FINAL CODE

Date	16 November 2022
Team ID	PNT2022TMID24340
Project Name	Fertilizers Recommendation system for disease prediction

FINAL CODE:

```
from flask import Flask, render_template, request, Markup
import numpy as np
import pandas as pd
from utils.disease import disease_dic
from utils.fertilizer import fertilizer_dic
import requests
import config
import pickle
import io
import torch
from torchvision import transforms
from PIL import Image
from utils.model import ResNet9
import os
disease_classes = ['Apple___Apple_scab',
                'Apple___Black_rot',
                'Apple___Cedar_apple_rust',
                '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']
disease_model_path = 'models/plant_disease_model.pth'
disease_model = ResNet9(3, len(disease_classes))
disease_model.load_state_dict(torch.load(
    disease_model_path, map_location=torch.device('cpu')))
disease_model.eval()
crop_recommendation_model_path = 'models/RandomForest.pkl'
crop_recommendation_model = pickle.load(
    open(crop_recommendation_model_path, 'rb'))
def weather_fetch(city_name):
    api_key = config.weather_api_key
    base_url = "http://api.openweathermap.org/data/2.5/weather?"

    complete_url = base_url + "appid=" + api_key + "&q=" + city_name
    response = requests.get(complete_url)
    x = response.json()

    if x["cod"] != "404":
        y = x["main"]
        temperature = round((y["temp"] - 273.15), 2)
        return temperature
    else:
            return None
def predict_image(img, model=disease_model):
    transform = transforms.Compose([
        transforms.Resize(256),
        transforms.ToTensor(),
    ])
    image = Image.open(io.BytesIO(img))
    img_t = transform(image)
    img_u = torch.unsqueeze(img_t, 0)

    # Get predictions from model
    yb = model(img_u)
    # Pick index with highest probability
    _, preds = torch.max(yb, dim=1)
    prediction = disease_classes[preds[0].item()]
    # Retrieve the class label
    return prediction
app=Flask(_name_)
@ app.route('/crop-predict', methods=['POST'])
def crop_prediction():
    title = 'Harvestify - Crop Recommendation'
    if request.method == 'POST':
        N = int(request.form['nitrogen'])
        P = int(request.form['phosphorous'])
        K = int(request.form['pottasium'])
        ph = float(request.form['ph'])
        rainfall = float(request.form['rainfall'])

        # state = request.form.get("stt")
        city = request.form.get("city")

        if weather_fetch(city) != None:
            temperature, humidity = weather_fetch(city)
            data = np.array([[N, P, K, temperature, humidity, ph, rainfall]])
            my_prediction = crop_recommendation_model.predict(data)
            final_prediction = my_prediction[0]
            return render_template('crop-result.html', prediction=final_prediction, title=title)
        else:
            return render_template('try_again.html', title=title)
@ app.route('/fertilizer-predict', methods=['POST'])
def fert_recommend():
    title = 'Harvestify - Fertilizer Suggestion'
    crop_name = str(request.form['cropname'])
    N = int(request.form['nitrogen'])
    P = int(request.form['phosphorous'])
    K = int(request.form['pottasium'])
    # ph = float(request.form['ph'])
    df = pd.read_csv('Data/fertilizer.csv')
    nr = df[df['Crop'] == crop_name]['N'].iloc[0]
    pr = df[df['Crop'] == crop_name]['P'].iloc[0]
    kr = df[df['Crop'] == crop_name]['K'].iloc[0]
    n = nr - N
    p = pr - P
    k = kr - K
    temp = {abs(n): "N", abs(p): "P", abs(k): "K"}
    max_value = temp[max(temp.keys())]
    if max_value == "N":
        if n < 0:
            key = 'NHigh'
        else:
            key = "Nlow"
    elif max_value == "P":
        if p < 0:
            key = 'PHigh'
        else:
            key = "Plow"
    else:
        if k < 0:
            key = 'KHigh'
        else:
            key = "Klow"
            response = Markup(str(fertilizer_dic[key]))
            return render_template('fertilizer-result.html', recommendation=response, title=title)
@app.route('/disease-predict', methods=['GET', 'POST'])
def upload():
    if request.method=='POST':
        f=request.files['image']
        basepath=os.path.dirname(_file_)
        filepath=os.path.join(basepath,'uploads',f.filename)
        f.save(filepath)
        print('File Save')
        img=image.load_img(filepath,target_size=(128,128))
        x=image.img_to_array(img)
        print('Image to gray')
        x=np.expand_dims(x,axis=0)
        plant=request.form['plant']
        if(plant=='vegetable'):
            model=load_model("vegitable.h5")
            y=np.argmax(model.predict(x),axis=1)
            df=pd.read_excel('precautions_veg.xlsx')
        if(plant=='fruit'):
            model=load_model('fruit.h5')
            y=np.argmax(model.predict(x),axis=1)
            df=pd.read_excel('precautions_fruits.xlsx')
        return df.iloc[y[0]]['caution']
if _name_=='_main_':
    temp.run(debug=False)
```