import RPi.GPIO as GPIO

import time

import cv2

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler, MultiLabelBinarizer

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report

from tensorflow import keras

from tensorflow.keras.layers import Input, Dense, Activation

from tensorflow.keras.models import Model

# Set up GPIO

GPIO.setmode(GPIO.BCM)

GPIO.setup(18, GPIO.IN) # GPIO pin connected to the soil moisture sensor

# Define a function to read soil moisture

def read\_soil\_moisture():

return GPIO.input(18) # 0 for dry, 1 for wet

# Initialize the USB camera

cap = cv2.VideoCapture(0) # 0 represents the default camera (you can change it as needed)

# Load your dataset

data = pd.read\_csv('your\_dataset.csv')

# Extract features and labels

X = data[['temperature', 'humidity', 'ph', 'water availability']]

y = data[['Stress Level\_A', 'Stress Level\_P']]

# Add soil moisture reading to features

X['soil\_moisture'] = [read\_soil\_moisture() for \_ in range(len(X))]

# Encode categorical variables like 'season' and 'label'

X = pd.get\_dummies(X, columns=['season', 'label'], drop\_first=True)

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Standardize features

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Binarize labels (convert them to multi-label format)

mlb = MultiLabelBinarizer()

y\_train = mlb.fit\_transform(y\_train)

y\_test = mlb.transform(y\_test)

# Define the model

input\_layer = Input(shape=(X\_train.shape[1],))

hidden\_layer = Dense(64, activation='relu')(input\_layer)

output\_layer = Dense(2, activation='sigmoid')(hidden\_layer) # 2 output neurons for two labels

model = Model(inputs=input\_layer, outputs=output\_layer)

# Compile the model

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

# Train the model

model.fit(X\_train, y\_train, epochs=20, batch\_size=32, validation\_split=0.2)

# Set up GPIO for LED control

GPIO.setup(17, GPIO.OUT) # GPIO pin connected to the LED

# Define a function to control the LED based on soil moisture

def control\_led():

moisture = read\_soil\_moisture()

if moisture == 0: # If soil is dry

GPIO.output(17, GPIO.HIGH) # Turn on the LED

else:

GPIO.output(17, GPIO.LOW) # Turn off the LED

# Define a function to analyze plant growth based on captured image

def analyze\_plant\_growth(image):

# Implement your plant growth analysis here

# You can use image processing techniques like color analysis, size measurement, etc.

# Update the features accordingly

growth\_features = [0.75, 0.82] # Example growth features (replace with actual measurements)

return growth\_features

# Example integration in your main loop

while True:

control\_led() # Control the LED based on soil moisture

ret, image = cap.read() # Capture an image

if ret:

growth\_features = analyze\_plant\_growth(image) # Analyze plant growth

features = [temperature, humidity, ph, water\_availability, read\_soil\_moisture()]

features.extend(growth\_features)

features.extend(model.predict(np.array([features]))[0]) # Add classifier predictions

print(features) # Do something with the features (e.g., log or control other devices)

time.sleep(1) # Sleep for 1 second before repeating