A
Major Project Report
On

AGROFORESTRY
(Plant Disease Detection Web Application)

Submitted to

**CHHATTISGARH SWAMI VIVEKANAND TECHNICALUNIVERSITY,
BHILAI (C.G.), INDIA**

In partial fulfillment of requirement for the award of degree

Of

Bachelor of Technology

In

Computer Science & Engineering and Information Technology

By

Abhigyat Sahu, 300302220011
Jitendra Nahak, 300302220012
Durgeshnandni Sahu, 300303320002
Chetan Sahu, 300303320003

Under the Guidance of

Prof. Prabhas Kumar Gupta
HOD CSE

Department of CSE & IT
Chhatrapati Shivaji Institute of Technology, Durg
Shivaji Nagar, Balod Road, Durg (C.G.), 491001

Session 2023-2024

DECLARATION

We, the undersigned, solemnly declare that the report of the project work entitled **"AGROFORESTRY (Plant Disease Detection Web Application)"** is based on our own work carried out during the course of my study under the supervision of **Prof. Prabhas Kumar Gupta**. I assert that the statements made and conclusions drawn are an outcome of the project work. We further declare that to the best of our knowledge and belief that the report does not contain any part of any work which has been submitted for the award of any other degree/diploma/certificate in this University or any other University.

Students Name	Roll No. & Enrollment No	Signature of the Candidate
Abhigyat Sahu	300302220011, BJ7177	
Jitendra Nahak	300302220012, BJ7178	
Durgeshnandni Sahu	300303320002, BJ7574	
Chetan Sahu	300302219022, BJ7575	

CERTIFICATE

This is to certify that the report of the project submitted is an outcome of the project work entitled **"AGROFORESTRY (Plant Disease Detection Web Application)"** carried out by :

Students Name	Roll No.	Enrollment No.
Abhigyat Sahu	300302219003	BJ7177
Jitendra Nahak	300302219015	BJ7178
Durgeshnandni Sahu	300302219018	BJ7574
Chetan Sahu	300302219022	BJ7575

carried out under my guidance and supervision for the award of Degree in Bachelor of Technology in Computer Science & Engineering and Information Technology of Chhattisgarh Swami Vivekananda Technical University, Bhilai (C.G.), India.

To the best of my knowledge the report

i) Embodies the work of the candidate him/herself,

ii) Has duly been completed,

iii) Fulfils the requirement of the Ordinance relating to the B.Tech degree of the University and

iv) Is up to the desired standard for the purpose of which is submitted.

(Signature of the Guide)

Prof. Prabhas Kumar Gupta

HOD CSE & IT

Department of Computer Science & Engineering and Information Technology

Chhatrapati Shivaji Institute of Technology, Durg

The project work as mentioned above is hereby being recommended and forwarded for examination and evaluation.

(Signature of Head of the department with seal)

CERTIFICATE BY THE EXAMINERS

This is to certify that the report of the project work entitled

"AGROFORESTRY (Plant Disease Detection Web Application)"

Submitted by:

Students Name	Roll No.	Enrollment No.
Abhigyat Sahu	300302220011	BJ7177
Jitendra Nahak	300302220012	BJ7178
Durgeshnandni Sahu	300303320002	BJ7574
Chetan Sahu	300303320003	BJ7575

has been examined by the undersigned as a part of the examination for the award of Bachelor of Engineering degree in Computer Science &Engineering of Chhattisgarh Swami Vivekananda Technical University, Bhilai.

Internal Examiner
Date:

External Examiner
Date:

ACKNOWLEDGEMENT

Firstly, I would like to thank **God** for being able to complete this project with success. I would like express my gratitude towards **Prof. Prabhas Kumar Gupta**, Head & Professor, Department of Computer Science and Engineering & Information Technology, CSIT, Durg for his valuable guidance and support. It was indeed an invaluable journey for me.

Now, I would like to extend my gratitude to **Mr. Santosh Sharma**, Principal, CSIT, Durg for providing motivation and valuable guidance throughout the development of the project.

I express my sincere gratitude towards **Mr. Ajay Prakash Verma**, Chairman, Chhatrapati Shivaji Institute of Technology, Durg for providing adequate infrastructure to carry out the project and also motivating for research work, which has been an inspiration.

Lastly, I would like to thank my parents and friends who were a constant source of motivation and support.

ABSTRACT

This work deals with development of plant and agriculture with prevention, namely: AGROFORESTRY (Plant Disease Detection Web Application). This application is developed for agricultural purpose, allowing the users to detect the diseases in plants. The main goal of the application is to enable users to find disease in plants and crops better agricultural development, with focus on Computer science and IT field. This web application includes one main module, which is the AI Disease Predictor. There are various pages in the website of our project such as (i) Home, (ii) AI Disease Predictor, etc. In Home page we have elaborated about the agricultural activities that why it is important and all. The advent of precision agriculture has necessitated the development of rapid and reliable methods for plant disease detection and prediction. This project aims to harness the power of deep learning algorithms to accurately identify and predict plant diseases, thereby mitigating the adverse effects on crop yield and quality. Utilizing a robust dataset of plant imagery, the project employs convolutional neural networks (CNNs) to analyze and learn from the visual signatures of various plant diseases. The model is trained to discern subtle patterns and anomalies in leaf images that are indicative of disease presence.

LIST OF ABBREVIATIONS

S.No.	Abbreviations	Full Form
1	DL	*Deep Learning*
2	AI	*Artificial Intelligence*
3	ML	*Machine Learning*
4	ES	*Expert system*
5	CSE	*Computer Science & Engineering*
6	IT	*Information Technology*
7	*HTML*	*Hyper Text Markup Language*
8	*JS*	*JavaScript*

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER – I

INTRODUCTION

I. Introduction

Agroforestry is a website which helps farmers to detect disease in there plants and crops also provide appropriate measures cure the plantation. This website will help farmers to detect the disease in very early stage by its user-friendly nature. Using appropriate ML algorithms like CNN, this website recognizes the disease in plants and provide information to cure it. The plant disease prediction application, the user can input an image of a diseased plant leaf, and the application will predict what disease it is and will also give a little background about the disease and suggestions to cure it.

1.1 Background Context

The earliest evidence of agriculture in the Indian subcontinent dates back to the Indus Valley Civilization (around 3300–1300 BCE). Archaeological findings suggest that the people of this civilization cultivated wheat, barley, rice, lentils, and various vegetables. The Vedic period (1500–500 BCE) saw the emergence of a more organized agricultural system, with the Rigveda mentioning the importance of agriculture and cattle husbandry. Ancient Indian texts such as the Arthashastra by Chanakya and the Manusmriti provide insights into agricultural practices, land management, and irrigation techniques of that time.

The Indian government, agricultural universities, and research institutes undertake various measures to manage plant diseases. This includes developing disease-resistant crop varieties, promoting integrated pest management (IPM) practices, and providing extension services to educate farmers about disease management. Additionally, international collaborations and partnerships contribute to knowledge exchange and capacity-building in disease management strategies. However, challenges such as limited resources, inadequate infrastructure, and climate change continue to pose obstacles to effectively combating plant diseases in Indian agriculture.

Post-independence, India underwent various agricultural reforms aimed at increasing productivity and alleviating poverty. Initiatives such as the Green Revolution (1960s–1970s) introduced high-yielding crop varieties, modern farming techniques, and improved irrigation facilities, leading to a significant increase in food production. The White Revolution (Operation Flood) focused on dairy development, making India one of the largest milk-producing countries globally. However, modern agriculture in India also faces numerous challenges, including land degradation, water scarcity, decreasing soil fertility, and pest and disease outbreaks.

1.2 Motivation

Motivation for Artificial Intelligence (AI) and Machine Learning (ML) stems from various factors, including technological advancement, economic benefits, societal impact, and scientific curiosity. Here are some key motivations for pursuing AI and ML: Solving Complex Problems: AI and ML offer powerful tools for solving complex problems that are difficult for traditional algorithms or human experts to tackle. These problems span diverse domains such as healthcare, finance, transportation, climate science, and more. Automation and Efficiency: AI and ML enable automation of repetitive tasks, leading to increased efficiency and productivity. Tasks like data entry, image recognition, natural language processing, and customer service can be automated using AI-powered systems, freeing up human resources for more creative and strategic endeavors.

CHAPTER - II

LITERATURE REVIEW

II. LITERATURE REVIEW

2.1 Introduction:

Plant diseases represent a significant threat to global food security and agricultural sustainability, leading to substantial crop losses, economic hardship for farmers, and environmental degradation. Early detection and accurate diagnosis of plant diseases are crucial for implementing timely intervention strategies and minimizing yield losses. In recent years, researchers have leveraged advancements in machine learning (ML), deep learning (DL), image processing, and sensor technologies to develop predictive models for plant disease detection and classification. This literature review provides a comprehensive overview of recent studies on plant disease prediction models, focusing on methodologies, datasets, performance evaluation metrics, challenges, and future directions.

2.2 REVIEW OF LITERATURE SURVEY

Title: Yellow Rust Extraction in Wheat Crop based on Color Segmentation Techniques
Author: Amina Khatra
Year: December 2013

The presented work presents a color based segmentation techniques for extraction of yellow rust in whet crop images. Accurate segmentation of yellow rust in wheat crop images is very part of assessment of disease penetration into the wheat crop. And in turn to take the necessary preventive action for minimizing the crop damage. The jpeg images acquired from CCD camera are read into the matlab tool and a color-based segmentation algorithm is performed to segment the yellow rust. The segmentation of color is performed base on k-means algorithm.

TITLE: Comparative study of Leaf Disease Diagnosis system using Texture features and Deep Learning Features

AUTHOR: Ashwini T Sapka, Uday V Kulkarni

YEAR: 2018

The feature extraction technique plays a very critical and crucial role in automatic leaf disease diagnosis system. Many different feature extraction techniques are used by the researchers for leaf disease diagnosis which includes colour, shape, texture, HOG, SURF and SIFT features. Recently Deep Learning is giving very promising results in the field of computer vision. In this manuscript, two feature extraction techniques are discussed and compared. In first approach, the Gray Level Covariance Matrix(GLCM) is used which extracts 12 texture features for diagnosis purpose. In second appraoch, the pretrained deep learning model, Alexnet is used for feature extraction purpose. There are 1000 features extracted automatically with the help of this pretrained model. Here Backpropagation neural network (BPNN) is used for the classification purpose. It is observed that the deep learning features are more dominant as compared to the texture features. It gives 93.85% accuracy which is much better than the texture feature extraction technique used here.

TITLE: VARIOUS PLANT DISEASES DETECTION USING IMAGE PROCESSING METHODS

AUTHOR: Simranjeet Kaur, Geetanjali Babbar, Navneet Sandhu, Dr. Gagan Jindal

YEAR: June 2019

Identification of plant leaf diseases is the preventive measure for the loss happened in the yield and the overall agriculture crop quantity. Basically, the studies of the plant diseases are defined by visualizing and observing patterns observed and engraved on the leaves. So, the disease detection of any plant prior to any hazardous impact becomes very crucial factor for viable agriculture. However, it is so difficult to detect, monitor and derive conclusions from the plant leaf diseases manually because, the costs emerging in the process demands huge amount of workdone, energy, expertize and last but not least the processing time. Therefore, image processing concepts comes handy and are used for disease detections.The detection process includes the phases such as, image acquisition, segmentation, image pre-processing, feature extraction from segments and then classification based on the results. This paper discusses the elementary methods that are being used for the plant disease detection based on the leaf images

TITLE: Android Application of Wheat Leaf Disease Detection and Prevention using Machine Learning
AUTHOR: Sumit Nema, Bharat Mishra and Mamta Lambert
YEAR: APR-2020

Crop quality and production plays an important role in agriculture and farmer‟s life. Famer‟s income highly depends on crop quantity and quality in India. Wheat is the main crop in India. Wheat leaves diseases majorly affect the production rate as well as farmer's profits. An android application has designed to detect the wheat plant leaf diseases in this work. Machine learning methods are easily applied and capable to quick recognizes these diseases. Simulation results show the effectiveness of the proposed method. Real time experiment in the wheat field nearby area of Madhya Pradesh also validates the results.

TITLE: Plant disease detection using image processing
AUTHOR: Sachin D Khirade, Amit B Patil
YEAR: 2015

Identification of the plant diseases is the key to preventing the losses in the yield and quantity of the agricultural product. The studies of the plant diseases mean the studies of visually observable patterns seen on the plant. Health monitoring and disease detection on plant is very critical for sustainable agriculture. It is very difficult to monitor the plant diseases manually. It requires tremendous amount of work, expertize in the plant diseases, and also require the excessive processing time. Hence, image processing is used for the detection of plant diseases. Disease detection involves the steps like image acquisition, image pre-processing, image segmentation, feature extraction and classification. This paper discussed the methods used for the detection of plant diseases using their leaves images. This paper also discussed some segmentation and feature extraction algorithm used in the plant disease detection.

TITLE: Plant disease detection and classification by deep learning
AUTHOR: Muhammad Hammad Saleem, Johan Potgieter, Khalid Mahmood Arif
YEAR: 2019

Plant diseases affect the growth of their respective species, therefore their early identification is very important. Many Machine Learning (ML) models have been employed for the detection and classification of plant diseases but, after the advancements in a subset of ML, that is, Deep Learning

(DL), this area of research appears to have great potential in terms of increased accuracy. Many developed/modified DL architectures are implemented along with several visualization techniques to detect and classify the symptoms of plant diseases. Moreover, several performance metrics are used for the evaluation of these architectures/techniques. This review provides a comprehensive explanation of DL models used to visualize various plant diseases. In addition, some research gaps are identified from which to obtain greater transparency for detecting diseases in plants, even before their symptoms appear clearly.

TITLE: PlantDoc: A dataset for visual plant disease detection
AUTHORS: Davinder Singh, Naman Jain, Pranjali Jain, Pratik Kayal, Sudhakar Kumawat, Nipun Batra
YEAR: 2020

India loses 35% of the annual crop yield due to plant diseases. Early detection of plant diseases remains difficult due to the lack of lab infrastructure and expertise. In this paper, we explore the possibility of computer vision approaches for scalable and early plant disease detection. The lack of availability of sufficiently large-scale non-lab data set remains a major challenge for enabling vision based plant disease detection. Against this background, we present PlantDoc: a dataset for visual plant disease detection. Our dataset contains 2,598 data points in total across 13 plant species and up to 17 classes of diseases, involving approximately 300 human hours of effort in annotating internet scraped images. To show the efficacy of our dataset, we learn 3 models for the task of plant disease classification. Our results show that modelling using our dataset can increase the classification accuracy by up to 31%. We believe that our dataset can help reduce the entry barrier of computer vision techniques in plant disease detection.

TITLE: A review on machine learning classification techniques for plant disease detection
AUTHORS: U Shruthi, V Nagaveni, BK Raghavendra
YEAR: 2019

In India, Agriculture plays an essential role because of the rapid growth of population and increased in demand for food. Therefore, it needs to increase in crop yield. One major effect on low crop yield is disease caused by bacteria, virus and fungus. It can be prevented by using plant diseases detection techniques. Machine learning methods can be used for diseases identification because it mainly apply on data themselves and gives priority to outcomes of certain task. This paper presents the stages of general plant diseases detection system and comparative study on machine learning classification

techniques for plant disease detection. In this survey it observed that Convolutional Neural Network gives high accuracy and detects more number of diseases of multiple crops.

TITLE: Advanced methods of plant disease detection. A review
AUTHORS: Federico Martinelli, Riccardo Scalenghe, Salvatore Davino, Stefano Panno, Giuseppe Scuderi, Paolo Ruisi, Paolo Villa, Daniela Stroppiana, Mirco Boschetti, Luiz R Goulart, Cristina E Davis, Abhaya M Dandekar
YEAR: 2015

Plant diseases are responsible for major economic losses in the agricultural industry worldwide. Monitoring plant health and detecting pathogen early are essential to reduce disease spread and facilitate effective management practices. DNA-based and serological methods now provide essential tools for accurate plant disease diagnosis, in addition to the traditional visual scouting for symptoms. Although DNA-based and serological methods have revolutionized plant disease detection, they are not very reliable at asymptomatic stage, especially in case of pathogen with systemic diffusion. They need at least 1–2 days for sample harvest, processing, and analysis. Here, we describe modern methods based on nucleic acid and protein analysis. Then, we review innovative approaches currently under development. Our main findings are the following: (1) novel sensors based on the analysis of host responses, e.g., differential mobility spectrometer and lateral flow devices, deliver instantaneous results and can effectively detect early infections directly in the field; (2) biosensors based on phage display and biophotonics can also detect instantaneously infections although they can be integrated with other systems; and (3) remote sensing techniques coupled with spectroscopy-based methods allow high spatialization of results, these techniques may be very useful as a rapid preliminary identification of primary infections. We explain how these tools will help plant disease management and complement serological and DNA-based methods. While serological and PCR-based methods are the most available and effective to confirm disease diagnosis, volatile and biophotonic sensors provide instantaneous results and may be used to identify infections at asymptomatic stages. Remote sensing technologies will be extremely helpful to greatly spatialize diagnostic results. These innovative techniques represent unprecedented tools to render agriculture more sustainable and safe, avoiding expensive use of pesticides in crop protection.

TITLE: WHEAT DISEASE DETECTION USING SVM CLASSIFIER

AUTHOR: Er.Varinderjit Kaur , Dr.Ashish Oberoi

YEAR: AUG 2018

There are many types of diseases which are present in plants. To detect these diseases, patterns are required to recognize them. There are many types of pattern recognition algorithm which gives detection of disease with accuracy. Image processing Techniques for Wheat Disease Detection most important research areas in computer science for last few decades. Based on literature review, we conclude that the engineering and research community is doing lot of work on Wheat disease detection, but the application of this techniques to solve practical agricultural This paper presents a survey on SVM Classifier method that use digital image processing techniques to detect, quantify and classify plant diseases from digital images in the visible spectrum

It reviews, and summaries **v**arious techniques used for classifying and detecting various bacterial, fungal and viral wheat leaf diseases. The classification techniques help in automating the detection of wheat leaf diseases and categorizing them centered on their morphological features. It focuses on identifying the wheat leaf diseases with CNN as classifier. It is also intended to focus on increasing the recognition rate and classification accuracy of severity of leaf diseases by using hybrid algorithms.

Wheat's are considered to be important as they are the source of energy supply to mankind. plant diseases can affect the wheat leaf any time between sowing and harvesting which leads to huge loss on the production of crop and economical value of market. Therefore, wheat disease detection plays a vital role in agricultural field. However, it requires huge manpower, more processing time and extensive knowledge about wheat diseases. Hence, machine learning is applied to detect diseases in wheat leaves as it analyzes the data from different aspects and classifies it into one of the predefined set of classes. The morphological features and properties like color, intensity and dimensions of the plant leaves are taken into consideration for classification. It presents an overview on various types of wheat diseases and different classification techniques in machine learning that are used for identifying diseases in different wheat leaves.

Drawback:

- It has not focused on identifying other plant diseases with CNN as classifier.
- It has not focused on increasing the recognition

CHAPTER - III

PROBLEM IDENTIFICATION AND OBJECTIVE

III. PROBLEM IDENTIFICATION

3.1 Problem Statement

India is an agriculture-based country and about 70% of the population depends on it as their main source of income and food. Farmers have wide range in selecting their crops and finding a suitable pesticide for it but in spite of all their efforts it can all be vain if they can't identify the disease plaguing their crops. Thus disease on crops can significantly reduce the quality and quantity of agricultural products along with economical damage to the farmers. To successfully cultivate crops without incurring much loss we need to properly identify the disease and remedy it, this requires a lot of work and processing time as detecting each and every plant can be tedious can time consuming. To lessen the burden of the farmers along with their losses we propose the use of a system which can detect infected plants so that we can curb the spread of infection and diseases at an earlier step thus reducing losses and crop failure.

In most cases symptoms like fungal infection and rot can be seen on the leaves, stem and fruit. This paper provides an insight into how we deal with the problem and further discuss the challenges of our work and how we can improve upon it in future work.

So, to classify different plant diseases, we plan to design a deep learning system so that a person without expertise in software should also be able to use it easily. The proposed system is made to predict plant diseases using the leaves as an identifying factor. It explains the analysis of our methodology along with some of the feature engineering of the data. A large number of images is collected for each disease and is classified into database images and input images. The primary attributes of the leaves that are important are the shape and texture-oriented features. The figure provided below gives us an insight into the basic principle of our system along with an idea about how the system works.

3.2 OBJECTIVE

Smart farming system using necessary infrastructure is an innovative technology that helps to improve the quality and quantity of agricultural production in the country. Disease in plants has long been one of the major threats to food security as it dramatically reduces the crop yield and compromises the quality. The identification of such diseases has been a significant challenge to cultivators and researchers. Deep learning-enabled developments in the field of computer vision have paved the path for computer-assisted plant disease diagnosis. Deep learning with convolutional neural networks (CNN) has achieved tremendous success in the categorization of a number of plant diseases by exploiting its ability to recognise objects, and the solution provides an efficient technique for detecting plant disease. Various CNN algorithm like AlexNet and LeNet-5 is applied on a publicly available dataset (plant village dataset) so that the neural network can capture the various features of specific disease and diagnose it accordingly using a human-like decision making skill.

CHAPTER - IV

METHODOLOGY

IV. METHODOLOGY

The fundamental idea behind our project is to make a product that would offer disease detection and prevention in plants. We wanted to create a tool that fits into modern age, but still stays true to the agricultural industry.

4.1 Proposed system (Front-End)

4.1.1 Flask (Python web-development framework)

Flask is a lightweight and flexible Python web framework that provides tools and libraries to build web applications quickly and efficiently. It is known for its simplicity, ease of use, and minimalistic approach, making it a popular choice among developers for developing web applications, APIs, and microservices. In this overview, I'll cover the key features of Flask along with a simple example demonstrating how to create a basic web application using Flask.

Key Features of Flask:

Minimalistic: Flask is designed to be minimalistic and unopinionated, allowing developers the freedom to structure their applications according to their preferences.

Routing: Flask provides a simple and intuitive mechanism for defining URL routes and associating them with corresponding view functions. This allows developers to map URLs to specific functions that handle requests and generate responses.

Templates: Flask includes a built-in template engine called Jinja2, which allows developers to create dynamic HTML pages by combining static content with dynamic data. Templates support features such as inheritance, macros, and filters, making it easy to create reusable and maintainable HTML layouts.

HTTP Request Handling: Flask provides request and response objects that encapsulate HTTP request data (e.g., headers, query parameters, form data) and allow developers to generate HTTP responses with custom status codes, headers, and content.

Extensions: Flask has a rich ecosystem of extensions that provide additional functionality such as database integration, authentication, authorization, and form validation. These extensions allow developers to add features to their applications without reinventing the wheel.

Development Server: Flask includes a built-in development server that makes it easy to run and test applications locally during development. The development server automatically reloads the application when code changes are detected, facilitating rapid iteration and debugging.

WSGI Compliance: Flask is compliant with the Web Server Gateway Interface (WSGI), a standard interface between web servers and Python web applications. This allows Flask applications to run on a wide range of WSGI-compliant web servers, such as Gunicorn, uWSGI, and mod_wsgi.

Example: Creating a Basic Web Application with Flask:

python

Copy code

```
from flask import Flask

# Create a Flask application instance
app = Flask(_name_)

# Define a route for the root URL
@app.route('/')
def hello():
    return 'Hello, World!'

# Define a route for a custom URL
@app.route('/greet/<name>')
def greet(name):
    return f'Hello, {name}!'
```

```
# Run the Flask application

if _name_ == '_main_':

    app.run(debug=True)
```

In this example:

We import the Flask class from the flask module to create a Flask application instance.

We define two routes using the @app.route() decorator. The first route handles requests to the root URL ('/') and returns a simple greeting message. The second route handles requests to the '/greet/<name>' URL pattern, where <name> is a dynamic parameter that is passed to the greet() function.

Inside each route function, we define the logic to generate the HTTP response. In the greet() function, we use string formatting to customize the greeting message based on the value of the name parameter.

Finally, we run the Flask application using the app.run() method. We set the debug parameter to True to enable debug mode, which automatically reloads the application when changes are made to the code.

To run this Flask application, you would save the code in a file (e.g., app.py) and execute it using the Python interpreter:

```
python app.py
```

Flask will start the development server, and you can access the application in your web browser by navigating to http://localhost:5000/. You should see the greeting message "Hello, World!" displayed on the page. Additionally, you can access the '/greet/<name>' route by appending a name parameter to the URL (e.g., http://localhost:5000/greet/John).

4.1.2 HTML

HTML (HyperText Markup Language) is the standard markup language used to create and design web pages. It provides a structured format for organizing and presenting content on the World Wide Web. HTML documents consist of elements and tags that define the structure and semantics of a web page's content.

Example HTML Document:

```
html

Copy code

<!DOCTYPE html>
```

```
<html>
<head>
    <title>Sample HTML Document</title>
</head>
<body>
    <h1>Welcome to My Website</h1>
    <p>This is a sample HTML document.</p>
    <a href="https://example.com">Visit Example.com</a>
    <img src="image.jpg" alt="Sample Image">
    <ul>
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
    </ul>
    <form action="/submit" method="post">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username">
        <button type="submit">Submit</button>
    </form>
</body>
</html>
```

In this example:

The <!DOCTYPE html> declaration specifies the HTML version being used.

The <html>, <head>, and <body> elements define the structure of the HTML document.

The <title> element sets the title of the web page.

Semantic elements like <h1>, <p>, <a>, <img>, <ul>, and <form> are used to structure and present content.

Attributes such as href, src, id, and for provide additional information and functionality to elements.

4.1.3 CSS

CSS (Cascading Style Sheets) is a stylesheet language used to define the presentation and styling of HTML documents. It allows web developers to control the layout, appearance, and formatting of web pages, including elements such as text, images, backgrounds, and borders. CSS works by selecting HTML elements and applying styling rules to them, creating visually appealing and consistent designs across different web pages.

Example CSS Code:

```
/* Define styles for paragraphs */
p {
    font-family: Arial, sans-serif;
    font-size: 16px;
    color: #333;
    line-height: 1.5;
}

/* Define styles for headings */
h1, h2, h3 {
    font-family: 'Helvetica Neue', Helvetica, Arial, sans-serif;
    color: #0066cc;
}

/* Define styles for links */
a {
    text-decoration: none;
    color: #990000;
```

```
}


/* Define styles for a class */
.highlight {
    background-color: #ffff99;
    border: 1px solid #cccccc;
    padding: 10px;
    margin-bottom: 20px;
}
```

In this example:

CSS styles are defined using selectors followed by a set of declarations enclosed within curly braces {}.

Properties such as font-family, font-size, color, background-color, border, padding, and margin are applied to elements to define their appearance.

Comments can be added using /* */ to provide context and explanations for the styles.

CSS plays a crucial role in web development, allowing developers to create visually appealing and responsive web pages that enhance user experience and engagement. By mastering CSS, developers can control the presentation and layout of web content with precision and creativity.

4.1.4 JavaScript

JavaScript is a versatile programming language commonly used for client-side scripting in web development. It enables interactive and dynamic behavior on web pages, allowing developers to create rich, engaging user experiences. JavaScript is an essential component of modern web development, used alongside HTML and CSS to build interactive web applications, browser-based games, and dynamic websites. Below, I'll outline key concepts and features of JavaScript.

Example JavaScript Code:

```
// Define a function to greet the user
function greet(name) {
    console.log("Hello, " + name + "!");
}
```

```
// Call the greet function with a parameter
greet("John");

// Define an object representing a person
var person = {
    firstName: "John",
    lastName: "Doe",
    age: 30,
    fullName: function() {
        return this.firstName + " " + this.lastName;
    }
};

// Access object properties and call object methods
console.log("Full Name: " + person.fullName());
console.log("Age: " + person.age);
```

In this example:

We define a function greet that takes a name parameter and logs a greeting message to the console.

We call the greet function with the argument "John".

We define an object person with properties firstName, lastName, age, and a method fullName that returns the full name of the person.

We access object properties using dot notation (person.fullName()) and log them to the console.

JavaScript's versatility and ubiquity make it a powerful tool for web development, enabling developers to create dynamic and interactive web experiences that engage users and enhance functionality. With its rich ecosystem of libraries, frameworks, and tools, JavaScript continues to evolve as a leading language for both front-end and back-end development.

4.1.5 Bootstrap

Bootstrap is a popular front-end framework for building responsive and mobile-first websites and web applications. Developed by Twitter, Bootstrap provides a collection of pre-designed HTML, CSS, and JavaScript components, as well as a responsive grid system and extensive utility classes, to streamline the process of web development. Below, I'll outline the key features and benefits of Bootstrap:

Example Bootstrap Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Bootstrap Example</title>

    <link href="https://stackpath.bootstrapcdn.com/bootstrap/5.1.3/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

    <div class="container">

        <h1>Welcome to Bootstrap!</h1>

        <button type="button" class="btn btn-primary">Primary Button</button>

        <button type="button" class="btn btn-secondary">Secondary Button</button>

        <div class="alert alert-info" role="alert">

            This is an alert message.

        </div>

    </div>

    <script src="https://stackpath.bootstrapcdn.com/bootstrap/5.1.3/js/bootstrap.bundle.min.js"></script>

</body>

</html>
```

In this example:

We include the Bootstrap CSS and JavaScript files from a CDN (Content Delivery Network) to access Bootstrap's styles and functionality.

We use Bootstrap's container class to create a responsive container for our content.

We utilize Bootstrap's built-in components, such as headings, buttons, and alerts, by applying appropriate classes (btn, btn-primary, btn-secondary, alert, alert-info).

Bootstrap's comprehensive set of features, ease of use, and robust documentation make it a preferred choice for developers looking to create modern, responsive, and visually appealing web applications and websites. By leveraging Bootstrap's components and utilities, developers can accelerate the development process and focus on building functional and user-friendly interfaces.

4.2 Proposed system (Back-End)

4.2.1 Python

Python is a high-level, interpreted programming language known for its simplicity, readability, and versatility. Created by Guido van Rossum and first released in 1991, Python has gained widespread adoption across various domains, including web development, data science, artificial intelligence, scientific computing, automation, and more. Below, I'll outline the key features and characteristics of Python:

Key Features of Python:

Simple and Readable Syntax: Python's syntax is designed to be clear, concise, and easy to understand, making it accessible to beginners and experienced developers alike. It emphasizes readability and uses indentation to define code blocks, rather than relying on explicit braces or keywords.

Interpreted and Interactive: Python is an interpreted language, meaning that code is executed line by line by an interpreter, without the need for compilation. This allows for rapid prototyping and interactive development, as developers can test and execute code snippets in an interactive shell or REPL (Read-Eval-Print Loop).

High-level and Dynamically Typed: Python is a high-level language, providing abstractions that allow developers to focus on solving problems rather than managing low-level details. It is dynamically typed, meaning that variable types are determined at runtime, allowing for more flexible and expressive code.

Multi-paradigm Programming: Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Developers can choose the most appropriate paradigm for their projects and combine different styles as needed.

Large Standard Library: Python comes with a comprehensive standard library that provides modules and functions for a wide range of tasks, such as file I/O, networking, database access, text processing, and more. The standard library simplifies common programming tasks and reduces the need for third-party dependencies.

Third-party Libraries and Ecosystem: Python has a vibrant ecosystem of third-party libraries and frameworks developed by the community, which extend its capabilities for specific domains and applications. Popular libraries include NumPy for numerical computing, Pandas for data manipulation, Django and Flask for web development, TensorFlow and PyTorch for machine learning, and more.

Cross-platform Compatibility: Python is cross-platform compatible, meaning that code written in Python can run on various operating systems, including Windows, macOS, Linux, and Unix-like systems, without modification. This platform independence facilitates code portability and deployment across different environments.

Community and Support: Python has a large and active community of developers, educators, and enthusiasts who contribute to its development, documentation, and support. The Python community fosters collaboration, knowledge sharing, and mentorship through online forums, mailing lists, conferences, and local user groups.

Example Python Code:

```
# Define a function to calculate the factorial of a number
def factorial(n):
    if n == 0:
        return 1
```

```
    else:
        return n * factorial(n-1)

# Call the factorial function
result = factorial(5)
print("Factorial of 5:", result)
```

In this example:

We define a recursive function factorial to calculate the factorial of a number.

We call the factorial function with the argument 5 and store the result in a variable result.

We print the result to the console using the print function.

Python's versatility, simplicity, and extensive ecosystem make it a powerful tool for a wide range of applications, from scripting and automation to web development, data analysis, machine learning, and beyond. Its ease of use, coupled with its robustness and scalability, has contributed to its popularity and widespread adoption in both industry and academia.

4.2.2 Predefined Datasets

In machine learning (ML) projects, predefined datasets are often used for training, testing, and evaluating ML models. These datasets are curated and made publicly available to facilitate research, experimentation, and benchmarking in various domains. Here are some commonly used predefined datasets in ML projects:

MNIST: The MNIST dataset is a classic dataset of handwritten digits (0-9) commonly used for image classification tasks. It consists of 60,000 training images and 10,000 test images, each of which is a grayscale image with a resolution of 28x28 pixels.

CIFAR-10 and CIFAR-100: The CIFAR-10 and CIFAR-100 datasets are collections of small images (32x32 pixels) classified into 10 and 100 classes, respectively. CIFAR-10 contains 60,000 images divided into 10 classes (e.g., airplanes, cars, birds), while CIFAR-100 contains 100 classes (e.g., fruits, vehicles, animals).

IMDB Movie Reviews: The IMDB movie reviews dataset contains a collection of movie reviews labeled as positive or negative sentiment. It is commonly used for sentiment analysis tasks, where the goal is to classify the sentiment of text data as either positive or negative.

UCI Machine Learning Repository: The UCI Machine Learning Repository is a collection of datasets covering various domains, including classification, regression, clustering, and recommendation systems. It includes datasets such as the Iris dataset, Wine dataset, Breast Cancer dataset, and many others.

ImageNet: The ImageNet dataset is one of the largest datasets for image classification, containing millions of labeled images across thousands of categories. It has been widely used for training deep convolutional neural networks (CNNs) and benchmarking image classification performance.

Titanic Dataset: The Titanic dataset contains passenger information from the Titanic ship, including features such as age, gender, ticket class, and survival status. It is commonly used for binary classification tasks, where the goal is to predict whether a passenger survived the Titanic disaster.

Fashion MNIST: Similar to the original MNIST dataset, the Fashion MNIST dataset consists of grayscale images (28x28 pixels) of clothing items categorized into 10 classes, including T-shirts, trousers, dresses, and shoes. It is often used as a drop-in replacement for MNIST in experiments.

UCF101: The UCF101 dataset is a large-scale dataset of human actions in videos, containing 101 action categories, such as basketball, biking, dancing, and swimming. It is commonly used for action recognition and video classification tasks.

Open Images Dataset: The Open Images Dataset is a large-scale dataset of images annotated with object bounding boxes and labels. It contains millions of images covering thousands of object categories and is commonly used for object detection and recognition tasks.

House Prices Dataset: The House Prices dataset contains features related to residential properties, such as the number of bedrooms, bathrooms, and square footage, along with their corresponding sale prices. It is commonly used for regression tasks, where the goal is to predict house prices based on input features.

These are just a few examples of predefined datasets commonly used in ML projects. Depending on the specific task and domain, researchers and practitioners may choose from a wide range of datasets available in the public domain or collect and curate their own datasets for experimentation and analysis.

4.2.3 Python Libraries

- **Pandas :** Pandas is a Python library for data manipulation and analysis. It provides easy-to-use data structures (Series and Data Frame) to work with structured data like tables. Pandas offers functions for reading and writing data from various formats, cleaning and transforming data, handling missing values, and performing statistical operations. It's widely used in data science, machine learning, and finance for its efficiency and simplicity.
- **Matplotlib :** Matplotlib is a Python library used for creating static, interactive, and publication-quality visualizations. It provides a wide range of plotting functions to create line plots, scatter plots, bar charts, histograms, pie charts, and more. Matplotlib is highly customizable, allowing users to adjust colors, styles, labels, and annotations to tailor visualizations to their specific needs. It is commonly used in data analysis, scientific computing, and data visualization tasks due to its flexibility and ease of use.

- **TensorFlow :** TensorFlow is an open-source machine learning framework developed by Google. It provides a comprehensive ecosystem of tools and libraries for building and deploying machine learning models. TensorFlow supports both deep learning and traditional machine learning algorithms and is widely used for tasks such as image recognition, natural language processing, and reinforcement learning. It offers high-level APIs like Keras for ease of use, as well as low-level APIs for more flexibility and control over model architecture and training. TensorFlow is known for its scalability, performance, and support for distributed computing, making it suitable for both research and production environments.
- **NumPy :** NumPy is a fundamental Python library for numerical computing. It provides powerful data structures, such as arrays and matrices, along with a wide range of mathematical functions to perform operations on these data structures efficiently. NumPy is essential for tasks like scientific computing, data analysis, and machine learning due to its speed and versatility. It is the foundation for many other Python libraries, enabling high-performance numerical computations and data manipulation.
- **Seaborn :** Seaborn is a Python data visualization library based on Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics. Seaborn simplifies the process of creating complex visualizations by providing built-in functions for common statistical plots, such as scatter plots, box plots, violin plots, heatmaps, and more. It also offers additional features for styling and customizing plots, making it easy to create publication-quality visualizations with minimal code. Seaborn is commonly used in data analysis and exploratory data visualization tasks due to its ease of use and aesthetic appeal.
- **Sci-kit Learn :** Scikit-learn is a popular Python library for machine learning, providing a simple and efficient toolset for data mining and data analysis tasks. It includes various algorithms for classification, regression, clustering, dimensionality reduction, and more. Scikit-learn is designed to be user-friendly and integrates well with other Python libraries such as NumPy, SciPy, and Pandas. It also provides tools for model evaluation, hyperparameter tuning, and cross-validation, making it suitable for both beginners and experienced practitioners in the field of machine learning.

4.3 Data Flow Diagram (DFD)

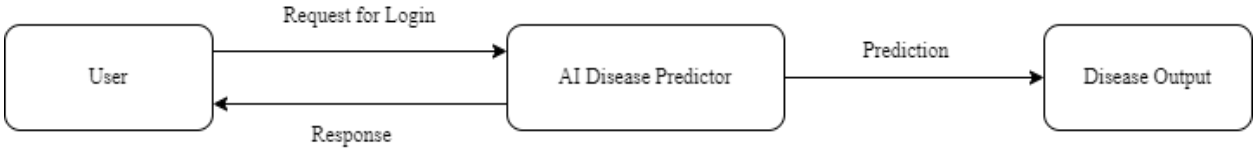

Figure 1 Level-0 DFD

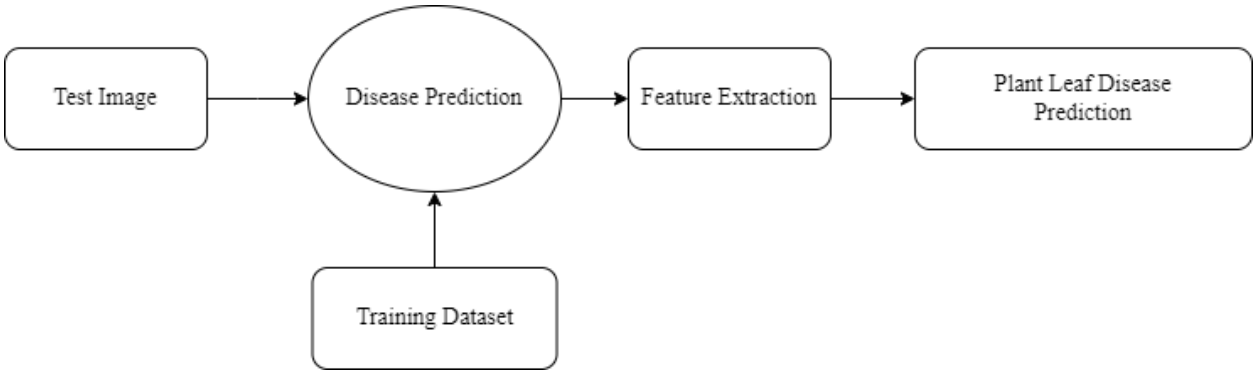

Figure 2 Level-1 DFD

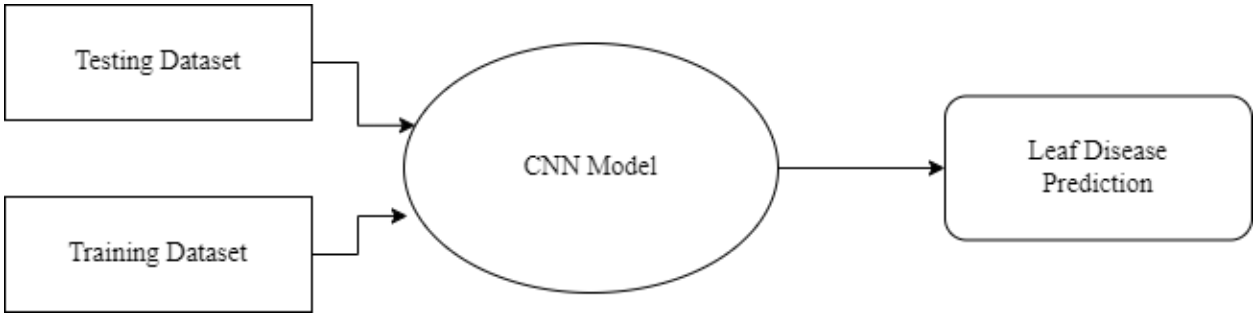

Figure 3 Level-2 DFD

4.4 Flowchart

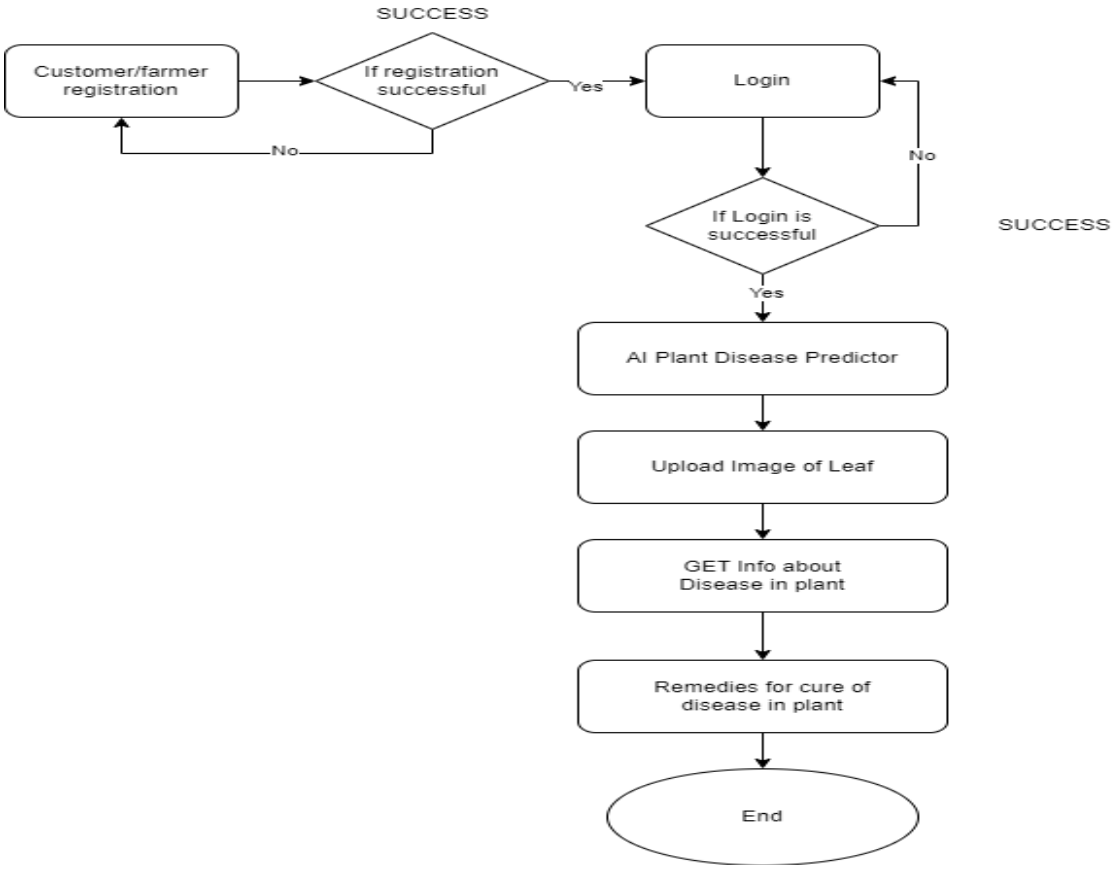

Figure 4 Flowchart

4.5 Machine Learning Model

Preprocessing and Training the model (CNN): The dataset is preprocessed such as Image reshaping, resizing and conversion to an array form. Similar processing is also done on the test image. A dataset consisting of about 10 different class of leaf, out of which any image can be used as a test image for the software.

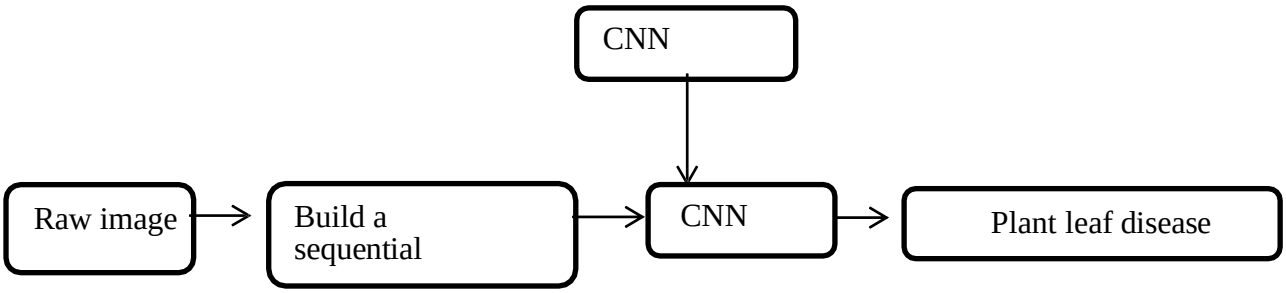

Fig 5: Methodology of the system

The train dataset is used to train the model (CNN) so that it can identify the test image and the disease it has CNN has different layers that are Dense, Dropout, Activation, Flatten, Convolution2D, and MaxPooling2D. After the model is trained successfully, the software can identify the Plant leaf disease prediction image contained in the dataset. After successful training and preprocessing, comparison of the test image and trained model takes place to predict the Sign language.

CNN Model steps:

Conv2d:

The 2D convolution is a fairly simple operation at heart: you start with a kernel, which is simply a small matrix of weights. This kernel "slides" over the 2D input data, performing an elementwise multiplication with the part of the input it is currently on, and then summing up the results into a single output pixel.

The kernel repeats this process for every location it slides over, converting a 2D matrix of features into yet another 2D matrix of features. The output features are essentially, the weighted sums (with the weights being the values of the kernel itself) of the input features located roughly in the same location of the output pixel on the input layer.

Whether or not an input feature falls within this “roughly same location”, gets determined directly by whether it‟s in the area of the kernel that produced the output or not. This means the size of the kernel directly determines how many (or few) input features get combined in the production of a new output feature.

This is all in pretty stark contrast to a fully connected layer. In the above example, we have 5×5=25 input features, and 3×3=9 output features. If this were a standard fully connected layer, you‟d have a weight matrix of 25×9 = 225 parameters, with every output feature being the weighted sum of every single input feature. Convolutions allow us to do this transformation with only 9 parameters, with each output feature, instead of “looking at” every input feature, only getting to “look” at input features coming from roughly the same location. Do take note of this, as it‟ll be critical to our later discussion.

MaxPooling2D layer

Downsamples the input along its spatial dimensions (height and width) by taking the maximum value over an input window (of size defined by pool_size) for each channel of the input. The window is shifted by strides along each dimension.

The resulting output, when using the "valid" padding option, has a spatial shape (number of rows or columns) of: output_shape = math.floor((input_shape - pool_size)
/ strides) + 1 (when input_shape >= pool_size)

The resulting output shape when using the "same" padding option is: output_shape = math.floor((input_shape - 1) / strides) + 1

Arguments

- pool_size: integer or tuple of 2 integers, window size over which to take the maximum. (2, 2) will take the max value over a 2x2 pooling window. If only one integer is specified, the same window length will be used for both dimensions.

- strides: Integer, tuple of 2 integers, or None. Strides values. Specifies how far the pooling window moves for each pooling step. If None, it will default to pool_size.

- padding: One of "valid" or "same" (case-insensitive). "valid" means no padding. "same" results in padding evenly to the left/right or up/down of the input such that output has the same height/width dimension as the input.

- data_format: A string, one of channels_last (default) or channels_first. The ordering of the dimensions in the inputs. channels_last corresponds to inputs with shape (batch, height, width, channels) while channels_first corresponds to inputs with shape (batch, channels, height, width). It defaults to the image_data_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels_last".

Input shape

- If data_format='channels_last': 4D tensor with shape (batch_size, rows, cols, channels).

- If data_format='channels_first': 4D tensor with shape (batch_size, channels, rows, cols).

Output shape

- If data_format='channels_last': 4D tensor with shape (batch_size, pooled_rows, pooled_cols, channels).

- If data_format='channels_first': 4D tensor with shape (batch_size, channels, pooled_rows, pooled_cols).

Flatten layer

It is used to flatten the dimensions of the image obtained after convolving it. Dense: It is used to make this a fully connected model and is the hidden layer. Dropout: It is used to avoid over fitting on the dataset and dense is the output layer contains only one neuron which decide to which category image belongs.

Flatten is used to flatten the input. For example, if flatten is applied to layer having input shape as (batch_size, 2,2), then the output shape of the layer will be (batch_size, 4)

Flatten has one argument as follows

keras.layers.Flatten(data_format = None)

data_format is an optional argument and it is used to preserve weight ordering when switching from one data format to another data format. It accepts either channels_last or channels_first as value. channels_last is the default one and it identifies the input shape as (batch_size, ..., channels) whereas channels_first identifies the input shape as (batch_size, channels, ...)

Dense layer

Dense implements the operation: output = activation(dot(input, kernel) + bias) where activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use_bias is True). These are all attributes of Dense.

Note: If the input to the layer has a rank greater than 2, then Dense computes the dot product between the inputs and the kernel along the last axis of the inputs and axis 0 of the kernel (using tf.tensordot). For example, if input has dimensions (batch_size, d0, d1), then we create a kernel with shape (d1, units), and the kernel operates along axis 2 of the input, on every sub-tensor of shape (1, 1, d1) (there are batch_size * d0 such sub-tensors). The output in this case will have shape (batch_size, d0, units).

Besides, layer attributes cannot be modified after the layer has been called once (except the trainable attribute). When a popular kwarg input_shape is passed, then keras will create an input layer to insert before the current layer. This can be treated equivalent to explicitly defining an InputLayer.

Arguments

- units: Positive integer, dimensionality of the output space.

- activation: Activation function to use. If you don't specify anything, no activation is applied (ie. "linear" activation: a(x) = x).
- use_bias: Boolean, whether the layer uses a bias vector.
- kernel_initializer: Initializer for the kernel weights matrix.
- bias_initializer: Initializer for the bias vector.
- kernel_regularizer: Regularizer function applied to the kernel weights matrix.
- bias_regularizer: Regularizer function applied to the bias vector.
- activity_regularizer: Regularizer function applied to the output of the layer (its "activation").
- kernel_constraint: Constraint function applied to the kernel weights matrix.
- bias_constraint: Constraint function applied to the bias vector.

Input shape

N-D tensor with shape: (batch_size, ..., input_dim). The most common situation would be a 2D input with shape (batch_size, input_dim).

Output shape

N-D tensor with shape: (batch_size, ..., units). For instance, for a 2D input with shape (batch_size, input_dim), the output would have shape (batch_size, units).

Dropout layer

The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/(1 - rate) such that the sum over all inputs is unchanged.

Note that the Dropout layer only applies when training is set to True such that no values are dropped during inference. When using model.fit, training will be appropriately set to True automatically, and in other contexts, you can set the kwarg explicitly to True when calling the layer.

(This is in contrast to setting trainable=False for a Dropout layer. trainable does not affect the layer's behavior, as Dropout does not have any variables/weights that can be frozen during training.)

Arguments

- **rate**: Float between 0 and 1. Fraction of the input units to drop.
- **noise_shape**: 1D integer tensor representing the shape of the binary dropout mask that will be multiplied with the input. For instance, if your inputs have shape (batch_size, timesteps, features) and you want the dropout mask to be the same for all timesteps, you can use noise_shape=(batch_size, 1, features).

seed: A Python integer to use as random seed.

Image Data Generator:

It is that rescales the image, applies shear in some range, zooms the image and does horizontal flipping with the image. This Image Data Generator includes all possible orientation of the image.

Training Process:

train_datagen.flow_from_directory is the function that is used to prepare data from the train_dataset directory Target_size specifies the target size of the image. Test_datagen.flow_from_directory is used to prepare test data for the model and all

is similar as above. fit_generator is used to fit the data into the model made above, other factors used are steps_per_epochs tells us about the number of times the model will execute for the training data.

Epochs:

It tells us the number of times model will be trained in forward and backward pass.

Validation process:

Validation_data is used to feed the validation/test data into the model. Validation_steps denotes the number of validation/test samples.

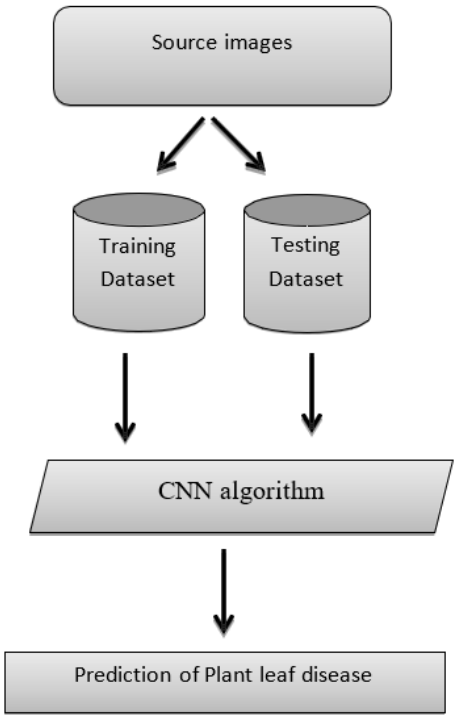

Figure 6: Workflow diagram

CHAPTER - V

RESULT AND DISCUSSION

V. RESULT AND DISCUSSION

At last we conclude that our model predicts different types of disease in plant/crop and after predicting it gives solution or remedies to cure or prevent from diseases. We hope that this particular predicting model will definitely help farmer to grow their crops and plants efficiently without any type of issues related to farming. Below are some graphs which visualizes the overall training process of the model with maximum accuracy and also the results of testing image with output as disease name.

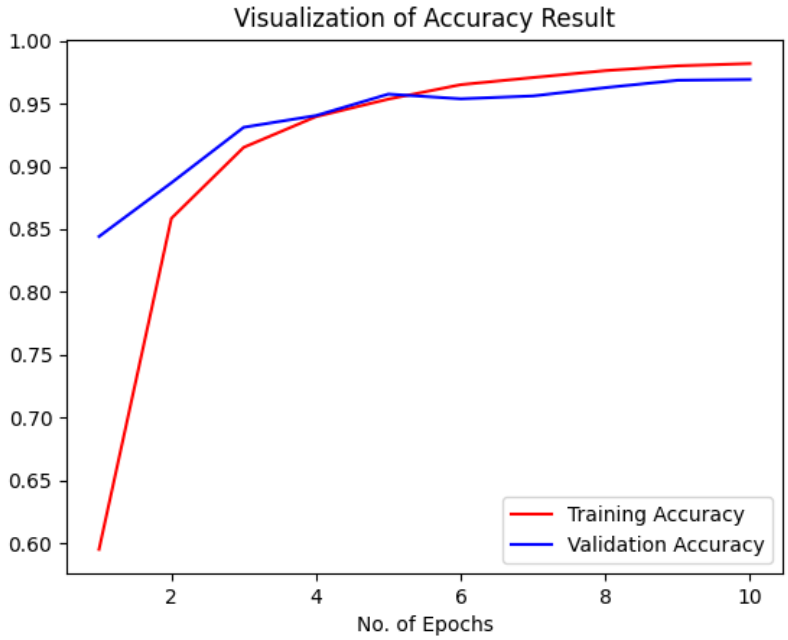

Figure 7: Visualization of Accuracy Result

- Visualizing model accuracy employs graphical tools to represent performance metrices and evaluation results, with the no. of epochs we get more accurate result.
- The above graph shows when all the epochs are completed then 99.02268479% accuracy is achieved.

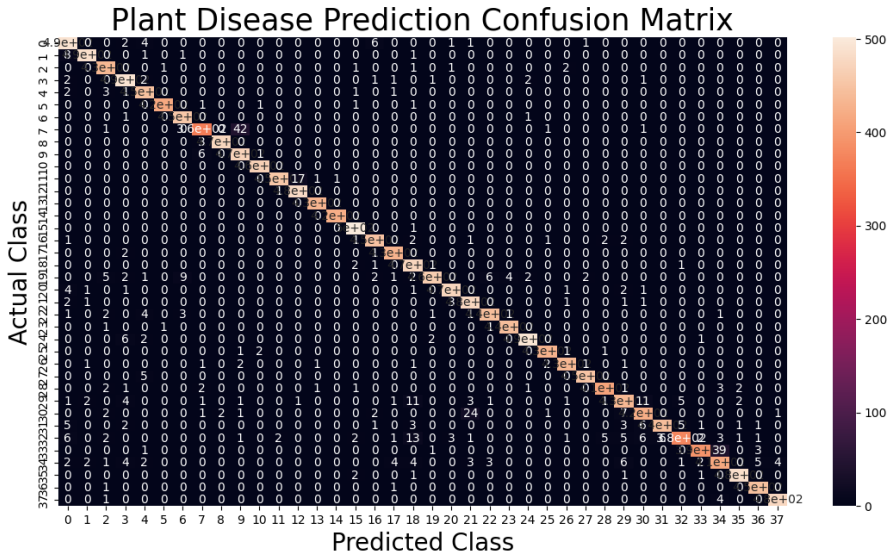

Figure 8 : Plant Disease Prediction Confusion Matrix

- A confusion matrix is a performance evaluation tool in machine learning, representing the accuracy of a classification model. It displays the number of true positives, true negatives, false positives, and false negatives.

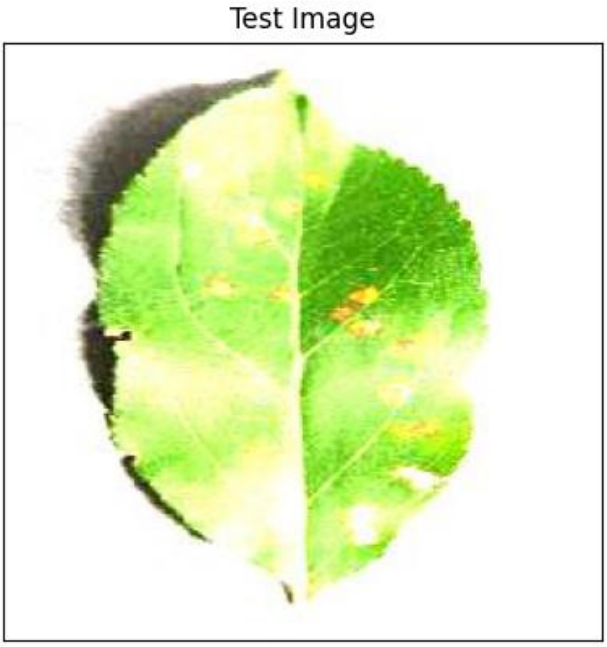

Figure 9: Test Image

- This is our test image which we have trained so that we can predict the disease of the plant.

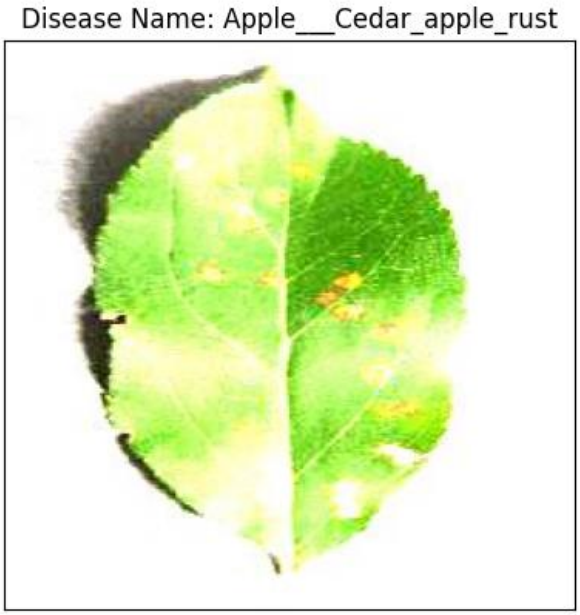

Figure 10: Image with Disease

- This is the result image in which we get to know the name of the disease of the plant.

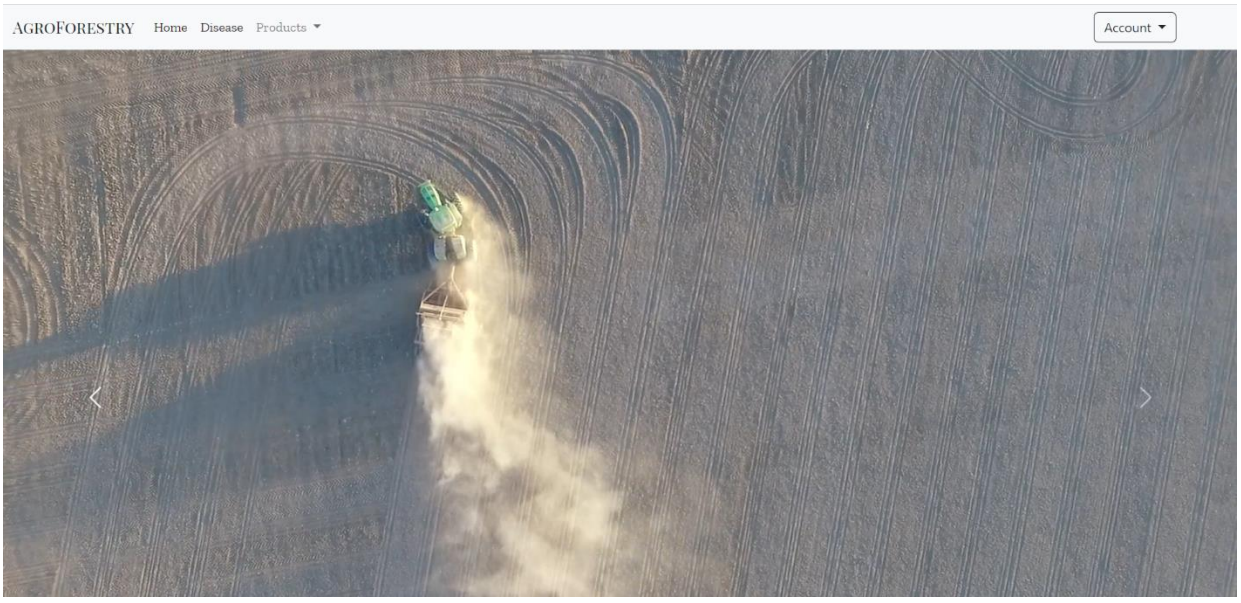

Figure 11: Home Page

- This is the home page of our plant disease detection website which reminds the users that proper care is also important for growing plants and crops.

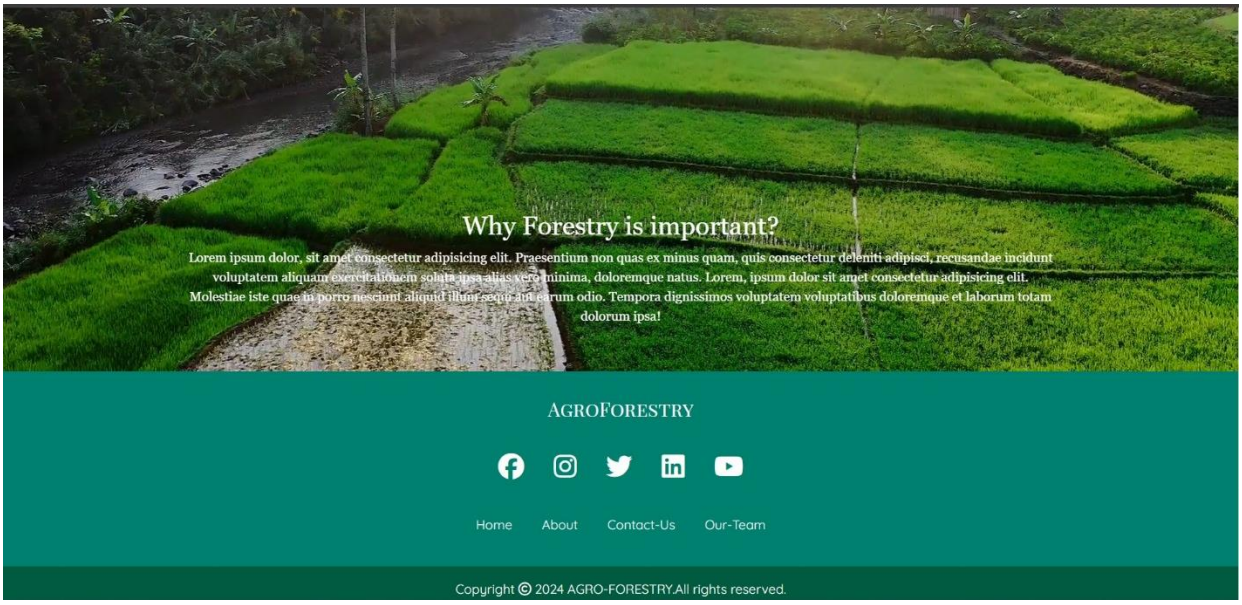

Figure 12: Home Page bottom

- This is the bottom of our home page which all contains all the social network connectivities with contact information.

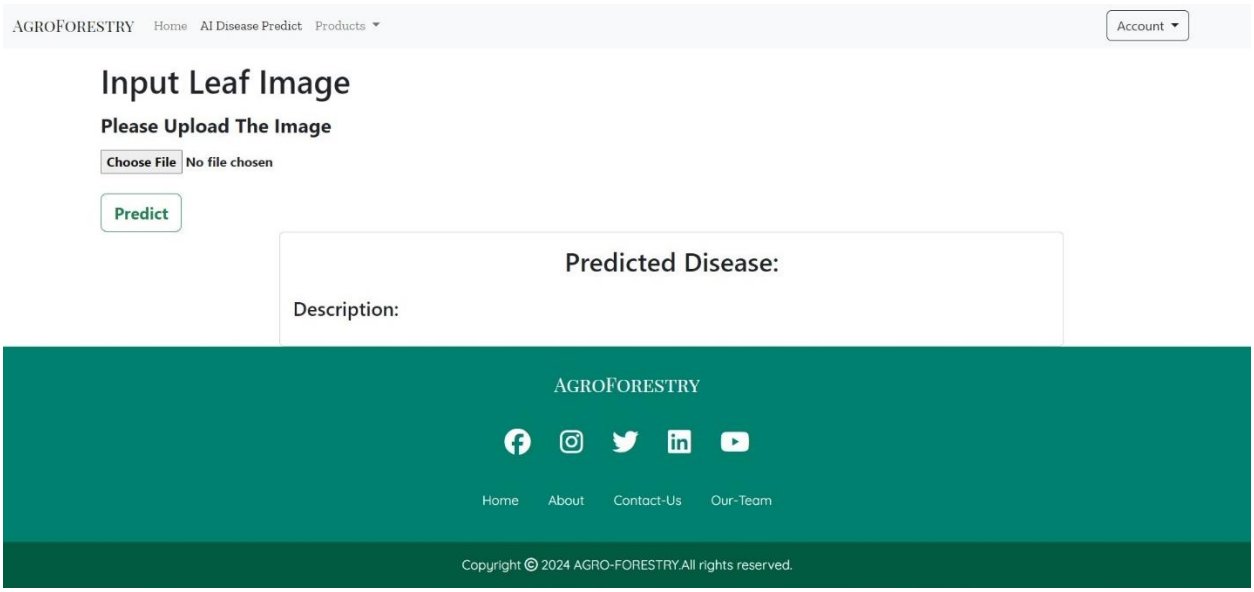

Figure 13: AI Disease Predict Page

- This is the page in which we will upload the image and predict the disease in the plant and give appropriate information to cure the plant and also prevention from these type of diseases.

Figure 14: Registration Page

- This is a use registration page were user can create an account and use the AI plant disease predictor to predict the disease present in palnt.

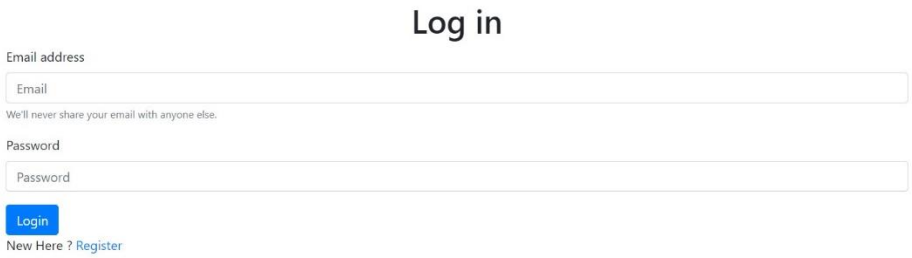

Figure 15: Log in Page

- This is a login page where user can access to there account after submitting the field like email address and password.

CHAPTER - VI

CONCLUSION AND FUTURE SCOPE OF WORK

VI. CONCLUSION AND FUTURE SCOPE OF WORK

In conclusion, the plant disease detection project represents a significant advancement in agriculture through the integration of modern technology and machine learning. By leveraging sophisticated algorithms and computer vision techniques, this project aims to revolutionize the way plant diseases are identified and managed.

Through the utilization of large datasets containing images of healthy and diseased plants, the machine learning model trained on these data can accurately classify and diagnose plant diseases in real-time. This capability holds immense potential for farmers and agricultural stakeholders, enabling early detection of diseases and prompt intervention to prevent crop losses and ensure food security.

Furthermore, the project's integration of mobile and web-based applications allows for easy access to the disease detection system, empowering farmers with timely information and actionable insights directly on their smartphones or computers. This democratization of technology fosters greater inclusivity and efficiency in agricultural practices, particularly in remote or underserved regions where access to expert agronomic advice may be limited.

Overall, the plant disease detection project represents a powerful example of how technology can be harnessed to address critical challenges in agriculture, leading to improved crop yields, reduced pesticide usage, and sustainable farming practices for a healthier planet and better livelihoods.

7.2 Future work

We wish to keep working on our project to make it a marketable and leading product. Some of the additions that we feel are needed for it to happen are :

- Make it deployable on Cloud for global accessibility.
- Adding Fertilizer recommendation section in future to help farmer to use best fertilizer required at the moment.
- Adding E-commerce section in future ease of accessibility of goods required for farming.
- Predicts and filter the product as per the user’s behaviour and track all user activity for future recommendations using collaborative algorithm with cosine similarity.

CHAPTER - VII

REFERENCES

VII. REFERENCES

[1] Amina Khatra, “Yellow Rust Extraction in Wheat Crop based on Color Segmentation Techniques”, December 2013.

[2] Ashwini T Sapka, Uday V Kulkarni, “Comparative study of Leaf Disease Diagnosis system using Texture features and Deep Learning Features”, 2018.

[3] Simranjeet Kaur, Geetanjali Babbar, Navneet Sandhu, Dr. Gagan Jindal, “VARIOUS PLANT DISEASES DETECTION USING IMAGE PROCESSING METHODS”, June 2019.

[4] Sumit Nema, Bharat Mishra and Mamta Lambert, “Android Application of Wheat Leaf Disease Detection and Prevention using Machine Learning”, April 2020.

[5] Sachin D Khirade, Amit B Patil, “Plant disease detection using image processing”, 2015.

[6] Muhammad Hammad Saleem, Johan Potgieter, Khalid Mahmood Arif, “Plant disease detection and classification by deep learning”, 2019.

[7] Davinder Singh, Naman Jain, Pranjali Jain, Pratik Kayal, Sudhakar Kumawat, Nipun Batra, “PlantDoc: A dataset for visual plant disease detection”, 2020.

[8] U Shruthi, V Nagaveni, BK Raghavendra, “A review on machine learning classification techniques for plant disease detection”, 2019.

[9] Federico Martinelli, Riccardo Scalenghe, Salvatore Davino, Stefano Panno, Giuseppe Scuderi, Paolo Ruisi, Paolo Villa, Daniela Stroppiana, Mirco Boschetti, Luiz R Goulart, Cristina E Davis, Abhaya M Dandekar, “Advanced methods of plant disease detection. A review”, 2015

[10] Er.Varinderjit Kaur , Dr.Ashish Oberoi, “WHEAT DISEASE DETECTION USING SVM CLASSIFIER”, August 2018