BE PROJECT CODES

Code for data visualization  
Correlation Matrices and heatmap

import numpy as np

import matplotlib

import matplotlib.pyplot as plt

matplotlib.style.use('ggplot')

import pandas as pd

import seaborn as sns

prevsoil = [67, 70, 72.6, 65.6, 67, 68.6, 66.6, 69.6, 66, 65.6, 63, 65, 64.6, 65.3, 58.6, 65,

64, 64, 67, 59.6, 61.6, 54.6, 62, 60, 64.3, 64, 63, 60.66, 65.3, 61, 64, 59, 61, 65, 67]

water = [120, 70, 30, 0, 75, 55, 100, 65, 25, 80, 50, 90, 50, 50, 50, 100, 45,

55, 55, 40, 90, 80, 120, 75, 90, 55, 60, 60, 90, 45, 90, 50, 100, 90, 50]

currtemp = [30, 25, 25, 25, 26, 27, 28, 26, 26, 25, 26, 27, 26, 27, 28, 28,

27, 26, 26, 28, 29, 31, 28, 28, 29, 29, 29, 28, 27, 28, 29, 28, 28, 29, 29]

currhumid = [26, 72, 63, 62, 62, 56, 45, 69, 62, 65, 59, 40, 57, 57, 45, 51,

57, 69, 64, 54, 49, 45, 72, 72, 63, 68, 72, 65, 71, 71, 70, 80, 74, 66, 63]

z = np.corrcoef(prevsoil,currtemp)

x = np.corrcoef(currhumid,prevsoil)

y = np.corrcoef(currtemp,currhumid)

w = np.corrcoef(currtemp,water)

m= np.corrcoef(water,currhumid)

n= np.corrcoef(prevsoil,water)

print(" Temperature and soil moisture correlation \n",z)

print(" humidity and soil moisture correlation \n",x)

print(" Temperature and humidity correlation \n",y)

print(" Temperature and water correlation \n",w)

print(" Humidity and water correlation \n",m)

print(" Soil Moisture and water correlation \n",n)

data= pd.read\_csv ('data\_base.csv')

data.drop(['constant'], axis=1, inplace=True)

corr= data.corr()

fig = plt.figure()

ax = sns.heatmap(

corr,

vmin=-1, vmax=1, center=0,

cmap=sns.diverging\_palette(10, 10, as\_cmap=True),

square=True

)

ax.set\_xticklabels(

ax.get\_xticklabels(),

rotation=15,

horizontalalignment='right'

)

plt.show()

3D Plots

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

import pandas as pd

from pandas import DataFrame

from sklearn import linear\_model

import statsmodels.api as sm

import seaborn as sns

prevsoil = [67, 70, 72.6, 65.6, 67, 68.6, 66.6, 69.6, 66, 65.6, 63, 65, 64.6, 65.3, 58.6, 65,

64, 64, 67, 59.6, 61.6, 54.6, 62, 60, 64.3, 64, 63, 60.66, 65.3, 61, 64, 59, 61, 65, 67]

water = [120, 70, 30, 0, 75, 55, 100, 65, 25, 80, 50, 90, 50, 50, 50, 100, 45,

55, 55, 40, 90, 80, 120, 75, 90, 55, 60, 60, 90, 45, 90, 50, 100, 90, 50]

currtemp = [30, 25, 25, 25, 26, 27, 28, 26, 26, 25, 26, 27, 26, 27, 28, 28,

27, 26, 26, 28, 29, 31, 28, 28, 29, 29, 29, 28, 27, 28, 29, 28, 28, 29, 29]

currhumid = [26, 72, 63, 62, 62, 56, 45, 69, 62, 65, 59, 40, 57, 57, 45, 51,

57, 69, 64, 54, 49, 45, 72, 72, 63, 68, 72, 65, 71, 71, 70, 80, 74, 66, 63]

plt.scatter(prevsoil,water,c='g',marker='o')

plt.scatter(currtemp,water,c='r',marker='o')

plt.scatter(currhumid,water,c='b',marker='o')

plt.show()

fig = plt.figure()

ax = fig.add\_subplot(311, projection='3d')

ax.scatter(prevsoil, currtemp, water, c='r', marker='o')

ax.set\_xlabel('Soil Moisture')

ax.set\_ylabel('Temperature')

ax.set\_zlabel('water amount output (ml)')

ax = fig.add\_subplot(312, projection='3d')

ax.scatter(currhumid, currtemp, water, c='r', marker='o')

ax.set\_xlabel('Humidity')

ax.set\_ylabel('Temperature')

ax.set\_zlabel('water amount output (ml)')

ax = fig.add\_subplot(313, projection='3d')

ax.scatter(prevsoil, currhumid, water, c='r', marker='o')

ax.set\_xlabel('Soil Moisture')

ax.set\_ylabel('Humidty')

ax.set\_zlabel('water amount output (ml)')

plt.show()

m1 = plt.hist(prevsoil, facecolor='red')

m2 = plt.hist(currhumid, facecolor='green')

m3 = plt.hist(currtemp, facecolor='yellow')

m3 = plt.hist(currtemp, facecolor='yellow')

plt.show()

Code for Linear Regression

from sklearn.metrics import mean\_squared\_error

from pandas import DataFrame

import matplotlib.pyplot as plt

from math import sqrt

from sklearn import neighbors

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, cross\_validate, cross\_val\_predict

import pandas as pd

from sklearn import linear\_model

import statsmodels.api as sm

df = pd.read\_csv('data\_base.csv')

soilval = df['soil\_moisture'].values.tolist()

tempval = df['temperature'].values.tolist()

humval = df['humidity'].values.tolist()

waterval = df['water'].values.tolist()

const = df['constant'].values.tolist()

# normalization

smin = min(soilval)

smax = max(soilval)

k= 0

for i in soilval:

soilval[k] = (i - smin)/(smax-smin)

k = k+1

tmin = min(tempval)

tmax = max(tempval)

k= 0

for i in tempval:

tempval[k] = (i - tmin)/(tmax-tmin)

k = k+1

hmin = min(humval)

hmax = max(humval)

k= 0

for i in humval:

humval[k] = (i - hmin)/(hmax-hmin)

k = k+1

wmin = min(waterval)

wmax = max(waterval)

k= 0

for i in waterval:

waterval[k] = (i - wmin)/(wmax-wmin)

k = k+1

d = {'const':const,'soilval':soilval,'tempval':tempval,'humval':humval,'waterval':waterval}

df\_new = pd.DataFrame(d)

# # Plotting normalized values

plt.scatter(soilval,tempval,c='g',marker='o')

plt.title('Scatter plot ')

plt.xlabel('Soil moisture')

plt.ylabel('Temperature')

plt.show()

plt.scatter(soilval,humval,c='y',marker='o')

plt.title('Scatter plot ')

plt.xlabel('Soil Moisture')

plt.ylabel('Humidity')

plt.show()

plt.scatter(soilval,waterval,c='c',marker='o')

plt.title('Scatter plot ')

plt.xlabel('Soil Moisture')

plt.ylabel('Water level')

plt.show()

plt.scatter(tempval,waterval,c='r',marker='o')

plt.title('Scatter plot ')

plt.xlabel('Temperature')

plt.ylabel('Water Level')

plt.show()

plt.scatter(humval,waterval,c='b',marker='o')

plt.title('Scatter plot ')

plt.xlabel('Humidity')

plt.ylabel('water level')

plt.show()

train, test = train\_test\_split(df\_new, test\_size=0.4)

# only temp

x\_train = train[['tempval','const']]

y\_train = train['waterval']

x\_test = test[['tempval','const']]

y\_test = test['waterval']

#all Data

X = df\_new[['tempval','const']]

y = df\_new['waterval']

# only humidity

x\_train = train[['humval','const']]

y\_train = train['waterval']

x\_test = test[['humval','const']]

y\_test = test['waterval']

#all Data

X = df\_new[['humval','const']]

y = df\_new['waterval']

# only soil moisture

x\_train = train[['soilval','const']]

y\_train = train['waterval']

x\_test = test[['soilval','const']]

y\_test = test['waterval']

#all Data

X = df\_new[['soilval','const']]

y = df\_new['waterval']

# only temp and humidity

x\_train = train[['tempval','humval','const']]

y\_train = train['waterval']

x\_test = test[['tempval','humval','const']]

y\_test = test['waterval']

#all Data

X = df\_new[['tempval','humval','const']]

y = df\_new['waterval']

# only temp and soilmoisture

x\_train = train[['tempval','soilval','const']]

y\_train = train['waterval']

x\_test = test[['tempval','soilval','const']]

y\_test = test['waterval']

#all Data

X = df\_new[['tempval','soilval','const']]

y = df\_new['waterval']

# only humidity and soilmoisture

x\_train = train[['humval','soilval','const']]

y\_train = train['waterval']

x\_test = test[['humval','soilval','const']]

y\_test = test['waterval']

#all Data

X = df\_new[['humval','soilval','const']]

y = df\_new['waterval']

# only temp and soilmoisture and humval

x\_train = train[['tempval','soilval','const','humval']]

y\_train = train['waterval']

x\_test = test[['tempval','soilval','const','humval']]

y\_test = test['waterval']

#all Data

X = df\_new[['tempval','soilval','const','humval']]

y = df\_new['waterval']

# # only temp and soilmoisture and humval without const

x\_train = train[['tempval','soilval','humval']]

y\_train = train['waterval']

x\_test = test[['tempval','soilval','humval']]

y\_test = test['waterval']

#all Data

X = df\_new[['tempval','soilval','humval']]

y = df\_new['waterval']

regr = linear\_model.LinearRegression()

regr.fit(x\_train, y\_train)

y\_pred = regr.predict(x\_test)

print("The output waterlevel using linear regression is=",y\_pred[-1]\*100,"ml")

error = sqrt(mean\_squared\_error(y\_test, y\_pred)) # calculate rmse

print('RMSE value for linear regression is =', error\*100)

model = sm.OLS(y,X).fit()

p= model.summary()

print(p)

Code for Support Vector Regression

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, cross\_validate, cross\_val\_predict

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

from math import sqrt

# Importing the dataset

df = pd.read\_csv('data\_base.csv')

# df.drop(['sr\_no'], axis=1, inplace=True)

# df = pd.get\_dummies(df)

train, test = train\_test\_split(df, test\_size=0.3)

x\_train = train.drop('water', axis=1)

y\_train = train['water']

x\_test = test.drop('water', axis=1)

y\_test = test['water']

#all Data

X = df.drop('water', axis = 1)

y = df['water']

plt.scatter(df['soil\_moisture'], df['temperature'])

plt.show()

from sklearn.svm import SVR

regressor1 = SVR(kernel='rbf', gamma='auto', epsilon=0.02)

regressor2 = SVR(kernel='linear', gamma='auto')

regressor3 = SVR(kernel='poly', gamma='auto', degree=2, epsilon=0.3,

coef0=1)

regressor1.fit(x\_train, y\_train)

regressor2.fit(x\_train, y\_train)

regressor3.fit(x\_train, y\_train)

# Predicting a new result

y\_rbf = regressor1.predict(x\_test)

y\_lin = regressor2.predict(x\_test)

y\_poly = regressor3.predict(x\_test)

print("The predicted output waterlevel using rbf kernel is=",y\_rbf[-1], "ml")

print("The predicted output waterlevel using linear kernel is=",y\_lin[-1], "ml")

print("The predicted output waterlevel using polynomial kernel is=",y\_poly[-1], "ml")

error = sqrt(mean\_squared\_error(y\_test, y\_rbf)) # calculate rmse

print('RMSE value for rbf kernel', error)

error1 = sqrt(mean\_squared\_error(y\_test, y\_lin)) # calculate rmse

print('RMSE value for linear kernel', error1)

error2 = sqrt(mean\_squared\_error(y\_test, y\_poly)) # calculate rmse

print('RMSE value for polynomial kernel', error2)

from sklearn.metrics import r2\_score

print(r2\_score(y\_test,y\_rbf))

print(r2\_score(y\_test,y\_lin))

print(r2\_score(y\_test,y\_poly))

#################################################

# Cross Validation

# #Simple 3 way crossvalidation

# scores = cross\_val\_score(regressor1, X, y, cv=3)

# print("Cross validation scores for rbf = ", scores)

# scores = cross\_val\_score(regressor2, X, y, cv=3)

# print("Cross validation scores for Linear kernel = ", scores)

# scores = cross\_val\_score(regressor3, X, y, cv=3)

# print("Cross validation scores for polynomial kernel = ", scores)

# #crossval predict

# y\_coss\_predic = cross\_val\_predict(regressor1, X, y, cv=3)

# print("Cross validation prediction of water level for rbf", y\_coss\_predic[-1], "ml")

# y\_coss\_predic = cross\_val\_predict(regressor2, X, y, cv=3)

# print("Cross validation prediction of water level for linear kernel", y\_coss\_predic[-1], "ml")

# y\_coss\_predic = cross\_val\_predict(regressor3, X, y, cv=3)

# print("Cross validation prediction of water level for polynomial kernel", y\_coss\_predic[-1], "ml")

# #cross validation plotting

# fig, ax = plt.subplots()

# ax.scatter(y, y\_coss\_predic, edgecolors=(0, 0, 0))

# ax.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=4)

# ax.set\_title('Cross Validation in SVR')

# ax.set\_xlabel('Measured value of y')

# ax.set\_ylabel('Predicted value of y')

# plt.show()

Code for KNN Regression

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

from math import sqrt

from sklearn import neighbors

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

import pandas as pd

df = pd.read\_csv('data\_base.csv')

df.head()

train, test = train\_test\_split(df, test\_size=0.4)

x\_train = train.drop('water', axis=1)

y\_train = train['water']

x\_test = test.drop('water', axis=1)

y\_test = test['water']

# rmse\_val = [] # to store rmse values for different k

# for K in range(8):

# K = K+1

# model = neighbors.KNeighborsRegressor(n\_neighbors=K)

# model.fit(x\_train, y\_train) # fit the model

# pred = model.predict(x\_test) # make prediction on test set

# error = sqrt(mean\_squared\_error(y\_test, pred)) # calculate rmse

# rmse\_val.append(error) # store rmse values

# print('RMSE value for k= ', K, 'is:', error)

# # plotting the rmse values against k values

# print("The predicted water level is ",pred[-1], "ml")

# curve = pd.DataFrame(rmse\_val) # elbow curve

# curve.plot()

# plt.show()

from sklearn.model\_selection import GridSearchCV

params = {'n\_neighbors':[2,3,4,5,6,7,8,9]}

knn = neighbors.KNeighborsRegressor()

model = GridSearchCV(knn, params, iid = 'warn' ,cv=5)

model.fit(x\_train,y\_train)

model.best\_params\_

print("The best value of K is= \n")

print(model.best\_params\_)

y\_pred = model.predict(x\_test)

print("The predicted water level is ",y\_pred[-1], "ml")

error = sqrt(mean\_squared\_error(y\_test, y\_pred))

print("The RMSE value for the KNN regressor is", error)

Main script and Backend code

from backend import \*

import numpy as np

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

from math import sqrt

from sklearn import neighbors

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, cross\_validate, cross\_val\_predict

import pandas as pd

from sklearn import linear\_model

import statsmodels.api as sm

#Data Reading from CSV

df = getdbdata()

X = df[['constant','soil\_moisture','temperature','humidity']]

Y = df['water']

soilval = df['soil\_moisture'].values.tolist()

tempval = df['temperature'].values.tolist()

humval = df['humidity'].values.tolist()

waterval = df['water'].values.tolist()

const = df['constant'].values.tolist()

# normalization

smin = min(soilval)

smax = max(soilval)

k= 0

for i in soilval:

soilval[k] = (i - smin)/(smax-smin)

k = k+1

tmin = min(tempval)

tmax = max(tempval)

k= 0

for i in tempval:

tempval[k] = (i - tmin)/(tmax-tmin)

k = k+1

hmin = min(humval)

hmax = max(humval)

k= 0

for i in humval:

humval[k] = (i - hmin)/(hmax-hmin)

k = k+1

wmin = min(waterval)

wmax = max(waterval)

k= 0

for i in waterval:

waterval[k] = (i - wmin)/(wmax-wmin)

k = k+1

d = {'const':const,'soilval':soilval,'tempval':tempval,'humval':humval,'waterval':waterval}

df\_new = pd.DataFrame(d)

# # Plotting normalized values

plt.scatter(soilval,tempval,c='g',marker='o')

plt.title('Scatter plot ')

plt.xlabel('Soil moisture')

plt.ylabel('Temperature')

plt.show()

plt.scatter(soilval,humval,c='y',marker='o')

plt.title('Scatter plot ')

plt.xlabel('Soil Moisture')

plt.ylabel('Humidity')

plt.show()

plt.scatter(soilval,waterval,c='c',marker='o')

plt.title('Scatter plot ')

plt.xlabel('Soil Moisture')

plt.ylabel('Water level')

plt.show()

plt.scatter(tempval,waterval,c='r',marker='o')

plt.title('Scatter plot ')

plt.xlabel('Temperature')

plt.ylabel('Water Level')

plt.show()

plt.scatter(humval,waterval,c='b',marker='o')

plt.title('Scatter plot ')

plt.xlabel('Humidity')

plt.ylabel('water level')

plt.show()

train, test = train\_test\_split(df\_new, test\_size=0.4)

# temp and soilmoisture and humval with constant

x\_train = train[['tempval','soilval','const','humval']]

y\_train = train['waterval']

x\_test = test[['tempval','soilval','const','humval']]

y\_test = test['waterval']

#all Data

X = df\_new[['tempval','soilval','const','humval']]

y = df\_new['waterval']

regr = linear\_model.LinearRegression()

regr.fit(x\_train, y\_train)

y\_pred = regr.predict(x\_test)

finalwater= y\_pred[-1]\*100

finalwaterlvl = int(finalwater)

print("The output waterlevel using linear regression is=",finalwaterlvl,"ml")

error = sqrt(mean\_squared\_error(y\_test, y\_pred)) # calculate rmse

print('RMSE value for linear regression is =', error\*100)

model = sm.OLS(y,X).fit()

p= model.summary()

print(p)

# #soilmoisture value from sensor

# sensorvalue = int(getmoisinfo())

# print("actual soil moisture value")

# print(sensorvalue)

# #Sending data to arduino

# sendwater(finalwaterlvl)

# #adding data to the database for more accuracy

# temp1,humid1 = getweatherdata()

# adddata(sensorvalue,temp1,humid1,finalwaterlvl)

# import mysql.connector

# mydb = mysql.connector.connect(

# host="localhost",

# user="root",

# passwd="safety",

# database="plantdb"

# )

# mycursor = mydb.cursor()

# sql = "INSERT INTO plantdb1 (wateradded, waterlevel) VALUES (%s, %s)"

# val = (finalwaterlvl, waterlevel)

# mycursor.execute(sql, val)

# mydb.commit()

# print(mycursor.rowcount, "record inserted.")

Backend code

#libraries for database, HTTP connection and Json handling

import requests

import json

import pandas as pd

#URl connection for weather API

url = 'http://api.openweathermap.org/data/2.5/weather?q=Mumbai&appid=f19183c7cecdf1a18c3e84b138490e3e'

'''

#connection for databse

conn = sqlite3.connect('db1.db')

c = conn.cursor()

#creates a databse (if does not already exists) fields - Soil moisture, Temperature, humidity, and water to be poured

c.execute('CREATE TABLE IF NOT EXISTS weatherdata (moisture REAL, temperature REAL, humidity REAL, waterlevel REAL)')

'''

def getweatherdata():

#getting data from the API

res = requests.get(url)

data = res.json()

#seperation of data

inter = data["main"]

temp = inter["temp"]

hum = inter["humidity"]

return temp,hum

#get data from web server

#Demo code --- add IP from C++ code

def getmoisinfo():

mois = requests.get('http://192.168.43.161/Python')

x = mois.text

return x

# extracting response text

def sendwater(waterlvl):

API\_ENDPOINT = "http://192.168.43.161/Python"

PARAMS= {'%d' % int(waterlvl):waterlvl}

r = requests.post(url = API\_ENDPOINT, params = PARAMS)

# funtion to add data to the database

def adddata(mois,temp,hum,water):

'''

c.execute("INSERT INTO weatherdata (moisture, temperature, humidity, waterlevel) VALUES (?,?,?,?)",(mois,temp,hum,water))

conn.commit()

'''

#TODO add important code

df = pd.read\_csv('data\_base.csv')

df\_new = pd.DataFrame([mois,temp,hum,water])

with open('foo.csv', 'a') as f:

(df\_new).to\_csv(f, header=False)

def getdbdata():

#database checking

# displays all the contents of the data base

'''

c.execute('SELECT \* FROM weatherdata')

x = c.fetchall()

print(x)

y = []

z = []

for i in x:

y.append([i[3],i[1],i[2]])

z.append([i[0]])

'''

df = pd.read\_csv('data\_base.csv')

#print(df.head())

return df

Code for NodeMCU Arduino IDE

#include <ESP8266WiFi.h>

#include <ESP8266WebServer.h>

ESP8266WebServer server(80);

int Readpin=A0;

int wlevelpin=A1; //check the pin numberon the board and change this

int Writepin=D0;

void setup() {

pinMode(Readpin,INPUT);

pinMode(wlevelpin,INPUT);

pinMode( Writepin,OUTPUT);

Serial.begin(115200);

WiFi.begin("My\_wifi", "adnanadnan"); //Connect to the WiFi networks

while (WiFi.status() != WL\_CONNECTED) { //Wait for connection

delay(500);

Serial.println("Waiting to connect…");

}

Serial.print("IP address: ");

Serial.println(WiFi.localIP()); //Print the local IP of the webserver

server.on("/Python", handlePath); //Associate the handler function to the path

server.begin(); //Start the server

Serial.println("Server listening");

}

void loop() {

server.handleClient(); //Handling of incoming requests

}

int callerValue(){

int sVal=0;

int temp = analogRead(A0);

Serial.print("Actual Sensor Value: ");

Serial.println(temp);

sVal=map(temp, 0, 1023, 0, 100);

Serial.print("Mapped Sensor Value: ");

Serial.println(sVal);

return sVal;

}

int callerValue1(){

int wlevel=0;

int wlevel = analogRead(A1);

Serial.print("Water level sensor value: ");

Serial.println(wlevel);

return wlevel;

}

void handlePath() {

int a;

//Handler for the path

int ca=callerValue();

int ca1=callerValue1();

String thisString=String(ca);

String thisString1=String(ca1);

server.send(200, "int", thisString);

server.send(300, "int",

if (server.args() > 0 ) { // Arguments were received

for ( uint8\_t i = 0; i < server.args(); i++ ) {

Serial.print("Input received was: ");

Serial.println(server.arg(i)); // Display the argument

a = server.arg(i).toInt();

a\*=0.0592\*1000;

boolean flag=true;

while(flag){

digitalWrite(Writepin,HIGH);

delay(a+200);

flag=false;

}

digitalWrite(Writepin,LOW);

Serial.println(a);

// String Str1[1] = {server.arg(i)};

// Argument\_Name = server.argName(i);

// if (server.argName(i) == "user\_input") {

// Serial.print(" Input received was: ");

// Serial.println(server.arg(i));

// Clients\_Response = server.arg(i);

// Serial.println(Clients\_Response);}

}

}}

Code for web app

The backend code with Rest API

from flask import Flask

from flaskext.mysql import MySQL

from flask import Flask,request, render\_template

mysql = MySQL()

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

app.secret\_key = 'key'

app.config['MYSQL\_DATABASE\_USER'] = 'root'

app.config['MYSQL\_DATABASE\_PASSWORD'] = 'safety'

app.config['MYSQL\_DATABASE\_DB'] = 'plantdb'

app.config['MYSQL\_DATABASE\_HOST'] = 'localhost'

mysql.init\_app(app)

@app.route("/")

@app.route("/welcome")

def hello():

return render\_template('welcome.html')

@app.route("/add")

def add():

cursor= mysql.connect().cursor()

cursor.execute('SELECT wateradded FROM plantdb1 ORDER BY plant\_id DESC LIMIT 1')

data = cursor.fetchone()

datax= ''.join((data))

datax2= int(datax)

return render\_template('add.html', water=datax2)

@app.route("/level")

def level():

cursor= mysql.connect().cursor()

cursor.execute('SELECT waterlevel FROM plantdb1 ORDER BY plant\_id DESC LIMIT 1')

data1 = cursor.fetchone()

data11= ''.join((data1))

data2= int(data11)

# if data2<50:

# flash(u'Waterlevel too low, refill container!', 'Warning')

# return redirect(url\_for('hello'))

# else:

return render\_template('level.html', waterlevel=data2)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

HTML codes for frontend

<!DOCTYPE html>

<html>

<head>

<style>

body{

background-color: wheat;

}

p{

color: coral;

text-align: center;

font-family: 'Times New Roman', Times, serif;

font-size: 400%;

}

h2{

color: rgb(255, 58, 32);

text-align: center;

font-family: 'Times New Roman', Times, serif;

font-size: 450%;

}

</style>

<title>Plant Added Water</title>

</head>

<body>

<h2>Amount of added water</h2>

<p>The added water is= {{ water }} ml</p>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Plant Remaining Water</title>

<style>

body{

background-color: rgb(6, 248, 99);

}

p{

color: rgb(27, 165, 9);

text-align: center;

font-family: 'Times New Roman', Times, serif;

font-size: 400%;

}

h2{

color: rgb(4, 54, 15);

text-align: center;

font-family: 'Times New Roman', Times, serif;

font-size: 450%;

}

</style>

</head>

<body>

<h2>Amount of water remaining</h2>

{% if waterlevel>=30 %}

<p>The remaining water is= {{ waterlevel }} ml</p>

{% else %}

<p>Water Level is too low, please refill the container!!</p>

<p>The remaining water is= {{ waterlevel }} ml</p>

{% endif %}

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<style>

html,body{

height:50%;

}

.box{

position: absolute;

border-left: 10px;

background: paleturquoise;

margin: 10px;

padding: 0 7px 0 7px;

max-width: 400px;

border-radius: 10px;

/\* text-align:center; \*/

}

.box1 {

display: inline-block;

padding: 10px;

margin: 50px 0;

max-width:400px;

}

.box2 {

display: inline-block;

padding: 10px;

margin: 50px 0;

/\* margin-top: 100px; \*/

}

body{

background-color: paleturquoise;

background-image: url("img1.jpg");

background-blend-mode:multiply;

background-position: center;

background-repeat: no-repeat;

background-size: auto;

text-align:center;

}

body:before{

content:'';

height:50%;

display:inline-block;

vertical-align:middle;

}

button{

background:#1AAB8A;

color:#fff;

border:none;

position:relative;

height:60px;

font-size:1.6em;

padding:0 2em;

cursor:pointer;

transition:800ms ease all;

outline:none;

}

button:hover{

background:#fff;

color:#1AAB8A;

}

button:before,button:after{

content:'';

position:absolute;

top:0;

right:0;

height:2px;

width:0;

background: #1AAB8A;

transition:400ms ease all;

}

button:after{

right:inherit;

top:inherit;

left:0;

bottom:0;

}

button:hover:before,button:hover:after{

width:100%;

transition:800ms ease all;

}

.container {

text-align: center;

top: 20%;

left: 35%;

position:relative;

}

</style>

</head>

<body>

<div class="container">

<div class="box" >

<h1>Plant Watering System</h1>

<div class="box1" >

<button id="button" type="button">Find Water Added</button>

</div>

<div class="box2" >

<button id="button1" type="button">Find Water Remaining</button>

</div>

</div>

</div>

</body>

</html>

<script type="text/javascript">

document.getElementById("button").onclick = function(){

location.href= "http://localhost:5000/add"

};

document.getElementById("button1").onclick = function(){

location.href= "http://localhost:5000/level"

};

</script>