```
#Intergrated Sensor code
import requests
import RPi.GPIO as GPIO
import time
import Adafruit_DHT
import time
from urllib.request import urlopen
#ph sensor
import busio
import digitalio
import board
import adafruit_mcp3xxx.mcp3008 as MCP
from adafruit_mcp3xxx.analog_in import AnalogIn
spi = busio.SPI(clock=board.SCK, MISO=board.MISO, MOSI=board.MOSI)
cs = digitalio.DigitalInOut(board.D5)
mcp = MCP.MCP3008(spi, cs)

#GPIO SETUP for soil moisture
channel = 16
GPIO.setup(channel, GPIO.IN)

#flame sensor
flame_channel = 20
GPIO.setmode(GPIO.BCM)
GPIO.setup(flame_channel, GPIO.IN)

#motor and relay
relay_out = 21
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(relay_out, GPIO.IN)

def callback(channel):
        if GPIO.input(channel):
                print("No Water Detected")
                token = "W8rhtZNPJm9x4CoKlYwbM1vSzjf3HT0uBFLsIginD56qEOdk2VX82yz6IqH5vCpVBJEFZok
3AmUbPnwu"
                mobile= "9113697895"
                url = "https://www.fast2sms.com/dev/bulk"
                payload = "sender_id=FSTSMS&message=No Water Detected &language=english&route=p&numbers={
}".format(
                                        mobile)
                headers = {
                                        'authorization': token,
                                        'Content-Type': "application/x-www-form-urlencoded",
                                        'Cache-Control': "no-cache",
                                    }
                response = requests.request("POST", url, data=payload, headers=headers)
                print("response is", response.text)
                print("mobile", mobile)
                print("relay on")
                GPIO.setup(relay_out, GPIO.OUT)
                time.sleep(2)
                GPIO.setup(relay_out, GPIO.IN)
```

```
            print("relay off")
        else:
            print("Water Detected")


def flame_callback(flame_channel):
    print('fire  detected')
    token = "W8rhtZNPJm9x4CoKlYwbM1vSzjf3HT0uBFLsIginD56qEOdk2VX82yz6IqH5vCpVBJEFZok3AmUb
Pnwu"
    mobile= "9113697895"

    url = "https://www.fast2sms.com/dev/bulk"
    payload = "sender_id=FSTSMS&message=fire detected &language=english&route=p&numbers={}".format(
                      mobile)
    headers = {
                      'authorization': token,
                      'Content-Type': "application/x-www-form-urlencoded",
                      'Cache-Control': "no-cache",
                  }
    response = requests.request("POST", url, data=payload, headers=headers)
    print("response is", response.text)
    print("mobile", mobile)
    print("relay on")
    GPIO.setup(relay_out, GPIO.OUT)
    time.sleep(5)
    GPIO.setup(relay_out, GPIO.IN)
    print("relay off")


GPIO.add_event_detect(flame_channel, GPIO.BOTH, bouncetime=300)
GPIO.add_event_callback(flame_channel, flame_callback)

GPIO.add_event_detect(channel, GPIO.BOTH, bouncetime=300)
GPIO.add_event_callback(channel, callback)


#DHT
DHT_SENSOR = Adafruit_DHT.DHT11
DHT_PIN = 12

while True:
    humidity, temperature = Adafruit_DHT.read(DHT_SENSOR, DHT_PIN)
    if humidity is not None and temperature is not None:
        ph_channel = AnalogIn(mcp, MCP.P0)
        #print('Raw PH Value: ', ph_channel.value)
        #print('ADC Voltage: ' + str(channel.voltage) + 'V')
        print("Temp={0:0.1f}c Humidity={1:0.1f}%".format(temperature, humidity))
        url = "https://api.thingspeak.com/update?api_key=W162W447JQYE40WM&field1&field1&field1={}&field2
={}".format(humidity,temperature)
        urlopen(url)
        if humidity < 35:
            token = "W8rhtZNPJm9x4CoKlYwbM1vSzjf3HT0uBFLsIginD56qEOdk2VX82yz6IqH5vCpVBJEFZok3A
mUbPnwu"
            mobile= "9113697895"
```

```
        url = "https://www.fast2sms.com/dev/bulk"
        payload = "sender_id=FSTSMS&message=humidity is less &language=english&route=p&numbers={}".for
mat(
                        mobile)
        headers = {
                        'authorization': token,
                        'Content-Type': "application/x-www-form-urlencoded",
                        'Cache-Control': "no-cache",
                    }
        response = requests.request("POST", url, data=payload, headers=headers)
        print("response is", response.text)
        print("mobile", mobile)

      if temperature > 35:
        token = "W8rhtZNPJm9x4CoKlYwbM1vSzjf3HT0uBFLsIginD56qEOdk2VX82yz6IqH5vCpVBJEFZok3A
mUbPnwu"
        mobile= "9113697895"

        url = "https://www.fast2sms.com/dev/bulk"
        payload = "sender_id=FSTSMS&message=Temperature is more &language=english&route=p&numbers={}
".format(
                        mobile)
        headers = {
                        'authorization': token,
                        'Content-Type': "application/x-www-form-urlencoded",
                        'Cache-Control': "no-cache",
                    }
        response = requests.request("POST", url, data=payload, headers=headers)
        print("response is", response.text)
        print("mobile", mobile)
        print("relay on")
        GPIO.setup(relay_out, GPIO.OUT)
        time.sleep(2)
        GPIO.setup(relay_out, GPIO.IN)
        print("relay off")
   else:
      print("Sensor failure")

   time.sleep(3)

#Machine Learning Code

#importing lirarires
import pandas as pd
import os
#print(os.listdir())
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn import svm # svm
from sklearn.naive_bayes import BernoulliNB # navie bayes
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics

#importing data
data = pd.read_excel('crop_csv_file_old.xlsx')

#to find out missing data
data.info()
# no missing data in selected features
#data = data.mean()
#independent varibles  we considered temperature, humidity and soil moisture
X = data.iloc[:10000,5:8]
#dependent varible  fourth column is crop
y = data.iloc[:10000,4]

#caterorical data handling


#split the data into train data and test data,
#we are considering 80% of training data and 20% of testing data
X_train,X_test,Y_train,Y_test = train_test_split(X,y,test_size=0.20,random_state=100)

#training alogithems
"""
# Support vector machine

sv = svm.SVC(kernel='rbf')
sv.fit(X_train, Y_train)
Y_pred_svm = sv.predict(X_test)
score_svm = round(accuracy_score(Y_pred_svm,Y_test)*100,2)
print("The accuracy score achieved using Linear SVM is: "+str(score_svm)+" %")
"""
#Applying  Bernoulli Naive Baye's Algorithm on the dataset
"""
BernNB = BernoulliNB(binarize=.1)
BernNB.fit(X_train,Y_train)
print(BernNB)
y_expect = Y_test
y_pred = BernNB.predict(X_test)
print(accuracy_score(y_expect,y_pred)*100)
out = BernNB.predict([[129,129,129]])
print(out)
"""
"""
# logistic regression

logreg = LogisticRegression()
# fit the model with data
logreg.fit(X_train,Y_train)
y_pred=logreg.predict(X_test)
print(accuracy_score(Y_test,y_pred)*100)
out = logreg.predict([[342,342,342]])
print(out)
"""
```

```
#decision tree
clf_entropy = DecisionTreeClassifier(criterion="entropy",random_state=100,max_depth=5,min_samples_leaf=3)
clf_entropy.fit(X_train,Y_train)
y_pred_en=clf_entropy.predict(X_test)
y_pred_en
print('Accuracy is',accuracy_score(Y_test,y_pred_en)*100)
out = clf_entropy.predict([[35,50,70]])
print(out)

# GUI
from tkinter import *
from tkinter import ttk

root = Tk()
root.title('Crop Prediction System')
root.geometry('850x650')
root.configure(background="grey")
var = StringVar()
label = Label( root, textvariable = var,font=('arial',20,'bold'),bd=20,background="grey")
var.set('Crop Prediction System')
label.grid(row=0,columnspan=6)

label_1 = ttk.Label(root, text ='Temperture',font=("Helvetica", 16),background="grey")
label_1.grid(row=11,column=0)

Entry_1= Entry(root)
Entry_1.grid(row=11,column=1)

label_2 = ttk.Label(root, text ='Humidity',font=("Helvetica", 16),background="grey")
label_2.grid(row=12,column=0)

Entry_2 = Entry(root)
Entry_2.grid(row=12,column=1)


label_3 = ttk.Label(root, text ='Soil Moisture',font=("Helvetica", 16),background="grey")
label_3.grid(row=13,column=0)

Entry_3 = Entry(root)
Entry_3.grid(row=13,column=1)

def predict():
    T_i = Entry_1.get()
    H_i = Entry_2.get()
    S_i = Entry_3.get()

    out = clf_entropy.predict([[T_i,H_i,S_i]])

    output.delete('1.0',END)
    output.insert('1.0',out[0])


b1 = Button(root, text = 'predict',font=("Helvetica", 16),background="grey",command = predict)
```

```
b1.grid(row=20,column=0)


output = Text(root)
output.grid(row=20,column=1)

root.mainloop()
```

Execution Steps

code shuold be dumped in raspberry pi with all the required package installed
1)To run intergrated sensor code
 In raspbrian os it has inbulit option to run the code or python filename.py
2)Next to run machine learning code
 In raspbrian os it has inbulit option to run the code or python filename.py