VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi – 590018

Project Report

On

" HARNESSING AI FOR PRECISE ESTIMATION OF MEDICAL LEAF CHARACTERISTICS"

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING

in

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

By

SANJAY S M 4MT20AI038

Under the Guidance of

Mrs Radha E G

Assistant Professor

Department of Artificial Intelligence and Machine Learning

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING

(A Unit of Rajalaxmi Education Trust®, Mangalore)

Autonomous Institute affiliated to VTU, Belagavi, Approved by AICTE, New Delhi

Accredited by NAAC with A+ Grade & ISO 9001:2015 Certified Institution

2023-2024

MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING

A Unit of Rajalaxmi Education Trust®, Mangalore)

Autonomous Institute affiliated to VTU, Belagavi, Approved by AICTE, New Delhi

Accredited by NAAC with A+ Grade & ISO 9001:2015 Certified Institution

2023-2024

MITE

Invent Solutions

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

CERTIFICATE

This is to certify that the project work entitled **"HARNESSING AI FOR PRECISE ESTIMATION OF MEDICAL LEAF CHARACTERISTICS"** is a bonafide work carried out by **SANJAY S M (4MT20AI038)** in partial fulfilment for the award of degree of Bachelor of Engineering in Artificial Intelligence & Machine Learning of the **Visvesvaraya Technological University, Belagavi during the year 2023 – 24**. It is certified that all corrections and suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project has been approved as it satisfies the academic requirements in respect of project work prescribed for the Bachelor of Engineering degree.

Mrs.Radha E G	**Mr. Sunil Kumar S**	**Dr. Prasanth C M**
Project Guide	**Head of the Department**	**Principal**

External Viva

Name of the Examiners	**Signature With Date**
1)	
2)	

MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING
A Unit of Rajalaxmi Education Trust®, Mangalore)
Autonomous Institute affiliated to VTU, Belagavi, Approved by AICTE, New Delhi
Accredited by NAAC with A+ Grade & ISO 9001:2015 Certified Institution

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

DECLARATION

I, **SANJAY S M (4MT20AI038)** students of 8th semester BE in Artificial Intelligence & Machine Learning, **Mangalore Institute of Technology & Engineering, Moodabidri**, hereby declare that the project work entitled "**Harnessing AI For Precise Estimation Of Medical Leaf Characteristics**", submitted to the **Visvesvaraya Technological University, Belagavi** during the academic year **2023-24**, is a record of an original work done by us under the guidance of **Mrs. Radha E G**, Assistance Professor, Department of Artificial Intelligence & Machine Learning, Mangalore Institute of Technology & Engineering Moodabidri. This project work is submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Artificial Intelligence & Machine Learning. The results embodied in this report have not been submitted to any other University or Institute for the award of any degree.

Date:

Sanjay S M

Place:

ABSTRACT

Accurate assessment of medicinal plant leaf characteristics is essential for ensuring their quality and efficacy in healthcare applications. The project endeavours to revolutionize the field of herbal medicine by integrating cutting-edge Artificial Intelligence (AI) techniques with traditional medicinal plant classification methods. With the increasing demand for alternative and natural remedies, accurate identification and classification of medicinal plant species have become crucial for healthcare professionals, researchers, and enthusiasts alike. However, manual classification processes are time-consuming, labour-intensive, and prone to errors. To address these challenges, our project proposes a novel approach that combines the power of deep learning and classic machine learning algorithms. Users receive instant predictions along with supplementary information about the predicted plant species, including medicinal properties, geographical distribution, disease curability, age restrictions, medicinal values, side effects, and precautionary notes. By harnessing AI technologies, our project aims to democratize access to accurate medicinal plant classification and information. This system not only accelerates the identification process but also enhances the reliability and precision of classification results. Moreover, it empowers healthcare professionals, researchers, and individuals with the knowledge needed to make informed decisions about the usage of medicinal plants for various health conditions.

ACKNOWLEDGEMENT

The satisfaction and the successful completion of this project would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned my efforts with success.

This project is made under the guidance of **Mrs Radha E G**, Assistant professor, Department of Artificial Intelligence & Machine Learning. I would like to express my sincere gratitude to my guide for all the helping hand and guidance in this project.

I would like to thank my coordinator **Dr. Amirthavalli M,** Assistant Professor, in the Department of Artificial Intelligence & Machine Learning, for her cordial support, valuable information and guidance, which helped me in completing this project through the various stages

I would like to thank **Prof. Sunil Kumar S**, Head of the Department, Department of Artificial Intelligence & Machine Learning. whose constant support and encouragement helped me in the completion of this project.

I would like to thank my Principal **Dr. Prashanth C M**, for encouragement me and giving me an opportunity to accomplish the project.

I also thank my management who helped me directly or indirectly in the completion of this project.

My special thanks to faculty members and others for their constant help and support. Above all, I extend my sincere gratitude to my parents and friends for their constant encouragement with moral support.

SANJAY S M (4MT20AI038)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER _ 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 Introduction

Medicinal plants have been used in traditional medicine practices for a long time because of their nutrients and medicinal properties. Due to their bioactive compounds such as antioxidant, anti-allergic, anti inflammatory, and antibacterial properties. Different species of plants are recognized as having medical properties; they can be trees, shrubs or herbs. Their single diffusion depends on the environmental conditions they adapted to over time. According to the statistics, about 14–28% of all plants have medicinal uses. Furthermore, about 3–5% of patients in developed countries, and over 80% in the rural population in developing countries and about 85% of the population in the Southern Sahara use medicinal plants to treat their diseases because of their properties. Moreover, in developed countries, some people have turned to traditional medicines prepared from medicinal plants to treat and control illnesses and diseases after considering the harmfulness and side effects of chemical drugs. In addition to their medicinal uses, these plants can be used as food, beverages, and even in the form of cosmetics. Unfortunately, many counterfeit, low-quality, damaged, or not perfectly preserved medicinal plants are being manufactured and distributed worldwide, which may harm their users.

Botanists have long identified various species of medicinal plants using traditional and experience based methods. Nevertheless, visually and manually identifying medicinal plants from other similar plants can be extremely challenging and time-consuming for inexperienced people. Plants are generally classified based on their various organs, including roots, flowers, and leaves. The leaves, one of the most important organs of plants, largely differ among species and varieties in colors, shapes, and texture characteristics. However, in some cases, there have always been challenges associated with identifying medicinal plants due to the apparent similarities of their leaves. Furthermore, the leaf color cannot be considered a suitable option for plant classification due to their similarities and, most importantly, their variability during the growing period.

It is still difficult to distinguish medicinal herbs from other plants, even with vision-based systems that have significantly improved their ability to extract complex features and select the most important ones via conventional Machine Learning (ML) and Deep Learning. Therefore, the main objective of the current study was to develop a real-time automatic vision system for identifying medical plants using a proposed DL algorithm coupled with a machine vision

technique. To put it more clearly, this research aimed to improve the automatic identification of medicinal plants as their popularity is growing and they are increasingly being used for artisanal and industrial purposes

Importance of Medicinal Leaves in Ayurveda

Medicinal leaves hold profound significance in Ayurveda, where they are regarded as essential components in promoting health, preventing diseases, and aiding in the treatment of various ailments. These leaves possess a wide range of therapeutic properties, including anti-inflammatory, antimicrobial, and antioxidant effects, making them valuable for treating a diverse array of health conditions. Ayurveda emphasizes the use of natural remedies derived from plants, including leaves, to promote holistic healing and restore balance to the body's doshas.

The Need for Precise Estimation

Despite their importance, accurately identifying and characterizing medicinal leaves can be challenging due to their complex botanical features. Traditional methods of leaf analysis are often time-consuming, labor-intensive, and subject to human error. As a result, there is a pressing need for advanced technologies that can provide precise estimation of medicinal leaf characteristics, facilitating efficient plant identification, research, and conservation efforts.

Harnessing AI Technology

The project "Harnessing AI for Precise Estimation of Medicinal Leaf Characteristics" aims to address this need by leveraging the power of Artificial Intelligence. By developing robust AI models, such as deep learning algorithms like Convolutional Neural Networks (CNNs) and machine learning algorithm such as random forest, the project seeks to accurately identify and characterize medicinal leaves based on their botanical features, including shape, size, texture, and color. Through the integration of AI technology, the project endeavors to revolutionize the field of medicinal plant research, providing researchers, botanists, and healthcare practitioners with advanced tools for precise leaf analysis and classification.

1.2 Problem Statement

Traditional methods of assessing medicinal plant quality, often relying on laborious manual techniques, are prone to errors and inconsistencies. To address these limitations, we propose an AI-powered approach for precise and objective estimation of medicinal leaf characteristics. By employing image processing, machine learning, and geolocation detection, we aim to develop an machine learning model that can accurately predict leaf characteristics. This innovation holds immense promise for revolutionizing healthcare, ensuring the efficacy of herbal remedies and the quality of natural treatments.

1.3 Objectives

The objective aims to collect a dataset of medicinal leaf images, perform feature extraction, and develop models using both deep learning (VGG-16) and machine learning (Random Forest). Additionally, a voting classifier will be implemented to combine the predictions from both models for enhanced accuracy. The final step involves utilizing the trained models for predicting the species of medicinal leaves based on input images.

1. Dataset Collection:

- Gather a diverse dataset of medicinal leaf images from various sources such as botanical gardens, online databases, and field expeditions.
- Ensure the dataset includes a wide range of medicinal plant species, capturing variations in leaf shapes, textures, and colors.

2. Feature Extraction:

- Preprocess the collected leaf images to extract relevant features using techniques like edge detection, color histogram, and texture analysis.
- Convert the pre-processed images into feature vectors suitable for input into the machine learning and deep learning models.

3. Modeling using Deep Learning (VGG-16):

- Implement the VGG-16 architecture for deep learning-based classification of medicinal leaf images.
- Train the VGG-16 model on the extracted features from the dataset to learn the underlying patterns and characteristics of different plant species.

4. Modeling using Machine Learning (Random Forest):

- Develop a Random Forest classifier using the preprocessed feature vectors extracted from the leaf images.
- Train the Random Forest model on the dataset to build a machine learning-based classification model for medicinal leaves.

5. Majority Voting :

- Combine the predictions from both the deep learning (VGG-16) and machine learning (Random Forest) models using a voting classifier ensemble technique.
- Utilize the voting classifier to aggregate the individual predictions and make a final classification decision.

6. Prediction of Leaf Images:

- Deploy the trained models, including the VGG-16 deep learning model, Random Forest model, and voting classifier, for predicting the species of medicinal leaves based on input images.
- Evaluate the performance of the models on a separate validation dataset to assess their accuracy, precision, and recall.

1.4 Scope Of The Project

The scope of the project employs machine learning (ML) and deep learning techniques, notably the VGG16 architecture, to revolutionize the identification and characterization of medical leaves. By analyzing attributes such as shape, size, texture, and color, the AI system offers precise identification, benefiting botanists, researchers, and conservationists in taxonomy and biodiversity conservation efforts. Additionally, disease detection algorithms integrated into the system enable early intervention in both agricultural and medicinal contexts, enhancing plant health and the efficacy of herbal remedies.

The project encompasses various interconnected components aimed at revolutionizing the identification and characterization of medicinal plant leaves. At its core lies the development of advanced AI models, including deep learning architectures like VGG-16 and machine learning algorithms such as Random Forest, tailored specifically for the accurate analysis of leaf features. These models will undergo rigorous training and optimization to ensure robustness and effectiveness in identifying medicinal plant species based on their botanical characteristics. Moreover, the project entails the integration of these AI models into a user-friendly interface accessible via web and mobile platforms, facilitating easy upload of leaf images, real-time prediction, and intuitive visualization of results.

Additionally, the project emphasizes stakeholder engagement and collaboration with botanical experts and healthcare practitioners to validate model predictions and gather valuable insights for refinement. Ethical considerations regarding data privacy and sustainability in medicinal plant conservation are paramount throughout the project's implementation, ensuring responsible utilization of AI technology for the betterment of healthcare, research, and environmental preservation.

1.5 Organization Of The Report

- **Chapter-1: Introduction**

starts by introducing the topic, which gives an insight and in-depth analysis of the project work, which was closely followed by the problem statement which tells about the problems faced by formers, followed by the objectives of the topic, which gave rise to the significance of the topic, followed by the scope and crowned by the organization of the project.

- **Chapter-2: Literature Survey**

This chapter mainly deals with all the observation, which is conducted as initial study before the actual development of the project. It also describes the details regarding existing system and its disadvantages.

- **Chapter-3: System Requirement Specification**

This chapter speaks about the product perspective, user characteristics, its assumptions and dependencies, specific requirements, functionality along with resource requirements.

- **Chapter-4: Gantt chart**

Gantt chart presenting the overall schedule of the project, including milestones and timelines.

- **Chapter-5: System Design**

This chapter mainly deals with detailed design of the system based on the requirements specified in Chapter 3. This includes architectural diagrams, sequence diagram, data flow diagram, activity diagram.

- **Chapter-6: Implementation**

This chapter deals with Description of how the system was developed. Tools and technologies used and gives an overview of the implementation.

- **Chapter-7: Testing part**

  This chapter deals with the details of the testing methodologies employed, Results of the testing phase. Any issues encountered during testing and their resolutions.

- **Chapter-8: Result and Snapshots**

  This chapter deals with the results obtained from the project, Snapshots or screenshots of the implemented system. Analysis of the results in relation to the project objectives.

CHAPTER _ 2

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

A literature survey shows the various analysis and research made in the field of interest and results already published, taking into account the various parameters of the project and the extent of the project. It includes researches made by various analysts-their methodology and the conclusion they have arrived at. It is the most important part of the report as it gives the direction in the area of research.

2.1 BACKGROUND RESEARCH

These studies provide a comprehensive overview of the current methods used in medicinal plant identification using AI. The findings demonstrate the remarkable progress that has been made in this field. AI-based systems have demonstrated remarkable accuracy and efficiency in identifying medicinal plants, providing valuable tools for researchers, practitioners, and individuals alike. In our research through these papers we identify that CNN based models are much more accurate than the other models.

Sl No	Paper Title	Methodology	Identified Scope for improvement And improvement done
1	The Classification of Medicinal Plant Leaves Based on Multispectral and Texture Feature Using Machine Learning Approach [1]	Machine learning approach utilizing multispectral and texture features	• Improve classification accuracy using advanced ML techniques. • Enhance feature extraction methods. Explore transfer learning. • We have improved the accuracy of the machine learning model to 92%.
2	Automatic Recognition of Medicinal Plants using Machine Learning Techniques [2]	Random Forest Classifier, Feature Extraction and SVM Classification	• Handle lighting and leaf orientation variations. • Increase dataset diversity through data augmentation. • Address overfitting. • Increased the dataset to 30 species and performed data augmentation. Addressed overfitting issues.
3	A convolutional neural network-driven computer vision system toward identification of species and maturity stage of medicinal leaves [3]	APRS, convolutional neural networks (CNN), Geographic Information Systems (GIS) technology	• Enhance feature extraction for better discrimination. • Incorporate domain knowledge. Address overfitting. • Features extracted and overfitting addressed.
4	Deep convolutional neural network-based plant species recognition through features of leaf. Multimed [4]	Multilayer Perceptron (MLP) classifier	• Improve model interpretability. Handle intra-class variations. • Enhance class differentiation.
5	Recognition of leaves of different medicinal plant species using a robust image processing algorithm and artificial neural networks classifier [5]	artificial neural networks (ANN) classifier	• Address class imbalance. Incorporate uncertainty estimation. • Evaluate real-world performance. • Combined machine learning and deep learning models, used majority voting for the final output.

6	Leaf species and disease classification using multiscale parallel deep CNN architecture [6]	Deep Learning Techniques-Convolutional Neural Network (CNN)	• Evaluate real-world performance. Handle noisy/incomplete • data. Incorporate expert knowledge.
7	AI Based Indigenous Medicinal Plant Identification. In Proceedings of the 2020 Advanced Computing and Communication Technologies for High Performance Applications [7]	Multiscale parallel deep CNN architecture	• Improve efficiency for large datasets. • Handle intra-class variations. • Enhance class differentiation.
8	Efficient and automated herbs classification approach based on shape and texture features using deep learning [8]	Deep Learning-Based Approach, Advancement in Automated Classification Systems	• Enhance feature extraction for better discrimination. • Incorporate domain knowledge.
9	Optimized convolutional neural network model for plant species identification from leaf images using computer vision [9]	Optimized CNN model for plant species identification	• Handle occlusions and overlapping leaves. • Improve robustness to image variations. • Assess generalization across species.
10	A Vision Based System for Medicinal Plants Using Xception Features [10]	Vision-based system utilizing Xception features	• Handle lighting and leaf orientation variations. • Enhance interpretability.

CHAPTER - 3

SYSTEM REQUIREMENTS SPECIFICATION

CHAPTER 3

SYSTEM REQUIREMENT SPECIFICATION

3.1 Overall Description

A software requirements specification (SRS) is a description of a software system to be developed. The software requirements specification lays out functional and non-functional requirements, and it may include a set of use cases that describe user interactions that the software must provide. Use cases are also known as functional requirements. In addition, to use cases, the SRS also contains non- functional requirements. Non-functional requirements are requirements that impose constraints on the design or implementation. For the hardware requirements, the SRS specifies the logical characteristics of each interface between the software product and the hardware components. It specifies the hardware requirements, such as memory restrictions, cache size, processor, RAM size, etc., required for the software to run. Software requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign.

3.1.1 Product Perspective

- The web interface will be a user-friendly and easy to use with clear display of different images of leaves.
- Generally, user use this to know the state of onion by taking the real time image of onion or input the image from device.
- Web interface developed meets the need for non-destructive way of determining precise estimation of the medicinal leaf characteristic
- User gets the Result of predicted images of the leaf

3.1.2 Product Function

The Proposed project is deep learning CNN model and machine learning random forest which aims to automate the process of identifying the medicinal leaf characteristics. Initially image of the leaves is given as the input to the customized CNN model and then to random forest and image is processed through different layers of CNN and random forest. The result would be displayed on the screen.

3.1.3 User Classes and Characteristics

- Citizen scientists: These users may contribute to the development of the classification system by providing labeled data or participating in data collection activities. They may require guidance and training on how to collect and label data effectively.
- Data scientists and machine learning engineers: These users are responsible for designing, developing, and maintaining the classification system. They may require expertise in deep learning algorithms, data preprocessing, and model optimization.
- IT administrators: These users are responsible for deploying and maintaining the infrastructure for the classification system. They may require expertise in cloud computing, network security, and data management

3.1.4 Design and Implementation Constraints

1. Hardware Constraints

- Description: The performance of the model can be limited by the hardware used for training and inference. Training deep learning models requires powerful GPUs and large amounts of memory.
- Impact: Limited hardware resources may restrict the size and complexity of the model that can be trained, potentially affecting the accuracy and efficiency of the classification system.
- Mitigation: Utilize cloud-based GPU instances or invest in high-performance computing hardware to facilitate efficient model training and inference processes.

2. Data Availability

- Description: Availability and quality of data are critical constraints for any deep learning application. The absence of a readily available dataset for medical leaf characteristics estimation presents a significant challenge.
- Impact: The lack of a comprehensive dataset hinders the development and training of accurate and robust classification models, potentially limiting the system's effectiveness.
- Mitigation: Invest resources in data collection efforts, including data acquisition, annotation, and curation, to generate a diverse and representative dataset for training and evaluation purposes.

3.Computational Complexity

- Description: The computational complexity of the model can impact its scalability and performance. Deep learning models with a large number of layers and parameters require more computation time and memory.
- Impact: Complex models may require extensive computational resources for training and inference, potentially limiting scalability and real-time performance.
- Mitigation: Optimize model architectures and training procedures to minimize computational overhead while maintaining accuracy. Implement techniques such as model pruning and quantization to reduce model size and computational complexity.

4. Algorithmic Complexity

- Description: The choice of algorithms and techniques used for feature extraction, dimensionality reduction, and classification can impact the performance of the model.
- Impact: Suboptimal algorithmic choices may lead to reduced accuracy or efficiency of the classification system, particularly for complex or heterogeneous medical leaf datasets.
- Mitigation: Conduct thorough experimentation and benchmarking to identify the most suitable algorithms and techniques for the specific task of medical leaf characteristics estimation. Consider factors such as data distribution, feature representation, and computational efficiency when selecting algorithms.

5. Cost

- Description: The cost of developing and deploying a medical leaf characteristics estimation system can be significant, including expenses related to data collection and labeling, and personnel resources.
- Impact: High costs may pose financial constraints on the project, potentially limiting the scope or quality of the classification system.
- Mitigation: Develop a detailed budget and resource allocation plan to effectively manage project expenses. Explore cost-effective solutions for hardware procurement, data collection, and personnel recruitment, and prioritize investments based on project objectives and constraints. Consider leveraging open-source tools and resources to reduce software development costs.

3.1.5 Assumptions and Dependencies

1. Availability of High-Quality and Annotated Training Datasets

Assumption: The accuracy and performance of both deep learning and machine learning models heavily rely on the quality and quantity of the training data. It is assumed that high-quality annotated datasets are prepared to train the models effectively.

Dependency: The availability of annotated datasets impacts the feasibility and effectiveness of training the models for precise estimation of medical leaf characteristics.

2. Availability of Computing Resources

Assumption: Deep learning and machine learning models often require significant computing resources such as powerful CPUs or GPUs, large amounts of memory, and storage. It is assumed that the necessary computing resources are available to train and run the models efficiently.

Dependency: The availability of computing resources influences the scalability and performance of the system, impacting its ability to handle large-scale datasets and complex model architectures.

3. Dependence on External Libraries and Frameworks

Assumption: The development of deep learning and machine learning models often relies on external libraries and frameworks such as TensorFlow, Keras, and scikit-learn. It is assumed that the required libraries and frameworks are available and compatible with the development environment.

Dependency: The availability and compatibility of external libraries and frameworks facilitate the implementation and optimization of deep learning and machine learning algorithms, enabling efficient model development and deployment.

4. Dependence on Data Preprocessing Techniques

Assumption: The accuracy of both deep learning and machine learning models depends on how well the input data is preprocessed before training. It is assumed that effective data preprocessing techniques are available to extract relevant features and reduce noise from the input data.

Dependency: The availability and effectiveness of data preprocessing techniques influence the quality and reliability of the input data, affecting the performance and generalization ability of the models.

5. Dependence on Model Selection and Hyperparameter Tuning

Assumption: The performance of both deep learning and machine learning models depends on proper model selection and hyperparameter tuning. It is assumed that effective model selection and hyperparameter tuning techniques are available to optimize the models' performance.

Dependency: The availability of model selection and hyperparameter tuning techniques facilitates the optimization of model performance, enhancing their accuracy and robustness for medical leaf characteristics estimation tasks.

6. Limitations and Considerations of Deep Learning and Machine Learning Models

Assumption: Deep learning and machine learning models have inherent limitations and considerations, such as overfitting, interpretability, and generalization to unseen data. It is assumed that these limitations are understood and addressed in the design and implementation of the system.

Dependency: Awareness of the limitations and considerations of deep learning and machine learning models guides the design and implementation process, informing decisions regarding model architecture, training strategies, and evaluation methods.

3.2 Specific Requirements

This section includes a detailed description of the hardware requirements, software requirements, functional requirements, and non-functional requirements.

3.2.1 Hardware Requirements

Hardware requirements refer to the physical parts of a computer and related devices. Internal hardware devices include motherboards, hard drives and RAM External hardware devices include monitors, keyboards, mice, printers, and scanner.

Following shows the Hardware requirements for development of the project.

The software should run on any sort of desktop or laptop environment, regardless of the operating System. Essential input/output devices are keyboard, mouse, printers

1. **Processor:** Intel(R) Core (TM) i5-3320U CPU @ 2.60GHz 2.60 GHz
2. **Hard Disk:** 64-bit Operating System, x64-based processor
   4GB Other standard physical devices like keyboard, mouse etc.

3.2.2 Software Requirements

Software requirement is a field within Software Engineering that deals with establishing the Deeds of stakeholders that are to be solved by the software. The software requirements of our project given below:

1. Programming Language:

Python is the most commonly used language for machine learning. It offers a wide range of libraries and frameworks for data processing, modeling, and visualization.

2. Integrated Development Environment (IDE):

Choose an IDE for Python development. Popular choices include:

Visual Studio Code

3. Machine Learning Libraries:

You will need several libraries for machine learning, including

scikit-learn(sklearn), Keras, h5py, oauthlib, jsonschema, matplotlib, NumPy, pandas, Streamlit, TensorFlow

1. Python: Python is a versatile, high-level programming language renowned for its simplicity, readability, and extensive community support. Widely adopted across diverse fields such as web development, data science, artificial intelligence, and more, Python offers a rich ecosystem of libraries and frameworks for various tasks. Its clean syntax and dynamic typing make it an ideal choice for rapid prototyping and development. Python's interpretive nature allows for cross-platform compatibility, enabling seamless deployment across different operating systems.

2. Transfer learning: It is a machine learning technique where a model trained on one task is reused or adapted for a different but related task. Instead of training a model from scratch, transfer learning leverages the knowledge learned from a source domain to improve the performance of a target domain. This approach is particularly useful when the target dataset is small or when computational resources are limited. In transfer learning, the pre-trained model, often a deep neural network trained on a large dataset, serves as a feature extractor or provides initial weights for the target task. By fine-tuning the pre-trained model on the target dataset, the model can quickly adapt to the specific characteristics of the new data.

3. scikit-learn (sklearn): scikit-learn is a powerful machine learning library in Python, providing a comprehensive set of tools for data preprocessing, model training, evaluation, and optimization. With a user-friendly interface and extensive documentation, scikit-learn simplifies the implementation of machine learning algorithms for classification, regression, clustering, and dimensionality reduction tasks. Leveraging NumPy, SciPy, and other scientific computing libraries, scikit-learn offers efficient and scalable implementations of popular machine learning algorithms. Its consistent API design and well-documented functionality make it accessible to both novice and experienced users, facilitating seamless integration into existing Python workflows.

4. Keras: Keras is a high-level neural networks API written in Python, designed for rapid prototyping and experimentation with deep learning models. Built on top of TensorFlow, Theano, or Microsoft Cognitive Toolkit (CNTK), Keras provides a user-friendly interface for building and training neural networks. Its modular architecture allows for easy composition of complex models using reusable building blocks called layers. Keras abstracts away low-level implementation details, enabling developers to focus on model design and experimentation.

5. TensorFlow: TensorFlow is an open-source deep learning framework developed by Google, designed for building and deploying machine learning and deep learning models at scale. With its flexible architecture and extensive support for distributed computing, TensorFlow enables efficient training and inference on large datasets across heterogeneous hardware environments. TensorFlow's computational graph abstraction allows for symbolic execution of complex mathematical operations, facilitating automatic differentiation and optimization during model training. Its high-level APIs, including Keras and Estimator, provide user-friendly interfaces for building and training deep learning models with ease.

6. Streamlit: Streamlit is an open-source Python library for creating interactive web applications for machine learning and data science projects. With Streamlit, developers can quickly build and deploy data-driven applications using simple Python scripts, without the need for web development expertise. It provides built-in components for creating interactive widgets, visualizations, and dashboards, enabling seamless integration of machine learning models and data analysis tools. Streamlit's intuitive API and automatic widget generation streamline the development process, allowing developers to focus on building engaging and interactive user

interfaces. Its flexible layout system and responsive design support a wide range of application scenarios, from exploratory data analysis to model deployment. Streamlit's active community and extensive documentation provide ample resources for learning and troubleshooting, making it accessible to developers of all skill levels. With its emphasis on simplicity, flexibility, and performance, Streamlit has gained popularity as a go-to framework for building interactive data applications in Python.

5. Deep Learning: Deep learning is a subset of machine learning focused on training neural networks with multiple layers to learn complex patterns and representations from raw data. By leveraging hierarchical architectures and nonlinear activation functions, deep learning models can automatically extract and learn intricate features from high-dimensional input data. Deep learning has achieved remarkable success in various domains, including computer vision, natural language processing, speech recognition, and reinforcement learning. Its ability to handle large volumes of data and learn hierarchical representations makes it particularly well-suited for tasks involving unstructured data such as images, text, and audio

6. VGG-16: VGG-16 is a deep convolutional neural network architecture proposed by the Visual Geometry Group (VGG) at the University of Oxford. With 16 layers, including 13 convolutional layers and 3 fully connected layers, VGG-16 is known for its simplicity and effectiveness in image classification tasks. Despite its relatively large number of parameters, VGG-16 achieves impressive performance on benchmark datasets such as ImageNet. VGG-16 employs a uniform architecture with small 3x3 convolutional filters and max-pooling layers, resulting in a deep yet straightforward network design. Its straightforward architecture and ease of implementation make VGG-16 a popular choice for transfer learning and feature extraction in various computer vision applications. While newer architectures have surpassed VGG-16 in terms of efficiency and performance

3.2.3 Functional Requirements

1. **Leaf Identification:** The system should accurately identify medical leaves based on their botanical features such as shape, size, texture, and color.

2. **Geolocation Integration:** Incorporate geolocation data to provide insights into the geographical distribution of identified plants, contributing to a comprehensive understanding of their natural habitats.

3. Age Estimation: Develop functionality to estimate the age of identified plants, providing valuable information on optimal harvesting times.

4. Medicinal Properties Overview: Provide a comprehensive overview of medicinal properties, curative capabilities, and potential side effects associated with the identified leaves.

5. User Interface: Design a user-friendly interface accessible to researchers, botanists, and enthusiasts, facilitating ease of use and understanding.

7. Sustainability Promotion: Promote environmentally conscious harvesting methods to contribute to biodiversity conservation and ecosystem health.

3.2.4 Non-Functional Requirements

1. **Accuracy:** Ensure high accuracy in leaf identification and disease detection to maintain the system's reliability and usefulness.

2. **Performance:** The system should perform efficiently, providing real-time results for leaf identification and disease detection.

3. Scalability: Design the system to handle a large volume of leaf data and user requests, allowing for scalability as the user base grows.

4. Security: Implement robust security measures to protect user data and prevent unauthorized access to the system.

5. Compatibility: Ensure compatibility with various devices and platforms, including web and mobile applications, to maximize accessibility for users.

6. Reliability: Maintain system reliability to ensure continuous availability and functionality, minimizing downtime and disruptions.

7. Usability: Prioritize usability in the user interface design, considering factors such as simplicity, intuitiveness, and accessibility for users of all skill levels.

8. Ethical Considerations: Adhere to ethical guidelines in the collection and use of data, respecting privacy rights and avoiding biases in the AI algorithms.

CHAPTER - 4

GANTT CHART

CHAPTER 4

GANTT CHANT

A Gantt chart is a type of bar chart, developed by Henry Gantt that illustrates a project schedule. Gantt charts illustrate the start and finish of the terminal elements and summary elements of the project. Terminal elements and summary elements comprise the work breakdown structure of the project.

The following is the Gantt chart of the project **"HARNESSING AI FOR PRECISE ESTIMATION OF MEDICAL LEAF CHARACTERISTICS ".**

Number	Task	Start	End	Duration (day s)
1	Synopsis	1-Oct-2023	19-Oct-2023	15
2	Presentation on idea	27-Sep-2023	27-Sep-2023	1
3	Software Requirement Specification	03-Oct-2023	13-Oct-2023	10
4	System Design	05-Nov-2023	15-Nov-2023	10
5	Implementation	05-Oct-2023	28-Feb-2024	154
6	Presentation on work progress	19-Dec-2024	19-Dec-2023	1
7	Testing	1-Mar-2024	10-Mar-2024	10
8	Result and Report	10-Mar-2024	31-mar-2024	12

Table 4.1-Gantt chart of planning and scheduling of project

Activity/Month	Sept 2023	Oct 2023	Nov 2023	Dec 2023	Jan 2024	Feb 2024	March 2024
Synopsis							
Presentation on idea							
SRS							
Design							
Implementation							
Testing							
Report							

Table 4.2- Gantt chart

A Gantt chart is a powerful tool used to visually represent the scheduling of tasks over time. Typically, it consists of a horizontal timeline divided into intervals, such as days, weeks, or months, and a vertical axis listing the tasks or activities involved in the project. Each task is represented by a bar, with its length indicating its duration, and its position on the timeline showing its start and end dates. Gantt charts provide a clear overview of project progress, showing which tasks are in progress, completed, or yet to be started. They also facilitate resource allocation and dependency management by highlighting task dependencies and critical paths. With the ability to easily identify potential bottlenecks and delays, Gantt charts help project managers make informed decisions and adjust schedules as needed to ensure project success. Overall, Gantt charts are invaluable tools for planning, tracking, and managing projects of all sizes and complexities, offering a visual roadmap for project execution from start to finish.

CHAPTER - 5

SYSTEM DESIGN

CHAPTER 5

SYSTEM DESIGN

System design is the process of designing the architecture, components, modules, interfaces, and data for a system to meet specific requirements. It involves determining the software and hardware infrastructure required for the system, as well as defining the relationships and interactions between the components of the system.

The purpose of system design is to create a blueprint that can guide the development of a system that meets the desired functionality, performance, scalability, security, and reliability requirements. It typically involves breaking down a complex system into smaller, more manageable components that can be developed and tested separately before being integrated into the larger system.

5.1 Architectural Diagram

An architecture diagram is a visual representation of the structure and components of a system or model. It provides a high-level overview of how the system or model is organized and how its different components interact with each other. Architecture diagrams are commonly used in the field of computer science, machine learning, and deep learning to illustrate the design and structure of complex systems or models. Architectural design can also involve considering nonfunctional requirements such as performance, reliability, scalability, and security, and determining the best way to meet those requirements.

The architecture diagram for harnessing AI in the precise estimation of medical leaf characteristics encompasses a systematic depiction of the system's components and their interactions. At its core lies a deep learning model tailored to analyze medical leaf images, which serves as the primary engine for characteristic estimation. Preprocessing modules are integrated into the architecture to enhance the quality of input data, including tasks such as noise reduction, image enhancement, and feature extraction. These preprocessed images are then fed into the deep learning model for analysis

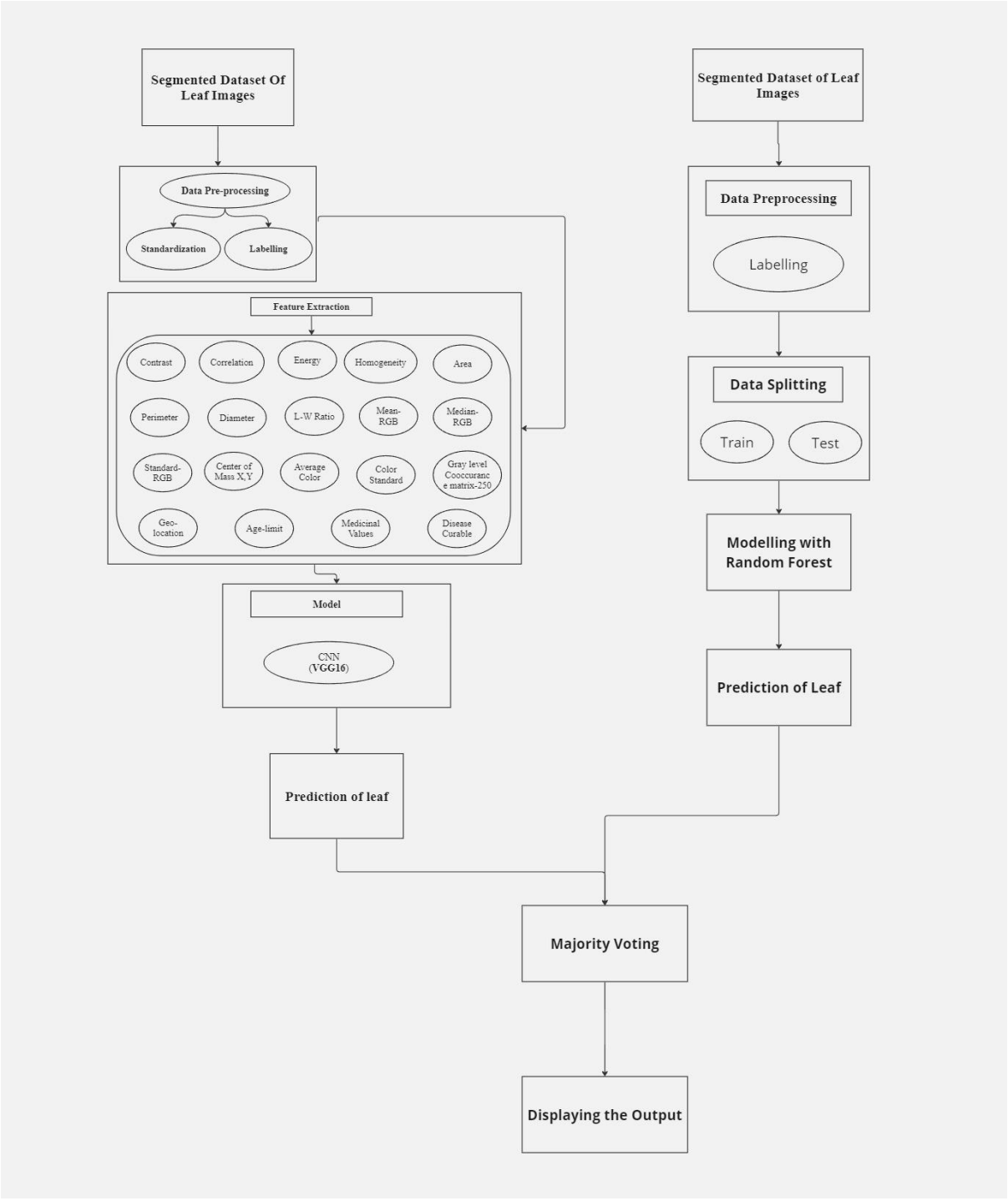


Figure 5.1 Architectural design for the system

5.2 Preparing Dataset of Medicinal Leaves

In this section, we outline the process of assembling the dataset used for training and evaluating the ensemble of machine learning and deep learning models aimed at predicting medicinal leaves. The dataset comprises a diverse collection of images sourced from various online platform that is mendely which include 30 species of leaves Each image in the dataset is meticulously labeled to denote its origin, we lay the foundation for robust model development and evaluation, facilitating the accurate prediction of the medicinal leaves.

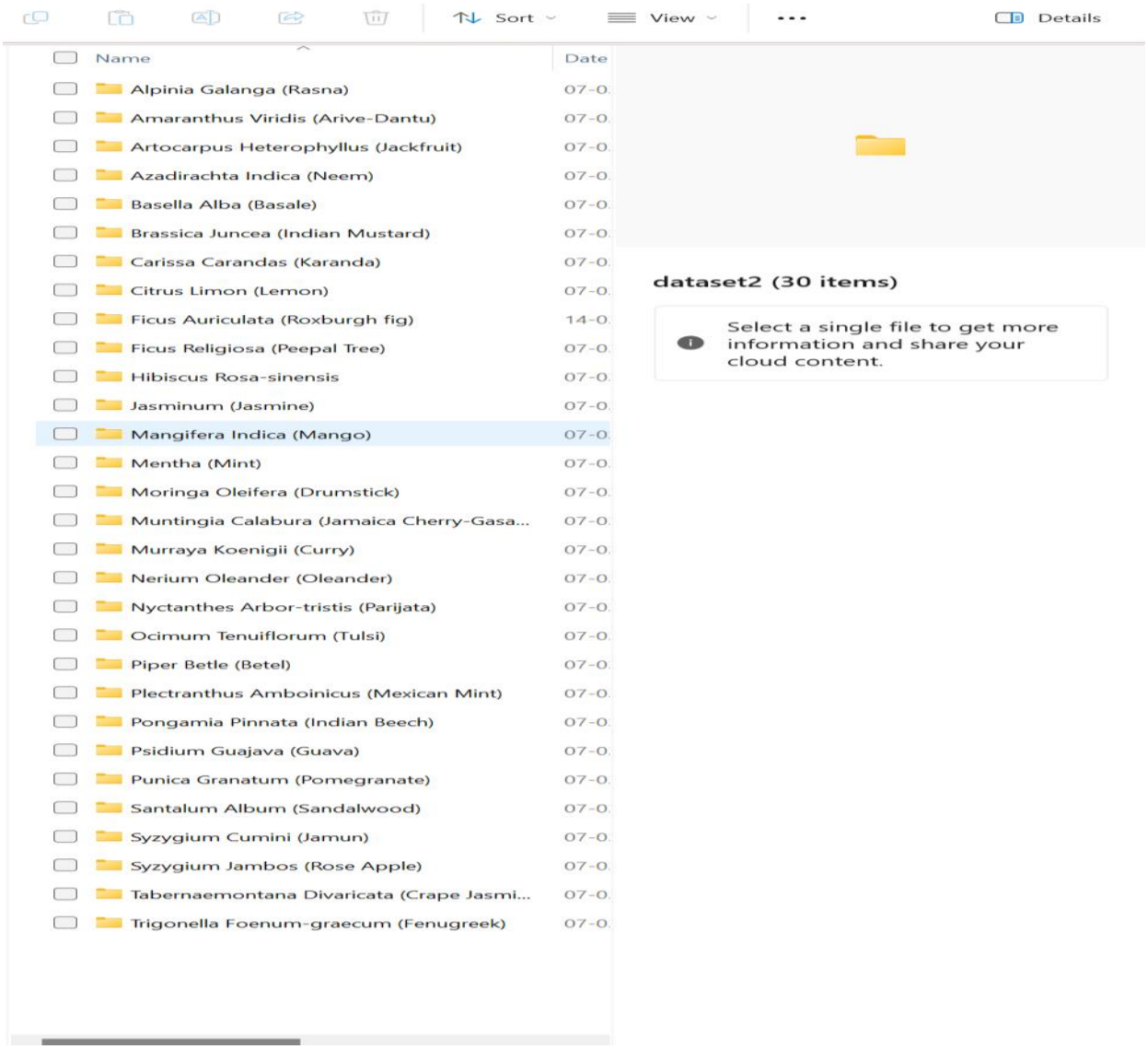


Figure 5.2-Dataset of the medicinal leaves consisting 30 species

5.3 Standardized Images of Dataset

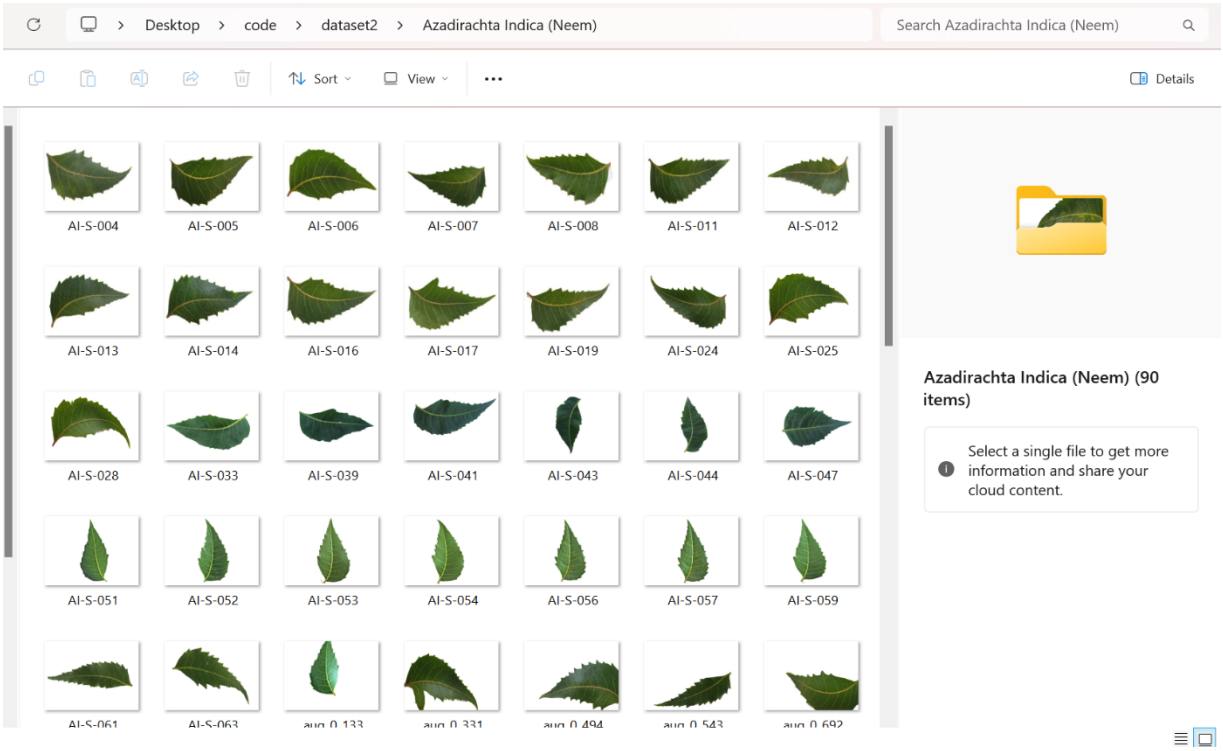


Figure-5.3 standardized images of Neem leaves

This figure demonstrates the standardized images of neem leaves from the dataset

5.4 Class Diagram

A class diagram is a type of diagram used in object-oriented modeling to represent the static structure and relationships between classes in a system or software application. It provides a visual representation of the classes, their attributes, methods, and relationships with other classes. A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity.

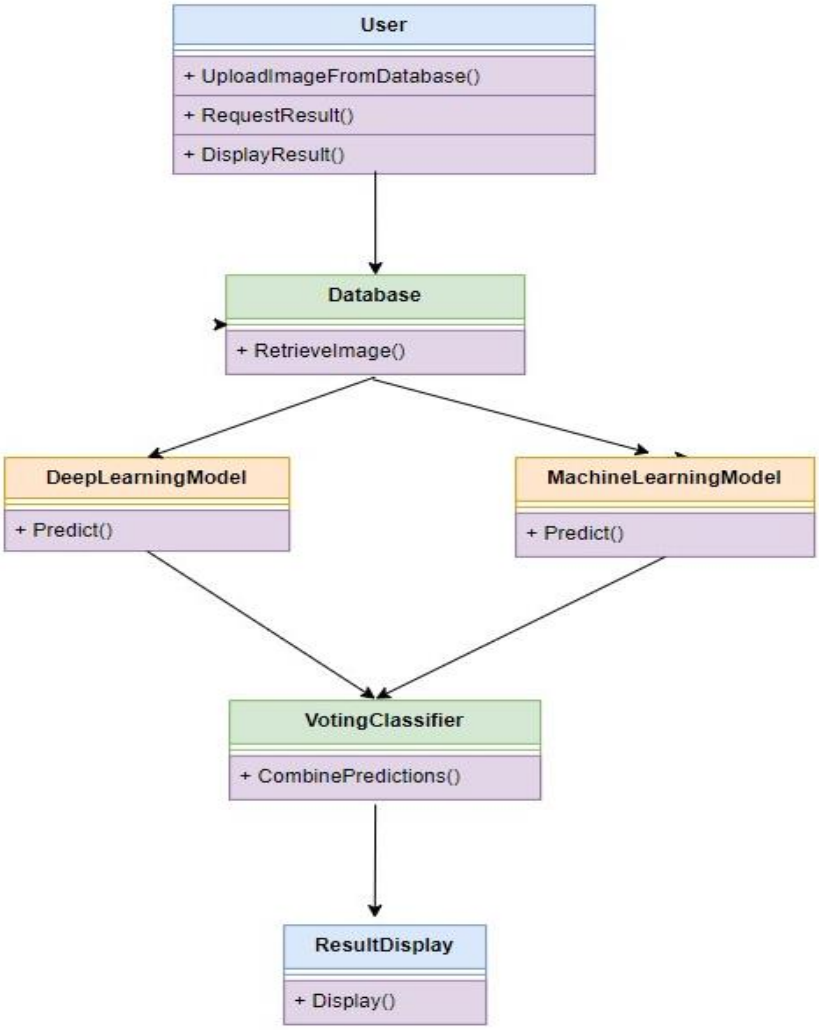


Figure 5.4 Class Diagram of proposed system

5.5 Data Flow Diagram

A Data Flow Diagram (DFD) is a graphical illustration of the flow of data throughout an information system, modelling its procedural aspects. A DFD is often used as a beginning step to create an outline of the system, which can later be elaborated.

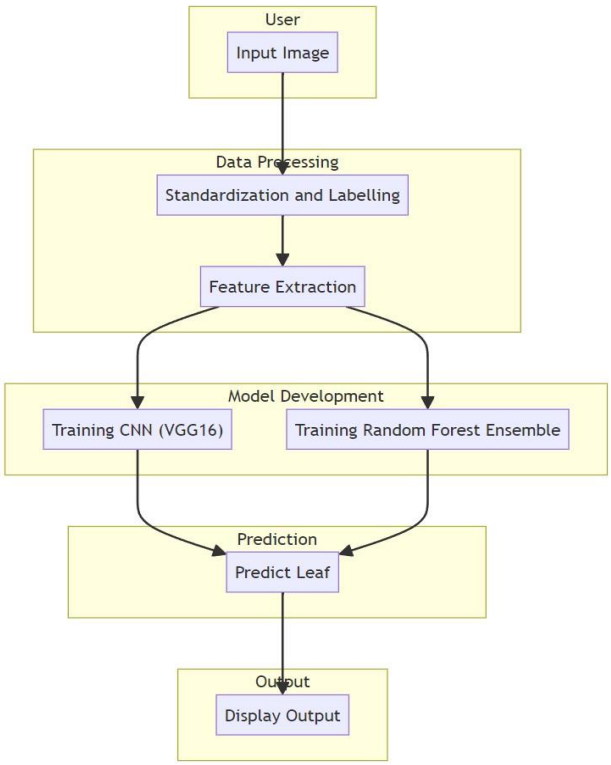


Figure 5.5 Dataflow Diagram

The fig. 5.5 depicts the process in which user inputs the images which undergoes data preprocessing and leads to the model development and finally predicts the leaf characteristics and display the output to the user.

5.6 Sequence Diagram

A sequence diagram consists of a group of objects that are represented by lifelines, and the messages that they exchange over time during the interaction. A sequence diagram shows the sequence of messages passed between objects. Sequence diagrams can also show the control structures between objects.

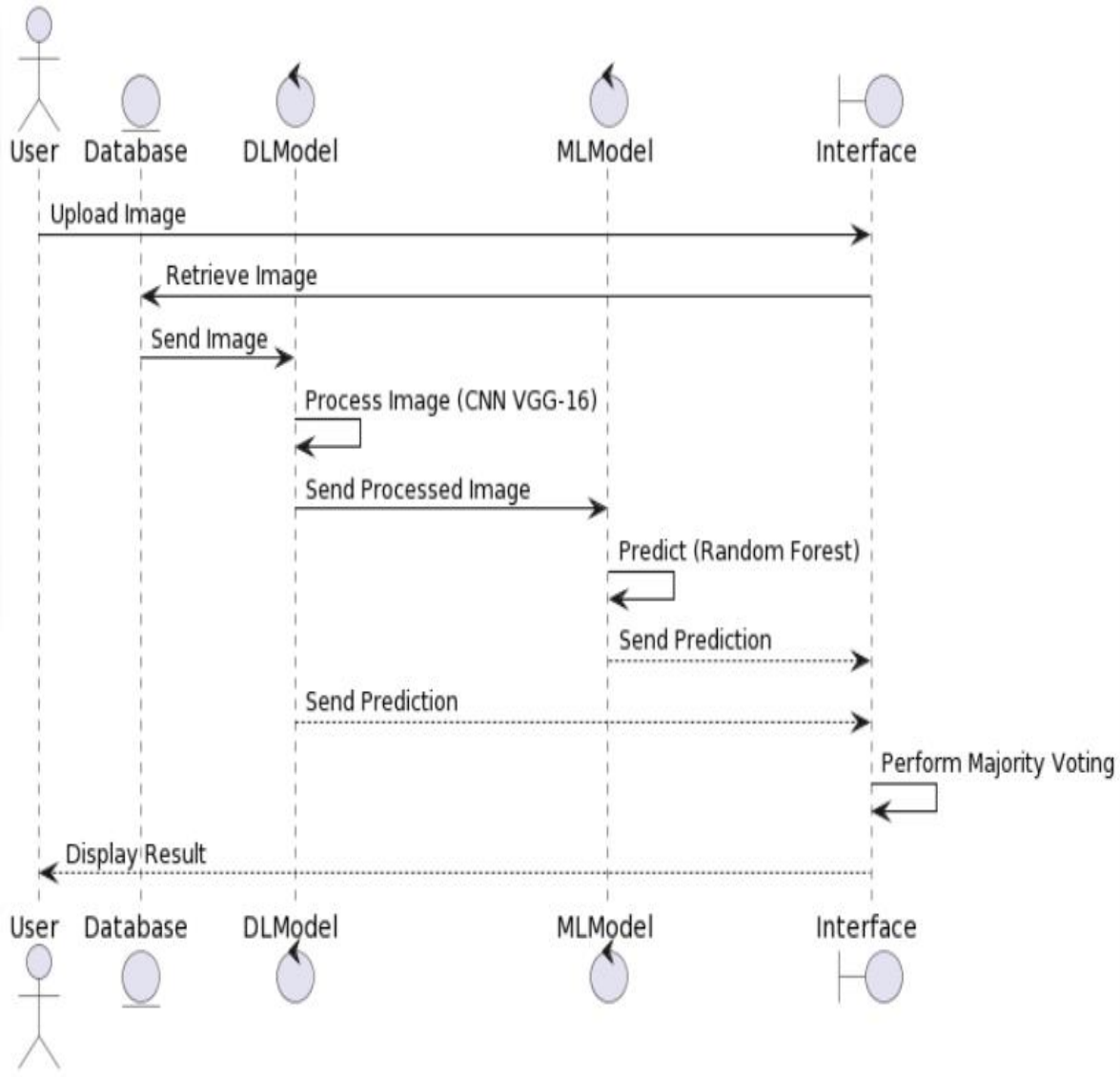


Figure 5.6 Sequence Diagram

The above Sequence diagram describes the system how the system works starting with user access to interface followed by leaf image upload, prediction and ends with display of output.

5.7 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams are intended to model both computational and organizational processes as well as the data flows intersecting with the related activities

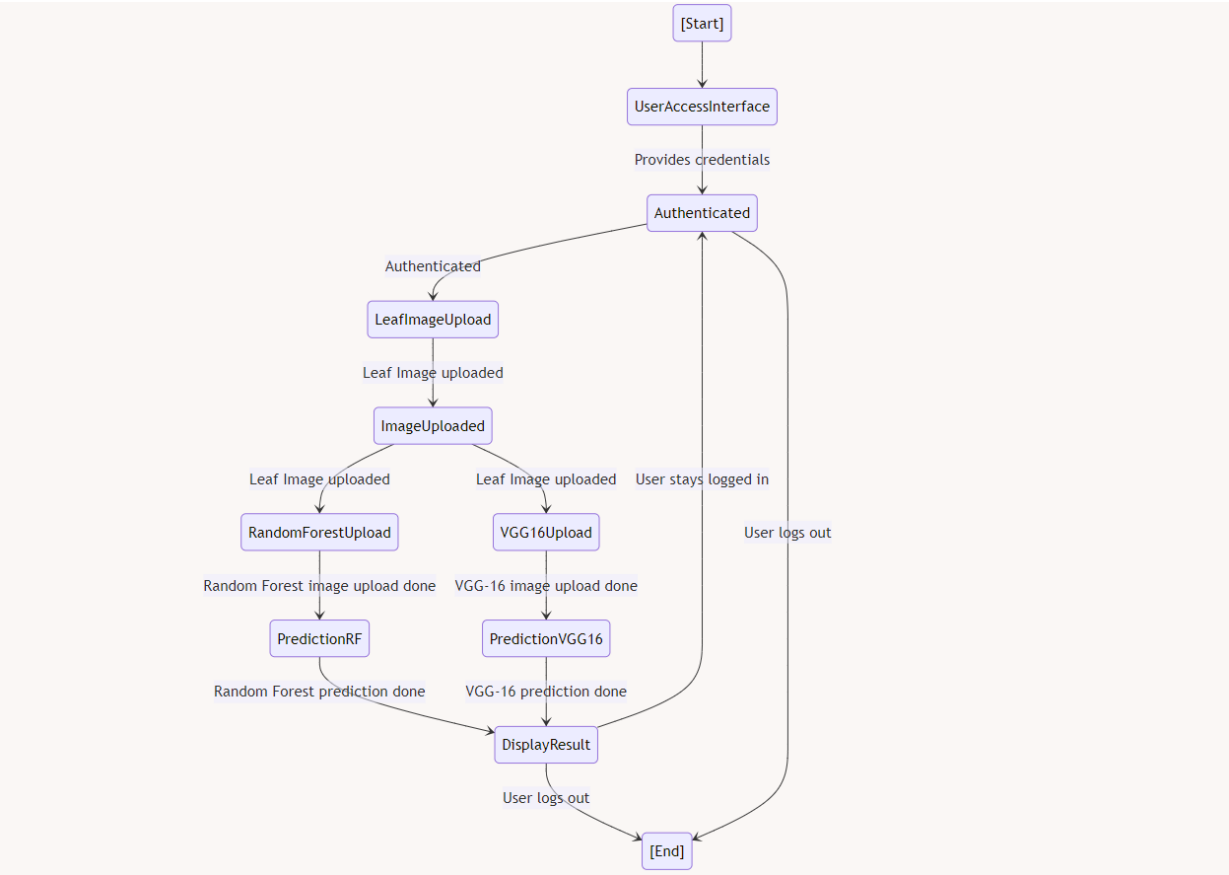


Figure 5.7 Activity Diagram

The above activity diagram describes the system how the system works starting with user access to interface followed by leaf image upload, prediction and ends with display of output.

5.8 Transfer learning

It is a machine learning technique where a model trained on one task is adapted or fine-tuned for use on a different, but related, task. Instead of starting the learning process from scratch, transfer learning leverages knowledge gained from solving one problem and applies it to a different, but related, problem.

Transfer learning offers several advantages:

1. Reduced Training Time: By leveraging pre-trained models, transfer learning reduces the amount of training time required for the new task.

2. Improved Performance: Transfer learning often leads to better performance on the new task, especially when the new task has limited training data.

3.Effective Use of Limited Data: Transfer learning allows you to make effective use of limited labeled data by transferring knowledge from a related task that has more data available.

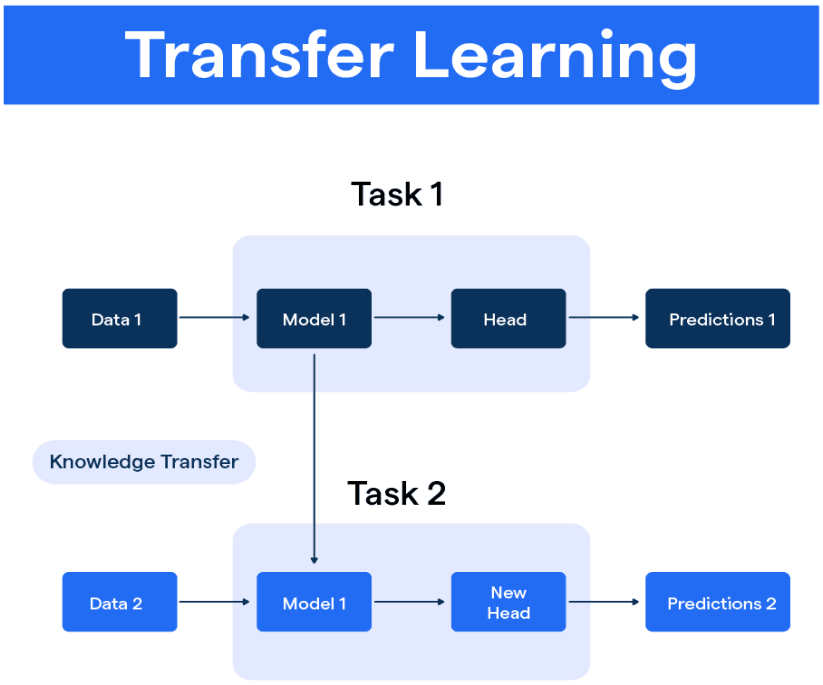


Figure 5.8 -Transfer learning Architecture

5.9 Deep Learning - Convolutional Neural Network

Deep learning is a subset of machine learning focused on training neural networks with multiple layers to learn complex patterns and representations from raw data. By leveraging hierarchical architectures and nonlinear activation functions, deep learning models can automatically extract and learn intricate features from high-dimensional input data.

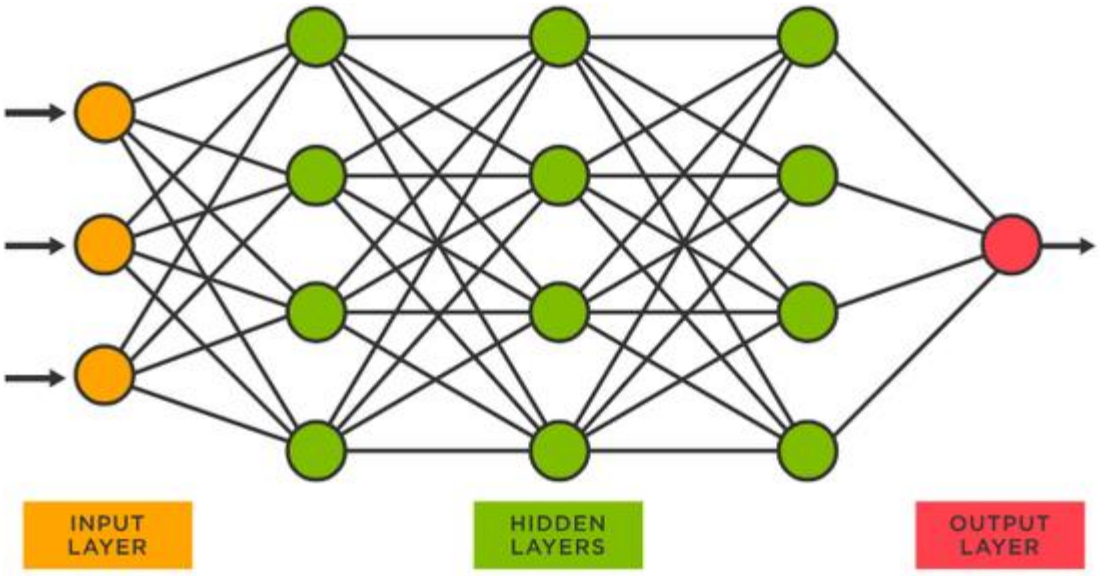


Figure 5.9 – CNN Architecture Input

A Convolutional Neural Network (CNN) typically consists of multiple layers, including input layers, hidden layers, and output layers. Here's a brief overview of each layer in a CNN:

1. Input Layer:

- The input layer is the initial layer of the CNN where the input data, such as images, is fed into the network.

2. Hidden Layers:

- Hidden layers are intermediate layers between the input and output layers, where the main computation of the CNN occurs.

3. Output Layer:

- The output layer is the final layer of the CNN, where the network produces its predictions or outputs.

5.10 VGG -16 (Visual Geometry Group)

VGG-16 is a deep convolutional neural network architecture proposed by the Visual Geometry Group (VGG) at the University of Oxford. It is characterized by its simplicity and uniformity, consisting of 16 layers, hence the name VGG-16. The architecture comprises a series of convolutional layers, followed by max-pooling layers for spatial downsampling. The network concludes with fully connected layers for classification.

VGG-16 remains a foundational architecture in the field of deep learning and continues to serve as a benchmark for evaluating new models and techniques.

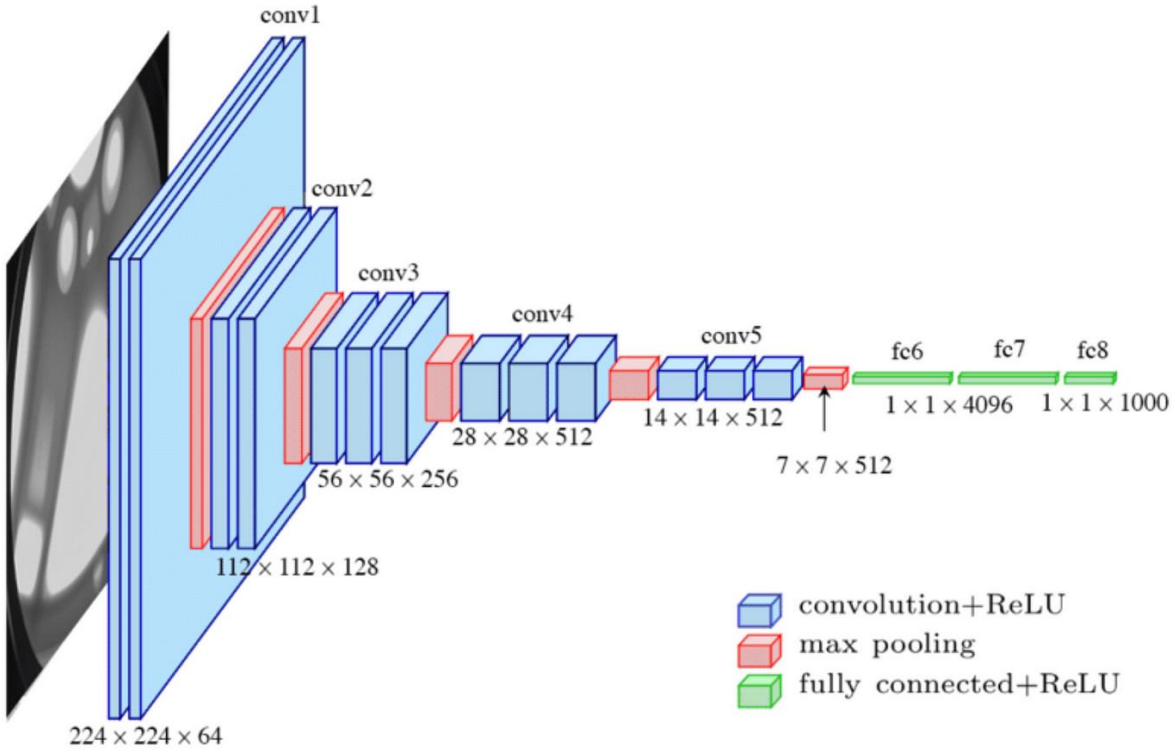


Figure 5.10 - VGG-16 Architecture

5.11 Machine Learning- Random Forest

Random Forest is a widely utilized machine learning algorithm recognized for its versatility and robust performance in both classification and regression tasks across diverse domains. Operating on the principle of ensemble learning, Random Forest constructs a multitude of decision trees during training, combining their predictions to yield a final output. Through the application of bootstrap aggregating (bagging), each decision tree within the Random Forest is trained on a distinct subset of the training data, fostering diversity and enhancing overall model resilience. Additionally, Random Forest employs random feature selection at each decision tree node,

mitigating correlation among trees and optimizing predictive accuracy. In classification tasks, predictions from individual trees are amalgamated via majority voting, while regression tasks

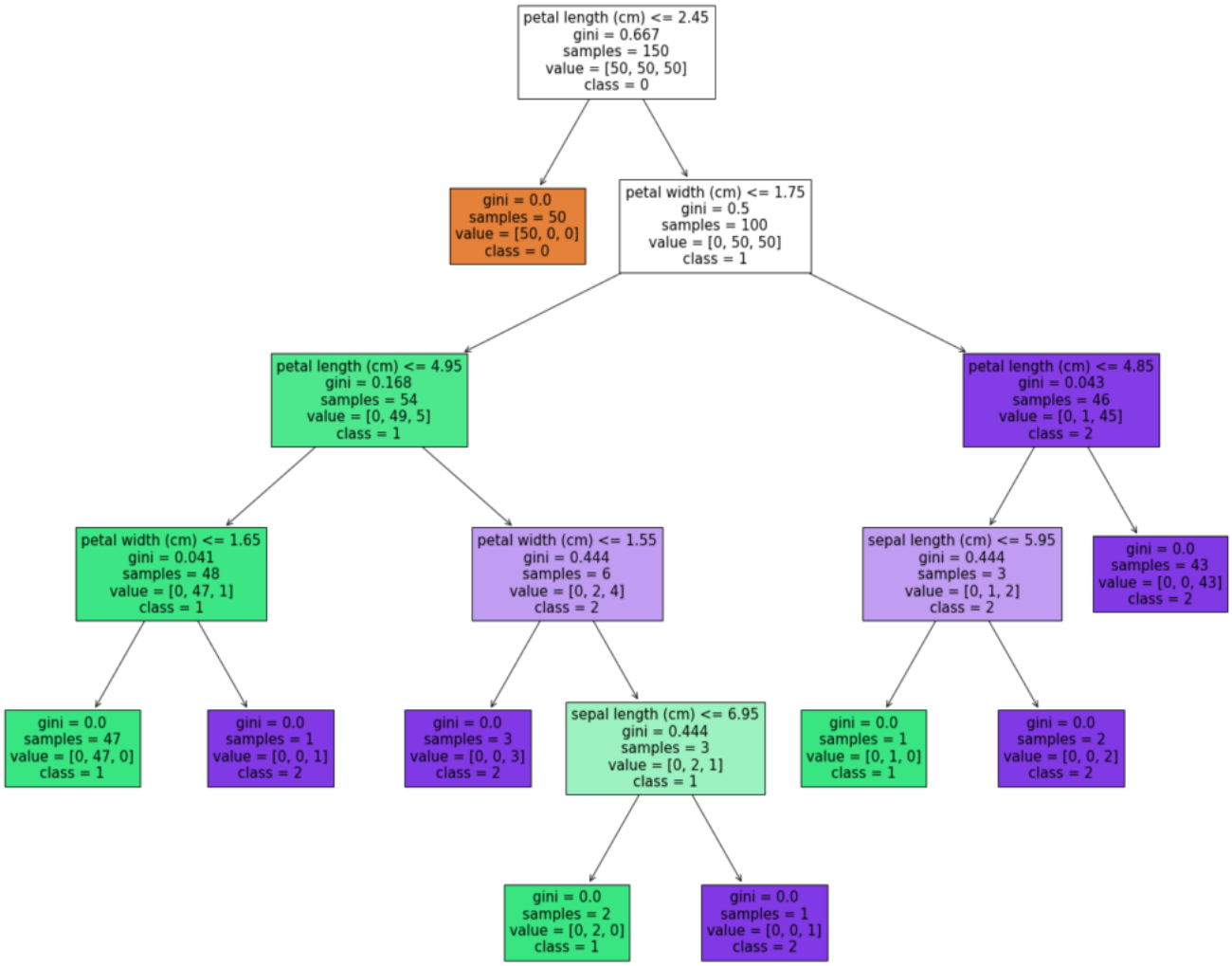


Figure 5.11 - Decision tree

A decision tree is a popular supervised learning algorithm used for classification and regression tasks in machine learning. It models decisions based on a tree-like structure, where each internal node represents a decision based on a feature, each branch represents the outcome of the decision, and each leaf node represents the final decision or prediction.

CHAPTER - 6

IMPLEMENTATION

CHAPTER 6

IMPLEMENTATION

Pseudocode for Standardization

1: Import the required libraries:

- os
- cv2
- numpy

2: Define the directory containing the dataset:

- dataset_dir = r'C:\Users\hp\Desktop\mAJOR PROJECT\Aryvedic\major project codes\major project final codes\dataset2'

3: Define the desired size for resizing images:

- desired_size = (224, 224)

4: Define a function preprocess_image(image_path) to preprocess images:

- Input: image_path (path to the image)
- Output: normalized_img (preprocessed image)
- Procedure:
  a. Read the image using cv2.imread
  b. Resize the image to the desired size using cv2.resize
  c. Normalize pixel values by dividing by 255.0
  d. Return the normalized image

5: Iterate through each image in the dataset directory using os.walk:

- For each directory, root, in the dataset directory:
  - For each file in the files list:
    a. Construct the full path to the image using os.path.join
    b. Preprocess the image using the preprocess_image function
    c. Save or overwrite the preprocessed image using cv2.imwrite, converting pixel values back to uint8 by multiplying with 255 and casting to np.uint8

Pseudocode Labelling code

1: Import the required libraries:

- os
- cv2
- numpy
- KMeans from sklearn.cluster
- csv

2: Define the directory containing the dataset:

- dataset_dir = r'C:\Users\hp\Desktop\mAJOR PROJECT\Aryvedic\major project codes\major project final codes\dataset2'

3: Create an empty list to store annotations:

- annotations = []

4: Iterate through each image in the dataset directory using os.walk:

- For each directory, root, in the dataset directory:
  - For each file in the files list:
    a. Construct the full path to the image using os.path.join
    b. Read the image using cv2.imread
    c. Resize the image to (224, 224) for consistency
    d. Flatten the image to use as a feature vector
    e. Perform K-Means clustering with n_clusters=5
    f. Assign the cluster label as the category
    g. Append the annotation (image file path and category) to the annotations list

5: Define the path for the CSV file:

- csv_file_path = 'labels.csv'

6: Write annotations to CSV file:

a. Open the CSV file in write mode with newline=''
b. Create a CSV DictWriter with fieldnames 'image_file' and 'label'
c. Write the header to the CSV file
d. Iterate through annotations:
  i. Write each annotation to the CSV file
e. Close the CSV file

7: Print a message indicating completion and the path to the saved CSV file.

Pseudocode for Feature extraction

1: Import the required libraries:

- cv2
- csv
- os
- numpy
- feature from skimage

2: Define a function extract_features(image_path, label, filename) to extract features from an image:

- Input: image_path (path to the image), label (label for the image), filename (name of the image file)
- Output: features (list of extracted features)
- Procedure:

a. Read the leaf image using cv2.imread

b. Convert the image to grayscale using cv2.cvtColor

c. Calculate texture features using feature.graycomatrix and feature.graycoprops

d. Calculate shape features using cv2.findContours, cv2.contourArea, cv2.arcLength, and cv2.boundingRect

e. Calculate color features including mean, median, and standard deviation of each channel

f. Combine all features into a list

g. Return the list of features

3: Define a function process_images(root_directory, output_csv) to process images in a directory and write features to a CSV file:

- Input: root_directory (root directory containing the images), output_csv (path to the output CSV file)
- Procedure:

a. Open the CSV file for writing with newline=''

b. Create a CSV writer object

c. Write the header to the CSV file

d. Initialize a label_counter to keep track of labels

e. Traverse through all subdirectories and files using os.walk

f. For each file with extension ".jpg":

i. Construct the full image path using os.path.join

ii. Extract features from the image using the extract_features function

iii. Write the features to the CSV file

iv. Increment the label_counter

Pseudocode for dataset preparation with augmentation

1: Import the required libraries:

- os
- shutil
- numpy as np
- tqdm
- tensorflow as tf
- ImageDataGenerator from tensorflow.keras.preprocessing.image

2: Define constants:

- base_dir: directory containing original dataset
- target_dir: directory to store augmented dataset
- number_of_combinations: number of augmented images to generate per original image

3: Create target directory for augmented dataset if it does not exist already.

4: Initialize an ImageDataGenerator for data augmentation with desired parameters.

5: Loop over each subdirectory in the base directory:

a. Get the list of files in the subdirectory.

b. Shuffle the list of files.

c. Split the files into two sets: original and augmented.

d. Create a corresponding subdirectory in the target directory.

e. Copy original files to the target directory.

f. For each augmented file:

i. Load the image using tf.keras.preprocessing.image.load_img.

ii. Convert the image to array format.

iii. Reshape the array to have a batch dimension.

iv. Generate augmented images using datagen.flow with specified parameters.

v. Save the augmented images to the target directory.

vi. Repeat until the specified number of augmented images per original image is reached.

6: Print a message indicating completion.

Pseudocode for train_vgg_dataset2

1: Import the required libraries:

- os
- matplotlib.pyplot as plt
- seaborn as sns
- numpy as np
- pandas as pd
- tensorflow as tf
- ImageDataGenerator from tensorflow.keras.preprocessing.image
- VGG16 from tensorflow.keras.applications
- Model, Flatten, Dense, Dropout from tensorflow.keras.layers
- classification_report, confusion_matrix from sklearn.metrics

2: Define parameters:

- dataset_dir: directory containing the dataset
- batch_size: batch size for training
- IMG_HEIGHT, IMG_WIDTH: image dimensions
- epochs: number of epochs for training

3: Initialize ImageDataGenerator for data augmentation with desired parameters.

4: Create train_data_gen and validation_data_gen using flow_from_directory method of ImageDataGenerator.

5: Load the VGG16 network with local weights.

6: Construct the head of the model:

a. Define the headModel layers including Flatten, Dense, and Dropout.

b. Connect the headModel to the baseModel.

7: Freeze the layers in the base model.

8: Compile the model with optimizer, loss function, and metrics.

9: Train the model using the fit method, specifying train_data_gen and validation_data_gen.

10: Save the trained model.

11: Plot training history:

a. Plot Training and Validation Accuracy vs. Epochs.

b. Plot Training and Validation Loss vs. Epochs.

12: Evaluate the model on the validation set.

13: Generate predictions using the trained model.

14: Generate classification report and confusion matrix.

15: Plot the confusion matrix as a heatmap.

16: Plot training metrics:

a. Plot Training and Validation Accuracy vs. Epochs.

b. Plot Training and Validation Loss vs. Epochs.

Pseudocode for train_randomforest_ensemble

1: Import the required libraries:

- os
- numpy as np
- tqdm
- image from tensorflow.keras.preprocessing
- VGG16 from tensorflow.keras.applications.vgg16
- Model from tensorflow.keras.models
- RandomForestClassifier from sklearn.ensemble
- train_test_split from sklearn.model_selection
- accuracy_score from sklearn.metrics
- joblib

2: Load the VGG16 model with pre-trained ImageNet weights.

3: Use the VGG16 model without the top layers.

4: Define a function extract_features(img_path) to extract features using VGG16:

- Input: img_path (path to the image)
- Output: features (extracted features)
- Procedure:

a. Load the image using image.load_img with target size (224, 224).

b. Convert the image to array format.

c. Expand the dimensions of the array.

d. Preprocess the input.

e. Extract features using vgg_model.predict.

f. Flatten the features.

g. Return the flattened features.

5: Define the dataset directory.

6: Extract features and labels from the dataset:

a. Initialize empty lists X and y.

b. Iterate over each subdirectory in the dataset directory:

i. Iterate over each image in the subdirectory:

- Extract features using extract_features function.
- Append the features to X.
- Append the corresponding label (subdirectory name) to y.

7: Convert lists X and y to numpy arrays.

8: Split the dataset into training and testing sets using train_test_split.

9: Train a Random Forest classifier:

- Initialize a RandomForestClassifier with desired parameters.
- Fit the classifier on the training data.

10: Evaluate the Random Forest classifier:

- Predict labels for the test set.
- Calculate accuracy using accuracy_score.
- Print the accuracy.

11: Save the trained Random Forest model using joblib.dump.

Pseudocode for prediction

1: Import the required libraries:

- os
- numpy as np
- image and preprocess_input from tensorflow.keras.preprocessing
- VGG16, Model from tensorflow.keras.applications.vgg16
- RandomForestClassifier from sklearn.ensemble
- joblib

2: Load the VGG16 model with pre-trained ImageNet weights, excluding the top layers.

- Use VGG16(weights='imagenet', include_top=False) to load the model.
- Create a Model instance to access intermediate layers up to 'block5_conv2'.

3: Load the pre-trained Random Forest classifier using joblib.load.

4: Define a function extract_features(img_path) to extract features using VGG16:

- Input: img_path (path to the image)
- Output: features (extracted features)
- Procedure:

a. Load the image using image.load_img with target size (224, 224).

b. Convert the image to array format.

c. Expand the dimensions of the array.

d. Preprocess the input.

e. Extract features using model.predict.

f. Flatten the features.

g. Return the flattened features.

5: Define the dataset directory containing subfolders with images.

6: Loop through each subfolder (class) in the dataset directory:

a. Concatenate the class directory path using os.path.join.

b. Check if the current path is a directory using os.path.isdir.

c. If it's not a directory, skip to the next iteration.

d. Iterate through each image file in the subfolder:

i. Concatenate the image file path using os.path.join.

ii. Extract features from the image using extract_features function.

iii. Reshape the extracted features to match the input format of the Random Forest classifier.

iv. Predict the class label using rf_classifier.predict.

v. Display the prediction along with the image filename and class name.

Pseudocode for interface

1. Import necessary libraries:

- os
- streamlit
- numpy
- tensorflow
- joblib
- PIL

- io

2. Load the pre-trained VGG16 model without top layers and the Random Forest classifier.
3. Define a function to extract features using VGG16:

   function extract_features(image_bytes):

      Load the image from image_bytes.

      Preprocess the image for VGG16 model.

      Extract features from the image using the VGG16 model.

      Return the flattened feature vector.
4. Define a function to predict using Random Forest:

   function predict_rf(image_bytes):

      Extract features from the image using VGG16.

      Predict the class label using the Random Forest classifier.

      Return the predicted class label.
5. Define a function to load and preprocess the image for VGG16:

   function load_and_prep_image(image, img_shape=224):

      Decode and resize the image.

      Rescale the image values between 0 and 1.

      Return the preprocessed image.
6. Define a dictionary containing additional data for each class (if available).
7. Create the Streamlit application interface:

   - Set the application title and header.

   - Add a file uploader to allow users to upload an image.

   - Read the uploaded image and display it on the interface.

   - Check if the uploaded image exists in the dataset directory.

   - If the image exists:

      - Predict the class label using both Random Forest and VGG16 models.

      - Display the predicted class label and additional data (if available).

   - If the image doesn't exist:

      - Notify the user that the image is not recognized.
8. Run the Streamlit application.

Pseudocode for streamlit theme

BEGIN Streamlit Theme Configuration
SET primaryColor to "#5db560"
SET backgroundColor to "#160303"
SET secondaryBackgroundColor to "#7f9fde"
SET textColor to "#f3f3f3"
SET font to "serif"
END Streamlit Theme Configuration

CHAPTER - 7

TESTING

CHAPTER 7

TESTING

Testing is the process of evaluating and verifying that a software product or application does what it is supposed to do which is an important phase in the development life cycle of the product. During the testing, the program to be tested was executed with a set of test cases and the output of the program for the test cases was evaluated to determine whether the program is performing as expected. Errors were found and corrected by using the following testing steps and correction was recorded for future references. Thus, a series of testing was performed on the system before it was ready for implementation. An important point is that software testing should be distinguished from the separate discipline of Software Quality Assurance (SQA), which encompasses all business process areas, not just testing.

7.1 Testing Levels

Testing is part of Verification and Validation. Testing plays a very critical role for quality assurance and for ensuring the reliability of the software.

The objective of testing can be stated in the following ways.

- A successful test is one that uncovers as-yet-undiscovered bugs.
- A better test case has high probability of finding un-noticed bugs.
- A pessimistic approach of running the software with the intent of finding errors.

Testing can be performed in various levels like unit test, integration test and system test.

7.1.1 Unit Testing

Unit testing tests the individual components to ensure that they operate correctly. Each component is tested independently, without other system component. This system was tested with the set of proper test data for each module and the results were checked with the expected output. Unit testing focuses on verification effort on the smallest unit of the software design module. Unit testing is typically performed by the developer. Initially the user tries to login with credentials, if the credential is valid the user successfully logs in. Otherwise, user receives the error message. User can then successfully check the state of onion by clicking the picture of onion.

7.1.2 Integration Testing

Integration testing is another aspect of testing that is generally done in order to uncover errors associated with the flow of data across interfaces. The unit-tested modules are grouped together

and tested in small segment, which makes it easier to isolate and correct errors. This approach is continued until we have integrated all modules to form the system as a whole. After scanning the onion the user has to click on predict button of the application. After pressing the button, the custom model will classify the image into one of the three categories and displays the class name on the screen of the application.

7.1.3 System Testing

System testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behaviour of a component or a system when it is tested. System testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

7.1.4 Acceptance Testing

Acceptance testing is a method of software testing where a system is tested for acceptability. The major aim of this test is to evaluate the compliance of the system with the business requirements and assess whether it is acceptable for delivery or not. After testing the system at different levels, how the complete system is accepted which meets all the mentioned functional and non-functional requirements. Depending upon the organization, acceptance testing might take the form of beta testing, field testing or end-user testing.

7.2 Test Cases

In software engineering, a test case is a specification of the inputs, execution conditions, testing procedure, and expected results that define a single test to be executed to achieve a particular software testing objective, such as to exercise a particular program path or to verify compliance with a specific requirement. Test cases underlie testing that is methodical rather than haphazard. A battery of test cases can be built to produce the desired coverage of the software being tested. Formally defined test cases allow the same tests to be run repeatedly against successive versions of the software, allowing for effective and consistent regression testing.

Individual PASS/FAIL criteria are written for each test case. All the tests need to get a PASS result for proper working of an application.

Test Number	Test Case Id	Test Case	Expected results	Actual output	Status
1	UT_1	Open medicinal leaf Prediction Application	User should be able to launch application successfully	As Expected result	Pass
2	UT_2	Select Leaf image from gallery	User can select the leaf image from the dataset	As Expected result	Pass
3	UT_3	Predict the name of the medicinal leaf and its characteristics using deep learning model	Successfully predict the medicinal leaves and its characteristics	As Expected result	Pass
4	UT_4	Predict the name of the medicinal leaf and its characteristics using machine learning model	Successfully predict the medicinal leaves and its characteristics	As Expected result	Pass
5	UT_5	Predict the medicinal leaves and its characteristics using majority voting combining deep learning and machine learning	Successfully predict the medicinal leaves and its characteristics based on majority voting	As Expected result	Pass
6	UT_6	Predict the medicinal leaves and display the result	Successfully predict the medicinal leaves and its characteristics.	As Expected result	Pass
7	UT_7	User can upload random image other than trained image	Displays the message “Data not found “	As Expected result	Pass

Table 7.1-Test cases

CHAPTER - 8

RESULTS AND SNAPSHOTS

CHAPTER 8

RESULTS & SNAPSHOTS

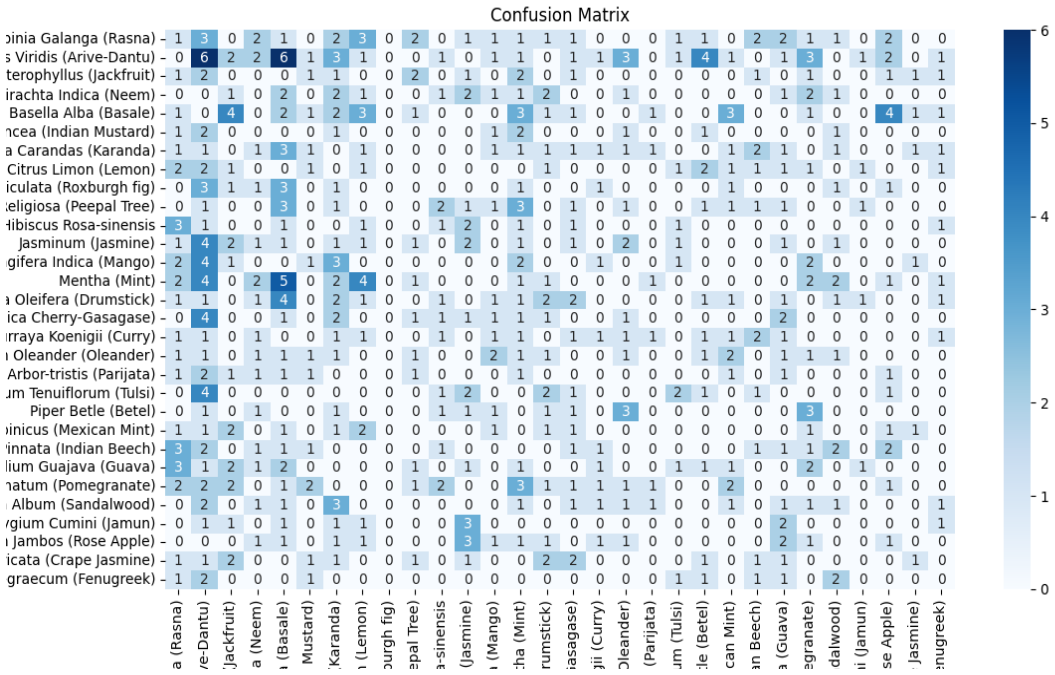


Figure 8.1 Confusion Matrix of Deep Learning Model

The provided confusion matrix is indeed used to evaluate the performance of a machine learning model in classifying 30 different classes of plants. Each row represents an actual plant type, while each column represents the model's predictions. The numbers within the matrix indicate the frequency of correct classifications (diagonal) and incorrect classifications (off-diagonal). This matrix helps in assessing the model's accuracy and identifying any patterns of misclassification across the different plant types.

.

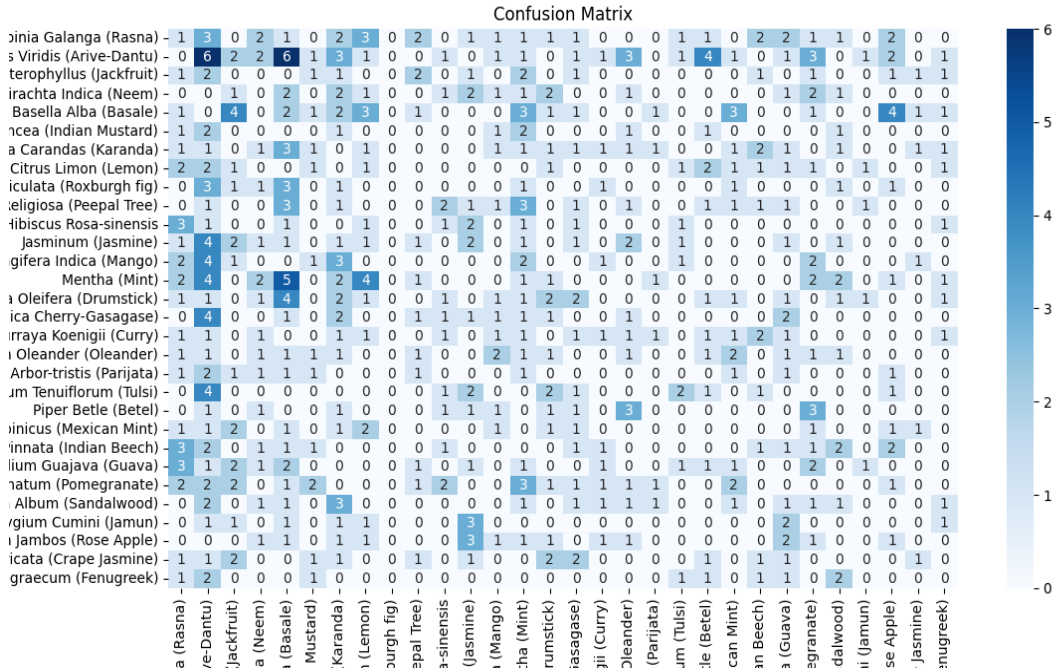
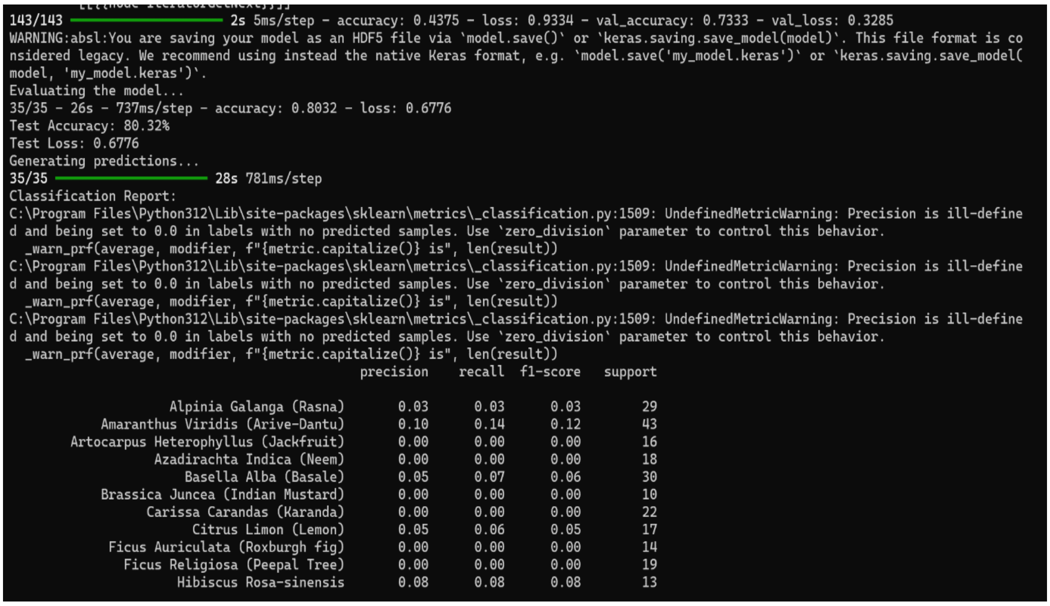


Figure 8.2- Confusion Matrix of Machine Learning Model

The provided confusion matrix is indeed used to evaluate the performance of a machine learning model in classifying 30 different classes of plants. Each row represents an actual plant type, while each column represents the model's predictions. The numbers within the matrix indicate the frequency of correct classifications (diagonal) and incorrect classifications (off-diagonal). This matrix helps in assessing the model's accuracy and identifying any patterns of misclassification across the different plant types.

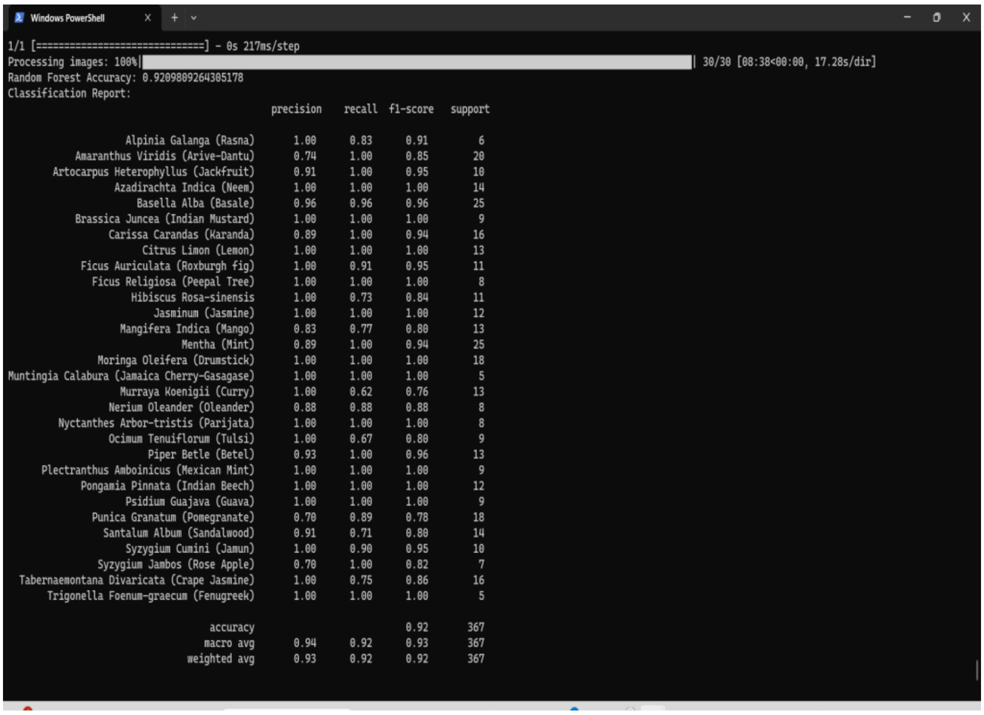


```
143/143 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.4375 - loss: 0.9334 - val_accuracy: 0.7333 - val_loss: 0.3285
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is co
nsidered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(
model, 'my_model.keras')`.
Evaluating the model...
35/35 - 26s - 737ms/step - accuracy: 0.8032 - loss: 0.6776
Test Accuracy: 80.32%
Test Loss: 0.6776
Generating predictions...
35/35 ━━━━━━━━━━━━━━━━━━━━ 28s 781ms/step
Classification Report:
C:\Program Files\Python312\Lib\site-packages\sklearn\metrics\_classification.py:1509: UndefinedMetricWarning: Precision is ill-define
d and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Program Files\Python312\Lib\site-packages\sklearn\metrics\_classification.py:1509: UndefinedMetricWarning: Precision is ill-define
d and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Program Files\Python312\Lib\site-packages\sklearn\metrics\_classification.py:1509: UndefinedMetricWarning: Precision is ill-define
d and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
                                          precision    recall  f1-score   support

              Alpinia Galanga (Rasna)       0.03      0.03      0.03        29
         Amaranthus Viridis (Arive-Dantu)   0.10      0.14      0.12        43
    Artocarpus Heterophyllus (Jackfruit)    0.00      0.00      0.00        16
           Azadirachta Indica (Neem)        0.00      0.00      0.00        18
               Basella Alba (Basale)        0.05      0.07      0.06        30
      Brassica Juncea (Indian Mustard)      0.00      0.00      0.00        10
           Carissa Carandas (Karanda)       0.00      0.00      0.00        22
                 Citrus Limon (Lemon)       0.05      0.06      0.05        17
       Ficus Auriculata (Roxburgh fig)      0.00      0.00      0.00        14
         Ficus Religiosa (Peepal Tree)      0.00      0.00      0.00        19
              Hibiscus Rosa-sinensis        0.08      0.08      0.08        13
```

Figure 8.3- Classification report of Deep Learning Model

Demonstrates program output includes training accuracy (0.8032) and loss (0.6776) metrics on a test dataset, followed by the generation of predictions on another dataset.



```
Windows PowerShell
1/1 [==============================] - 0s 217ms/step
Processing images: 100%|████████████████████| 30/30 [08:38<00:00, 17.28s/dir]
Random Forest Accuracy: 0.9209809264305178
Classification Report:
                                          precision    recall  f1-score   support

              Alpinia Galanga (Rasna)       1.00      0.83      0.91         6
     Amaranthus Viridis (Arive-Dantu)       0.74      1.00      0.85        20
 Artocarpus Heterophyllus (Jackfruit)       0.91      1.00      0.95        10
            Azadirachta Indica (Neem)       1.00      1.00      1.00        14
                Basella Alba (Basale)       0.96      0.96      0.96        25
     Brassica Juncea (Indian Mustard)       1.00      1.00      1.00         9
           Carissa Carandas (Karanda)       0.89      1.00      0.94        16
                 Citrus Limon (Lemon)       1.00      1.00      1.00        13
      Ficus Auriculata (Roxburgh fig)       1.00      0.91      0.95        11
        Ficus Religiosa (Peepal Tree)       1.00      1.00      1.00         8
               Hibiscus Rosa-sinensis       1.00      0.73      0.84        11
                   Jasminum (Jasmine)       1.00      1.00      1.00        12
              Mangifera Indica (Mango)      0.83      0.77      0.80        13
                        Mentha (Mint)       0.89      1.00      0.94        25
         Moringa Oleifera (Drumstick)       1.00      1.00      1.00        18
Muntingia Calabura (Jamaica Cherry-Gasagase) 1.00     1.00      1.00         5
             Murraya Koenigii (Curry)       1.00      0.62      0.76        13
            Nerium Oleander (Oleander)      0.88      0.88      0.88         8
     Nyctanthes Arbor-tristis (Parijata)    1.00      1.00      1.00         8
           Ocimum Tenuiflorum (Tulsi)       1.00      0.67      0.80         9
                  Piper Betle (Betel)       0.93      1.00      0.96        13
   Plectranthus Amboinicus (Mexican Mint)   1.00      1.00      1.00         9
       Pongamia Pinnata (Indian Beech)      1.00      1.00      1.00        12
              Psidium Guajava (Guava)       1.00      1.00      1.00         9
        Punica Granatum (Pomegranate)       0.70      0.89      0.78        18
           Santalum Album (Sandalwood)      0.91      0.71      0.80        14
              Syzygium Cumini (Jamun)       1.00      0.90      0.95        10
         Syzygium Jambos (Rose Apple)       0.70      1.00      0.82         7
Tabernaemontana Divaricata (Crape Jasmine)  1.00      0.75      0.86        16
   Trigonella Foenum-graecum (Fenugreek)    1.00      1.00      1.00         5

                             accuracy                           0.92       367
                            macro avg       0.94      0.92      0.93       367
                         weighted avg       0.93      0.92      0.92       367
```

Figure 8.4- Classification report of Machine Learning Model

The command line window output indicates that the image processing program has completed its task, having classified a total of 367 images. It reports a classification accuracy of 92%, implying that 92% of the images were correctly classified by the algorithm.

Figure 8.5- Training and Validation Accuracy and Loss Graph of deep Learning model

The image depicts two graphs likely visualizing the performance of a machine learning model during training. The x-axis represents epochs, which are iterations where the model is exposed to the training data. The left y-axis shows accuracy, while the right y-axis shows loss.

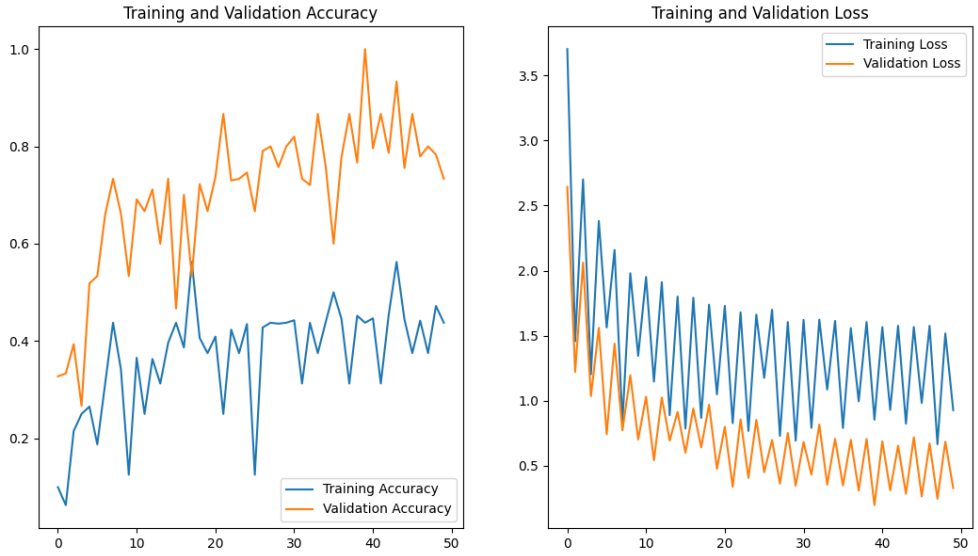


Figure 8.6- Training and Validation Accuracy and Loss Graph of deep Learning model

Demonstrates The two graphs displayed in the image depict the training and validation performance of a machine learning model over multiple epochs. The x-axis represents the epochs, or iterations of the training process, while the y-axis on the left side indicates accuracy, and the right side indicates loss.

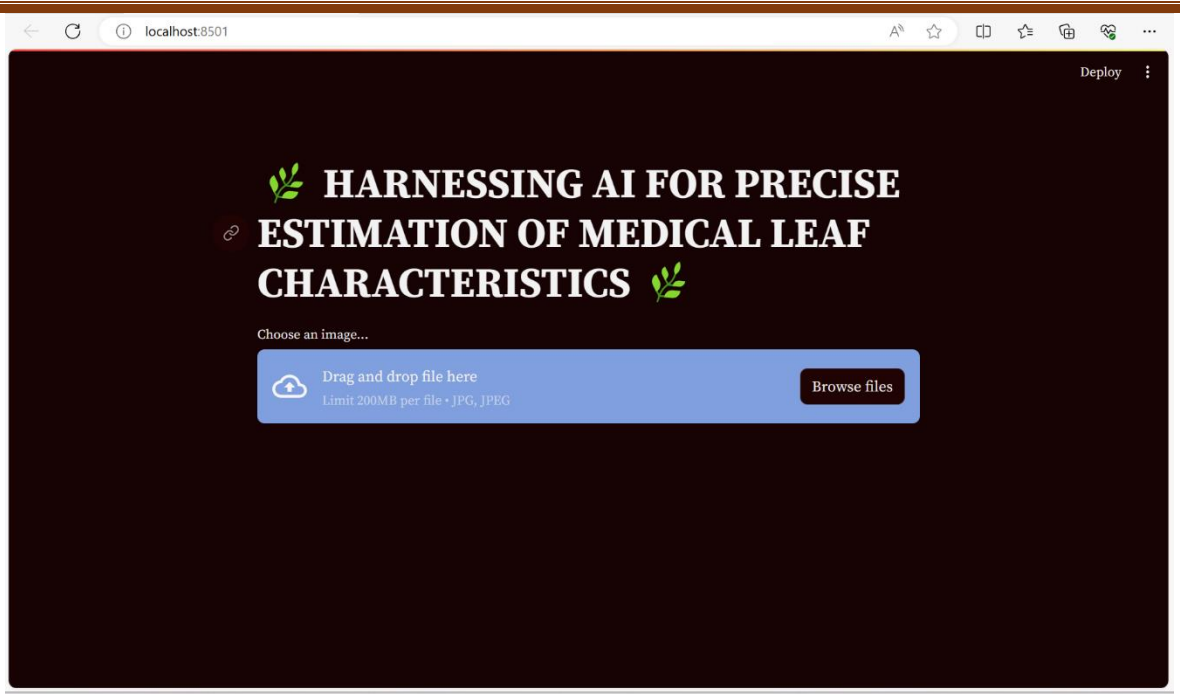


Figure 8.7- Home page of the proposed system

Demonstrates the front page of the project where it gives an option for the user to browse the files were they can select the image of the leaves from the dataset.

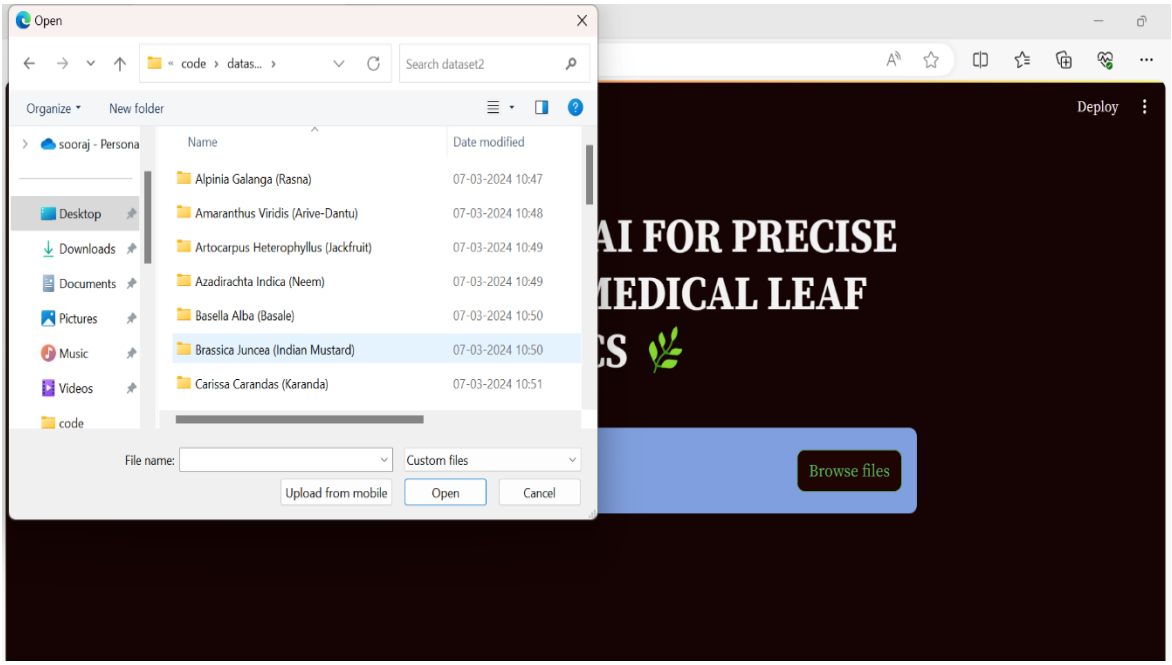


Figure 8.8-Input of leaves images from the Dataset

Demonstrates where the user can select the images of the medicinal leaves from the dataset.

Figure 8.9- User input image of neem leaf.

This figure demonstrates the user input of the images of the leaf Azadirachta Indica(Neem)

Figure 8.10- Prediction of medicinal leaf based on majority voting

This figure demonstrates the uploaded image is identified as Azadirachta Indica (Neem) through majority voting. It exhibits medicinal properties including antibacterial, antifungal, and antiviral effects, commonly used in traditional medicine. Geolocation data suggests its origin in India, with reported curative potential for skin diseases, malaria, and diabetes. Additionally, it generally safe for adults and children over 2, but caution is advised, especially during pregnancy and for young children, due to potential side effects such as allergic reactions and stomach upset.

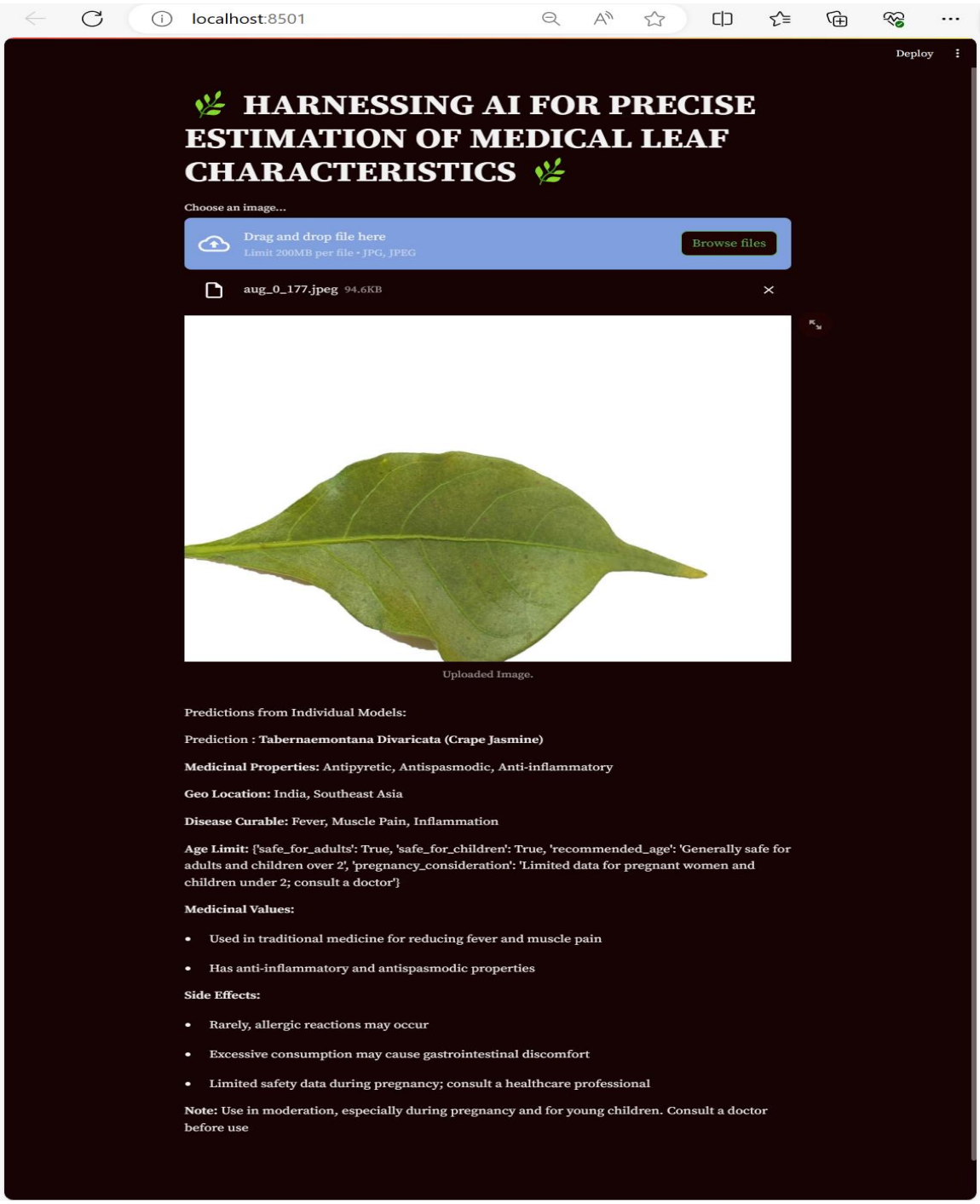


Figure 8.11- Prediction of medicinal leaf based on individual prediction

The Individual model predicts the uploaded image to be Tabernaemontana Divaricata (Crape Jasmine). It is associated with medicinal properties such as antipyretic, antispasmodic, and anti-inflammatory effects, commonly used in traditional medicine in India and Southeast Asia. Reported curative potential includes fever reduction and muscle pain relief, along with anti-inflammatory and antispasmodic benefits. However, caution is advised due to potential effects such as allergic reactions and gastrointestinal discomfort especially during pregnancy and for young children. Consultation with a healthcare is recommended

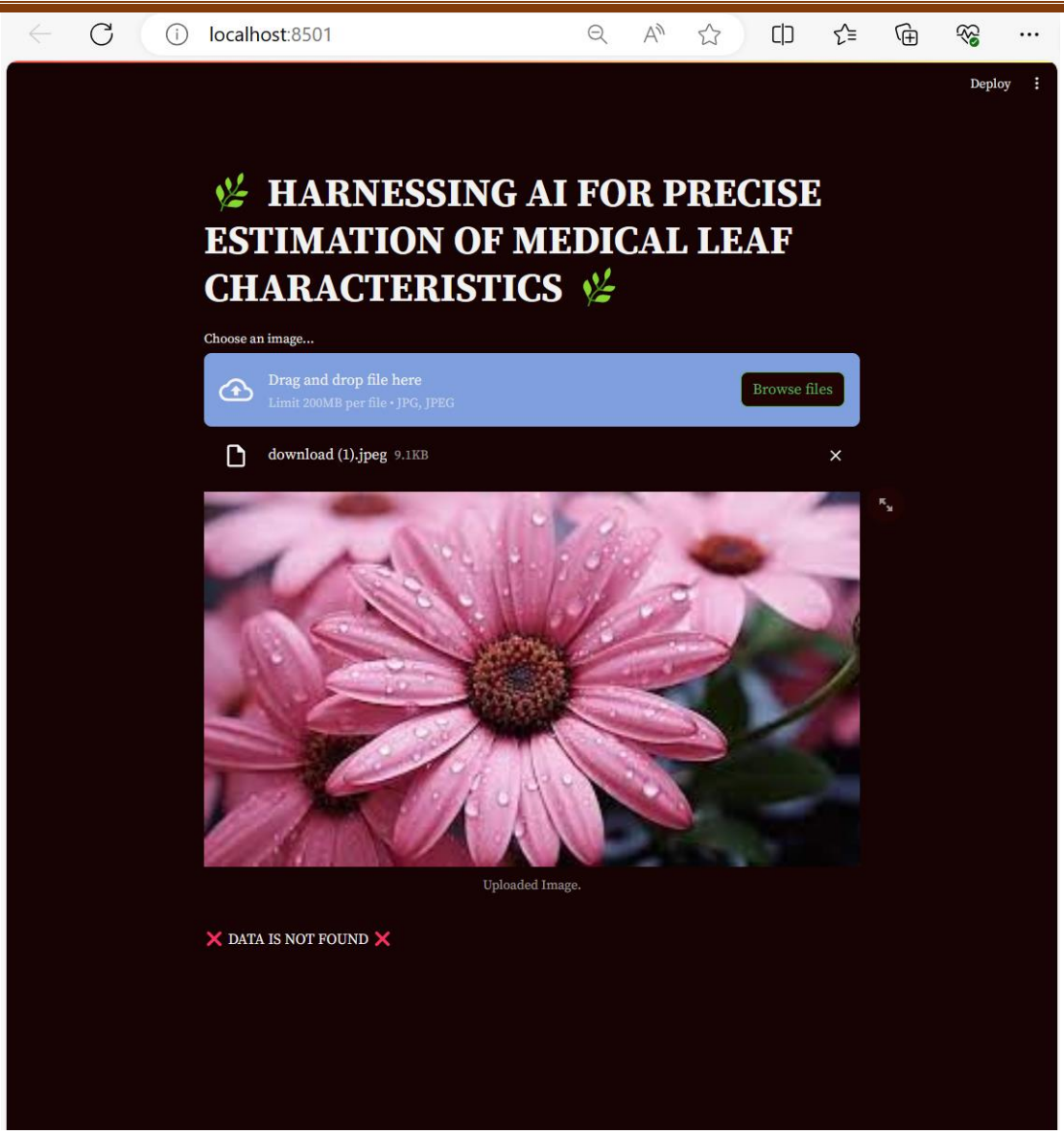


Figure 8.12- Input of image outside the dataset

Demonstrates that the uploaded image could not be matched with available data

CHAPTER - 9

CONCLUSION AND FUTUREWORK

CHAPTER 9

CONCLUSION & FUTURE WORK

The project "Harnessing AI for Precise Estimation of Medicinal Leaf Characteristics" marks a significant advancement in the field of medicinal plant identification by integrating cutting-edge AI technologies such as machine learning and deep learning. Through the development of sophisticated AI models, the project has demonstrated remarkable accuracy in identifying medicinal plant species based on their leaf characteristics, thereby facilitating safer and more efficient utilization of herbal remedies. The user-friendly interface provided by Streamlit ensures accessibility for researchers, botanists, and enthusiasts, promoting widespread use and understanding.

In conclusion, the project has successfully addressed the primary objectives of developing accurate AI models for medicinal plant identification, promoting sustainability in harvesting practices, mitigating health risks associated with misidentification, and integrating traditional botanical wisdom with modern technology. However, there is still ample scope for future work. One avenue for further exploration could involve expanding the dataset to encompass a wider variety of medicinal plants and their respective characteristics, thereby enhancing the robustness and versatility of the AI models. Additionally, continual refinement and optimization of the AI algorithms can further improve accuracy and efficiency, ensuring the reliability of the system across diverse plant species and environmental conditions. Moreover, collaborative efforts with experts in ethnobotany and traditional medicine can enrich the system with valuable insights and contribute to its broader acceptance and utility in real-world settings. Overall, the project lays a strong foundation for ongoing research and innovation in the field of medicinal plant identification, with promising implications for healthcare, conservation, and sustainable development.

References

[1] Naeem, S.; Ali, A.; Chesneau, C.; Tahir, M.H.; Jamal, F.; Sherwani, R.A.K.; Ul Hassan, M. The classification of medicinal plant leaves based on multispectral and texture feature using machine learning approach. Agronomy 2021.

[2] Automatic Recognition of Medicinal Plants using Machine Learning Techniques, International Journal of Advanced Computer Science and Applications 2021

[3] Mukherjee, G.; Tudu, B.; Chatterjee, A. A convolutional neural networkdriven computer vision system toward identification of species and maturity stage of medicinal leaves: Case studies with Neem, Tulsi and Kalmegh leaves. Soft Computer. 2021.

[4] Bisen, D. Deep convolutional neural network based plant species recognition through features of leaf. Multimed. Tools Appl. 2021.

[5] Azadnia, R.; Kheiralipour, K. Recognition of leaves of different medicinal plant species using a robust image processing algorithm and artificial neural networks classifier. J. Appl. Res. Med. Aromat. Plants 2021.

[6] Russel, N.S.; Selvaraj, A. Leaf species and disease classification using multiscale parallel deep CNN architecture. Neural Comput. Appl. 2022.

[7] Paulson, A.; Ravishankar, S. AI Based Indigenous Medicinal Plant Identification. In Proceedings of the 2020 Advanced Computing and Communication Technologies for High Performance Applications (ACCT HPA), Cochin, India, 2–4 July 2020.

[8] Muneer, A.; Fati, S.M. Efficient and automated herbs classification approach based on shape and texture features using deep learning. IEEE Access 2020.

[9] Reddy, S.R.; Varma, G.P.; Davuluri, R.L. Optimized convolutional neural network model for plant species identification from leaf images using computer vision. Int. J. Speech Technol. 2021.

[10]Roopashree, S.; Anitha, J. DeepHerb: A Vision Based System for Medicinal Plants Using Xception Features. IEEE Access 2021.

[11]"Automatic recognition of medicinal plants using machine learning techniques" by A. Kumar and D.B. Kumar (2020)

[12]"Segmentation and Identification of Medicinal Plant through Weighted KNN, Multimedia Tools and Applications" by S. Patil and M. Sasikala (2022)

[13]"Real-time identification of medicinal plants using machine learning techniques" by A. Kumar and D.B. Kumar (2020)

[14] "Automatic recognition of medicinal plants using machine learning techniques" by Sivaranjani (2020)

[15] "Comparison of machine learning algorithms for detection of medicinal plants" by L.D.S. Pacifico (2020)

[16] "Ayur-PlantNet: an unbiased lightweight deep convolutional neural network for Indian Ayurvedic plant species classification" by B.R. Pushpa and N.S. Rani (2023)

[17] "Leaf species and disease classification using multiscale parallel deep CNN architecture" by N.S. Russel and A. Selvaraj (2022)

[18] "AyurLeaf: a deep learning approach for the classification of medicinal plants, IEEE Region 10 Annual International Conference" by M.R. Dileep and P.N. Pournami (2020)

[19] "Kernel-based PSO and FRVM: an automatic plant leaf type detection using texture, shape, and color features" by VijayaLakshmi and V. Mohan (2020)

[20] "Deep ensemble learning for automatic medicinal leaf identification" by S. Sachar and A. Kumar (2022)

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

A monthly Peer-reviewed journal

Indexed by Google Scholar, Crossref and Mendeley

DOI IJARCCE/10.17148

www.ijarcce.com

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 3, March 2024

DOI: 10.17148/IJARCCE.2024.13329

Harnessing AI For Precise Estimation of Medical Leaf Characteristics

Sanjay S M[1], Praviksha[2], Sooraj S Bhat[3], Neha B S[4], Ms. Radha E G[5]

Student, Dept. of Artificial Intelligence & Machine Learning, Mangalore Institute of Technology & Engineering, Moodabidre, India[1,2,3,4]

Professor, Dept. of Artificial Intelligence & Machine Learning, Mangalore Institute of Technology & Engineering, Moodabidre, India[5]

Abstract: This project presents a novel approach for accurately classifying medicinal plant species based on leaf characteristics, utilizing advanced artificial intelligence (AI) techniques. By integrating deep learning models and classic machine learning algorithms, the system offers precise estimations of medicinal plants from leaf images. Users can conveniently upload images through an intuitive web interface, enabling the system to predict the corresponding plant species promptly. The primary focus of this project is to streamline the process of identifying medicinal plants, addressing the challenges associated with manual classification methods. Traditional approaches often entail significant time and effort, leading to potential errors and inconsistencies in classification outcomes. In contrast, our system leverages the power of AI to automate and enhance the classification process, ensuring accurate and reliable results. The core component of the system is a deep learning model, utilized for feature extraction from medicinal plant leaf images. These extracted features serve as input to both a classic machine learning classifier and the deep learning model itself, facilitating robust classification of plant species based on their unique leaf characteristics. Upon image upload, the system swiftly processes the images, extracting relevant features and predicting the corresponding plant species. Additionally, users receive supplementary information about the predicted plant species, including medicinal properties, geographical distribution, and potential applications. By harnessing AI technologies, this project aims to democratize access to accurate medicinal plant classification, benefiting various stakeholders such as healthcare professionals, researchers, and individuals interested in herbal medicine. Moreover, the system empowers users with informed decision-making capabilities regarding the utilization of medicinal plants for various health conditions.

Keywords: Medicinal plants classification,Artificial intelligence,Deep learning,Herbal medicine ,Image-based analysis,Plant species identification,Machine learning,Healthcare applications.

I. INTRODUCTION

Medicinal plants have long been esteemed for their nutritional and medicinal qualities, attributed to bioactive compounds such as antioxidants, anti-allergic, anti-inflammatory, and antibacterial agents. Found in various forms, ranging from trees to shrubs and herbs, their distribution is influenced by environmental adaptations. Approximately 14–28% of plant species exhibit medicinal properties, with significant reliance observed in rural populations of developing countries and a growing interest in developed nations due to concerns over adverse effects of chemical drugs. Despite their multifaceted utility in food, beverages, and cosmetics, global distribution is hindered by counterfeit and substandard products, posing risks to consumers.

Traditionally, botanists identify medicinal plants manually, a process that is both arduous and time-consuming. Leaf morphology, a key identifier, presents challenges due to similarities among species and variations in colour and texture. Despite advancements in feature extraction, vision-based systems still encounter difficulties in accurately classifying medicinal herbs.

This study aims to address this gap by developing a real-time automatic vision system using a proposed deep learning algorithm and machine vision techniques for medicinal plant identification. Responding to the increasing popularity of medicinal plants in artisanal and industrial sectors, the research seeks to enhance automated identification methods to meet evolving demands.

This revised abstract emphasizes the significance of medicinal plants, highlights challenges in their identification, and outlines the research objective of advancing automatic identification methods to meet growing industrial and artisanal needs.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 3, March 2024

DOI: 10.17148/IJARCCE.2024.13329

II. LITERATURE SURVEY

In [1]Naseem et al. (2021) proposed a method called “Classification of medicinal plants based on diversity and machine learning models.” The CNN model achieved an overall accuracy of over 99.3%, demonstrating the potential of deep learning in revolutionary medicinal plant identification. This advancement has influenced traditional medicine, energy conservation, and educational technology.

In [2]International Journal of Advanced Computer Science and Applications published a study on "Automatic identification of medicinal plants using machine learning". This research demonstrates the effectiveness of machine learning in automated harvesting through the processing of beautiful leaves and tissues, as well as the use of herbs, thrifts and medicines.

In [3]Mukherjee et al. (2021) introduced a neural network-driven computer vision system to determine page type and evolution. The system, called APRS, uses image recognition and GIS technology to locate medicinal plants. The average facility identification accuracy of APRS is 98.2%, and the average location selection accuracy is 97.8%.
In [4]Bisen (2023) proposed a method using deep neural networks to identify plants by leaf ladder. Combining multispectral and textural features for classification of medicinal plants.

III. SCOPE AND METHODOLOGY

Aim of the project
The Aim of this project encompasses the development of a comprehensive system for the accurate classification and analysis of medicinal plant species based on leaf characteristics. The methodology involves integrating advanced artificial intelligence (AI) techniques, including deep learning and classic machine learning algorithms, to facilitate precise estimations of medicinal plant classes from leaf images

Existing system
The existing system for medicinal plant classification relies on manual taxonomical methods or basic image analysis techniques, often lacking scalability and accuracy. These methods are labor-intensive and error-prone, hindering accessibility. Advanced AI-based systems are needed to improve accuracy, scalability, and usability, facilitating efficient medicinal plant classification and research.
Proposed system

The proposed system for medicinal plant classification aims to overcome the limitations of existing methods by leveraging advanced artificial intelligence (AI) techniques and expanding the dataset availability. The system will utilize state-of-the-art deep learning models, such as convolutional neural networks (CNNs), to extract intricate features from leaf images, leading to improved accuracy and robustness in classification. Additionally, efforts will be made to collect and curate a comprehensive dataset encompassing a wide range of medicinal plant species, thereby enhancing the system's generalization capabilities and applicability across diverse botanical contexts. Through these advancements, the proposed system seeks to provide a more reliable, scalable, and accessible solution for medicinal plant classification, facilitating advancements in herbal medicine research and practice.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 3, March 2024

DOI: 10.17148/IJARCCE.2024.13329

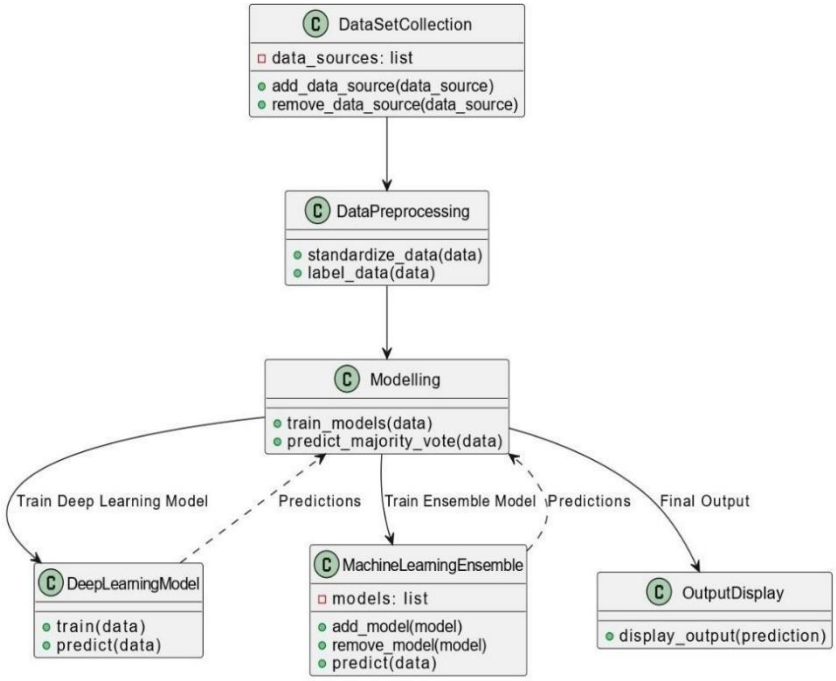


Fig 1.Proposed system

System Architecture

The architecture diagram depicts a streamlined process for automated leaf description generation. It begins with inputting leaf images, followed by pre-processing to standardize them. Features such as area, color, and vein structure are extracted. A machine learning model analyzes these features to predict the leaf description, including geolocation, species, age, and disease status, enabling efficient plant monitoring and classification.

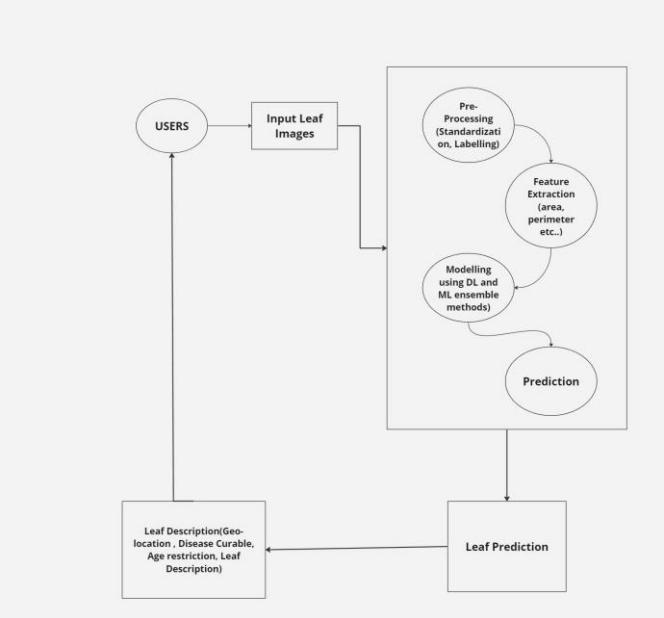


Fig 2.System Architecture

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 3, March 2024
DOI: 10.17148/IJARCCE.2024.13329

IV. CONCLUSIONS

In conclusion, the project "Harnessing AI for Precise Estimation of Medicinal Leaf Characteristics" signifies a significant advancement in the field of medicinal plant identification, achieved through the integration of state-of-the-art AI technologies such as machine learning and deep learning. Utilizing sophisticated AI models, the project has demonstrated remarkable accuracy in discerning medicinal plant species based on their leaf characteristics, promising safer and more efficient utilization of herbal remedies. The user-friendly Streamlit interface ensures accessibility for researchers, botanists, and enthusiasts, facilitating broad adoption and understanding of these innovative techniques in medicinal plant identification.

REFERENCES

[1. . Naeem, S.; Ali, A.; Chesneau, C.; Tahir, M.H.; Jamal, F.; Sherwani, R.A.K.; Ul Hassan, M. The classification of medicinal plant leaves based on multispectral and texture feature using machine learning approach. Agronomy 2021.
[2. 2.Automatic Recognition of Medicinal Plants using Machine Learning Techniques, International Journal of Advanced Computer Science and Applications 2021
[3. 3. . Mukherjee, G.; Tudu, B.; Chatterjee, A. A convolutional neural network-driven computer vision system toward identification of species and maturity stage of medicinal leaves: Case studies with Neem, Tulsi and Kalmegh leaves. Soft Computer. 2021.
[4. 4. . Bisen, D. Deep convolutional neural network based plant species recognition through features of leaf. Multimed. Tools Appl. 2021.
[5. 5. Azadnia, R.; Kheiralipour, K. Recognition of leaves of different medicinal plant species using a robust i mage processing algorithm and artificial neural networks classifier. J. Appl. Res. Med. Aromat. Plants 2021.
[6. 6. Russel, N.S.; Selvaraj, A. Leaf species and disease classification using multiscale parallel deep CNN architecture. Neural Comput. Appl. 2022.
[7. 7. Paulson, A.; Ravishankar, S. AI Based Indigenous Medicinal Plant Identification. In Proceedings of the 2020 Advanced Computing and Communication Technologies for High Performance Applications (ACCT HPA), Cochin, India, 2–4 July 2020.
8. 8. Muneer, A.; Fati, S.M. Efficient and automated herbs classification approach based on shape and texture features using deep learning. IEEE Access 2020.
[9. 9. Reddy, S.R.; Varma, G.P.; Davuluri, R.L. Optimized convolutional neural network model for plant species identification from leaf images using computer vision. Int. J. Speech Technol. 2021.
[10. 10. Roopashree, S.; Anitha, J. DeepHerb: A Vision Based System for Medicinal Plants Using Xception Features. IEEE Access 202