SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

FACULTY OF ENGINEERING & TECHNOLOGY

(Formerly SRM University, Under section 3 of UGC Act, 1956)

S.R.M. NAGAR, KATTANKULATHUR –603 203, KANCHEEPURAM DISTRICT

SCHOOL OF COMPUTING

DEPARTMENT OF NETWORKING AND COMMUNICATIONS

Course Code: 18CSE305J

Course Name: Artificial Intelligence

Course Project Title: AI – ML based Agriculture recommendation system

Team Members:

RA1911003010796(Aditya singh)

RA1911003010799(D.yashaswi)

RA1911003010800(Ashutosh Mandhani)

RA1911003010804(Anish mahajan)

Title: AI – ML based Agriculture recommendation system

PROBLEM STATEMENT:

A simple ML and DL based website which recommends the best crop to grow, fertilizers to use and the diseases caught by your crops.

AIM:

To build a website which recommends the best crop to grow, fertilizers to use and the diseases caught by the crops.

DESCREPTION:

->In this project, we present a website in which the following applications are implemented; Crop recommendation, Fertilizer recommendation and Plant disease prediction, respectively.

- In the crop recommendation application, the user can provide the soil data from their side and the application will predict which crop should the user grow.
- For the fertilizer recommendation application, the user can input the soil data and the type of crop they are growing, and the application will predict what the soil lacks or has excess of and will recommend improvements.
- For the last application, that is the plant disease prediction application, the user can input an image of a diseased plant leaf, and the application will predict what disease it is and will also give a little background about the disease and suggestions to cure it.

How to use

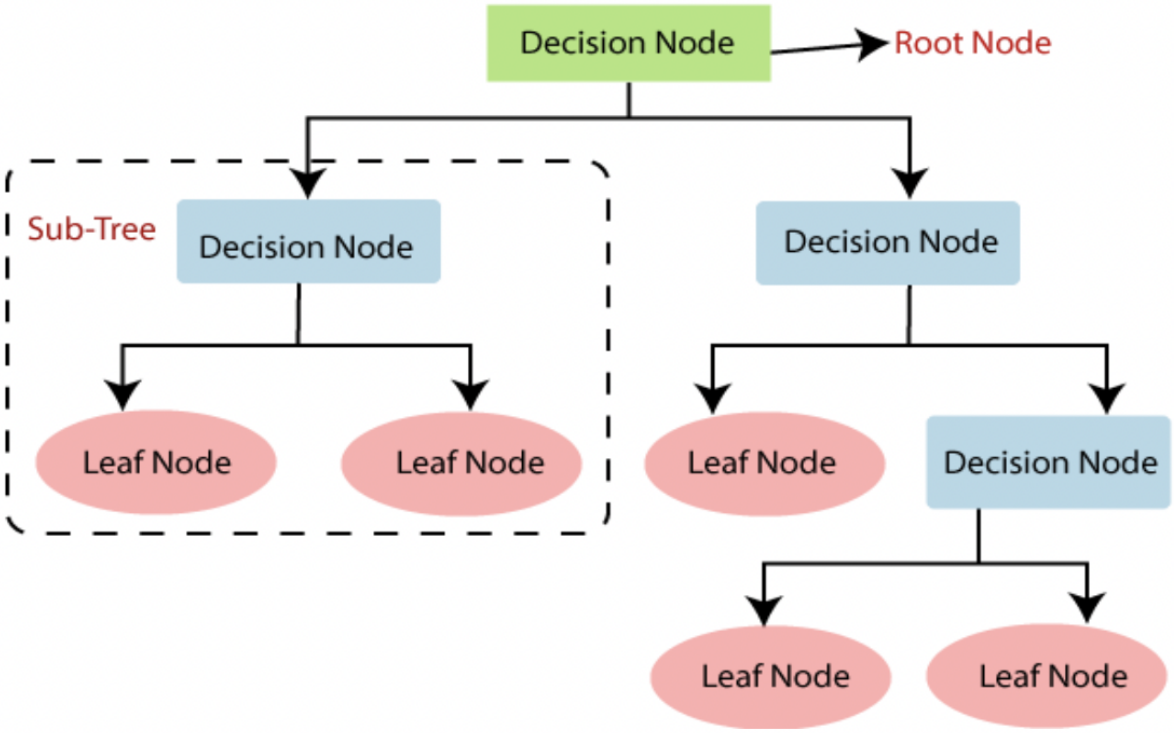
- Crop Recommendation system ==> enter the corresponding nutrient values of your soil, state and city. Note that, the N-P-K (Nitrogen-Phosphorous-Pottasium) values to be entered should be the ratio between them. Refer this website for more information. Note: When you enter the city name, make sure to enter mostly common city names. Remote cities/towns may not be available in the Weather API from where humidity, temperature data is fetched.
- Fertilizer suggestion system ==> Enter the nutrient contents of your soil and the crop you want to grow. The algorithm will tell which nutrient the soil has excess of or lacks. Accordingly, it will give suggestions for buying fertilizers.
- Disease Detection System ==> Upload an image of leaf of your plant. The algorithm will tell the crop type and whether it is diseased or healthy. If it is diseased, it will tell you the cause of the disease and suggest you how to prevent/cure the disease accordingly. Note that, for now it only supports following crops

Decision Tree

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes).

Logistic Regression

Logistic regression is basically **a supervised classification algorithm**. In a classification problem, the target variable(or output), y, can take only discrete values for a given set of features(or inputs), X. Contrary to popular belief, logistic regression IS a regression model.



Main Code :

```
# Importing essential libraries and modules

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
#
================================================================================
==============

# ------------------------LOADING THE TRAINED MODELS ----------------------------------------------

# Loading plant disease classification model

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()


# Loading crop recommendation model

crop_recommendation_model_path = 'models/RandomForest.pkl'
crop_recommendation_model = pickle.load(
    open(crop_recommendation_model_path, 'rb'))


#
==========================================================================
=========

# Custom functions for calculations


def weather_fetch(city_name):
    """
    Fetch and returns the temperature and humidity of a city
    :params: city_name
    :return: temperature, humidity
    """
    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)
        humidity = y["humidity"]
        return temperature, humidity
    else:
        return None


def predict_image(img, model=disease_model):
    """
    Transforms image to tensor and predicts disease label
    :params: image
    :return: prediction (string)
    """
```

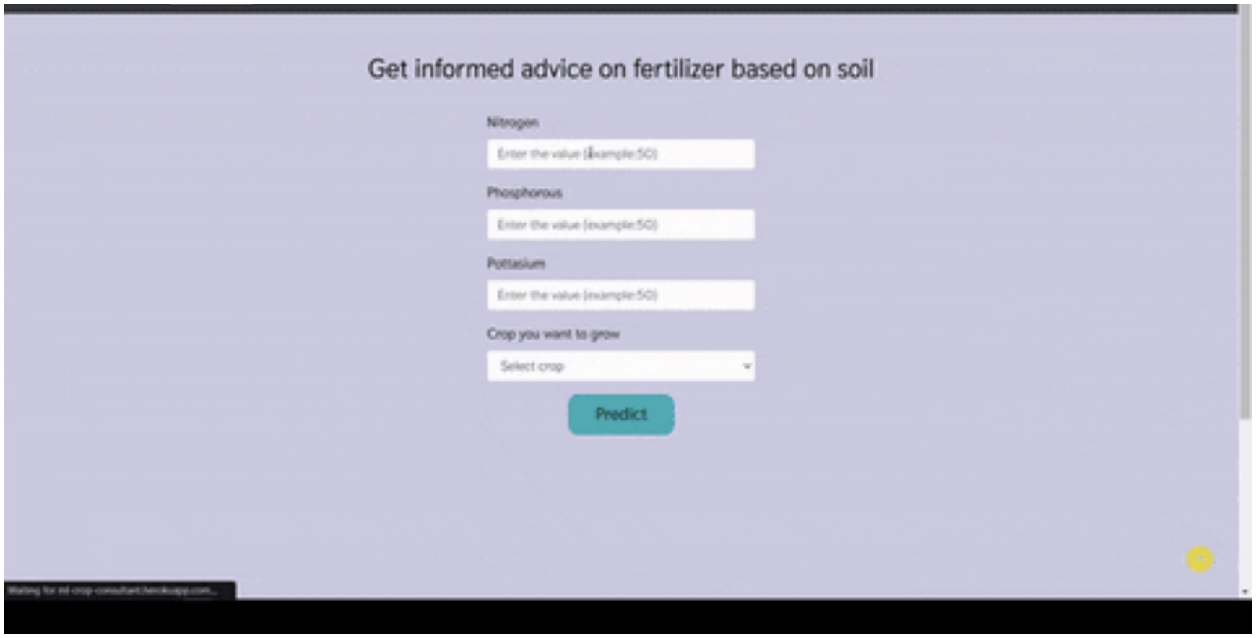
```
    transform = transforms.Compose([
        transforms.Resize(256),
        transforms.ToTensor(),
    ])
    image = Image.open(io.BytesIO(img))
    img_t = transform(image)
    img_u = torch.unsqueeze(img_t, 0)

    # 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

#
================================================================================
===============
# ---------------------------------- FLASK APP -------------------------------------------------


app = Flask(__name__)

# render home page


@ app.route('/')
def home():
    title = 'Harvestify - Home'
    return render_template('index.html', title=title)

# render crop recommendation form page


@ app.route('/crop-recommend')
def crop_recommend():
    title = 'Harvestify - Crop Recommendation'
    return render_template('crop.html', title=title)

# render fertilizer recommendation form page


@ app.route('/fertilizer')
def fertilizer_recommendation():
    title = 'Harvestify - Fertilizer Suggestion'

    return render_template('fertilizer.html', title=title)
```

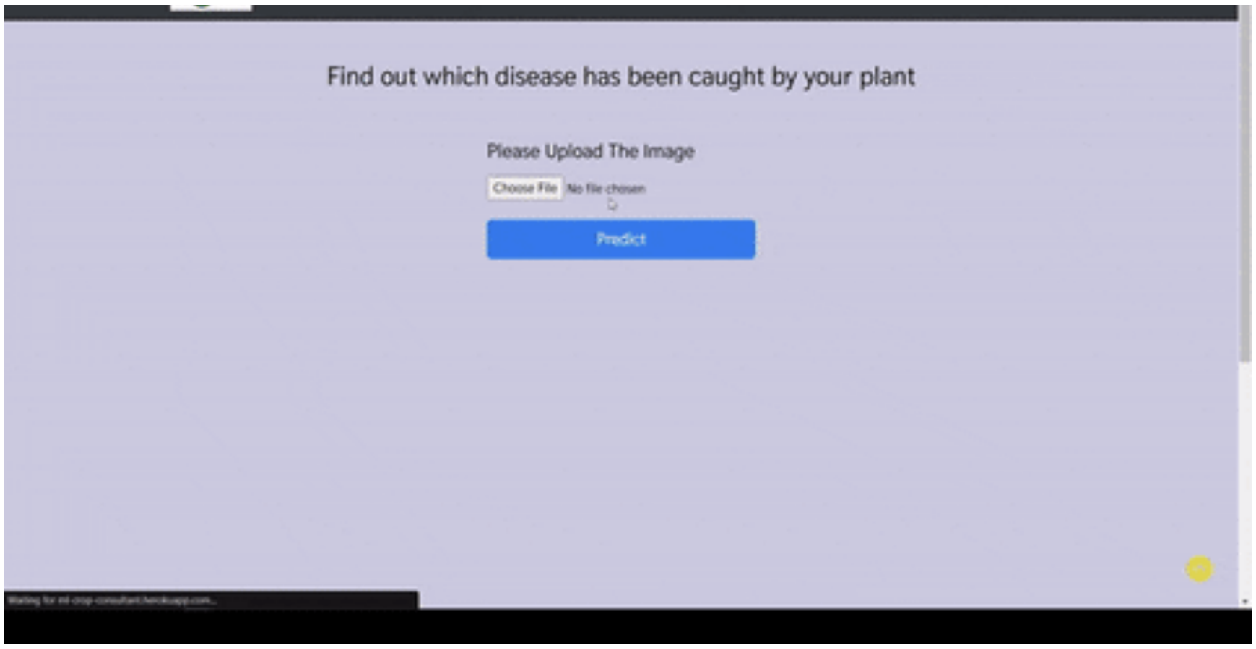
```
# render disease prediction input page




#
================================================================================
===============

# RENDER PREDICTION PAGES

# render crop recommendation result page


@ 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)

# render fertilizer recommendation result page


@ 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)

# render disease prediction result page


@app.route('/disease-predict', methods=['GET', 'POST'])
def disease_prediction():
    title = 'Harvestify - Disease Detection'

    if request.method == 'POST':
        if 'file' not in request.files:
            return redirect(request.url)
        file = request.files.get('file')
        if not file:
            return render_template('disease.html', title=title)
        try:
```

```
            img = file.read()

            prediction = predict_image(img)

            prediction = Markup(str(disease_dic[prediction]))
            return render_template('disease-result.html', prediction=prediction, title=title)
        except:
            pass
    return render_template('disease.html', title=title)


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

Screenshots/ Output :


Get informed advice on fertilizer based on soil
Nitrogen
Enter the value (example:50)
Phosphorous
Enter the value (example:50)
Pottasium
Enter the value (example:50)
Crop you want to grow
Select crop
Predict


Find out which disease has been caught by your plant
Please Upload The Image
Choose File
No file chosen
Predict