PLANT DISEASE DETECTION USING MACHINE LEARNING

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Vaibhav Sharma (191545)

Under the supervision of
Dr. Ruchi Verma
to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology
Waknaghat, Solan- 173234, Himachal Pradesh

CERTIFICATE

I hereby declare that the work presented in this report entitled **" Plant Disease Detection using Machine Learning"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from July 2022 to May 2023 under the supervision of **Dr. Ruchi Verma, Associate Professor(SG), Dept. CSE&IT**.

I also authenticate that I have carried out the above mentioned project work under the proficiency stream Machine Learning.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)
Vaibhav Sharma(191545)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)
Dr. Ruchi Verma
Associate Professor (SG)
Department of CSE & IT

PLAGIARISM REPORT

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: .....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: _____ **Department:** _____ **Enrolment No** _____

Contact No. _____ **E-mail.** _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at .....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	**Excluded**	**Similarity Index (%)**	**Generated Plagiarism Report Details** (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature **Librarian**

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

At the onset, I express my heartfelt thanks and gratefulness to God for his pure and divine blessing that makes it possible for us to complete the project work successfully within the right time. I am really humbled to do this endeavor project under my respected professor and I wish my profound indebtedness to Supervisor **Dr. Ruchi Verma**, **Assistant Professor**, Department of CSE & IT, Jaypee University of Information Technology (JUIT), Waknaghat. She has deep Knowledge of this project related stuff & her keen interest in guiding us in the field of "**Machine Learning**" to carry out this project. Her endless patience, scholarly guidance, perennial encouragement, constant and energetic supervision, and valuable advice pertaining to many pre-published drafts have made it possible for me to complete this project.

I would like to express my deep gratitude from the bottom of my heart to **Dr. Ruchi Verma**, Department of CSE, for her generous help to finish my project.

I would also acknowledge each one of those individuals who have helped me directly or indirectly in making this project a possibility. In this juncture, I would like to thank the staff fraternity, individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Vaibhav Sharma(191545)

TABLE OF CONTENT

LIST OF FIGURES

ABSTRACT

Plant disease detection is a cutting-edge and enlightening system that helps users learn about diseases, training, and other fascinating events happening in their local area. This organization helps the local population stay informed about activities in and around their town, region, or locale. This approach requires both machine learning and image processing in order to function. The accuracy of the results has been improved by using contemporary methods like machine learning and deep learning algorithms. As a whole, random forests are a learning technique for problems like classification, regression, and others that work by building a forest of decision trees during the training period.

A component descriptor used in computer vision and image processing for object detection is the histogram of oriented gradients (HOG). In this case, we are using three component descriptors:

1. Hu moments
2. Haralick texture
3. Colour Histogram

Keywords: HOG, Object recognition, Random Forests, Image processing, Histogram graph

Chapter-1 INTRODUCTION

1.1 Introduction

Plant disease detection is a cutting-edge and enlightening system that helps users learn about diseases, trainings, and other fascinating events happening nearby. This organisation helps the local population stay informed about activities in and around their town, region, or locale. This approach requires both machine learning and image processing in order to function. The user is only allowed to add diseases related to his town, albeit they can be added for any town. When a user adds an unsuitable, fictitious, or misused ailment, the administrator will display it and take appropriate action. Only 500 words are available for the user to lecture on diseases. The system gives a swipe to advance to the next or earlier disease with transition features, and the appearance and texture of the disease evaluation are exciting and outstanding.Farmers in rural areas might believe it is difficult to distinguish the diseases that might affect their harvests. They cannot easily visit the agricultural office to find out what the infection might be. Our main goal is to identify the disease that is introduced in a plant by observing its shape using image processing and machine learning.

Pests and diseases harm crops or plant parts, reducing food production and escalating food poverty. Toxic infections, inadequate disease management, and significant climatic change are some of the major contributors to decreasing food production. Numerous innovative technologies have emerged in order to decrease post-harvest processing, improve agricultural sustainability, and boost production. Numerous laboratory-based techniques, such as polymerase chain reaction, gas chromatography, mass spectrometry, thermography, and hyperspectral approaches, have been employed to identify illnesses. These techniques are time-consuming and inefficient from a financial standpoint. Recently, server-based and mobile-based methods have been used to identify diseases. The high resolution camera, among other

things, makes it possible to automatically identify diseases.Using modern techniques like machine learning and deep learning algorithms has increased the accuracy of the outcomes. Numerous research have been carried out utilising traditional machine learning techniques, including random forests, artificial neural networks, support vector machines (SVM), fuzzy logic, K-means method, and convolution neural networks, among others, for the detection and diagnosis of plant illnesses. In general, random forests are a learning method that develops a forest of decision trees during the training phase and applies it to problems like classification, regression, and others.

A component descriptor used in computer vision and image processing for object detection is the histogram of oriented gradients (HOG). In this case, we are using three component descriptors:

1. Hu moments
2. Haralick texture
3. Color Histogram Hu moments is basically used to extract the shape of the leaves.

To obtain the colour and texture of the leaves, haralick texture is utilised. A histogram is a graph that shows how the colours in an image are distributed. Farmers in rural areas might believe it is difficult to distinguish the diseases that might affect their harvests. They cannot easily visit the agricultural office to find out what the infection might be. Our main goal is to identify the disease that is introduced in a plant by observing its morphology, processing its images, and using machine learning. Pests and diseases cause crops or parts of plants to be destroyed, which lowers food output and increases food insecurity. Various less developed nations also have a lower level of expertise on the prevention and control of illnesses and pests.

In the recent past, disease identification has been done using server-based and mobile-based approaches. Automatic disease recognition is made possible by a

number of elements, including the high resolution camera, high performance processing, and numerous built-in accessories. A portable system for detecting plant diseases operates in an automated environment built with the Java language. The app will operate quickly and smoothly. Because Google Material Design is used, apps look more elegant and beautiful and provide a positive user experience. It manages news, different categories, notifications, and many other things whenever we want when admin and user are combined.

Good design and good programming are priorities. Using this app allows us to save time and money while also allowing us to construct our own unique or distinct types of apps based on our needs. Plant disease detection is a cutting-edge and enlightening system that helps users learn about diseases, trainings, and other fascinating events happening nearby. The local community is assisted by this organization in staying informed about events taking place in and around their town, area, or location. In order for this method to work, both machine learning and image processing are necessary. The user is only allowed to view diseases related to his town, though they can be added for any town. When a user adds an unsuitable, fictitious, or misused ailment, the administrator will display it and take appropriate action. Android Studio is the front end, and SQL Server is the back end. To use this app, the user must register with the system and may also update his information.

1.2 Problem Statement

To create a trustworthy and accurate system that can automatically detect and diagnose plant illnesses in a quick and timely manner is the problem statement for plant disease detection using machine learning. This system should be able

to generalise to new locations and plant species while processing massive amounts of data rapidly and accurately.

The present methods for identifying plant diseases, such visual inspection or laboratory analysis, take time, are subjective, and frequently need specialised knowledge. Furthermore, these techniques might not be able to identify diseases in the earliest stages, when treatments are most efficient.

The potential for machine learning-based methods to get over these restrictions and offer an automated, objective solution for plant disease diagnosis. However, the creation of such systems necessitates the accessibility of sizable and varied datasets, the choice of suitable machine learning algorithms, and the hyperparameter optimisation for the particular issue at hand.

1.3 Objectives

1) Early Detection: Preventing the spread of plant diseases requires early detection. By assisting in the early detection of plant illnesses, machine learning algorithms enable farmers to respond quickly to limit additional harm.

2) Accuracy: Based on the examination of sizable datasets of plant photos, machine learning algorithms can precisely diagnose and categorize many forms of plant diseases. This precision can decrease the possibility of a wrong diagnosis and improve the management of plant diseases.

3) Efficiency: By automating the process of finding plant diseases, machine learning algorithms may assist increase efficiency by saving time and effort. This strategy can help farmers and experts in plant diseases swiftly monitor broad regions of crops and spot possible outbreaks.

4) Cost-effectiveness: Developing cost-effective detection methods is important for widespread adoption, especially in resource-limited agricultural systems. The objective is to create affordable and accessible technologies that can be utilized by farmers and agricultural practitioners.

1.4 Methodology

The following models have been used and trained for the detection of Plant Diseases and the datasets have been taken from Kaggle.com, then the models with the highest accuracy are selected and they undergo an ensemble model.

1) **VGG-16 Model**- The Visual Geometry Group (VGG) at the University of Oxford created the convolutional neural network (CNN) model known as the VGG16 in 2014. The moniker VGG16 refers to the deep learning architecture's 16 layers.Multiple convolutional layers with tiny 3x3 filters make up the VGG16 model.
2) **DenseNet201**-Researchers at Facebook AI Research presented the DenseNet201 model, a deep convolutional neural network (CNN) architecture, in 2017. It is a variation on the DenseNet (Dense Convolutional Network) model, which was created to deal with the issue of vanishing gradients in extremely deep neural networks.The DenseNet201 model's architecture is made up of a number of dense blocks, which are collections of convolutional layers with a same number of output channels.
3) **InceptionV3**- Researchers at Google introduced the InceptionV3 model, a deep convolutional neural network (CNN) architecture, in 2015. It is an adaptation of the Inception model, which was created to solve the issue of computing effectiveness in extremely deep neural networks. The 48 layers of the InceptionV3 model are organised into many Inception modules, which are collections of convolutional layers with various filter sizes.

Ensemble learning is a machine learning technique that involves combining multiple models or learners to make predictions or solve complex problems. The idea behind ensemble learning is that by combining the predictions of multiple models, the overall performance and accuracy can be improved compared to using a single model. The individual models in an ensemble are often referred to as "base models" or "weak learners." These base models can be of any type, such as decision

trees, neural networks, support vector machines, or any other machine learning algorithm. Each base model in the ensemble is trained
independently on a subset of the training data or using a different algorithm variation to introduce diversity.

There are several common techniques for combining the predictions of the base models in an ensemble. The two most popular methods are:

1) **Voting**: In voting-based ensembles, each base model independently predicts the target variable, and the final prediction is made by aggregating the individual predictions. This can be done through majority voting (where the most frequent prediction is selected) or weighted voting (where each model's prediction is given a specific weight).

2) **Averaging**: Averaging-based ensembles compute the average or weighted average of the predictions made by the base models. This approach is commonly used when dealing with regression problems, where the final prediction is a continuous value.

Ensemble learning offers several advantages over using a single model. It helps in reducing overfitting by introducing model diversity and capturing different aspects of the data. Ensemble models tend to have better generalization capabilities and can handle complex patterns in the data. They are also more robust to noise and outliers in the training data.

Some popular ensemble learning algorithms include Random Forest, AdaBoost, Gradient Boosting, and Bagging. Each algorithm employs a specific strategy to train and combine the base models effectively.

Overall, ensemble learning is a powerful technique that can significantly improve the performance and accuracy of machine learning models by leveraging the collective knowledge of multiple models.

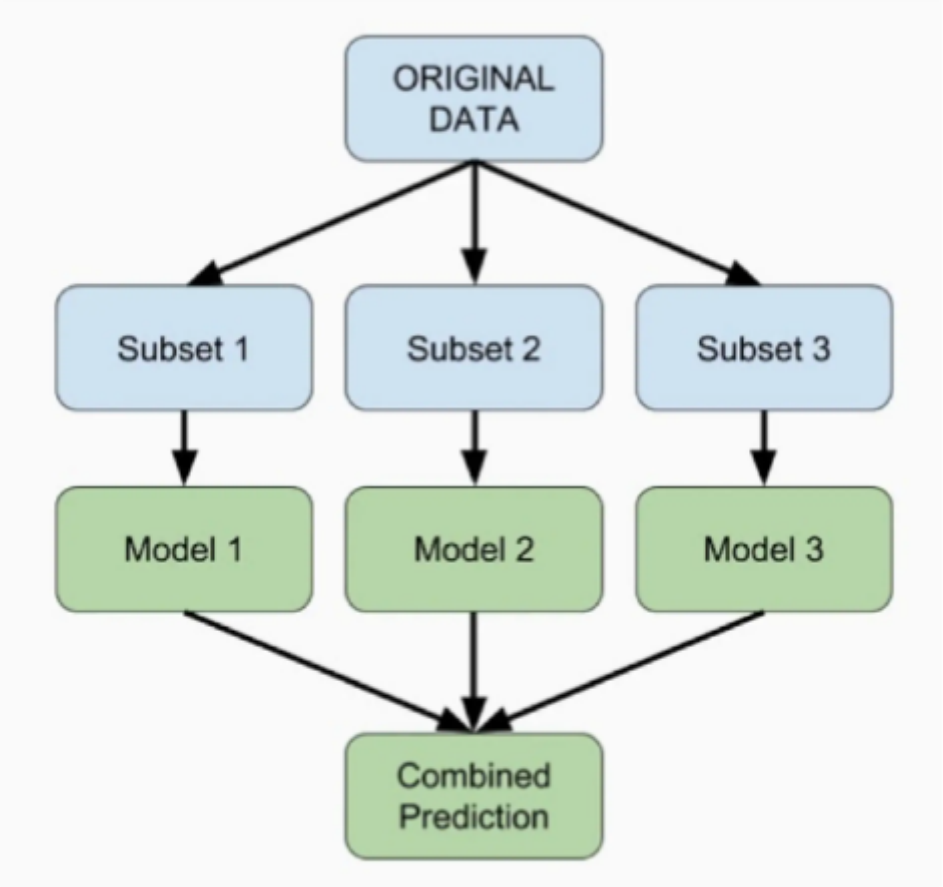


Fig. Ensemble Learning through a flowchart

Ensemble methods can help reduce the bias introduced by individual models. If a base model is biased in some way, combining it with other models that have different biases can lead to a more balanced and unbiased overall prediction.

The dataset used is the Banana Leaf Dataset from Kaggle.com and it has 2000 images across 2 classes of healthy and unhealthy leaves with 256x256 pixels of each leaf.

\

1.4.1 Data Augmentation

In machine learning, data augmentation is utilized to expand the size and variety of a training dataset by generating synthetic data from existing data. This method is particularly advantageous for detecting plant diseases, as there may be a restricted number of images accessible for training.

There are several ways to perform data augmentation in the detection of plant diseases using machine learning, including:

1. Image rotation: By rotating the image at various angles, new images can be created from the existing data.
2. Image flipping: Flipping the image horizontally or vertically can create new images.
3. Image scaling: Scaling the image up or down can create new images.
4. Image cropping: Cropping the image at different sizes and positions can create new images.
5. Image translation: Shifting the image in different directions can create new images.
6. Color augmentation: Changing the brightness, contrast, and saturation of the image can create new images.

By applying these techniques, a larger and more diverse training dataset can be created, which can improve the accuracy of the machine learning model in detecting plant diseases. It's worth noting, however, that data augmentation alone cannot guarantee good performance; other factors such as the choice of algorithm, hyperparameters, and the quality of the original dataset also play important roles.

Fig 1.1 Infected leaf

Fig 1.2 Infected leaf

Fig 1.3 Infected leaf

1.4.2 Data Acquisition

Acquiring data is an essential aspect of developing machine learning models for detecting plant diseases. There are several methods available for data acquisition, including capturing images of diseased plants using a high-resolution camera in a greenhouse, field, or laboratory setting. Multiple images should be taken from different angles and lighting conditions to create a diverse dataset. Another option is to use existing datasets, such as PlantVillage, Plant Pathology 2020, and FungiNet, which already contain images of plants with and without diseases. Crowdsourcing platforms like Zooniverse or Amazon Mechanical Turk can also be utilized to collect labeled images of plant diseases from numerous contributors. This approach may require some supervision to ensure accurate labeling and can be time-consuming.

Engaging with plant pathologists or agronomists who specialize in plant diseases can offer access to crucial data and expertise. They can advise on the selection of images to collect, which diseases to prioritize, and how to label images accurately. After acquiring the data, it's vital to correctly label the images with the appropriate disease class. This process is typically accomplished manually by human experts or with the aid of crowdsourcing platforms. Once the images have been labeled, they are utilized to train a machine learning model to effectively classify new, unseen images of diseased plants.

1.5 Organization

The organisation of the report is as follows:

1) Chapter 1- Introduction

2) Chapter 2- Literature Survey 3)

Chapter 3- System Development 4)

Chapter 4- Performance Analysis 5)

Chapter 5- Conclusions

Chapter-2 LITERATURE SURVEY

Machine learning approaches offer several advantages over traditional methods in detecting plant diseases, including the ability to process large amounts of data quickly and accurately, detect subtle changes in plant health, and provide standardized diagnoses without subjectivity. Despite significant advancements in machine learning for plant disease detection, challenges remain, such as the requirement for large and varied datasets, generalization of models to new environments, and the interpretation of models for decision-making. However, the application of machine learning has the potential to create efficient and effective tools for detecting and managing plant diseases.

1) **S. S. Sannakki Et al. [1]** proposed a "Classification of Pomegranate Diseases Based on Back Propagation Neural Network" which mainly works on the method of Segment the defected area and color and texture are used as the features. They have used neural networks here.
2) **P. R. Rothe Et al. [2]** introduced a" Cotton Leaf Disease Identification using Pattern Recognition Techniques" which Uses snake segmentation, here Hu's moments are used as distinctive attribute. Active contour model used to limit the vitality inside the infection spot, BPNN classifier tackles the numerous class problems. The average classification is found to be 85.52%.
3) **Aakanksha Rastogi Et al. [3]** " Leaf Disease Detection and Grading using Computer Vision Technology &Fuzzy Logic". K-means clustering used to segment the defected area; GLCM is used for the extraction of texture features, Fuzzy logic is used for disease grading. They used artificial neural network (ANN) as a classifier which mainly helps to check the severity of the diseased leaf.

4) **Godliver Owomugisha Et al. [4]** Automated Vision-Based Diagnosis of Banana Bacterial Wilt Disease and Black Sigatoka Disease “Color histograms are extracted and transformed from RGB to HSV. Peak components are used to create max tree, five shape attributes are used and area under the curve analysis is used for classification.They used nearest neighbors, Decision tree, random forest, extremely randomized tree, Naïve bayes and SV classifier. In seven classifiers extremely, randomized trees yield a very high score, provide real time information provide flexibility to the application.

5) **Chunjiang Zhao Et al. [5]** SVM-based Multiple Classifier System for Recognition of Wheat Leaf Diseases,” Color features are represented in RGB to HIS, by using GLCM, seven invariant moment are taken as shape parameter. They used SVM classifier which has MCS, used for detecting disease in wheat plant offline.

Chapter-3 SYSTEM DEVELOPMENT

3.1 Analysis/Design/Development/Algorithm

Augmentation Process: The augmentation process involves increasing the size of the dataset and introducing slight distortions to images to prevent overfitting during training. Various transformation techniques, including affine and perspective transformations and image rotations, can be used. Affine transformations are useful for translations and rotations, where parallel lines in the original image remain parallel in the output image. To find the transformation matrix, three points from the original image and their corresponding locations in the output image are required. Straight lines are not affected by the transformation. Simple image rotations are commonly applied during the augmentation process, with rotations occurring on different axes by various degrees. Functional requirements in software engineering refer to defining the function of a software system or its component.

Functional requirements refer to the specific tasks that a software system or its components must perform, including calculations, data processing, and technical details. In this context, the system needs to perform several tasks, including taking in simple data in the form of .excel or .csv extensions, performing data cleaning or preprocessing to prepare the data for further processing, visualizing the data through charts or images of leaves, and ultimately providing a predicted output based on the processed data.

Having appropriate datasets is crucial for all stages of object recognition research, including the training phase and evaluating the performance of recognition algorithms. The dataset used in this research was collected from the Internet and included images of plant diseases and healthy leaves, as well as background images. The images were grouped into fifteen different classes, including thirteen classes representing plant diseases and one class

representing healthy leaves. Additionally, a class for background images was added to improve classification accuracy.

To distinguish the leaves from the surrounding, a deep neural network was trained on the dataset, including the background images taken from the Stanford background dataset. In systems engineering and requirements engineering, non-functional requirements refer to the criteria that can be used to judge the operation of a system, rather than specific behaviors, in contrast to functional requirements that define specific behavior or functions. The plan for implementing functional requirements is detailed in the system design, while the plan for implementing non-functional requirements is detailed in the system architecture. Other terms for non-functional requirements include "constraints", "quality attributes", "quality goals", "quality of service requirements", and "non-behavioral requirements".

3.2 HARDWARE REQUIREMENTS

Processor : Any Processor

RAM : more than 2GB

Hard Disk : 10 GB

Input device : Standard Keyboard and Mouse Output device : High Resolution Monitor

Memory : 1GB

RAM Requirements :Different plant leaf images (healthy and affected) 3.2.2 SOFTWARE

REQUIREMENTS Operating system : Windows XP and above, Windows 7

IDE : Python IDLE

Data : Leaf Dataset Modules : Numpy, pandas, matplotlib, sklearn Processor :Ip3 processor

3.3 INPUT REQUIREMENTS

Functional requirements in software engineering define the function of a software system or its components. Such requirements specify a set of inputs, behavior, and outputs that may include calculations, technical details, data manipulation, processing, and other functionalities that explain what the system aims to accomplish. Use cases capture the behavioral requirements that detail how the system uses the functional requirements. In this case, the system is expected to perform the following tasks: (1) acquiring simple data in an Excel or CSV format, (2) performing data cleaning or preprocessing to transform raw data for another processing procedure, (3) generating visual representations such as charts or images to display information, and (4) producing predicted output.

Techniques used:

Image processing: The user has to record into the system by giving his own information.

HOG: The registered user has to login into the app to add or see the disease, the user is recorded once he logs in until he logs out holding his time to login every other time.

Machine learning: The user is given chance to add disease choosing a town or area to refer to also add an image related to the news.

CNN: The user can see the disease based on the zone the user will select.

Artificial Intelligence : User can also view the list of disease added by him and action taken by admin if any.

Several disease: If the user finds any disease is not real or unpleasant he can say it to the admin, admin will take some act.

Random forest: Admin should login into the app to check disease or report submitted by the user.

SMV: Admin can see all the disease of all zones.

Viewing diseases: Admin is answerable to check any disease is been conveyed to him to cross checked.

To conduct object recognition research, appropriate datasets are essential throughout the process of training and evaluating recognition algorithms. In the case of a plant disease dataset, images were collected from various online sources using multiple languages to search for specific plant names and diseases. The images were categorized into fifteen classes, with thirteen classes representing diseased plants and one class for healthy leaves. An additional class for background images helped improve classification accuracy by training a deep neural network to distinguish leaves from the surrounding environment. These background images were obtained from the Stanford background dataset. Once a simple dataset is obtained in .excel or .csv format, data cleaning and preprocessing are performed to prepare the raw data for further processing. Following this step, visualization is done to represent the information as an image or chart. Finally, the predicted output is generated.

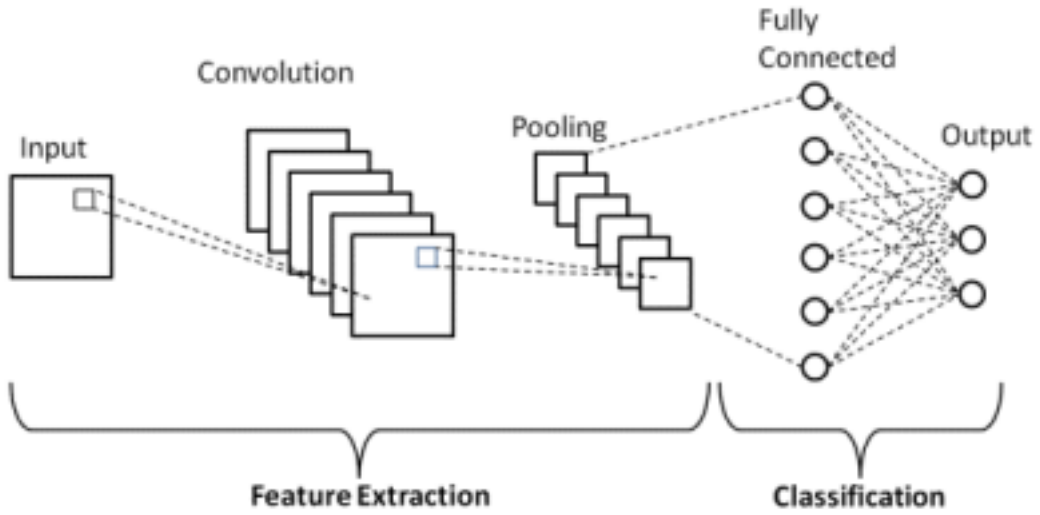


Fig. 3.1 Description of CNN

3.4 OUTPUT REQUIREMENTS

In the field of systems engineering and requirements engineering, a type of requirement known as output requirement defines criteria that can be used to evaluate the operation of a system, as opposed to specifying specific behaviors. This differs from functional requirements, which describe specific behaviors or functions of a system. The process of implementing functional requirements is described in the system design, while the implementation of non-functional requirements is detailed in the system architecture. Non-functional requirements may also be referred to as "constraints", "quality attributes", "quality goals", "quality of service requirements", or "non-behavioral requirements". Various quality attributes may be considered as part of non-functional requirements.

- **ACCESSIBILITY:** Accessibility refers to the extent to which a product, device, service, or environment can be easily used and accessed by the maximum number of people. In the context of our

project, it is accessible to anyone who has Python installed on their system.

- **MAINTAINABILITY:** In the field of software engineering, maintainability refers to the level of ease with which a software product can be altered to address defects, meet new requirements, or add new features/data. In the case of this specific project, new images of leaves can be easily added as new features/data.

- **SCALABILITY:** The system has the ability to handle an increase in total throughput when there is an increase in load, provided that additional resources, typically hardware, are added. Moreover, the system is capable of functioning properly even under conditions involving large datasets and low bandwidth.

- **PORTABILITY:**Portability is an important characteristic of high-level programming. It refers to the ability of software code to be reused when moving a program from one environment to another, rather than having to create new code. Our project is portable in that it can be executed under different operating conditions, as long as the minimum requirements are met and a suitable dataset is available. Specifically, our application is written in the Python programming language and packaged as an APK file for use on Android devices. The APK contains compiled Java code as well as other resources like scripts and images.

- **ACTIVITIES:**Artificial intelligence and machine learning are a subset of computer science that deals with the development of intelligent algorithms that can learn from data. These algorithms can be integrated into software applications and typically occupy a section of

the application's codebase. In the case of mobile devices, the AI/ML functionality often takes up the entire screen and is responsible for managing the user interface and interactions with the device's screen.

- **FRAGMENTS:** In Android, a Fragment is a modular section of a user interface that can be combined with other fragments to create a complete UI. A fragment can occupy only a part of the screen or the entire screen, depending on the size and complexity of the UI. Fragments are used to build flexible UI designs that can be reused across different activities. Fragments can contain views and view groups, and can be added or removed dynamically at runtime to create rich and responsive user experiences.

- **SMV AND HOG:** In Android, GUI elements are called Views, such as a TextView displaying text or a Button that users can click on. View Groups are containers for Views. A ViewGroup can contain a collection of Views together, and Views and View Groups can be nested within each other. Fragments and Activities can use XML files to define their layout and content. The layout XML files define the GUI elements included in an Activity or Fragment, as well as the layout of those GUI elements (size, margins, padding, etc.).

- **INTENTS:** System intents are minor substances that an activity can permit to the python operating organization, to tell the operating system that certain other act or action is required.

- **WIDGETS:** Machine widgets refer to graphical user interface elements that can be displayed independently of an activity. In the case

of plant disease detection, these widgets are used to display and visualize diseases as part of a detection request. In some cases, views of plant diseases may also be referred to as widgets.

- **SERVICES:** Even if no requests are displayed, family processes for plant disease can be implemented on a machine device. A user barrier is not what services want. A detection service might, for instance, periodically hold up data or check a remote server for the presence of illnesses.

3.5 LANGUAGE SPECIFICATION

1) PYTHON

Python is a popular high-level programming language known for its readability and use of significant whitespace. It was created in 1991 by Guido van Rossum and supports both small and large scale programming. C Python is the reference implementation of Python and is open source with a community-based development model. Python supports many programming paradigms including object-oriented, structured, functional, and aspect-oriented programming. It also uses dynamic typing and reference counting with a garbage collector for memory management. Python has built-in support for functional programming features like filter(), map(), and reduce() functions, as well as list comprehensions, dictionaries, and sets. The language has a core philosophy summarized in "The Zen of Python" document which includes aphorisms like "Simple is better than complex" and "Explicit is better than implicit."

2) CLUSTERING

A cluster refers to a set of data objects that share similar characteristics, while dissimilar data objects belong to different clusters. Clustering refers to the process of organizing abstract objects into similar groups. A cluster of data objects is treated as a single group. During cluster analysis, data is divided into groups based on their similarity and given labels. Unlike classification, clustering is adaptive and can identify features that distinguish different groups. The k-means algorithm will be used to cluster crime data in the proposed system.

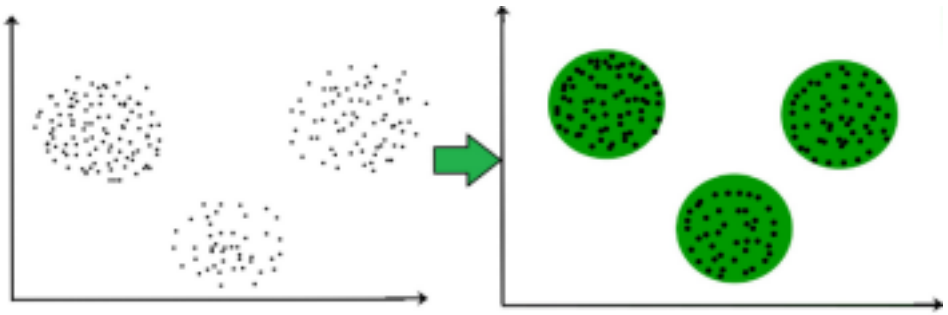


Fig 3.2 Example of Clustering

3) CLASSIFICATION

This method is employed to categorize various types of data. The way this works is comparable to clustering. It divides data records into several segments, also referred to as classes. In contrast to clustering, we are aware of the many clusters here. For instance, Outlook email uses an algorithm to classify emails as real or spam.

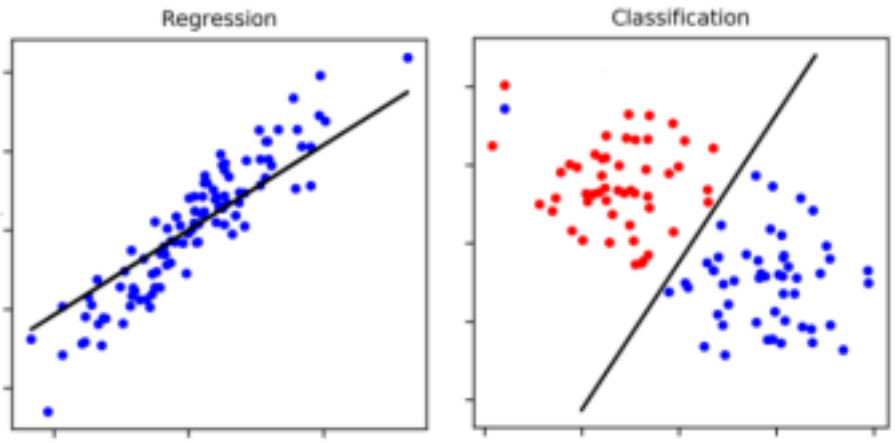


Fig 3.3 Classification versus Regression

4) **DECISION TREE**

This method plays a crucial part in machine learning since it is simpler for people to comprehend. The decision tree starts with a root that is a straightforward query. The questions on the root node of the decision tree may lead to more questions because they are open to multiple replies. As a result, the decision tree's nodes continue to grow. We're finally allowed to decide on it definitively.

Advantages:

1) More efficient.

2) Better crime monitoring systems.

3) Reduces the costs of storage, maintenance and personnel.

4) It reduces the time complexity of the system.

5) System that has a simpler architecture to understand.

6) Processing of large amount of data becomes easier.

JDK

(Java SE Development Kit) Contains a full JRE (Java Runtime Environment) in addition to tools for creating, troubleshooting, and keeping track of Java applications.

To create and run Java programmes and applets, JDK is necessary. Five categories are used to group JDK tools:
Simple Tools Tools for Remote Method Invocation (RMI) Internationalisation Tools for Security IDL Java Tools

JDK Starter Tools The Java Development Kit is built on these tools.

javac

The .java file is compiled using the Java programming language's compiler, javac. It generates a classfile that may be used with a Java command to run it. Java The java command can be used to execute a Java programme after a class file has been created.The command prompt is used to run both.

java

It is the extension for text files that include Java source code. After it has been coded and saved, the javac compiler is used to produce.class files. The Java command can be used to launch the Java programme as the.classfiles are prepared.

javadoc

JavaDoc is an API documentation generator for the Java programming language that creates documentation from Java source code in HTML format.

appletviewer

With the standalone command-line programme appletviewer, Java applets can be run and debugged without the need for a web browser. jar The jar file format (manage Java archive) is a collection of class, text, image, and sound files for a Java programme or applet that is compressed into a single file.

3.6 ECLIPSE SDK FEATURES

The Eclipse Project offers a range of resources and can be downloaded from its downloads page. Its sixteenth annual release provides several new features for the Platform and Equinox, Java developers, and plug-in developers. Eclipse is an IDE used for computer programming, particularly as a Java IDE, and includes a customizable plug-in system within its base workspace.

Additionally, the K means clustering algorithm is a simple unsupervised learning algorithm used for cluster analysis and is illustrated in a flowchart.

3.7 Decision Tree

This method plays a crucial part in machine learning since it is simpler for people to comprehend. The decision tree starts with a root that is a straightforward query. The questions on the root node of the decision tree may lead to more questions because they are open to multiple replies. As a result, the decision tree's nodes continue to grow. We're finally allowed to decide on it definitively.

Advantages:

1)More efficient.

2)Better crime monitoring systems.

3)Reduces the costs of storage, maintenance and

personnel. 4)It reduces the time complexity of the system.

5)System that has a simpler architecture to understand.

6)Processing of a large amount of data becomes easier.

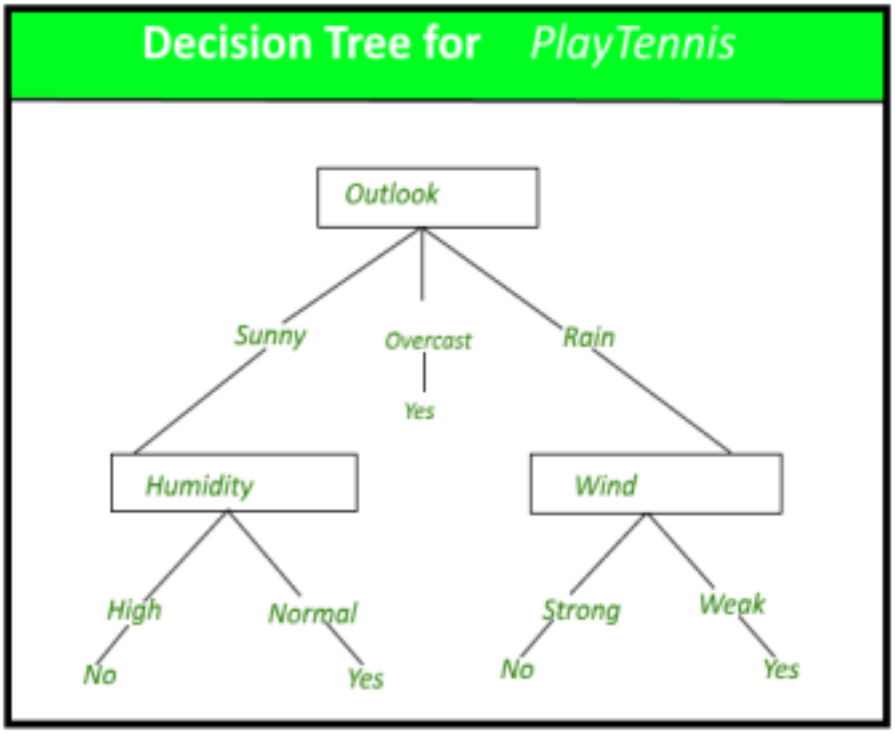


Fig. 3.4 Decision Tree for Playing Tennis

3.8 Support Vector Machine (SVM)

They have remained the go-to approach for an algorithm with great performance and little tuning since they were developed in the 1990s. Support vector machines are a class of supervised learning models with corresponding learning algorithms used in machine learning to analyse data used in classification and regression analysis. Unsupervised learning is not possible with unlabeled data. In a high-dimensional, infinite-dimensional space, it creates a hyperlane and a group of hyperlanes that can be utilised for various tasks, such as outlier detection.

Advantages

1) Effective in spaces with high dimensions.

2) Still useful in situations where there are more dimensions than

samples. 3) It also uses memory effectively.

4)Versatile.

5) Support vector machine (SVM) evaluates the data, classifies it, and then does the regression.

6) The two primary characteristics for diagnosing the condition are accuracy and detection time. SVMs, or support vector machines, boost recognition rates.

3.9 NEURAL NETWORKS

The most common architecture, multilayer perceptron neural networks, and Self-Organizing maps are the foundations of the illness detection systems.A hardware or software system known as a neural network is modelled after how neurons in the human brain function. Artificial neural networks are another name for neural networks. Neural networks are a type of machine learning model that are inspired by the structure and function of the human brain. They are composed of layers of interconnected nodes or neurons, each of which performs a simple mathematical operation on its inputs and passes the result on to the next layer. By adjusting the strength of the connections between

neurons during training, neural networks can learn to recognize patterns in data and make predictions or classifications.

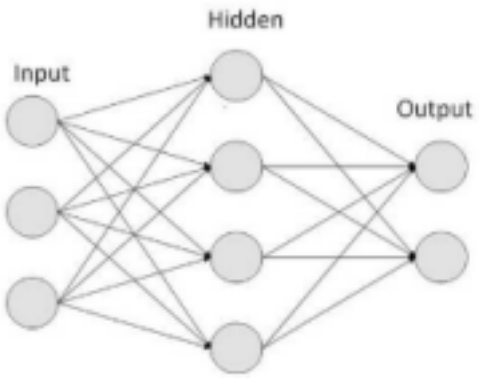


Fig. 3.5 Neural network

3.10 FUNCTIONALITIES USED

The connectivity among different plants is established through a series of activities. Initially, a picture of the plant leaf is captured in the first activity. The second activity involves image processing to analyze the captured image. A binary code of 1 and 0 is generated to indicate whether the leaf is healthy or unhealthy. If the leaf is healthy, it will be marked as 1, and if not, it will be marked as 0. The next step involves using machine learning algorithms to determine the type of disease that has affected the plant if it is unhealthy. Upon selecting the appropriate disease category, the final activity displays the disease name, allowing the farmer to apply appropriate fertilizer to the plant to eradicate the disease and pests. The last screen includes a message thanking the user for reading and an option to go back and read about other diseases.

3.11 KEY ISSUES AND CHALLENGES

Detecting wheat leaf diseases is important for crop health and productivity, but current machine learning methods face challenges. Digital images of wheat leaves are captured to measure their shape, size, and texture through feature extraction. Yellow rust disease caused by Puccinia striiformis Tritici produces yellow spores on leaves during winter and spring, and new protection methods like BTH aim to provide inherent disease resistance mechanisms. Seed-borne diseases can reduce yield and quality of grains. Machine vision uses color and geometric features for identification, but KNN classification is not suitable for large applications due to distance calculations. Histogram equalization enhances image contrast for better human perception and is used in wheat and plant disease applications. Hyperspectral data is collected for healthy and diseased wheat to build SVR models for disease index inversion and Random Forest classification of wheat leaf diseases.

Although the use of machine learning for plant disease detection shows promising results, there are still some key issues and challenges that need to be addressed.

1) Lack of Standardized Datasets: One of the major challenges is the lack of standardized datasets that can be used for training and testing machine learning models. This can lead to bias and inaccurate results, as different datasets may have different characteristics and variations.

2) Limited Availability of Expertise: Machine learning requires expertise in both computer science and plant pathology. The availability of individuals with expertise in both areas is limited, which can hinder the development and implementation of accurate models.

3) Variability in Environmental Conditions: Environmental factors such as lighting, humidity, and temperature can have a significant impact on plant health and appearance. Variability in these factors can make it challenging to develop accurate machine learning models that can detect diseases across different environments.

4) Limited Access to High-Quality Imaging Devices: High-quality imaging devices such as hyperspectral cameras or drones can provide detailed and accurate images of plants. However, these devices can be expensive and may not be accessible to all growers or researchers, limiting the data that can be collected for machine learning models.

5) Computational Complexity: Machine learning models can be computationally complex, requiring significant processing power and memory. This can make it challenging to develop models that can be implemented in low-resource settings.

6) Lack of Integration with Decision-Making Tools: While machine learning models can accurately detect plant diseases, they may not provide actionable recommendations for growers or decision-makers. There is a need for integration with decision-making tools to ensure that the results from machine learning models can be effectively translated into action.

Model Development

- Analytical
- Computational
- Experimental
- Mathematical

Chapter 4- PERFORMANCE ANALYSIS

The application of different models for prediction has provided the results in different forms such as a confusion matrix, a comparison graph, and a bar graph to compare the Accuracy, Precision, F1- Score and Recall of different models. These values have been figured out using the formulae given below.

4.1 Formulae

1) St = Tp / (Tp + Fn) Sensitivity (St) = True Positives / (True Positives + False Negatives) In simple terms, Sensitivity quantifies the proportion of actual positive instances that are correctly classified as positive by the model, out of all positive instances.
2) Sf = Tn/ (Tn + Fp) Specificity (Sf) = True Negatives / (True Negatives + False Positives) In simple terms, Specificity quantifies the proportion of actual negative instances that are correctly classified as negative by the model, out of all negative instances.
3) TPR = Tp/ (Tp + Fn)

   True Positive Rate is the number of instances that were predicted to be True and correctly identified as True.

4) FPR = Fp/ (Fp + Tn)

   False Positive Rate = False Positives / (False Positives + True Negatives)

   In simple terms, FPR quantifies the proportion of instances that are truly negative but are incorrectly classified as positive by the model, out of all actual negative instances.

5) A = (Tp + Tn)/ (Tp + Tn + Fp + Fn)

This is the formula for calculating the accuracy of the model. In simple terms, accuracy quantifies the proportion of correctly classified instances out of the total number of instances. It provides an overall measure of how well the model performs in terms of correctly predicting both positive and negative instances.

6) Precision(P) = Tp/ (Tp + Fp)

Precision = True Positives / (True Positives + False Positives) In simple terms, precision measures how many of the instances that the model predicted as positive are actually positive. It represents the model's ability to avoid false positives and is indicative of its accuracy when identifying positive instances.

7) Recall(R) = Tp/(Tp + Fn)

Recall = True Positives / (True Positives + False Negatives) In simple terms, recall quantifies the model's ability to correctly identify positive instances and captures the proportion of actual positives that were correctly classified as positive. It is a measure of the model's completeness or the proportion of relevant instances that were successfully retrieved.

8) ERR = (Fp + Fn)/ (Tp + Tn + Fp + Fn)

ERR = (False Positives + False Negatives) / (True Positives + False Positives + True Negatives + False Negatives) ERR (Error Rate) is a performance metric used in machine learning and classification tasks to measure the overall classification error of a model. It represents the proportion of misclassified instances, both false positives and false negatives, out of the total number of instances.

4.2 Result Analysis

The dataset is the Banana Leaves dataset, taken from Kaggle.com with 2000 different images, in two classes of Healthy and Unhealthy leaves, and the following models were applied on the dataset, the results are as follows.

1)VGG-16 Model

In Python, a file is regarded as a module. The module must be imported using the import keyword before being used. By importing the module, the function or variables existing in the file can be utilized in another file. Instead of importing the entire module, we can merely import the necessary functions and variable names from the module. The "from" keyword can be used to import only certain items when we only want those items imported. By using import and the module name, i.e., the filename or the library to be used, we can import the entire module. A Python library is a term that is widely used to describe a group of linked modules. It contains code bundles that can be used with many different software.

```
[ ] %time
    history=model.fit(X_train, y_train,validation_data=(X_test, y_test),verbose = 1,epochs = 15,callbacks=callbacks)

    CPU times: user 2 µs, sys: 1 µs, total: 3 µs
    Wall time: 6.68 µs
    Epoch 1/15
    8/8 [==============================] - 3s 234ms/step - loss: 39.0322 - accuracy: 0.7702 - val_loss: 20.2991 - val_accuracy: 0.8548
    Epoch 2/15
    8/8 [==============================] - 2s 345ms/step - loss: 9.7375 - accuracy: 0.6532 - val_loss: 0.7908 - val_accuracy: 0.8548
    Epoch 3/15
    8/8 [==============================] - 5s 769ms/step - loss: 0.4497 - accuracy: 0.9153 - val_loss: 0.2993 - val_accuracy: 0.9355
    Epoch 4/15
    8/8 [==============================] - 2s 213ms/step - loss: 0.1358 - accuracy: 0.9758 - val_loss: 0.2358 - val_accuracy: 0.9355
    Epoch 5/15
    8/8 [==============================] - 0s 36ms/step - loss: 0.0816 - accuracy: 0.9677 - val_loss: 0.4480 - val_accuracy: 0.8548
    Epoch 6/15
    8/8 [==============================] - 0s 40ms/step - loss: 0.0514 - accuracy: 0.9839 - val_loss: 0.4194 - val_accuracy: 0.9194
    Epoch 7/15
    8/8 [==============================] - 0s 37ms/step - loss: 0.0569 - accuracy: 0.9677 - val_loss: 0.4051 - val_accuracy: 0.9194
    Epoch 8/15
```

Fig 4.1 VGG-16 Epoch Results

Overfitting is a major issue when neural networks are trained using sample data. A neural network model will learn very precise patterns in the sample data if more training epochs are employed than are

necessary. As a result, the model won't fit the new data set properly.On The training data set (sample data), this model achieves great accuracy;however, on the test data set, it does not. In other words, by overfitting the training set, the model loses its capacity to generalize.

```
[ ]  from sklearn.metrics import classification_report
     print(classification_report(Y_true, Y_pred_classes))
```

	precision	recall	f1-score	support
0	0.93	1.00	0.96	53
1	1.00	0.56	0.71	9
accuracy			0.94	62
macro avg	0.96	0.78	0.84	62
weighted avg	0.94	0.94	0.93	62

Fig 4.2. The results of VGG-16 with an accuracy of 94%

The Accuracy achieved by the VGG-16 model is 94%. When the result might potentially comprise two or more classes, it is a technique to evaluate how effectively a deep learning categorization system works. The table contains four possible combinations of predicted and actual values. This program can be very effectively used to evaluate AUC-ROC curves, recall, precision, specificity, accuracy, and—most crucially—all of the aforementioned. The dataset was divided into two classes the first being the healthy class to train the model and the second being unhealthy leaves to check on the accuracy of the model.

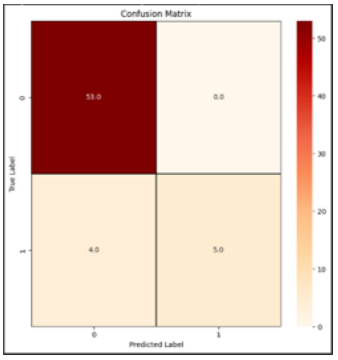


Fig 4.3 The Confusion Matrix for VGG-16 Model

The confusion matrix representing the True Positive, True Negative, False Positive and False Negative values across the two curves of Predicted values and Actual values. The True Positive value in this case is 5.0, meaning that of all the values in the dataset the predicted value was true as well as the actual value. The True Negative value is 53.0 meaning that the model predicted the value to be negative and the actual value was negative. False Positive value is 0.0 meaning that the predicted no value was predicted to be false and came out as true. The False Negative value is 4.0 meaning 4 of the values were predicted to be true but the actual value was false.

2) DenseNet201

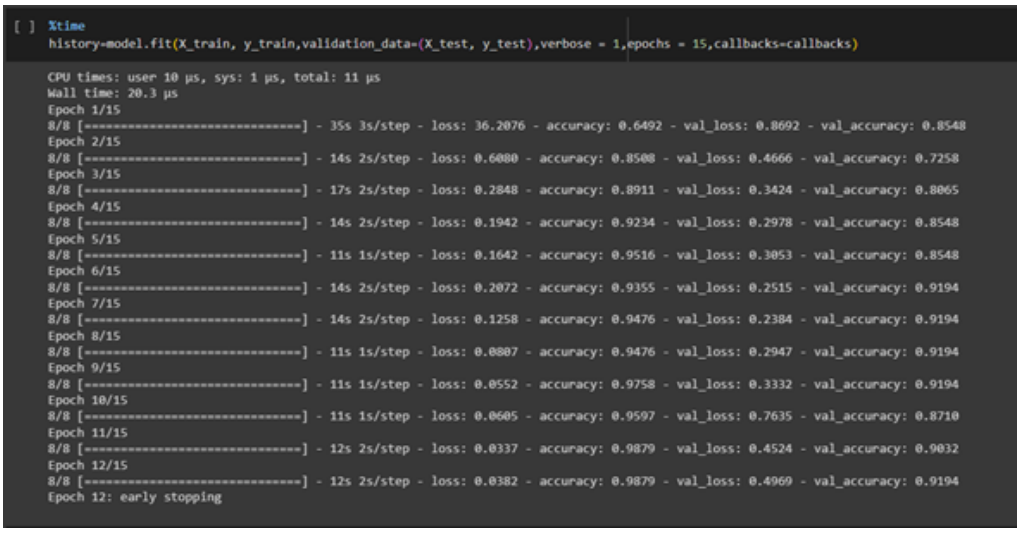


Fig 4.4 The epochs for DenseNet Model

The DenseNet Model stopped after 12 epochs due to early stopping. If we let a complex model train long enough on a given data set it can eventually learn the data exactly. Given data that isn't represented in the training set, the model will perform poorly when analyzing the data (overfitting). Conversely if the model is only trained for a few epochs, the model could generalize well but will not have a desirable accuracy (underfitting). Hence the model stops early to prevent it from learning the exact data.

```
[ ]  from sklearn.metrics import classification_report
     print(classification_report(Y_true, Y_pred_classes))
```

	precision	recall	f1-score	support
0	0.91	1.00	0.95	53
1	1.00	0.44	0.62	9
accuracy			0.92	62
macro avg	0.96	0.72	0.79	62
weighted avg	0.93	0.92	0.91	62

Fig 4.5 The results for the DenseNet Model with an accuracy of 92%

The accuracy of the DenseNet Model is 92% , with the Recall being 1.00, meaning that the False Negative value was 0, which means that there was no value where the model predicted the value to be False and the actual value was True.The Precision is 0.91 which means that the False Positive was greater than 0 meaning that there was a False value that was predicted to be True. The F1 Score is 0.95 which is the harmonic representation of both Precision and Recall, being this high means that the model is working well.

3) InceptionV3 Model

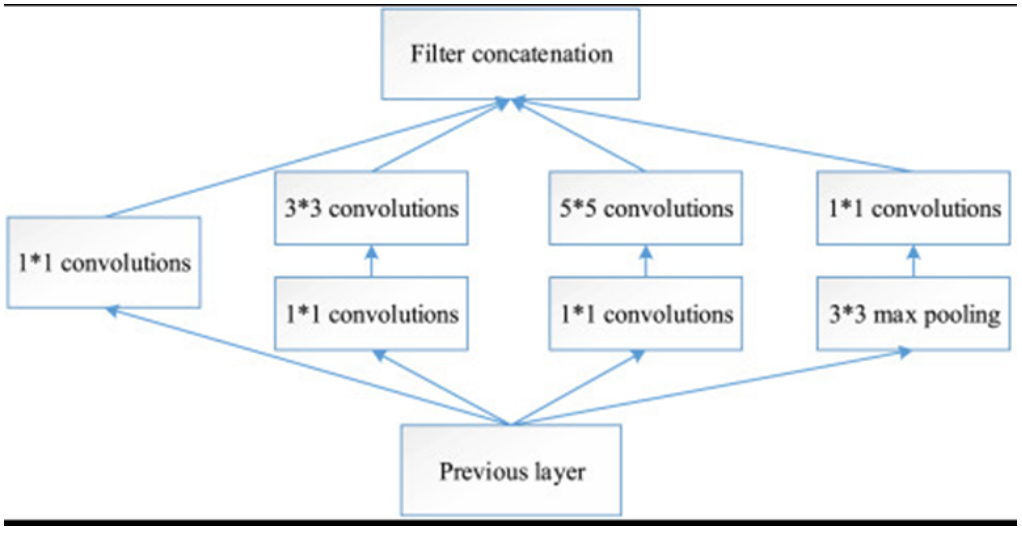


Fig 4.6 Architecture of the InceptionV3 Model

```
[ ] %time
    history=model.fit(X_train, y_train,validation_data=(X_test, y_test),verbose = 1,epochs = 15,callbacks=callbacks)

    CPU times: user 3 µs, sys: 1 µs, total: 4 µs
    Wall time: 7.39 µs
    Epoch 1/15
    8/8 [==============================] - 20s 2s/step - loss: 194.9368 - accuracy: 0.6653 - val_loss: 3.6731 - val_accuracy: 0.8548
    Epoch 2/15
    8/8 [==============================] - 10s 1s/step - loss: 4.9735 - accuracy: 0.7782 - val_loss: 3.4121 - val_accuracy: 0.8548
    Epoch 3/15
    8/8 [==============================] - 11s 1s/step - loss: 1.2405 - accuracy: 0.8105 - val_loss: 1.1455 - val_accuracy: 0.8548
    Epoch 4/15
    8/8 [==============================] - 11s 1s/step - loss: 0.5374 - accuracy: 0.8347 - val_loss: 0.3870 - val_accuracy: 0.8387
    Epoch 5/15
    8/8 [==============================] - 8s 1s/step - loss: 0.3565 - accuracy: 0.8589 - val_loss: 0.4164 - val_accuracy: 0.8387
    Epoch 6/15
    8/8 [==============================] - 12s 2s/step - loss: 0.2643 - accuracy: 0.8750 - val_loss: 0.3766 - val_accuracy: 0.8548
    Epoch 7/15
    8/8 [==============================] - 11s 1s/step - loss: 0.2108 - accuracy: 0.9194 - val_loss: 0.3673 - val_accuracy: 0.8548
    Epoch 8/15
    8/8 [==============================] - 11s 1s/step - loss: 0.1392 - accuracy: 0.9516 - val_loss: 0.3661 - val_accuracy: 0.8548
    Epoch 9/15
    8/8 [==============================] - 8s 996ms/step - loss: 0.2271 - accuracy: 0.9274 - val_loss: 0.4236 - val_accuracy: 0.8871
    Epoch 10/15
    8/8 [==============================] - 9s 1s/step - loss: 0.2102 - accuracy: 0.9113 - val_loss: 0.5105 - val_accuracy: 0.7903
    Epoch 11/15
    8/8 [==============================] - 9s 1s/step - loss: 0.1849 - accuracy: 0.9315 - val_loss: 0.4429 - val_accuracy: 0.8226
    Epoch 12/15
    8/8 [==============================] - 8s 997ms/step - loss: 0.1487 - accuracy: 0.9476 - val_loss: 0.4777 - val_accuracy: 0.8871
    Epoch 13/15
    8/8 [==============================] - 12s 2s/step - loss: 0.1045 - accuracy: 0.9476 - val_loss: 0.3414 - val_accuracy: 0.9032
    Epoch 14/15
    8/8 [==============================] - 8s 1s/step - loss: 0.0685 - accuracy: 0.9677 - val_loss: 0.3935 - val_accuracy: 0.9032
    Epoch 15/15
    8/8 [==============================] - 8s 926ms/step - loss: 0.0707 - accuracy: 0.9677 - val_loss: 0.4041 - val_accuracy: 0.9032
```

Fig 4.7 The epochs running for InceptionV3 Model

The InceptionV3 Model ran all the epochs without stopping early meaning that there was enough variation in the dataset for the model to run all the way through because the model stops early if there is not enough variation to prevent overfitting.

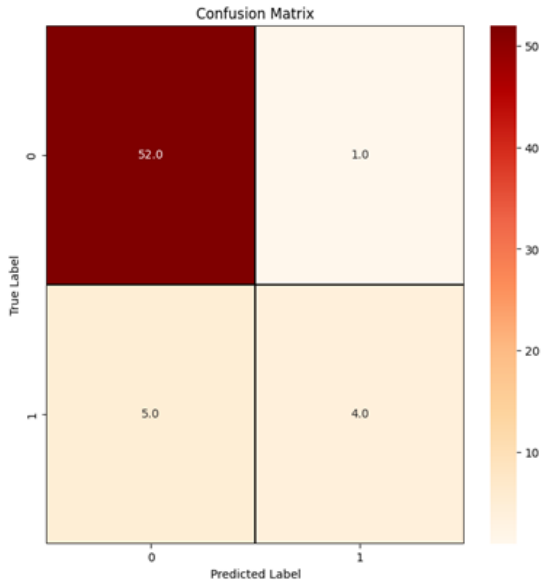


Fig 4.8 Confusion Matrix for InceptionV3 Model

The True Positive value for this model is 4.0 which means that there were 4 cases that were predicted to be True and their actual value was True, there are 53.0 cases of True Negative meaning they were predicted to be False and the actual value was False.The False Negative value is 1.0 meaning that one value was predicted to be false but it turned out to be True.

```
[ ]  from sklearn.metrics import classification_report
     print(classification_report(Y_true, Y_pred_classes))
```

	precision	recall	f1-score	support
0	0.91	0.98	0.95	53
1	0.80	0.44	0.57	9
accuracy			0.90	62
macro avg	0.86	0.71	0.76	62
weighted avg	0.90	0.90	0.89	62

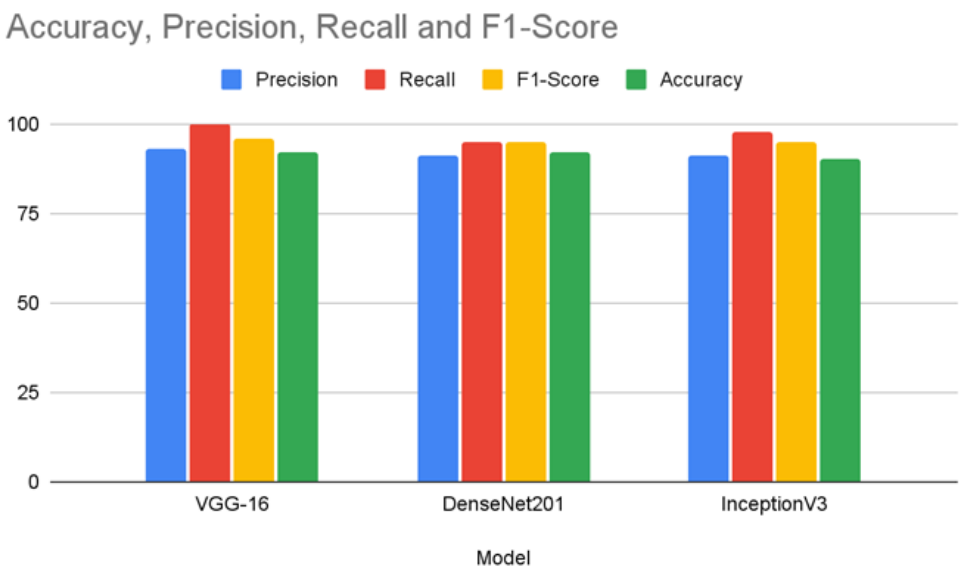
Fig 4.9 The results for InceptionV3 with an accuracy of 90%

The accuracy for the InceptionV3 Model is 90%, with the Precision being 91% and the Recall being 98% which means neither the False Positive was 0 nor the False Negative, therefore there were a few cases where the model predicted the wrong value. The F1- Score is 95% which is the harmonic representation of both the Precision and Recall, meaning that the model is giving a majority correct prediction.

Model	Precision	Recall	F1-Score	Accuracy
VGG-16	0.93	1.0	0.96	94
DenseNet201	0.91	0.95	0.95	92
InceptionV3	0.91	0.98	0.95	90

Table 1. Comparison of results between different models with the Banana Leaf Dataset

The highest accuracy is 94% achieved by the VGG-16 Model.



Graph 1. Graph showing results on the Banana Leaf Dataset

The above graph compares the Precision, F1- Score and the Recall of all three models and the precision is the highest for the VGG-16 Model, the Recall is the highest for the VGG-16 Model, InceptionV3 Model has the highest F1-Score and overall DenseNet201 has the highest accuracy.

Applying Ensemble Learning

In ensemble learning, the averaging technique is a common method used to combine the predictions of multiple base models. It is primarily employed when dealing with regression problems, where the goal is to predict a continuous value rather than a class label. The averaging technique works by taking the average of the predictions made by each base model in the ensemble. This can be done using a simple arithmetic mean or by applying weighted averaging, where each model's prediction is assigned a specific weight.

We have used the two models with the highest accuracy and applied ensemble learning to it, the models being DenseNet201 and VGG-16 with the Banana Leaf Dataset.

```
from sklearn.metrics import classification_report
print(classification_report(Y_true, Y_pred_classes))
```

```
              precision    recall  f1-score   support

           0       0.85      1.00      0.92        53
           1       0.00      0.00      0.00         9

    accuracy                           0.85        62
   macro avg       0.43      0.50      0.46        62
weighted avg       0.73      0.85      0.79        62
```

Fig 4.10 The Results of Ensemble Learning on both the classes

Class	Precision	Recall	F1-Score	Accuracy
0	0.85	1.0	0.92	
1	0.00	0.00	0.00	85

Table 2. The Results of Ensemble Learning on both the classes

Macro average calculates the Precision, F1- Score and Recall for each class and then finds the average by combining it with the other class and calculating the average of both. This helps provide a definitive view of both the classes, giving equal weight to each class. Weighted average Precision is calculated by taking the sum of all True Positives and then divided it by the sum of all True Positives and False Positives of both the classes which gives a definite view of the whole dataset. The same happens with Weighted average recall and F1-Score. The accuracy is 85% for this combined model as it takes out the redundancies in this previous models and gives a more precise viewpoint of the situation.

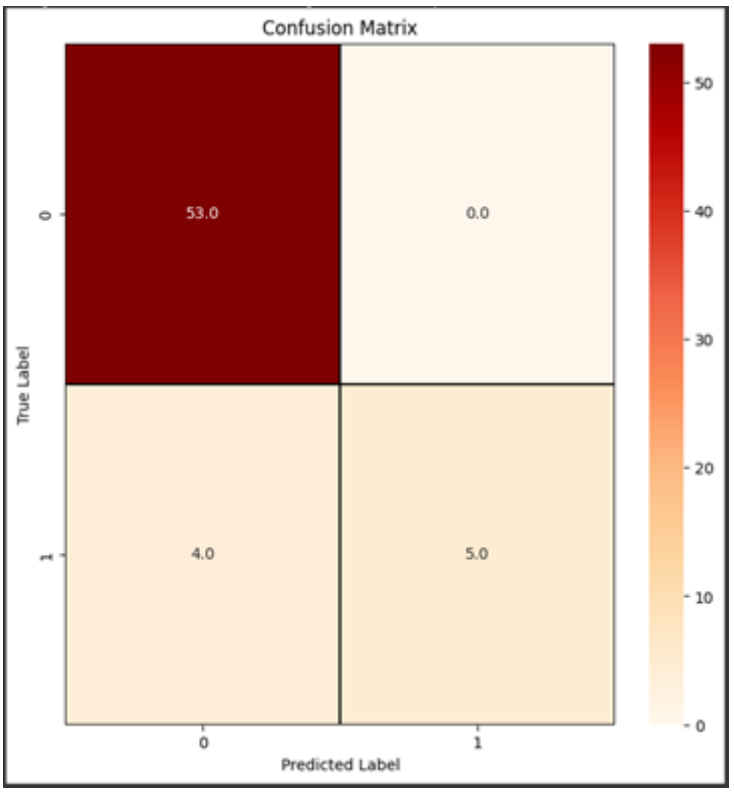


Fig. 4.11 The Confusion Matrix Representing for the Ensemble Learning Model

The confusion matrix representing the True Positive, True Negative, False Positive and False Negative values across the two curves of Predicted values and Actual values. The True Positive value in this case is 5.0, meaning that of all the values in the dataset the predicted value was true as well as the actual

value. The True Negative value is 53.0 meaning that the model predicted the value to be negative and the actual value was negative. False Positive value is 0.0 meaning that the predicted no value was predicted to be false and came out as true. The False Negative value is 4.0 meaning 4 of the values were predicted to be true but the actual value was false.

Chapter- 5 CONCLUSIONS

After applying ensemble learning, there are several potential conclusions that can be drawn based on the results and performance of the ensemble model. These conclusions may vary depending on the specific problem, the ensemble technique used, and the evaluation metrics employed. Here are the conclusions that can be made:

1) Improved performance: Ensemble learning leads to improved performance compared to using a single model. If the ensemble model outperforms the individual base models, it suggests that the combination of diverse models has effectively captured different patterns and achieved better predictive accuracy or generalization. The use of Averaging Technique has given a better view of the whole dataset and prevented overfitting as the model had gotten used to the dataset but with Averaging Technique this problem was prevented.
2) Reduced overfitting: Ensemble learning helped reduce overfitting, particularly if the individual base models is prone to overfitting the training data. If the ensemble model shows improved performance on unseen data or during cross-validation compared to individual models, it indicates that the ensemble has successfully mitigated overfitting.
3) Robustness to noise and outliers: Ensemble models are more robust to noise and outliers in the data compared to single models. If the ensemble model exhibits better performance in the presence of noisy or outlier data points, it suggests that the ensemble has effectively learned to ignore or handle these anomalies. The anomalies can cause the model to give exaggerated performances and give very high accuracies which was prevented by the use of this Ensemble Model.

4) Model diversity: Ensemble learning relies on the diversity of the base models to improve performance. Analyzing the base models' diversity, such as their differences in training algorithms, hyperparameters, or feature subsets, can provide insights into the effectiveness of ensemble learning. If the base models are diverse and yet contribute positively to the ensemble, it indicates the successful utilization of ensemble techniques

In conclusion, using machine learning algorithms to identify plant diseases has the potential to completely transform the plant health management industry. Machine learning algorithms can accurately diagnose and categorize plant diseases using a variety of data sources, including pictures, sensor data, and environmental characteristics. This technology has several benefits, such as increased accuracy, early detection, scalability, and the capacity to manage intricate data interactions.

The potential for advancements in areas like the fusion of various data sources, real-time monitoring and early warning systems, transfer learning, deep learning for image analysis, and integration with decision support systems make ML-based disease detection look promising in the future. These advancements may result in more specialized and proactive illness management approaches, allowing for prompt treatments and reducing financial losses.

Overall, the application of machine learning algorithms for plant disease detection has the potential to transform agriculture by enabling early and precise disease identification, optimizing resource allocation, and encouraging sustainable and effective farming methods. The creation of creative methods for enhancing agricultural yields, lowering pesticide facilitated by continued developments in this field.

5.1 Future Scope

For the future of this project, the best way to go forward with it would be the use of more Machine Learning Models and combining their results using Ensemble learning or the use of Deep Learning Models which can bring a huge change to the accuracy and prediction rate of this project. Moreover the use of different datasets can also help, using the datasets for different species of leaves plants. The use of drones in photographing leaves can change the game for detection of plant diseases as the use of drones combined with machine learning algorithms has emerged as a powerful tool for several advantages in this context, including the ability to capture high-resolution aerial imagery, cover large areas efficiently, and provide real-time data collection. When combined with machine learning algorithms, these advantages can significantly enhance disease detection and management efforts.

Drones can cover large areas of agricultural land quickly and efficiently, allowing for the collection of real-time data on plant health. This rapid data acquisition enables early detection and timely intervention, reducing the spread and impact of diseases.

Machine learning algorithms, such as convolutional neural networks (CNNs), can be trained to analyze drone-captured images and identify disease symptoms. By training the algorithms on large datasets of annotated images, they can learn to recognize various diseases. interventions in disease management. By identifying and mapping disease hotspots, farmers can apply treatments only where necessary, reducing the use of pesticides and optimizing resource allocation.

REFERENCES

1) Ghaiwat Savita N, Arora Parul. Detection and classification of plant leaf diseases using image processing techniques: a review. Int J Recent Adv Eng Technol 2014. (Online).
2) Dhaygude Sanjay B, Kumbhar Nitin P. Agricultural plant leaf disease detection using image processing. Int J Adv Res Electr Electron Instrum Eng 2013.
3) Mrunalini R Badnakhe, Deshmukh Prashant R. An application of K-means clustering and artificial intelligence in pattern recognition for crop diseases. Int Conf Adv Inf Technol 2011 IPCSIT.
4) Arivazhagan S, Newlin Shebiah R, Ananthi S, Vishnu Varthini S. Detection of unhealthy region of plant leaves and classification of plant leaf diseases using texture features. Agric Eng Int CIGR 2013.
5) Kulkarni Anand H, Ashwin Patil RK. Applying image processing technique to detect plant diseases. Int J Mod Eng Res 2012.
6) Bashir Sabah, Sharma Navdeep. Remote area plant disease detection using image processing. IOSR J Electron Commun Eng.
7) Naikwadi Smita, Amoda Niket. Advances in image processing for detection of plant diseases. Int J Appl Innov Eng Manage 2013.
8) Patil Sanjay B et al. Leaf disease severity measurement using image processing. Int J Eng Technol 2011.
9) Chaudhary Piyush et al. Color transform based approach for disease spot detection on plant leaf. Int Comput Sci Telecommun 2012.

APPENDICE

Here are some additional details that could be included in an appendix for the detection of plant diseases using machine learning:

1) Types of machine learning algorithms: There are several types of machine learning algorithms that can be used for plant disease detection, including supervised learning, unsupervised learning, and reinforcement learning. Each algorithm has its own strengths and weaknesses and is suited to different types of data and problems.

2) Performance metrics: There are several performance metrics that can be used to evaluate the performance of a machine learning model for plant disease detection, including accuracy, precision, recall, F1-score, and area under the curve (AUC). These metrics can help determine the effectiveness of the model and identify areas for improvement.

3) Image preprocessing techniques: Image preprocessing techniques such as normalization, resizing, and histogram equalization can be used to improve the quality and consistency of the images in the dataset. These techniques can help reduce noise and variability in the images, making it easier for the machine learning model to learn the patterns associated with diseased plants.

4) Transfer learning: Transfer learning is a technique that involves using a pre-trained machine learning model as a starting point for training a new model. This can be particularly useful for plant disease detection, as pre-trained models may have already learned features that are relevant to the problem at hand.

5) Deployment considerations: Once a machine learning model has been trained for plant disease detection, it must be deployed in a real-world setting. This may require integrating the model with existing software systems, developing a user interface, and ensuring that the model is robust and reliable.