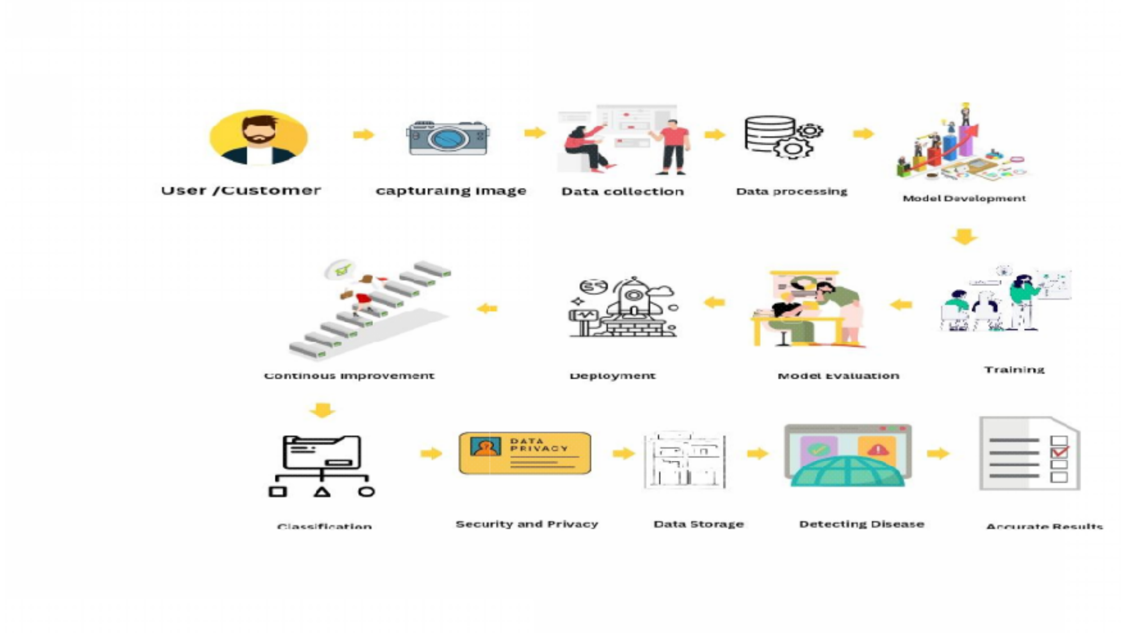
DEEP LEARNING MODEL FOR DISEASE DETECTION IN TEA LEAVES

Project Description:

Tea leaves were using widely all over the world, many are addicted to it just a like daily chore to complete in their day to get rid of all their stressful life. So we have to be careful of which tea leaves we are using to lead a healthy life, The manufacturers have to be very clear not to leave any simple matter like DISEASES IN TEA LEAVES. we propose a system that uses a deep convolutional neural network (DCNN) to automatically identify and classify different types of diseases from tea leaf images. The DCNN is a machine learning model that can learn from images and extract features that are relevant for classification. The system can take an image of a tea leaf as input and output a prediction of the disease type, such as normal, algal leaf spot, gray blight, white spot, brown blight, red scab, bud blight, or grey blight. The system can also provide suggestions and recommendations for the treatment and prevention of the diseases. The system can be deployed on a web or mobile application that allows users to upload tea leaf images and receive the diagnosis of the disease type.

Technical Architecture:



Pre requisites:

To complete this project, you must required following software's, concepts and packages

- **Visual studio**

- **Python packages:**
  - o Type “pip install numpy” and click enter.
  - O Type “pip install pandas ” and click enter.
  - o Type “pip install opencv-python” and click enter.
  - o Type “pip install scikit-learn” and click enter.
  - o Type ”pip install matplotlib” and click enter.
  - o Type ”pip install Keras” and click enter.
  - o Type ”pip install tensorflow ” and click enter.
  - o Type ”pip install OS” and click enter.
  - o Type “pip install Flask” and click enter.
  - o Type “pip install seaborn” and click enter.

Prior Knowledge:

You must have prior knowledge of following topics to complete this project. ●

DEEP LEARNING Concepts

- O Image processing
- o CNN
- o Transfer learning
- o Evaluation metrics

- **Flask Basics**

Project Objectives:

By the end of this project you will able:

- To develop a deep learning model that can accurately and efficiently detect and classify different types of tea leaf diseases from leaf images.
- To compare the performance and effectiveness of different deep learning models and algorithms for tea leaf disease detection, such as CNN.
- To provide a user-friendly and intuitive interface that allows the users to interact with the system and obtain the results and feedback of the disease detection.
- Get to know about ML and DL concepts, gaining vast experience about data and pre-processing along with transformation techniques, visualization concepts.

Project Flow:

- User interacts with the web Application toUpload/ insert the image .
- Uploaded image is analyzed by the model which is integrated.

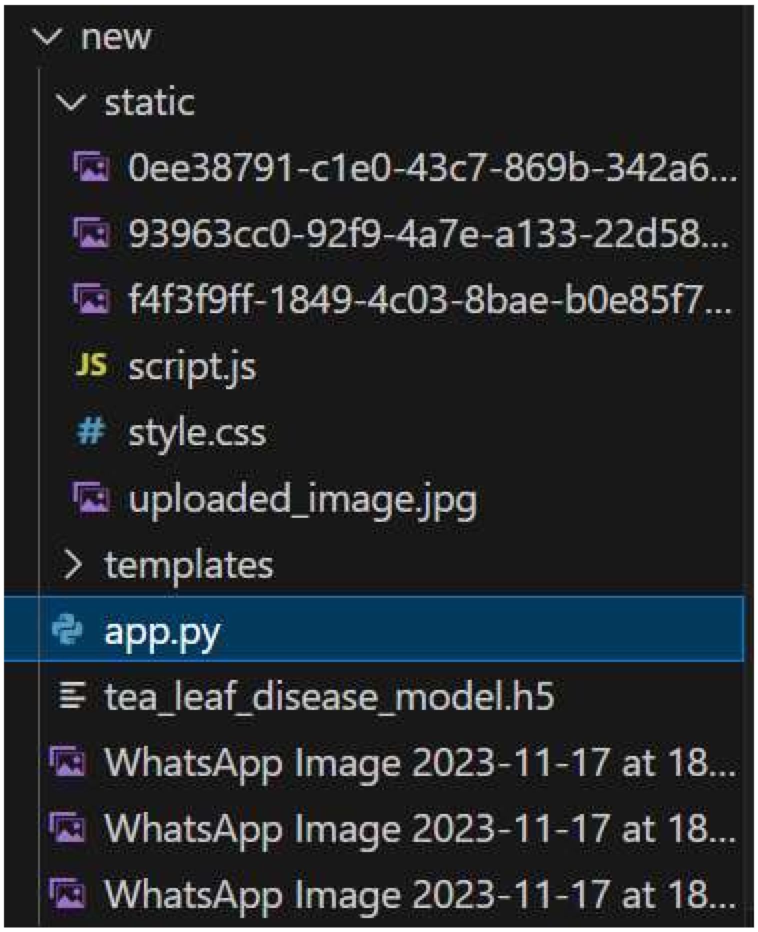
- Once model analyzes the image the prediction is showcased on the web application

To accomplish this, we have to complete all the activities listed below,

- **Data Collection**
    - Collect Tea Leaf Images
    - Label and Annotate Images
    - Collect Environmental Data (Optional)
- **Data Preprocessing**
    - Image Preprocessing
    - Resize, Normalize, Augment
    - Segment Tea Leaves (if needed)
    - Data Splitting
    - Train, Validation, Test Sets
- **Model Development**
    - Choose Deep Learning Architecture (e.g., CNN)
    - Transfer Learning (Pre-trained Models)
    - Customize Model Architecture
    - Define Loss Function (e.g., Cross-Entropy)
    - Train the Model
    - Apply Regularization Techniques
- **Model Evaluation**
    - Evaluate Model on Test Set
    - Metrics (Accuracy, Precision, Recall, etc.)
    - Ablation Studies
    - Robustness Testing
- **Deployment**
- Edge Deployment or Cloud Deployment
    - Implement User Interface
    - Ensure Scalability and Monitoring
- **Continuous Improvement**
    - Collect User Feedback
    - Continuous Learning and Retraining
    - Update Dataset
- **Security and Privacy**
    - Implement Security Measures
    - Ensure Privacy Compliance
- **Documentation**
    - Create Comprehensive Documentation
- **Maintenance and Support**
    - Establish Maintenance and Update System

Project Structure:

Create the Project folder which contains files as shown below



```
v new
  v static
      0ee38791-c1e0-43c7-869b-342a6...
      93963cc0-92f9-4a7e-a133-22d58...
      f4f3f9ff-1849-4c03-8bae-b0e85f7...
      script.js
      style.css
      uploaded_image.jpg
  > templates
    app.py
    tea_leaf_disease_model.h5
    WhatsApp Image 2023-11-17 at 18...
    WhatsApp Image 2023-11-17 at 18...
    WhatsApp Image 2023-11-17 at 18...
```

- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- app.py is our saved model. Further we will use this model for flask integration.

Milestone 1: Define Problem / Problem Understanding Activity 1: Specify the

business problem

The business problem for the project of deep learning model for disease detection in tea leaves is to develop an automated and technological system for detecting diseases as a preventative measure to help the farmers detect the tea leaf disease at the initial stage. This can help to minimize the tea yield loss and improve the quality of tea production.

The most common objective is to acquire high-quality images of tea leaves that are affected by different types of diseases and extract the disease areas from the background using image processing techniques, such as Non-dominated Sorting Genetic Algorithm (NSGA-II) based image clustering. Along with the training and test a deep learning model that can accurately classify the tea leaf diseases, such as Convolutional Neural Networks (CNNs)

Activity 2: Business requirements

A. **Accurate forecasting:** The system should be able to forecast the accuracy of the disease detection based on the input images and the trained model. The accuracy should be expressed as a Disease Name or percentage, and it should reflect the probability of the correct classification of the disease. The system should also provide feedback or suggestions on how to improve the accuracy, such as by collecting more data, fine-tuning the model, or changing the parameters.

B. **Real-time data acquisition:** The system should be able to acquire the data from the tea fields in real time using cameras, drones, or other devices. The data should be transmitted to the system via wireless or wired networks, and it should be stored and processed in a secure and efficient manner. The system should also be able to handle different types of data, such as images, videos or text.

C. **User-friendly interface:** The system should have a user-friendly and intuitive interface that allows the users to interact with the system easily and effectively. The interface should provide the following features . A dashboard that shows the overview of the tea fields, the status of the data acquisition, the accuracy of the disease detection, and the summary of the results. A menu that allows the users to select the options for the data acquisition, the model training, the disease detection, and the report generation.A display that shows the input images, the segmented disease areas, the classified disease labels, and the accuracy scores.A feedback mechanism that allows the users to provide comments, ratings, or corrections to the system.

D. **Report generation:** The system should be able to generate reports that summarize the results of the disease detection, such as the number of images, the number of diseases, the distribution of diseases, the accuracy of the classification, and the recommendations for the prevention or treatment of the diseases. The reports should be formatted in a clear and concise way, using tables, charts, graphs, or other visual elements. The reports should also be exportable in different formats, such as Word.

Activity 3: Literature Survey

The Grey Blight Disease Detection on Tea Leaves Using Improved Deep Convolutional Neural Network" by J.Arun Pandian , paper proposes a novel DCNN using inverted residuals and linear bottleneck layers for diagnosing grey blight disease on tea leaves, and reports superior performance over existing techniques. The "Image Analysis and Detection of Tea Leaf Disease using Deep Learning" by S. Sivaranjani , paper develops a deep CNN called LeNet to detect the tea plant diseases from leaf image set, and claims that LeNet is the perfect CNN model for improving the diagnostic measurement of tea leaves as well as other plant leaves. The third paper "Machine Learning-Based Tea Leaf Disease Detection: A Comprehensive Review" by Md. Rashedul Islam provides a systematic review of the literature on machine learning methodologies applied to diagnose tea leaf disease via image classification, and discusses the challenges and future directions of tea leaf disease detection using machine learning.

Activity 4: Social or Business Impact.

The Blood Donation Analysis project can have both social and business impacts.

Social Impact:

The social impact of this project is to provide a beneficial and innovative solution for the tea farmers and the tea industry. Tea is one of the most popular and widely consumed beverages in the world, and it has significant economic, cultural, and health value. However, tea production is threatened by various diseases that affect the quality and quantity of tea leaves. These diseases can cause severe losses to the farmers and the industry, as well as affect the consumers' satisfaction and health. Therefore, developing a deep learning model for disease detection in tea leaves can help to Enhance the productivity and profitability of the tea farmers by enabling them to identify and treat the diseases at an early stage, before they spread and damage the tea plants. Improve the quality and safety of the tea products by reducing the risk of contamination and deterioration caused by the diseases.

- Increase the competitiveness and sustainability of the tea industry by providing a reliable and efficient system for quality control and management.
- Promote the awareness and education of the tea farmers and the consumers about the types, causes, symptoms, and prevention of the tea diseases.
- Contribute to the scientific and technological advancement of the field of plant disease detection and diagnosis using deep learning methods.

Business Impact:

The business impact of this project is to create a competitive advantage and a value proposition for the tea farmers and the tea industry.

By using deep learning to detect tea leaf diseases, the project can reduce the cost and time of manual inspection and diagnosis of tea leaves, which can be labor-intensive, error-prone, and inconsistent.Which can increase the efficiency and effectiveness of disease management and control, which can prevent the spread and severity of the diseases, and reduce the use of pesticides and fungicides.

- Enhance the customer satisfaction and loyalty, by providing high-quality and safe tea products that meet the standards and expectations of the consumers.
- Generate new insights and opportunities, by using the data and the model to discover the patterns, trends, and factors that affect the tea production and quality, and to develop new products, services, or strategies.

Milestone 2: Data Collection and Visualizing and analyzing the data ML depends heavily on data, it is the most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

Activity 1: Download the dataset
There are many popular open sources for collecting the data.

In this project, we have used “Tea sickness dataset”. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

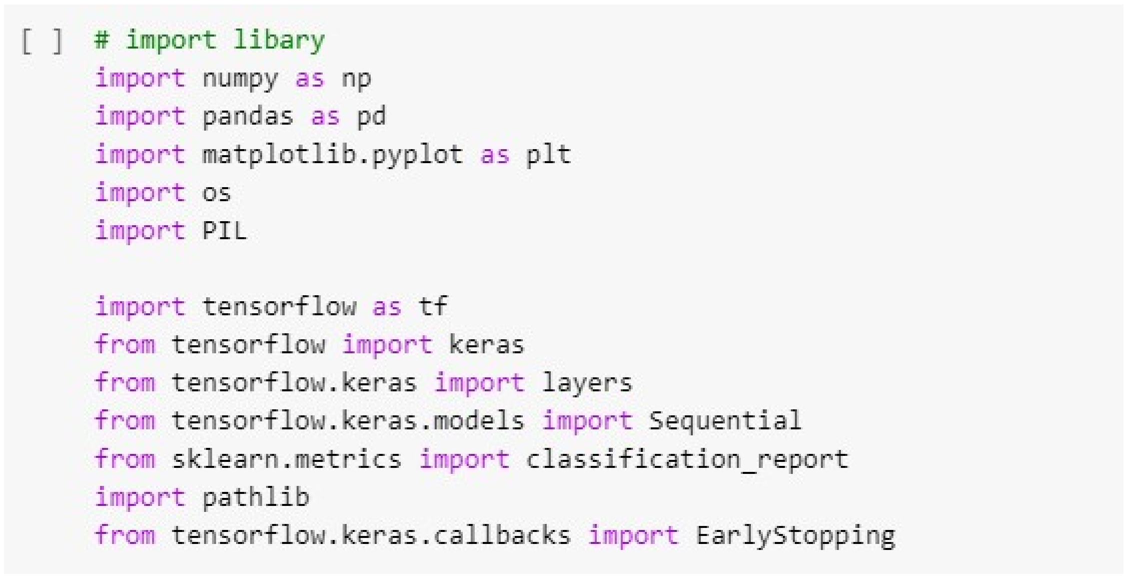
Link:https://www.kaggle.com/datasets/riazulhasanprince/tea-sickness-dataset

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

Note: There are n number of methods for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 2: Importing the libraries

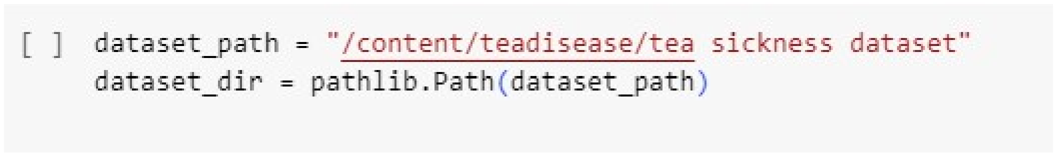
Import the necessary libraries as shown in the image. Here we have used visualization style as five thirty eight.



```
[ ]  # import libary
     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import os
     import PIL

     import tensorflow as tf
     from tensorflow import keras
     from tensorflow.keras import layers
     from tensorflow.keras.models import Sequential
     from sklearn.metrics import classification_report
     import pathlib
     from tensorflow.keras.callbacks import EarlyStopping
```

Activity 3: Read the Dataset

Our dataset format is in the form .json etc.

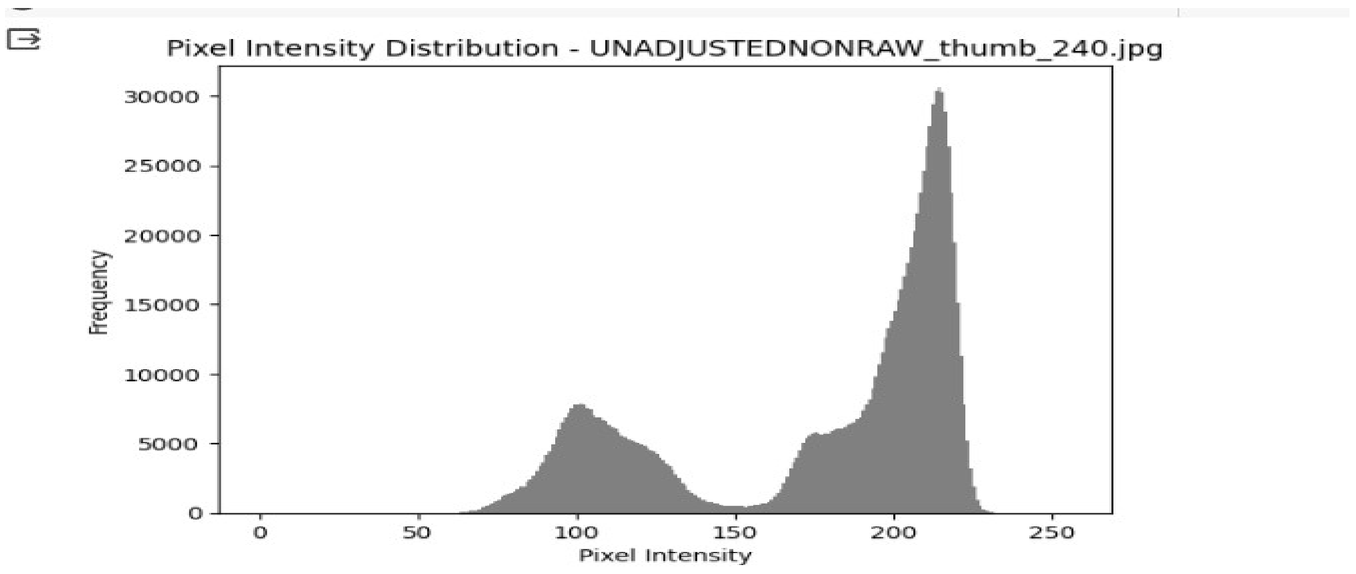


```
[ ]  dataset_path = "/content/teadisease/tea sickness dataset"
     dataset_dir = pathlib.Path(dataset_path)
```

Activity 4: Univariate analysis

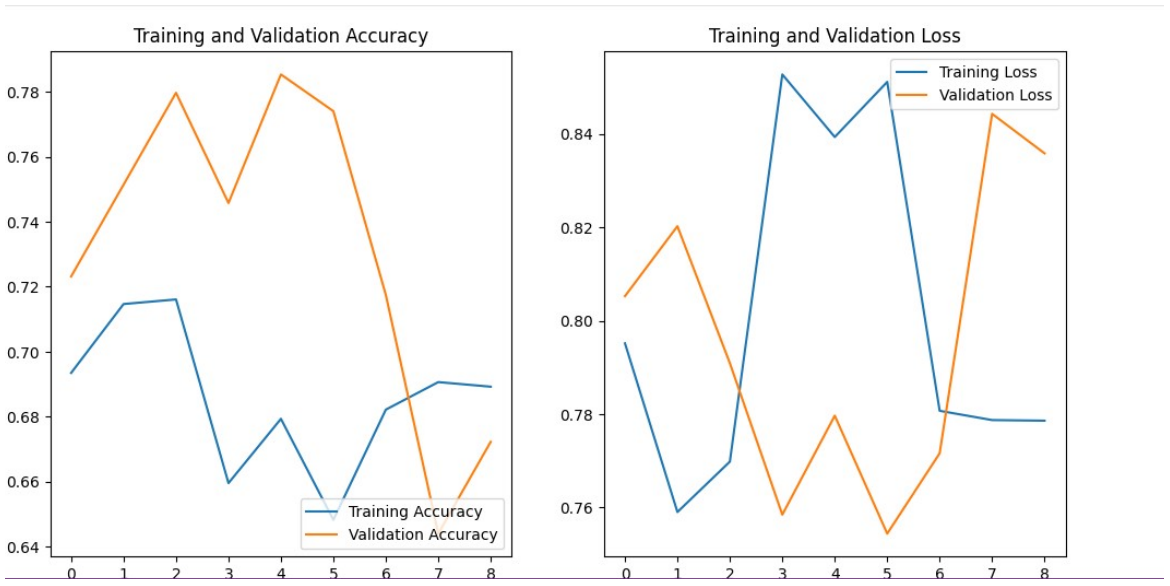
Pixel Intensity Distribution: you can create histograms to visualize the distribution of pixel intensities.
This provides insights into the overall brightness and contrast of the images.

For example, one image is provided below:



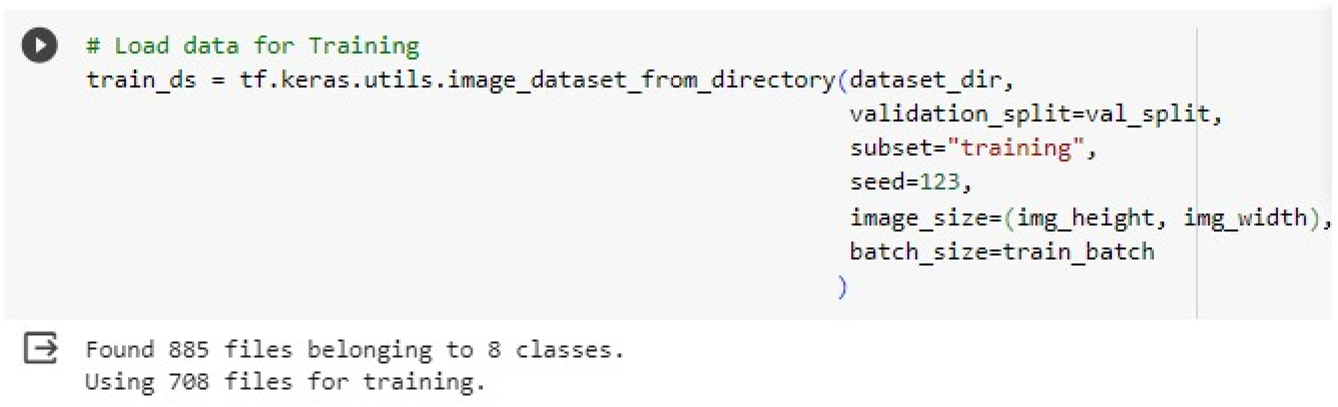
Activity 5: Bivariate analysis

To find the relation between two features we use bivariate analysis. Here we are visualizing the relationship between training and validation accuracy and also training and validation loss



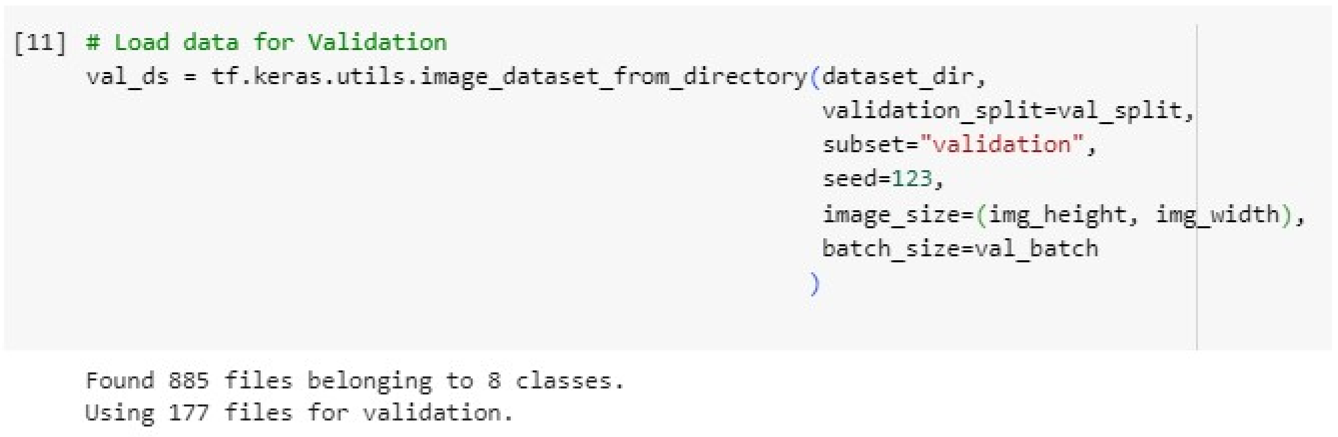
Milestone 3: Data Pre-processing and validation data splitting

This code efficiently loads training and validation datasets from a directory with minimal lines of code, taking care of data pre-processing, validation data splitting, and batching.



```
# Load data for Training
train_ds = tf.keras.utils.image_dataset_from_directory(dataset_dir,
                                                        validation_split=val_split,
                                                        subset="training",
                                                        seed=123,
                                                        image_size=(img_height, img_width),
                                                        batch_size=train_batch
                                                       )
```

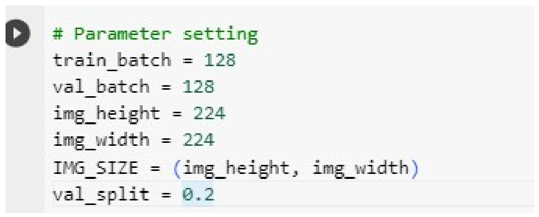
```
Found 885 files belonging to 8 classes.
Using 708 files for training.
```



```
[11] # Load data for Validation
val_ds = tf.keras.utils.image_dataset_from_directory(dataset_dir,
                                                      validation_split=val_split,
                                                      subset="validation",
                                                      seed=123,
                                                      image_size=(img_height, img_width),
                                                      batch_size=val_batch
                                                     )
```

```
Found 885 files belonging to 8 classes.
Using 177 files for validation.
```

Validation_split=val_split=0.2

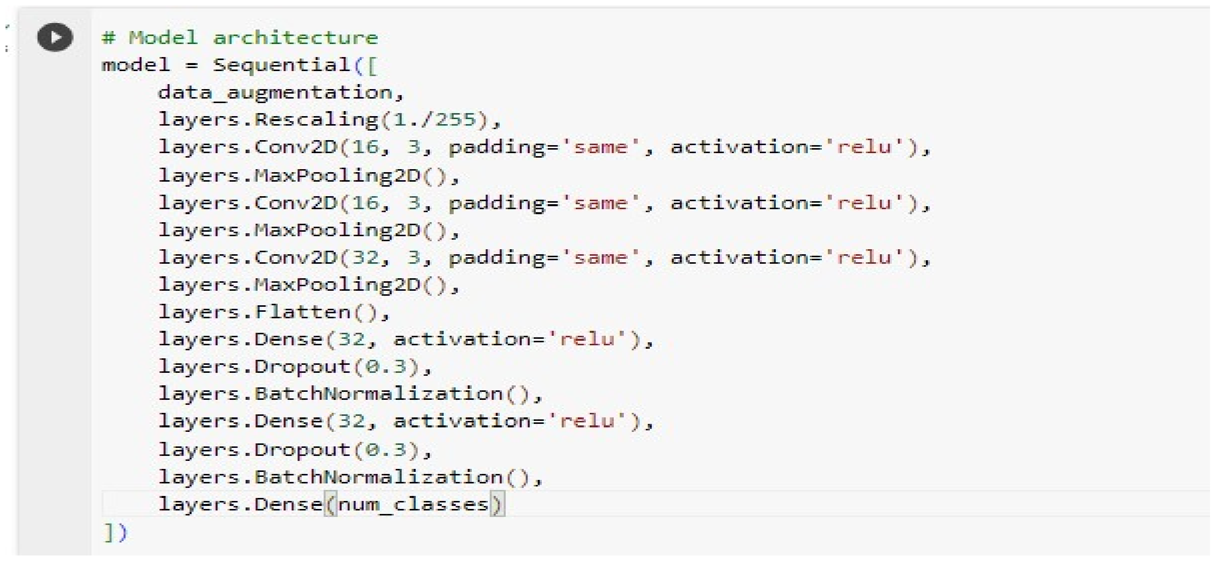


```
# Parameter setting
train_batch = 128
val_batch = 128
img_height = 224
img_width = 224
IMG_SIZE = (img_height, img_width)
val_split = 0.2
```

Milestone 4: Model Building

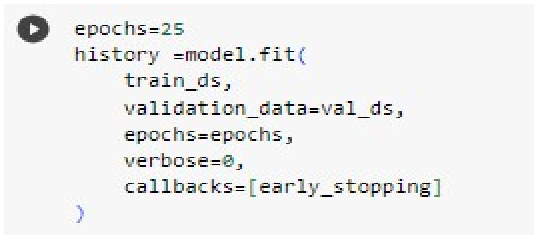
Now our data is ready to build the model. We can train our data on different algorithms. For this project we have used CNN algorithm

Activity 1: CNN model Architecture



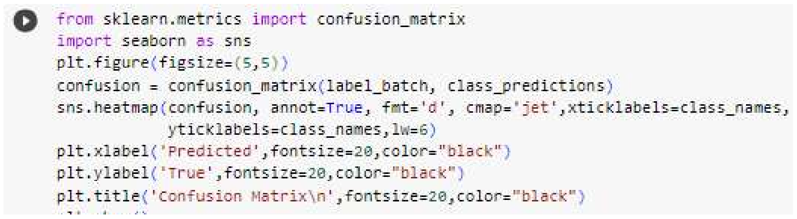
```
# Model architecture
model = Sequential([
    data_augmentation,
    layers.Rescaling(1./255),
    layers.Conv2D(16, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(16, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(32, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Flatten(),
    layers.Dense(32, activation='relu'),
    layers.Dropout(0.3),
    layers.BatchNormalization(),
    layers.Dense(32, activation='relu'),
    layers.Dropout(0.3),
    layers.BatchNormalization(),
    layers.Dense(num_classes)
])
```

CNN algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

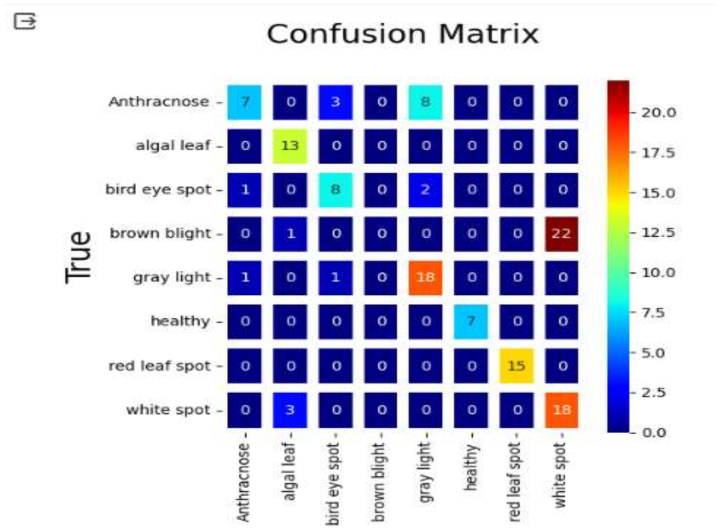


```
epochs=25
history =model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=epochs,
    verbose=0,
    callbacks=[early_stopping]
)
```

Confusion matrix:



```
from sklearn.metrics import confusion_matrix
import seaborn as sns
plt.figure(figsize=(5,5))
confusion = confusion_matrix(label_batch, class_predictions)
sns.heatmap(confusion, annot=True, fmt='d', cmap='jet',xticklabels=class_names,
            yticklabels=class_names,lw=6)
plt.xlabel('Predicted',fontsize=20,color="black")
plt.ylabel('True',fontsize=20,color="black")
plt.title('Confusion Matrix\n',fontsize=20,color="black")
```



Milestone 5: Application Building

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the user, where they have to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building serverside script

Activity1: Building Html Pages:

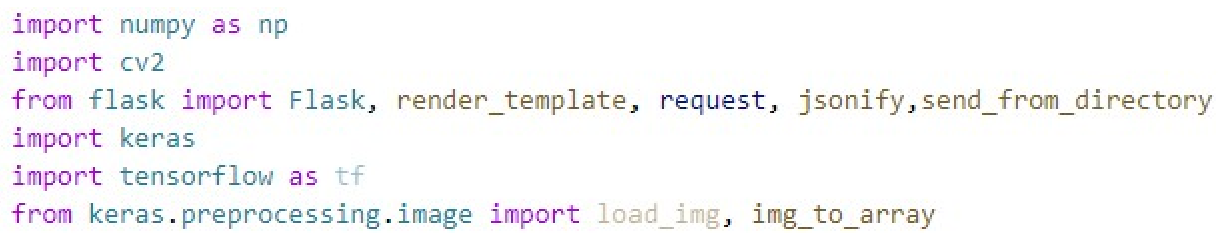
For this project create one HTML page, js file, python file, css file

- home.html
- predict.html
- Script.js
- App.py

- Style.css

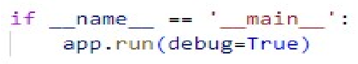
Activity 2: Build Python code:

Import the libraries



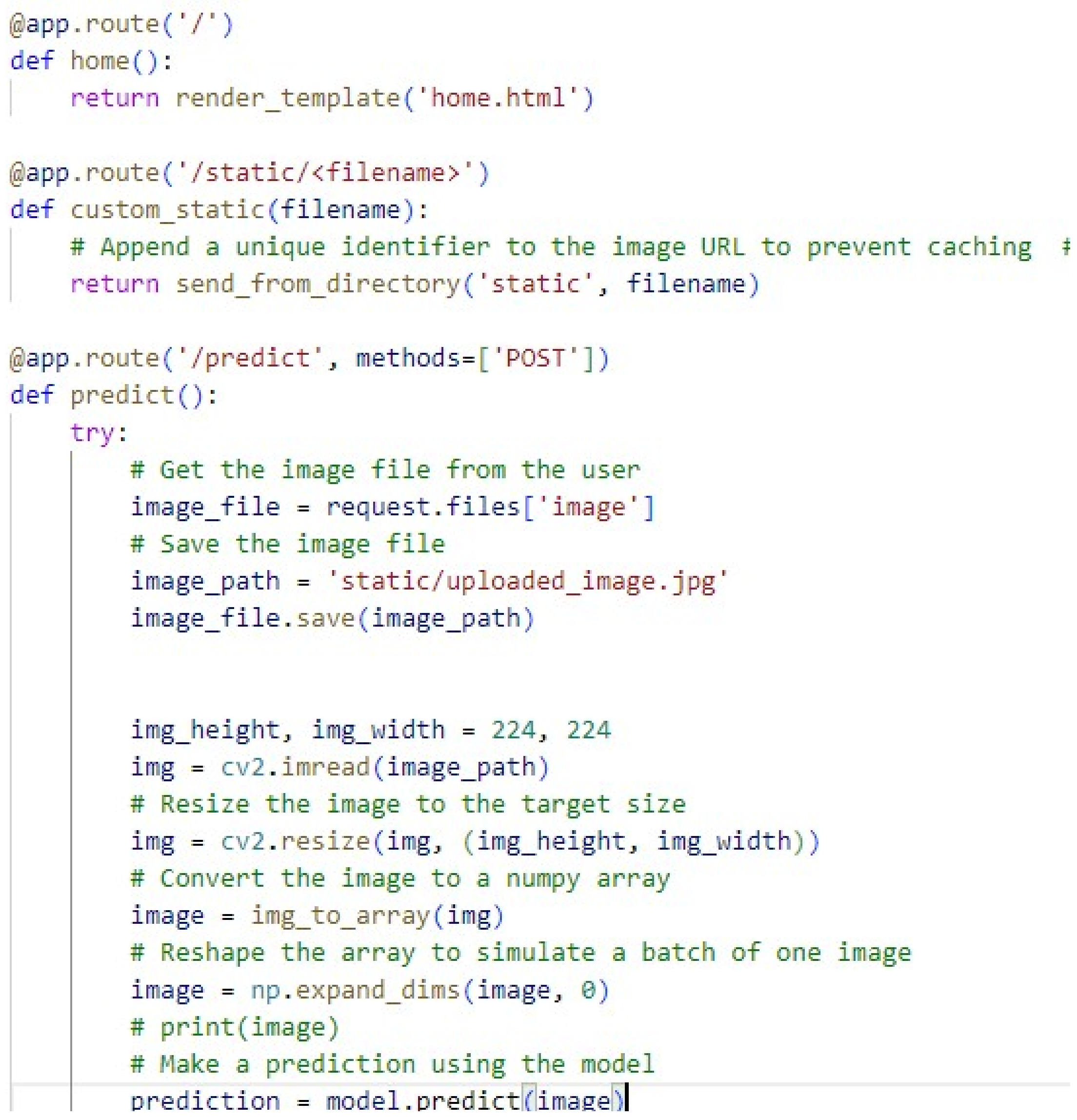
```
import numpy as np
import cv2
from flask import Flask, render_template, request, jsonify,send_from_directory
import keras
import tensorflow as tf
from keras.preprocessing.image import load_img, img_to_array
```

Load the saved model. Importing flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument.



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

Render HTML page:

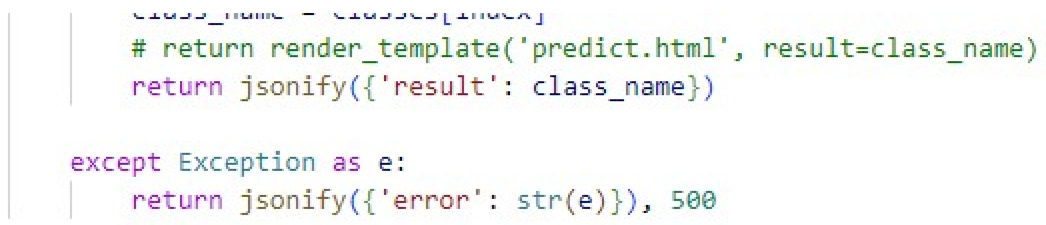


```
@app.route('/')
def home():
    return render_template('home.html')

@app.route('/static/<filename>')
def custom_static(filename):
    # Append a unique identifier to the image URL to prevent caching  #
    return send_from_directory('static', filename)

@app.route('/predict', methods=['POST'])
def predict():
    try:
        # Get the image file from the user
        image_file = request.files['image']
        # Save the image file
        image_path = 'static/uploaded_image.jpg'
        image_file.save(image_path)


        img_height, img_width = 224, 224
        img = cv2.imread(image_path)
        # Resize the image to the target size
        img = cv2.resize(img, (img_height, img_width))
        # Convert the image to a numpy array
        image = img_to_array(img)
        # Reshape the array to simulate a batch of one image
        image = np.expand_dims(image, 0)
        # print(image)
        # Make a prediction using the model
        prediction = model.predict(image)
```

Here we will be using declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with home.html function. Hence, when the home page of the web server is opened in browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

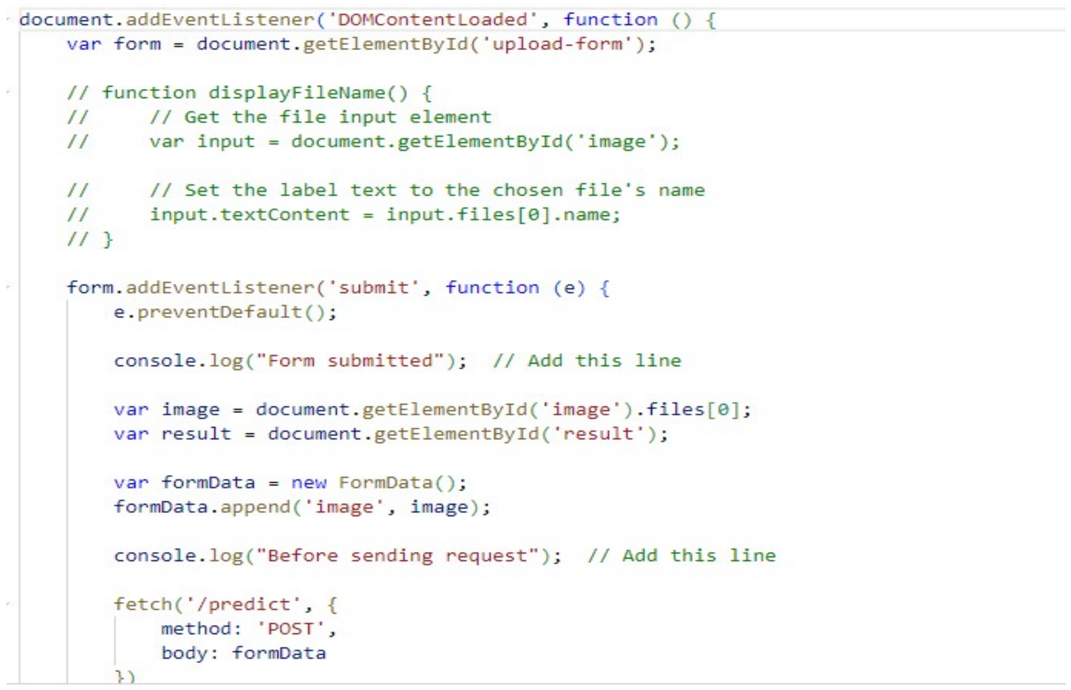
Retrieves the value from UI:



```
        class_name = classes[index]
        # return render_template('predict.html', result=class_name)
        return jsonify({'result': class_name})

    except Exception as e:
        return jsonify({'error': str(e)}), 500
```

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will rendered to the text that we have mentioned in the home.html page earlier.

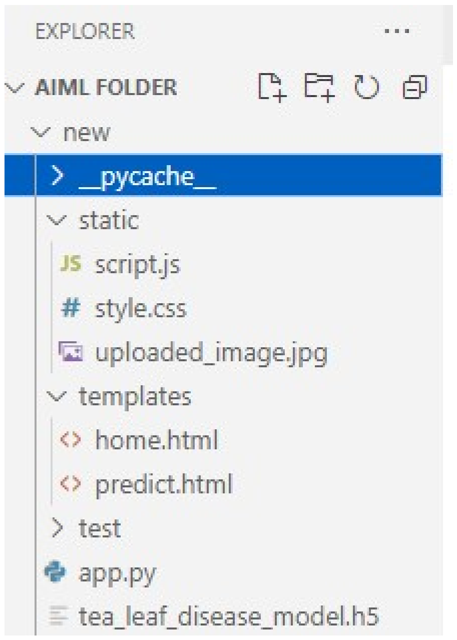
Main Function:



```
document.addEventListener('DOMContentLoaded', function () {
    var form = document.getElementById('upload-form');

    // function displayFileName() {
    //     // Get the file input element
    //     var input = document.getElementById('image');

    //     // Set the label text to the chosen file's name
    //     input.textContent = input.files[0].name;
    // }

    form.addEventListener('submit', function (e) {
        e.preventDefault();

        console.log("Form submitted");  // Add this line

        var image = document.getElementById('image').files[0];
        var result = document.getElementById('result');

        var formData = new FormData();
        formData.append('image', image);

        console.log("Before sending request");  // Add this line

        fetch('/predict', {
            method: 'POST',
            body: formData
        })
```

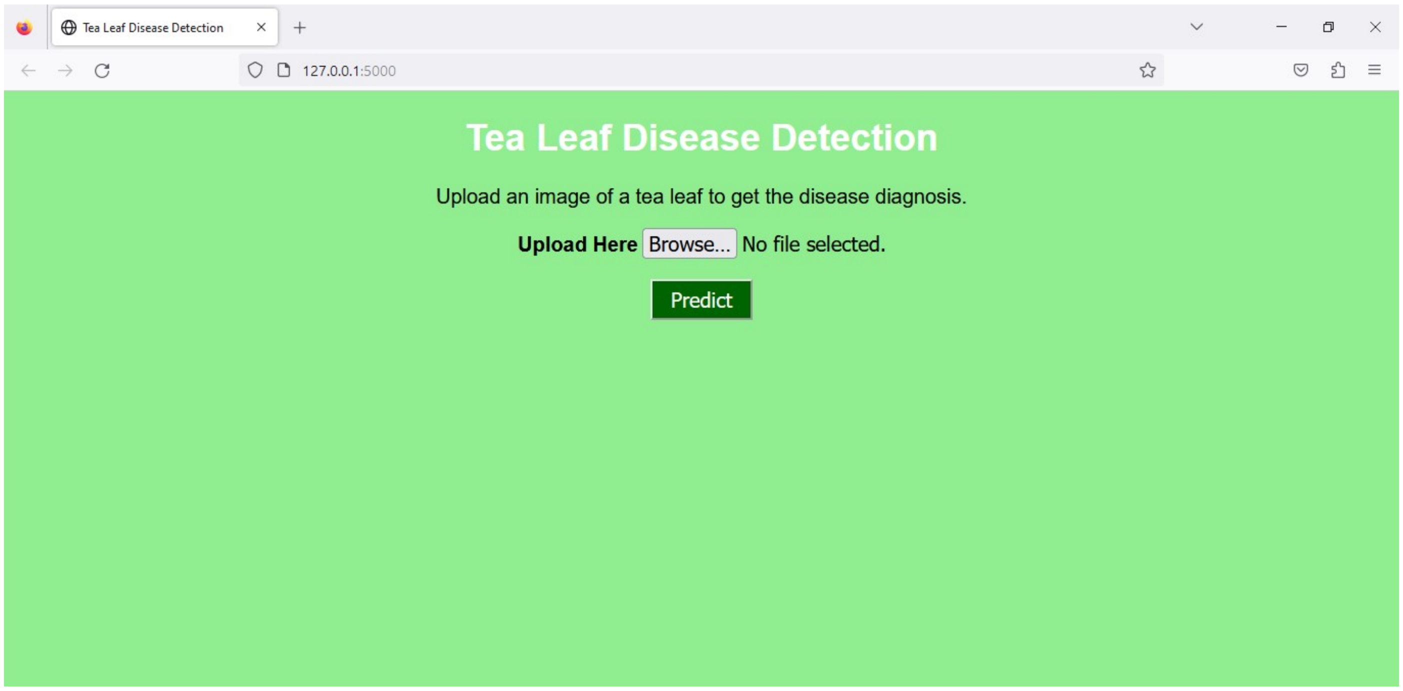
Activity 3: Run the application

Open Visual studio code and Import all the project folders.



```
EXPLORER
AIML FOLDER
  new
    __pycache__
    static
      script.js
      style.css
      uploaded_image.jpg
    templates
      home.html
      predict.html
    test
    app.py
    tea_leaf_disease_model.h5
```

When you run the app.py file and click on the server url in terminal, you will redirected to home page. The home page will looks like:

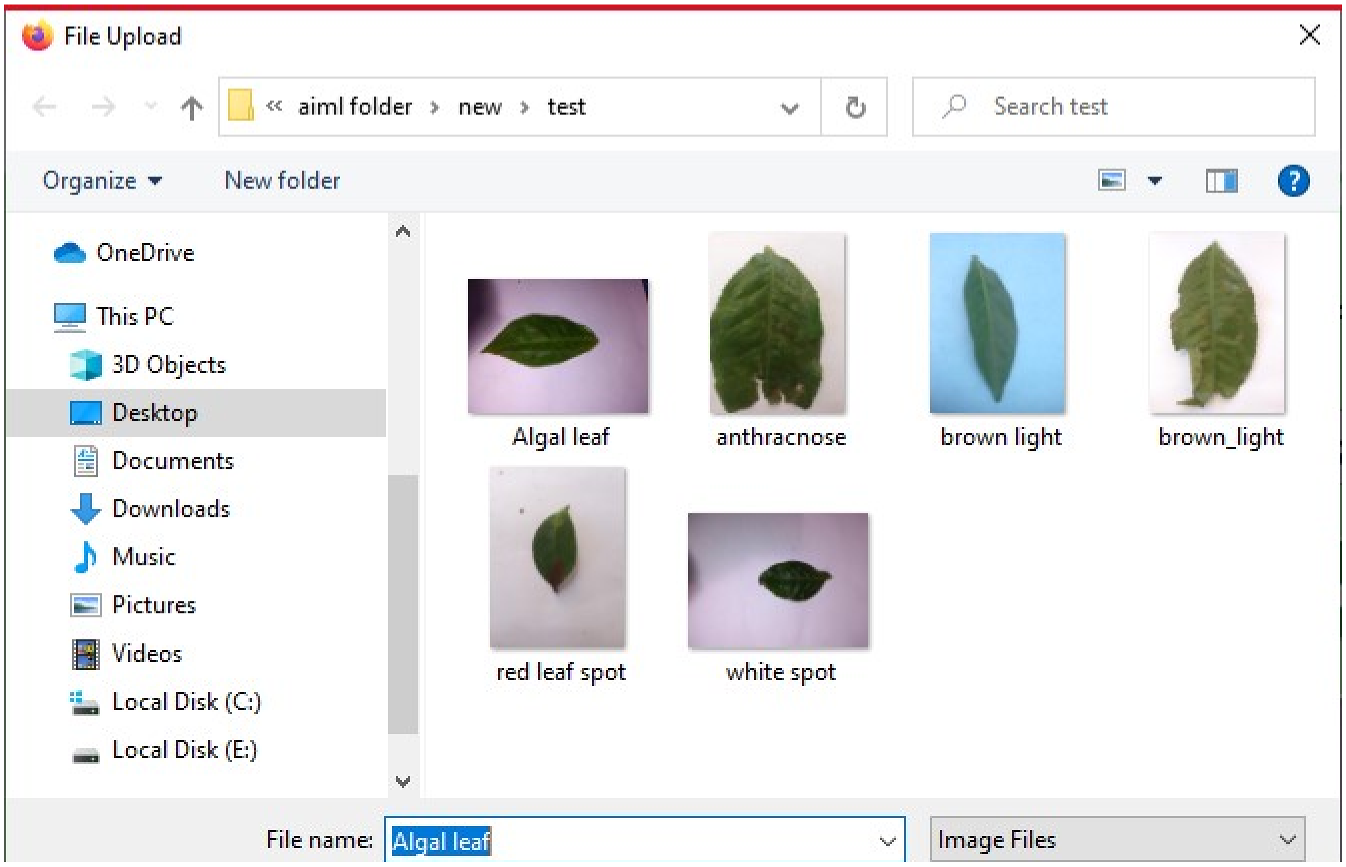


Tea Leaf Disease Detection

Upload an image of a tea leaf to get the disease diagnosis.

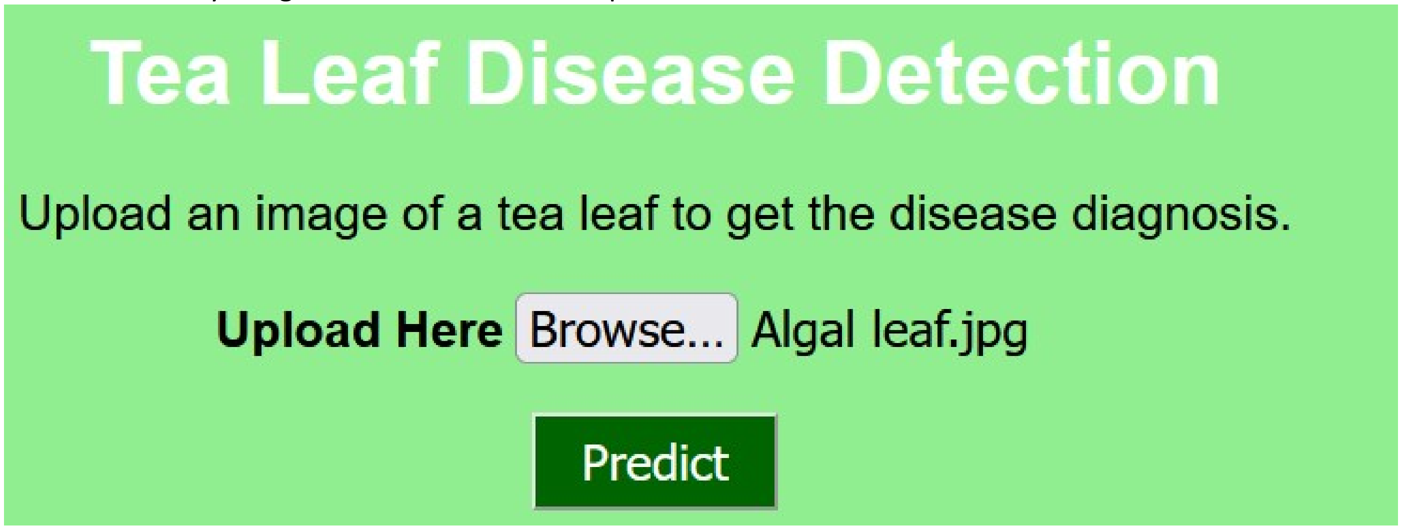
Upload Here No file selected.

If you click on Browse option it will redirect to the test page which include images:

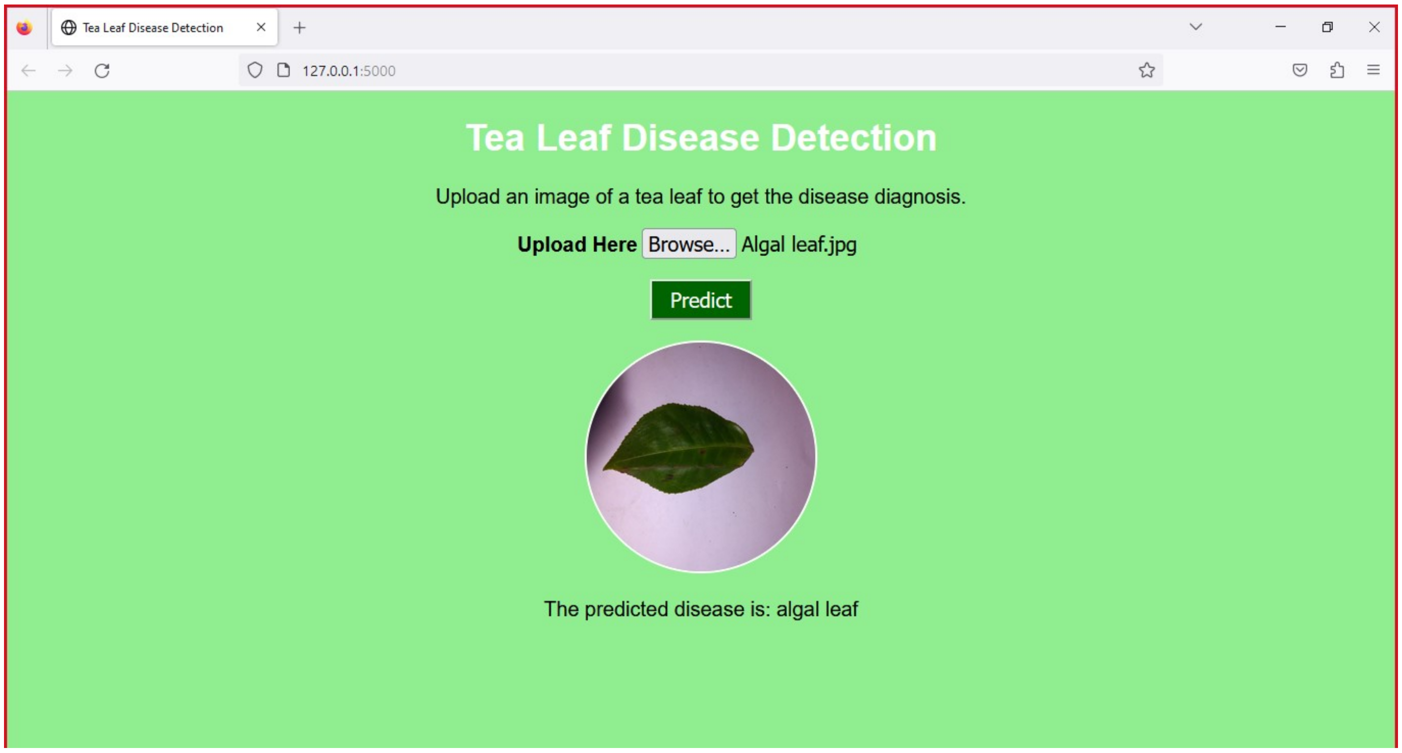


File name: Algal leaf

Then click on any image in test file and select open:


Tea Leaf Disease Detection
Upload an image of a tea leaf to get the disease diagnosis.
Upload Here
Browse...
Algal leaf.jpg
Predict

When we click on predict button it gives the output of the leaf.



Tea Leaf Disease Detection

Upload an image of a tea leaf to get the disease diagnosis.

Upload Here Browse... Algal leaf.jpg

Predict

The predicted disease is: algal leaf

Conclusion:

The Summary of this project can be simplified with deep learning , which is a powerful and promising technique for tea leaf disease detection, which can provide various benefits for the tea farmers, the tea industry, and the tea consumers. The project demonstrates the feasibility and effectiveness of using deep learning models, such as CNN to classify different types of tea diseases from leaf images. The project also shows the challenges and limitations of the current methods, such as data quality, model complexity, and generalization, and suggests some possible future directions, such as data augmentation, transfer learning, and ensemble learning. The project aims to create a novel and innovative solution for tea leaf disease detection, which can improve the tea production, quality, and safety, as well as increase the knowledge and innovation in the field of plant disease detection.