PLANT LEAF DISEASE DETECTION SYSTEM:

Name : VADDI SIVA NAGA SAI Email : [sivanagasaivaddi@gmail.com](mailto:sivanagasaivaddi@gmail.com)

Domain : AI-ML Ph no.: 7075405923

Task-2: Plant leaf disease detection system using AI algorithms.

Abstract :

Plant leaf diseases pose a significant threat to agriculture, leading to reduced crop yields and substantial economic losses. Early and accurate detection of these diseases is crucial for effective management and mitigation. This paper presents a plant leaf disease detection system leveraging advanced AI algorithms, particularly Convolutional Neural Networks (CNNs), for automated identification and classification of various leaf diseases. The system encompasses several stages: data collection, preprocessing, model selection, training, and evaluation. A comprehensive dataset comprising images of healthy and diseased leaves is compiled and augmented to enhance model robustness. The CNN model is trained and validated using this dataset, employing techniques such as dropout and regularization to mitigate overfitting. The trained model is then evaluated on an independent test set, demonstrating high accuracy and reliability in detecting and classifying plant leaf diseases. This automated detection system promises to aid farmers and agricultural professionals in timely diagnosis and intervention, ultimately contributing to improved crop health and productivity.

Steps that I followed:

Dataset :

We provide a Kaggle link to download the large data set for trained and tested of plant diseases.

<https://www.kaggle.com/datasets/rashikrahmanpritom/plant-disease-recognition-dataset>

Static and Templates :

We need to download static and template zip file in Git-hub and extract it.

The Model Training code:

That had been provided in the git-hub repository.

The App code :

import os

import tensorflow as tf

import numpy as np

from tensorflow.keras.preprocessing import image

from PIL import Image

import cv2

from keras.models import load\_model

from flask import Flask, request, render\_template

from werkzeug.utils import secure\_filename

from tensorflow.keras.preprocessing.image import load\_img, img\_to\_array

app = Flask(\_\_name\_\_)

model =load\_model('model.h5')

print('Model loaded. Check <http://127.0.0.1:5000/>')

labels = {0: 'Healthy', 1: 'Powdery', 2: 'Rust'}

def getResult(image\_path):

img = load\_img(image\_path, target\_size=(225,225))

x = img\_to\_array(img)

x = x.astype('float32') / 255.

x = np.expand\_dims(x, axis=0)

predictions = model.predict(x)[0]

return predictions

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

def index():

return render\_template('index.html')

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

def upload():

if request.method == 'POST':

f = request.files['file']

basepath = os.path.dirname(\_\_file\_\_)

file\_path = os.path.join(

basepath, 'uploads', secure\_filename(f.filename))

f.save(file\_path)

predictions=getResult(file\_path)

predicted\_label = labels[np.argmax(predictions)]

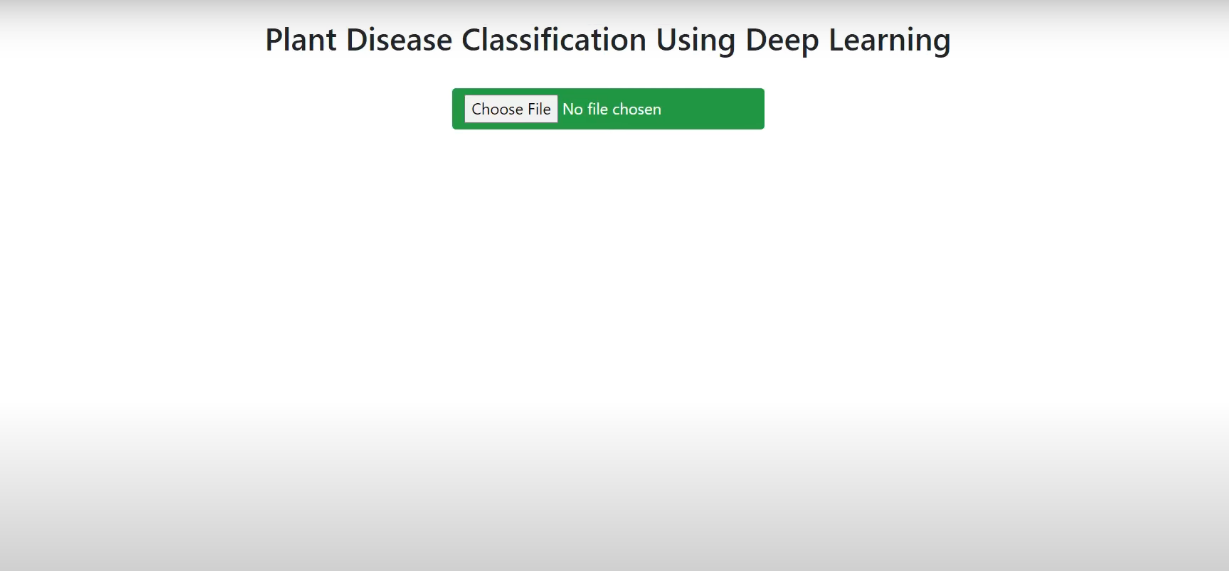
return str(predicted\_label)

return None

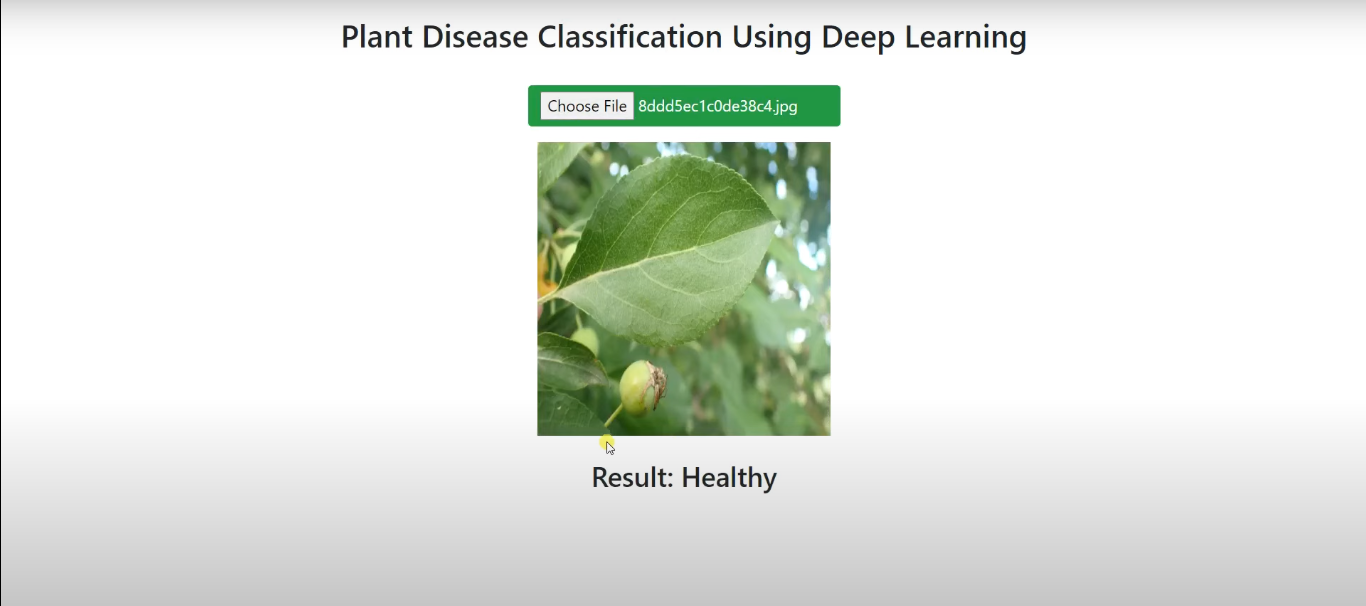
if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

The interface of app taking an input look like this :



Finally the output of the this task is look like this:



.Thankyou.