to categorize plant diseases precisely into two types based on whether the disease is present or absent, offering an automated and accurate solution for detecting plant diseases [33]. Another research work, "Identifying Crop Diseases using Attention-Embedded MobileNet-V2 Model," explores the application of MobileNet-V2 model with attention mechanisms, showing better performance compared to other advanced models in accurately identifying crop diseases [34].

3.4 Preprocessing and Augmentation of Plant Images

Preprocessing and augmentation are essential steps in preparing plant images for effective analysis and improving the performance of plant disease prediction models.

Preprocessing

Preprocessing is an essential step in image processing where we apply various techniques to improve the quality of images and eliminate any unwanted elements.

- **Noise removal**: Images often suffer from random disturbances called noise, which can appear as graininess or black and white dots. To counter this, we employ methods to eliminate noise, such as Gaussian noise (random variations) or salt and pepper noise (randomly occurring black and white pixels). [35]
- **Contrast enhancement**: By increasing the contrast of an image, we amplify the distinction between light and dark areas. This adjustment enhances the visual details and makes them more apparent, facilitating better observation and analysis. [36]
- **Brightness adjustment**: Manipulating the overall brightness of an image can significantly impact its visibility. By adjusting the brightness, we can make the image brighter or darker, thereby enhancing the visibility of the underlying features. [37]
- **Color balancing**: Colors play a crucial role in perceiving and comprehending images. In color balancing, we aim to harmonize the colors in the image, making them appear more natural and balanced. This adjustment aids in better feature recognition and analysis. [37]
- **Feature extraction**: Extracting relevant features from an image allows us to isolate and analyze specific attributes. For instance, we can identify and extract features like leaf color, leaf shape, and leaf size. This process helps us understand the image content better and enables further analysis and interpretation.

Augmentation

Augmentation is a process in which we create new images by applying different changes to the original images. This help to increase the variety of data available for training machine learning models.

- **Rotation**: We rotate the image by a specific angle. This can be useful for scenarios where the orientation of objects in the image may vary. [38] [39]
- **Scaling**: We resize the image by increasing or decreasing its size. Scaling up can help us focus on finer details while scaling down can help us capture a broader view. [39]
- **Translation**: We shift the image horizontally or vertically by a certain amount. This can simulate changes in the position of objects within the image. [40]
- **Shear**: We deform the image by tilting or slanting it in a specific direction. This can be helpful in scenarios where objects are not aligned in a straight manner. [41]
- **Brightness adjustment**: We modify the overall brightness of the image. Increasing or decreasing the brightness can mimic variations in lighting conditions. [39]
- **Contrast enhancement**: We amplify the difference between light and dark areas in the image. This adjustment helps in emphasizing the details and making them more distinguishable. [39]
- **Color balancing**: We adjust the color distribution in the image to make it more visually appealing and realistic. [41]
- **Noise addition**: We introduce random variations or disturbances into the image, such as graininess or speckles. This can help models learn to be robust in handling noisy data. [40]

These techniques can help increase the size of the dataset and improve the robustness of the model. A larger dataset will help the model to learn more about the different types of plant diseases.

Chapter 4

EXPERIMENTS AND RESULTS

4.1 Introduction

In this chapter, we'll use different types of machine learning and deep learning and convolutional neural network architectures to predict plant diseases based on weather parameters and identify diseases using plant images. The main goal of this chapter is to see how well these models work in solving the problem of accurately diagnosing and predicting plant diseases.

Research Question: Can we accurately predict plant diseases based on weather parameters and identify diseases using convolutional neural networks, and how do different models perform in comparison?

4.1.1 Models for Plant Disease Prediction

In this section, we describe the three models utilized for plant disease prediction based on weather parameters:

a) **Logistic Regression with PCA:** We'll start with logistic regression basic model, which is a simple model used as a starting point for predicting plant diseases. because our data set have the problem of multicollinearity , we apply principal component analysis (PCA) to reduce the dimensionality of the input features while preserving most of the variance. We then train logistic regression on these principal components.

b) **Artificial Neural Network (ANN):** We'll use an artificial neural network to find more complex relationships between weather data and plant diseases. This network can handle large and diverse datasets,making it suitable for this prediction task.

c) **Support Vector Machine (SVM):** SVM is a powerful model for classification tasks. We employ the SVM with a radial basis function (RBF) kernel to classify plant diseases based on weather parameters.

4.1.2 CNN Architectures for Disease Identification

Now, we'll talk about the special neural networks used for identifying diseases in pictures of plants:

a) **VGG16 and VGG19:** We'll use two neural networks, called VGG16 and VGG19, which are known for being very good at extracting features from images. We'll fine-tune these networks to recognize diseases in our plant images.

b) **MobileNetV1:** We'll also try MobileNetV1, which is a more efficient neural network suitable for situations where we have limited resources. We'll see how well it identifies diseases while still being computationally efficient.

4.1.3 Tools

This section will discuss the environment and tools we used to train our models. We implemented the algorithms using Python and trained them with the TensorFlow framework using the Keras library. Additionally, we used the Kaggle platform for model training.

4.1.3.1 Python

Python is a high-level, interpreted programming language known for its simplicity, readability, and versatility. It emphasizes code readability and allows developers to write concise yet expressive code.

4.1.3.2 TensorFlow

TensorFlow is an open-source software library for machine learning and artificial intelligence. It was originally developed by researchers and engineers at Google Brain, and it is now used by a wide range of companies and organizations.

4.1.3.3 Keras

Keras is a high-level API that makes it easy to build and train deep-learning models. It is designed to be user-friendly and to provide a high level of abstraction so that developers can focus on the model architecture and not the underlying implementation details.

4.1.4 Kaggle

Kaggle is a platform that hosts various machine learning and data science competitions. It provides a community of data scientists and enthusiasts with datasets, notebooks, and a collaborative environment for solving complex problems and improving data-driven models.

4.2 Dataset

In this section, we will discuss the process of data collection and the steps taken to prepare it for modeling.

4.2.1 Dataset Presentation

In our study we have 2 types of data: tabular data containing weather information and images of plant leaves with diseases.

Meteorological Dataset

For the weather-related data, we use a meteorological dataset that may have an impact on plant health and the occurrence of diseases. The dataset was sourced from Visual Crossing Weather[1], which collects daily meteorological data from various weather stations across Algiers.

The dataset includes measurements of several variables 4.1:

Element	Description	Metric
datetime	YYYY-MM-DD HH:MM:SS	-
temp	Min,Max and Avg Temperature	C
dew	Dew Point	C
humidity	Relative Humidity	%
precip	Precipitation	mm
precipprob	Precipitation chance	%
precipcover	Precipitation cover	–
snow	Snow	cm
snowdepth	Snow Depth	cm
windspeed	Wind Speed	km/h
winddir	Wind Direction	degrees
visibility	Visibility	km
cloudcover	Cloud Cover	%
pressure	Sea Level Pressure	mb
solarradiation	Solar Radiation	W/m^2
solarenergy	Solar Energy	MJ/m^2
uvindex	UV Index	–

Tableau 4.1: Weather Data

Plant leaf disease images

In addition to the tabular data, we also have a collection of images showing plant leaves affected by different diseases 4.1.These images were gathered from PlantVillage, a repository of plant disease images available on Kaggle. Each image corresponds to a specific case of a plant disease, and we use these images to train and evaluate the performance of our convolutional neural network models for disease identification.

[1]A weather data provider that offers a wide range of weather-related services and solutions.

Figure 4.1: plants leaf images sample

4.2.2 Disease Risk Assessment

Tomato Late Blight Model

The dataset used in this study includes daily measurements of weather factors.These factors are important for evaluating the risk of tomato late blight. The specific variables used in the model are:

- Average daily temperature in Celsius (Tav)
- Minimum daily temperature in Celsius (Tmin)
- Average daily humidity (RHav)
- Rainfall accumulation in millimeters over 24 and 48 hours (R)

To calculate the risk index for tomato late blight, a model with polynomial and quadratic functions is used. The model follows these steps:

- Calculate the daily TIndex using the following equation:

$$TIndex = (-2.19247 + 0.259906 \cdot T - 0.000139 \cdot T^3 - 6.095832 \times 10^{-6} \cdot T^4) \cdot Fc$$

Here, **T** represents the daily average temperature in **Celsius**, and **Fc** is a correction factor calculated as:

$$Fc = 0.35 + 0.05 \cdot Tmin$$

Tmin is **the minimum temperature** for the day in Celsius.

- Calculate the daily RHIndex using the following equation:

$$RHIndex = -34.9972725 + 0.751 \cdot RH - 0.003909 \cdot RH^2$$

RH represents **the average daily humidity**.

- Calculate the daily Rindex using the following equation:

$$Rindex = 0.006667 + 0.194405 \cdot R + 0.0002239 \cdot R^2$$

R corresponds to **the accumulated rainfall** in millimeters over the last 48 hours.

- Determine whether the **Rindex** or **RHIndex** is higher, and multiply it by **the TIndex** to find **the daily IPI** (Integrated Pest Index).
- **The IPI** is calculated only when certain weather conditions are met:
  - Tmin is above 7°C
  - Tav is between 9°C and 25°C
  - Total rainfall is greater than 0.2 mm
  - RHav is above 80%
- **The daily IPI** is then compared to a threshold:
  - IPI values below 15 indicate no disease risk
  - An IPI above 15 indicates a high risk of tomato late blight.

By using this model in our analysis, we can assess the risk of tomato late blight based on daily meteorological data.

After assesing the disease risk to our inetial dataset the view of the data gives the following:

	datetime	tempmax	tempmin	temp	feelslikemax	feelslikemin	feelslike	dew	humidity	precip	...	snowdepth	windspeed	winddir	sealevelpressure	cloudcover	visibility	solarradiation	solarenergy	uvindex	disease
0	2012-01-01	18.7	6.3	12.0	18.7	4.7	11.5	6.1	71.3	0.0	...	0.0	11.0	147.1	1028.1	3.9	9.9	129.2	11.1	5	1
1	2012-01-02	19.0	6.0	13.2	19.0	4.6	13.0	7.8	73.2	0.0	...	0.0	30.5	247.3	1027.8	20.6	10.3	126.1	10.8	5	1
2	2012-01-03	17.4	9.2	13.9	17.4	8.7	13.9	9.4	76.2	0.0	...	0.0	23.7	265.4	1031.6	40.1	10.5	130.0	11.4	5	1
3	2012-01-04	17.4	7.1	12.4	17.4	5.3	12.0	8.5	79.3	0.0	...	0.0	25.2	234.4	1031.6	27.8	10.4	133.5	11.5	5	1
4	2012-01-05	18.8	9.0	13.6	18.8	7.8	13.4	7.7	71.2	0.0	...	0.0	27.5	230.5	1027.7	44.4	10.4	126.2	10.9	5	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4013	2022-12-27	22.0	6.0	13.5	22.0	4.5	13.3	8.3	73.4	0.0	...	0.0	9.4	256.9	1029.1	22.9	9.7	120.3	10.5	5	1
4014	2022-12-28	25.0	5.0	13.2	25.0	4.0	13.0	6.6	68.8	0.0	...	0.0	11.2	193.5	1026.0	11.7	9.0	121.7	10.6	5	1
4015	2022-12-29	19.0	6.0	12.9	19.0	4.5	12.7	8.5	75.8	0.0	...	0.0	16.6	238.5	1025.0	37.6	10.0	100.7	8.8	4	1
4016	2022-12-30	21.0	6.0	13.3	21.0	3.6	12.9	7.1	70.0	0.0	...	0.0	15.7	215.0	1026.9	26.2	9.8	104.6	9.1	4	1
4017	2022-12-31	23.0	5.0	13.6	23.0	4.0	13.4	5.6	62.2	0.0	...	0.0	9.4	26.8	1027.9	6.8	10.3	119.2	10.3	5	1

4018 rows × 23 columns

Figure 4.2: dataset

The target variable (disease) distrtibuted as the following :

	0 (disease risk)	**1 (no risk of disease)**
1	289	3729

Observations from the correlation matrix 4.3:

- **Temperature Variables (tempmax, tempmin, temp):** these variables have strong positive correlation with one another. This is expected because higher maximum temperatures (*tempmax*) are likely to be linked to higher minimum temperatures (*tempmin*) and overall higher temperatures (*temp*).
- **Dew Point (*dew*):**Dew has a positive correlation with temperature variables (tempmax, tempmin, temp). This is reasonable because dew point is related to the moisture content in the air and tends to be higher in warmer conditions.
- **Humidity:**Humidity has a negative correlation with temperature variables (tempmax, tempmin, temp). Warmer temperatures are usually associated with lower humidity levels, as warm air can hold more moisture.
- **Precipitation and Snow-related Variables (precip, precipprob, precipcover, snow, snowdepth):** these variables snow-related variables (snow, snowdepth) show positive correlation with each other. It indicates that when there is precipitation, it is more likely to result in snow when the temperature is low.
- **Wind Speed (*windspeed*) and Wind Direction (*winddir*):**these variables have a weak correlation with most other variables. This suggests that these meteorological features are less influenced by other factors in the dataset.
- **Solar-related Variables (solarradiation, solarenergy, uvindex):** these variables have positive correlation with each other. This indicates that they are related to the intensity of sunlight during the day.

4.3 Models for Plant Disease Prediction

4.3.1 Logistic Regression

The main objective of logistic regression is to estimate the correct values for the parameters (β) and examine the relationship between the risk of disease (Y) and weather descriptions (X).
The relationship is represented by $Y = f(X, \beta)$, where Y indicates the absence of disease (1) or the presence of disease (0).
The process of parameter estimation involves finding the optimal values for the parameters (β) by maximizing the likelihood of observing the given data.

Based on 4.2 the variables that are statistically significant at the 0.05 significance level ($\alpha = 0.05$) are: temp, dew, humidity, precip, precipcover, snow and snowdepth
The variables *tempmax*, *tempmin*, *precipprob*, *windspeed*, *winddir*, *sealevelpressure*, *cloudcover*,

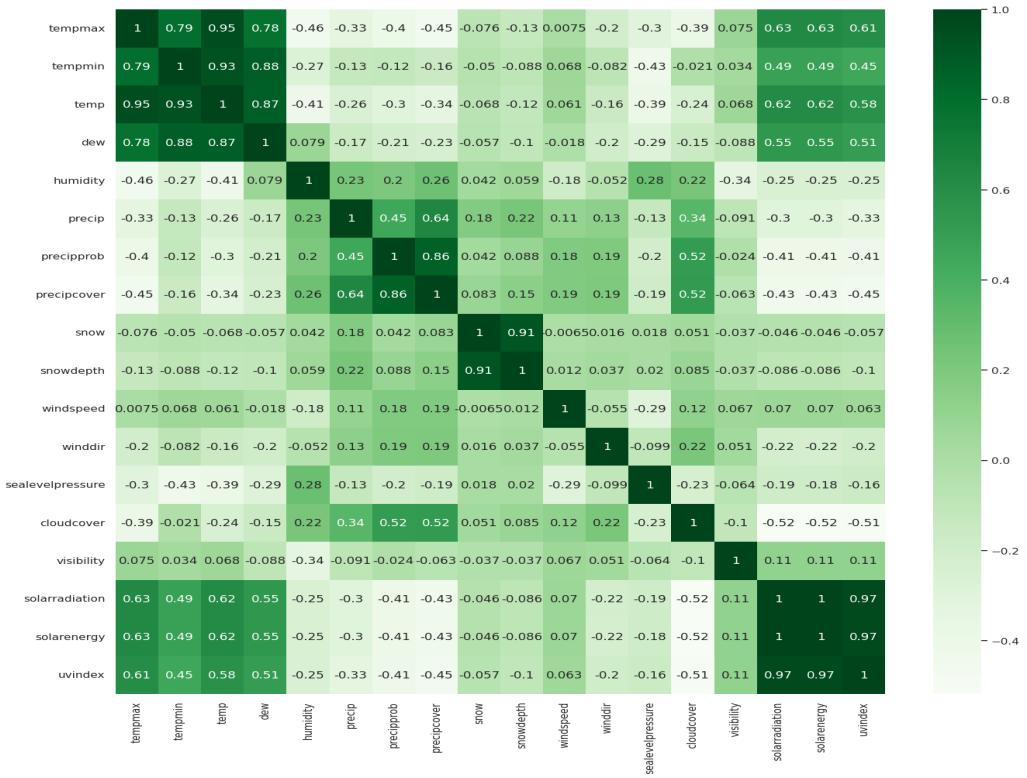

Figure 4.3: Correlation matrix

Variable	coef	std err	z	P>|z|
tempmax	0.3339	0.192	1.742	0.081
tempmin	-0.0994	0.186	-0.535	0.593
temp	-3.6968	0.615	-6.007	0.000 ***
dew	3.1468	0.457	6.887	0.000 ***
humidity	-1.9596	0.249	-7.859	0.000 ***
precip	-0.3595	0.099	-3.636	0.000 ***
precipprob	0.0556	0.084	0.666	0.506
precipcover	-0.4405	0.110	-4.008	0.000 ***
snow	-0.4377	0.151	-2.890	0.004 **
snowdepth	0.5854	0.162	3.612	0.000 ***
windspeed	0.0331	0.042	0.791	0.429
winddir	0.0059	0.040	0.147	0.883
sealevelpressure	0.0296	0.051	0.584	0.559
cloudcover	-0.0778	0.056	-1.379	0.168
visibility	-0.0783	0.040	-1.947	0.052
solarradiation	-2.8390	2.091	-1.358	0.175
solarenergy	2.6871	2.079	1.292	0.196
uvindex	0.1447	0.153	0.945	0.345

Tableau 4.2: Logistic Regression Model Output

visibility, *solarradiation*, *solarenergy*, and *uvindex* don't have statistically significant coefficients ($p > 0.05$) in predicting the risk of the disease.

Both variables, *dew* and *temp*, are important in predicting the disease risk as their coeffi-

cients are statistically significant (p-values < 0.05). However, since they are strongly related to each other, it becomes difficult to understand their separate effects.

Similarly, both *snow* and *snowdepth* are meaningful predictors with statistically significant coefficients (p-values < 0.05). Nevertheless, their strong correlation raises doubts about interpreting their individual impacts.

To handle the issue of multicollinearity, one could consider using dimension reduction techniques like principal component analysis (PCA).

PCA Process

PCA is a method for reducing the number of variables in a dataset. It takes a group of related variables and transforms them into a smaller set of new variables called principal components. The goal is to keep most of the original dataset's variability while making the new variables uncorrelated with each other. The number of principal components is fewer than the original variables.

To find the right number of PCs for our data, we first use PCA with all components set to the same number as the original dimensions (18). Then, we observe how well PCA captures the variance in our data.

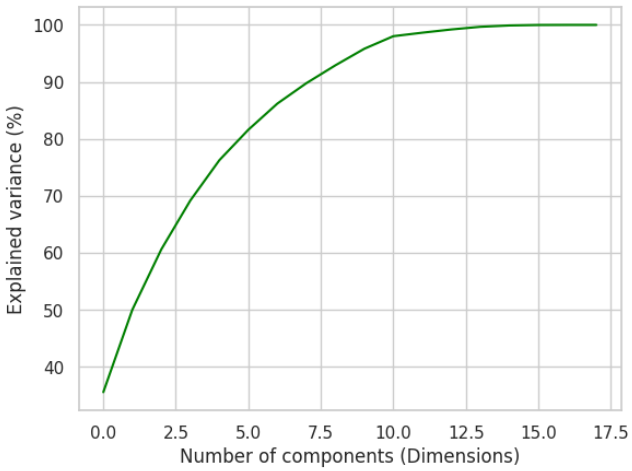

Figure 4.4: screeplot

4.4 shows that the first component explains around 35% of the data's variability, and the second component explains about 14%. As we include more components, they add more variability. The first ten components capture approximately 95.82% of the total variability in the data. So, we decide to keep these first ten components as they capture a large part of the data's variability.

PCA results

Principal Component	Eigenvalue	Explained Variance Ratio	Cumulative Variance
PC1	6.40971434	0.35600659	0.35600659
PC2	2.60178488	0.14450762	0.50051421
PC3	1.90744479	0.10594278	0.60645699
PC4	1.54197012	0.08564368	0.69210067
PC5	1.27110205	0.0705992	0.76269987
PC6	0.96192998	0.05342725	0.81612712
PC7	0.82590047	0.04587194	0.86199906
PC8	0.64434359	0.03578796	0.89778702
PC9	0.56362975	0.03130497	0.92909199
PC10	0.52408345	0.0291085	0.95820049

Tableau 4.3: Explanation of Principal Components

Representation quality of variables on the factor axes

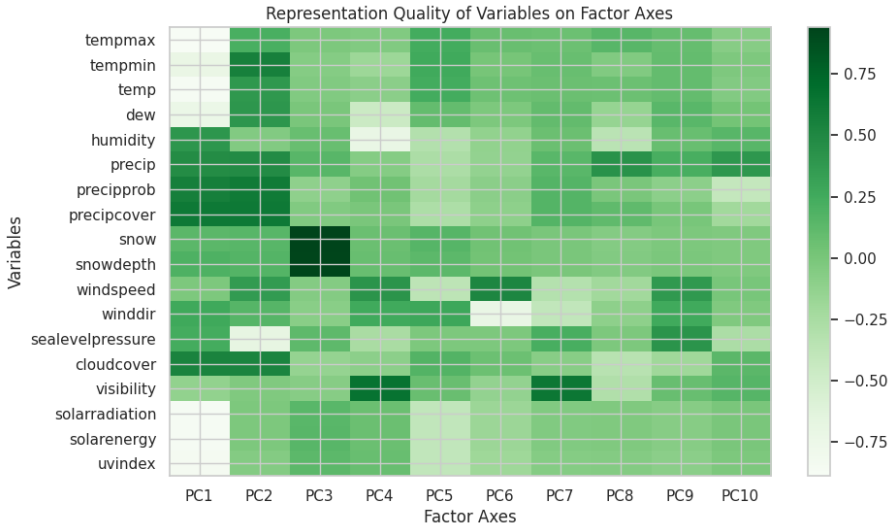

Figure 4.5: heatmap

Results interpretation

- PC1: The first main component (PC1) has strong positive connections with variables like "humidity," "precip," "precipprob," "precipcover," "cloudcover," "solarradiation," "solarenergy," and "uvindex." It also has strong negative connections with variables like "tempmax," "tempmin," "temp," "dew," "windspeed," "winddir," and "sealevelpressure." PC1 seems to combine weather-related variables, with high values meaning more humidity and cloudiness, and low values representing clear and windy conditions.

- PC2: The second main component (PC2) has positive connections with variables like "tempmax," "tempmin," "temp," "dew," "windspeed," "winddir," "sealevelpressure," "visibility," "solarradiation," "solarenergy," and "uvindex." This component might represent temperature and solar energy-related variables. High values of PC2 might mean hotter and sunnier conditions, while low values indicate cooler and less sunny conditions.

- PC3: The third main component (PC3) has strong positive connections with variables "snow" and "snowdepth." It suggests that PC3 represents snow-related variables. High values of PC3 might mean snowy conditions, while low values indicate little or no snow.
- Other PCs: The other main components have varying connections with different variables. Each main component represents a combination of variables with unique patterns and meanings.

Now, we can use this new dataset instead of the original weather parameters dataset for our analysis

4.3.1.1 Model Performance

Building a logistic regression model on the transformed data

Before proceeding to build a logistic regression model on the transformed data,let's first construct the model using the original dataset

logistic regression Base model

Accuracy: 0.96

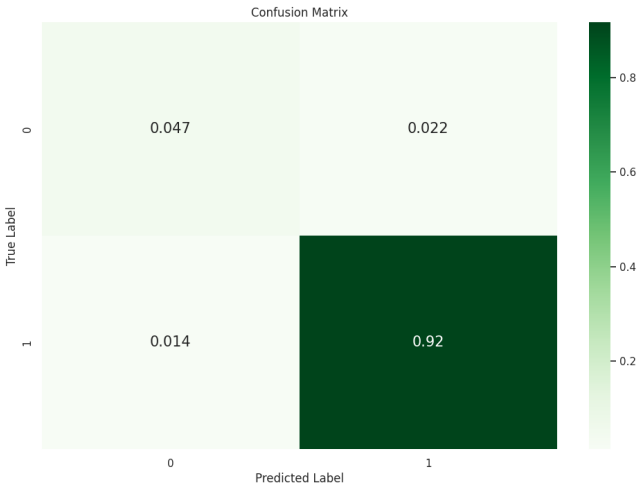

Figure 4.6: Logistic Regression Base Model Confusion matrix

The model shows good performance. It can work well with new data. Now, let's see if we can make the model better by using Principal Component Analysis (PCA).

logistic regression model on the transformed dataset

Accuracy: 97%

When we compare the current result with the previous one from our basic model, we can see that the test accuracy has gone up by 1%. the true negative have also been improved and false negatives have been reduced. However,PCA helped to deal with the problem of multicollinearity, which improved the performance of the model.

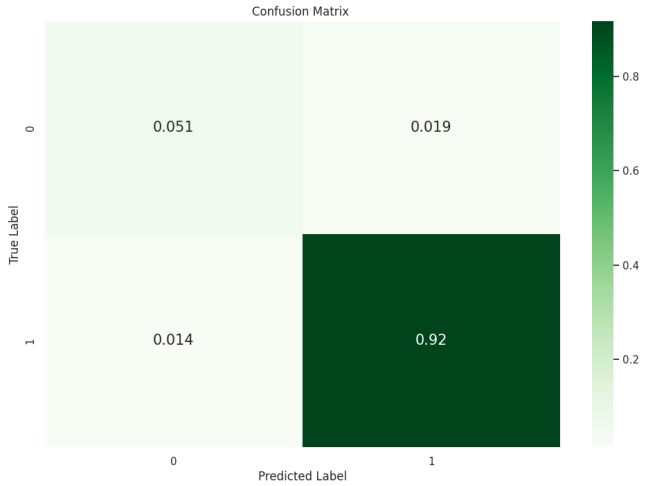

Figure 4.7: Logistic Regression-PCA model Confusion matrix

4.3.2 Artificial Neural Network (ANN)

Before we go into the details of our ANN implementation, it's important to address the issue of class imbalance in our datase.Class imbalance, where one class has significantly fewer instances (risk of disease) compared to others (no risk of disease).
the challenges that imbalanced data can pose to classification models :

- High accuracy on the majority class but low accuracy on the minority class(Biased Model).
- Poor generalization of the model to new and unseen data.

To solve these problems, we will use a technique called SMOTE (Synthetic Minority Over-sampling Technique).

4.3.2.1 SMOTE: Synthetic Minority Over-sampling Technique

SMOTE (Synthetic Minority Oversampling Technique) is a helpful method used to tackle a problem in classification where some categories have very few examples compared to others. This is called "imbalanced data."

In cases where the dataset has one category (class) with very few samples, while another category has many more, traditional methods might not work well because they tend to favor the majority class, making it challenging to correctly detect the minority class.

To fix this, SMOTE creates new data points for the minority class. Here's how it works:

- Pick a sample from the minority class randomly.
- Find its nearest neighbors in the minority class (k nearest neighbors).
- Create a new data point that is a combination of the selected sample and one of its neighbors. It's like making a new friend who is similar to you and your neighbor.
- Repeat this process for multiple samples from the minority class.

By doing this, SMOTE increases the number of minority class samples, making the data more balanced.

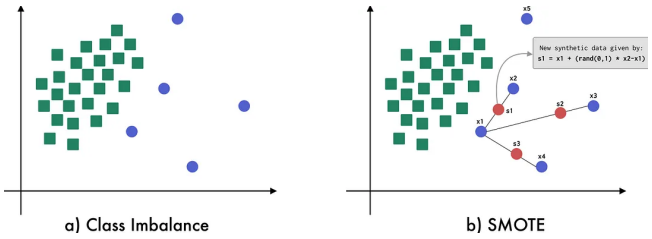

Figure 4.8: SMOTE Process

However, there's a trade-off. While SMOTE helps in identifying more of the positive cases in the minority class (recall), it may sometimes make mistakes and identify some cases incorrectly (false positives). But usually, in real-world situations, it's okay to have a few false positives compared to missing important cases (false negatives).

4.3.2.2 Architecture

For our Artificial Neural Network (ANN), we will design a feedforward architecture with three layers : an input layer, two hidden layers, and an output layer.

4.3.2.3 Hyperparameters Tuning

We worked on improving the performance of the Artificial Neural Network (ANN) model by adjusting its hyperparameters.The focuse is on three hyperparameters: batch size, number of epochs, and the optimizer.

During the hyperparameter tuning process, we carefully explored various combinations of batch sizes (5, 15, 30, 32, 50, and 100), epochs (30 and 100), and two optimizers (Adam and RMSprop), seeking the best accuracy on the validation data.

We have found that the combination {'batch_size': 15, 'epochs': 100, 'optimizer': 'adam'} showed high predictive accuracy in classifying plant diseases based on weather parameters.

4.3.2.4 Model performence

Accuracy: 95.27%

4.3.3 Support Vector Machine (SVM)

4.3.3.1 Hyperparameters Tuning

First, we performed a hyperparameter tuning process for the Support Vector Machine (SVM) model. The main goal was to find the best configuration for accurately classifying plant diseases based on weather parameters. The hyperparameters we considered were C, the kernel type, and gamma.

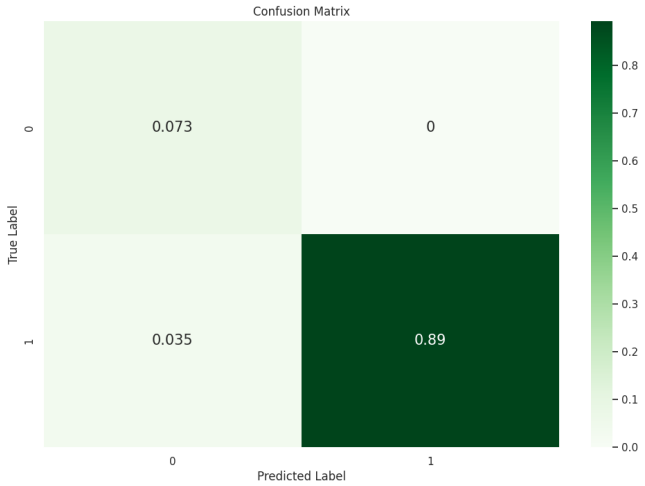

Figure 4.9: ANN Model Confusion Matrix

GridSearchCV method was employed to systematically assess different combinations of hyperparameters (C, kernel type, and gamma).This aproach helped identify the most favorable combination of hyperparameters, which resulted in the best-performing SVM model with 'C': 10, 'gamma': 1, 'kernel': 'rbf'.

4.3.3.2 Model performence

Accuracy: 95%

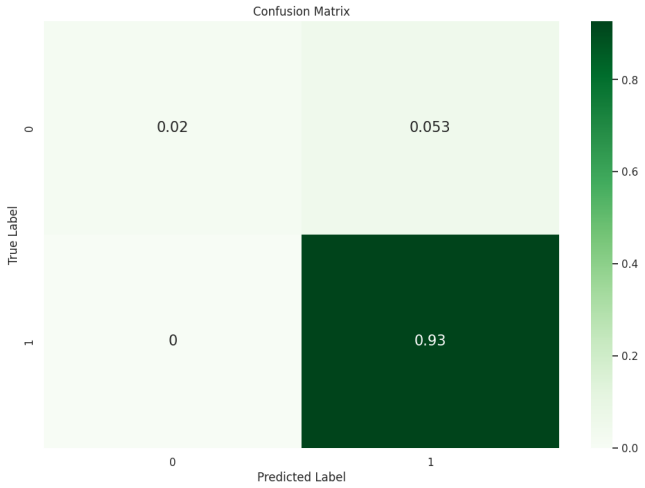

Figure 4.10: SVM Model Confusion Matrix

4.4 Model Evaluation

The table presents the evaluation results for three models: Logistic Regression + PCA, SVM, and ANN. The performance metrics include precision, recall, and F1-score for both class 0 (disease risk) and class 1 (no risk of disease), as well as overall accuracy.

Tableau 4.4: Model Evaluation Results

Model	**Precision (Class 0)**	**Recall (Class 0)**	**F1-score (Class 0)**
Logistic Regression + PCA	0.78	0.68	0.72
SVM	0.54	0.77	0.64
ANN	0.71	1.00	0.83
Model	**Precision (Class 1)**	**Recall (Class 1)**	**F1-score (Class 1)**
Logistic Regression + PCA	0.98	0.99	0.98
SVM	0.98	0.95	0.97
ANN	1.00	0.97	0.98

Model	**Accuracy**
Logistic Regression + PCA	0.96
SVM	0.95
ANN	0.97

4.4.1 Results interpretation

- The logistic regression model with PCA achieved an accuracy of 96%. It works well in identifying healthy plants (class 1) with high precision, recall, and F1-score. However, it is less accurate in identifying plants with disease risk (class 0) compared to class 1, with lower precision, recall, and F1-score.

- The SVM model demonstrates high precision (0.98) for detecting healthy plants (class 1). However, it tends to miss many plants with disease (class 0), as indicated by the low recall (0.77) for class 0 The overall accuracy of the SVM model is reported as 95

- The ANN model performs well in detecting individuals with disease risk (class 0) when they are present, but it sometimes misclassifies healthy individuals. There is a trade-off between precision and recall for class 0. For class 1 (no risk of disease), the model has high precision and good recall, leading to a high F1-score. The overall accuracy of the ANN model is 97%.

4.4.2 Model Selection

In this study, we evaluated three different models: Logistic Regression + PCA, Support Vector Machine (SVM), and Artificial Neural Network (ANN).

The ANN model was chosen because it achieved the highest accuracy of 97%, which was better than the other models. It was good at correctly identifying both individuals with disease risk and those with no risk of disease, as shown by its balanced precision and recall for both classes. The ANN model was also able to learn complex patterns from the data, making it suitable for this task. Its high F1-score of 0.98 for class 1 further confirmed its ability to classify healthy individuals accurately. Overall, the ANN model performed the best among the three models, which is why it was selected.

4.5 CNN Architectures for Disease Identification

4.5.1 Preprocessing and Augmentation of Plant Images

Before feeding the images into the deep learning model

- we normalized it to values between 0 and 1 by dividing each pixel value by 225.
- we resizes it to the default image size of (250, 250) pixels.
- The dataset we used for training was split into 80% for training and 20% for validation.

The data augmentation techniques used are as follows:

- **Rotation**: Randomly rotates the images by a maximum of 25 degrees in either direction.
- **Width and Height Shift**: Randomly shifts the images horizontally and vertically by a maximum of 10% of their width or height.
- **Shear**: Randomly applies shearing transformations to the images by a maximum of 0.2 radians.
- **Zoom**: Randomly zooms into or out of the images by a maximum of 20%.
- **Horizontal Flip**: Allows for random horizontal flipping of the images to add diversity to the training data.
- **Fill Mode**: Specifies the strategy used to fill in pixels created by the transformations, using the "nearest" fill mode which duplicates edge pixels to fill empty spaces.

By applying these techniques to the training data,we generates new variations of the original images, effectively increasing the diversity of the training set and improving the model's ability to generalize to unseen data.

4.5.2 Model Training and Validation

we have chosen the following hyperparameters to train our models :

- **Batch size:** 32
- **Epochs:** 25

- **Learning rate:** 0.01
- **Optimizer:** Adam
- **Loss:** categorical cross-entropy (it is commonly used for multi-class classification)

Here are the validation results for the two models: VGG16 and MobileNetV1, including the final validation loss and validation accuracy, are shown in Table 4.5 below

Model	Validation Loss	Validation Accuracy
VGG16	0.5650	0.8477
MobileNetV1	0.2691	0.9171

Tableau 4.5: Validation Results

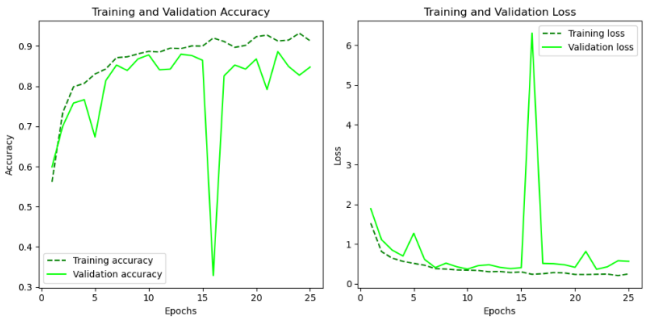

Figure 4.11: VGG16 model training and validation loss and accuracy

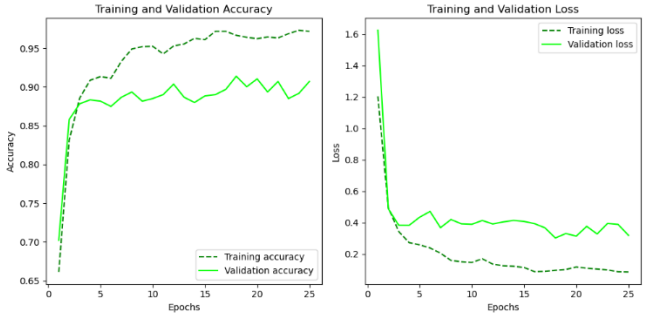

Figure 4.12: MobileNetV1 model training and validation loss and accuracy

4.5.2.1 Results Interpretation

- MobileNetV1 achieved a validation accuracy of 0.9069 after 25 epochs, while VGG16 achieved a validation accuracy of 0.8477 after 25 epochs. This indicates that MobileNetV1 is slightly more accurate than VGG16 on the validation set.
- The MobileNetV1 model has a lower validation loss than VGG16 at all epochs. This suggests that MobileNetV1 is generalizing better to the unseen validation data.
- The MobileNetV1 model's loss decreased more rapidly than the VGG16 model's loss.
- The MobileNetV1 model's validation accuracy was more stable than the VGG16 model's validation accuracy.

Class	Precision	Recall	F1-score
Pepper_bell_Bacterial_spot	1.00	0.87	0.93
Pepper_bell_healthy	0.85	1.00	0.92
Potato_Early_blight	0.95	0.95	0.95
Potato_Late_blight	0.90	0.97	0.93
Potato_healthy	0.97	0.94	0.95
Tomato_Bacterial_spot	0.83	0.91	0.87
Tomato_Early_blight	0.87	0.77	0.82
Tomato_Late_blight	0.88	0.90	0.89
Tomato_Leaf_Mold	0.98	0.88	0.92
Tomato_Septoria_leaf_spot	0.90	0.70	0.79
Tomato_Spider_mites_Two_spotted_spider_mite	0.81	1.00	0.90
Tomato_Target_Spot	0.88	0.79	0.83
Tomato_Tomato_YellowLeaf_Curl_Virus	0.91	0.97	0.94
Tomato_Tomato_mosaic_virus	0.97	1.00	0.98
Tomato_healthy	0.98	0.98	0.98

Tableau 4.6: MobileNetV1 Model Classification Report

4.5.3 Models Evaluation and Classification Report

In this section, we will evaluate the performance of two models, VGG16 and MobileNetV1, for plant leaf disease classification.

4.5.3.1 Results Interpretation

- Both models seems to perform well, with most classes having a high F1-score. However, there are a few classes with lower F1-scores, such as Tomato_Septoria_leaf_spot and Tomato_Spider_mites_Two_spotted_spider_mite indicating that it may not be as accurate at classifying these classes
- Both models have a high precision for most classes, indicating that it rarely makes false positives. A false positive is when the model predicts that a plant has a disease when it does not actually have the disease. A high precision means that the model is very good at avoiding false positives.
- Both models have a high recall for most classes, indicating that it rarely makes false negatives.A false negative is when the model predicts that a plant does not have a disease when it actually does have the disease. A high recall means that the model is very good at avoiding false negatives.

4.5.4 Model Selection

Based on the results ,we would select MobileNetV1 as the better model for plant leaf disease classification.

Here are the reasons why:

Class	Precision	Recall	F1-score
Pepper_bell_Bacterial_spot	0.95	0.89	0.92
Pepper_bell_healthy	0.86	0.97	0.91
Potato_Early_blight	1.00	0.92	0.96
Potato_Late_blight	0.87	0.92	0.89
Potato_healthy	0.97	0.94	0.95
Tomato_Bacterial_spot	0.90	0.86	0.88
Tomato_Early_blight	0.68	0.54	0.60
Tomato_Late_blight	0.73	0.90	0.81
Tomato_Leaf_Mold	0.91	0.88	0.89
Tomato_Septoria_leaf_spot	0.84	0.73	0.78
Tomato_Spider_mites_Two_spotted_spider_mite	0.84	0.98	0.90
Tomato_Target_Spot	0.74	0.82	0.78
Tomato_Tomato_YellowLeaf_Curl_Virus	1.00	0.91	0.95
Tomato_Tomato_mosaic_virus	0.86	1.00	0.93
Tomato_healthy	1.00	0.78	0.88

Tableau 4.7: VGG16 Model Classification Report

- MobileNetV1 has a slightly higher validation accuracy than VGG16.
- MobileNetV1 has a lower validation loss than VGG16 at all epochs. This suggests that MobileNetV1 is generalizing better to the unseen validation data.
- MobileNetV1's loss decreased more rapidly than the VGG16 model's loss.
- MobileNetV1's validation accuracy was more stable than the VGG16 model's validation accuracy.

In addition, the classification reports for both models show that MobileNetV1 has a higher F1-score for most classes. This means that MobileNetV1 is more accurate at classifying these classes.

4.6 Conclusion

In conclusion, we compared different models to predict plant disease risk based on weather factors. We used logistic regression with PCA, artificial neural networks (ANN), and support vector machines (SVM).

By using PCA, we addressed multicollinearity, improving the models' performance. Both ANN and SVM models accurately classified disease risk. This comparison provided insights into the suitability and effectiveness of these models for predicting plant diseases.

Furthermore, we explored deep learning models VGG16 and MobileNetV1 for automating plant leaf disease identification.

General Conclution

In this thesis, we explored the application of machine learning and Deep Learning techniques for plant disease prediction and identification using both structured and unstructured data. We investigated two different approaches: structured data-based prediction and unstructured data-based identification.

For structured data-based prediction, We explored various models, including logistic regression with PCA, artificial neural networks (ANN), and support vector machines (SVM) for structured data analysis. The application of PCA helped improve model performance by addressing multicollinearity. Both ANN and SVM demonstrated high accuracy in classifying disease risk.

On the other hand, for unstructured data-based identification, we focused on image processing, computer vision techniques, and CNN architectures. Two CNN models, VGGNet and MobileNet, were applied to identify plant diseases from images. The models were trained and evaluated using an PlantVillage dataset, and we analyzed their performance using classification reports. The results demonstrated the effectiveness of CNNs in accurately classifying plant diseases from images.

Overall, this research showcases the potential of machine learning and Deep Learning in revolutionizing plant disease prediction and identification.

However, there are still opportunities for improvement and further research. For structured data-based prediction, exploring more advanced classification models and feature engineering techniques could enhance the accuracy of disease prediction. For unstructured data-based identification, experimenting with different CNN architectures and data augmentation techniques might lead to even better results.

Bibliography

[1] Katja Wiedner and Bruno Glaser. *Biochar-Fungi Interactions in Soils*. 02 2013.

[2] Kirill Kaplan and Valeriy Kopylov. *Hands-On Mathematics for Deep Learning*. Packt Publishing, Birmingham, UK, 2020.

[3] Analytics Vidhya. Gradient descent vs backpropagation: What's the difference? https://www.analyticsvidhya.com/blog/2023/01/gradient-descent-vs-backpropagation-whats-the-difference/, January 2023. Accessed on June 18, 2023.

[4] Cheonshik Kim, Dongkyoo Shin, Ching-Nung Yang, and Lu Leng. Data hiding method for color ambtc compressed images using color difference. *Applied Sciences*, 11:3418, 04 2021.

[5] ML Notebook. Convolutional neural networks - part 1, 2017.

[6] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014.

[7] Bulent Tugrul, Elhoucine Elfatimi, and Recep Eryigit. Convolutional neural networks in detection of plant leaf diseases: A review. *Agriculture*, 2022.

[8] Wei Wang, Jinge Tian, Chengwen Zhang, Yanhong Luo, Xin Wang, and Ji Li. An improved deep learning approach and its applications on colonic polyp images detection. *BMC Medical Imaging*, 20, 07 2020.

[9] Baojin Han, Min Hu, Xiaohua Wang, and Fuji Ren. A triple-structure network model based upon mobilenet v1 and multi-loss function for facial expression recognition. *Symmetry*, 14:2055, 10 2022.

[10] Joaquin Guillermo Gil, Gabriel Giraldo Martínez, and Juan Osorio. Design of electronic devices for monitoring climatic variables and development of an early warning system for the avocado wilt complex disease. *Computers and Electronics in Agriculture*, 153:134–143, 10 2018.

[11] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. http://www.deeplearningbook.org.

[12] Benjamin Lee, Anthony Gitter, Casey Greene, Sebastian Raschka, Finlay Maguire, Alexander Titus, Michael Kessler, Alexandra Lee, Marc Chevrette, Paul Stewart, Thiago Britto-Borges, Evan Cofer, Kun-Hsing Yu, Juan Carmona, Elana Fertig, Alexandr Kalinin, Brandon Signal, Benjamin Lengerich, Timothy Triche, and Simina Boca. Ten quick tips for deep learning in biology. *PLOS Computational Biology*, 18:e1009803, 03 2022.

[13] Giorgio Roffo. Ranking to learn and learning to rank: On the role of ranking in pattern recognition applications. 06 2017.

[14] Harvard-IACS. Cs109a lecture 32. https://harvard-iacs.github.io/2020-CS109A/lectures/lecture32/notebook-2/, 2020. Accessed on June 18, 2023.

[15] Saheba Bhatnagar, Laurence Gill, and Bidisha Ghosh. Drone image segmentation using machine and deep learning for mapping raised bog vegetation communities. *Remote Sensing*, 12:2602, 08 2020.

[16] Nathaniel Newlands. Model-based forecasting of agricultural crop disease risk at the regional scale, integrating airborne inoculum, environmental, and satellite-based monitoring data. *Frontiers in Environmental Science*, 6, 06 2018.

[17] Sharada Mohanty, David Hughes, and Marcel Salathe. Using deep learning for image-based plant disease detection. *Frontiers in Plant Science*, 7, 04 2016.

[18] Sebastian Raschka. Principle component analysis in three simple steps, 2015.

[19] Sebastian Raschka. Python lda: Topic modeling and document clustering with scikit-learn, 2014.

[20] Authors. Pathogens, precipitation and produce prices. *Nat. Clim. Chang.*, 11:635, 2021.

[21] Jun Liu and Xuewei Wang. Plant diseases and pests detection based on deep learning: a review. *Plant Methods*, 17, 02 2021.

[22] Imtiaz Ahmed and Pramod Yadav. Plant disease detection using machine learning approaches. *Expert Systems*, 40, 10 2022.

[23] Mrunalini Badnakhe, Surya Durbha, Adinarayana Jagarlapudi, and R. Gade. Evaluation of citrus gummosis disease dynamics and predictions with weather and inversion based leaf optical model. *Computers and Electronics in Agriculture*, 155, 12 2018.

[24] Sanjeev Sannakki, V. Rajpurohit, F. Sumira, and Venkatesh Hosahalli. A neural network approach for disease forecasting in grapes using weather parameters. pages 1–5, 07 2013.

[25] Carlos Cabrera, Leonardo Ramirez, and Flavio Ortiz. Spectral analysis for the early detection of anthracnose in fruits of sugar mango (mangifera indica). *Computers and Electronics in Agriculture*, 173:105357, 06 2020.

[26] Parisa Ahmadi, Farrah Melissa Muharam, Khairulmazmi Ahmad, Shattri Mansor, and Idris Abu Seman. Early detection of ganoderma basal stem rot of oil palms using artificial neural network spectral analysis. *Plant Disease*, 101:PDIS–12, 04 2017.

[27] Gianni Fenu and Francesca Malloci. *Artificial Intelligence Technique in Crop Disease Forecasting: A Case Study on Potato Late Blight Prediction*, pages 79–89. 06 2020.

[28] Gianni Fenu and Francesca Malloci. An application of machine learning technique in forecasting crop disease. pages 76–82, 11 2019.

[29] Jean Ristaino, Pamela Anderson, Daniel Bebber, Kate Brauman, Nik Cunniffe, Nina Fedoroff, Cambria Finegold, Karen Garrett, Christopher Gilligan, Christopher Jones, Michael Martin, Graham MacDonald, Patricia Neenan, Angela Records, David Schmale, Laura Tateosian, and Qingshan Wei. The persistent threat of emerging plant disease pandemics to global food security. *Proceedings of the National Academy of Sciences*, 118:e2022239118, 2021.

[30] Michael Jeger, Robert Beresford, Clive Bock, Nathan Brown, Adrian Fox, Adrian Newton, Antonio Vicent, Xiangming Xu, and Jonathan Yuen. Global challenges facing plant pathology: multidisciplinary approaches to meet the food security and environmental challenges in the mid-twenty-first century. *CABI Agriculture and Bioscience*, 2, 12 2021.

[31] Hannah Dee and Andrew French. From image processing to computer vision: Plant imaging grows up. *Functional Plant Biology*, 42:iii, 05 2015.

[32] Chongke Bi, Jiamin Wang, Yulin Duan, Baofeng Fu, and Jia-Rong Kang. Mobilenet based apple leaf diseases identification. *Mobile Networks and Applications*, 27, 02 2022.

[33] S. Ashwinkumar, S. Rajagopal, V. Manimaran, and B. Jegajothi. Automated plant leaf disease detection and classification using optimal mobilenet based convolutional neural networks. *Materials Today: Proceedings*, 51, 07 2021.

[34] Junde Chen, Defu Zhang, Md Suzauddola, and Adnan Zeb. Identifying crop diseases using attention embedded mobilenet-v2 model. *Applied Soft Computing*, 113:107901, 09 2021.

[35] Modepalli Kavitha. Proportional learning on noise removal methods. *Imperial journal of interdisciplinary research*, 3, 2017.

[36] Binny Thakral, Tausif Ahmad, Mandeep Kaur, Ishteaque Ahmad, and Sonia. A review of various image enhancement and noise removal methods. 2016.

[37] Eunkyeong Kim, Hyunhak Cho, Hansoo Lee, Jongeun Park, and Sungshin Kim. Image brightness adjustment system based on anfis by rgb and cie lab. *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 471–476, 2017.

[38] Zeshan Hussain, Francisco Gimenez, Darvin Yi, and Daniel Rubin. Differential data augmentation techniques for medical imaging classification tasks. *AMIA ... Annual Symposium proceedings. AMIA Symposium*, 2017:979–984, 04 2018.

[39] Pornntiwa Pawara, Emmanuel Okafor, Lambert Schomaker, and Marco Wiering. Data augmentation for plant classification. 09 2017.

[40] Maayan Frid-Adar, Idit Diamant, Eyal Klang, Marianne Amitai, Jacob Goldberger, and Heather Greenspan. Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification. *Neurocomputing*, 321, 03 2018.

[41] Sarah Ismael, Ammar Mohammed, and Hesham Hefny. An enhanced deep learning approach for brain cancer mri images classification using residual networks. *Artificial Intelligence in Medicine*, 102:101779, 12 2019.

Appendix A

Neural Network Architecture and Tomato Late Blight Model

A.1 Neural Network Architecture

Algorithm 1 Neural Network Architecture

1: Initialize the neural network classifier
2: Add a fully connected layer with 18 units, ReLU activation, and uniform weight initialization
3: Add a dropout layer with a dropout rate of 0.2
4: Add another fully connected layer with 6 units and ReLU activation
5: Add the output layer with 1 unit and sigmoid activation for binary classification
6: Compile the classifier using the Adam optimizer and binary cross-entropy loss

A.2 Tomato Late Blight model Architecture

Algorithm 2 Tomato late blight model pseudo code

Data : Dataframe containing temperature (T), minimum temperature (Tmin), relative humidity (RH), and precipitation (R)
Result : Tomato Disease Risk Index (DRI)
Function `calculate_TIndex(`*T, Tmin*`):`
  Fc $\leftarrow$ 0.35 + 0.05 $\times$ Tmin TIndex $\leftarrow$ (-2.19247 + 0.259906 $\times$ T - 0.000139 $\times$ T^3 - 6.095832 $\times$ 10^{-6} $\times$ T^4) $\times$ Fc **return** *TIndex*
Function `calculate_RHIndex(`*RH*`):`
  RHIndex $\leftarrow$ -34.9972725 + 0.751 $\times$ RH - 0.003909 $\times$ RH^2 **return** *RHIndex*
Function `calculate_RIndex(`*R*`):`
  Rindex $\leftarrow$ 0.006667 + 0.194405 $\times$ R + 0.0002239 $\times$ R^2 **return** *Rindex*
Function `calculate_IPI(`*T, Tmin, RH, R*`):`
  if *Tmin* $\leq$ *7 or not (9* $\leq$ *T* $\leq$ *25) or RH* $\leq$ *80 or R* $\leq$ *0.2* **then**
    return *None* `// IPI cannot be calculated`
  end
  Tindex $\leftarrow$ `calculate_TIndex(`*T, Tmin*`)` RHindex $\leftarrow$ `calculate_RHIndex(`*RH*`)` Rindex $\leftarrow$ `calculate_RIndex(`*R*`)` **if** *RHindex* $>$ *Rindex* **then**
    IPI $\leftarrow$ Tindex $\times$ RHindex
  end
  else
    IPI $\leftarrow$ Tindex $\times$ Rindex
  end
  return *IPI*
Function `Tomato_DRI(`*IPI*`):`
  if *IPI is not None* **then**
    if *IPI* $<$ *15* **then**
      return *0* `// High risk of disease`
    end
    else
      return *1* `// No risk of disease`
    end
  end
  return *None*

Appendix B

Generalities about Plant Diseases

B.1 Plant Pathology

Plant pathology, or phytopathology, is the study of plant diseases caused by pathogens and environmental conditions. These pathogens can be fungi, oomycetes, bacteria, viruses, viroids, virus-like organisms, phytoplasmas, protozoa, nematodes, or parasitic plants

B.2 Plant Diseases

Plant diseases are a result of the continuous disturbance of a plant by some causal agent that disrupts its normal structure, growth, function, or other activities, resulting in an abnormal physiological process

Plant diseases can be classified into two categories: abiotic or non-infectious diseases and biotic or infectious diseases

Category	**Causes**	**Spread**	**Examples**
Abiotic diseases	External conditions (e.g., nutritional deficiencies, soil compaction, salt injury, ice, sun scorch)	Cannot spread from plant to plant	Nutritional deficiency symptoms, sunburned leaves
Biotic diseases	Living organisms (plant pathogens)	Can spread from plant to plant	Fungal leaf spots, bacterial wilts, viral infections

Tableau B.1: Plant Diseases Types

B.3 Tomato Late Blight

Late blight is a severe plant disease that affects tomatoes and other Solanum spp., caused by the fungus Phytophthora infestans. It leads to water-soaked lesions on leaves, stems, and

fruits, which turn brown, resembling frost or fire damage. The disease spreads through infected transplants, volunteer plants, and related weeds.

B.4 Importance of Early Detection

Early detection of tomato late blight is essential for the following reasons:

1. **Preventing disease spread:** Detecting late blight early stops its transmission to other plants through wind, rain, or human activity.
2. **Minimizing crop losses:** Late blight can severely damage tomato plants, leading to total crop failure if not addressed promptly. Early detection allows for timely action to manage the disease and minimize losses.
3. **Effective management strategies:** Early detection enables the implementation of strategies like removing infected plants, using fungicides, and improving plant spacing and airflow, which are effective in controlling the disease.
4. **Saving time and money:** Early detection conserves resources by reducing the need for expensive fungicides and avoiding replanting due to crop loss.