**GROUP 1 : PLANT DISEASE DETECTION AND PESTIDICDE RECOMMENDATION USING MACHINE LEARNING**

**GUIDE** : DR.V.A. SURYAWANSHI

**GROUP MEMBERS** : 1. GAYATRI GAIKWAD

2. MIHIR BHATTAD

3. SANYUKTA HOLKAR

4. PRITAMSHINGH SOLANKI

**MODULE 1 : GUI\_Master1.py**

import tkinter as tk

from tkinter import ttk, LEFT, END

from PIL import Image , ImageTk

from tkinter.filedialog import askopenfilename

import cv2

import numpy as np

import time

import sqlite3

#import tfModel\_test as tf\_test

global fn

fn=""

##############################################+=============================================================

root = tk.Tk()

root.configure(background="seashell2")

#root.geometry("1300x700")

w, h = root.winfo\_screenwidth(), root.winfo\_screenheight()

root.geometry("%dx%d+0+0" % (w, h))

root.title("MAIN PAGE")

#430

#++++++++++++++++++++++++++++++++++++++++++++

#####For background Image

image2 =Image.open('I5.jpg')

#image2 =image2.resize((false,false), Image.ANTIALIAS)

background\_image=ImageTk.PhotoImage(image2)

background\_label = tk.Label(root, image=background\_image)

background\_label.image = background\_image

#background\_label.place(x=850, y=0) #, relwidth=1, relheight=1)

#

#frame\_display = tk.LabelFrame(root, text=" --Display-- ", width=900, height=250, bd=5, font=('times', 14, ' bold '),bg="lightblue4")

#frame\_display.grid(row=0, column=0, sticky='nw')

#frame\_display.place(x=300, y=100)

#frame\_display1 = tk.LabelFrame(root, text=" --Result-- ", width=900, height=200, bd=5, font=('times', 14, ' bold '),bg="lightblue4")

#frame\_display1.grid(row=0, column=0, sticky='nw')

#frame\_display1.place(x=300, y=430)

#frame\_display2 = tk.LabelFrame(root, text=" --Calaries-- ", width=900, height=50, bd=5, font=('times', 14, ' bold '),bg="lightblue4")

#frame\_display2.grid(row=0, column=0, sticky='nw')

#frame\_display2.place(x=300, y=380)

frame\_alpr = tk.LabelFrame(root, width=1850, height=1850, bd=5, font=('times', 14, ' bold '),bg="#013220")

frame\_alpr.grid(row=0, column=0)

frame\_alpr.place(x=0, y=0)

image2 =Image.open('I6.jpg')

image2 =image2.resize((140,120), Image.ANTIALIAS)

background\_image=ImageTk.PhotoImage(image2)

background\_label = tk.Label(frame\_alpr, image=background\_image)

background\_label.image = background\_image

background\_label.place(x=680, y=25)

lbl = tk.Label(frame\_alpr, text="PLANT DISEASE DETECTION ", font=('Times New Roman', 35,' bold '),bg="#013220",fg="white")

lbl.place(x=425, y=180)

lbl = tk.Label(frame\_alpr, text="AND CURE RECCOMENDATION", font=('Times New Roman', 35,' bold '),bg="#013220",fg="white")

lbl.place(x=400, y=250)

logo\_label=tk.Label()

logo\_label.place(x=300,y=55)

logo\_label1=tk.Label(text='...To develop a system that capable \n To Detect And \n Identify the type of disease...',compound='bottom',font=("Times New Roman", 20, 'bold', 'italic'),width=35, bg="#cce6ff", fg="black")

#logo\_label=tk.Label(height=500, width=400)

logo\_label1.place(x=500,y=590)

def login():

from subprocess import call

call(["python", "GUI\_main.py"])

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

# def register():

# from subprocess import call

# call(["python", "registration.py"])

def window():

root.destroy()

button1 = tk.Button(frame\_alpr, text=" START",command=login,width=15, height=1, font=('times', 15, ' bold '),bg="#3BB9FF",fg="black")

button1.place(x=650, y=450)

# button2 = tk.Button(frame\_alpr, text="LOGIN",command=login,width=15, height=1, font=('times', 15, ' bold '),bg="#3BB9FF",fg="black")

# button2.place(x=200, y=450)

# button3 = tk.Button(frame\_alpr, text="Train Model", command=train\_model, width=12, height=1, font=('times', 15, ' bold '),bg="white",fg="black")

# button3.place(x=10, y=160)

#

root.mainloop()

**MODULE 2 : GUI\_main.py**

# -\*- coding: utf-8 -\*-

"""

Created on Fri Jan 29 14:27:05 2021

@author: om

"""

import tkinter as tk

from tkinter import ttk, LEFT, END

from PIL import Image, ImageTk

from tkinter.filedialog import askopenfilename

from tkinter import messagebox as ms

import cv2

import sqlite3

import os

import numpy as np

import time

#import video\_capture as value

#import lecture\_details as detail\_data

#import video\_second as video1

#import lecture\_video as video

global fn

fn = ""

##############################################+=============================================================

root = tk.Tk()

root.configure(background="white")

# root.geometry("1300x700")

w, h = root.winfo\_screenwidth(), root.winfo\_screenheight()

root.geometry("%dx%d+0+0" % (w, h))

root.title("PLANT DISEASE DETECTION USING MACHINE LEARNING")

# 43

# ++++++++++++++++++++++++++++++++++++++++++++

#####For background Image

image2 = Image.open('gui.jpg')

image2 = image2.resize((1100, 900), Image.ANTIALIAS)

background\_image = ImageTk.PhotoImage(image2)

background\_label = tk.Label(root, image=background\_image)

background\_label.image = background\_image

background\_label.place(x=0, y=0) # , relwidth=1, relheight=1)

#

label\_l1 = tk.Label(root, text="PLANT DISEASE DETECTION USING MACHINE LEARNING",font=("Times New Roman", 30, 'bold'),

background="brown", fg="white", width=70, height=1)

label\_l1.place(x=0, y=0)

#T1.tag\_configure("center", justify='center')

#T1.tag\_add("center", 1.0, "end")

################################$%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

#def clear\_img():

# img11 = tk.Label(root, background='bisque2')

# img11.place(x=0, y=0)

#################################################################$%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

frame\_alpr = tk.LabelFrame(root, text=" ", width=450, height=850, bd=5, font=('times', 14, ' bold '),bg="#00563F")

frame\_alpr.grid(row=0, column=0)

frame\_alpr.place(x=1100, y=50)

################################$%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

# def cap\_video():

# video1.upload()

# #from subprocess import call

# #call(['python','video\_second.py'])

def reg():

from subprocess import call

call(["python","registration.py"])

def log():

from subprocess import call

call(["python","login.py"])

def window():

root.destroy()

button1 = tk.Button(root, text="Login", command=log, width=14, height=1,font=('times', 20, ' bold '), bg="#152238", fg="white")

button1.place(x=1200, y=260)

button2 = tk.Button(root, text="Register",command=reg,width=14, height=1,font=('times', 20, ' bold '), bg="#152238", fg="white")

button2.place(x=1200, y=340)

button3 = tk.Button(root, text="Exit",command=window,width=14, height=1,font=('times', 20, ' bold '), bg="brown", fg="white")

button3.place(x=1200, y=430)

root.mainloop()

**MODULE 3 : registration.py**

import tkinter as tk

# from tkinter import \*

from tkinter import messagebox as ms

import sqlite3

from PIL import Image, ImageTk

import re

import random

import os

window = tk.Tk()

window.geometry("1800x800")

window.title("REGISTRATION FORM")

window.configure(background="grey")

Fullname = tk.StringVar()

address = tk.StringVar()

username = tk.StringVar()

Email = tk.StringVar()

Phoneno = tk.IntVar()

var = tk.IntVar()

age = tk.IntVar()

password = tk.StringVar()

password1 = tk.StringVar()

policeno = tk.IntVar()

value = random.randint(1, 1000)

print(value)

# database code

db = sqlite3.connect('evaluation.db')

cursor = db.cursor()

cursor.execute("CREATE TABLE IF NOT EXISTS admin\_registration"

"(Fullname TEXT, address TEXT, username TEXT, Email TEXT, Phoneno TEXT,Gender TEXT,age TEXT , password TEXT)")

db.commit()

def password\_check(passwd):

SpecialSym =['$', '@', '#', '%']

val = True

if len(passwd) < 6:

print('length should be at least 6')

val = False

if len(passwd) > 20:

print('length should be not be greater than 8')

val = False

if not any(char.isdigit() for char in passwd):

print('Password should have at least one numeral')

val = False

if not any(char.isupper() for char in passwd):

print('Password should have at least one uppercase letter')

val = False

if not any(char.islower() for char in passwd):

print('Password should have at least one lowercase letter')

val = False

if not any(char in SpecialSym for char in passwd):

print('Password should have at least one of the symbols $@#')

val = False

if val:

return val

def insert():

fname = Fullname.get()

addr = address.get()

un = username.get()

email = Email.get()

mobile = Phoneno.get()

gender = var.get()

time = age.get()

pwd = password.get()

cnpwd = password1.get()

with sqlite3.connect('evaluation.db') as db:

c = db.cursor()

# Find Existing username if any take proper action

find\_user = ('SELECT \* FROM admin\_registration WHERE username = ?')

c.execute(find\_user, [(username.get())])

# else:

# ms.showinfo('Success!', 'Account Created Successfully !')

# to check mail

#regex = '^\w+([\.-]?\w+)\*@\w+([\.-]?\w+)\*(\.\w{2,3})+$'

regex='^[a-z0-9]+[\.\_]?[a-z0-9]+[@]\w+[.]\w{2,3}$'

if (re.search(regex, email)):

a = True

else:

a = False

# validation

if (fname.isdigit() or (fname == "")):

ms.showerror("showerror", "Error")

ms.showinfo("Message", "please enter valid name")

elif (addr == ""):

ms.showerror("showerror", "Error")

ms.showinfo("Message", "Please Enter Address")

elif (email == "") or (a == False):

ms.showerror("showerror", "Error")

ms.showinfo("Message", "Please Enter valid email")

elif((len(str(mobile)))<10 or len(str((mobile)))>10):

ms.showerror("showerror", "Error")

ms.showinfo("Message", "Please Enter 10 digit mobile number")

elif ((time > 100) or (time == 0)):

ms.showerror("showerror", "Error")

ms.showinfo("Message", "Please Enter valid age")

elif (c.fetchall()):

ms.showerror('Error!', 'Username Taken Try a Diffrent One.')

elif (pwd == ""):

ms.showerror("showerror", "Error")

ms.showinfo("Message", "Please Enter valid password")

elif (var == False):

ms.showinfo("Message", "Please Enter gender")

elif(pwd=="")or(password\_check(pwd))!=True:

ms.showerror("showerror", "Error")

ms.showinfo("Message", "password must contain atleast 1 Uppercase letter,1 symbol,1 number")

elif (pwd != cnpwd):

ms.showerror("showerror", "Error")

ms.showinfo("Message", "Password Confirm password must be same")

else:

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

with conn:

cursor = conn.cursor()

cursor.execute(

'INSERT INTO admin\_registration(Fullname, address, username, Email, Phoneno, Gender, age , password) VALUES(?,?,?,?,?,?,?,?)',

(fname, addr, un, email, mobile, gender, time, pwd))

conn.commit()

db.close()

ms.askquestion("askquestion", "Are you sure?")

ms.askokcancel("askokcancel", "Want to continue?")

ms.showinfo('Success!', 'Account Created Successfully !')

# window.destroy()

window.destroy()

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

def login():

from subprocess import call

call(["python", "login.py"])

def login1():

from subprocess import call

call(["python", "GUI\_main.py"])

#def register():

# from subprocess import call

# call(["python", "lecture\_login.py"])

# assign and define variable

# def login():

#####For background Image

image2 = Image.open('reg.jpg')

image2 = image2.resize((900, 900), Image.ANTIALIAS)

background\_image = ImageTk.PhotoImage(image2)

background\_label = tk.Label(window, image=background\_image)

background\_label.image = background\_image

background\_label.place(x=0, y=0) # , relwidth=1, relheight=1)

image2 = Image.open('reg2 (2).jpg')

image2 = image2.resize((800, 900), Image.ANTIALIAS)

background\_image = ImageTk.PhotoImage(image2)

background\_label = tk.Label(window, image=background\_image)

background\_label.image = background\_image

background\_label.place(x=800, y=0)

l1 = tk.Label(window, text="Registration Form", font=("Times new roman", 30, "bold"), bg="#192841", fg="white")

l1.place(x=990, y=50)

# that is for label1 registration

l2 = tk.Label(window, text="Full Name :", width=12, font=("Times new roman", 15, "bold"), bg="snow")

l2.place(x=890, y=150)

t1 = tk.Entry(window, textvar=Fullname, width=20, font=('', 15))

t1.place(x=1150, y=150)

# that is for label 2 (full name)

l3 = tk.Label(window, text="Address :", width=12, font=("Times new roman", 15, "bold"), bg="snow")

l3.place(x=890, y=200)

t2 = tk.Entry(window, textvar=address, width=20, font=('', 15))

t2.place(x=1150, y=200)

# that is for label 3(address)

# that is for label 4(blood group)

l5 = tk.Label(window, text="E-mail :", width=12, font=("Times new roman", 15, "bold"), bg="snow")

l5.place(x=890, y=250)

t4 = tk.Entry(window, textvar=Email, width=20, font=('', 15))

t4.place(x=1150, y=250)

# that is for email address

l6 = tk.Label(window, text="Phone number :", width=12, font=("Times new roman", 15, "bold"), bg="snow")

l6.place(x=890, y=300)

t5 = tk.Entry(window, textvar=Phoneno, width=20, font=('', 15))

t5.place(x=1150, y=300)

# phone number

l7 = tk.Label(window, text="Gender :", width=12, font=("Times new roman", 15, "bold"), bg="snow")

l7.place(x=890, y=350)

# gender

tk.Radiobutton(window, text="Male", padx=5, width=5, bg="snow", font=("bold", 15), variable=var, value=1).place(x=1300,

y=350)

tk.Radiobutton(window, text="Female", padx=20, width=4, bg="snow", font=("bold", 15), variable=var, value=2).place(

x=1150, y=350)

l8 = tk.Label(window, text="Age :", width=12, font=("Times new roman", 15, "bold"), bg="snow")

l8.place(x=890, y=400)

t6 = tk.Entry(window, textvar=age, width=20, font=('', 15))

t6.place(x=1150, y=400)

l4 = tk.Label(window, text="User Name :", width=12, font=("Times new roman", 15, "bold"), bg="snow")

l4.place(x=890, y=450)

t3 = tk.Entry(window, textvar=username, width=20, font=('', 15))

t3.place(x=1150, y=450)

l9 = tk.Label(window, text="Password :", width=12, font=("Times new roman", 15, "bold"), bg="snow")

l9.place(x=890, y=500)

t9 = tk.Entry(window, textvar=password, width=20, font=('', 15), show="\*")

t9.place(x=1150, y=500)

l10 = tk.Label(window, text="Confirm Password:", width=13, font=("Times new roman", 15, "bold"), bg="snow")

l10.place(x=890, y=550)

t10 = tk.Entry(window, textvar=password1, width=20, font=('', 15), show="\*")

t10.place(x=1150, y=550)

btn = tk.Button(window, text="Register", bg="#192841",font=("",20),fg="white", width=9, height=0, command = insert)

btn.place(x=970, y=600)

btn = tk.Button(window, text="Back", bg="#192841",font=("",20),fg="white", width=9, height=0, command = login1)

btn.place(x=1200, y=600)

#btn = tk.Button(window, text="login", bg="#192841",font=("",20),fg="white", width=9, height=0, command=login)

#btn.place(x=350, y=600)

# tologin=tk.Button(window , text="Go To Login", bg ="dark green", fg = "white", width=15, height=2, command=login)

# tologin.place(x=330, y=600)

window.mainloop()

**MODULE 4 : login.py**

import tkinter as tk

from tkinter import ttk, LEFT, END

from tkinter import messagebox as ms

import sqlite3

from PIL import Image, ImageTk

import re

##############################################+=============================================================

root = tk.Tk()

root.configure(background="black")

# root.geometry("1300x700")

w, h = root.winfo\_screenwidth(), root.winfo\_screenheight()

root.geometry("1700x950")

root.title("Login Form")

username = tk.StringVar()

password = tk.StringVar()

# ++++++++++++++++++++++++++++++++++++++++++++

#####For background Image

image2 = Image.open('reg2 (2).jpg')

image2 = image2.resize((w,h), Image.ANTIALIAS)

background\_image = ImageTk.PhotoImage(image2)

background\_label = tk.Label(root, image=background\_image)

background\_label.image = background\_image

background\_label.place(x=0, y=0) # , relwidth=1, relheight=1)

image2 =Image.open('log (1).jpg')

image2 =image2.resize((750,890), Image.ANTIALIAS)

background\_image=ImageTk.PhotoImage(image2)

background\_label = tk.Label(root, image=background\_image)

background\_label.image = background\_image

background\_label.place(x=0, y=0)

def registration():

from subprocess import call

call(["python","registration.py"])

root.destroy()

def login():

# Establish Connection

with sqlite3.connect('evaluation.db') as db:

c = db.cursor()

# Find user If there is any take proper action

db = sqlite3.connect('evaluation.db')

cursor = db.cursor()

cursor.execute("CREATE TABLE IF NOT EXISTS admin\_registration"

"(Fullname TEXT, address TEXT, username TEXT, Email TEXT, Phoneno TEXT,Gender TEXT,age TEXT , password TEXT)")

db.commit()

find\_entry = ('SELECT \* FROM admin\_registration WHERE username = ? and password = ?')

c.execute(find\_entry, [(username.get()), (password.get())])

result = c.fetchall()

if result:

msg = ""

# self.logf.pack\_forget()

# self.head['text'] = self.username.get() + '\n Loged In'

# msg = self.head['text']

# self.head['pady'] = 150

print(msg)

ms.showinfo("messege", "LogIn sucessfully")

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

from subprocess import call

call(['python','GUI\_Master\_old.py'])

root.destroy()

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

else:

ms.showerror('Oops!', 'Username Or Password Did Not Found/Match.')

def Login():

from subprocess import call

call(["python", "GUI\_main.py"])

title=tk.Label(root, text="Login Here", font=("Algerian", 30, "bold","italic"),bd=5,bg="black",fg="white")

title.place(x=1100,y=150,width=250)

# Login\_frame=tk.Frame(root,bg="white")

# Login\_frame.place(x=900,y=300)

frame=tk.Frame(root,width=550,height=400,bg="white")

frame.place(x=900,y=240)

logolbl=tk.Label(frame,bd=0).grid(row=0,columnspan=2,pady=20)

lbluser=tk.Label(frame,text="Username",compound=LEFT,font=("Times new roman", 20, "bold"),bg="white").grid(row=1,column=0,padx=20,pady=10)

txtuser=tk.Entry(frame,bd=5,border=0,textvariable=username,font=("",15))

txtuser.grid(row=1,column=1,padx=20)

# frame=tk.Frame(root,width=550,height=400,bg="white")

# frame.place(x=530,y=240)

tk.Frame(frame,width=300,height=2,bg='black').place(x=220,y=100)

lblpass=tk.Label(frame,text="Password",compound=LEFT,font=("Times new roman", 20, "bold"),bg="white").grid(row=2,column=0,padx=50,pady=10)

txtpass=tk.Entry(frame,bd=5,border=0,textvariable=password,show="\*",font=("",15))

txtpass.grid(row=2,column=1,padx=20)

tk.Frame(frame,width=300,height=2,bg='black').place(x=220,y=150)

btn\_log=tk.Button(frame,text="Login",command=login,width=15,font=("Times new roman", 14, "bold"),bg="Green",fg="black")

btn\_log.grid(row=3,column=1,pady=10)

btn\_reg=tk.Button(frame,text="Create Account",command=registration,width=15,font=("Times new roman", 14, "bold"),bg="red",fg="black")

btn\_reg.grid(row=3,column=0,pady=10)

btn\_reg=tk.Button(frame,text="Back",command=Login,width=10,font=("Times new roman", 14, "bold"),bg="red",fg="black")

btn\_reg.grid(row=4,column=3,pady=10)

root.mainloop()

**MODULE 5 : GUI\_Master\_old.py**

import tkinter as tk

from tkinter import ttk, LEFT, END

from PIL import Image , ImageTk

from tkinter.filedialog import askopenfilename

import cv2

import numpy as np

import time

import CNNModel

import sqlite3

#import tfModel\_test as tf\_test

global fn

fn=""

##############################################+=============================================================

root = tk.Tk()

root.configure(background="#00563f")

#root.geometry("1300x700")

w, h = root.winfo\_screenwidth(), root.winfo\_screenheight()

root.geometry("%dx%d+0+0" % (w, h))

root.title("PLANT DISEASE DETECTION USING MACHINE LEARNING")

#430

#++++++++++++++++++++++++++++++++++++++++++++

#####For background Image

# image2 =Image.open('a4.jpg')

# image2 =image2.resize((w,h), Image.ANTIALIAS)

# background\_image=ImageTk.PhotoImage(image2)

# background\_label = tk.Label(root, image=background\_image)

# background\_label.image = background\_image

# background\_label.place(x=0, y=0) #, relwidth=1, relheight=1)

#img=ImageTk.PhotoImage(Image.open("B4.png"))

#img2=ImageTk.PhotoImage(Image.open("B5.jpg"))

#img3=ImageTk.PhotoImage(Image.open("B7.jpg"))

#

lbl = tk.Label(root, text="PLANT DISEASE DETECTION USING MACHINE LEARNING", font=('times', 35,' bold '), height=1, width=55,bg="maroon",fg="white")

lbl.place(x=0, y=3)

frame\_alpr = tk.LabelFrame(root, text=" --Process-- ", width=220, height=350, bd=5, font=('times', 14, ' bold '),bg="green")

frame\_alpr.grid(row=0, column=0, sticky='nw')

frame\_alpr.place(x=10, y=90)

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

def train\_model():

update\_label("Model Training Start...............")

start = time.time()

X= CNNModel.main()

end = time.time()

ET="Execution Time: {0:.4} seconds \n".format(end-start)

msg="Model Training Completed.."+'\n'+ X + '\n'+ ET

print(msg)

import functools

import operator

def convert\_str\_to\_tuple(tup):

s = functools.reduce(operator.add, (tup))

return s

def test\_model\_proc(fn):

from tensorflow.keras.models import load\_model

#from keras.optimizers import Adam

# global fn

IMAGE\_SIZE = 64

LEARN\_RATE = 1.0e-4

CH=3

print(fn)

if fn!="":

# Model Architecture and Compilation

model = load\_model('plant\_model.h5')

img = Image.open(fn)

img = img.resize((IMAGE\_SIZE,IMAGE\_SIZE))

img = np.array(img)

img = img.reshape(1,IMAGE\_SIZE,IMAGE\_SIZE,3)

img = img.astype('float32')

img = img / 255.0

print('img shape:',img)

prediction = model.predict(img)

print(np.argmax(prediction))

plant=np.argmax(prediction)

print(plant)

if plant == 0:

Cd='CROP NAME:- Apple'+'\n DISEASE NAME:- Apple\_scab Disease'+'\n PESTICIDE:- Superstar 65wp Dodine 65%wp Fungicide.'+'\n DIRECTION TO USE:- 600-800gm in 300 liter of water per acre.'

elif plant == 1:

Cd='CROP NAME:- Strawberry'+'\n DISEASE NAME:-Strawberry\_Leaf\_scorch Disease'+'\n PESTICIDE:- Cabrio Top.'+'\n DIRECTION TO USE:- 1-2gm in one liter of water per acre.'

elif plant == 2:

Cd='CROP NAME:- Corn'+'\n DISEASE NAME:-Corn\_Northern\_Blight Disease'+'\n PESTICIDE:- Deloro Fungicide.'+'\n DIRECTION TO USE:- 1.5-2gm in one liter of water per acre.'

A=Cd

return A

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

def update\_label(str\_T):

#clear\_img()

result\_label = tk.Label(root, text=str\_T, width=50, font=("bold", 25), bg='bisque2', fg='black')

result\_label.place(x=300, y=450)

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

def test\_model():

global fn

if fn!="":

update\_label("Model Testing Start...............")

start = time.time()

X=test\_model\_proc(fn)

#X1="Selected Image is {0}".format(X)

x2=format(X)+""

end = time.time()

ET="\n".format(end-start)

msg="Image Testing Completed.."+'\n'+ x2 + '\n'+ ET

fn=""

else:

msg="Please Select Image For Prediction...."

update\_label(msg)

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

def openimage():

global fn

fileName = askopenfilename(initialdir='C:/Users/Dell/OneDrive/Desktop/code 2/cure new code/test\_set', title='Select image for Aanalysis ',

filetypes=[("all files", "\*.\*")])

IMAGE\_SIZE=200

imgpath = fileName

fn = fileName

# img = Image.open(imgpath).convert("L")

img = Image.open(imgpath)

img = img.resize((IMAGE\_SIZE,200))

img = np.array(img)

# img = img / 255.0

# img = img.reshape(1,IMAGE\_SIZE,IMAGE\_SIZE,3)

x1 = int(img.shape[0])

y1 = int(img.shape[1])

im = Image.fromarray(img)

imgtk = ImageTk.PhotoImage(im)

img = tk.Label(root, image=imgtk, height=250, width=250)

img.image = imgtk

img.place(x=300, y=100)

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

def convert\_grey():

global fn

IMAGE\_SIZE=200

img = Image.open(fn)

img = img.resize((IMAGE\_SIZE,200))

img = np.array(img)

x1 = int(img.shape[0])

y1 = int(img.shape[1])

gs = cv2.cvtColor(cv2.imread(fn, 1), cv2.COLOR\_RGB2GRAY)

gs = cv2.resize(gs, (x1, y1))

retval, threshold = cv2.threshold(gs, 0, 255, cv2.THRESH\_BINARY\_INV + cv2.THRESH\_OTSU)

print(threshold)

im = Image.fromarray(gs)

imgtk = ImageTk.PhotoImage(image=im)

#result\_label1 = tk.Label(root, image=imgtk, width=250, font=("bold", 25), bg='bisque2', fg='black',height=250)

#result\_label1.place(x=300, y=400)

img2 = tk.Label(root, image=imgtk, height=250, width=250,bg='white')

img2.image = imgtk

img2.place(x=580, y=100)

im = Image.fromarray(threshold)

imgtk = ImageTk.PhotoImage(image=im)

img3 = tk.Label(root, image=imgtk, height=250, width=250)

img3.image = imgtk

img3.place(x=880, y=100)

#result\_label1 = tk.Label(root, image=imgtk, width=250,height=250, font=("bold", 25), bg='bisque2', fg='black')

#result\_label1.place(x=300, y=400)

def Next():

from subprocess import call

call(["python","ImageP.py"])

root.destroy()

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

def window():

root.destroy()

button1 = tk.Button(frame\_alpr, text=" Select\_Image ", command=openimage,width=15, height=1, font=('times', 15, ' bold '),bg="white",fg="black")

button1.place(x=10, y=40)

button2 = tk.Button(frame\_alpr, text="Image\_preprocess", command=convert\_grey, width=15, height=1, font=('times', 15, ' bold '),bg="white",fg="black")

button2.place(x=10, y=100)

button5 = tk.Button(frame\_alpr, text="Next", command=Next,width=15, height=1, font=('times', 15, ' bold '),bg="yellow4",fg="white")

button5.place(x=10, y=160)

#button3 = tk.Button(frame\_alpr, text="Train Model", command=train\_model, width=12, height=1, font=('times', 15, ' bold '),bg="white",fg="black")

#button3.place(x=50, y=260)

# #

button4 = tk.Button(frame\_alpr, text="CNN\_Prediction", command=test\_model,width=15, height=1,bg="white",fg="black", font=('times', 15, ' bold '))

button4.place(x=10, y=160)

# #

#

#button5 = tk.Button(frame\_alpr, text="button5", command=window,width=8, height=1, font=('times', 15, ' bold '),bg="yellow4",fg="white")

#button5.place(x=450, y=20)

exit = tk.Button(frame\_alpr, text="Exit", command=window, width=15, height=1, font=('times', 15, ' bold '),bg="red",fg="white")

exit.place(x=10, y=220)

root.mainloop()

**MODULE 6 : CNNModel.py**

def main():

import numpy as np

from keras.models import Sequential

from keras.layers import Convolution2D

from keras.layers import MaxPooling2D

from keras.layers import Flatten

from keras.layers import Dense, Dropout

from keras import optimizers

basepath="C:/Users/Dell/OneDrive/Desktop/code 2/cure new code"

# Initialing the CNN

classifier = Sequential()

# Step 1 - Convolution Layer

classifier.add(Convolution2D(32, 1, 1, input\_shape = (64, 64, 3), activation = 'relu'))

#step 2 - Pooling

classifier.add(MaxPooling2D(pool\_size =(2,2)))

# Adding second convolution layer

classifier.add(Convolution2D(32, 1, 1, activation = 'relu'))

classifier.add(MaxPooling2D(pool\_size =(2,2)))

#Adding 3rd Concolution Layer

classifier.add(Convolution2D(64, 1, 1, activation = 'relu'))

classifier.add(MaxPooling2D(pool\_size =(2,2)))

#Step 3 - Flattening

classifier.add(Flatten())

#Step 4 - Full Connection

classifier.add(Dense(256, activation = 'relu'))

classifier.add(Dropout(0.5))

classifier.add(Dense(3, activation = 'softmax')) #change class no.

#Compiling The CNN

classifier.compile(

optimizer = optimizers.SGD(lr = 0.01),

loss = 'categorical\_crossentropy',

metrics = ['accuracy'])

#Part 2 Fittting the CNN to the image

from keras.preprocessing.image import ImageDataGenerator

train\_datagen = ImageDataGenerator(

rescale=1./255,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True)

test\_datagen = ImageDataGenerator(rescale=1./255)

training\_set = train\_datagen.flow\_from\_directory(

basepath + '/training\_set',

target\_size=(64, 64),

batch\_size=32,

class\_mode='categorical')

test\_set = test\_datagen.flow\_from\_directory(

basepath + '/test\_set',

target\_size=(64, 64),

batch\_size=32,

class\_mode='categorical')

steps\_per\_epoch = int( np.ceil(training\_set.samples / 32) )

val\_steps = int( np.ceil(test\_set.samples / 32) )

model = classifier.fit\_generator(

training\_set,

steps\_per\_epoch=steps\_per\_epoch,

epochs=300,

validation\_data = test\_set,

validation\_steps =val\_steps

)

#Saving the model

#import h5py

classifier.save(basepath + '/plant\_model.h5')

scores = classifier.evaluate(test\_set, verbose=1)

B="Testing Accuracy: %.2f%%" % (scores[1]\*100)

print(B)

scores = classifier.evaluate(training\_set, verbose=1)

C="Training Accuracy: %.2f%%" % (scores[1]\*100)

print(C)

msg=B+'\n'+ C

import matplotlib.pyplot as plt

# summarize history for accuracy

plt.plot(model.history['accuracy'])

plt.plot(model.history['val\_accuracy'])

plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper left')

plt.savefig(basepath + "/accuracy.png",bbox\_inches='tight')

plt.show()

# summarize history for loss

plt.plot(model.history['loss'])

plt.plot(model.history['val\_loss'])

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper left')

plt.savefig(basepath + "/loss.png",bbox\_inches='tight')

plt.show()

return msg

**MODULE 7 : model\_CNN.py**

# -\*- coding: utf-8 -\*-

"""

Created on Thu Jan 7 19:11:21 2021

@author: srcdo

"""

from keras.models import Sequential

from keras.layers.core import Dense, Dropout, Activation, Flatten

from keras.layers.convolutional import Convolution2D, MaxPooling2D

from keras.optimizers import SGD,RMSprop,Adam

from keras.utils import np\_utils

import numpy as np

import matplotlib.pyplot as plt

import matplotlib

import os

import theano

from PIL import Image

from numpy import \*

# SKLEARN

from sklearn.utils import shuffle

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

#from Alexnet import AlexNet

# input image dimensions

img\_rows, img\_cols = 64, 64

# number of channels

img\_channels = 3

#%%

# data

path1 = 'F:/Softech Solution Code/Plant Disease Detection/t' #path of folder of images

path2 = 'F:/Softech Solution Code/Plant Disease Detection' #path of folder to save images

listing = os.listdir(path1)

num\_samples=size(listing)

print(num\_samples)

for file in listing:

im = Image.open(path1 + '\\' + file)

img = im.resize((img\_rows,img\_cols))

gray = img.convert(mode='RGB')

#need to do some more processing here

gray.save(path2 +'\\' + file, "JPEG")

imlist = os.listdir(path2)

im1 = array(Image.open('input\_data\_resized/5/' + imlist[0])) # open one image to get size

m,n = im1.shape[0:2] # get the size of the images

imnbr = len(imlist) # get the number of images

# # create matrix to store all flattened images

# immatrix = array([array(Image.open('input\_data\_resized/'+ im2)).flatten()

# for im2 in imlist],'f')

# label=np.ones((num\_samples,),dtype = int)

# label[0:245]=0

# label[245:288]=1

# size(label)

# data,Label = shuffle(immatrix,label, random\_state=2)

# train\_data = [data,Label]

# img=immatrix[439].reshape(img\_rows,img\_cols)

# # plt.imshow(img)

# # plt.imshow(img,cmap='gray')

# print (train\_data[0].shape)

# print (train\_data[1].shape)

# #%%

# #batch\_size to train

# batch\_size = 32

# # number of output classes

# nb\_classes = 7

# # number of epochs to train

# nb\_epoch = 100

# # number of convolutional filters to use

# nb\_filters = 64

# # size of pooling area for max pooling

# nb\_pool = 2

# # convolution kernel size

# nb\_conv = 3

# #%%

# (X, y) = (train\_data[0],train\_data[1])

# # STEP 1: split X and y into training and testing sets

# X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

# X\_train = X\_train.reshape(X\_train.shape[0], img\_rows, img\_cols , 1)

# X\_test = X\_test.reshape(X\_test.shape[0], img\_rows, img\_cols , 1)

# X\_train = X\_train.astype('float32')

# X\_test = X\_test.astype('float32')

# X\_train /= 255

# X\_test /= 255

# print('X\_train shape:', X\_train.shape)

# print(X\_train.shape[0], 'train samples')

# print(X\_test.shape[0], 'test samples')

# # convert class vectors to binary class matrices

# Y\_train = np\_utils.to\_categorical(y\_train, nb\_classes)

# Y\_test = np\_utils.to\_categorical(y\_test, nb\_classes)

# i = 20

# #plt.imshow(X\_train[i, 0], interpolation='nearest')

# print("label : ", Y\_train[i,:])

# #%%

# model = Sequential()

# # #model.add(Convolution2D(nb\_filters, nb\_conv, nb\_conv, padding='same',input\_shape=(1, img\_rows, img\_cols)))

# # model.add(Convolution2D(64,3, 3, strides=(4, 4), padding='same',input\_shape=(1, img\_rows, img\_cols), activation='relu'))

# # convout1 = Activation('relu')

# # model.add(convout1)

# # model.add(Convolution2D(nb\_filters, nb\_conv, nb\_conv))

# # convout2 = Activation('relu')

# # model.add(convout2)

# # model.add(MaxPooling2D(pool\_size=(nb\_pool, nb\_pool)))

# # model.add(Dropout(0.5))

# # model.add(Flatten())

# # model.add(Dense(128))

# # model.add(Activation('relu'))

# # model.add(Dropout(0.5))

# # model.add(Dense(nb\_classes))

# # model.add(Activation('softmax'))

# # model.compile(loss='categorical\_crossentropy', optimizer='adadelta')

# # model.add(Conv2D(

# # 96, 11, strides=(4, 4), input\_shape=(IMAGE\_SIZE, IMAGE\_SIZE, CHANNELS), activation='relu'))

# # model.add(MaxPooling2D(pool\_size=(3, 3), strides=2))

# # model.add(BatchNormalization())

# # # Second Convolution Block

# # model.add(Conv2D(256, 5, strides=(1, 1),

# # activation='relu', padding='same'))

# # model.add(MaxPooling2D(pool\_size=(3, 3), strides=2))

# # model.add(BatchNormalization())

# # # Third Convolution Block

# # model.add(Conv2D(384, 3, strides=(1, 1),

# # activation='relu', padding='same'))

# # model.add(MaxPooling2D(pool\_size=(3, 3), strides=2))

# # # Fourth Convolution Block

# # model.add(Conv2D(384, 3, strides=(1, 1),

# # activation='relu', padding='same'))

# # # Fifth Convolution Block

# # model.add(Conv2D(256, 3, strides=(1, 1),

# # activation='relu', padding='same'))

# # model.add(Dropout(0.5))

# # # Fully connected layer

# # model.add(Flatten())

# # # First hidden unit

# # model.add(Dense(512, activation='relu', kernel\_initializer='uniform'))

# # model.add(Dropout(0.5))

# # # Second hidden unit

# # model.add(Dense(512, activation='relu', kernel\_initializer='uniform'))

# # model.add(Dropout(0.5))

# # # Output layer

# # model.add(Dense(CLASSES, activation='softmax',

# # kernel\_initializer='uniform'))

# # model.summary()

# LEARN\_RATE = 1.0e-4

# # Model Architecture and Compilation

# model = AlexNet(7, 200, 1)

# adam = Adam(lr=LEARN\_RATE, beta\_1=0.9, beta\_2=0.999, epsilon=None, decay=0.0)

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

# from keras.callbacks import ModelCheckpoint

# from keras import callbacks

# filename='model\_train\_new.csv'

# csv\_log=callbacks.CSVLogger(filename, separator=',', append=False)

# early\_stopping=callbacks.EarlyStopping(monitor='val\_loss', min\_delta=0, patience=0, verbose=0, mode='min')

# checkpoint = ModelCheckpoint('best\_model.h5', monitor='val\_loss', save\_best\_only=True, mode='auto')

# #steps\_per\_epoch = int(len(Y\_train) / BATCH\_SIZE) # 300

# # validation\_steps = int(len(Y\_test) / BATCH\_SIZE) # 90

# callbacks\_list = [csv\_log,early\_stopping,checkpoint]

# #%%

# hist = model.fit(X\_train, Y\_train, batch\_size=batch\_size, epochs=100,

# verbose=1, validation\_data=(X\_test, Y\_test))

# model.save('best\_model.h5')

# model.save('best\_model\_CSV.csv')

# # hist = model.fit(X\_train, Y\_train, batch\_size=batch\_size, epochs=nb\_epoch,

# # show\_accuracy=True, verbose=1, validation\_split=0.2)

# # visualizing losses and accuracy

# train\_loss=hist.history['loss']

# val\_loss=hist.history['val\_loss']

# train\_acc=hist.history['accuracy']

# val\_acc=hist.history['val\_accuracy']

# xc=range(nb\_epoch)

# # plt.figure(1,figsize=(7,5))

# # plt.plot(xc,train\_loss)

# # plt.plot(xc,val\_loss)

# # plt.xlabel('num of Epochs')

# # plt.ylabel('loss')

# # plt.title('train\_loss vs val\_loss')

# # plt.grid(True)

# # plt.legend(['train','val'])

# # print(plt.style.available) # use bmh, classic,ggplot for big pictures

# # plt.style.use(['classic'])

# # plt.figure(2,figsize=(7,5))

# # plt.plot(xc,train\_acc)

# # plt.plot(xc,val\_acc)

# # plt.xlabel('num of Epochs')

# # plt.ylabel('accuracy')

# # plt.title('train\_acc vs val\_acc')

# # plt.grid(True)

# # plt.legend(['train','val'],loc=4)

# # #print plt.style.available # use bmh, classic,ggplot for big pictures

# # plt.style.use(['classic'])

# #%%

# score = model.evaluate(X\_test, Y\_test, verbose=0)

# print('Test score:', score[0])

# print('Test accuracy:', score[1])

# print(model.predict\_classes(X\_test[1:20]))

# print(Y\_test[1:20])

# #%%

# # visualizing intermediate layers

# # output\_layer = model.layers[1].get\_output()

# # output\_fn = theano.function([model.layers[0].get\_input()], output\_layer)

# # # the input image

# # input\_image=X\_train[0:1,:,:,:]

# # print(input\_image.shape)

# # plt.imshow(input\_image[0,0,:,:],cmap ='gray')

# # plt.imshow(input\_image[0,0,:,:])

# # output\_image = output\_fn(input\_image)

# # print(output\_image.shape)

# # # Rearrange dimension so we can plot the result

# # output\_image = np.rollaxis(np.rollaxis(output\_image, 3, 1), 3, 1)

# # print(output\_image.shape)

# # fig=plt.figure(figsize=(8,8))

# # for i in range(32):

# # ax = fig.add\_subplot(6, 6, i+1)

# # #ax.imshow(output\_image[0,:,:,i],interpolation='nearest' ) #to see the first filter

# # ax.imshow(output\_image[0,:,:,i],cmap=matplotlib.cm.gray)

# # plt.xticks(np.array([]))

# # plt.yticks(np.array([]))

# # plt.tight\_layout()

# # plt

# # Confusion Matrix

# from sklearn.metrics import classification\_report,confusion\_matrix

# Y\_pred = model.predict(X\_test)

# print(Y\_pred)

# y\_pred = np.argmax(Y\_pred, axis=1)

# print(y\_pred)

# # (or)

# # y\_pred = model.predict\_classes(X\_test)

# # print(y\_pred)

# p=model.predict(X\_test) # to predict probability

# target\_names = ['class 0(Normal)',

# 'class 1(Abnormal\_Agenesis\_of\_the\_Corpus\_Callosum)',

# 'class 2(Abnormal\_Agenesis\_of\_the\_Septi\_Pellucidi)',

# 'class 3(Abnormal\_Cerebellar\_Hypoplasia)',

# 'class 4(Abnormal\_Dandy\_Walker\_VariantMalformation)',

# 'class 5(Abnormal\_Megacisterna\_Magna)',

# 'class 6(Abnormal\_Venous\_Malformation)']

# size(target\_names)

# print(classification\_report(np.argmax(Y\_test,axis=1), y\_pred,target\_names=target\_names))

# print(confusion\_matrix(np.argmax(Y\_test,axis=1), y\_pred))

# # saving weights

# fname = "weights-Test-CNN.h5"

# model.save\_weights(fname,overwrite=True)

# # Loading weights

# fname = "weights-Test-CNN.h5"

# model.load\_weights(fname)