Guava Fruit Disease Classification and Prediction

UIT2502 – DATA ANALYTICS AND VISUALIZATION

A PROJECT REPORT

Guided by

Dr. V. Durga Devi

Assistant Professor, Department of IT.

Submitted by

R.Nithyasri 3122 22 5002 086
R.Nitish 3122 22 5002 088

SSN COLLEGE OF ENGINEERING,
KALAVAKKAM

NOVEMBER 2024

Sri Sivasubramaniya Nadar College of Engineering (An Autonomous Institution, Affiliated to Anna University)

BONAFIDE CERTIFICATE

Certified that this project titled "Guava Fruit Disease Prediction and Classification" is the bonafide work of "R.Nithyasri – 3122 22 5002 086 and R.Nitish – 3122 22 5002 088 and is submitted for project viva-voce examination held on XX.11.2024.

Signature of examiner(s)

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to S. Mohanavalli who gave us the opportunity to work on this project. A special thanks to Dr. V. Durga Devi and Dr.A.Saravanan for their constant guidance and support throughout the project.

Secondly,we would like to thank our parents,relatives and friends who constantly supported and encouraged us to complete this project within the limited time frame.

We are highly obliged to all those who guided and helped us out in completing this project successfully.

TABLE OF CONTENTS

CLIENT CERTIFICATE

Name of the project : Guava Fruit Disease Classification and Prediction

Members:

1. R.Nithyasri – 3122 22 5002 086
2. R.Nitish – 3122 22 5002 088

Client details :

Rating System – 1 : Strongly disagree 2:Disagree 3: Neutral 4: Agree 5: Strongly Agree

Questions	**1**	**2**	**3**	**4**	**5**
The problem was well discussed and, the requirements and goals were clear.					
The project plan was well defined and communicated from the start.					
The resources were adequate for achieving the goals.					
The original timeline was realistic and was followed.					
The teamwork was well demonstrated.					
The client was communicated on regular intervals and given updates on the progress of the report.					
The expected project requirements have been satisfied.					

CHAPTER 1

PROBLEM DESCRIPTION

1.1 Statement of problem and scope

In modern agricultural research and disease management, analyzing plant health through image-based disease classification presents significant challenges, such as handling the complexity of visual data, inefficient analysis workflows, and limited insights into the impact of various diseases on crop yield and quality. Researchers and agricultural practitioners often face difficulties in accurately processing and interpreting plant disease images, which can lead to incomplete analyses, difficulty in identifying disease patterns, and delayed interventions.

Manual image analysis of plant diseases is often error-prone, inconsistent, and time-consuming, limiting researchers' ability to gain accurate insights into crop health and effectively manage diseases such as phytophthora, scab, and root rot in guava plants. The lack of an integrated system complicates visualization, disease detection, and real-time health assessment, making collaboration and workflow optimization challenging.

This project aims to address these issues by developing an automated **Guava Disease Prediction System**. Using image preprocessing techniques such as **Image Augmentation, RGB Channel Extraction**, and usage of **Image-based Deep Learning Models** which include **Convolutional Neural Networks** for efficient classification of the diseased from the non-diseased, the system will enable efficient disease classification and real-time health assessment. By streamlining data analysis, enabling real-time processing, and enhancing feature extraction, the project will support early disease detection, improve crop management, and facilitate better decision-making in guava farming and disease prevention.

1.2 Scope:

- Real-time image acquisition and preprocessing of guava leaves and fruit.
- Detection and classification of disease indicators on leaves and fruit, including symptoms of Phytophthora, Scab, Root Rot diseases(on guava fruit) and Red Rust(on guava leaf).
- Extraction of relevant colour and texture features (e.g., RGB values) from images to facilitate disease classification.
- Implementation of machine learning models to classify disease presence and type, assisting in early detection and treatment recommendations.

1.3 Limitations:

- Limited to specific guava diseases (Phytophthora, Scab, and Root Rot) and Guava leaf diseases(and may not generalize to other diseases or plant types.
- Dependent on image quality and environmental conditions (e.g., lighting, camera quality) for accurate disease classification.
- Tested primarily on pre-existing datasets rather than extensively validated in real field conditions.

CHAPTER 2

DESIGN OF AI SOLUTION AND DATA VISUALIZATION

2.1. Overview of the Model Architecture

The AI solution utilizes advanced signal processing techniques combined with machine learning algorithms to analyze ECG (Electrocardiogram) signals and detect anomalies such as Premature Ventricular Contractions (PVCs) or other irregularities. The architecture integrates feature extraction using filtering techniques and classification via machine learning models, specifically designed to recognize patterns in ECG signals. This setup allows the system to automatically detect abnormal heart rhythms in real-time.

2.2. Key Components

2.2.1 Image Preprocessing

- **Description:** Raw images often vary in quality, lighting, and resolution. The first step in the system is to preprocess these images, including resizing, grayscale conversion, and noise reduction.

- **Function:** This preprocessing step normalizes the images, enhancing key features and preparing them for consistent feature extraction. Techniques such as resizing, colour normalization, and filtering are applied to improve the reliability of subsequent analysis.

2.2.2 Feature Extraction

- **Description:** To classify diseases accurately, relevant features are extracted from the pre-processed images. These features include RGB colour averages, texture patterns, and shape descriptors that help in identifying disease-specific characteristics.

- **Function:** By analysing the colour (e.g., RGB values) and texture of leaves and fruit, the system can detect symptoms of diseases such as lesions, rust, or mold patterns. This extracted information serves as the input for the classification model.

2.2.3 Classification Model

- **Description:** A Convolutional Neural Network (CNN) is trained on a labelled dataset of diseased and healthy guava images to classify the extracted features into categories: healthy, Phytophthora, Scab Red Rust or Root Rot.

- **Function**: The classification model takes the extracted features and predicts the presence or type of disease. The model is trained using labelled data to learn the visual patterns that distinguish each disease.

2.2.4 Anomaly Detection Using Thresholding

- **Description:** After classification, thresholding techniques are applied to detect significant deviations in colour or texture patterns associated with diseased leaves and fruits**.**

- **Function:** If an image's feature values exceed a predefined threshold, it is flagged as diseased. This allows the system to automatically identify images that deviate from healthy leaf and fruit patterns, highlighting them for further inspection.

2.2.5 Post-Processing and Visualization

- **Description:** After disease detection, the system provides visualization that highlights affected areas and displays the classification results.

- **Function:** The processed image data can be visualized, marking diseased areas and labelling images with disease classifications. This visualization helps agricultural experts or farmers to quickly assess the health of guava plants and take appropriate action.

2.3 Training Process

2.3.1 Training with Labelled Image Data

- **Description:** The machine learning model is trained on a labelled dataset of guava leaf and fruit images, where each image is annotated with information about the type of disease or health status.

- **Function:** The model learns to identify patterns within the images that correspond to healthy or diseased states (e.g., Phytophthora, Scab, or Root Rot). This supervised learning process provides both the features and corresponding labels, enabling the model to distinguish between different conditions.

2.3.2 Feature Selection and Optimization

- **Description:** During training, an important step is to select the most informative features from the image data. Techniques such as Principal Component Analysis (PCA) or feature selection algorithms may be used to optimize and refine feature sets.

- **Function:** This process reduces dimensionality and improves the model's performance by focusing on the most relevant information, leading to more accurate disease detection and classification.

2.3.3 Model Evaluation

- **Description:** After training, the model's performance is evaluated on a test set of guava images that were not used during training. Metrics such as accuracy, precision, recall, and F1-score are used to assess the model's effectiveness in detecting and classifying diseases.

- **Function:** Model evaluation ensures that the system performs well in recognizing diseases, minimizing false positives (healthy plants classified as diseased) and false negatives (diseased plants classified as healthy). This step is crucial for validating the model's reliability in real-world applications.

2.4 Inference and Detection

2.4.1 Image Segmentation

- **Description:** For effective disease detection, guava leaf and fruit images are segmented into relevant sections (e.g., diseased and non-diseased regions).

- **Function**: Each segment is analysed independently, enabling the model to detect disease in specific areas of the image. Segmentation makes the model suitable for detailed and precise analysis, especially in cases where disease is localized.

2.4.2 Disease Scoring

- **Description:** During inference, each segmented image is passed through the trained model to calculate a disease score.

- **Function:** The model assigns a score that reflects how likely a segment is to be diseased. Higher scores indicate a greater likelihood of disease presence, such as Phytophthora, Scab, or Root Rot.

2.4.3 Thresholding for Real-Time Alerts

- **Description:** If the disease score of a segment exceeds a predefined threshold, it triggers an alert, indicating a potentially diseased area.

- **Function:** The threshold-based approach enables timely alerts, allowing farmers or agricultural experts to take action promptly. This step minimizes false alarms and ensures that only significant signs of disease are flagged.

2.5 Graphical Visualization for Disease Monitoring

- **Description:** The system generates a visual representation of the guava leaf or fruit image, marking diseased segments and showing corresponding disease scores.

- **Function:** The visualization highlights diseased regions for easy assessment, allowing farmers or experts to quickly locate and evaluate the disease severity. The output is designed to be intuitive, with options to zoom into specific areas for closer inspection.

2.6 Limitations

- **High Sensitivity to Image Quality:** Images may contain noise or variations due to lighting, which can impact the system's accuracy despite preprocessing efforts.

- **Dependence on Quality Training Data:** The performance of the model relies on the quality and diversity of the training data. If the dataset does not represent all possible disease conditions, the model may struggle to generalize well.
- **Real-Time Processing Challenges:** While the model is designed for timely detection, continuous analysis in field environments may require substantial computational resources, which can be challenging in low-resource settings.

2.7 Future Extensions

2.7.1 Multi-Class Classification for Detailed Disease Identification

- **Description:** The model could be extended to classify not only disease presence but also identify specific disease types (e.g., Phytophthora, Scab, Root Rot) with more precision.

- **Function:** This would provide farmers with detailed insights into the specific type of disease affecting the crop, enabling more accurate and tailored treatment options.

2.7.2 Integration with Drone or IoT-Based Monitoring for Large-Scale Detection

- **Description:** The system could be integrated with drones or IoT-based devices for real-time, large-scale monitoring of guava orchards.

- **Function:** This would enable continuous monitoring and early detection across larger areas, allowing for timely intervention and reducing the impact of disease spread on crops.

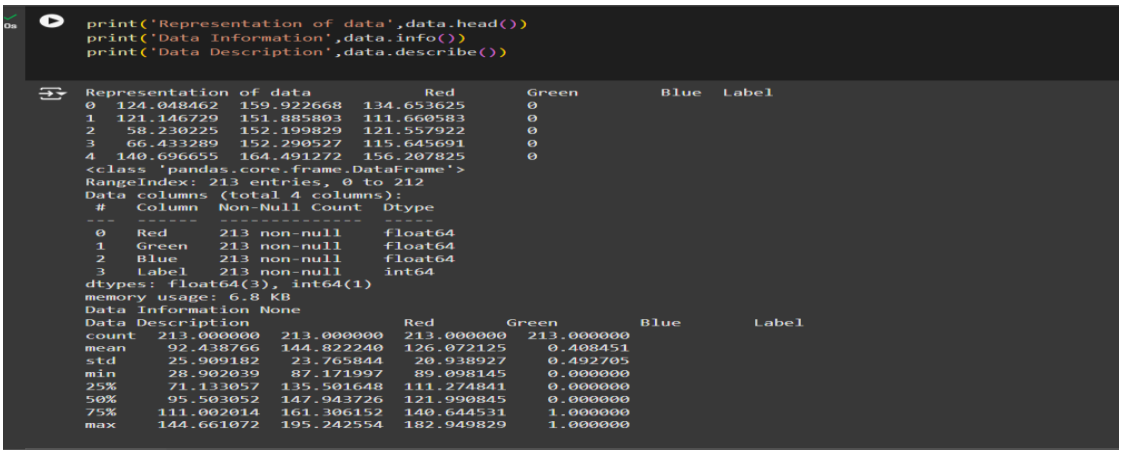
2.8 Data Visualization for Disease Monitoring

2.8.1 Extraction of RGB Values from the Image Dataset:

- Numeric data consisting of the RGB Values from the images of the dataset (pertaining to Red Rust and Healthy Guava Leaves) were deduced and compiled into a Comma Separated Values(.csv) file for perform statistical and data visualization and observe inferences and create an analytics dashboard as part of the software solution.

2.8.2 Exploratory Data Analysis (EDA)

- **Description:**
  With the RGB dataset in hand, Exploratory Data Analysis (EDA) is conducted to uncover patterns, trends, and relationships within the data. EDA is used to understand the distribution of RGB values across different disease categories of Guava Leaves(Healthy or Red Rust affected leaves),identify outliers, and visualize the data to gain insights into potential disease indicators.



```
print('Representation of data',data.head())
print('Data Information',data.info())
print('Data Description',data.describe())
```

```
Representation of data          Red       Green        Blue  Label
0  124.048462  159.922668  134.653625      0
1  121.146729  151.885803  111.660583      0
2   58.230225  152.199829  121.557922      0
3   66.433289  152.290527  115.645691      0
4  140.696655  164.491272  156.207825      0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 213 entries, 0 to 212
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Red     213 non-null    float64
 1   Green   213 non-null    float64
 2   Blue    213 non-null    float64
 3   Label   213 non-null    int64
dtypes: float64(3), int64(1)
memory usage: 6.8 KB
Data Information None
Data Description                Red       Green        Blue       Label
count  213.000000  213.000000  213.000000  213.000000
mean    92.438766  144.822240  126.072125    0.408451
std     25.900182   23.765844   20.938927    0.492705
min     28.902039   87.171997   89.098145    0.000000
25%     71.133057  135.501648  111.274841    0.000000
50%     95.503052  147.943726  121.990845    0.000000
75%    111.002014  161.306152  140.644531    1.000000
max    144.661072  195.242554  182.949829    1.000000
```

Fig: 2.8.2(a) Description of the dataset as observed on Google Colab

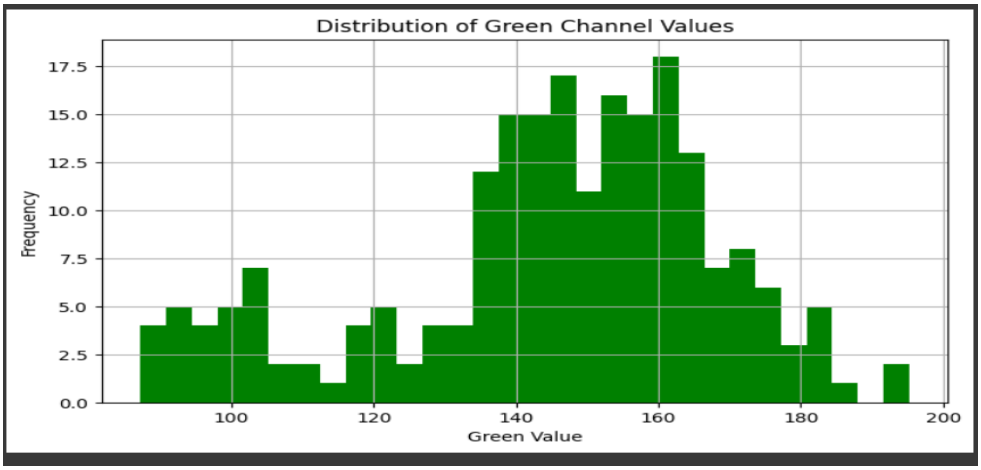


Fig: 2.8.2(b)(1) Histogram plotted on the trends of Green Value obtained from the images.

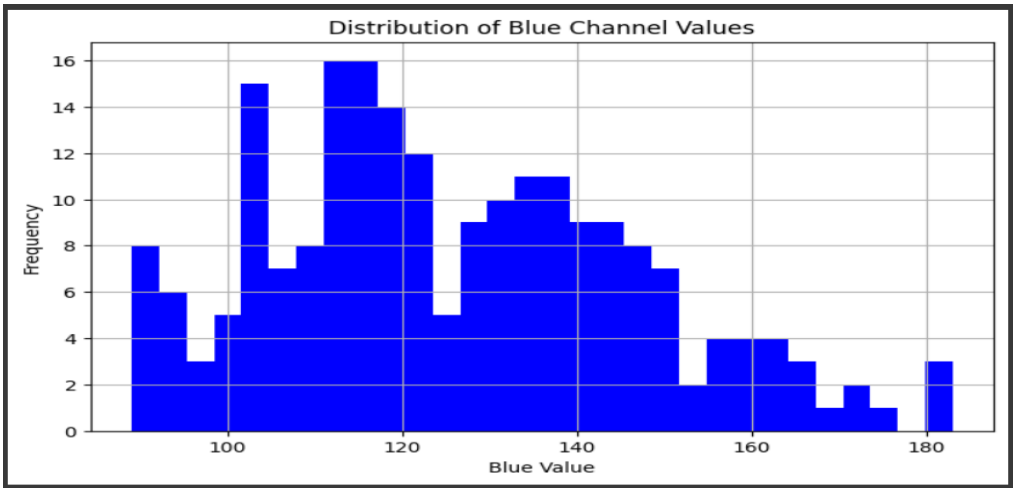


Fig: 2.8.2(b)(2) Histogram plotted on the trends of Blue Value obtained from the images.

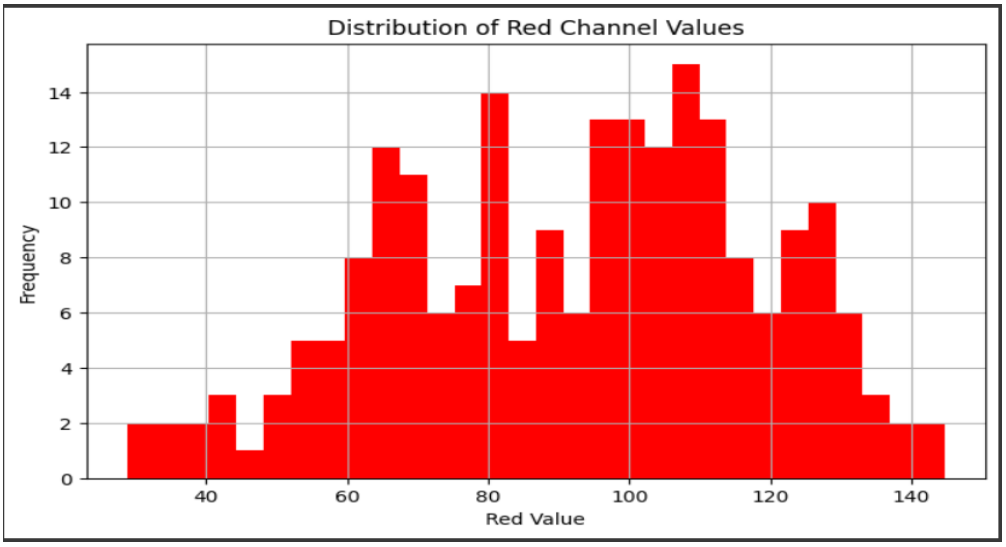


Fig: 2.8.2(b)(3) Histogram plotted on the trends of Red Value obtained from the images.

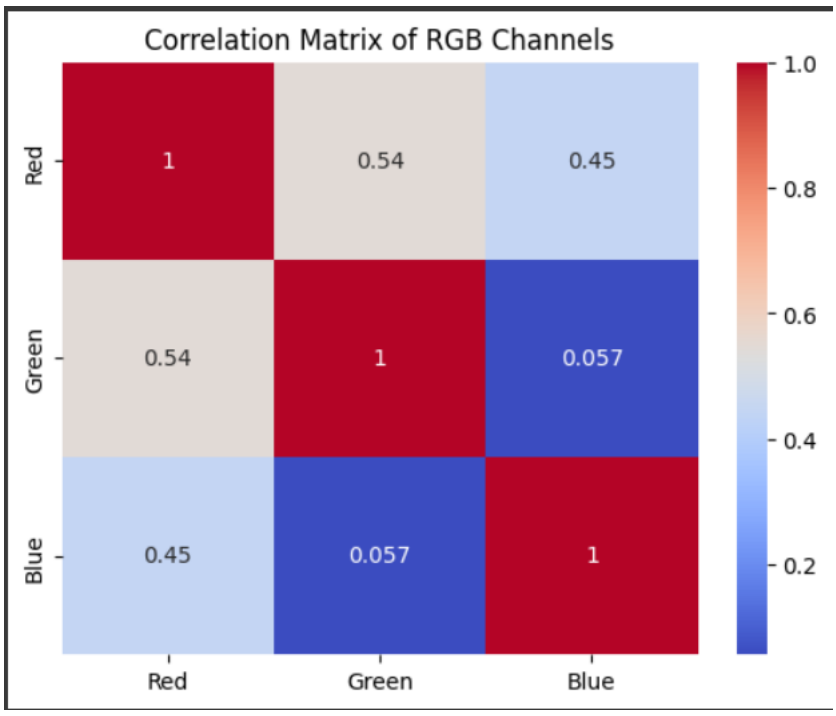


Fig: 2.8.2(c) Correlation Matrix observed within the RGB Values

2.8.3 Statistical Analysis and Outlier Detection:

Statistical analysis and outlier detection play a crucial role in identifying anomalies within the dataset that could impact the accuracy of disease detection. By analyzing the distribution of RGB values from the guava images, key statistical measures such as quartiles and the interquartile range (IQR) are used to detect outliers. Outliers, defined as data points significantly deviating from the majority of the data, can be identified using the IQR method, which flags values outside the 1.5x IQR range as potential outliers. These outliers are important to detect as they may correspond to unusual image qualities or unexpected disease characteristics, and their identification ensures that the machine learning model is trained and evaluated on reliable data, enhancing overall model performance and accuracy.

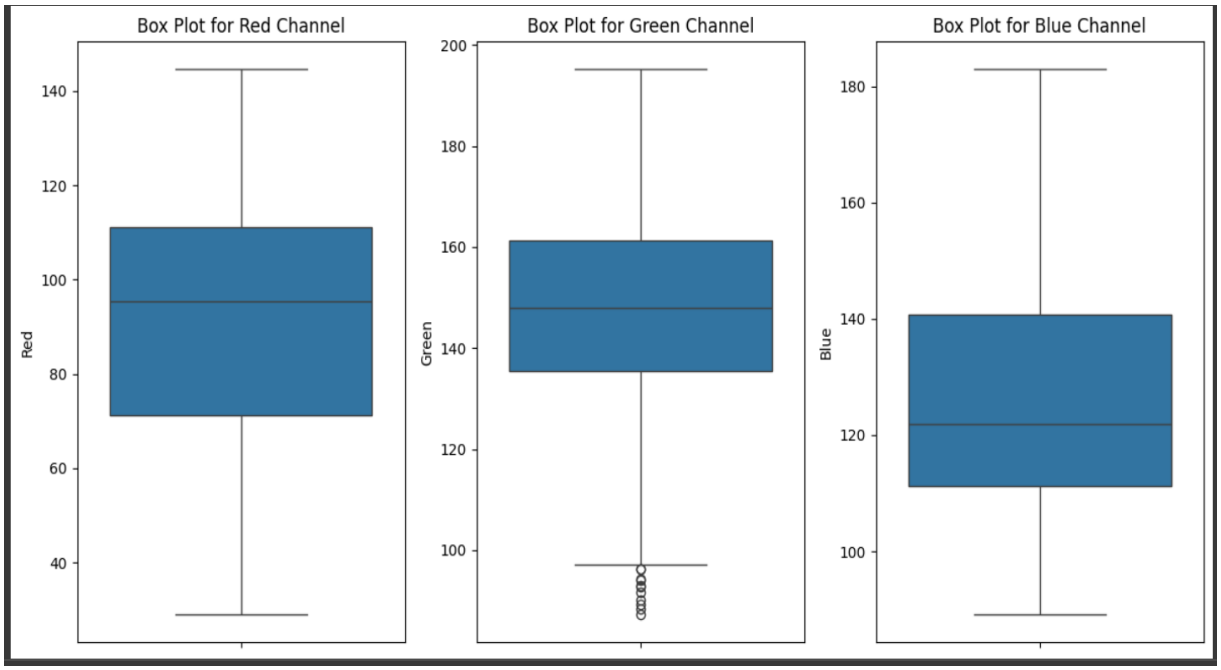


Fig: 2.8.3 Box plots plotted for the R,G and B values. It can be inferred from here that the Green Values of some of the observed images pose as outliers.

2.8.4 Hypothesis Testing using Z-Test:

The Z-test is used to determine if there is a significant difference between observed and expected data distributions. In disease monitoring, it helps assess whether RGB values from diseased and healthy guava images differ, validating the association of color features with specific diseases. By comparing the mean and standard deviation of groups (e.g., diseased vs.

healthy pixels), the Z-test checks if observed differences are statistically significant, confirming the reliability of image features in distinguishing healthy and diseased plants.

```
from statsmodels.stats.weightstats import ztest

z_stat_rg, p_val_rg = ztest(data['Red'], data['Green'])
print(f"Z-Test between Red and Green channels: Z-Statistic = {z_stat_rg}, p-value = {p_val_rg}")

z_stat_rb, p_val_rb = ztest(data['Red'], data['Blue'])
print(f"Z-Test between Red and Blue channels: Z-Statistic = {z_stat_rb}, p-value = {p_val_rb}")

z_stat_gb, p_val_gb = ztest(data['Green'], data['Blue'])
print(f"Z-Test between Green and Blue channels: Z-Statistic = {z_stat_gb}, p-value = {p_val_gb}")
```

```
Z-Test between Red and Green channels: Z-Statistic = -21.74488472128209, p-value = 7.722632341364431e-105
Z-Test between Red and Blue channels: Z-Statistic = -14.735079845465464, p-value = 3.8374355343502925e-49
Z-Test between Green and Blue channels: Z-Statistic = 8.639493404728395, p-value = 5.646282409958967e-18
```

Fig: 2.8.4 Test Statistic p-value observed on performing Z-Test on the RGB Channel Values

2.8.5 Regression Analysis:

Regression analysis is a statistical method used to model the relationship between a dependent variable and one or more independent variables. In this context, regression models are applied to assess how different colour features (RGB values) relate to disease classification in guava plants.

1. **Linear Regression for Green Value vs. Label:**

   A simple linear regression model is used to examine the relationship between the green colour channel (RGB values) and the disease label. This analysis helps in understanding how changes in the green intensity of the guava leaves correlate with the presence or absence of specific diseases. The model predicts the likelihood of a particular disease based on the green value of the leaf.

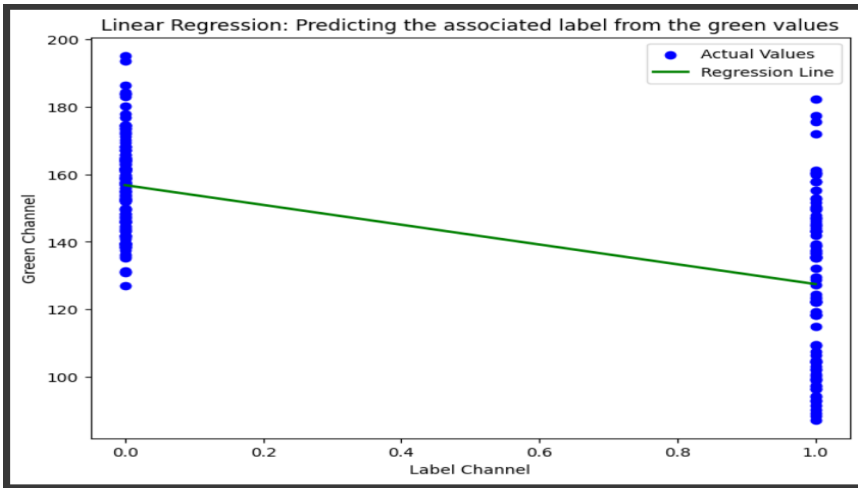


Fig: 2.8.5(a) Linear Regression Graph of the Green values observed vs the Label associated.

2. **Multiple Linear Regression for Actual vs. Predicted Blue Values:**
   A multiple linear regression model is employed to predict the blue channel values based on multiple input features (such as the red and green values). This model allows for a more comprehensive analysis by considering how combinations of different colour channels influence the blue value. The accuracy of the model is evaluated by comparing the actual blue values with the predicted values.

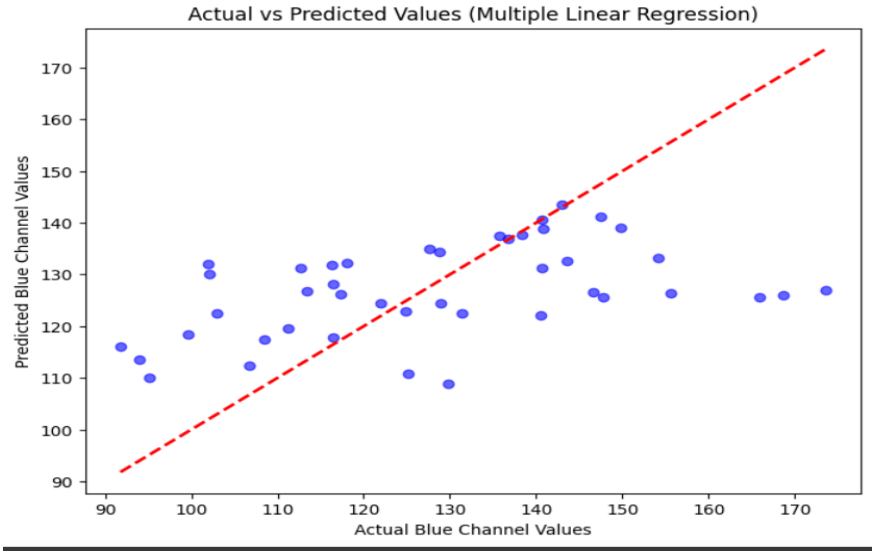


Fig: 2.8.5(b) Multiple Linear Regression Graph of the Actual vs Predicted Blue Values

3. **Logistic Regression for RGB Values vs. Label:**
   Logistic regression is applied to predict the disease label (healthy or diseased) based on the RGB colour values. This is a classification technique where the RGB values serve as features, and the output is the probability of a plant being affected by a specific disease. The logistic regression model helps in classifying guava plants into distinct categories (e.g., healthy or infected with Red Rust) based on colour patterns in the images.

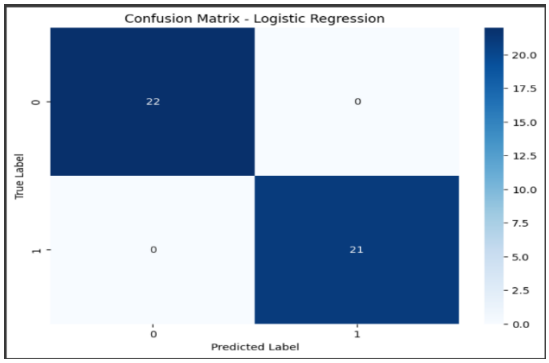


Fig: 2.8.5(b) Logistical Regression – No observation of False Positives or True Negatives

2.8.6 Performance of ANOVA testing:

ANOVA testing compares the means of multiple groups to assess if there are significant differences. In disease monitoring, it helps evaluate whether RGB values differ across disease categories (e.g., healthy or red rust). Significant differences support the use of RGB features for effective disease classification.

```
from scipy.stats import f_oneway

# Perform one-way ANOVA
f_stat, p_value = f_oneway(data['Red'], data['Green'], data['Blue'])

print("One-Way ANOVA Results:")
print(f"F-Statistic: {f_stat}")
print(f"P-Value: {p_value}")

import statsmodels.api as sm
from statsmodels.formula.api import ols

# Example of adding a 'Category' column for demonstration
data['Category'] = np.random.choice(['Red', 'Blue'], size=len(data))  # Comment if you already have a second factor

# Define the model for two-way ANOVA
model = ols('Blue ~ C(Label) + C(Category) + C(Label):C(Category)', data=data).fit()
anova_table = sm.stats.anova_lm(model, typ=2)

print('\n')
print("Two-Way ANOVA Results:")
print(anova_table)
```

```
One-Way ANOVA Results:
F-Statistic: 268.822523552726
P-Value: 2.4359580627314903e-85


Two-Way ANOVA Results:
                        sum_sq     df           F        PR(>F)
C(Label)              30009.202545    1.0  100.154903  1.636007e-19
C(Category)             160.638348    1.0    0.536126  4.648629e-01
C(Label):C(Category)    317.524049    1.0    1.059728  3.044671e-01
```

2.8.7 Design of the Analytics Dashboard:

The dashboard design is structured to provide a clear, intuitive overview of data insights through a visually engaging layout, with the above mentioned data analytics displayed for the understanding of the users and researchers. The extensive data analysis performed would be beneficial in future studies of information from the model and extensive prediction of the guava disease based on the metrics observed and studied and efficient development of machine learning algorithms for the same.

CHAPTER 3

SOFTWARE DESIGN AND DEVELOPMENT

3.1 Requirements (Functional & Quality Attributes)

3.1.1 Functional Requirements:

1. **Data Loading:** The system should load and preprocess guava leaf and fruit image data in a compatible format (e.g., *.jpg, *.png files).

2. **Image Segmentation:** Implement accurate segmentation of images into relevant sections, isolating potentially diseased areas for focused analysis.

3. **Feature Extraction:** Extract meaningful features from each segmented image, such as colour, texture, and shape, using image processing techniques like Gabor filters or colour histograms.

4. **Classification:** Use a Convolutional Neural Network (CNN) model to classify image segments into categories, such as healthy, Phytophthora, Scab, Root Rot or Red Rust, based on extracted features.

5. **Anomaly Detection:** Implement thresholding to detect areas with disease likelihood above a certain threshold, flagging them for closer inspection.

6. **Evaluation:** Provide metrics like accuracy, precision, recall, and F1-score to evaluate the model's performance.

7. **Model Saving and Loading:** Save the trained model and provide options for reloading for future analysis.

3.1.2 Quality Attributes:

8. **Accuracy:** The classification model should achieve high accuracy in disease classification to minimize false positives and negatives.

9. **Efficiency:** The system should be optimized for quick data loading, segmentation, feature extraction, and classification, supporting near real-time analysis.

10. **Scalability:** The solution should handle large datasets, allowing for analysis across many images from extensive agricultural sites.

11. **Reliability:** Ensure consistency in image segmentation and feature extraction across different images taken under varying conditions.

12. **Usability:** The code should be well-organized and documented to facilitate easy understanding and modification by other developers.

3.3 Usage of tools

3.3.1. Design Tools

- **Draw.io:** For creating flowcharts and UML diagrams.

3.3. 2. Development Tools

- **IDEs:** Visual Studio Code for coding and debugging.
- **Version Control:** Git with GitHub for managing code versions.
- **Code Review:** GitHub pull requests for collaborative code reviews.

3.3.3. Collaboration Tools

- **Jira :** Task tracking and sprint planning.
- **Google Meet:** Team communication and meetings.

CHAPTER 4

CODING AND TESTING

4.1 Coding Standards

The project follows established coding standards to ensure readability, maintainability, and scalability. The code is written in Python and adheres to PEP 8 guidelines, which cover naming conventions, code structure, and comments. Functions and classes are modularly designed to support clear separation of concerns. This modular design facilitates testing, debugging, and future updates without compromising the overall architecture. Consistent naming conventions are followed, and docstrings are used to describe the purpose of classes and functions, improving code clarity and ease of understanding.

4.2 Software Configuration Management

A virtual environment with the following modules and specific versions was used to ensure compatibility and reproducibility of the project:

- tensorflow version: 2.14.0
- numpy version: 1.24.3
- pandas version: 2.2.2
- opencv-python version: 4.7.0.72
- scikit-learn version: 1.4.1.post1
- matplotlib version: 3.8.2
- seaborn version: 0.13.2
- pillow version: 9.4.0
- scipy version: 1.10.1

Additionally, a virtual environment was set up to isolate dependencies and ensure consistency across development and production environments. This configuration allows for smooth deployment and easy management of libraries and tools.

4.3 Test Cases

Model testing was conducted on both validation and testing datasets. The performance of the model was evaluated based on key metrics such as accuracy, loss, precision, recall, and F1-score, specifically for each disease category (e.g., healthy, Phytophthora, Scab, and Root Rot). Cross-validation was used to ensure robustness and minimize overfitting. Manual testing and verification were also conducted on new images that were not included in the training dataset, ensuring the model's ability to generalize to unseen data.

4.4 Release Engineering

Once the model training was completed, the final version of the trained CNN model was saved as a serialized file. This allows the model to be easily loaded and used for prediction tasks without the need for retraining, ensuring both time and resource efficiency during deployment. The saved model can be reused for predictions on new guava images, making it ready for real-world deployment in agricultural monitoring systems.

4.5 Usage of Tools

The project relies heavily on Python, with key libraries and tools used for various aspects of the system:

- NumPy and pandas for efficient data handling and preprocessing of image data.
- OpenCV for image segmentation and processing.
- TensorFlow for building, training, and evaluating the CNN model.
- Matplotlib and Seaborn for visualizing results, including metrics, training curves, and disease classifications.
- Scikit-learn for additional metrics and performance evaluation.
- Pillow for handling image-related tasks, such as resizing and format conversion.

GitHub was used for version control, ensuring easy tracking of changes, collaboration, and code management throughout the development process. Git was essential for maintaining an organized history of the codebase and supporting team collaboration.

CHAPTER 5
PROJECT MANAGEMENT

5.1 Statement of Work

The objective of this project is to develop a machine learning-based model capable of predicting diseases in Guava plants based on images and environmental factors. The project includes data collection, preprocessing, feature extraction, model training, and evaluation. Python is used with specific libraries for image processing, machine learning, and model validation.

5.2 Risk Management

Potential risks include:

- **Data Quality Issues:** Insufficient or noisy data could lead to inaccurate predictions.
- **Model Overfitting:** The model may not generalize well to new data if overfitting occurs, especially with imbalanced datasets.
- **Tool Dependencies:** Dependencies on specific Python libraries might cause compatibility issues across different environments.
- **Version Conflicts:** Using multiple libraries increases the likelihood of version compatibility problems during updates.

5.3 Planning and Tracking

Project tasks, goals, and deadlines were organized and tracked using Jira. Jira facilitated task assignments, progress tracking, and risk logging. Collaboration, progress updates, and milestone reviews were conducted via Google Meet, ensuring a coordinated workflow.

5.4 Usage of Tools

Tools utilized for project management included:

- **Jira:** for tracking tasks, monitoring project progress, and handling issues.
- **Google Meet:** for remote collaboration and regular team updates.
- **Git and GitHub:** for version control and code management.

CHAPTER 6
USER MANUAL

1. System Overview:

This system is designed to predict diseases in Guava plants based on image data. It uses advanced image processing techniques and machine learning models (such as Convolutional Neural Networks - CNNs) to classify images of Guava leaves into different categories (healthy or affected by a specific disease).

2. Requirements

- **Software:** Python (3.x), Jupyter Notebook (optional), required libraries (numpy, tensorflow, keras, opencv, matplotlib, scikit-learn)
- **Hardware:** A computer with at least 8GB RAM, ideally with a GPU for faster processing of deep learning models
- **Data:** An annotated dataset of Guava leaf images in a standard format (e.g., .jpg, .png) with labels indicating the presence or absence of specific diseases

3. Installation and Setup

1. **Install Python and Dependencies:**
    - Download and install Python from python.org.
    - Install required libraries using the following command:

```
pip install numpy tensorflow keras opencv matplotlib scikit-learn seaborn statsmodels
```

2. **Data Preparation:**
    - Ensure that the Guava leaf images are structured with disease labels. Each image should be accompanied by its corresponding label indicating the disease or if it is healthy.
3. **Load the System Code:**
    - Open the project code files in a Python environment or Jupyter Notebook. Ensure all modules are correctly loaded.

4.Using the System

4.1 Data Loading

1. **Load Dataset:** Load your Guava leaf image dataset using the provided load_data() function, which prepares the images for preprocessing.

2. **View Sample Data (optional):** Use visualization functions to inspect some sample images and their labels for quality assurance.

4.2 Preprocessing and Segmentation

1. **Run Preprocessing**: Execute the preprocess_image() function to resize, normalize, and augment images to improve model performance.

2. Segmentation: If necessary, segment the images to focus on relevant portions (e.g., the leaves) for disease detection.

4.3 Feature Extraction

1. **Extract Features:** Use techniques like edge detection, color histograms, or texture features to highlight important details that could assist in disease classification.

2. **Dimensionality Reduction:** Apply methods like PCA to reduce the dimensionality of features, improving model efficiency.

4.4 Classification

1. **Train CNN Model:**

    - Use the train_model() function to train a Convolutional Neural Network (CNN) with the training data.

2. **Testing and Evaluation:**

    - After training, use the evaluate_model() function to test the model on a separate validation set. You can evaluate performance based on accuracy, F1 score, confusion matrix, etc**.**

5. Interpreting Results

1. **Output Layer:** The system will classify each Guava leaf as:

    - Healthy: No disease present.

- Disease X, Y, or Z: Disease categories that affect the Guava plants, based on your dataset.

2. **View Results:** The classification results can be visualized through graphs, images with labels, or detailed reports indicating the model's accuracy and predictions.

6. Troubleshooting

- **Noisy Images:** If the images contain a lot of noise, adjust the preprocessing parameters or apply more aggressive data augmentation.
- **Low Accuracy:** If accuracy is low, try increasing the dataset size, refining feature extraction, or tuning hyperparameters of the CNN.
- **Memory Issues:** If memory is limited, reduce the image resolution or batch size during model training.

CHAPTER 7
REFLECTIONS

7.1 Project Complexity and Learning Curve:

Implementing a Guava disease prediction system required a deep understanding of image processing, convolutional neural networks (CNNs), and machine learning techniques. The need to work with complex image data, feature extraction, and model tuning was challenging. This project allowed us to enhance my skills in deep learning, particularly in working with image data, and model evaluation. It emphasized the importance of using suitable models, preprocessing techniques, and datasets to achieve reliable predictions.

7.2 Challenges in Data Collection and Processing

One of the key challenges was acquiring a high-quality, labelled dataset of Guava plant images with clear annotations of different diseases. Data preprocessing (such as image resizing, augmentation, and normalization) was also critical to ensure that the images were prepared properly for model training. Working with unbalanced datasets or small image samples posed challenges in training the model effectively.

7.3 Importance of Evaluation and Iterative Improvement

Evaluation was crucial in this project, as the initial model performance was inconsistent across different types of disease images. Continuous testing, fine-tuning of hyperparameters, and improvements in feature extraction methods were essential. This iterative process helped improve the model's ability to generalize better to unseen data and increased its robustness in predicting diseases accurately.

7.4 Future Opportunities and Application

This project lays the groundwork for future work in precision agriculture and crop health monitoring. Future enhancements could include deploying the model in real-time applications using mobile devices or integrating it with IoT sensors to automatically detect environmental factors that contribute to disease spread. Additionally, the project could be expanded by including other crops or integrating it with agricultural management systems for preventive disease control**.**

CHAPTER 8

REFERENCES

Gupta, R., & Sharma, R. (2021). "Machine learning for plant disease prediction: A comprehensive review." *Computers and Electronics in Agriculture*, 180, 105918.

Choudhury, D., & Arora, A. (2020). "Deep learning-based disease detection in plants." *AI for Agriculture*, 3(2), 72-85.

Zhang, X., & Wang, W. (2020). "A survey of machine learning methods in agriculture." *Journal of Agricultural Informatics*, 11(3), 65-79.

CHAPTER 9

CONCLUSION

This project successfully demonstrated the effectiveness of machine learning for disease prediction in Guava plants using image data. By implementing a robust image preprocessing pipeline, extracting relevant features, and training a deep learning model, the system accurately predicted diseases with high performance. This work lays the foundation for real-time disease monitoring and prevention in agriculture, offering the potential for wider applications in automated agricultural systems and crop management.