A Project Report
on

Identification of Paddy Leaf Disease and Measuring Affected Part of Leaf

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

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SICENCE AND ENGINEERING

by

20WH1A0528	Ms. Narla Sathvika
20WH1A0554	Ms. Peddi Vahnika
20WH1A0557	Ms. Kashetty Deekshitha

under the esteemed guidance of

Dr. Nara Sreekanth
Associate Professor

Department of Computer Science and Engineering
BVRIT HYDERABAD
College of Engineering for Women
(NBA Accredited – EEE, ECE, CSE and IT)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

May, 2024

DECLARATION

We hereby declare that the work presented in this project entitled **“Identification of Paddy Leaf Disease and Measuring Affected Part of Leaf ”** submitted towards completion of Project Work in IV year of B.Tech., CSE at ‘BVRIT HYDERABAD College of Engineering For Women’, Hyderabad is an authentic record of our original work carried out under the guidance of Dr. N. Sreekanth, Associate Professor, Department of CSE.

Sign. with date:
Ms. Narla Sathvika
(20WH1A0528)

Sign. with date:
Ms. Peddi Vahnika
(20WH1A0554)

Sign. with date:
Ms. Kashetty Deekshitha
(20WH1A0557)

BVRIT HYDERABAD
College of Engineering for Women
(NBA Accredited – EEE, ECE, CSE and IT)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

Department of Computer Science and Engineering

Certificate

This is to certify that the Project Work report on "**Identification of Paddy Leaf Disease and Measuring Affected Part of Leaf**" is a bonafide work carried out by Ms. Narla Sathvika (20WH1A0528) ; Ms. Peddi Vahnika (20WH1A0554) ; Ms. Kashetty Deekshitha (20WH1A0557) in the partial fulfillment for the award of B.Tech. degree in **Computer Science and Engineering, BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Head of the Department	**Guide**
Dr. M Sree Vani,	**Dr. N. Sreekanth**
Professor,	**Associate Professor**
Department of CSE	

External Examiner

Acknowledgements

We would like to express our sincere thanks to **Dr. K V N Sunitha, Principal**, **BVRIT HYDERABAD College of Engineering for Women**, for providing the working facilities in the college.

Our sincere thanks and gratitude to our **Dr. M. Sree Vani, Professor**, Department of CSE, **BVRIT HYDERABAD College of Engineering for Women** for all the timely support and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Mr. N. Sreekanth, Associate Professor**, Department of CSE**, BVRIT HYDERABAD College of Engineering for Women** for his constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank our Project Coordinator, all the teaching and non-teaching staff of **CSE** Department who helped us directly or indirectly, parents, siblings and friends for their cooperation in completing the project work.

Ms. Narla Sathvika

(20WH1A0528)

Ms. Peddi Vahnika

(20WH1A0554)

Ms. Kashetty Deekshitha

(20WH1A0557)

Contents

ABSTRACT

Farmers are facing difficulties with their crops lately, mainly because they're not growing properly due to diseases. The problem is, farmers don't always know what these diseases are or how to treat them effectively with the right chemicals.This is time consuming and expensive. Therefore highly effective and cost efficient method is needed for fast detection of crop disease. A machine learning model for disease detection and the analysis of affected areas is being developed to tackle the issue. By analyzing images of paddy leaves, the model extracts relevant features and applies trained model to detect and classify diseases. The images of the diseased and healthy rice plant are given as input to the model and it produces detected diseases and affected area in the leaf as an output. Remedies will be recommended based on the disease identified. The remedies provide proper information regarding the pesticide or insecticide to be used in order to cure the disease.

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Paddy crop disease detection stands as a cornerstone for global food security, economic stability, and the pursuit of sustainable agricultural practices. Its importance lies in the early identification of diseases, facilitating timely intervention to mitigate crop damage and uphold yield and quality standards. Such interventions not only safeguard the interests of individual farmers but also contribute to the overall economic prosperity of agricultural communities. Moreover, the adoption of technologies like machine learning and image recognition enhances disease detection efficacy, paving the way for precision agriculture methodologies. By accurately pinpointing diseases, these technologies aid in minimizing environmental impact and optimizing resource utilization, thus bolstering the sustainability of crop production systems. Importantly, the ramifications of effective disease detection transcend local contexts, exerting influence on global food trade dynamics and emphasizing the intricate interconnectedness of agricultural practices on a broader scale.

In prior research endeavors, a plethora of models spanning from Support Vector Machines, Convolutional Neural Networks, K Nearest Neighbors, Decision Trees, to Random Forest have been harnessed with success in detecting diseases on plant leaves. This diverse array of methodologies has been instrumental in advancing disease detection across a wide spectrum of crops, ranging from jamun, citrus, and apple to staple crops like paddy, maize, grape, mango, and tomato. The culmination of these efforts has not only facilitated the identification of various diseases affecting these crops but has also enriched the field of plant pathology with valuable insights and methodologies.

In essence, the intersection of paddy crop disease detection with advanced technologies heralds a new era in agricultural management. Its pivotal role in ensuring food security, bolstering economic stability, and fostering sustainable agricultural practices underscores its significance in addressing the multifaceted challenges of the modern agricultural landscape. As these technologies continue to evolve and proliferate, the potential for further advancements in disease detection and management holds promise for enhancing agricultural productivity and resilience on a global scale.

The extensive exploration into disease detection methodologies across multiple crops underscores the interdisciplinary nature of agricultural research. By leveraging a combination of machine learning algorithms and domain-specific knowledge, researchers have been able to tailor detection strategies to suit the unique characteristics of each crop and disease. This holistic approach not only expands our understanding of plant diseases but also equips farmers and agricultural stakeholders with the tools necessary to combat them effectively, thereby safeguarding crop health and global food security.

Moreover, the cross-pollination of ideas and techniques between different crop species serves to amplify the impact of research efforts in plant pathology. Lessons learned from one crop can often be applied to others, leading to synergistic advancements in disease detection and management practices. As such, the collaborative nature of agricultural research continues to drive innovation and progress in the quest for sustainable and resilient crop production systems. Our project is meticulously focused on the thorough examination of paddy crops, prioritizing the detection of prevalent diseases such as Bacterial Leaf Blight, Leaf Smut, Brown Spot, and Narrow Brown. We recognize the critical importance of not only identifying these diseases but also understanding the extent of their spread within the crop. Therefore, our endeavor encompasses the crucial task of recognizing affected areas within the paddy fields to assess the magnitude of disease dissemination accurately. To achieve these objectives, we have harnessed the power of two machine learning models: Support Vector Machine and K Nearest Neighbors. These models have been strategically deployed to scrutinize subtle differences in leaf patterns and symptoms, enabling precise and timely disease identification. Through the application of these advanced machine learning techniques, our aim is to contribute significantly to the enhancement of disease detection methodologies specifically tailored for paddy crops.

By leveraging machine learning models such as Support Vector Machine and K Nearest Neighbors, our project endeavors to revolutionize disease detection in paddy crops. Our primary focus lies in detecting prevalent diseases like Bacterial Leaf Blight, Leaf Smut, Brown Spot, and Narrow Brown, which pose significant threats to crop health and yield. In addition to identifying these diseases, we place particular emphasis on discerning the affected regions within the crop, thereby providing valuable insights into the extent of disease spread. This meticulous approach enables us to develop a comprehensive understanding of disease dynamics in paddy fields, facilitating targeted interventions and management strategies. Through the integration of cutting-edge machine learning techniques, our project aims to elevate disease detection methodologies in paddy crops to new heights, ultimately contributing to the advancement of agricultural sustainability and food security. Our project takes a comprehensive approach that extends beyond merely identifying specific diseases in paddy crops. We place significant importance on quantifying the affected regions within the crop, thereby adopting a dual-pronged strategy to enhance the precision of our diagnostic system. By integrating this aspect into our research, we aim to provide farmers and agricultural stakeholders with more nuanced insights into the distribution and severity of diseases in their fields. This, in turn, enables targeted interventions that are tailored to the specific needs of each crop, contributing to more effective disease management practices and improved agricultural outcomes.

As our research progresses, we remain committed to addressing the practical challenges associated with disease management in paddy crops. By delving deeper into understanding the complexities of disease dynamics and their interactions with environmental factors, we strive to develop innovative solutions that can revolutionize crop protection strategies. Our efforts are geared towards not only enhancing the accuracy and reliability of disease detection but also empowering farmers with actionable information that enables them to make informed decisions about disease control measures. Through collaboration with agricultural communities and stakeholders, we aim to translate our research findings into practical tools and interventions that can make a tangible difference in safeguarding paddy crops and ensuring food security.

In pursuit of our objectives, we are dedicated to pushing the boundaries of knowledge and innovation in the field of paddy crop disease management. By combining rigorous scientific inquiry with practical application, we seek to bridge the gap between research and real-world impact. Our ultimate goal is to contribute to the development of sustainable agricultural practices that promote resilience, productivity, and prosperity in paddy farming communities. As we continue to advance our research, we remain steadfast in our commitment to addressing the evolving challenges of disease management and offering valuable insights that benefit farmers, communities, and global food security efforts alike.

Area recognition in paddy leaf disease detection holds immense significance due to its capability to precisely pinpoint the exact regions affected by diseases like Bacterial Leaf Blight, Leaf Smut, and Brown Spot. This granular understanding enables farmers to accurately assess the extent and severity of crop diseases, laying the foundation for targeted interventions. By integrating this crucial component into our project, we aim to equip farmers with precise information that empowers them to take proactive measures to mitigate the impact of diseases on their crops. Ultimately, this approach contributes to more effective and sustainable paddy crop management practices by minimizing yield losses and optimizing resource utilization.

The ability to accurately recognize affected areas within paddy crops is instrumental in facilitating targeted interventions that address specific disease hotspots. By precisely identifying the regions affected by diseases, farmers can implement localized management strategies such as targeted spraying of pesticides or adopting resistant crop varieties. This not only reduces the overall use of agrochemicals but also minimizes the risk of pesticide resistance and environmental contamination. Moreover, targeted interventions based on area recognition enable farmers to optimize their resources and minimize input costs, thereby enhancing the economic sustainability of paddy crop production systems.

In addition to its practical implications for on-the-ground crop management, area recognition in paddy leaf disease detection also contributes to broader agricultural sustainability goals. By providing farmers with precise information about disease distribution patterns, this approach supports the adoption of integrated pest management (IPM) strategies that prioritize biological, cultural, and mechanical control methods over chemical interventions. This holistic approach to disease management promotes ecological balance, reduces the environmental impact of agricultural practices, and fosters long-term sustainability in paddy crop production. Ultimately, the integration of area recognition into disease detection methodologies represents a significant step forward in achieving the dual objectives of crop protection and environmental stewardship in paddy farming systems.

The integration of machine learning and image processing techniques marks a significant milestone in the realm of paddy crop disease detection, ushering in new possibilities and avenues for innovation. These advanced systems leverage the power of artificial intelligence to offer faster, more objective, and accurate disease diagnosis compared to traditional methods. By analyzing intricate patterns and subtle visual cues in crop images, machine learning algorithms can identify diseases with high precision, providing farmers with timely information crucial for effective decision-making.

Furthermore, the adoption of machine learning and image processing techniques enables farmers to mitigate the impact of diseases on their crops more efficiently. Armed with accurate and timely disease diagnosis, farmers can implement targeted interventions such as adjusting irrigation schedules, applying appropriate fungicides, or deploying resistant crop varieties. These interventions are not only more effective in controlling diseases but also contribute to minimizing yield losses and optimizing resource utilization, thereby improving overall crop health and productivity.

As technology continues to advance, the development of robust and user-friendly disease detection tools tailored specifically for paddy crops holds immense promise for the agricultural sector. By integrating machine learning algorithms into handheld devices or mobile applications, farmers can access real-time disease diagnosis and management recommendations directly in the field. This democratization of technology empowers farmers of all backgrounds and levels of expertise to make informed decisions about disease control measures, ultimately enhancing agricultural productivity and ensuring food security in the face of evolving plant diseases.

In conclusion, the integration of machine learning and image processing techniques represents a transformative shift in paddy crop disease detection, offering unprecedented capabilities for accurate and timely diagnosis. As these technologies continue to evolve and become more accessible, they hold the potential to revolutionize agricultural practices and contribute to global efforts towards sustainable food production. By harnessing the power of artificial intelligence, we can pave the way for a future where farmers are equipped with the tools and knowledge needed to safeguard crop health, ensure food security, and promote agricultural sustainability

1.1 Objectives

Sometimes farmers are forced to sell the crop even though it is contaminated with disease and when consumed by general public results in diseases. So this is where our model comes into action it quickly diagnoses the disease of the plant and provides instant results and solutions which keeps up the quality of crop. Using this model, they were able to determine the leaf disease, affected area in the leaf and the remedies required to cure the disease.

1.2 Methodology

Developing a Machine Learning model that detects paddy crop leaf disease, identifies the affected area in the leaf and suggests remedies to cure the disease.

1.2.1 Dataset

Proper and large dataset is required for all classification research during the training and the testing phase. The dataset for the experiment is downloaded from the Rice Leaf Disease Dataset which contains different plant leaf images and their labels. It contains a collection of images taken at different environment. A dataset containing 1500 leaf images of four classes including healthy leaves is downloaded.

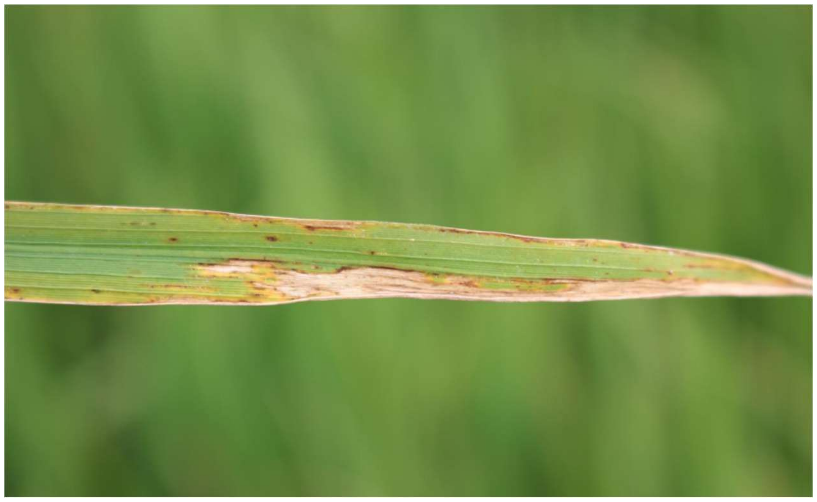


Fig 1.2.1.1 Bacterial Leaf Blight

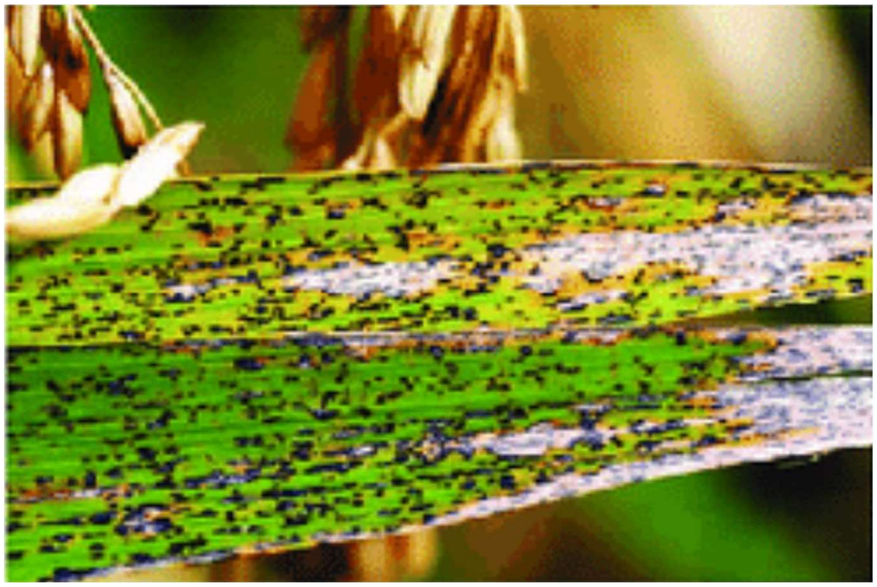


Fig 1.2.1.2 Leaf Smut

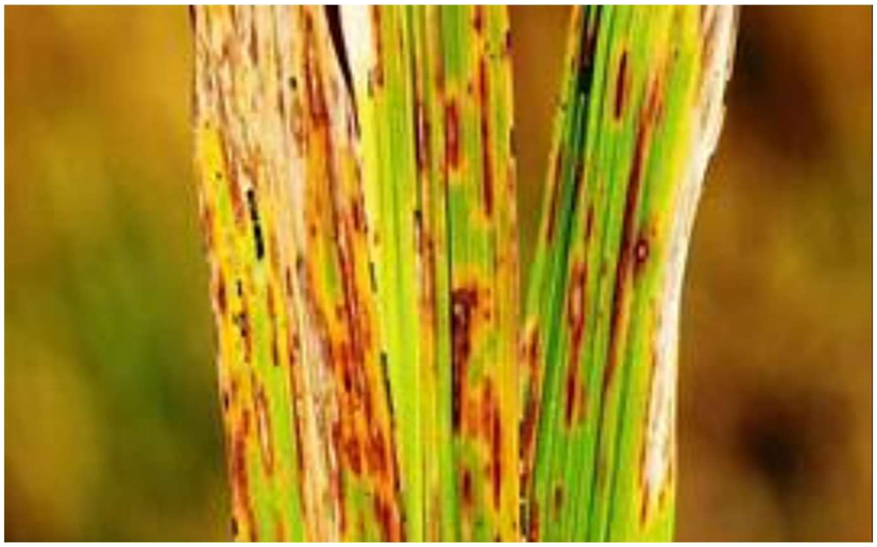


Fig 1.2.1.3 Brown Spot

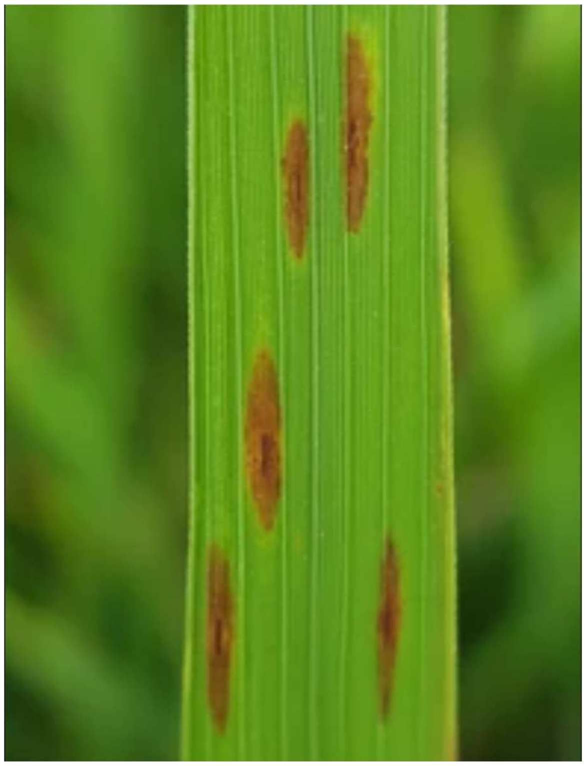


Fig 1.2.1.4 Narrow Brown

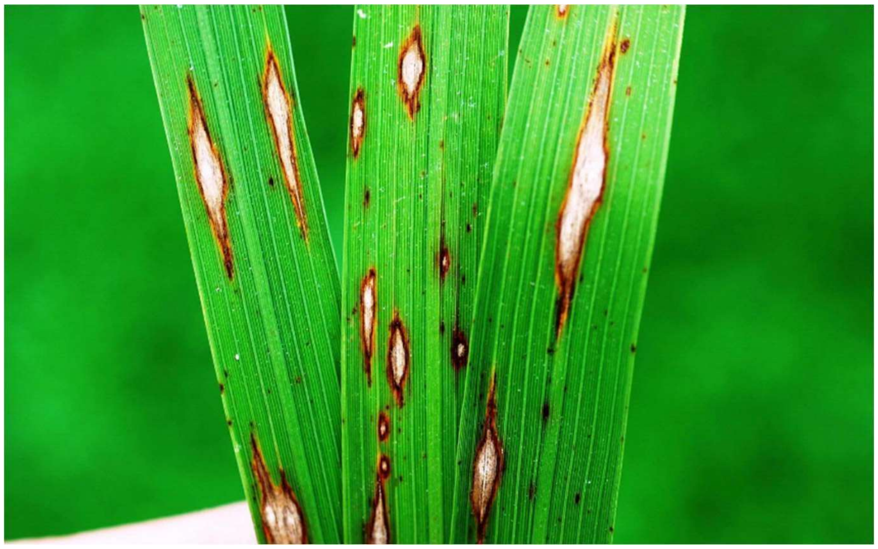


Fig 1.2.1.5 Rice Blast

1.3 Organization of Project

The technique which is developed is taking input as a plant image and compares the uploaded image from the dataset using Support Vector Machine. It identifies the disease of input image and gives remedies as a result.

We have three modules in our project.

- Disease prediction
- Providing remedy
- Calculating the Affected area

2. LITERATURE REVIEW

2.1 Paper – 1: PLDD: A Deep Learning-Based Plant Leaf Disease Detection.

Authors: R. Kavitha Lakshmi, Nickolas Savarimuthu

Summary:

The paper titled "PLDD: A Deep Learning-Based Plant Leaf Disease Detection" addresses the significant impact of plant diseases on food production and the need for automated detection to ensure sustainable agriculture. The authors propose a novel solution using an optimized EfficientDet deep learning framework, trained on 3,038 manually annotated images from two public datasets. The model achieves a mean Average Precision (mAP) of 74.10%, outperforming other state-of-the-art approaches with fewer parameters and computational resources. The framework proves effective in identifying small infected portions on diseased leaves, providing a practical tool for real-time surveillance in large-scale cultivations. The paper concludes with future directions, emphasizing the potential for extending the dataset to include more crop species for comprehensive disease detection.

2.2 Paper – 2: Paddy Crop Disease Detection using Machine Learning

Authors: PrajwalGowda B.S, Nisarga MA, Rachana M,Shashank S, Sahana Raj B.S

Summary:

This research addresses the challenge of detecting and controlling paddy crop diseases, proposing a Machine Learning model based on Convolutional Neural Network (CNN) algorithms. Specifically targeting Rice Blast and Bacterial Blight, the model is trained on a dataset of 2000 images. The CNN architecture, featuring convolutional layers and ReLu activation functions, demonstrates promising results in precision, recall, and F1-Score, with a focus on the EfficientDet-D2 architecture. Comparative analysis with other models underscores its efficiency in mean Average Precision (mAP) and resource utilization. The study suggests the potential for real-time surveillance in large-scale paddy cultivations, with future work aimed at expanding the dataset and enhancing the attention mechanism of the EfficientDet model.

2.3 Paper – 3: Detection and Recognition of Paddy Plant Leaf Diseases using Machine Learning Technique

Authors: Nargis Parven, Muhammad Rashiduzzaman, Nasrin Sultana, Md. Touhidur Rahman, Md. Ismail Jabiullah

Summary:
The research proposes an efficient approach for early detection of paddy plant leaf diseases using image processing and machine learning techniques. Focusing on four diseases—Brown Spot, Blast Disease, Narrow Brown Spot, and Sheath Blight—the system captures and preprocesses images, eliminates unnecessary backgrounds, and employs K-Means Clustering for segmentation. Feature extraction involves Gray-Level Co-occurrence Matrix (GLCM), and disease classification utilizes Support Vector Machine (SVM). The system achieves a commendable accuracy of 94%, outperforming other conventional methods. The methodology, implemented in MATLAB, provides a user-friendly interface for farmers, contributing to improved paddy production and serving as a valuable resource for researchers in image processing and machine learning for agricultural disease detection.

2.4 Paper – 4: Rice Plant Disease Detection and Classification Techniques : A Survey

Authors: Tejas Tawde, Lobhas Verekar, Shailendra Aswale, Kunal Deshmukh, Ajay Reddy et.al

Summary:
This research paper provides a survey on various techniques and briefly discusses significant aspects of different classifiers and techniques used to detect rice diseases. Papers of the last decade are studied thoroughly, including the work carried on various rice plant diseases, and a survey is presented based on essential aspects. The survey focuses to distinguish different methods based on the classifier used. The survey gives insights of the different techniques used for the identification of rice plant disease. In addition, a hardware prototype and model using

Convolutional Neural Network (CNN) is proposed that detect rice disease. It further identifies the type of rice disease into rice blast, rice blight, brown spots, leaf smut, tungro and sheath blight.

2.5 Paper – 5: Crop disease detection using Machine learning: Indian Agriculture

Authors: Anuradha, et al

Summary:
This model depicts how diseases affect yield and how machine learning techniques assisted in detecting the disease and assisting farmers in taking the necessary action. Periodically images are obtained by remote sensing. RGB values of the monitored images are extracted and compared with threshold images. If the threshold is greater or less than given value, histogram analysis and edge detection techniques are used to identify particular plant diseases. They considered wheat crop for their model. They used canny’s edge detection algorithm for the efficient detection of crop disease by taking image of crop].

2.6 Paper – 6: Plant disease detection using machine learning

Authors: Mr Ramachandra Hebbar, Mr. P. V Vinod, Shima Ramesh, Niveditha.M, Pooja R, Shashank.N, Prasad Bhat.N

Summary:
They proposed a model which includes various phases of implementation namely dataset creation, feature extraction,training the classifier and classification. They created datasets of diseased and healthy leaves. Those images are collectively trained under Random Forest for classification. For extracting features of an image they have used Histogram of an Oriented Gradient (HOG). So, finally they concluded that using machine learning to train the large data sets gives a clear way to detect the disease present in plants in a colossal scale.

2.7 Paper – 7: A review on machine learning classification techniques for plant disease detection

Authors: Shruthi U, Nagaveni V, Raghavendra B K, et al

Summary:

They have implemented the model using different techniques. The different classification methods are Artificial Neural Network (ANN) classification technique, K-nearest neighbour classification technique, Convolutional Neural Network classification, fuzzy classifier and Support Vector Machine (SVM) classification methods. From the above mentioned techniques they concluded that Convolution Neural Network provides high accuracy compared to other methods and detect more number of diseases in multiple crops. But the model is unable to identify the affected area in the leaf.

3. REQUIREMENTS

3.1 Software Requirements

Programming Language: Python 3.6

Operating System: Windows 10

IDE: Google Colab

3.2 Hardware Requirements

Processor : Intel Core i5

Hard Disk : 500GB

3.3 Technologies Description

Scikit-learn:

Scikit-Learn, a versatile machine learning library in Python, assumes a pivotal role in our project, particularly in data preparation and the implementation of traditional machine learning models. Leveraging Scikit-Learn's robust functionalities, we ensure the seamless normalization of pixel values within the image data to the standardized range of [0, 1]. This normalization procedure is fundamental, as it guarantees uniform scaling of input features across different datasets, facilitating consistent model performance and interpretation. By adhering to this standard practice, we ensure compatibility with both deep learning and traditional machine learning models, thereby maximizing the flexibility and interoperability of our system.

Moreover, Scikit-Learn's LabelEncoder proves instrumental in streamlining the preprocessing pipeline by transforming textual class labels into numerical representations. This conversion process is essential for training machine learning models, as it enables the algorithms to interpret and learn from the categorical target variables effectively. By encoding class labels into numerical formats, we establish a common ground for model training and evaluation, thereby enhancing the overall efficiency and effectiveness of our disease detection framework.

In addition to data preprocessing, Scikit-Learn facilitates the seamless partitioning of the dataset into training and testing subsets using its train_test_split function. This functionality enables us to stratify the data, ensuring that each class is proportionally represented in both the training and testing sets. By randomizing the data and splitting it into distinct subsets for training and evaluation, we mitigate the risk of overfitting and obtain unbiased estimates of the model's performance on unseen data. This robust splitting mechanism contributes to the reliability and generalizability of our disease detection system.

Furthermore, Scikit-Learn serves as a reliable framework for training traditional machine learning models, such as Support Vector Machines (SVMs), which constitute a cornerstone of our detection methodology. By leveraging Scikit-Learn's implementation of SVMs, we harnes the power of this versatile algorithm to delineate complex decision boundaries and discriminate between different disease classes with high accuracy. Through rigorous model training and optimization, we strive to develop a robust and reliable disease detection model that can effectively identify and classify paddy crop diseases, thereby empowering farmers with actionable insights for proactive crop management and protection.

Keras

Keras plays a pivotal role in our project as a fundamental tool for constructing and training machine learning models. Leveraging its user-friendly and high-level interface, we streamline the process of designing, compiling, and training neural network architectures. Keras abstracts away the complexities of neural network implementation, allowing us to focus on the conceptual aspects of model design and optimization. This intuitive framework facilitates rapid prototyping and experimentation, empowering us to iteratively refine our models for optimal performance.

In our project, we capitalize on the VGG16 model, which is readily available in Keras applications, as a feature extractor. By leveraging the pre-trained weights of VGG16, we tap into its robust capabilities to extract high-level features from input images. This approach enables us to leverage the representational power of VGG16, which has been trained on vast amounts of diverse image data, to effectively capture and encode meaningful patterns relevant to paddy crop disease detection. By integrating VGG16 as a feature extractor within our model architecture, we enhance the model's ability to discern subtle variations and characteristics indicative of different disease states.

Furthermore, Keras facilitates data augmentation through its ImageDataGenerator module, a crucial component for enhancing the generalization capability of our models. By augmenting the training dataset with synthesized variations of the original images, such as rotations, shifts, and flips, we introduce additional diversity and variability into the training samples. This augmentation process helps mitigate overfitting by exposing the model to a wider range of potential input variations, thereby improving its ability to generalize to unseen data. Through Keras's seamless integration of data augmentation techniques, we enhance the robustness and adaptability of our disease detection model to real-world scenarios and environmental conditions.

Overall, Keras serves as an indispensable tool in our project, providing a comprehensive framework for model construction, training, and evaluation. By harnessing its capabilities for feature extraction, data augmentation, and model training, we strive to develop a sophisticated and reliable disease detection system that can effectively address the challenges of paddy crop management and contribute to agricultural sustainability and food security.

VGG16 Model

In our project, VGG16 emerges as a cornerstone for feature extraction, leveraging its pre-trained convolutional neural network architecture available within the Keras framework. Specifically, we utilize the convolutional base of VGG16, excluding its fully connected layers, to extract hierarchical and meaningful features from the input images. This approach allows us to tap into the network's inherent ability to capture complex patterns and features inherent in visual data, facilitating the extraction of rich representations that encapsulate relevant information about paddy crop diseases.

By leveraging the pre-trained weights of VGG16 obtained from the ImageNet dataset, we harness the model's extensive knowledge and understanding of various patterns and features in images. The transfer of this pre-learned knowledge enables us to bootstrap the feature extraction process, capitalizing on the insights gained from training on a diverse array of image data. Consequently, the features extracted by VGG16 encapsulate a broad spectrum of visual characteristics, ranging from basic shapes to intricate textures, which are instrumental in discerning subtle differences indicative of different disease states in paddy crops.

Moreover, the hierarchical nature of features extracted by VGG16 allows us to capture both low-level details and high-level semantic information inherent in the input images. By traversing multiple convolutional layers, the network progressively abstracts and consolidates information, thereby forming a rich hierarchy of features that encode increasingly complex visual concepts. This hierarchical representation facilitates more nuanced and discriminative feature encoding, enhancing the model's ability to discern subtle variations and patterns relevant to paddy crop disease detection.

Overall, the integration of VGG16 for feature extraction represents a strategic decision in our project, enabling us to leverage the power of pre-trained convolutional neural networks for extracting rich and informative representations from input images. By harnessing the pre-learned knowledge encoded within VGG16's weights, we enhance the model's ability to understand and interpret visual data, thereby advancing the state-of-the-art in paddy crop disease detection and contributing to the advancement of agricultural sustainability and food security initiatives.

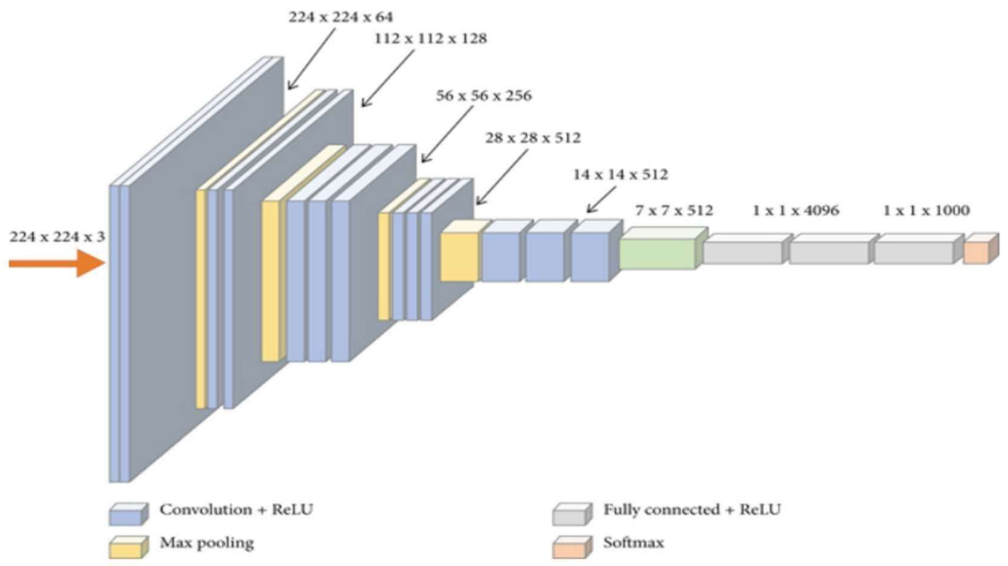


Fig 3.3.1 VGG16 Model Architecture

OpenCV

OpenCV emerges as a crucial tool in our project for image preprocessing, serving as a fundamental component in the initial stages of data preparation. We rely on OpenCV's robust functionalities to efficiently load and manipulate images from the dataset, ensuring consistency and integrity in the input data. By leveraging OpenCV for reading images, we mitigate potential inconsistencies and formatting issues, thereby laying a solid foundation for subsequent processing steps. This initial preprocessing step is essential for establishing a standardized data pipeline and facilitating seamless integration with downstream machine learning algorithms.

Furthermore, OpenCV plays a pivotal role in standardizing the dimensions of input images by resizing them to a uniform size of 224x224 pixels. This resizing operation is particularly crucial for compatibility with the VGG16 model, which expects input images of a specific dimension. By leveraging OpenCV's resizing capabilities, we ensure that all images in the dataset conform to the required dimensions, thereby enabling seamless integration with the feature extraction process using VGG16. This standardized preprocessing step enhances the efficiency and effectiveness of subsequent model training and evaluation tasks.

Additionally, OpenCV serves as a valuable tool for visualizing randomly selected images from the dataset, providing valuable insights into the characteristics and composition of the input data. By leveraging OpenCV's visualization capabilities, we gain a comprehensive understanding of the diversity and distribution of images within the dataset, enabling us to identify potential patterns or anomalies that may influence model performance. This exploratory analysis step is instrumental in informing subsequent preprocessing and model development decisions, ensuring the robustness and generalizability of our disease detection system.

Overall, OpenCV's versatile functionalities for image loading, manipulation, resizing, and visualization play a critical role in streamlining the image preprocessing pipeline in our project. By leveraging OpenCV as a foundational tool, we establish a standardized and efficient workflow for preparing input data, thereby laying the groundwork for the development of accurate and reliable machine learning models for paddy crop disease detection.

4. DESIGN

4.1 Introduction

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word “Quality”. Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer’s view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective.

4.2 Architecture Diagram

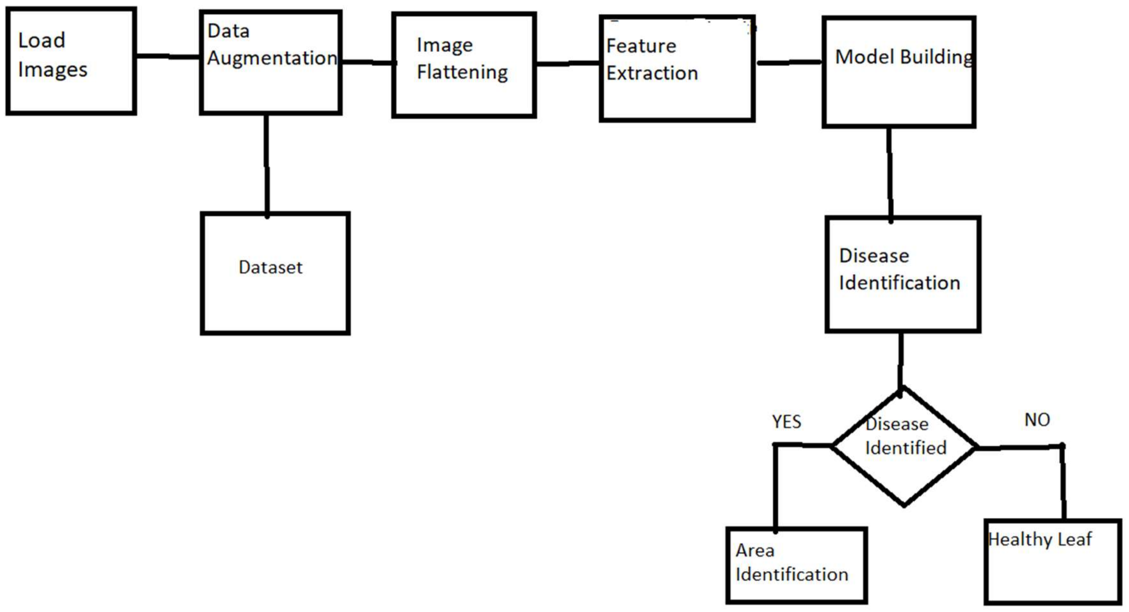


Fig 4.2.1 Architecture Diagram

1. **Load Images:**

In the initial step of our image preprocessing pipeline, we access the dataset containing images of three distinct types of diseases along with healthy leaf images from Kaggle. Leveraging the OpenCV library, specifically the 'cv2.imread' function, we efficiently load the images from the dataset directory. Following the loading process, we ensure uniformity in image dimensions by resizing them to a fixed size of 224x224 pixels using the 'cv2.resize' function. This standardization of image sizes is crucial for compatibility with subsequent processing steps and ensures consistency across the dataset.

Moreover, to facilitate optimal model performance, we normalize the pixel values of the images by dividing them by 255.0, effectively scaling them to fall within the range of [0, 1]. This normalization step is integral for enhancing the convergence and stability of machine learning algorithms during training, as it minimizes the influence of variations in pixel intensity across different images. By bringing all pixel values within a standardized range, we mitigate potential biases and facilitate more effective learning from the input data.

Additionally, to prepare the dataset for model training, we encode the categorical labels associated with each Image using scikit-learn's 'LabelEncoder'. This transformation converts textual class labels into numerical representations, thereby enabling the machine learning algorithms to interpret and learn from the target variable effectively.

Furthermore, to ensure randomness and reduce potential biases during model training, we shuffle the dataset using the 'shuffle' function from scikit-learn. This shuffling operation randomizes the order of samples within the dataset, thereby preventing any inherent ordering effects from influencing the learning process.

In summary, our preprocessing pipeline involves loading images from the dataset, resizing them to a fixed size, normalizing pixel values, encoding categorical labels, and shuffling the dataset to ensure randomness. These preparatory steps lay a solid foundation for subsequent model training and evaluation, enabling us to develop robust and reliable machine learning models for paddy crop disease detection.

2. Data Augmentation:

The image preprocessing pipeline in our project extends beyond simple loading and resizing of images. We incorporate advanced techniques such as data augmentation to enhance the diversity and richness of our dataset, thereby facilitating the training of a more robust model for paddy crop disease detection.

Through techniques like rotation and flipping, we introduce variations in the training data, effectively expanding its breadth and scope. This augmentation process not only increases the variability of the dataset but also improves the model's ability to generalize to unseen data by exposing it to a wider range of potential input variations.

Furthermore, the visualization of augmented images provides valuable insights into the transformations applied to the original dataset. By visualizing the augmented images, we gain a deeper understanding of the impact of data augmentation techniques on image appearance and structure. This visualization step allows us to assess the effectiveness of augmentation strategies in introducing meaningful variations while preserving the integrity of the underlying data. Additionally, it serves as a qualitative validation of our preprocessing pipeline, ensuring that the augmented dataset accurately reflects the diversity and complexity of real-world scenarios encountered in paddy crop disease detection.

Overall, the integration of data augmentation techniques and visualization of augmented images enriches our dataset and enhances the robustness of our model for detecting paddy crop diseases. By augmenting the training data with variations in rotation and flipping, we increase the model's exposure to diverse input patterns, leading to improved generalization performance.
Additionally, the visualization of augmented images provides valuable qualitative feedback on the effectiveness of our preprocessing pipeline, ensuring that our dataset accurately captures the complexities of real-world scenarios encountered in paddy crop disease detection.

3. **Image Flattening:**

After extracting the feature maps from the pre-trained VGG16 model, the next critical step in our pipeline involves reshaping these feature maps into a flat feature vector. This transformation is accomplished using the 'reshape()' function, which reconfigures the multi-dimensional feature maps into a one-dimensional array.

By flattening the feature maps, we convert them into a compact and structured representation that is suitable for input into a classifier. This step is essential as it prepares the extracted features for further processing and analysis, enabling us to leverage them effectively for disease classification tasks.

Reshaping the feature maps into a flat feature vector facilitates seamless integration with subsequent classification algorithms, allowing us to exploit the rich information encoded within the feature maps for accurate disease detection.

By representing the extracted features in a standardized format, we ensure compatibility with a wide range of classifier architectures, thereby enhancing the flexibility and versatility of our disease detection pipeline. Additionally, the flat feature vector representation enables efficient computation and manipulation of the extracted features, optimizing the overall performance and scalability of our model.

Overall, reshaping the extracted feature maps into a flat feature vector represents a crucial preprocessing step in our pipeline, enabling us to harness the discriminative power of the VGG16 model for disease detection tasks. By converting the multi-dimensional feature maps into a compact and structured representation, we facilitate seamless integration with classification algorithms, thereby laying the groundwork for accurate and efficient disease classification.

This transformation enhances the interpretability, compatibility, and scalability of our disease detection pipeline, ultimately contributing to the development of a robust and effective solution for paddy crop disease detection.

4. **Feature Extraction**:

   In our image processing pipeline, we leverage the VGG16 model from TensorFlow's Keras applications to extract informative features from preprocessed images. We instantiate the VGG16 model without including the fully connected layers by using the 'VGG16' function, which allows us to focus solely on leveraging its convolutional base for feature extraction.

   By excluding the fully connected layers, we retain the convolutional layers responsible for learning hierarchical representations of visual features, which are instrumental for our disease detection task. This streamlined approach enables us to efficiently harness the pre-trained VGG16 model's capacity to capture complex patterns and structures in the input images.

   Subsequently, we utilize the 'base_model.predict' function to extract features from the preprocessed images using the pre-trained VGG16 model. This process involves passing the preprocessed images through the convolutional layers of the VGG16 model to obtain feature maps, which encode rich information about the visual characteristics of the input images.

   These extracted features serve as high-level representations of the underlying image content, capturing essential patterns and structures relevant to paddy crop diseases. By leveraging the pre-trained weights of the VGG16 model, which have been learned from a diverse range of image data, we benefit from its capacity to discern meaningful features across different domains.

   Once the features are extracted, we flatten them into feature vectors for each image using the 'reshape' function. This reshaping operation converts the multi-dimensional feature maps into a one-dimensional array, creating a concise and structured representation of the extracted features. These feature vectors encapsulate the essential information extracted by the VGG16 model, facilitating subsequent processing and analysis. By transforming the extracted features into a standardized format, we prepare them for input into downstream classification algorithms, enabling us to leverage them effectively for disease detection tasks. This process ensures compatibility and interoperability with various machine learning models, paving the way for accurate and efficient paddy crop disease detection.

5. Model Building:

The SVM (Support Vector Machine) classifier is used for disease classification
Grid search is performed using ‘GridSearchCV’ to find the best hyperparameters for the SVM model.

The best SVM model is obtained from the grid search results.

Accuracy, classification report, and confusion matrix are computed for evaluating the SVM model’s performance.

The Decision Tree classifier and K-Nearest Neighbors (KNN) classifier are also used for disease classification.

Accuracy and confusion matrix are calculated for the Decision Tree and KNN models.

6. Disease Identification: -

In the final stages of our project pipeline, we transition from model training to practical application by demonstrating the disease identification process on new images. After training the Support Vector Machine (SVM) model on the preprocessed dataset, we validate its effectiveness by loading an example image, preprocessing it, and passing it through the trained model for disease prediction. The predicted disease label is then transformed back to its original name using the 'label encoder' inverse transform function, providing interpretable results for end-users. Subsequently, the predicted disease name is displayed alongside the visualized image, offering a clear and intuitive representation of the model's diagnosis.

Moreover, our project goes beyond mere disease identification by incorporating additional functionalities, such as identifying the level of area affected in the leaf. This feature provides valuable insights into the severity of the detected disease, enabling stakeholders to prioritize and tailor remedial actions accordingly. Additionally, remedies specific to each disease can be printed based on the predicted disease name, offering actionable recommendations to mitigate the impact of identified diseases on paddy crops. By encompassing these comprehensive functionalities, our project serves as a practical and versatile tool for disease management in agricultural settings, empowering stakeholders with actionable insights and targeted interventions to safeguard crop health and optimize yields. Overall, our project exemplifies a holistic pipeline for preprocessing dataset, extracting features using the VGG16 model, training and evaluating multiple classifiers, and deploying the trained model for real-world disease identification, thereby bridging the gap between research and practical application in the field of paddy crop disease management.

5.Testing

Model evaluation is the process of assessing the performance and effectiveness of a trained machine learning model using various metrics and techniques. It helps determine how well the model generalizes to unseen data and how accurately it predicts the target labels.

Model evaluation is performed to compare the performance of the four algorithms (SVM, KNN) in classifying paddy crop leaf diseases. The accuracy of each algorithm is calculated and compared to determine which algorithm performs the best.

SVM obtained the highest accuracy among the four algorithms. This indicates that SVM achieved the highest percentage of correct predictions on the test set compared to the other algorithms. This result suggests that SVM is the most effective algorithm for classifying paddy crop leaf diseases in the given dataset.

The usefulness of this finding is that it provides insights into the performance of different algorithms for the specific task of crop leaf disease classification. By identifying the algorithm with the highest accuracy, we can choose that algorithm for deployment in real-world applications. Additionally, knowing the accuracy of each algorithm allows us to make informed decisions about the choice of algorithm based on the desired level of accuracy and other evaluation metrics

1 .Import necessary libraries:

'cv2': OpenCV library for image processing.

'numpy': Library for numerical operations.

'cv2 imshow': A utility function from Google Colab for displaying images.

2.Load the leaf image:

The 'cv2.imread()' function is used to read the image from the specified file path.

3.Convert the image to HSV color space:

The 'cv2.cvtColor()' function is used to convert the image from the BGR color space (default in OpenCV) to the HSV color space.

This conversion allows for better color-based segmentation.

4.Define the lower and upper color thresholds for disease-affected areas:

The lower and upper color values define a range of HSV values that represent the color characteristics of the disease-affected areas in the image. These values need to be adjusted based on the specific characteristics of the disease being detected.

5.Threshold the image to get disease-affected areas:

The 'cv2.inRange()' function is used to create a binary mask, where white pixels represent the disease-affected areas and black pixels represent the rest of the image, based on the defined color thresholds.

6.Extract disease features from the image:

Once we have isolated the disease-affected areas using the binary mask, we can proceed to extract relevant features that describe the characteristics of the disease. These features may include texture, shape, and size properties of the affected regions. Extracting these features allows us to represent the disease in a more abstract and informative way, facilitating the classification process.

7.Preprocess the extracted features:

Before feeding the extracted features into the classification algorithms, it is crucial to preprocess them to ensure they are in a suitable format and scale for the machine learning model. Common preprocessing steps include normalization, scaling, and feature selection to remove any redundant or irrelevant information. Proper preprocessing enhances the performance and efficiency of the classification algorithms by improving the quality of the input data.

8.Split the dataset into training and testing sets:

To evaluate the performance of the classification algorithms, we divide the dataset into separate training and testing sets. The training set is used to train the machine learning models, while the testing set is used to assess their performance on unseen data. This separation helps us estimate how well the models generalize to new instances and avoid overfitting, where the model memorizes the training data but fails to generalize to new examples.

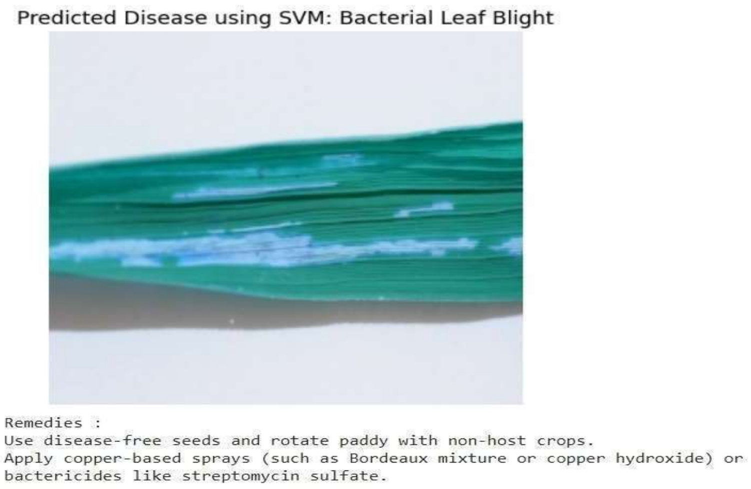
9.Train the classification algorithms:

With the dataset prepared, we can proceed to train the classification algorithms, including SVM and KNN, on the training data. During training, the algorithms learn patterns and relationships between the input features and the corresponding target labels, which in this case represent different types of paddy crop leaf diseases. The training process aims to optimize the model parameters to minimize prediction errors and improve overall accuracy.

10.Evaluate the performance of the algorithms:

Once the models are trained, we evaluate their performance using the testing set. We measure various metrics such as accuracy, precision, recall, and F1-score to assess how well the models classify disease-affected areas. By comparing the performance of different algorithms, we can identify the most effective approach for classifying paddy crop leaf diseases and make informed decisions about deploying the model in real-world applications. Additionally, we may fine-tune the model parameters or explore ensemble methods to further improve classification accuracy and robustness.

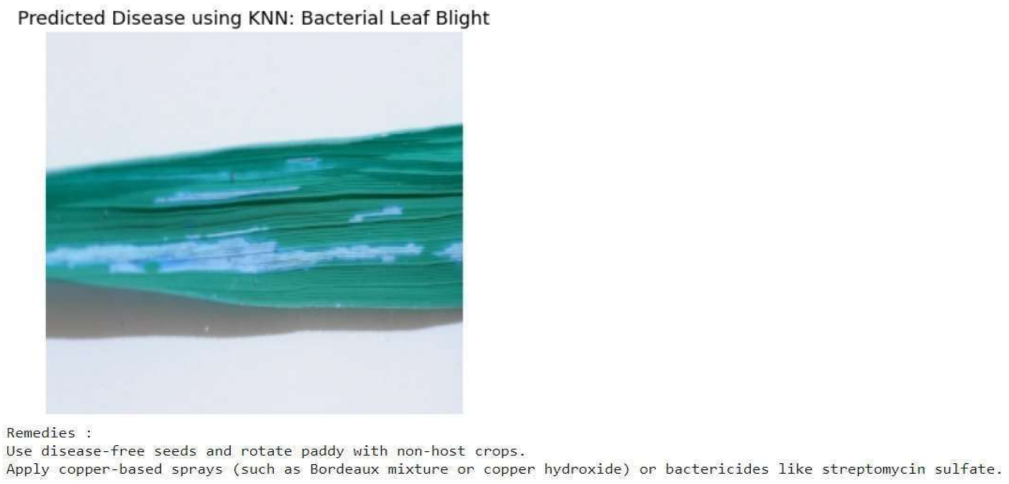
5.2 OUTPUT SCREENSHOTS



```
Predicted Disease using SVM: Bacterial Leaf Blight
```

```
Remedies :
Use disease-free seeds and rotate paddy with non-host crops.
Apply copper-based sprays (such as Bordeaux mixture or copper hydroxide) or
bacterticides like streptomycin sulfate.
```

Fig 5.3.1 Prediction of Bacterial Leaf Blight using SVM

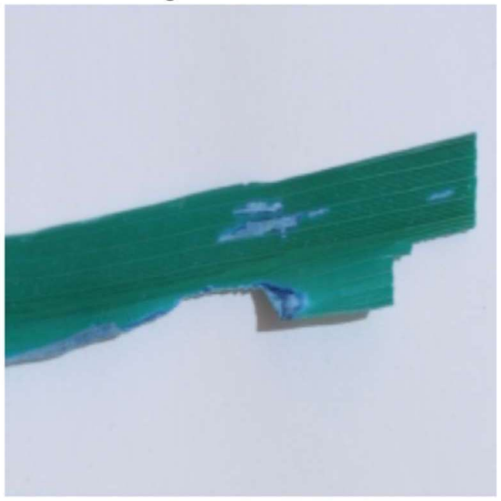


```
Predicted Disease using KNN: Bacterial Leaf Blight
```

```
Remedies :
Use disease-free seeds and rotate paddy with non-host crops.
Apply copper-based sprays (such as Bordeaux mixture or copper hydroxide) or bactericides like streptomycin sulfate.
```

Fig 5.3.2 Prediction of Bacterial Leaf Blight using KNN

```
Predicted Disease using Random Forest: Bacterial Leaf Blight
```



```
Remedies :
Use disease-free seeds and rotate paddy with non-host crops.
Apply copper-based sprays (such as Bordeaux mixture or copper hydroxide) or bactericides like streptomycin sulfate.
```

Fig 5.3.3 Prediction of Bacterial Leaf Blight using Random Forest

```
Total affected Area in mm^2 : 13648.830000000002 mm^2
Total affected area in pixels : 1055472.5 pixels
Total affected area in percentage: 38.19115396737004 %
```

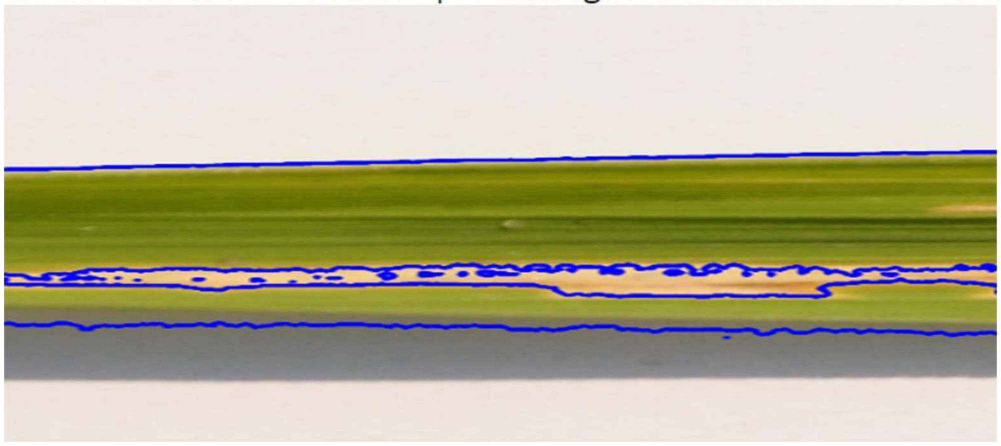
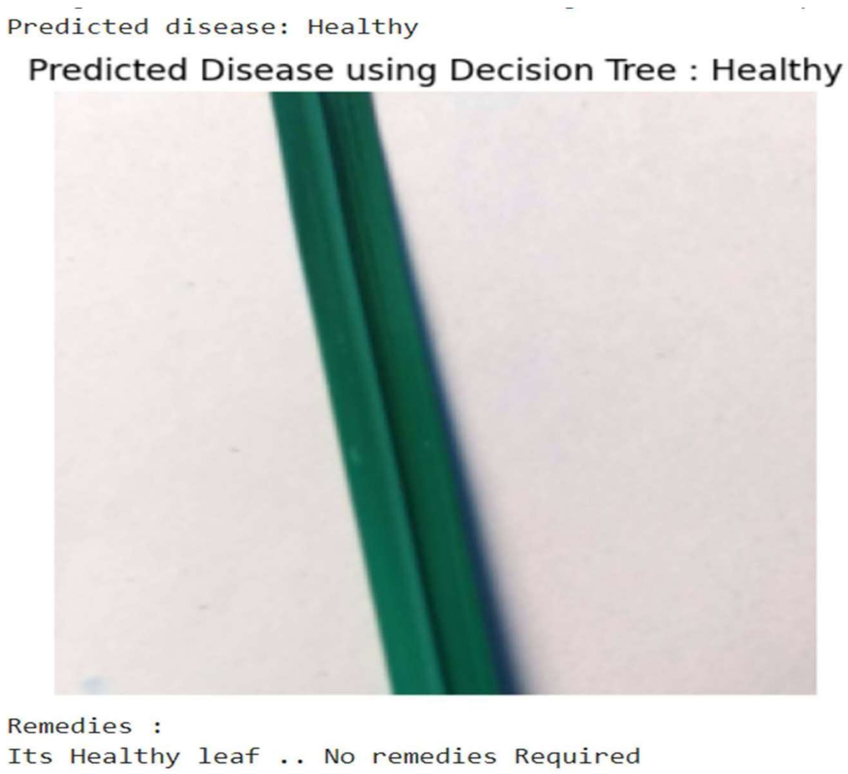


Fig 5.3.4 Area Intensity Detection for Bacterial Blight Disease Image

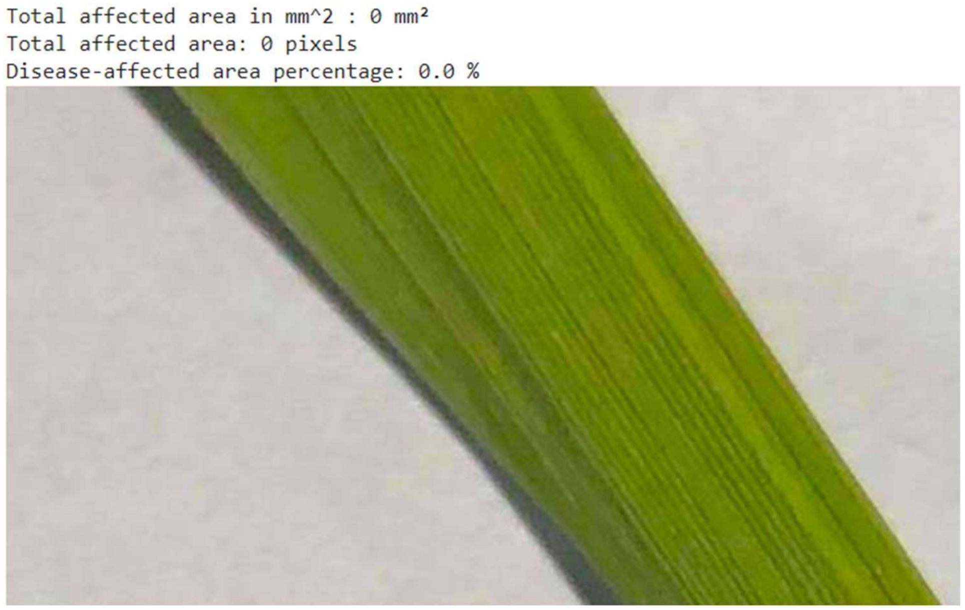


```
Predicted disease: Healthy
```

Predicted Disease using Decision Tree : Healthy

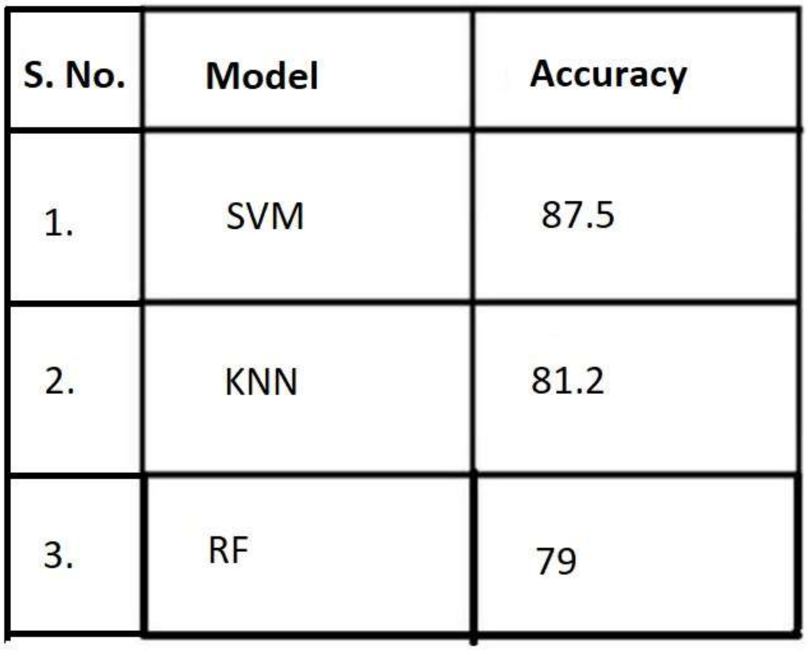
```
Remedies :
Its Healthy leaf .. No remedies Required
```

Fig 5.3.5 Disease Detection for Healthy Leaf Image



```
Total affected area in mm^2 : 0 mm²
Total affected area: 0 pixels
Disease-affected area percentage: 0.0 %
```

Fig 5.3.6 Area Intensity Detection for Healthy Leaf Image



S. No.	Model	Accuracy
1.	SVM	87.5
2.	KNN	81.2
3.	RF	79

Fig 5.3.7 Model Comparison

5. CONCLUSION AND FUTURE SCOPE

As farmers are facing issues with disease identification, level of the disease and unable to find effective remedies to cure the diseases. To solve the issues, a machine learning model is developed that detects paddy crop leaf disease and calculates the affected area. For better accuracy, the model is trained with different algorithms such as Support Vector Machine (SVM), K-Nearest Neigbor(KNN). Among all these SVM gives highest accuracy. Based on the disease identified, remedies are suggested. The remedies provide proper information regarding the pesticide or insecticide to be used in order to cure the disease.

In Future, there is scope for developing models capable of identifying multiple diseases simultaneously. This would provide a comprehensive assessment of crop health and enable farmers to implement holistic disease management strategies.

6. REFERENCES

[1] R. Kavitha Lakshmi, Nickolas Savarimuthu "PLDD- A Deep Learning Based Plant Leaf Disease Detection" Journal published by Institute of Electrical and Electronics Engineers (IEEE) 2021.

[2] PrajwalGowda B.S, Nisarga M A, Rachana M, Shashank S, Mrs. Sahana Raj B.S "Paddy Crop Disease Detection using Machine Learning" Journal published by International Journal of Engineering Research Technology (IJERT) 2020.

[3] Md. Touhidur Rahman, Md. Ismail Jabiullah, Nargis Parven, Muhammad Rashiduzzaman "Detection and Recognition of Paddy Plant Leaf Diseases using Machine Learning Technique" International Journal of Innovative Technology and Exploring Engineering (IJITEE) 2020.

[4] Shruti U, Nagaveni V, Raghavendra B K, "A review on machine learning classification techniques for plant disease detection" International Conference on Advanced Computing & Communication Systems (ICACCS), 2021.

[5] V. Vanitha "Rice disease detection using machine learning", International Journal of Recent Technology and Engineering, IJRTE, Feb 2020.

[6] Anuradha Badge, "Crop disease detection using Machine learning: Indian Agriculture", IJRTE, 2019

[7] Mr Ramachnadra Hebbar, Mr.P.V Vinod, Shima Ramesh, Niveditha.M, Pooja R, Shashank.N, Prasad Bhat.N, "Plant disease detection using machine learning", IEEE, 2018.

[8] Nargis Parven, Muhammad Rashiduzzaman, Nasrin Sultana, Md. Touhidur Rahman, Md.Ismail Jabiullah, "Detection and Recognition of Paddy Plant Leaf Diseases using Machine Learning Technique", IJIEE, 2020.

[9] Tejas Tawde, Lobhas Verekar, Shailendra Aswale, Kunal Deshmukh, Ajay Reddy "Rice Plant Disease Detection and Classification Techniques : A Survey", IJERT, 2021.

[10] Priyanka B Raj, Dr Neha Mangal, Pooja R ,Soumya G Hegde "Paddy leaf disease detection using image processing and machine learning", IJRTE, 2019.

[11] Tenzin Chokey, Sarika Jain "Quality assessment of crops using machine learning technique", IJRTE, 2019

[12] Anjali K, Mrs Divya unni, Arya M.S, "Detection of unhealthy plant leaves using image processing and genetic algorithm with arduino", ICPSCC, 2018.

[13] Manish Sah, Taohidul Islam,Rudra Roy Choudhry,Sudipto Baral "A faster technique on rice disease detection using image processing of affected area in agro-field", ICICCT, 2018

Appendices

```python
#Data Preprocessing

import os
import cv2
import numpy as np


dataset_path = "/content/drive/MyDrive/extracted_dataset"

images = []
labels = []

class_to_disease = {
    "Bacterial leaf blight": "Bacterial Leaf Blight",
    "Brown spot": "Brown Spot",
    "Leaf smut": "Leaf Smut",
    "Narrow Brown Spot" :
    "Narrow Brown Spot",
    "Rice Blast" : "Rice
    Blast",
    "Healthy": "Healthy"

}

for class_folder in os.listdir(dataset_path):
    class_path = os.path.join(dataset_path, class_folder)
    if os.path.isdir(class_path):
        for image_file in os.listdir(class_path):
            image_path = os.path.join(class_path, image_file)
            image = cv2.imread(image_path)
            if image is not None:
                image = cv2.resize(image, (224, 224))
                images.append(image)
                labels.append(class_to_disease[class_folder])

images = np.array(images)
labels = np.array(labels)

disease_names = np.unique(labels)

for i, disease_name in enumerate(disease_names):
    print("Disease Name:", disease_name, " - Label:", i)



# Normalize pixel values to [0, 1]
images = images.astype('float32') / 255.0
label_encoder = LabelEncoder()
labels = label_encoder.fit_transform(labels)
```

```
# Split the dataset into training and testing sets

images, labels = shuffle(images, labels, random_state = 42)
train_images, test_images, train_labels, test_labels =
train_test_split(
    images, labels, test_size=0.2, random_state=42)


import matplotlib.pyplot as plt

label_to_class = {label_idx: class_name for label_idx, class_name in
enumerate(label_encoder.classes_)}

fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(8, 8))
axes = axes.ravel()

for i in range(9):
    rand_index = np.random.randint(0, len(images))
    axes[i].imshow(images[rand_index])
    label_idx = labels[rand_index]
    class_name = label_to_class[label_idx]
    axes[i].set_title(f"{class_name}-{label_idx}")
    axes[i].axis('off')

plt.tight_layout()
plt.show()

#Feature Extraction

from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Model

base_model = VGG16(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))

# Extract features from the images using the VGG16 model
train_features = base_model.predict(train_images)
test_features = base_model.predict(test_images)


# Data augmentation
datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    vertical_flip=True,
    fill_mode='nearest'
)

datagen.fit(train_images)

# Visualize augmented images for the first training sample
img = train_images[0].reshape((1,) + train_images[0].shape)

augmented_images = []
for _ in range(9):
    augmented_img = next(datagen.flow(img, batch_size=1))[0]
    augmented_images.append(augmented_img)

fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(8, 8))
axes = axes.ravel()

axes[0].imshow(img[0])
axes[0].set_title("Original")
axes[0].axis('off')

for i in range(1, 9):
    axes[i].imshow(augmented_images[i-1])
    axes[i].set_title(f"Augmented {i}")
    axes[i].axis('off')

plt.tight_layout()
plt.show()

#Model Training using SVM

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

# Flatten the feature vectors
train_features_flat = train_features.reshape(train_features.shape[0], -
1)

# Initialize an SVM classifier
svm = SVC()

param_grid = {'C': [1, 10, 100], 'gamma': [0.1, 0.01, 0.001]}


grid_search = GridSearchCV(svm, param_grid, cv=5)
grid_search.fit(train_features_flat, train_labels)

# Get the best SVM model
```

```
svm_model = grid_search.best_estimator_
# Train the SVM model
svm_model.fit(train_features_flat, train_labels)

#Model Evaluation using SVM

from sklearn.metrics import accuracy_score,
classification_report,confusion_matrix

# Reshape the test features
test_features_flat = test_features.reshape(test_features.shape[0], -1)

# Predict labels for the test set
predicted_labels = svm_model.predict(test_features_flat)

# Calculate accuracy and other evaluation metrics
accuracy = accuracy_score(test_labels, predicted_labels)
report = classification_report(test_labels, predicted_labels,
target_names=label_encoder.classes_)

print("Accuracy:", accuracy)
print(confusion_matrix(test_labels,predicted_labels))

cm = confusion_matrix(test_labels, predicted_labels)
classes = np.unique(test_labels)

# Plot the confusion matrix
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Greens)
plt.title('Confusion Matrix')
plt.colorbar()
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes)
plt.yticks(tick_marks, classes)
# Label the matrix with the counts
thresh = cm.max() / 2.0
for i, j in np.ndindex(cm.shape):
    plt.text(j, i, cm[i, j], ha='center', va='center', color='white' if
cm[i, j] > thresh else 'black')

# Add axis labels
plt.xlabel('Predicted Label')
plt.ylabel('Test Label')

# Show the plot
plt.show()
```

```
#SVM Mdoel Prediction

from skimage.transform import resize

image_path = '/content/drive/MyDrive/DSC_0372.JPG'
img = cv2.imread(image_path)
resized_img = resize(img, (224, 224))
test_feature = base_model.predict(np.expand_dims(resized_img, axis=0))
test_feature_flat = test_feature.reshape(1, -1)

predicted_label = svm_model.predict(test_feature_flat)

predicted_disease_name =
label_encoder.inverse_transform(predicted_label)[0]

print("Predicted disease:",predicted_disease_name)
plt.imshow(resized_img)
plt.axis('off')
plt.title("Predicted Disease using SVM: " + predicted_disease_name)
plt.show()


print("Remedies : ")
if(predicted_disease_name == "Bacterial Leaf Blight"):
     print('Use disease-free seeds and rotate paddy with non-host
crops.\nApply copper-based sprays (such as Bordeaux mixture or copper
hydroxide) or \nbactericides like streptomycin sulfate.')
if(predicted_disease_name == "Brown Spot"):
     print("Plant paddy at the recommended time and use resistant
varieties.\nApply fungicides such as propiconazole or tebuconazole as
preventive measures or at the onset of symptoms.")
if(predicted_disease_name == "Healthy"):
     print("Its Healthy leaf .. No remedies Required")
if(predicted_disease_name == "Leaf Smut"):
     print("Treat paddy seeds with recommended fungicides like
carbendazim or thiophanate-methyl before planting.\nApply fungicides
during the early stages of the crop as a preventive measure against
leaf smut.")

#Model Training using KNN

from sklearn.neighbors import KNeighborsClassifier

train_features_flat = train_features.reshape(train_features.shape[0], -
1)
test_features_flat = test_features.reshape(test_features.shape[0], -1)

knn = KNeighborsClassifier(n_neighbors=5)
```

```
knn.fit(train_features_flat, train_labels)

knn_model = knn.fit(train_features_flat, train_labels)

accuracy_knn = knn.score(test_features_flat, test_labels)
print("Accuracy:", accuracy_knn)
print(confusion_matrix(test_labels,predicted_labels))
kcm = confusion_matrix(test_labels, predicted_labels)
classes = np.unique(test_labels)

# Plot the confusion matrix
plt.imshow(kcm, interpolation='nearest', cmap=plt.cm.Greens)
plt.title('Confusion Matrix')
plt.colorbar()
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes)
plt.yticks(tick_marks, classes)
# Label the matrix with the counts
thresh = kcm.max() / 2.0
for i, j in np.ndindex(kcm.shape):
    plt.text(j, i, kcm[i, j], ha='center', va='center', color='white'
if kcm[i, j] > thresh else 'black')

# Add axis labels
plt.xlabel('Predicted Label')
plt.ylabel('Test Label')

# Show the plot
plt.show()


#KNN Model Prediction


from skimage.transform import resize

image_path = '/content/drive/MyDrive/DSC_0113.jpg'
img = cv2.imread(image_path)
resized_img = resize(img, (224, 224))
test_feature = base_model.predict(np.expand_dims(resized_img, axis=0))
test_feature_flat = test_feature.reshape(1, -1)

predicted_label = knn_model.predict(test_feature_flat)

predicted_disease_name =
label_encoder.inverse_transform(predicted_label)[0]

print("Predicted disease:",predicted_disease_name)
plt.imshow(resized_img)
plt.axis('off')
plt.title("Predicted Disease using KNN: " + predicted_disease_name)
plt.show()


print("Remedies : ")
if(predicted_disease_name == "Bacterial Leaf Blight"):
     print('Use disease-free seeds and rotate paddy with non-host
crops.\nApply copper-based sprays (such as Bordeaux mixture or copper
hydroxide) or bactericides like streptomycin sulfate.')
if(predicted_disease_name == "Brown Spot"):
     print("Plant paddy at the recommended time and use resistant
varieties.\nApply fungicides such as propiconazole or tebuconazole as
preventive measures or at the onset of symptoms.")
if(predicted_disease_name == "Healthy"):
     print("Its Healthy leaf .. No remedies Required")
if(predicted_disease_name == "Leaf Smut"):
     print("Treat paddy seeds with recommended fungicides like
carbendazim or \nthiophanate-methyl before planting.\nApply fungicides
during the early stages of the crop as a preventive \nmeasure against
leaf smut.")
```

```
from sklearn.ensemble import RandomForestClassifier

train_features_flat = train_features.reshape(train_features.shape[0], -1)
test_features_flat = test_features.reshape(test_features.shape[0], -1)

rf = RandomForestClassifier(n_estimators=100, random_state=42)

rf.fit(train_features_flat, train_labels)
rf_model = rf.fit(train_features_flat, train_labels)

accuracy_r = rf.score(test_features_flat, test_labels)
print("Accuracy:", accuracy_r)
print(confusion_matrix(test_labels,predicted_labels))

rdf = confusion_matrix(test_labels, predicted_labels)
classes = np.unique(test_labels)

# Plot the confusion matrix
plt.imshow(rdf, interpolation='nearest', cmap=plt.cm.Greens)
plt.title('Confusion Matrix')
plt.colorbar()
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes)
plt.yticks(tick_marks, classes)
# Label the matrix with the counts
thresh = rdf.max() / 2.0
for i, j in np.ndindex(kcm.shape):
    plt.text(j, i, rdf[i, j], ha='center', va='center', color='white' if rdf[i,
j] > thresh else 'black')

# Add axis labels
plt.xlabel('Predicted Label')
plt.ylabel('Test Label')

# Show the plot
plt.show()
```

```
from skimage.transform import resize

image_path = '/content/drive/MyDrive/extracted_dataset/Narrow Brown
Spot/narrow_brown-109-_jpg.rf.3370f11672d9c2224338e1bed4c204f9.jpg'
img = cv2.imread(image_path)
resized_img = resize(img, (224, 224))
test_feature = base_model.predict(np.expand_dims(resized_img, axis=0))
test_feature_flat = test_feature.reshape(1, -1)

predicted_label = rf_model.predict(test_feature_flat)

predicted_disease_name = label_encoder.inverse_transform(predicted_label)[0]

print("Predicted disease:",predicted_disease_name)
plt.imshow(resized_img)
plt.axis('off')
plt.title("Predicted Disease using Random Forest: " + predicted_disease_name)
plt.show()


print("Remedies : ")
if(predicted_disease_name == "Bacterial Leaf Blight"):
     print('Use disease-free seeds and rotate paddy with non-host crops.\nApply
copper-based sprays (such as Bordeaux mixture or copper hydroxide) or
bactericides like streptomycin sulfate.')
if(predicted_disease_name == "Brown Spot"):
     print("Plant paddy at the recommended time and use resistant
varieties.\nApply fungicides such as propiconazole or tebuconazole as preventive
measures or at the onset of symptoms.")
if(predicted_disease_name == "Healthy"):
     print("Its Healthy leaf .. No remedies Required")
if(predicted_disease_name == "Leaf Smut"):
     print("Treat paddy seeds with recommended fungicides like carbendazim or
thiophanate-methyl before planting.\nApply fungicides during the early stages of
the crop as a preventive measure against leaf smut.")

if(predicted_disease_name == "Narrow Brown Spot"):
    print("Use balanced nutrients; make sure that adequate potassium is used and
spray propiconazole at booting to heading stages.")
```

```
    print("Accuracy (SVM):", accuracy)
    print("Accuracy (KNN): ", accuracy_knn) from
    tabulate import tabulate

    # Example accuracy values (replace with your own) accuracy_values =
    {
    'SVM': accuracy, 'KNN':
    accuracy_knn
    }

    # Prepare the data for the table table_data =
    []
for model, accuracy in accuracy_values.items():
    table_data.append([model, accuracy])

    # Set the headers for the table
    headers = ['Model', 'Accuracy']

    # Generate the table
    table = tabulate(table_data, headers, tablefmt='fancy_grid')

    # Print the table print(table)



    #area intensity detection for bacterila leaf blight disease

    image
```

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

# Load the leaf image
image = cv2.imread('/content/drive/MyDrive/DSC_0372.JPG')

# Convert the image to HSV color space
hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

# Defining the lower and upper color thresholds for disease-affected areas
lower_color = np.array([20, 50, 50])  # Adjusting these values according
to  specific disease characteristics
upper_color = np.array([40, 255, 255])  # Adjusting these values according to
specific disease characteristics

# Threshold the image to get disease-affected areas
mask = cv2.inRange(hsv, lower_color, upper_color)
```

```
# Apply a morphological operation to improve the mask
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (7, 7))
mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)

#contour_image = np.zeros_like(image)
# Find contours in the mask
contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Draw contours on the original image
cv2.drawContours(image, contours, -1, (255, 0, 0), 6)  # Draw blue contours around
disease-affected areas

# Calculate the total leaf area
total_area = image.shape[0] * image.shape[1]
#print(total_area)
# Initialize a variable to store the total affected area
affected_area = 0

# Iterate over the contours
for contour in contours:
    # Calculate the area of each contour
    area = cv2.contourArea(contour)

    x, y, w, h = cv2.boundingRect(contour)
    length = h
    width = w

    # Assuming you have the pixel resolution or scale factor in mm/pixel
    pixel_resolution_mm = 0.1  # Pixel resolution in mm/pixel

    # Calculate the length and width in millimeters
    length_mm = length * pixel_resolution_mm
    width_mm = width * pixel_resolution_mm

    # Update the total affected area
    affected_area += area

   # Calculate the affected area in mm^2
    affected_area_mm2 = length_mm * width_mm

# Print the length and width of the affected area
#print("Length:  {} mm".format(length_mm))
#print("Width: {} mm" .format(width_mm))

# Print the affected area
print("Total affected Area in mm^2 : {} mm^2".format(affected_area_mm2))

# Print the total affected area
print("Total affected area in pixels :", affected_area, "pixels")
```

```
# Calculate the affected area percentage
affected_area_percentage = (affected_area / total_area) * 100

# Print the affected area percentage
print("Total affected area in percentage:", affected_area_percentage, "%")

Disease_Area  = (affected_area_percentage / 100) * total_area

# Display the image with disease-affected areas highlighted
cv2_imshow(image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
    #area intensity detection for brownspot disease image
```

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow


# Load the leaf image
image = cv2.imread('/content/drive/MyDrive/DSC_0113.jpg')

# Convert the image to HSV color space
hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)



# Defining the lower and upper color thresholds for disease-affected areas
lower_color = np.array([0, 50, 50])  # Adjusting these values according to
specific disease characteristics
upper_color = np.array([20, 255, 255])  # Adjusting these values according
to  specific disease characteristics

# Threshold the image to get disease-affected areas
mask = cv2.inRange(hsv, lower_color, upper_color)

# Apply a morphological operation to improve the mask
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (7, 7))
mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)

#contour_image = np.zeros_like(image)

# Find contours in the mask
contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Draw contours on the original image
cv2.drawContours(image, contours, -1, (255, 0, 0), 6)  # Draw blue contours around
disease-affected areas
```

```python
# Calculate the total leaf area
total_area = image.shape[0] * image.shape[1]

# Initialize a variable to store the total affected area
affected_area = 0

# Iterate over the contours
for contour in contours:
    # Calculate the area of each contour
    area = cv2.contourArea(contour)
    x, y, w, h = cv2.boundingRect(contour)
    length = h
    width = w

    # Assuming you have the pixel resolution or scale factor in mm/pixel
    pixel_resolution_mm = 0.1  # Pixel resolution in mm/pixel

    # Calculate the length and width in millimeters
    length_mm = length * pixel_resolution_mm
    width_mm = width * pixel_resolution_mm

    # Update the total affected area
    affected_area += area

    # Calculate the affected area in mm^2
    affected_area_mm2 = length_mm * width_mm

# Print the affected area
print("Total affected Area in mm^2 : {} mm^2".format(affected_area_mm2))

# Print the total affected area
print("Total affected area in pixels :", affected_area, "pixels")

# Calculate the affected area percentage
affected_area_percentage = (affected_area / total_area) * 100

# Print the affected area percentage
print("Total affected area in percentage:", affected_area_percentage, "%")

# Display the image with disease-affected areas highlighted
cv2_imshow(image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
    #area intensity detection for leaf smut disease image

import cv2
import numpy as np
from google.colab.patches import cv2_imshow


# Load the leaf image
image = cv2.imread('/content/drive/MyDrive/DSC_0504.jpg')

# Convert the image to HSV color space
hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)



# Defining the lower and upper color thresholds for disease-affected areas
lower_color = np.array([0, 255, 50])  # Adjust these values according to your
specific disease characteristics
upper_color = np.array([180, 255, 255])  # Adjust these values according to your
specific disease characteristics

# Threshold the image to get disease-affected areas
mask = cv2.inRange(hsv, lower_color, upper_color)




# Apply a morphological operation to improve the mask
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5))
mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)

#contour_image = np.zeros_like(image)

# Find contours in the mask
contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Draw contours on the original image
cv2.drawContours(image, contours, -1, (255, 0, 0), 6)  # Draw blue contours around
disease-affected areas

# Calculate the total leaf area
total_area = image.shape[0] * image.shape[1]

# Initialize a variable to store the total affected area
affected_area = 0

# Iterate over the contours
for contour in contours:
    # Calculate the area of each contour
    area = cv2.contourArea(contour)
    x, y, w, h = cv2.boundingRect(contour)
    length = h
    width = w

    # Assuming you have the pixel resolution or scale factor in mm/pixel
    pixel_resolution_mm = 0.1  # Pixel resolution in mm/pixel

    # Calculate the length and width in millimeters
    length_mm = length * pixel_resolution_mm
    width_mm = width * pixel_resolution_mm

    # Update the total affected area
    affected_area += area

    # Calculate the affected area in mm^2
    affected_area_mm2 = length_mm * width_mm

# Print the affected area
print("Total affected Area in mm^2 : {} mm^2".format(affected_area_mm2))

# Print the total affected area
print("Total affected area in pixels:", affected_area, "pixels")

# Calculate the affected area percentage
affected_area_percentage = (affected_area / total_area) * 100

# Print the affected area percentage
print("Total affected area in percentage:", affected_area_percentage, "%")

# Display the image with disease-affected areas highlighted
cv2_imshow(image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
     #area intensity detection for healthy image
```

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow


# Load the leaf image
image = cv2.imread('/content/drive/MyDrive/IMG_3233.jpg')

# Convert the image to HSV color space
hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
```

```
# Defining the lower and upper color thresholds for disease-affected areas
lower_color = np.array([0, 50, 50])  # Adjust these values according to your
specific disease characteristics
upper_color = np.array([10, 255, 255])  # Adjust these values according to your
specific disease characteristics

# Threshold the image to get disease-affected areas
mask = cv2.inRange(hsv, lower_color, upper_color)




# Apply a morphological operation to improve the mask
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (7, 7))
mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)

#contour_image = np.zeros_like(image)

# Find contours in the mask
contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Draw contours on the original image
cv2.drawContours(image, contours, -1, (255, 0, 0), 6)  # Draw blue contours around
disease-affected areas

# Calculate the total leaf area
total_area = image.shape[0] * image.shape[1]

# Initialize a variable to store the total affected area
affected_area = 0

# Iterate over the contours
for contour in contours:
    # Calculate the area of each contour
    area = cv2.contourArea(contour)


    # Update the total affected area
    affected_area += area

total_affected_area = 0
print("Total affected area in mm^2 :", total_affected_area, "mm²")


# Print the total affected area
print("Total affected area:", affected_area, "pixels")

# Calculate the affected area percentage
affected_area_percentage = (affected_area / total_area) * 100

# Print the affected area percentage
```

```
print("Disease-affected area percentage:", affected_area_percentage, "%")

# Display the image with disease-affected areas highlighted
cv2_imshow(image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
    #area intensity for narrow brown spot disease image
```

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

# Load the leaf image
image = cv2.imread('/content/drive/MyDrive/extracted_dataset/Narrow Brown
Spot/narrow_brown-120-_jpg.rf.e861f0a2050113eb7306e99631d08335.jpg')

# Convert the image to HSV color space
hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

# Defining the lower and upper color thresholds for disease-affected areas
lower_color = np.array([20, 50, 50])  # Adjusting these values according
to  specific disease characteristics
upper_color = np.array([40, 255, 255])  # Adjusting these values according to
specific disease characteristics

# Threshold the image to get disease-affected areas
mask = cv2.inRange(hsv, lower_color, upper_color)

# Apply a morphological operation to improve the mask
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (7, 7))
mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)

#contour_image = np.zeros_like(image)
# Find contours in the mask
contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Draw contours on the original image
cv2.drawContours(image, contours, -1, (255, 0, 0), 2)  # Draw blue contours around
disease-affected areas

# Calculate the total leaf area
total_area = image.shape[0] * image.shape[1]
#print(total_area)
# Initialize a variable to store the total affected area
affected_area = 0

# Iterate over the contours
for contour in contours:
    # Calculate the area of each contour
    area = cv2.contourArea(contour)

    x, y, w, h = cv2.boundingRect(contour)
    length = h
    width = w

    # Assuming you have the pixel resolution or scale factor in mm/pixel
    pixel_resolution_mm = 0.1  # Pixel resolution in mm/pixel

    # Calculate the length and width in millimeters
    length_mm = length * pixel_resolution_mm
    width_mm = width * pixel_resolution_mm

    # Update the total affected area
    affected_area += area

   # Calculate the affected area in mm^2
    affected_area_mm2 = length_mm * width_mm

# Print the length and width of the affected area
#print("Length:  {} mm".format(length_mm))
#print("Width: {} mm" .format(width_mm))

# Print the affected area
print("Total affected Area in mm^2 : {} mm^2".format(affected_area_mm2))

# Print the total affected area
print("Total affected area in pixels :", affected_area, "pixels")

# Calculate the affected area percentage
affected_area_percentage = (affected_area / total_area) * 100

# Print the affected area percentage
print("Total affected area in percentage:", affected_area_percentage, "%")

Disease_Area  = (affected_area_percentage / 100) * total_area

# Display the image with disease-affected areas highlighted
cv2_imshow(image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
    # area intensity for rice blast disease image

import cv2
import numpy as np
from google.colab.patches import cv2_imshow

# Load the leaf image
image = cv2.imread('/content/drive/MyDrive/extracted_dataset/Rice Blast/Rice
blast_17.png')

# Convert the image to HSV color space
hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

# Defining the lower and upper color thresholds for disease-affected areas
lower_color = np.array([10, 50, 50])  # Adjusting these values according
to  specific disease characteristics
upper_color = np.array([40, 255, 255])  # Adjusting these values according to
specific disease characteristics

# Threshold the image to get disease-affected areas
mask = cv2.inRange(hsv, lower_color, upper_color)

# Apply a morphological operation to improve the mask
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (7, 7))
mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)

#contour_image = np.zeros_like(image)
# Find contours in the mask
contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Draw contours on the original image
cv2.drawContours(image, contours, -1, (255, 0, 0), 2)  # Draw blue contours around
disease-affected areas

# Calculate the total leaf area
total_area = image.shape[0] * image.shape[1]
#print(total_area)
# Initialize a variable to store the total affected area
affected_area = 0

# Iterate over the contours
for contour in contours:
    # Calculate the area of each contour
    area = cv2.contourArea(contour)

    x, y, w, h = cv2.boundingRect(contour)
    length = h
    width = w
    # Assuming you have the pixel resolution or scale factor in mm/pixel
    pixel_resolution_mm = 0.1  # Pixel resolution in mm/pixel

    # Calculate the length and width in millimeters
    length_mm = length * pixel_resolution_mm
    width_mm = width * pixel_resolution_mm

    # Update the total affected area
    affected_area += area

   # Calculate the affected area in mm^2
    affected_area_mm2 = length_mm * width_mm

# Print the length and width of the affected area
#print("Length:  {} mm".format(length_mm))
#print("Width: {} mm" .format(width_mm))

# Print the affected area
print("Total affected Area in mm^2 : {} mm^2".format(affected_area_mm2))

# Print the total affected area
print("Total affected area in pixels :", affected_area, "pixels")

# Calculate the affected area percentage
affected_area_percentage = (affected_area / total_area) * 100

# Print the affected area percentage
print("Total affected area in percentage:", affected_area_percentage, "%")

Disease_Area  = (affected_area_percentage / 100) * total_area

# Display the image with disease-affected areas highlighted
cv2_imshow(image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```