PLANT DISEASE IDENTIFICATION

A Project report submitted to the

G.T.N ARTS COLLEGE (AUTONOMOUS)

(Affiliated to Madurai Kamaraj University)

(Re-accredited by NAAC with “A++”Grade (2nd cycle))

DINDIGUL

In the partial fulfillment for the award of the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

Submitted by

M.MOHAMEDTHARIK (22PCSA012)

Under the guidance of

R.SANTHINI RAJESWARI., M.C.A., M.Phil.,

(Assistant Professor, PG Computer Science)

PG DEPARTMENT OF M.SC COMPUTER SCIENCE

G.T.N ARTS COLLEGE (AUTONOMOUS)

DINDIGUL-624005

APRIL-2024

GTN ARTS COLLEGE (AUTONOMOUS)

DINDIGUL-624005

PG DEPARTMENT OF M.SC COMPUTER SCIENCE

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **“PLANT DISEASE IDENTIFICATION”** is a bonafide record work done by **M.MOHAMEDTHARIK (Reg.No:22PCSA012)** in partial fulfillment of the requirement for the Degree of Master of Science in computer Science during the year 2023- 2024 and this represents the original work of the candidates.

PROJECT GUIDE **HEAD OF THE DEPARTMENT**

Submitted for the viva – voce examination held on____________at G.T.N Arts College (Autonomous), Dindigul.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

SAMCORE SOLUTION

PROJECT COMPLETION LETTER

Dear Sir/Madam,

This is to certify that **Mr. MOHAMEDTHARIK M (Reg. No: 22PCSA012)** student of **GTN ARTS COLLEGE** studying **MSC CS (MASTER OF COMPUTER SCIENCE)** has completed his final year project title as **"PLANT DISEASE IDENTIFICATION"** in our organization from **January 2024 to April 2024**.

We congratulate you on your presence with us. We are confident that your presence with would bear a resemblance to success in the future endeavors.

For **SAMCORE SOLUTION,**

SAMCORE SOLUTION ★ TRICHY-2 ★

[illegible]

ANNAPOORANI
HEAD - ADMIN

#16, Rangaa Complex, Anna Silai, Chatram Bus Stand, Trichy - 620 002
+91-98432-07599 samcoresolution@gmail.com www.samcoresolution.in

PG DEPARTMENT OF COMPUTER SCIENCE

G.T.N ARTS COLLEGE (AUTONOMOUS)

DINDIGUL-624005

DECLARATION

I hereby declare that the Project entitled **"PLANT DISEASE IDENTIFICATION"** submitted to **G.T.N. ARTS COLLEGE (AUTONOMOUS),** Dindigul. In the partial fulfillment for the award of the degree of Master of Computer Science **"PLANT DISEASE IDENTIFICATION"** is a record of original work done by me during the period of my study in the PG Department of M.Sc. Computer Science, G.T.N. Arts College (Autonomous), Dindigul. Under the supervision and guidance of **R.SANTHINI RAJESWARI., M.C.A., M.Phil.,** Assistant Professor, PG Department of M.Sc. Computer Science, G.T.N. Arts College (Autonomous),Dindigul.

I hereby declare this project has not been submitted anywhere else for the award of any diploma or degree.

Place:- Dindigul **M.MOHAMEDTHARIK**

Date:- **(22PCSA012)**

ACKNOWLEDGEMENET

This report is an outcome individual work. I express my profound gratitude to all the persons who involved in this process.

I show my gratitude and sincere thanks to my Principal **DR.P.BALAGURUSAMY**, **M.A., M.Phil., M.Ed., PGDCA., Ph.D., G.T.N. Arts College (Autonomous),** who is a man inspiring us with distinct qualities and abilities. I show my grateful to him providing for us a conductive atmosphere in laboratory.

I also thank my **Vice Principal (SSC) DR. U.NATARAJAN, M.A., M.Phil., B.Ed., MBA. ,Ph.D., G.T.N. Arts College (Autonomous),** who gave us an opportunity.

I express my sincere gratitude to **DR. K. BOOPATHI, MCA., M.Phil., Ph.D., Head and Assistant Professor,** PG Department of M.Sc Computer Science and also extend my gratitude to my project guide **R.SANTHINI RAJESWARI., M.C.A., M.Phil., Assistant Professor,** PG Department of M.Sc Computer Science**.** She is a woman with distinct qualities, who has immediately helped to complete the project successfully by giving necessary guidelines. For showing her interest at every stage and her valuable suggestion and having guided us right from the beginning of this project.

I extend my genuine thanks to all the teaching faculties of my department for the precious guidance provided by them.

I extend my gratitude to my parents for their financial support without which my higher studies would have been only a dream.

ABSTRACT

The project entitled **“Plant Disease Identification”** is crucial for ensuring plant quality and food security. Recent advances in computer vision and machine learning techniques have made it possible to accurately detect, classify, and monitor plant diseases based on images of leaves. Early and accurate identification of plant diseases can help farmers take preventive measures to control their spread. In recent years, image processing and machine learning techniques have shown promising results in identifying plant diseases from images. One of the important and tedious tasks in agricultural practices is detection of disease on plant leafs. It requires huge time as well as skilled labour. This proposes a smart and efficient technique for detection of plant disease

TABLE OF CONTENTS

List of Figures

1. INTRODUCTION

1.1 Problem Description

Plant disease analysis using machine learning involves developing a computer system that can quickly and accurately detect diseases in plants by analyzing images of their leaves or other parts. The system is trained on a large dataset of images, including healthy plants and those with different diseases, to learn how to identify signs of disease. The system typically involves several steps, including taking pictures of the plants, cleaning and enhancing the images, identifying key characteristics of the images that indicate disease, and using a machine learning algorithm to classify the images as healthy or diseased.

1.2 Existing System

Leaf shape description is that the key downside in leaf identification. Up to now, several form options are extracted to explain the leaf form. However, there's no correct application to classify the leaf once capturing its image and identifying its attributes, however. In plant leaf classification leaf is classed supported its completely different morphological options.

A number of the classification techniques used are:

- Fuzzy logic
- Principal component Analysis
- k-Nearest Neighbors Classifier

1.3 Proposed System

The main purpose of proposed system is to detect the diseases of plant leaves by using feature extraction methods where features such as shape, color, and texture are taken into consideration. Convolutional neural network (CNN), a machine learning technique is used in classifying the plant leaves into healthy or diseased and if it is a diseased plant leaf, CNN will give the name of that particular disease. Suggesting remedies for particular disease is made which will help in growing healthy plants and improve the productivity.

2. ORGANIZATION PROFILE

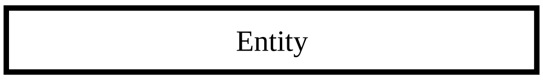
SamCore Solution as a leading IT solution and service provider provides innovative information technology - enabled solutions and services to meet the demands arising from social transformation, shaping new life styles for individuals and creating values for the society. Focusing on software technology, SamCore solution provides industry solutions and product engineering solutions, related software products, platforms, and services, through seamless integration of software and services, software and manufacturing, as well as technology and industrial management capacity. SamCore Solution helps industry customers establish best practices in business development and management. The SamCore solution serves include real time projects, web designing, web hosting, software development and training etc, in many of which, has a leading market share. Notably, SamCore Solution has participated in the formulation of many national IT standards and specifications. SamCore solution has the world's leading product engineering capabilities, ranging from consultation, design and integration to testing of embedded software, in the fields of automotive electronics, smart devices, digital home products, and IT products. The software provided by SamCore solution runs in a number of globally renowned brands.

3. SYSTEM DESIGN

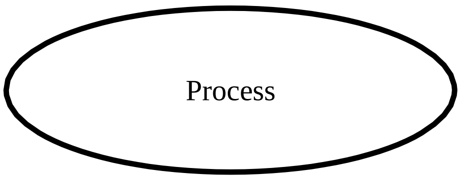
3.1 Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. s. DFD is a designing tool used in the top down approach to Systems Design. Symbols and Notations Used in DFDs Using any 10 convention’s DFD rules or guidelines, the symbols depict the four components of data flow diagrams

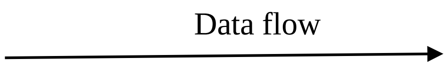
External entity: an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.



Process: any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules.



Data flow: the route that data takes between the external entities, processes and data stores.



3.2 System flow diagram

Figure 2.1 shows the system flow for the plant disease detection in the images. It is a way to display the data flow of the system and how the decisions are made during the condition and to control the events. It is the graphical representation of flow of data in a system. Each symbol is specified a process.

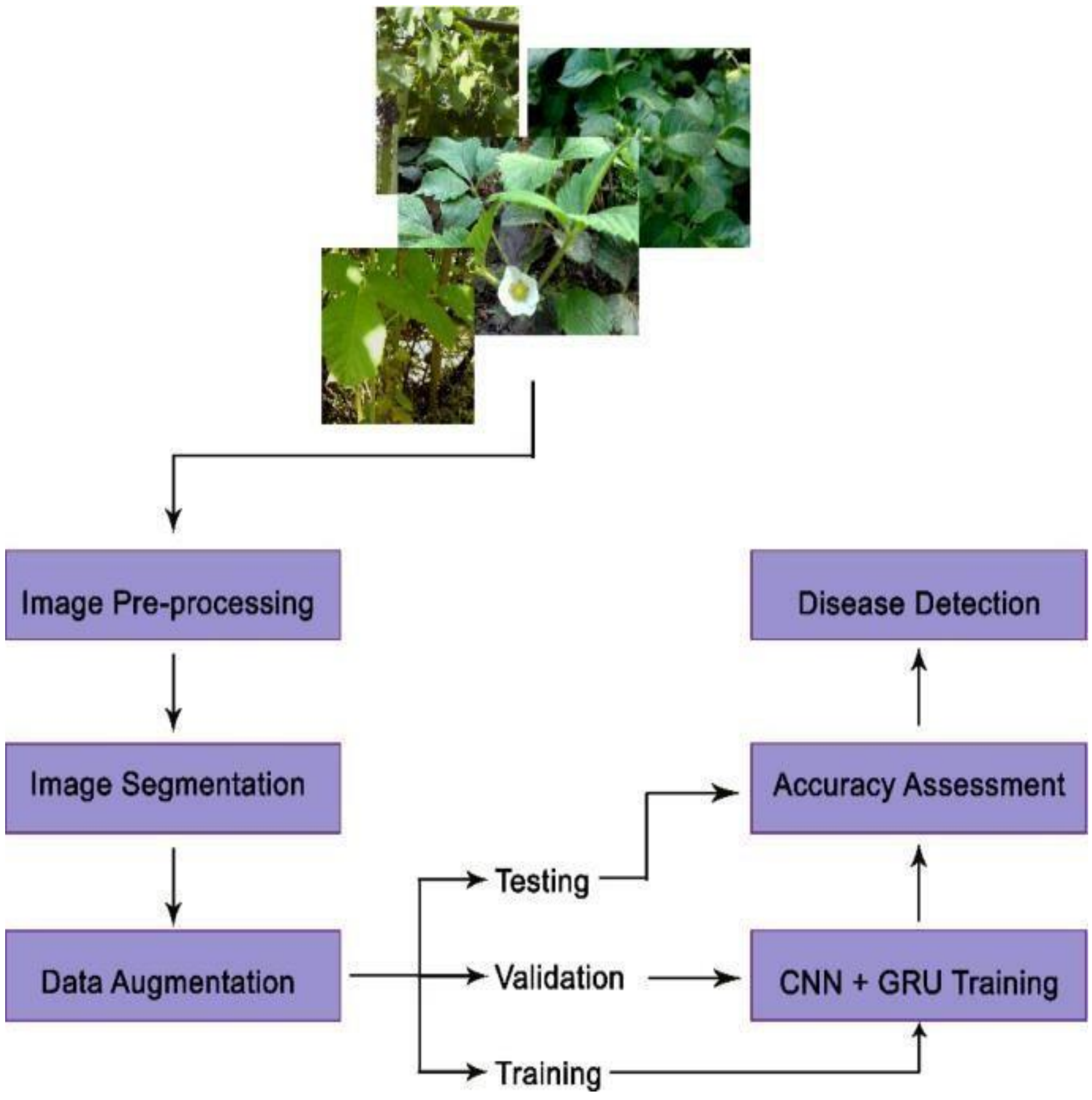


Fig. 3.1 System Flow Diagram

3.3 Use case Diagram

Figure 2.2 shows the use case diagram for the plant disease detection. It is the oneof graphical representation of the user's interaction with the system. Use case diagram is used to describe the high-level functions and scope of the system. It is also used to identify the interactions between the user and the system.

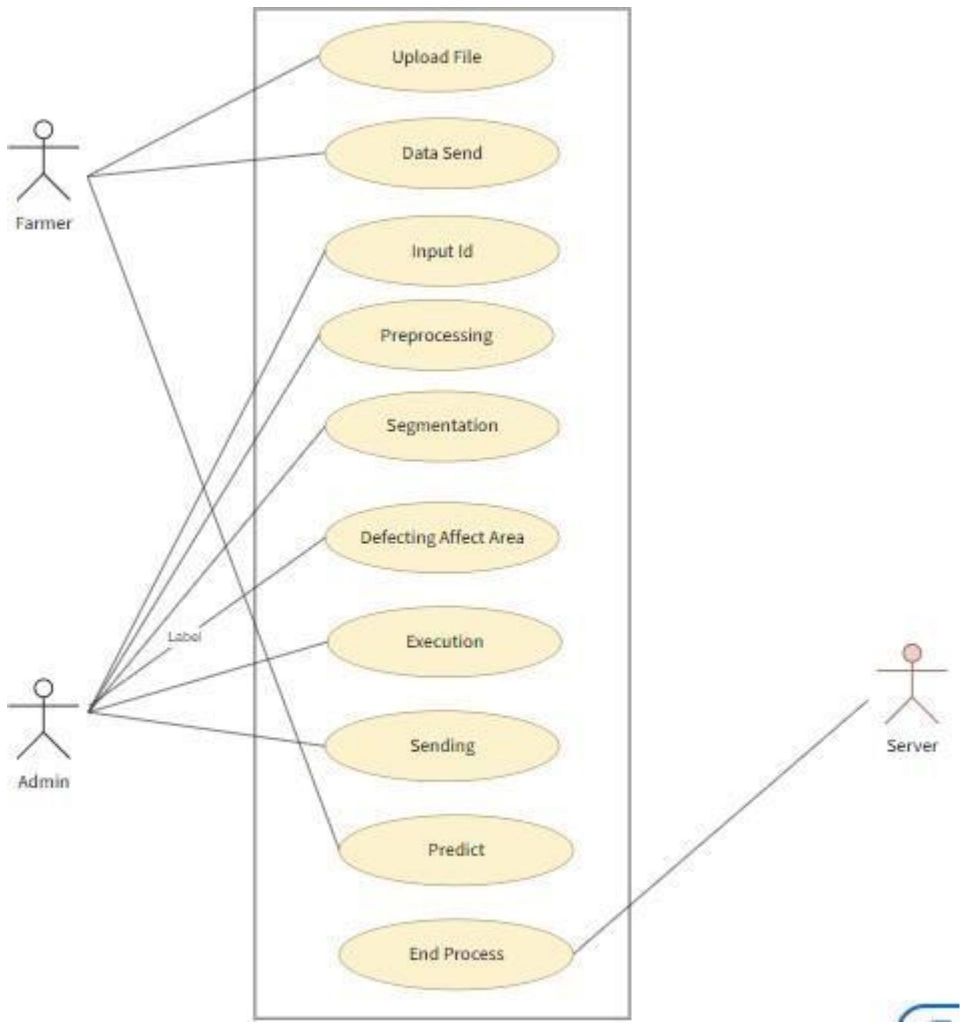


Fig. 3.2 Use case diagram

3.4 Module Design

- Data Collection
- Model Building
- Model Training
- Model Testing
- Predict
- Display result

Module: Model Testing

Model testing is the process of evaluating the performance of the trained machine learning model on a new and unseen set of images that were not used during training. During testing, the model is presented with a set of unlabeled images, and it makes predictions about whether the plant is healthy or diseased based on what it learned during training. The accuracy of the model is calculated by comparing its predicted output to theactual output for each image in the test set. The purpose of testing is to ensure that the model can generalize to new and unseen data and to identify any potential issues with overfitting or underfitting. The performance of the model during testing can guide further improvements to the model and help determine its suitability for real-world applications.

Module: Predict

Model prediction is the process of using the trained machine learning model to make predictions about the health status of plants based on new, unseen images. Once the modelis trained and tested, it can be deployed in a real-world application for plant disease diagnosis and management. When a new image of a plant is presented to the model, it analyzes the image and makes a prediction about whether the plant is healthy or diseased. The output of the model can be in the form of a binary classification (healthy/diseased) or a multi-class classification (different types of diseases). The accuracy of the model's predictions will depend on the quality of the training data, the complexity of the model, and the specific requirements of the application.

Module: Display result

The image with the detected plants, the plant disease and the solutions are display. The resultant images are show by using the “disease_disc()” function. The result is display on the website that's visible to the user.

4. SYSTEM IMPLEMENTATION

4.1 SOFTWARE SPECIFICATION

OS	: Windows 10
Platform	: Spyder
Languages	: Python Language

4.1.1 HARDWARE REQUIREMENTS

Processor	: Intel Core 3 or Higher
RAM Memory	: 2GB or Above
Disk Space	: Minimum 10 GB

4.2 SOFTWARE PACKAGE DETAILS

Flask

In python, Flask is a popular Python web framework used for building web applications, including those that integrate machine learning models. Flask can be used in plant disease analysis applications to create a web interface that allows users to upload images of plants for diagnosis.

Here are some possible steps for using Flask in plant disease analysis:

- Develop a machine learning model for plant disease analysis using libraries like Keras, Tensorflow, or PyTorch.
- Build a Flask application that accepts image uploads from users and passes them to the machine learning model for diagnosis. The Flask app can use the request module to receive image files from the user and the predict function of the machine learning model to make predictions.

- Use Flask to create a user-friendly interface that displays the results of the machine learning model's diagnosis. The interface could include a summary of the plant's health status, an image of the plant with the diseased area highlighted, and recommendations for treatment.
- Deploy the Flask application to a web server to make it accessible to users.

Overall, using Flask in plant disease analysis can provide a simple and scalable way to deploy machine learning models and make them accessible to users. Flask's flexibility and ease of use make it an excellent choice for developing web interfaces for machine learning models.

Numpy

NumPy is a popular Python package for scientific computing and data analysis that can be used in plant disease analysis using machine learning. NumPy provides support for large multidimensional arrays and matrices, and a wide range of mathematical functions to manipulate these arrays. NumPy can be used for data preprocessing and feature extraction tasks, which are critical steps in building machine learning models for plant disease analysis.

Here are some ways to use NumPy in plant disease analysis using machine learning:

- Load and preprocess image data: NumPy can be used to read and preprocess image data before feeding it into the machine learning model. NumPy functions like load and save can be used to read and write image data in various formats. NumPy functions like reshape, transpose, and stack can be used to manipulate the arrays and prepare the data for the machine learning model.
- Normalize image data: NumPy can be used to normalize image data before feeding it into the machine learning model. Normalization involves scaling the pixel values in the image data to a range between 0 and 1 or -1 and 1. This can be done using NumPy functions like min, max, mean, and std.
- Perform feature extraction: NumPy can be used to perform feature extraction on image data to identify relevant features for classification. NumPy functions like convolve and

pool can be used to extract features like edges, corners, and textures from the image data.

- Store and manipulate data: NumPy can be used to store and manipulate data generated by the machine learning model. NumPy functions like save and load can be used to store and retrieve model weights and other data. NumPy functions like argmax and sort can be used to manipulate and analyze model output.

Overall, NumPy is an essential package in plant disease analysis using machine learning that provides support for data preprocessing, feature extraction, and data manipulation. NumPy's powerful array manipulation functions and mathematical operations make it a valuable tool for building machine learning models for plant disease diagnosis and analysis.

Pandas

Pandas is a powerful Python package for data manipulation and analysis that can be used in plant disease analysis using machine learning. Here are some ways to use Pandas in plant disease analysis using machine learning:

- Data loading: Pandas can be used to load data from various sources such as CSV files, Excel sheets, SQL databases, and more. Pandas' read_csv() and read_excel() functions make it easy to load data into a Pandas DataFrame, which is a two-dimensional labeled data structure with columns of potentially different types.
- Data preprocessing: Pandas can be used for data preprocessing tasks such as handling missing values, data cleaning, data transformation, and data aggregation. Pandas' functions such as dropna(), fillna(), replace(), and groupby() are useful for handling missing data, cleaning data, transforming data, and aggregating data respectively.
- Feature engineering: Pandas can be used for feature engineering tasks such as creating new features, selecting important features, and encoding categorical features. Pandas' functions such as apply(), map(), and get_dummies() are useful for creating new features, encoding categorical features, and selecting important features respectively.
- Data visualization: Pandas can be used for data visualization tasks such as creating bar charts, histograms, scatter plots, and more. Pandas' plot() function makes it easy to create various types of plots directly from a Pandas DataFrame.

- Data merging: Pandas can be used for merging data from different sources into a single DataFrame. Pandas' merge() function makes it easy to merge data from different sources based on common columns.
- Model evaluation: Pandas can be used for evaluating the performance of machine learning models. Pandas' functions such as crosstab() and pivot_table() are useful for comparing predicted results with actual results.

Overall, Pandas is a powerful package that can be used in various tasks related to data manipulation, analysis, and visualization in plant disease analysis using machine learning. Pandas' DataFrame object provides a flexible and powerful data structure that can handle large datasets and various data types.

Pillow

Pillow is a popular Python package for image processing and manipulation that can be used in plant disease analysis using machine learning. Here are some ways to use Pillow in plant disease analysis using machine learning:

- Image loading: Pillow can be used to load image data from various file formats, such as JPEG, PNG, BMP, and more. Pillow's Image.open() function makes it easy to load image data into a PIL (Python Imaging Library) object.
- Image preprocessing: Pillow can be used for image preprocessing tasks such as resizing, cropping, rotating, and filtering. Pillow's Image.resize(), Image.crop(), Image.rotate(), and ImageFilter module functions are useful for these tasks respectively.
- Image augmentation: Pillow can be used for image augmentation tasks such as flipping, shearing, and adjusting brightness and contrast. Pillow's ImageOps module provides functions like flip(), shear(), and autocontrast() for these tasks respectively.
- Image visualization: Pillow can be used for visualizing image data by displaying images in a Jupyter notebook or saving them to a file. Pillow's Image.show() function can be used to display an image in a Jupyter notebook, while Image.save() function can be used to save an image to a file.
- Image feature extraction: Pillow can be used for extracting features from images, such as color histograms, edge detection, and texture analysis. Pillow's ImageOps module provides

functions like color_histogram(), grayscale(), and crop() for these tasks respectively.

Overall, Pillow is a powerful package that can be used in various tasks related to image processing and manipulation in plant disease analysis using machine learning. Pillow's PIL object provides a flexible and powerful data structure that can handle various image file formats and perform various image processing tasks.

Gunicorn

Gunicorn is a Python package that is commonly used as a web server gateway interface (WSGI) HTTP server to deploy web applications. In the context of plant disease analysis using machine learning, Gunicorn can be used to deploy a web application that utilizes machine learning algorithms to detect plant diseases. Here are some ways to use Gunicorn in plant disease analysis using machine learning:

- Web application deployment: Gunicorn can be used to deploy a web application that uses machine learning algorithms to detect plant diseases. The web application can be built using a web framework such as Flask or Django and can utilize machine learning models built using packages like scikit-learn or TensorFlow.
- Scalability: Gunicorn is designed to handle multiple concurrent requests and can be scaled horizontally by adding more worker processes or vertically by using a more powerful server.
- Performance: Gunicorn is known for its high performance and low memory usage. It is designed to be used with pre-fork worker model which provides a very low overhead, making it suitable for deploying web applications with high traffic.
- Load balancing: Gunicorn can be used with a load balancer such as NGINX to distribute incoming requests across multiple worker processes.
- Security: Gunicorn provides various security features such as access control, SSL encryption, and request throttling to secure the web application.

Overall, Gunicorn is a powerful package that can be used to deploy web applications that use machine learning algorithms to detect plant diseases. Its high performance, scalability, and security features make it suitable for deploying web applications with high traffic.

PyTorch

PyTorch is a popular open-source Python package used for building and training deep neural networks. In the context of plant disease analysis using machine learning, PyTorch can be used to build and train convolutional neural networks (CNNs) to classify plant diseases based on images. Here are some ways to use PyTorch in plant disease analysis using machine learning:

- Building CNNs: PyTorch provides a flexible and easy-to-use framework for building CNNs, which are widely used for image classification tasks. PyTorch's nn module provides pre-defined layers such as convolutional layers, pooling layers, and fully connected layers that can be used to build a CNN.
- Training CNNs: PyTorch provides powerful tools for training CNNs, such as automatic differentiation, which makes it easy to calculate gradients for backpropagation. PyTorch's torch.optim module provides various optimization algorithms such as Stochastic Gradient Descent (SGD) and Adam that can be used to train CNNs.
- Data loading: PyTorch provides data loading utilities such as the DataLoader class, which can be used to load image data and labels from various formats such as numpy arrays, folders, or databases.
- Transfer learning: PyTorch provides pre-trained CNN models such as VGG, ResNet, and Inception that can be used for transfer learning. Transfer learning allows using pre-trained models as a starting point and then fine-tuning them on a specific dataset.
- Inference: PyTorch provides utilities for inference, which involves using a trained model to classify new images. PyTorch's torch.nn.functional module provides various functions such as softmax and log softmax, which can be used to predict class probabilities.

Overall, PyTorch is a powerful package that can be used to build and train CNNs for plant disease analysis using machine learning. Its flexibility, ease-of-use, and pre-trained models make it a popular choice for deep learning tasks.

Jsonify:

Jsonify ia a function from the flask,python package that is used to convert python to JSON(JavaScript object Notation) format. JSON is a lightweight data-interchange format that is widely used in web applications for data transfer between client and server. In plant disease analysis, jsonify can be used to convert the output of a machine learning model to JSON format andsend it as a response to a client request. For example, if a client sends an image of a plant to a server for disease classification, the server can use a machine learning model to classify the image and send the result back to the client in JSON format using jsonify. Jsonify ensures that the data is correctly formatted as JSON and includes the necessary headers for a valid HTTP response. This makes it easy to send machine learning model outputs as JSON responses in web applications. Overall, jsonify is a useful tool for plant disease analysis using machine learning in web applications, enabling easy conversion ofPython objects to JSON format and sending them as HTTP responses.

Rendor Tmplate

In plant disease analysis using machine learning, render_template is a function from the Flask Python package that is used to generate HTML templates for web applications. It allows developers to create dynamic web pages by combining static HTML with dynamic content generated by the application. In the context of plant disease analysis using machine learning, render_template can be used to display the results of a machine learning model on a web page. For example, if a user uploads an image of a plant for disease classification, the application can use a machine learning model to classify the image and generate an HTML page showing the classification result and other relevant information. Eender_template works by taking a template file as input and rendering it with dynamic data generated by the application. The template file typically includes placeholders for the dynamic data, which are replaced with actual values at runtime. This allows developers to create reusable HTML templates that can be populated with different data based on the application logic. Overall, render_template is a powerful tool for plant disease analysis using machine learning in web applications, enabling developers to generate dynamic HTML pages with the results of machine learning models.

Request

Python package that is used to handle HTTP requests in web applications. It provides a simple way to access information from the client, such as data submitted via forms or file uploads. In the context of plant disease analysis using machine learning, request can be used to handle user input, such as uploading an image of a plant for disease classification.For example, the application can use request to retrieve the uploaded image file and pass it to a machine learning model for classification. Request provides various methods to access the data submitted by the client, such as request.form to access form data, request.files to access file uploads, and request.args to access query string parameters. It also provides methods for handling cookies, headers, and other information in the HTTP request. Overall, request is a critical tool for plant disease analysis using machine learning in web applications, enabling developers to handle user input and pass it to machine learning models for processing.

Markup

In plant disease analysis using machine learning, Markup is a class from the Flask Python package that is used to mark a string as safe for HTML rendering. It is typically used when rendering dynamic content on a web page to prevent HTML injection attacks. In the context of plant disease analysis using machine learning, Markup can be used to render machine learning results on a web page in a safe way. For example, if the application is displaying the top-k predicted classes for a plant image, the names of the classes can be marked as safe using Markup to prevent malicious input from affecting the rendering of the web page. Markup works by wrapping a string in a safe tag that tells the web browser to render the content as HTML without interpreting any of the HTML tags or scripts in the content. This prevents HTML injection attacks, which occur when an attacker injects malicious code into the content of a web page that is executed by the victim's browser. Overall, Markup is a critical tool for plant disease analysis using machine learning in web applications, enabling developers to render dynamic content on a web pagesafely and prevent HTML injection attacks.

Python

Python 3.10 is the latest version of the Python programming language, released in October 2021. It comes with several new features and improvements that enhance the language's performance, readability, and functionality. Here are some notable features of Python 3.10:

- Parenthesized context managers: Python 3.10 introduces a new syntax that allows using multiple context managers in a single with statement using parentheses. This improves readability and reduces code duplication.
- Structural Pattern Matching: Python 3.10 introduces structural pattern matching, which allows matching complex data structures using patterns. This feature can simplify code and improve readability in certain use cases.
- Improved error messages: Python 3.10 comes with improved error messages that provide more context and information about the cause of an error. This can help developers debug their code more efficiently.
- New syntax for specifying decorators: Python 3.10 introduces a new syntax for specifying decorators that simplifies the process of decorating functions and methods.

Performance improvements: Python 3.10 includes several performance improvements, such as faster module loading and faster method calls, which make the language faster andmore efficient Overall, Python 3.10 is a significant release that brings several new features and improvements to the language. Its enhanced functionality, readability, and performance make it a great choice for developers who want to build high-quality applications.

As of January 2022, the Python Package Index (PyPI), the official repository for third-party Python software, contains over 350,000 packages with a wide range of functionality, including :.

- Automation
- Data analytics
- Databases
- Documentation
- Graphical user interfaces
- Image processing

• Machine learning

• Mobile apps

• Multimedia

• Computer networking

• Scientific computing

• System administration

• Test frameworks

• Text processing

• Web frameworks

• Web scraping

Anaconda Navigator

Anaconda Navigator is a graphical user interface (GUI) for managing and launching applications that are included in the Anaconda distribution of Python. It is designed to simplify the process of working with Python and related packages by providing a visual interface for common tasks such as creating and managing environments, installing packages, and launching applications. Anaconda Navigator includes a dashboard that displays all available applications and tools that are installed in the current environment. Users can easily create and switch between multiple environments for different projects, and can manage package dependencies for each environment. In addition to managing environments and packages, Anaconda Navigator also provides a way to launch popular applications such as Jupyter Notebook, JupyterLab, Spyder, and RStudio. These applications are pre-configured with the necessary packages and dependencies, making it easy to get started with data analysis, machine learning, and scientific computing. Anaconda Navigator also provides a way to manage and launch other applications and tools that are not included in the Anaconda distribution, such as web browsers and text editors.Overall, Anaconda Navigator is a powerful and user-friendly tool for managing and launching applications in the Anaconda distribution of Python. Its visual interface and intuitive workflow make it an ideal choice for beginners and experienced users alike.

Spyder (IDLE)

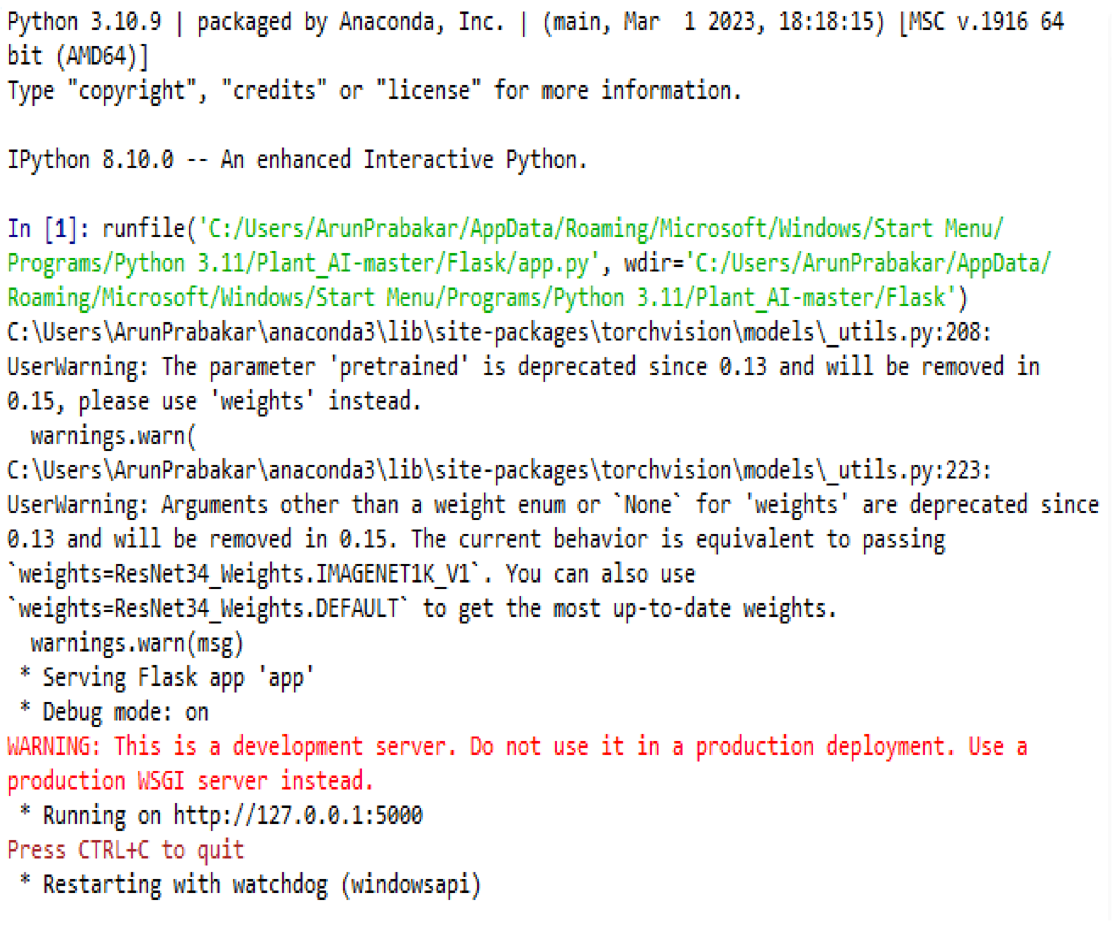
Spyder is an integrated development environment (IDE) for Python programming that is specifically designed for scientific computing, data analysis, and machine learning. It provides a comprehensive set of tools for developing, testing, and debugging Python code, as well as advanced features for data analysis and visualization. One of the main advantages of Spyder is its integrated console, which allows users to execute Python code and view the output directly within the IDE. This makes it easy to interactively test and debug code, and to explore data using Python's interactive data analysis libraries such as NumPy and Pandas. Spyder also includes a code editor with advanced features such as syntax highlighting, code folding, and code completion, as well as a debugger for tracing and fixing errors in code. It also supports version control systems like Git and Mercurial, and provides integration with popular scientific computing libraries such as Matplotlib, SciPy, and Scikit-learn. In addition to its development features, Spyder also includes advanced data analysis tools such as variable explorer, data viewer, and a profiler for analyzing code performance. It also includes a plot pane for creating and visualizing graphs and charts. Overall, Spyder is a powerful and user-friendly IDE for Python programming, particularly for scientific computing and data analysis. Its comprehensive set of features and integration with popular scientific computing libraries make it an ideal choice for developers and data scientists working with Python.

4.3 IMPLEMENTATION DETAILS

The following steps are required to load and run the program.

- Open the Spyder IDLE and Open the Python code.
- Run the Python File and the IP Address is Generated.
- Copy the Port number and Paste at the Browser.
- Run the Browser.

4.4 RESULT DISCUSSION



```
Python 3.10.9 | packaged by Anaconda, Inc. | (main, Mar  1 2023, 18:18:15) [MSC v.1916 64
bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.10.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/ArunPrabakar/AppData/Roaming/Microsoft/Windows/Start Menu/
Programs/Python 3.11/Plant_AI-master/Flask/app.py', wdir='C:/Users/ArunPrabakar/AppData/
Roaming/Microsoft/Windows/Start Menu/Programs/Python 3.11/Plant_AI-master/Flask')
C:\Users\ArunPrabakar\anaconda3\lib\site-packages\torchvision\models\_utils.py:208:
UserWarning: The parameter 'pretrained' is deprecated since 0.13 and will be removed in
0.15, please use 'weights' instead.
  warnings.warn(
C:\Users\ArunPrabakar\anaconda3\lib\site-packages\torchvision\models\_utils.py:223:
UserWarning: Arguments other than a weight enum or `None` for 'weights' are deprecated since
0.13 and will be removed in 0.15. The current behavior is equivalent to passing
`weights=ResNet34_Weights.IMAGENET1K_V1`. You can also use
`weights=ResNet34_Weights.DEFAULT` to get the most up-to-date weights.
  warnings.warn(msg)
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a
production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
```

Fig. 4.1 Output Prompt

Figure 4.1 shows the file running and the IP Address generation.

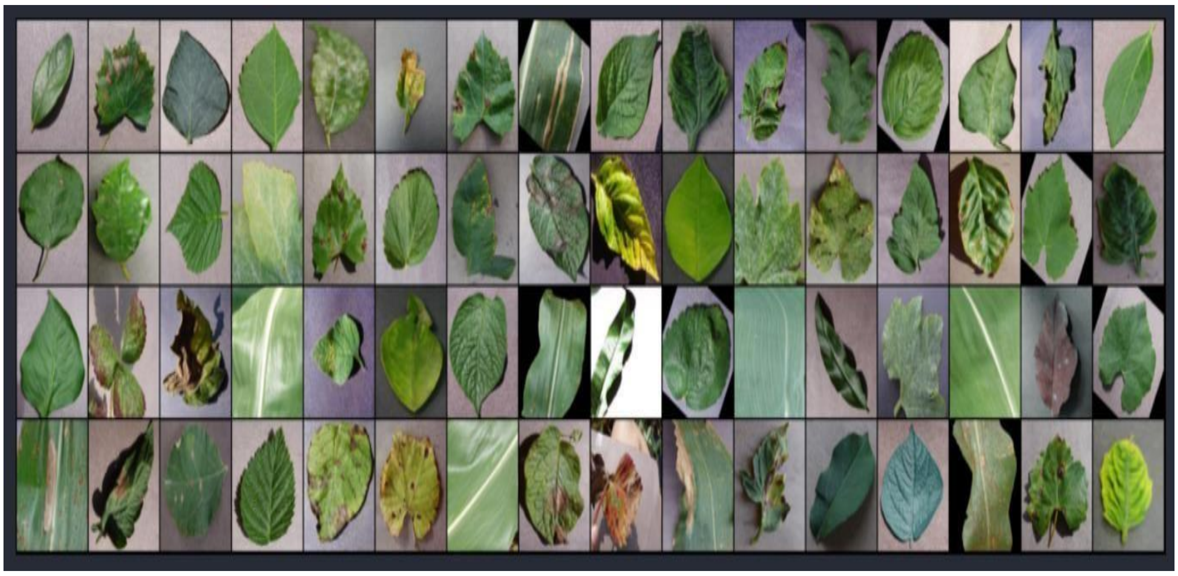
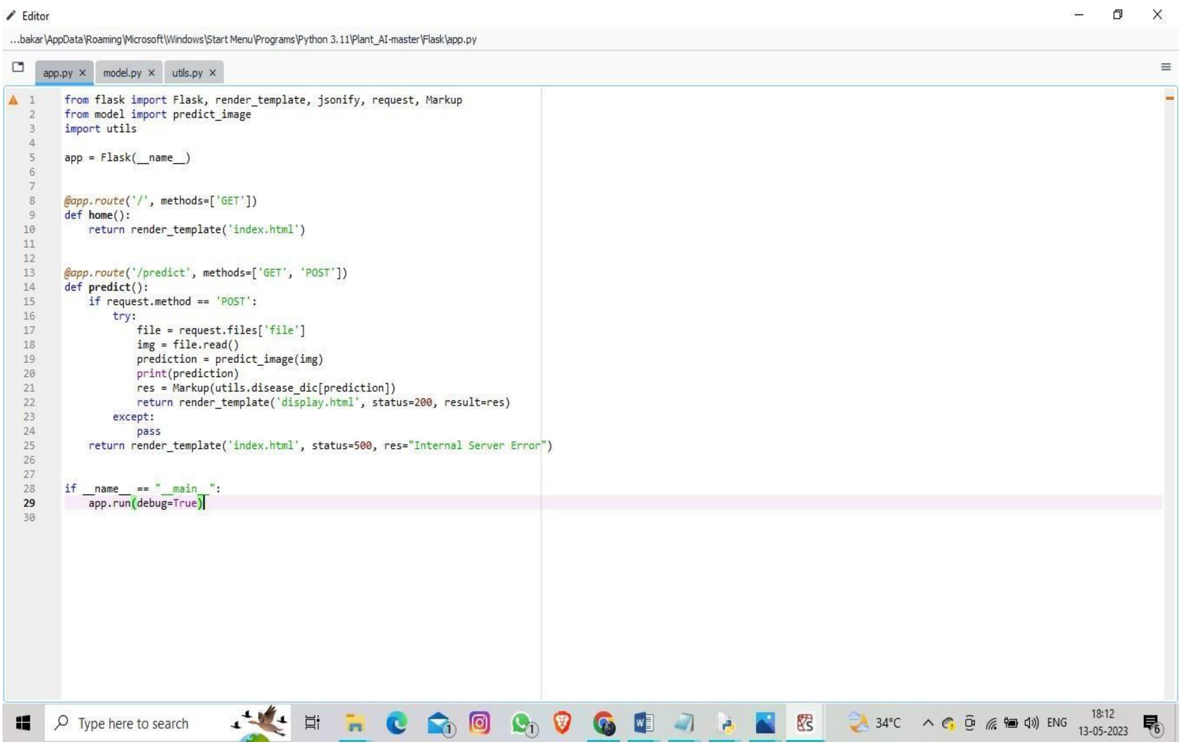


Fig. 4.2 Leaf Collection

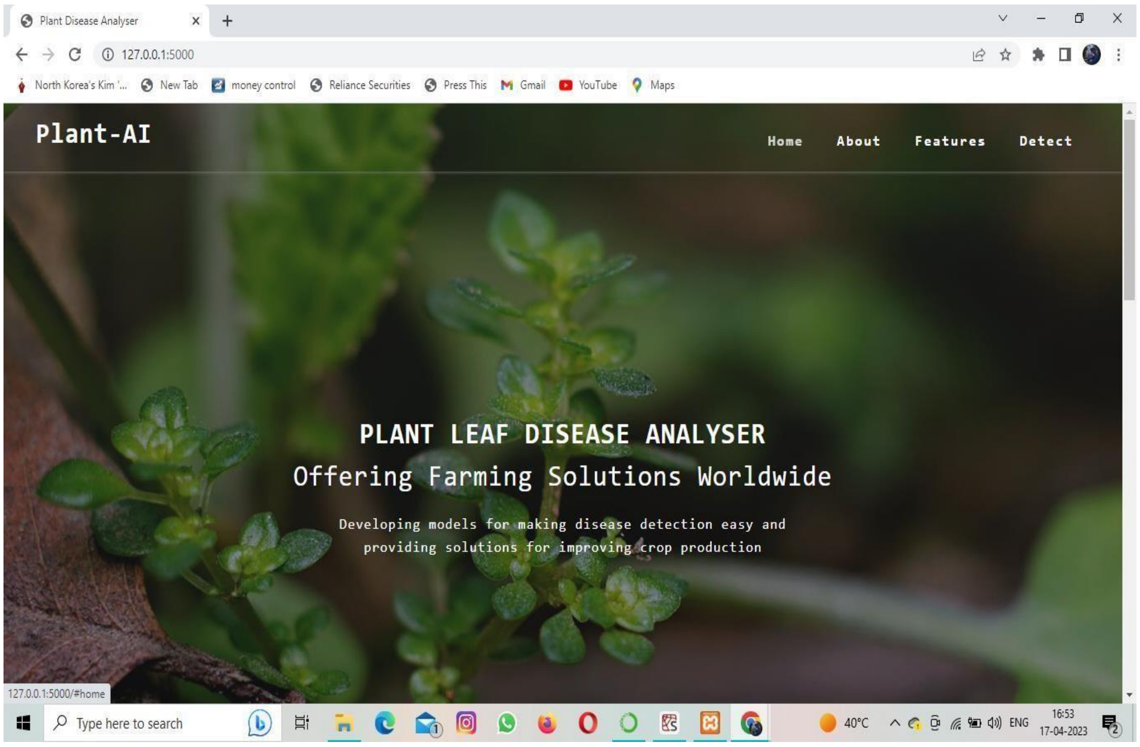
Fig 4.2 Shows The Total Leaf Collection Of this Project


Editor
...bakar\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Python 3.11\Plant_AI-master\Flask\app.py
app.py
model.py
utils.py
Type here to search
34°C
ENG
18:12
13-05-2023

```python
from flask import Flask, render_template, jsonify, request, Markup
from model import predict_image
import utils

app = Flask(__name__)


@app.route('/', methods=['GET'])
def home():
    return render_template('index.html')


@app.route('/predict', methods=['GET', 'POST'])
def predict():
    if request.method == 'POST':
        try:
            file = request.files['file']
            img = file.read()
            prediction = predict_image(img)
            print(prediction)
            res = Markup(utils.disease_dic[prediction])
            return render_template('display.html', status=200, result=res)
        except:
            pass
    return render_template('index.html', status=500, res="Internal Server Error")


if __name__ == "__main__":
    app.run(debug=True)

```

Fig. 4.3 Spyder Platform

Fig 4.3 Shows Spyder Platform used for this project

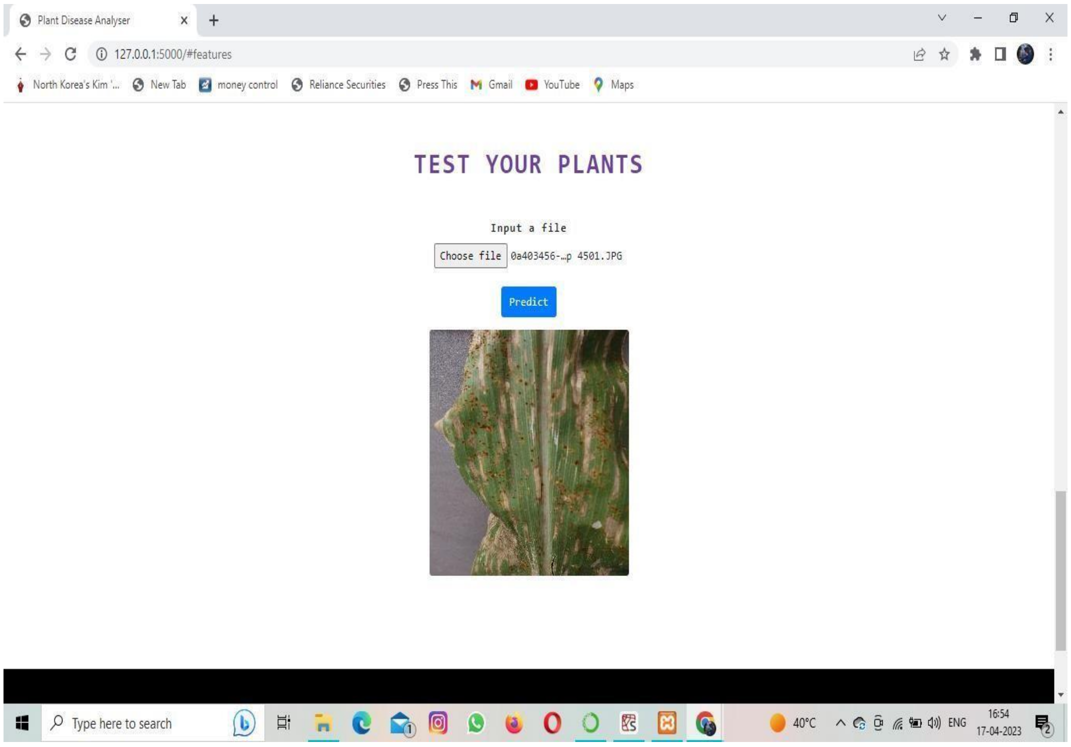


PLANT LEAF DISEASE ANALYSER

Offering Farming Solutions Worldwide

Developing models for making disease detection easy and providing solutions for improving crop production

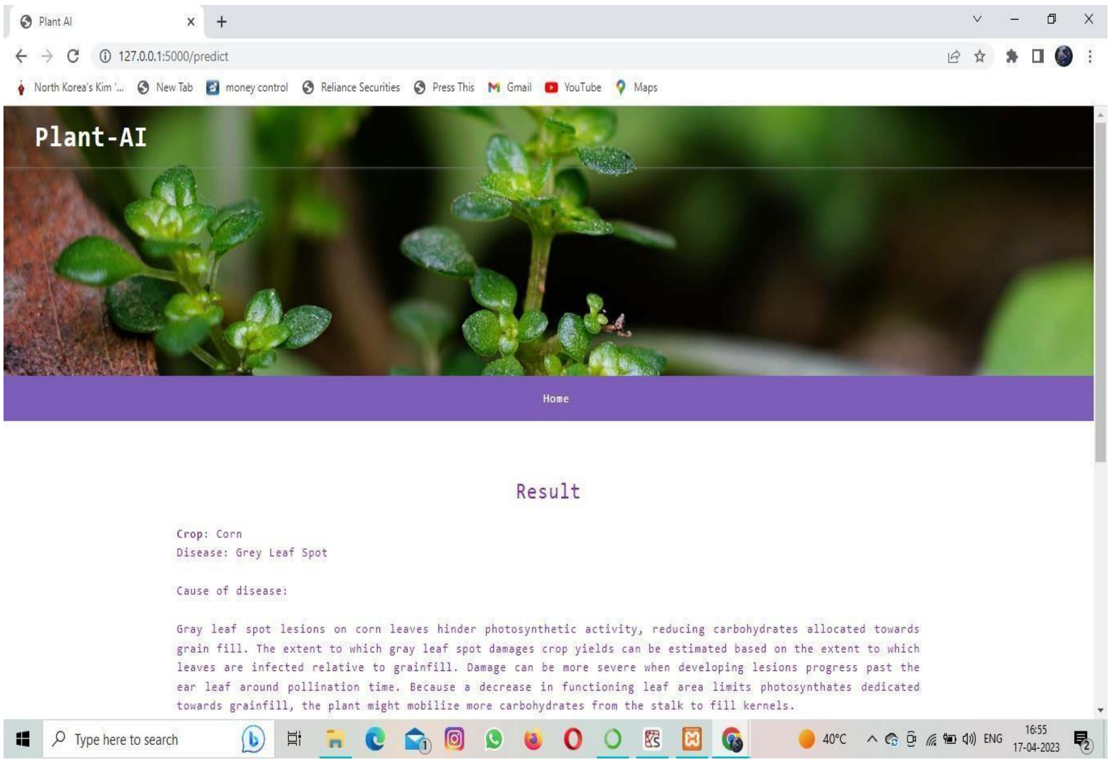
Fig. 4.4 Run at Browser



TEST YOUR PLANTS

Input a file

Fig 4.5 Disease Prediction Page



Result

Crop: Corn
Disease: Grey Leaf Spot

Cause of disease:

Gray leaf spot lesions on corn leaves hinder photosynthetic activity, reducing carbohydrates allocated towards grain fill. The extent to which gray leaf spot damages crop yields can be estimated based on the extent to which leaves are infected relative to grainfill. Damage can be more severe when developing lesions progress past the ear leaf around pollination time. Because a decrease in functioning leaf area limits photosynthates dedicated towards grainfill, the plant might mobilize more carbohydrates from the stalk to fill kernels.

Fig 4.6 Result Page

Fig 4.4, 4.5, 4.6 Shows the Project Runs on the Web Browser

4.5 SAMPLE CODE

App.py

```
from flask import Flask, render_template, jsonify, request, Markup

from model import predict_image

import utils

app = Flask(__name__)


@app.route('/', methods=['GET'])

def home():

    return render_template('index.html')



@app.route('/predict', methods=['GET', 'POST'])

def predict():

    if request.method == 'POST':

        try:

            file = request.files['file']

            img = file.read()

            prediction = predict_image(img)

            print(prediction)

            res = Markup(utils.disease_dic[prediction])

            return render_template('display.html', status=200, result=res)

        except:

            pass

    return render_template('index.html', status=500, res="Internal Server Error")
```

Model.py

```
import torch
import torch.nn as nn

import    torchvision.models    as    models

import torchvision.transforms as transforms

from PIL import Image

import io



class Plant_Disease_Model(nn.Module):



    def________init_(self):

        super()._init_()

        self.network = models.resnet34(pretrained=True)

        num_ftrs          =          self.network.fc.in_features

        self.network.fc = nn.Linear(num_ftrs, 38)



    def    forward(self,    xb):

        out = self.network(xb)

        return out



transform = transforms.Compose(

    [transforms.Resize(size=128),

    transforms.ToTensor()])
```

```
'Apple___healthy', 'Blueberry___healthy', 'Cherry_(including_sour)___Powdery_mildew',
'Cherry_(including_sour)___healthy', 'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot',
'Corn_(maize)___Common_rust_', 'Corn_(maize)___Northern_Leaf_Blight',
'Corn_(maize)___healthy', 'Grape___Black_rot', 'Grape___Esca_(Black_Measles)',
'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)', 'Grape___healthy',
'Orange___Haunglongbing_(Citrus_greening)', 'Peach___Bacterial_spot', 'Peach___healthy',
        'Pepper,_bell___Bacterial_spot', 'Pepper,_bell___healthy', 'Potato___Early_blight',
'Potato___Late_blight', 'Potato___healthy', 'Raspberry___healthy', 'Soybean___healthy',
'Squash___Powdery_mildew', 'Strawberry___Leaf_scorch', 'Strawberry___healthy',
'Tomato___Bacterial_spot', 'Tomato___Early_blight', 'Tomato___Late_blight',
'Tomato___Leaf_Mold', 'Tomato___Septoria_leaf_spot', 'Tomato___Spider_mites Two-
spotted_spider_mite', 'Tomato___Target_Spot', 'Tomato___Tomato_Yellow_Leaf_Curl_Virus',
'Tomato___Tomato_mosaic_virus', 'Tomato___healthy']


model = Plant_Disease_Model()
model.load_state_dict(torch.load(
  './Models/plantDisease-resnet34.pth',          map_location=torch.device('cpu')))
model.eval()


def predict_image(img):
  img_pil = Image.open(io.BytesIO(img))
  tensor = transform(img_pil)
  xb = tensor.unsqueeze(0)
  yb = model(xb)
```

```
    _, preds = torch.max(yb, dim=1)
    return num_classes[preds[0].item()]
```

Utils.py

```
disease_dic = {
    'Apple___Apple_scab': """ <b>Crop</b>: Apple <br/>Disease: Apple Scab<br/>
        <br/> Cause of disease:
        <br/><br/> 1. Apple scab overwinters primarily in fallen leaves and in the soil. Disease
development is favored by wet, cool weather that generally occurs in spring and early summer.
        <br/> 2. Fungal spores are carried by wind, rain or splashing water from the ground to
flowers, leaves or fruit. During damp or rainy periods, newly opening apple leaves are extremely
susceptible to infection. The longer the leaves remain wet, the more severe the infection will be.
Apple scab spreads rapidly between 55-75 degrees Fahrenheit.
        <br/><br/> How to prevent/cure the disease <br/>
        <br/>1. Choose resistant varieties when possible.
        <br/>2. Rake under trees and destroy infected leaves to reduce the number of fungal spores
available to start the disease cycle over again next spring
        <br/>3. Water in the evening or early morning hours (avoid overhead irrigation) to give the
leaves time to dry out before infection can occur.
        <br/>4. Spread a 3- to 6-inch layer of compost under trees, keeping it away from the trunk,
to cover soil and prevent splash dispersal of the fungal spores.""",
    'Apple___Black_rot': """ <b>Crop</b>: Apple <br/>Disease: Black Rot<br/>
        <br/> Cause of disease:
        <br/><br/>Black rot is caused by the fungus Diplodia seriata (syn Botryosphaeria
```

obtusa).The fungus can infect dead tissue as well as living trunks, branches, leaves and fruits. In wet weather, spores are released from these infections and spread by wind or splashing water. The fungus infects leaves and fruit through natural openings or minor wounds.

<br/><br/> How to prevent/cure the disease <br/>

<br/>1. Prune out dead or diseased branches.

<br/>2. Prune out dead or diseased branches.

<br/>3. Remove infected plant material from the area.

<br/>4. Remove infected plant material from the area.

<br/>5. Be sure to remove the stumps of any apple trees you cut down. Dead stumps can be a source of spores.""",

'Apple___Cedar_apple_rust': """ <b>Crop</b>: Apple <br/>Disease: Cedar Apple Rust<br/>

<br/> Cause of disease:

<br/><br/>Cedar apple rust (Gymnosporangium juniperi-virginianae) is a fungal disease that depends on two species to spread and develop. It spends a portion of its two-year life cycle on Eastern red cedar (Juniperus virginiana). The pathogen’s spores develop in late fall on the juniper as a reddish brown gall on young branches of the trees.

<br/><br/> How to prevent/cure the disease <br/>

<br/>1. Since the juniper galls are the source of the spores that infect the apple trees, cutting them is a sound strategy if there aren’t too many of them.

<br/>2. While the spores can travel for miles, most of the ones that could infect your tree are within a few hundred feet.

<br/>3. The best way to do this is to prune the branches about 4-6 inches below the galls.""",

'Apple___healthy': """ <b>Crop</b>: Apple <br/>Disease: No disease<br/>

```
        <br/><br/> Don't worry. Your crop is healthy. Keep it up !!!""",

    'Blueberry___healthy': """ <b>Crop</b>: Blueberry <br/>Disease: No disease<br/>
        <br/><br/> Don't worry. Your crop is healthy. Keep it up !!!""",

    'Cherry_(including_sour)___Powdery_mildew': """ <b>Crop</b>: Cherry <br/>Disease:
Powdery Mildew<br/>
        <br/> Cause of disease:
        <br/><br/>Podosphaera clandestina, a fungus that most commonly infects young, expanding
leaves but can also be found on buds, fruit and fruit stems. It overwinters as small, round, black
bodies (chasmothecia) on dead leaves, on the orchard floor, or in tree crotches. Colonies produce
more (asexual) spores generally around shuck fall and continue the disease cycle.
        <br/><br/> How to prevent/cure the disease <br/>
        <br/>1. Remove and destroy sucker shoots.
        <br/>2. Keep irrigation water off developing fruit and leaves by using irrigation that does
not wet the leaves. Also, keep irrigation sets as short as possible.
        <br/>3. Follow cultural practices that promote good air circulation, such as pruning, and
moderate shoot growth through judicious nitrogen management.""",
    'Cherry_(including_sour)___healthy': """ <b>Crop</b>: Cherry <br/>Disease: No
disease<br/>
        <br/><br/> Don't worry. Your crop is healthy. Keep it up !!!""",
    'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot': """ <b>Crop</b>: Corn
<br/>Disease: Grey Leaf Spot<br/>
        <br/> Cause of disease:
```

<br/><br/>Gray leaf spot lesions on corn leaves hinder photosynthetic activity, reducing carbohydrates allocated towards grain fill. The extent to which gray leaf spot damages crop yields can be estimated based on the extent to which leaves are infected relative to grainfill. Damage can be more severe when developing lesions progress past the ear leaf around pollination time. Because a decrease in functioning leaf area limits photosynthates dedicated towards grainfill, the plant might mobilize more carbohydrates from the stalk to fill kernels.

<br/><br/> How to prevent/cure the disease <br/>

<br/>1. In order to best prevent and manage corn grey leaf spot, the overall approach is to reduce the rate of disease growth and expansion.

<br/>2. This is done by limiting the amount of secondary disease cycles and protecting leaf area from damage until after corn grain formation.

<br/>3. High risk factors for grey leaf spot in corn: <br/>

a. Susceptible hybrid
b. Continuous corn
c. Late planting date
d. Minimum tillage systems
e. Field history of severe disease
f. Early disease activity (before tasseling)
g. Irrigation
h. Favorable weather forecast for disease.""",

'Grape___Black_rot': """ <b>Crop</b>: Grape <br/>Disease: Black Rot<br/>

<br/> Cause of disease:

<br/><br/> 1. The black rot fungus overwinters in canes, tendrils, and leaves on the grape vine and on the ground. Mummified berries on the ground or those that are still clinging to the vines become the major infection source the following spring.

<br/> 2. During rain, microscopic spores (ascospores) are shot out of numerous, black fruiting bodies (perithecia) and are carried by air currents to young, expanding leaves. In the presence of moisture, these spores germinate in 36 to 48 hours and eventually penetrate the leaves and fruit stems.

<br/> 3. The infection becomes visible after 8 to 25 days. When the weather is wet, spores can be released the entire spring and summer providing continuous infection.

<br/><br/> How to prevent/cure the disease <br/>

<br/>1. Space vines properly and choose a planting site where the vines will be exposed to full sun and good air circulation. Keep the vines off the ground and insure they are properly tied, limiting the amount of time the vines remain wet thus reducing infection.

<br/>2. Keep the fruit planting and surrounding areas free of weeds and tall grass. This practice will promote lower relative humidity and rapid drying of vines and thereby limit fungal infection.

<br/>3. Use protective fungicide sprays. Pesticides registered to protect the developing new growth include copper, captan, ferbam, mancozeb, maneb, triadimefon, and ziram. Important spraying times are as new shoots are 2 to 4 inches long, and again when they are 10 to 15 inches long, just before bloom, just after bloom, and when the fruit has set.""",

'Corn_(maize)___healthy': """ <b>Crop</b>: Corn(maize) <br/>Disease: No disease<br/>

<br/><br/> Don't worry. Your crop is healthy. Keep it up !!!""",

'Grape___healthy': """ <b>Crop</b>: Grape <br/>Disease: No disease<br/>

<br/><br/> Don't worry. Your crop is healthy. Keep it up !!!""",

'Orange___Haunglongbing_(Citrus_greening)': """ <b>Crop</b>: Orange <br/>Disease: Citrus Greening<br/>

<br/> Cause of disease:

<br/><br/> Huanglongbing (HLB) or citrus greening is the most severe citrus disease, currently devastating the citrus industry worldwide. The presumed causal bacterial agent Candidatus Liberibacter spp. affects tree health as well as fruit development, ripening and quality of citrus fruits and juice.

<br/><br/> How to prevent/cure the disease <br/>

<br/>1. In regions where disease incidence is low, the most common practices are avoiding the spread of infection by removal of symptomatic trees, protecting grove edges through intensive monitoring, use of pesticides, and biological control of the vector ACP.

<br/>2. According to Singerman and Useche (2016), CHMAs coordinate insecticide application to control the ACP spreading across area-wide neighboring commercial citrus groves as part of a plan to address the HLB disease.

<br/>3. In addition to foliar nutritional sprays, plant growth regulators were tested, unsuccessfully, to reduce HLB-associated fruit drop (Albrigo and Stover, 2015).""",

'Peach___Bacterial_spot': """ <b>Crop</b>: Peach <br/>Disease: Bacterial Spot<br/>

<br/> Cause of disease:

<br/><br/> 1. The disease is caused by four species of Xanthomonas (X. euvesicatoria, X. gardneri, X. perforans, and X. vesicatoria). In North Carolina, X. perforans is the predominant species associated with bacterial spot on tomato and X. euvesicatoria is the predominant species

```
    'Raspberry___healthy': """ <b>Crop</b>: Raspberry <br/>Disease: No disease<br/>
        <br/><br/> Don't worry. Your crop is healthy. Keep it up !!!""",


    'Soybean___healthy': """ <b>Crop</b>: Soyabean <br/>Disease: No disease<br/>
        <br/><br/> Don't worry. Your crop is healthy. Keep it up !!!""",

    'Strawberry___healthy': """ <b>Crop</b>: Strawberry <br/>Disease: No disease<br/>
        <br/><br/> Don't worry. Your crop is healthy. Keep it up !!!""",

    'Tomato___healthy': """ <b>Crop</b>: Tomato <br/>Disease: No disease<br/>
        <br/><br/> Don't worry. Your crop is healthy. Keep it up !!!""",

    'Potato___Early_blight': """ <b>Crop</b>: Potato <br/>Disease: Early Blight<br/>
        <br/> Cause of disease:
        <br/><br/> 1. Early blight (EB) is a disease of potato caused by the fungus Alternaria
solani. It is found wherever potatoes are grown.
        <br/> 2. The disease primarily affects leaves and stems, but under favorable weather
conditions, and if left uncontrolled, can result in considerable defoliation and enhance the chance
for tuber infection. Premature defoliation may lead to considerable reduction in yield.
        <br/> 3. Primary infection is difficult to predict since EB is less dependent upon specific
weather conditions than late blight.
        <br/><br/> How to prevent/cure the disease <br/>
```

<br/>1. Plant only diseasefree, certified seed.

<br/>2. Follow a complete and regular foliar fungicide spray program.

<br/>3. Practice good killing techniques to lessen tuber infections.

<br/>4. Allow tubers to mature before digging, dig when vines are dry, not wet, and avoid excessive wounding of potatoes during harvesting and handling.""",

'Tomato___Bacterial_spot': """ <b>Crop</b>: Tomato <br/>Disease: Bacterial Spot<br/>

<br/> Cause of disease:

<br/><br/> 1. The disease is caused by four species of Xanthomonas (X. euvesicatoria, X. gardneri, X. perforans, and X. vesicatoria). In North Carolina, X. perforans is the predominant species associated with bacterial spot on tomato and X. euvesicatoria is the predominant species associated with the disease on pepper.

<br/> 2. All four bacteria are strictly aerobic, gram-negative rods with a long whip-like flagellum (tail) that allows them to move in water, which allows them to invade wet plant tissue and cause infection.

<br/><br/> How to prevent/cure the disease <br/>

<br/>1. The most effective management strategy is the use of pathogen-free certified seeds and disease-free transplants to prevent the introduction of the pathogen into greenhouses and field production areas. Inspect plants very carefully and reject infected transplants- including your own!

<br/>2. In transplant production greenhouses, minimize overwatering and handling of seedlings when they are wet.

<br/>3. Trays, benches, tools, and greenhouse structures should be washed and sanitized between seedlings crops.

<br/>4. Do not spray, tie, harvest, or handle wet plants as that can spread the disease""",

'Tomato___Tomato_mosaic_virus': """ <b>Crop</b>: Tomato <br/>Disease: Mosaic Virus<br/>

<br/> Cause of disease:

<br/><br/> 1. Tomato mosaic virus and tobacco mosaic virus can exist for two years in dry soil or leaf debris, but will only persist one month if soil is moist. The viruses can also survive in infected root debris in the soil for up to two years.

<br/> 2. Seed can be infected and pass the virus to the plant but the disease is usually introduced and spread primarily through human activity. The virus can easily spread between plants on workers' hands, tools, and clothes with normal activities such as plant tying, removing of suckers, and harvest

<br/> 3. The virus can even survive the tobacco curing process, and can spread from cigarettes and other tobacco products to plant material handled by workers after a cigarette

<br/><br/> How to prevent/cure the disease <br/>

<br/>1. Purchase transplants only from reputable sources. Ask about the sanitation procedures they use to prevent disease.

<br/>2. Inspect transplants prior to purchase. Choose only transplants showing no clear symptoms.

<br/>3. Avoid planting in fields where tomato root debris is present, as the virus can survive long-term in roots.

<br/>4. Wash hands with soap and water before and during the handling of plants to reduce potential spread between plants."""

5. TESTING

5.1 Unit Testing

Unit testing is a software testing technique used to validate individual units or components of a software system. As such, unit testing is not directly applicable to plant disease identification using machine learning, which involves training and evaluating machine learning models on large datasets. However, to ensure the correctness and reliability of the software components that implement the machine learning algorithms, you can write unit tests for the following:

Data preprocessing: Test the correctness of the code that loads, cleans, and preprocesses the plant disease dataset. Verify that the data is loaded in the expected format, any missing or incorrect data is handled appropriately, and the preprocessing steps are performed correctly.

Model training: Test the correctness of the code that trains the machine learning model. Verify that the model is trained on the correct dataset, the hyperparameters are set correctly, and the model is saved in the expected format.

Model evaluation: Test the correctness of the code that evaluates the performance of the machine learning model. Verify that the evaluation metrics are computed correctly, the results are saved in the expected format, and the model is compared against the baseline performance.

Inference: Test the correctness of the code that applies the trained machine learning model to new data to predict plant diseases. Verify that the input data is in the expected format, the model makes accurate predictions, and the output is saved in the expected format.

Integration: Test the correct integration of the above components into a complete pipeline for plant disease identification using machine learning. Verify that the components communicate with each other correctly, any dependencies are resolved correctly, and the pipeline produces the expected results.

It is important to note that unit tests are just one part of a comprehensive testing strategy, and other types of testing, such as integration testing, system testing, and user acceptance testing, may also be necessary to ensure the reliability and correctness of the system.

Integration Testing:

Integration testing is a software testing technique used to verify that individual software components work together correctly as a larger system. In the context of plant disease identification using machine learning, integration testing ensures that the various software components that make up the system, such as data preprocessing, model training, and model evaluation, integrate correctly to produce the desired results. Here are some steps you can take to perform integration testing for plant disease identification using machine learning:

Identify the software components: First, identify all the software components that make up the plant disease identification system, including data preprocessing, model training, model evaluation, and inference.

Define interfaces: Next, define the interfaces between the components. For example, the data preprocessing component might output a preprocessed dataset, which is then input into the model training component. Similarly, the trained model might be input into the inference component, which produces predictions on new data.

Develop integration tests: Develop integration tests that exercise the interfaces between the components and verify that the components work together correctly. For example, you might develop a test that loads a raw plant disease dataset into the data preprocessing component, processes the data, and outputs a preprocessed dataset. This preprocessed dataset can then be input into the model training component, which trains a machine learning model. The trained model can then be input into the inference component, which produces predictions on new data. Verify that the predictions are accurate.

Run integration tests: Run the integration tests and verify that the system produces the desired results. You might also want to use mock datasets and models to simulate different scenarios and edge cases.

Fix any issues: If any issues are discovered during integration testing, fix them and rerun the integration tests until the system works correctly.

By performing integration testing for plant disease identification using machine learning, you can ensure that the various software components work together correctly to produce accurate predictions and identify any issues early in the development process**.**

Black Box Testing:

Black box testing is a software testing technique that verifies the functionality of the system without knowledge of the internal workings of the system. In the context of plant disease identification using machine learning, black box testing involves testing the system's functionality without examining the details of the machine learning algorithms used for disease identification. Here are some steps you can take to perform black box testing for plant disease identification using machine learning:

Define the requirements: Define the requirements of the system, including accuracy, performance, and usability. These requirements should be based on the needs of the users and the goals of the system.

Develop test cases: Develop test cases that verify that the system meets the requirements. These test cases should be based on the expected input and output of the system and should not require knowledge of the internal workings of the machine learning algorithms.

Test the system: Test the system using the test cases developed in step 2. This might involve running the system on different datasets and with different settings to ensure that it performs as expected.

Verify the results: Verify that the system produces the expected results. For example, you might verify that the system correctly identifies plant diseases and provides accurate recommendations for treatment.

Fix any issues: If any issues are discovered during black box testing, fix them and rerun the tests until the system works correctly.

Perform boundary testing: Perform boundary testing to verify that the system works correctly at the edge cases of the input range. For example, you might test the system's behavior when presented with an image of a diseased plant that is partially obscured.

By performing black box testing you can verify that the system meets the requirements and performs as expected without knowledge of the internal workings of the machine learning algorithms. This helpsto ensure that the system is reliable, accurate, and easy to use for the users.

White Box Testing:

White box testing is a software testing technique that examines the internal workings of the system, including the code and algorithms used. In the context of plant disease identification using machine learning, white box testing involves examining the internal workings of the machine learning algorithms to verify that they are working correctly and efficiently. Here are some steps you can take to perform white box testing for plant disease identification using machine learning:

Understand the algorithms: Understand the machine learning algorithms used for disease identification, including the data preprocessing, feature extraction, and classification methods used.

Analyze the code: Analyze the code used to implement the machine learning algorithms. This might involve reviewing the code for errors, examining the data structures used, and understanding the logic of the algorithms.

Develop test cases: Develop test cases that examine the internal workings of the machine learning algorithms. These test cases might involve testing the accuracy of the feature extraction or classification methods, or examining the performance of the algorithms on different datasets.

Test the algorithms: Test the machine learning algorithms using the test cases developed in step 3. This might involve running the algorithms on different datasets and with different settings to ensure that they perform correctly.

Verify the results: Verify that the machine learning algorithms produce the expected results. This might involve examining the output of the algorithms to ensure that they correctly identify plant diseases and provide accurate recommendations for treatment.

Optimize the algorithms: Optimize the machine learning algorithms to improve their performance. This might involve adjusting the hyper-parameters used in the algorithms or selecting different classification methods.

By performing white box testing for plant disease identification using machine learning, you can ensure that the machine learning algorithms are working correctly and efficiently. This helps to ensure that the system is accurate, reliable, and performs well for the users.

System Testing:

System testing is a software testing technique that tests the entire system as a whole to verify that it meets its requirements and performs as expected. In the context of plant disease identification using machine learning, system testing involves testing the entire system, including all software components and hardware, to ensure that it functions correctly. Here are some steps you can take to perform system testing for plant disease identification using machine learning:

Define the requirements: Define the requirements of the system, including performance, accuracy, and usability. These requirements should be based on the needs of the users and the goals of the system.

Develop test cases: Develop test cases that verify that the system meets the requirements. For example, you might develop test cases that verify the accuracy of the machine learning model, the performance of the system under different workloads, and the usability of the user interface.

Test the system: Test the system using the test cases developed in step 2. This might involve running the system on different datasets, with different settings, and under different conditions to ensure that it performs as expected.

Verify the results: Verify that the system produces the expected results. For example, you might verify that the system correctly identifies plant diseases and provides accurate recommendations for treatment.

Fix any issues: If any issues are discovered during system testing, fix them and rerun the tests until the system works correctly.

By performing system testing for plant disease identification using machine learning, you can ensure that the entire system meets the requirements and performs as expected. This helps to ensure that the system is reliable, accurate, and easy to use, which is essential for the success of the system.

User Acceptance Testing:

User acceptance testing (UAT) is a software testing technique that verifies that the system meets the needs of the users and is easy to use. In the context of plant disease identification using machine learning, UAT involves getting feedback from users to ensure that the system is effective, efficient, and satisfying to use. Here are some steps you can take to perform user acceptance testing for plant disease identification using machine learning:

Identify the users: Identify the users of the system, including farmers, agricultural experts, and researchers. It's important to involve users from different backgrounds and with different levels of expertise to ensure that the system meets the needs of all users.

Develop user acceptance test cases: Develop test cases that simulate real-world scenarios and tasks that the users might perform. For example, you might develop test cases that simulate a farmer using the system to identify plant diseases and get recommendations for treatment.

Perform user acceptance testing: Perform the user acceptance testing by having the users perform the test cases developed in step 2. Observe their behavior and collect feedback on the system's effectiveness, efficiency, and satisfaction. Ask users to rate the system on various metrics such as accuracy, ease of use, and usefulness.

Analyze feedback: Analyze the feedback collected from the users to identify areas for improvement. Look for common issues and suggestions for improvement.

Make adjustments: Make adjustments to the system based on the feedback collected during user acceptance testing. These adjustments might include improving the user interface, changing the way the system presents information, or improving the accuracy of the machine learning model.

Repeat user acceptance testing: Repeat the user acceptance testing after making adjustments to ensure that the changes have improved the system and that it meets the needs of the users.

By performing user acceptance testing for plant disease identification using machine learning, you can ensure that the system is effective, efficient, and satisfying to use for all users. This helps to ensure that the system is adopted by users and has a positive impact on agriculture and plant disease control.

5.2 Sample Test Cases

Input: An image of a healthy plant.

Expected Output: The model should correctly classify the plant as healthy and not indicate any disease.

Input: An image of a plant infected with a known disease.

Expected Output: The model should accurately identify the disease based on visual symptoms and provide the correct disease label.

Input: An image of a plant with multiple diseases or multiple symptoms.

Expected Output: The model should be able to detect and correctly classify each disease or symptom present in the image.

Input: An image of a plant with a rare or uncommon disease.

Expected Output: The model should have the ability to identify and classify uncommon diseases accurately.

Input: A set of images of the same plant taken over a period.

Expected Output: The model should be able to track the progression of the disease and provide accurate predictions or classifications for each stage.

Input: An image with poor image quality, such as low resolution, lighting issues, or image noise.

Expected Output: The model should still be able to make reasonable predictions despite the poor image quality.

Input: An image of a plant where the disease symptoms are not clearly visible or well-defined.

Expected Output: The model should provide the most probable disease prediction based on the available visual cues and its knowledge of plant diseases.

Input: An image with background noise or other objects present in the scene.

Expected Output: The model should be able to focus on the plant and ignore irrelevant objects or background noise to make accurate disease predictions.

Input: An image of a plant that appears diseased but is actually healthy.

Expected Output: The model should correctly identify the plant as healthy and not indicate the presence of any disease.

Input: An image of a plant with a disease not present in the training dataset.

Expected Output: The model should still attempt to classify the disease or indicate that it is an unknown disease if it hasn't been trained on that particular disease.

6. CONCLUSION

Plant disease analysis using machine learning has emerged as a promising approach to help identify and diagnose plant diseases quickly and accurately. With the help of machine learning algorithms, it is possible to classify plant images based on their disease categories, which can assist farmers and agronomists in making informed decisions regarding the management and treatment of crops. The implementation of a plant disease analysis system using machine learning involves several steps, including Data Collection, Data Preprocessing, Model Selection, Model Training, Model Testing, Predict and Display result. The use of frameworks such as Flask, TensorFlow, and Pillow, among others, can greatly facilitate this process, enabling developers to build efficient and scalable solutions.

Overall, plant disease analysis using machine learning has the potential to revolutionize the way we monitor and manage crop health, helping to increase crop yields and reduce losses due to diseases. As technology continues to evolve and data becomes more accessible, we can expect to see further advancements in this field, paving the way for more efficient and sustainable agricultural practices.

7. FUTURE ENHANCEMENT

Feature Enhancement for plant disease analysis using machine learning can be enhanced to improve its accuracy and usability. Some of these feature enhancements include:

- Incorporating multi-spectral imaging: Multi-spectral imaging can capture more detailed information about the plant's condition and health than traditional RGB imaging. Incorporating this technology can improve the accuracy of disease detection.
- Developing a mobile application: Developing a mobile application that can be used by farmers and agronomists to easily identify plant diseases in the field can greatly enhance the usability of the system.
- Implementing transfer learning: Transfer learning is a technique that allows the use of pre-trained models to improve the performance of a new model with limited data. Implementing this technique can improve the accuracy of disease detection even with limited data.
- Using ensemble models: Ensemble models are machine learning models that combine the predictions of multiple models to make a final decision. Using ensemble models can improve the accuracy and robustness of disease detection.
- Integrating real-time monitoring: Integrating real-time monitoring of environmental factors such as temperature, humidity, and soil moisture can provide more context for disease detection and help predict disease outbreaks.
- Providing treatment recommendations: Providing treatment recommendations based on the identified disease can help farmers and agronomists take appropriate actions to manage the disease and prevent its spread.

Overall, incorporating these feature enhancements can greatly improve the accuracy and usability of plant disease analysis using machine learning, leading to more efficient and sustainable agricultural practices.

REFERENCES

Books

[1]. "Machine Learning Techniques for Plant Disease Detection and Diagnosis" by S. Sathishkumar and S. R. Kannan.

[2]. "Artificial Intelligence and Machine Learning in Agriculture" by Y. Shrivastava, P. Shukla, and R. K. Singh.

[3]. "Handbook of Plant Disease Identification and Management: A Comprehensive Guide for Researchers, Students and Practitioners" edited by Balaji Aglave.

[4]. "Advanced Machine Learning Technologies and Applications for Smart Agriculture" edited by Eleni I. Vlahogianni and Angelos Amditis.

[5]. "Machine Learning for Agriculture" by Vikas Chandra and R. K. Singh.

[6]. "Plant Disease Diagnosis: Current Research and Future Trends" edited by Mohammad Manjur Shah and Shawkat Ali.

Websites

- Python Installation website: https://docs.python.org/3/install/
- Python packages: https://packaging.python.org/en/latest/tutorials/installing-packages/
- https://download.pytorch.org/whl/cpu/torch-1.8.0%2Bcpu-cp36-cp36m-linux_x86_64.whl
- https://download.pytorch.org/whl/cpu/torchvision-0.9.0%2Bcpu-cp36-cp36m-linux_x86_64.whl
- The Plant Pathology Lab: http://www.plantpathology.co/
- PhytopathologyNews: https://www.apsnet.org/publications/phytopathology/Pages/default.aspx
- International Society for Plant Pathology: https://www.isppweb.org/
- Crop Protection Compendium: https://www.cabi.org/cpc/