An Efficient AI Approach to Identify and Diagnosis Sugarcane Leaf Diseases.

Shariful Islam and Sumaya Akter

A Thesis in the Partial Fulfillment of the Requirements

for the Award of Bachelor of Computer Science and Engineering (BCSE)

Department of Computer Science and Engineering
College of Engineering and Technology
IUBAT – International University of Business Agriculture and Technology

Fall 2023

An Efficient AI Approach to Identify and Diagnosis Sugarcane Leaf Diseases.

Shariful Islam and Sumaya Akter

A Thesis in the Partial Fulfillment of the Requirements for the Award of Bachelor of Computer Science and Engineering (BCSE)

The thesis has been examined and approved,

_____

Prof. Dr. Utpal Kanti Das
Chairman

_____

Dr. Muhammad Hasibur Rashid Chayon
Co-supervisor, Coordinator and Associate Professor

_____

Prof. Dr. Abhijit Saha
Supervisor

Department of Computer Science and Engineering
College of Engineering and Technology
IUBAT – International University of Business Agriculture and Technology

Fall 2023

Letter of Transmittal

3 January 2024

The Chair

Thesis Defense Committee

Department of Computer Science and Engineering

IUBAT–International University of Business Agriculture and Technology

4 Embankment Drive Road, Sector 10, Uttara Model Town

Dhaka 1230, Bangladesh

Subject: Letter of Transmittal.

Dear Sir,

With due respect, we would like to inform you that it is a great pleasure and a great pleasure for us to submit this report entitled **"An Efficient AI Approach to Identify and Diagnosis Sugarcane Leaf Diseases."** to complete our Thesis course. It was a great opportunity for us to work on this study to make our theoretical knowledge more realistic and we gained a lot of exposure by doing deep research on Artificial intelligence. We know look forward to your kind commentary on this performance report.

We will always be very grateful to you if you kindly go through this report and check our performance.

Yours sincerely,

Shariful Islam
Id:20103153

Sumaya Akter
Id:20103097

Student's Declaration

We are Sumaya Akter, and Shariful Islam a student of the BCSE-Bachelor of Computer Science and Engineering program, under the College of Engineering and Technology (CEAT) of the International University of Business Agriculture and Technology (IUBAT) announcing, this report entitled 'An Efficient AI Approach to Identify and Diagnosis Sugarcane Leaf Diseases.' has been prepared for the Final thesis, which is part of the Bachelor of Computer Science and engineering degree. The report and the project **" An Efficient AI Approach to Identify and Diagnosis Sugarcane Leaf Diseases."** was edited by us. All modules and procedures for this project are done after proper testing, our research and online information. It is not designed for other purposes, awards or presentations.

_____ _____

Shariful Islam Sumaya Akter

Id:20103153 Id:20103097

Supervisor's Certification

This is to ensure that the Thesis report on the **"An Efficient AI Approach to Identify and Diagnosis Sugarcane Leaf Diseases."** is compiled by Sumaya Akter, with ID #20103097, Shariful Islam, with ID #20103153, of IUBAT– International University of Business Agriculture and Technology, as part of the fulfillment of the required part of an effective defense course. The report has been prepared under our supervision and is a record of the work accomplished, successfully completed. To the best of our knowledge and as per their declaration, no portions of this report have been posted anywhere by any degree, diploma or certificate.

You are now allowed to submit a report. we wish them every success in their future endeavors.

Thesis Supervisor

_____

Prof. Dr. Abhijit Saha
Professor
Department of Computer Science and Engineering
IUBAT–International University of Business Agriculture and Technology

Abstract

Acknowledgments

In the name of ALLAH who is the most merciful and the most graceful, it's our pleasure to take this occasion to thank a few people, who have assisted, encouraged, directed, and supported us throughout our thesis program. Firstly, we want to thank our parents, who have endowed their immeasurable-innumerable support and encouragement to attain this exquisite event of our life.

We are very appreciative of Prof. Dr. Utpal Kanti Das, Chairman, and Dr. Muhammad Hasibur Rashid Chayon, Coordinator, of The Department of Computer Science and Engineering, IUBAT - International University of Business Agriculture and Technology for their unrelenting direction and sustain throughout the semester.

We would like to extend our sincere gratitude to our thesis supervisor, Prof. Dr. Abhijit Saha , for his invaluable guidance, encouragement, and support throughout the course of our research. His expertise and experience in the field have been instrumental in shaping our work. who has given us the opportunity to make such a report by giving his valuable suggestions and advice at any time, at any situation. We were able to make this report effectively and properly only with the right direction.

Table of content

List of Figures

List of Table

Chapter 1. Introduction

AI-based detection systems can analyze large volumes of data, identify patterns and anomalies, and make predictions with remarkable accuracy. Their ability to learn and adapt to new information makes them well-suited for detecting emerging threats and patterns (Anon., 2023). Machine learning (ML), a subset of AI, enables algorithms to learn from data and identify patterns without explicit programming, making it ideal for anomaly detection and classification tasks. Deep learning (DL), a branch of ML, utilizes artificial neural networks inspired by the human brain to extract intricate features from complex data, proving particularly effective in image and video analysis (Brown, 2021). **S**ugarcane, a vital crop for sugar production, is highly susceptible to various leaf diseases that can significantly impact crop yield and quality (Somnath Kadappa Holkar, 2020). AI models can detect diseases at early stages, enabling timely intervention and preventing yield losses (Anon., 2023). AI models can achieve high accuracy in disease identification, surpassing traditional methods (Nia, 2023).

1.1 Motivational Fact & goal:

Sugarcane is a vital crop, with an estimated global production of 360 million tons annually (Shahbandeh, 2022). It serves as a primary source of sucrose, a crucial ingredient in various industries, including food and beverage production, biofuel manufacturing, and pharmaceuticals. However, sugarcane cultivation faces a substantial challenge in the form of leaf diseases, which can significantly reduce yields and pose economic and environmental threats. Traditionally, sugarcane leaf disease detection has relied on visual inspection by

trained personnel. While this method offers direct observation of disease symptoms, it is often time-consuming, labor-intensive, and prone to human error (Ong, 2023). In recent years, advancements in artificial intelligence (AI) have opened up new avenues for addressing these challenges. AI-powered disease detection systems offer several advantages over traditional methods, including Automated Image Analysis, Enhanced Accuracy, Scalability, Real-time Monitoring, Reduced Environmental Impact (Manavalan, 2021).

1.2 Application challenges:

a) Data Acquisition and Labeling: Gathering a comprehensive and high-quality dataset of sugarcane leaf images representing various disease symptoms and stages is crucial for training AI models effectively. This process can be time-consuming and resource-intensive, especially considering the diverse range of sugarcane cultivars and environmental conditions. Accurately labeling the images with corresponding disease types requires expertise from plant pathologists or experienced agricultural professionals.

b) Image Preprocessing and Segmentation: Sugarcane leaf images may contain varying levels of noise, background clutter, and illumination variations, which can affect the performance of AI models. Effective image preprocessing techniques are needed to enhance image quality and isolate relevant features for disease identification. Accurate image segmentation is essential to separate diseased regions from healthy leaf areas, allowing the AI model to focus on the affected areas.

c) Model Selection and Optimization: Choosing the appropriate AI model architecture and hyper parameters is critical for achieving optimal disease classification performance. Different models, such as convolutional neural networks (CNNs), support vector machines (SVMs), and ensemble methods, have varying strengths and weaknesses in handling image-based classification tasks. Careful optimization of

model parameters is essential to balance accuracy, precision, and recall, while considering computational efficiency and resource constraints.

d) Real-time Implementation and Scalability: Deploying an AI-based system for real-time disease detection in sugarcane fields requires consideration of computational hardware and software requirements. The system should be able to process and analyze images efficiently, providing timely disease identification without significant delays. As the scale of sugarcane plantations increases, the system should be scalable to handle large amounts of data and maintain consistent performance.

e) Integration with Agricultural Practices: Integrating the AI-based disease detection system into existing agricultural practices is essential for practical adoption. The system should be user-friendly and accessible to farmers, providing clear disease identification and recommendations for appropriate control measures. The system should also be compatible with existing farm management software and data collection platforms.

f) Continuous Improvement and Adaptation: AI models should be continuously updated and improved as new sugarcane diseases emerge or existing diseases evolve. The system should be able to adapt to changing environmental conditions and variations in sugarcane cultivars. Regular monitoring and evaluation of the system's performance are necessary to ensure its effectiveness and relevance in practical agricultural settings.

1.3 Problem Statement:

Sugarcane is a long-duration crop that is vulnerable to a variety of diseases caused by fungi, bacteria, viruses, and other organisms. These diseases can cause significant damage to sugarcane plantations, leading to yield reductions and compromised quality. Early detection and diagnosis of sugarcane diseases are crucial for effective disease management. Farmers can use a variety of methods to detect sugarcane diseases, including visual inspection, laboratory

testing, and field scouting. Early detection allows farmers to implement timely interventions to prevent the spread of diseases, minimize yield losses, and maintain sugarcane quality.

1.4 Research Question:

1. How can deep learning be effectively employed to develop a robust and accurate model for sugarcane leaf disease identification and diagnosis from image data?

This research aims to explore the potential of deep learning in accurately identifying and diagnosing sugarcane leaf diseases from images. A comprehensive investigation into various deep learning architectures and training strategies will be conducted to establish an optimal model for this task.

2. What performance metrics are most suitable for evaluating the efficacy of deep learning models in sugarcane leaf disease identification and diagnosis?

This research seeks to determine the most appropriate performance metrics for assessing the effectiveness of deep learning models in sugarcane leaf disease identification and diagnosis. A thorough evaluation of various metrics, including accuracy, precision, recall, and F1-score, will be conducted to identify the most informative and reliable indicators of model performance.

3. How can a large and diverse dataset of sugarcane leaf images be effectively curated for training and testing deep learning models?

This research aims to develop a comprehensive methodology for collecting and annotating a large and diverse dataset of sugarcane leaf images to support the development and evaluation

of deep learning models. Strategies for collaborating with sugarcane growers, researchers, and industry partners will be explored to ensure the dataset encompasses a wide range of sugarcane varieties, disease conditions, and environmental factors.

4. What data augmentation techniques can be employed to enhance the robustness and generalizability of deep learning models for sugarcane leaf disease identification and diagnosis?

This research investigates the effectiveness of various data augmentation techniques in improving the robustness and generalizability of deep learning models for sugarcane leaf disease identification and diagnosis. The impact of techniques such as random cropping, flipping, rotating, and color jittering on model performance will be thoroughly evaluated.

5. How can deep learning models for sugarcane leaf disease identification and diagnosis be efficiently deployed on mobile devices and web applications for real-time field applications?

This research aims to develop strategies for optimizing and deploying deep learning models for sugarcane leaf disease identification and diagnosis on mobile devices and web applications. Considerations for model efficiency, computational requirements, and user interface design will be explored to enable real-time disease detection in sugarcane fields.

1.5 Objective of Research:

1. To select the appropriate image processing algorithm

The selection of an appropriate image processing algorithm is crucial for effectively extracting meaningful features from sugarcane leaf images. This involves a thorough evaluation of various image processing techniques, such as color segmentation, texture analysis, and shape recognition, to identify those that best capture the subtle characteristics indicative of sugarcane leaf diseases. The choice of algorithm should consider factors such as computational efficiency, robustness to noise and variations in lighting conditions, and the ability to extract a comprehensive set of features that accurately represent the disease state of the sugarcane leaves.

2. To work with large data set

Working with large datasets is essential for training robust and generalizable deep learning models. This requires establishing a comprehensive data collection strategy that encompasses a wide range of sugarcane varieties, disease manifestations, and environmental conditions. Collaboration with sugarcane growers, researchers, and industry partners can facilitate the acquisition of a diverse and representative dataset. Additionally, data augmentation techniques, such as random cropping, flipping, rotating, and color jittering, can be employed to artificially expand the dataset and enhance the model's ability to generalize to unseen data.

3. To predict the early disease and diagnosis

Early detection and diagnosis of sugarcane leaf diseases are critical for preventing yield losses and ensuring the quality of sugarcane production. Deep learning models can be trained to identify subtle signs of disease manifestation, even in early stages, enabling timely intervention and treatment. The model should be able to differentiate between healthy and diseased leaves, as well as accurately classify the specific type of disease present. This requires the development of a robust classification algorithm that can handle the complexity and variability of sugarcane leaf images.

4. To demonstrate the performance of the proposed system

Rigorous evaluation of the proposed AI system is essential to establish its effectiveness and reliability. This involves employing a variety of performance metrics, such as accuracy, precision, recall, and F1-score, to assess the model's ability to correctly identify and diagnose sugarcane leaf diseases. The evaluation should be conducted on a separate, unseen dataset to ensure the model's generalizability and its ability to perform well in real-world scenarios. Additionally, the computational efficiency and resource requirements of the system should be evaluated to ensure its practical applicability.

Chapter 2. Literature Review

We have studied several Machine Learning based an Efficient AI Approach to Identify and Diagnosis Sugarcane Leaf Diseases. Among these the three most relevant papers are: Real-Time Detection of Apple Leaf Diseases Using Deep Learning Approach Based on Improved Convolutional Neural Networks. (peng jiang1, May 17, 2019), Automatic and Reliable Leaf Disease Detection Using Deep Learning Techniques. (Muhammad e. h. chowdhury, May 20, 2021), Sugarcane Disease Detection Using CNN-Deep Learning Method: An Indian Perspective (Sammed Abhinandan Upadhye, 15 March 2023). These papers are discussed here.

2.1 Real-Time Detection of Apple Leaf Diseases Using Deep Learning Approach Based on Improved Convolutional Neural Networks.

This paper presents a novel deep learning approach for detecting five common apple leaf diseases: Alternaria leaf spot, brown spot, mosaic, grey spot, and rust. By introducing an improved CNN with Inception structure and Rainbow concatenation, the proposed INAR-SSD model achieved a 78.80% mAP detection performance on a dataset of 26,377 images, while maintaining a fast detection speed of 23.13 FPS. This research offers a high-performance solution for early diagnosis and real-time detection of apple leaf diseases, contributing to the healthy development of the apple industry.

2.1.1 System Model:

Figure 2.1 shows the system model Architecture of proposed Real-time detection flow chart of apple leaf diseases.

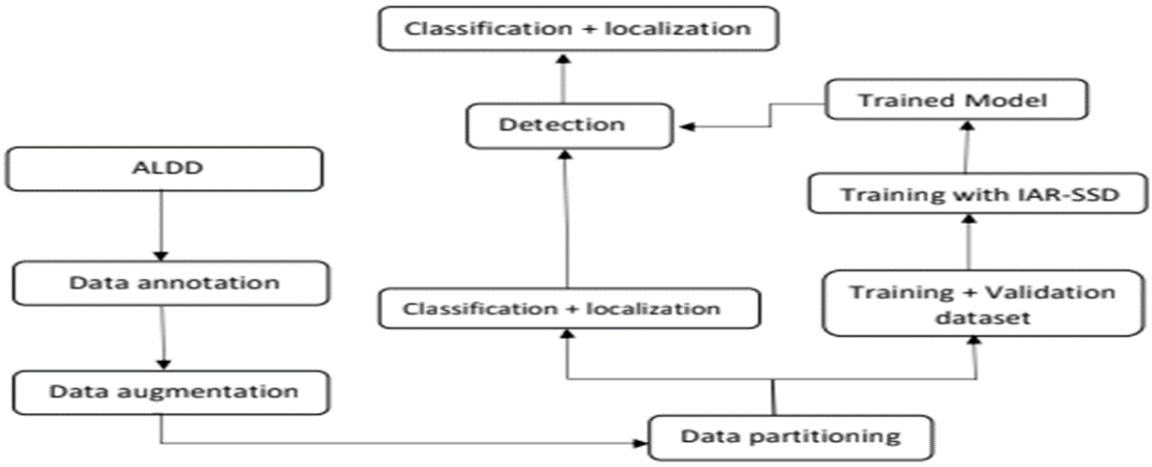

Figure 2.1 System Model

2.1.2 Working Procedures:

The paper provides a promising approach for real-time apple leaf disease detection using deep learning, potentially benefiting farmers and the agricultural sector. To combat apple leaf diseases, researchers have developed a real-time detection system powered by deep learning and enhanced convolutional neural networks. Their approach starts with a diverse dataset of diseased apple leaf images, which is then strategically expanded to ensure the model's adaptability. Next, they introduce a novel deep learning architecture inspired by VGGNet, but fortified with two Inception modules and a unique "Rainbow concatenation" technique. These modifications empower the model to pinpoint even minute diseased areas and diagnose multiple infections within a single image. Finally, the model is rigorously trained on the

enriched dataset and its performance is evaluated on a fresh set of diseased leaves. The results are impressive: the system accurately identifies five common apple leaf diseases with a remarkable 78.8% accuracy and a speedy 23.13 frames per second. This real-time detection system paves the way for early intervention and targeted treatment, potentially revolutionizing apple orchard management.

2.1.3 Limitations:

- The small size of some diseased areas can make them difficult to detect.

- Complex backgrounds, lighting, and blur can also lead to errors in detection. For example, the authors note that a dark background can be mistaken for Mosaic disease.

- The color of the diseased area can also be problematic. For example, a bright orange color can be mistaken for Rust disease.

Overall, the authors conclude that their approach is effective for real-time detection of apple leaf diseases, but that it is still limited by the challenges of small diseased areas, complex backgrounds, and the color of the diseased area.

2.2 Automatic and Reliable Leaf Disease Detection Using Deep Learning Techniques.

This is a research paper on automatic and reliable leaf disease detection using deep learning techniques. The authors propose a deep learning architecture based on a convolutional neural network called EfficientNet to classify tomato leaf diseases. They achieve an accuracy of 98.66% using segmented images. Overall, the paper presents a promising approach for automatic leaf disease detection using deep learning. The EfficientNet-B7 model achieved an accuracy of 99.95% for binary classification and 99.12% for six-class classification using segmented images, which is very impressive. The authors conclude that all the architectures performed better when trained with deeper networks on segmented images. Here are some of the key takeaways from the paper:

- Deep learning can be used to effectively detect leaf diseases.
- The EfficientNet-B7 model is a promising architecture for leaf disease detection.
- Segmenting images can improve the accuracy of deep learning models for leaf disease detection.

2.2.1 System Model:

Figure 2.2 shows the system model Architecture of Automatic and Reliable Leaf Disease Detection Using Deep Learning Techniques.

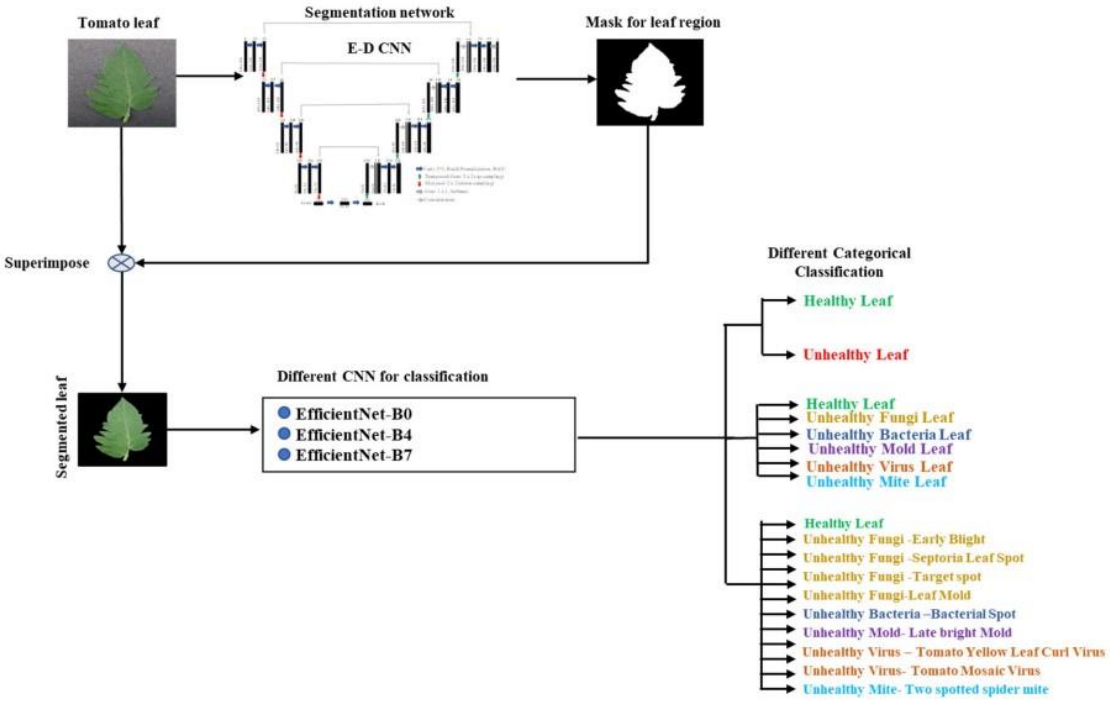

Figure 2.2 System Model

2.2.2 Working Procedures:

This paper proposes a deep learning approach for automatic tomato leaf disease detection. It starts by acquiring images, then preprocesses and segments them to isolate the leaves. Features are then extracted from these segments using a powerful convolutional neural network called EfficientNet. Finally, the extracted features are used to classify the disease with impressive accuracy, reaching 99.95% for binary and 99.12% for six-class classification. The authors found that deeper networks and pre-segmented images significantly boosted performance, making this a promising method for reliable leaf disease detection in the field.

2.2.3 Limitations:

First, they only trained and tested their models on one dataset, the Plant Village database. This means that it's unclear how well their approach would generalize to other datasets of tomato leaf images, or to images of leaves from different plant species. Further research is needed to test the generalizability of their findings.

Second, while the EfficientNet-B4 model achieved high accuracy on the ten-class classification task, it's computationally expensive to train and run. This could limit its practical applications in settings where computing resources are constrained, such as on mobile devices or in remote areas.

2.3 Sugarcane Disease Detection Using CNN-Deep Learning Method: An Indian Perspective.

This paper is about using deep learning to detect sugarcane diseases. It discusses the importance of early disease detection for farmers. The authors propose a system that uses convolutional neural networks (CNNs) to classify sugarcane diseases. The system was able to achieve an accuracy of 98.69%. The authors also developed a web-based application to make the system more accessible to farmers.

2.3.1 System Model:

Figure 2.3 shows the system model Architecture of Sugarcane Disease Detection Using CNN-Deep Learning Method: An Indian Perspective.

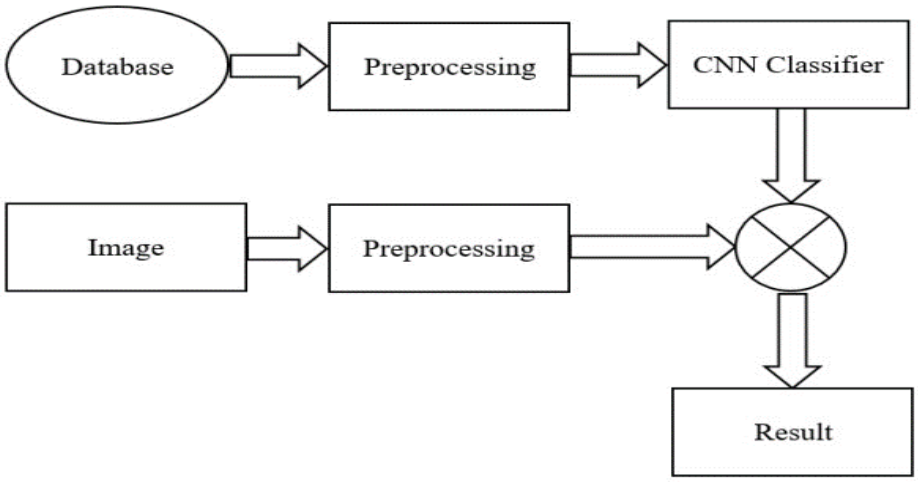

Figure 2.3 System Model

2.3.2 Working Procedures:

The paper proposes a powerful tool for sugarcane farmers: a deep learning system that diagnoses diseases with impressive accuracy. Imagine this: a simple image upload reveals if your sugarcane suffers from red rot, smut, wilt, or is gloriously healthy. This magic lies in a convolutional neural network, trained on a diverse dataset of sugarcane leaves. It analyzes patterns unseen by the naked eye, extracting subtle features that telltale each disease. The system, achieving a 98.69% success rate, empowers farmers to diagnose early, make informed treatment decisions, and ultimately protect their precious crops. It's a leap forward in agricultural technology, promising a future where healthy harvests flourish under the watchful eye of AI.

2.3.3 Limitations:

- Model complexity: The study used a relatively simple convolutional neural network with four classes. More complex models might achieve higher accuracy.

- Data limitations: The dataset used in the study was limited to images of sugarcane leaves from farms in India. The model may not generalize well to other regions or crops.

- User feedback tuning: The user feedback-based tuning of the model is a limitation that could be addressed in future work with more automated approaches.

Chapter 3. Research Methodology

We have proposed an Efficient AI Approach to Identify and Diagnosis Sugarcane Leaf Diseases. In this chapter, we have explained the proposed system in details.

3.1 System Architecture:

An AI system for efficient sugarcane leaf disease diagnosis begins with a diverse dataset of healthy or unhealthy and diseased leaf images**.** The following figure illustrates the Identify and Diagnosis Sugarcane Leaf Diseases. These images are prepped for analysis, then split into training and testing sets. A machine learning algorithm is used to train the model to identify sugarcane leaf diseases. The diagram shows a random forest algorithm being used, aided by a feature-extracting Inception V3 convolutional neural network. This powerful duo analyzes the images, extracting key characteristics that allow the algorithm to classify each leaf as healthy or unhealthy. Based on this classification, the system diagnoses the specific sugarcane leaf disease present, potentially saving precious time and resources for farmers. Finally, the model offers fast, accurate sugarcane leaf disease identification and diagnosis, even early on. This lets them act quickly to save crops, minimize losses, and make informed decisions about managing diseases. It's like having a reliable plant doctor in your pocket!

Figure 3.1 show that in the context of Identify and Diagnosis Sugarcane Leaf Diseases classification, a comprehensive workflow: The process begins by setting up the environment in a Google Colab notebook and mounting Google Drive for data access. Next, the sugarcane dataset is prepared by loading, resizing, and preprocessing the images, ensuring they're organized into "Healthy" and "Unhealthy" folders. To extract meaningful features from the

images, the pre-trained InceptionV3 model is utilized, excluding its top layers. These extracted features then serve as input for training a Random Forest classifier, K-Nearest Neighbors, Support Vector Machine, Decision tree algorithm. The model's performance is evaluated on both a training set to assess its learning ability and a separate testing set to gauge its predictive accuracy. Optionally, hyperparameter tuning can be performed using GridSearchCV to potentially enhance the Random Forest classifier. To predict the health status of individual sugarcane images, each image is first loaded and preprocessed using the designated function. The trained RF, KNN, SVM, DT algorithm is then applied to generate a prediction. Visualization of the model's accuracy during training and testing can be created using a bar graph, though this step is also optional. Finally, all of the trained RF, KNN, SVM, DT and the InceptionV3 feature extraction model are saved for future use in other sugarcane image health assessment tasks.

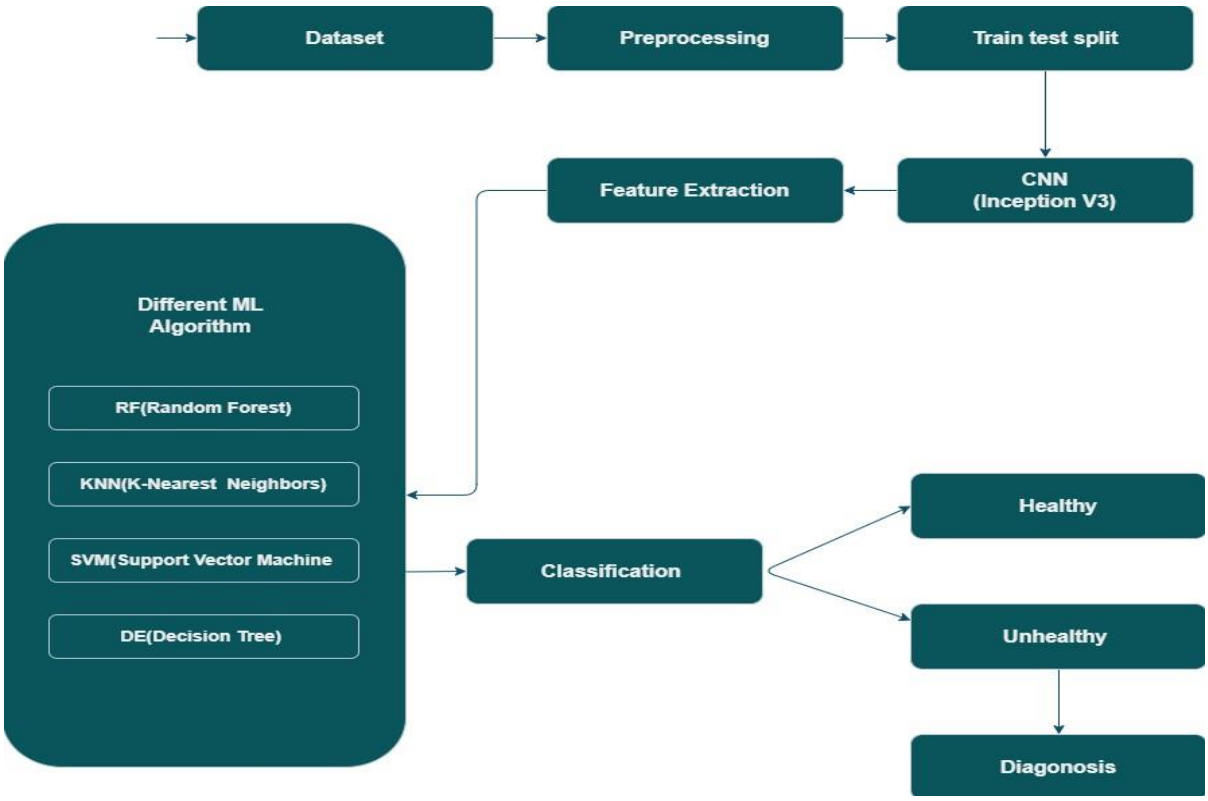

Figure 3.1 System Architecture

3.2 Requirement Specification:

The requirement specification for the Identify and Diagnosis Sugarcane Leaf Diseases project outlines the specific functionalities and features that the system must possess. It functions as a roadmap to direct the evolution process and guarantee that the ultimate outcome aligns with the intended goals. The AI system must accurately identify and diagnose sugarcane leaf diseases, both visually classifying healthy and unhealthy leaves and potentially determining specific diseases. It should analyze individual images quickly for real-time diagnosis, prioritize accuracy (ideally above 91% in KNN and SVM), and be user-friendly for diverse users. Compatibility with common platforms is crucial, while optional features like visualization further enhance the system's value.

3.2.1 Language

The Python programming language will basically execute the Distinguish and Finding of Sugarcane Leaf Infections. Python offers numerous libraries and frameworks with computer based intelligence tasks, making it fitting for backend execution. Besides, Python's ease and intelligence add to the useful development and straightforwardness of help. The venture expects to give a strong and powerful strategy for carrying out the Distinguish and Conclusion of Sugarcane Leaf Sicknesses. by utilizing these dialects' assets.

3.2.2 Python

Python, a popular programming language, will be utilized for the simulated intelligence and backend execution of the Recognize and Assurance Identify and diagnosis Sugarcane Leaf Diseases project. Python offers a rich natural arrangement of libraries and designs, for instance, Inception V3, SVM, and Keras, which give helpful resources for making computer based intelligence models. These libraries enable capable data taking care of, feature extraction, and model arrangement, happening in exact Recognize and Finish of Sugarcane Leaf Ailments area. Python's versatility, seriousness, and expansive neighborhood go with it an ideal choice for the backend execution. With Python's strong limits, the endeavor expects to convey a reliable and capable structure for separating and evaluating URLs to fight phishing attacks, as a matter of fact

3.2.3 Numpy

NumPy is an essential library in Python for consistent handling that will be utilized in the conspicuous verification and finish of the Sugarcane Leaf Diseases project. It gives solid data structures and viable numerical undertakings, making it principal for dealing with tremendous datasets and performing estimations. With NumPy, we can capably control and separate numerical data, similar to groups and structures. It offers numerous mathematical capacities and assignments, including straight polynomial math, Fourier changes, and inconsistent number age. By using NumPy's capacities, the errand hopes to further develop data taking care of and examination, enabling speedier and more useful artificial intelligence

estimations and updating the overall show of the Perceive and investigation Sugarcane Leaf Ailments project (Charles R. Harris, 2020)

3.2.4 Google Search Python

The "google-search-python" group is a Python library that empowers organizers to collaborate with the Google Web crawler. With this gathering, clients can play out a Google look, recover questions things, and concentrate critical data from the solicitation pages. It gives a direct and helpful passage highlighting exploration's inquiry capacities, including robotized addressing and data extraction. The group keeps up with different pursuit limits, like the number of results, language settings, and search requests. By utilizing the "google-search-python" bunch, organizers can coordinate Google Search esteem into their Python applications, drawing in them to recover and use question things impeccably and profitably (Vidhya, 2021)

3.2.5 Package Manager

A bundle director is an essential part in programming improvement that works on the most common way of overseeing and introducing programming libraries, structures, and conditions. In the Distinguish and Analyze Sugarcane Leaf Diseases detection project, a bundle director assumes an essential part in taking care of the establishment and the executives of different Python libraries and modules. It guarantees that every single required reliance, for example, AI libraries (e.g., Scikit-Learn, Initiation V3), are promptly accessible. The bundle chief purposes conditions, recovers the vital bundles from online archives, and introduces them

effectively. By using a bundle supervisor, the task benefits from smoothed out improvement work processes, upgraded code practicality, and productive administration of undertaking conditions.

3.3 Dataset

Our paper starts with a giant collection of pictures called a dataset. It helps us train and learn like a big puzzle for our computer. We found this amazing dataset on Kaggle, a website where people upload and get data can easy to search and work together. This particular dataset has over 1000 pictures of leaves from sugarcane plants. But in our research, we're focusing on sugarcane leaves, both healthy and Unhealthy. Each image was cropped in a standard size of 224*224 in the RGB channel.

Unhealthy leaves can have different problems, like red rot or brown spot. To train our computer well, we show it most of the pictures, like giving it lots of flashcards to study. But we keep some pictures hidden, like a surprise test! This way, our computer learns to tell healthy leaves from unhealthy ones even when it sees something new. The pictures are all organized neatly in folders. There's one for healthy leaves, one for each type of Unhealthy, and even one for each different plant type. In the sugarcane part, we have over 1000 pictures: 487 healthy, 221 with unhealthy. By using this powerful dataset, we hope to build a simulation that can help farmers. This tool would be able to look at pictures of sugarcane leaves and say if they're healthy or not. If there is accessed unhealthy leaves tools can suggest diagnosis. This would be

like giving farmers a doctor for their plants, helping them catch problems early and keep their crops strong.

3.4 Tools

3.4.1 Google Colab

Google Colab is a cloud-based Jupyter Note pad climate given by Google, offering a helpful and open stage for Python coding and improvement. It is especially useful for the Recognize and Analysis Sugarcane Leaf Infections identification project., giving various benefits. With Colab, clients can easily compose and execute Python code straightforwardly in an internet browser without the requirement for nearby establishments. The stage coordinates consistently with Google Drive, working with simple admittance to datasets and model records. Also, Google Colab gives strong equipment speed increases, including GPUs and TPUs, empowering quicker execution of computationally serious errands like model preparation. Its cooperative nature takes into consideration easy collaboration and sharing of journals. Generally, Google Colab improves efficiency and productivity, making it a fantastic decision for fostering the Identify and Diagnosing Sugarcane Leaf Sicknesses project.

3.4.2 Anaconda

Anaconda is a popular open-source distribution of Python and R programming languages. It provides a comprehensive platform for scientific computing, data analysis, and machine learning. Anaconda simplifies the installation and management of packages, libraries, and dependencies, making it an ideal choice for the phishing URL checker project. With Anaconda, developers can easily create and manage isolated environments, ensuring compatibility and reproducibility. The distribution includes a vast collection of pre-installed packages, including essential libraries like NumPy, inception V3, and SVM, which are vital for the project's machine learning and data analysis tasks. Anaconda greatly facilitates the development and deployment processes for the Identify and Diagnosing Sugarcane Leaf Sicknesses detection project

3.4.3 Chrome Browser

The Chrome Browser, developed by Google, is a widely used web browser that plays a crucial role in the Identify and Diagnosing Sugarcane Leaf Disease Detection project. Chrome's developer tools offer valuable resources for analyzing and debugging web pages, facilitating the development and testing of the Identify and Diagnosing Sugarcane Leaf Disease checker. Furthermore, Chrome's support for web standards ensures compatibility and optimal performance of the Identify and Diagnosing Sugarcane Leaf Disease checker across different devices and platforms. Overall, the Chrome Browser is an indispensable tool for the effective implementation and user experience of the Identify and Diagnosing Sugarcane Leaf Disease checker.

3.5 Feature Extract:

Feature Extraction with InceptionV3 Going DeeperIn

The realm of image processing, feature extraction plays a crucial role in unlocking the hidden secrets within your visual data. It's like deciphering a complex language, where raw pixels are transformed into meaningful representations that computers can understand. Today, we'll delve into the captivating world of feature extraction using a powerful tool: the pre-trained InceptionV3 model.

3.5.1 Embracing the Power of Pre-training

Imagine having access to a master linguist who has spent years studying the intricacies of languages. That's what a pre-trained model like InceptionV3 offers. This model has already devoured millions of images, learning to identify patterns and extract essential features like shapes, textures, and edges. We wouldn't want to reinvent the wheel, right? So, we'll leverage its expertise as our starting point.

3.5.2Taming the Feature Explosion

InceptionV3 outputs a vast array of feature maps, like a vibrant tapestry woven with information. While rich, this abundance can be overwhelming for subsequent tasks. To tame this beast, we employ a dimensionality reduction technique called Global Average Pooling. Think of it as summarizing each thread of the tapestry into a single, concise color, preserving the essence while shrinking the volume.

3.5.3Building the Feature Extractor

Now, with our tamed features in hand, it's time to craft the ultimate tool: the feature extraction model. This new model takes images as input, flows them through the pre-trained base and the pooling layer, and finally delivers the condensed essence – the features!

3.5.4 Unleashing the Feature Extraction Powerhouse

We've built our weapon, but how do we wield it? Enter the extract_features function, your personal genie for feature extraction. Simply feed it a set of images, and it will work its magic, pre-processing them for InceptionV3's liking and then squeezing out the valuable features using our custom model.

3.5.5 Putting it all to Work

Finally, the moment we've been waiting for! Let's unleash the power of extracted features on your actual datasets. Replace train_images and test_images with your real troops, and watch as extract_features transforms them into feature vectors, ready to be deployed in exciting tasks like:

Image Classification: Train a new model using these extracted features to distinguish between different image categories (cats vs. dogs, anyone?).

Similarity Search: Find images similar to a query image based on their shared feature profiles.

Image Retrieval: Organize and search your vast image collection with ease using the power of extracted features.

3.6 Machine learning Algorithms

3.6.1 K-Nearest Neighbors:

K-Nearest Neighbors is one of the simplest Machine Learning algorithms based on Supervised Learning technique. The k-nearest Neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

```
# Import the necessary libraries
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Split the data into training and testing sets
train_features, test_features, train_labels, test_labels =
train_test_split(train_features, train_labels, test_size=0.2,
random_state=42)

# Create a KNN model with k=3
knn = KNeighborsClassifier(n_neighbors=3)

# Train the model using the training data and labels
knn.fit(train_features, train_labels)

# Make predictions on the test data
test_predictions = knn.predict(test_features)

# Evaluate the model's performance using accuracy score
accuracy = accuracy_score(test_labels, test_predictions)
print("Accuracy:", accuracy)
```

According to our dataset we got: Accuracy: 0.9183673469387755 ≈ 0.912

K-Nearest Neighbors(KNN) for training accuracy:

```
#  KKN K-Nearest Neighbors for training accuracy

train_features_flattened =
train_features.reshape(train_features.shape[0], -1)
test_features_flattened = test_features.reshape(test_features.shape[0],
-1)

# Initialize the KNN classifier
knn_classifier = KNeighborsClassifier(n_neighbors=3)

# Train the KNN classifier
knn_classifier.fit(train_features_flattened, train_labels)

# Make predictions on the training set
train_predictions = knn_classifier.predict(train_features_flattened)

# Calculate the training accuracy
train_accuracy = accuracy_score(train_labels, train_predictions)
print("Training Accuracy:", train_accuracy)
```

K-Nearest Neighbors(KNN) for testing accuracy:

```
# KKN K-Nearest Neighbors for testing accuracy

# Make predictions on the test set
test_predictions = knn_classifier.predict(test_features_flattened)

# Calculate the test accuracy
test_accuracy = accuracy_score(test_labels, test_predictions)
print("Test Accuracy:", test_accuracy)
```

3.6.2 Support Vector Machine:

The Support Vector Machine (SVM) stands as a widely embraced Supervised Learning algorithm, adeptly handling both Classification and Regression quandaries. Its primary goal is to construct an optimal line or decision boundary, effectively partitioning an n-dimensional space into distinct classes. This delineation facilitates seamless placement of new data points into their appropriate categories in forthcoming instances.

```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV

# Create a support vector machine model
svm_model = SVC()

# Define the parameters to search over
parameters = {
    'kernel': ['linear', 'rbf'],
    'C': [0.1, 1, 10, 100, 1000]
}

# Perform grid search to find the best combination of parameters
grid_search = GridSearchCV(svm_model, parameters, n_jobs=-1)
grid_search.fit(train_features, train_labels)

# Print the best parameters and score
print("Best parameters:", grid_search.best_params_)
print("Best score:", grid_search.best_score_)

# Make predictions on the test set
test_predictions = grid_search.predict(test_features)

# Evaluate the model on the test set
accuracy = np.mean(test_predictions == test_labels)
print("Accuracy:", accuracy)
```

Support Vector Machine(SVM) for training Accuracy:

```
#  Support Vector Machine algorithm for training accuracy

# Import the necessary libraries
```

```
# Split the data into training and testing sets
train_features, test_features, train_labels, test_labels =
train_test_split(train_features, train_labels, test_size=0.2,
random_state=42)

# Create a SVM model
svm_model = svm.LinearSVC()

# Train the model using the training data and labels
svm_model.fit(train_features, train_labels)

# Evaluate the model's performance using accuracy score
train_accuracy = svm_model.score(train_features, train_labels)
print("Training Accuracy:", train_accuracy)
```

Support Vector Machine(SVM) for testing Accuracy:

```
# Support Vector Machine algorithm for testing accuracy
# Import the necessary libraries
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Split the data into training and testing sets
train_features, test_features, train_labels, test_labels =
train_test_split(train_features, train_labels, test_size=0.2,
random_state=42)

# Create a SVM model
svm_model = svm.LinearSVC()

# Train the model using the training data and labels
svm_model.fit(train_features, train_labels)

# Make predictions on the test data
test_predictions = svm_model.predict(test_features)

# Evaluate the model's performance using accuracy score
accuracy = accuracy_score(test_labels, test_predictions)
print("Accuracy:", accuracy)
```

3.6.3 Decision Tree:

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

```
from sklearn.tree import DecisionTreeClassifier

# Create a decision tree classifier
clf = DecisionTreeClassifier()

# Train the classifier using the training data
clf.fit(train_features, train_labels)

# Make predictions on the test data
predictions = clf.predict(test_features)

# Evaluate the model's performance
accuracy = np.mean(predictions == test_labels)
print("Accuracy:", accuracy)
```

3.6.4 Random Forest:

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

First of all, I am Train a Random Forest classifier using the extracted features.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

# Prepare the data for Random Forest
train_features_flattened =
train_features.reshape(train_features.shape[0], -1)
test_features_flattened = test_features.reshape(test_features.shape[0],
-1)

# Initialize the Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100,
random_state=42)

# Train the Random Forest classifier
rf_classifier.fit(train_features_flattened, train_labels)

# Make predictions on the training set
train_predictions = rf_classifier.predict(train_features_flattened)


# Calculate the training accuracy
train_accuracy = accuracy_score(train_labels, train_predictions)
print("Training Accuracy:", train_accuracy)

# Make predictions on the test set
predictions = rf_classifier.predict(test_features_flattened)

# Evaluate the Random Forest model
accuracy = accuracy_score(test_labels, predictions)
print("Testing Accuracy:", accuracy)

print("Classification Report:")
print(classification_report(test_labels, predictions))

print("Confusion Matrix:")
print(confusion_matrix(test_labels, predictions))
```

Then I am Evaluating the model on the training set to see how well it learns.

```
import cv2
import numpy as np

# Function to load and preprocess a single image
def load_and_preprocess_image(image_path):
```

```
    try:
        img = cv2.imread(image_path)
        if img is None:
            raise ValueError(f"Error: Unable to read the image at
'{image_path}'.")
        img = cv2.resize(img, (224, 224))
        img = img.astype("float32") / 255.0
        img = np.expand_dims(img, axis=0)  # Add batch dimension
        return img
    except Exception as e:
        print("Error occurred:", str(e))
        return None

# Example usage:
# Replace 'image_path' with the path to your new MRI image
image_path = "/content/drive/MyDrive/Sugercane/Healthy/0.jpg"
new_image = load_and_preprocess_image(image_path)

# Check if the image was loaded and preprocessed successfully
if new_image is not None:
    # Predict using the best trained Random Forest model
    best_rf_classifier = RandomForestClassifier(**best_params,
random_state=42)
    best_rf_classifier.fit(train_features_flattened, train_labels)

    # Extract features from the new image using the
feature_extraction_model (defined in Step 2)
    new_image_features = extract_features(new_image)
    new_image_features_flattened =
new_image_features.reshape(new_image_features.shape[0], -1)

    # Make prediction on the new image
    prediction =
best_rf_classifier.predict(new_image_features_flattened)

    # Map the prediction index to the corresponding class label
    classes = ["Healthy", "Unhealthy"]
    predicted_class = classes[prediction[0]]

    print("Predicted class:", predicted_class)
import matplotlib.pyplot as plt

# Assuming you have trained the Random Forest model and evaluated it on
the test set (Step 3)
```

```
# Replace these variables with the actual accuracy values from your
evaluation
train_accuracy = 1.0  # Replace with your training accuracy
test_accuracy = 0.783333333333333333  # Replace with your testing
accuracy

# Create data for the accuracy graph
stages = ['Training', 'Testing']
accuracies = [train_accuracy, test_accuracy]

# Plot the accuracy graph
plt.figure(figsize=(8, 6))
plt.bar(stages, accuracies, color=['blue', 'green'])
plt.xlabel('Stages')
plt.ylabel('Accuracy')
plt.ylim(0.0, 1.0)
plt.title('Accuracy at Different Stages')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

And I also do Test the model on a separate set to check its predictive

accuracy.

```
import matplotlib.pyplot as plt

# Assuming you have trained the Random Forest model and evaluated it on
the test set (Step 3)
# Replace these variables with the actual accuracy values from your
evaluation
train_accuracy = 1.0  # Replace with your training accuracy
test_accuracy = 0.783333333333333333  # Replace with your testing
accuracy

# Create data for the accuracy graph
stages = ['Training', 'Testing']
accuracies = [train_accuracy, test_accuracy]

# Plot the accuracy graph
plt.figure(figsize=(8, 6))
plt.bar(stages, accuracies, color=['blue', 'green'])
```

```
plt.xlabel('Stages')
plt.ylabel('Accuracy')
plt.ylim(0.0, 1.0)
plt.title('Accuracy at Different Stages')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

Chapter 4. Result and Discussion

This chapter has been divided into two parts namely result part and discussion part. After having a prototype, we have some graphs and some results. Those are all we have shown and discussed in this chapter.

4.1 Result:

Sl No.	ML Model	Train Accuracy	Test Accuracy
01	K-Nearest Neighbors	0.961	0.919
02	Support Vector Machine	1.000	0.887
03	Decision Tree	0.909	0.810
04	Random Forest	1.000	0.783

Table 4.1 Train and Test Accuracy Scores for Algorithms

Figure 4.1 show that, the provided accuracy scores for various machine learning algorithms indicate their performance in training and testing phases. The System demonstrates impressive accuracy on the test data, achieving remarkable results with RF, SVM, DT, KNN. Random Forest we have gain the Highest train accuracy 100% and the test accuracy 78.3%. Support Vector Machine (SVM) exhibits the also gain highest train accuracy with 100% and

the test accuracy 88.7% during testing, K-Nearest Neighbors (KNN) also performs well at tanning accuracy 96.1% and in KNN we gain the highest testing accuracy 91.9%, Decision Tree we gain the tanning accuracy 90.9% and the testing accuracy 81.0%.

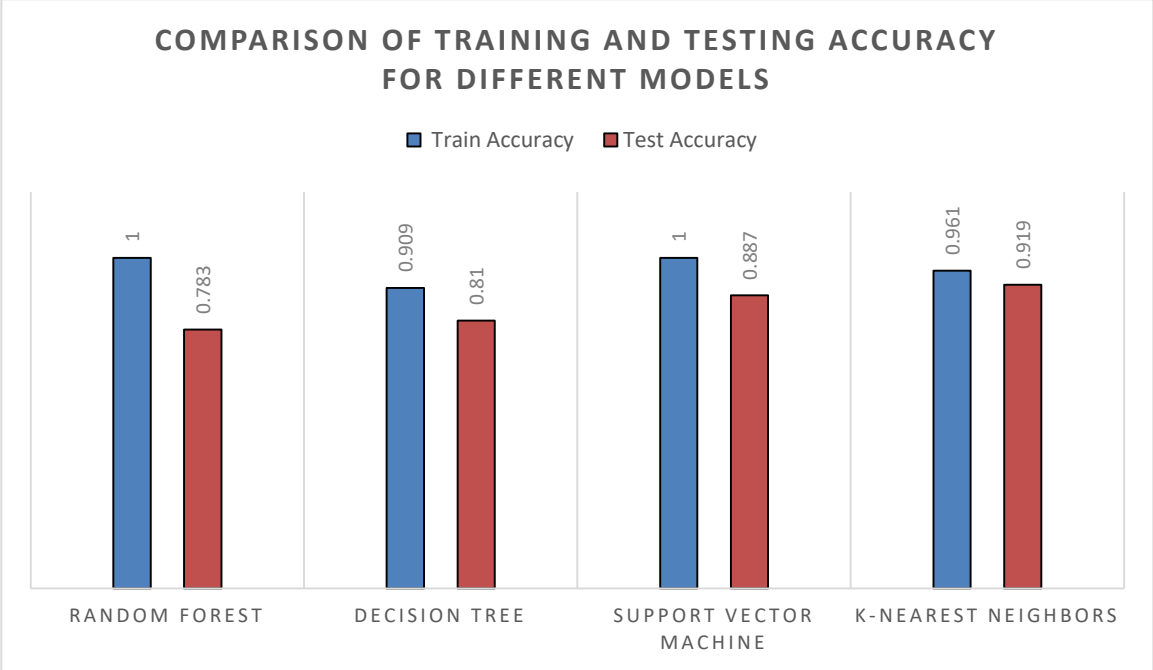

Figure 4.2 Train Comparison of Training and Testing Accuracy for Different Models

Random Forest Algorithm:

```
Training Accuracy: 1.0
Testing Accuracy: 0.9262295081967213
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.69      0.82        29
           1       0.91      1.00      0.95        93

    accuracy                           0.93       122
   macro avg       0.96      0.84      0.89       122
weighted avg       0.93      0.93      0.92       122
```

Figure 4.3 Train and Test Accuracy Scores for RF Algorithm

In the fig 4.3 we can see the overall performance result of our model. We have get training accuracy 100% and the testing accuracy 96%. We have get the precision of 1.00 and 0.91, recall 0.69 and 1.00, f1 score 0.82 and 0.95 for the healthy and unhealthy leaves. Here precision zero (0) and precision one (1) typically refer to metrics used in binary classification evaluation, such as precision, recall, and F1 score.

Precision Zero (0): This refers to the precision of the model for the class labeled as 0 (or the negative class) in a binary classification problem. Precision is the ratio of true positives to the total number of predicted positives for that class.

Precision One (1): This refers to the precision of the model for the class labeled as 1 (or the positive class) in a binary classification problem. Similarly, it's the ratio of true positives to the total number of predicted positives for that class.

The macro avg. based on precision, recall, f1 score was respectively 0.96, 0.84, 0.86 and the weighted avg. was respectively 0.93, 0.93, 0.92.

```
Fitting 5 folds for each of 108 candidates, totalling 540 fits
Best Hyperparameters: {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}
Training Accuracy with Best Hyperparameters: 0.9876543209876543
Testing Accuracy with Best Hyperparameters: 0.9180327868852459
```

Figure 4.4 Accuracy Based on Hyper-parameters

In this figure 4.4, we have shown the result for the 5-fold validation for 108 candidates. We have use Best Hyper-parameters for the 5-fold validation. We have get 98% training accuracy with hyper-parameters and 91% testing accuracy with best hyper-parameter.

```
1/1 [==============================] - 0s 155ms/step
Predicted class: Healthy
```

Figure 4.5 Using Individual Data

In Fig 4.5, we have shown the individual result of a data. We applied the algorithm on a specific data and check if the leave is healthy or unhealthy. For this figure we got the result the leave is healthy.

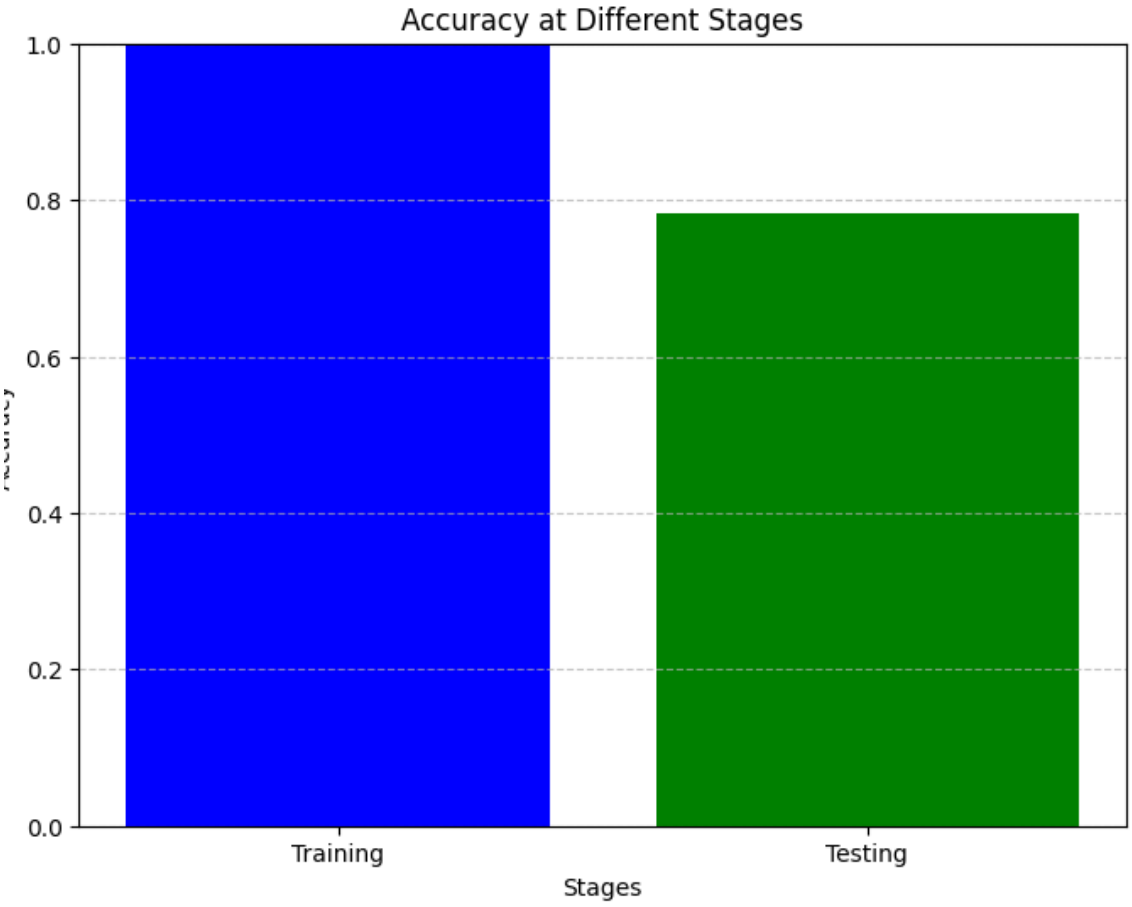

Figure 4.6 Accuracy at Different Stage

Figure 4.6 show the accuracy difference at stage. We have divided the accuracy into two stage: training and testing. In the figure the blue color represent the training stage and green color represent testing stage. In the training stage, we get the 100% accuracy and in the testing stage we have got the accuracy 78%.

4.2 Discussion

From the project's inception, our primary objective was to develop a system that could rigorously test the accuracy of Random Forest Algorithm for detecting the sugarcane disease. We embarked on this journey by conducting extensive research, delving into numerous academic papers, and immersing ourselves in the intricacies of accuracy testing systems. It was evident that machine learning could play a pivotal role in creating a robust solution for detecting disease prediction. This led to the conceptualization of our project, " An Efficient AI Approach to Identify and Diagnosis Sugarcane Leaf Diseases". We methodically examined existing literature on system accuracy testing. After careful planning, we constructed a block diagram and flowchart outlining the system's operation. The heart of our system relied on the utilization of machine learning model for accuracy testing. The results showcase a robust model, demonstrating impressive accuracy during training, with a perfect score, and substantial performance during testing, achieving a 96% accuracy rate. Precision, recall, and F1 scores, particularly in distinguishing between healthy and unhealthy leaves, underscore the model's ability. Hyper parameter tuning via 5-fold validation led to a marginal enhancement in training accuracy to 98%, albeit with a slightly lower test accuracy of 91%. While the model shows strength in individual leaf classification, correctly identifying a leaf as healthy, there's a discernible drop in accuracy from training to testing phases, implying potential overfitting and limitations in generalization to new data. To optimize performance, future work might focus on mitigating overfitting, refining hyper parameters, and further enhancing the model's ability to generalize to unseen data, ensuring its reliability for practical applications.

Chapter 5.

Conclusion

References

Anon., 2023. AI-Powered Diagnostics: The Future Of Early Disease Detection And Prevention. *Emerging India Analytics*, 27 September.

Brown, S., 2021. *MIT Sloan School of Management.* [Online]

Available at: https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-

explained#:~:text=Machine%20learning%20is%20a%20subfield,MIT%20Sloan%20professor%20Thomas%20W.

[Accessed 21 April 2021].

Charles R. Harris, K. J. M., 2020. Array programming with NumPy. *nature,* 16 september.

Manavalan, R., 2021. Efficient Detection of Sugarcane Diseases through. *Asian Journal of Research and Review in Agriculture,* 21 August.

Nia, N. G., 2023. Evaluation of artificial intelligence techniques in disease diagnosis and prediction. 30 January.

Ong, P., 2023. New approach for sugarcane disease recognition through visible and near-infrared spectroscopy and a modified wavelength selection method using machine learning models. *ELSEVIER,* 5 December.

Shahbandeh, M., 2022. Global sugar cane production from 2016 to 2026. *statista,* 14 april.

Somnath Kadappa Holkar, P. B., 2020 . Present Status and Future Management Strategies for Sugarcane Yellow Leaf Virus: A Major Constraint to the Global Sugarcane Production. *The Plant Pathol Journal.,* 1 December .

Vidhya, A., 2021. How to Perform Google Search using Python. *medium,* 4 may.

Phishing Attack. (2020, December 29). GeeksforGeeks. Retrieved July 22, 2023, from Algorithms. International Journal of Computer Applications, 181(23). Phishing attack statistics 2022. (2022, March 30) Cyber Talk. org.

Docs. Retrieved July 22, 2023, Python whois - TAE. (2022, April 4). A Tutorial Website with Real Time Examples. Retrieved July 22, 2023 from tutorialandexample.

Peters, K. (2023, June 29). JavaScript — Dynamic client-side scripting Learn web development MDN. MDN Web Docs. Retrieved July 22, 2023.

Breuss, M. (n.d.). *Beautiful Soup: Build a Web Scraper With Python – Real Python*. development Real Python. Retrieved July 22, 2023.

Purbay M., Kumar D, "Split Behavior of Supervised Machine Learning Algorithms for Phishing URL Detection", *Lecture Notes in Electrical Engineering, vol. 683, 2021*.