```
from tkinter import *
from PIL import ImageTk, Image
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from pandas import DataFrame,Series
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import QuantileTransformer
from sklearn.pipeline import make_pipeline
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score # R square
import geatpy as ea
import numpy as np
import cairosvg
from tkinter import messagebox
class Application(Frame):
    def __init__(self, master=None):
        super().__init__(master)
        self.master = master
        self.place()
        self.createWidget()
    def createWidget(self):
        self.lbl_intro=Label(text='Multi-objective optimization program for seismic performance of highway
bridges',bg='grey', fg='white',anchor=CENTER,font=('Times', 13), width=65, height=2)
        self.lbl_intro.place(x=100, y=0)
        global photo
        self.lbl1=Label(text='PGA(g)')
        self.lbl2=Label(text='Pier Height(m)')
        self.lbl3=Label(text='Span Length(m)')
        self.lbl4=Label(text='Beam Width(m)')
        self.lbl5=Label(text='Pier Diameter(m)')
        self.lbl6=Label(text='Steel Strength(MPa)')
        self.lbl7=Label(text='Concrete Strength(MPa)')
        self.lbl8=Label(text='Reinforcement Ratio(%)')
        self.lbl9=Label(text='Hoop Ratio(%)')
        self.t1=Entry()
        self.t2=Entry()
        self.t3=Entry()
        self.t4=Entry()
        self.t5=Entry()
self.t6=Entry()
self.t7=Entry()
self.t8=Entry()
self.t9=Entry()
self.lbl1.place(x=50, y=60)
self.lbl2.place(x=50, y=90)
self.lbl3.place(x=50, y=120)
self.lbl4.place(x=50, y=150)
self.lbl5.place(x=50, y=180)
self.lbl6.place(x=50, y=210)
self.lbl7.place(x=50, y=240)
self.lbl8.place(x=50, y=270)
self.lbl9.place(x=50, y=300)
self.t1.place(x=200, y=60)
self.t2.place(x=200, y=90)
self.t3.place(x=200, y=120)
self.t4.place(x=200, y=150)
self.t5.place(x=200, y=180)
self.t6.place(x=200, y=210)
self.t7.place(x=200, y=240)
self.t8.place(x=200, y=270)
self.t9.place(x=200, y=300)
self.img=Image.open("C:/Users/…/bridge.jpg")
self.img= self.img.resize((430, 250))
self.img = ImageTk.PhotoImage(self.img)
self.labelpic=Label(image = self.img, width=270, height=270)
self.labelpic.place(x=430, y=60)
self.lbl10=Label(text='Prototype bridge', font=('Times', 13))
self.lbl10.place(x=500, y=340)
self.b1=Button(text='Target 1', command=self.add1)
self.b1.place(x=50, y=340)
self.b2=Button(text='Target 2', command=self.add2)
self.b2.place(x=160, y=340)
self.b3=Button(text='Target 3', command=self.add3)
self.b3.place(x=270, y=340)
self.lbl12=Label( text='Pareto optimality', font=('Times', 13))
self.lbl12.place(x=130, y=390)
self.lbl=Label(bg='white', fg='white',anchor=CENTER,font=('Times', 13), width=38, height=14)
self.lbl.place(x=30, y=420)
#self.canvas = Canvas( width=400, height=250, bg="white")
#self.canvas.place(x=10, y=430)
self.lbl13=Label( text='Optimal design parameters of seismic isolation', font=('Times', 13))
self.lbl13.place(x=400, y=390)
self.lbl13=Label( text='Copyright by Bingzhe Zhang')
```

```
            self.lbl13.place(x=620, y=690)
            global photo
        def add1(self):
            num1=float(self.t1.get())
            num2=float(self.t2.get())
            num3=float(self.t3.get())
            num4=float(self.t4.get())
            num5=float(self.t5.get())
            num6=float(self.t6.get())
            num7=float(self.t7.get())
            num8=float(self.t8.get())
            num9=float(self.t9.get())
#ANN model training
            ph = num2
            ps = num3
            bw = num4
            pr = num5
            ss = num6
            sc = num7
            rr = num8
            rs = num9
            pga = num1
            xx1 = [pga, ph, ps, bw, pr, ss, sc, rr, rs]
            xx1 = np.array(xx1)
            class MyProblem(ea.Problem):
                def __init__(self):
                    name = 'MyProblem'
                    M = 2
                    maxormins = [1] * M
                    Dim = 7
                    varTypes = [1,0,1,1,0,0,0]
                    lb = [2,0.05,5,4,400,0.001,5]
                    ub = [5,0.5,10,10,1200,0.003,20]
                    lbin = [1] * Dim
                    ubin = [1] * Dim
                    ea.Problem.__init__(self, name, M, maxormins, Dim, varTypes, lb, ub, lbin, ubin)
                def evalVars(self, Vars):
                    v1 = np.array(Vars).astype(float)
                    v1 = np.insert(v1, 0, rs, axis=1).astype(float)
                    v1 = np.insert(v1, 0, rr, axis=1).astype(float)
                    v1 = np.insert(v1, 0, sc, axis=1).astype(float)
                    v1 = np.insert(v1, 0, ss, axis=1).astype(float)
                    v1 = np.insert(v1, 0, pr, axis=1).astype(float)
                    v1 = np.insert(v1, 0, bw, axis=1).astype(float)
                    v1 = np.insert(v1, 0, ps, axis=1).astype(float)
                    v1 = np.insert(v1, 0, ph, axis=1).astype(float)
                    v1 = np.insert(v1, 0, pga, axis=1).astype(float)
                    f1 = est1.predict(v1).reshape(500,1)
                    f2 = est2.predict(v1).reshape(500,1)
                    ObjV = np.hstack([f1, f2])
                    return ObjV
            problem = MyProblem()
            algorithm = ea.moea_NSGA2_templet(problem,
                                        ea.Population(Encoding='RI', NIND=500),
                                        MAXGEN=500,
                                        logTras=0)
            res = ea.optimize(algorithm, seed=0, verbose=True, drawing=1, outputMsg=True, drawLog=True,
saveFlag=True, dirName='result')
            #result=num1+num2
            #self.t3.insert(END, str(result))
            cairosvg.svg2png(url='C:/Users/…/Pareto Front Plot.svg',
                        write_to='C:/Users/…/Pareto Front Plot.png')
            self.img1=Image.open("C:/Users/…/Pareto Front Plot.png")
            self.img1= self.img1.resize((350, 280))
            self.photo = ImageTk.PhotoImage(self.img1)
            self.puke1 = Label(self.master,image=self.photo)
            self.puke1.place(x=30, y=420)
            ph = num2
            ps = num3
            bw = num4
            pr = num5
            ss = num6
            sc = num7
            rr = num8
            rs = num9
            pga = num1
            opt = pd.read_csv("C:/Users/…/optPop/ObjV.csv",header=None)
            Y1 = opt.values[:,0].astype(float) # Y1
            Y2 = opt.values[:,1].astype(float)    # Y2
            Y1=Y1*-1
            A1 = np.amax(Y1,0)
            print(A1)
            Y2=Y2*-1
            A2 = np.amax(Y2,0)
            print(A2)
            Y1=Y1*-1
            AA1 = np.amax(Y1,0)
            print(AA1)
```

```
Y2=Y2*-1
AA2 = np.amax(Y2,0)
print(AA2)
#Y = ((Y1+A1)/(AA1+A1)*3+(Y2+A2)/(AA2+A2))*-1
YY = ((Y1+A1)/(AA1+A1)+(Y2+A2)/(AA2+A2)*3)*-1
#vYY = ((Y1+A1)/(AA1+A1))**2*-1+((Y2+A2)/(AA2+A2))**2*-1
A = np.amax(YY,0)
#print(A)
df = pd.DataFrame(
    {
        "x": YY
    }
)
col = "x"
max_x = df.idxmax(axis = 0)
print(max_x)
opts = pd.read_csv("C:/Users/…/optPop/Chrom.csv",header=None)
Y1 = opts.values[max_x,:7].astype(float) # Y1
Y1 = Y1.reshape(-1)
print(Y1)
opts1 = pd.read_csv("C:/Users/…/optPop/ObjV.csv",header=None)
Y2 = opts1.values[max_x,:2].astype(float)
Y2 = Y2.reshape(-1)
print(Y2)
Fsp = (0.27*ps*0.06*0.19*bw*2+0.31*bw*2*ps*0.06-2*0.72*ps*0.06*0.241*bw)*ps*2500*10/1000
print(Fsp)
Fspb = Fsp / Y1[0]
A = Fspb / Y1[3] /1000
L = A ** 0.5
H = L / Y1[2]
K = 1200*A/H
self.label11 = Label(self.master)
self.label11['text'] = 'Bearings design:'
self.label11['font']=('Times', 13)
self.label11.place(x=425, y=470)
#self.lbl12=Label( text='Optimized response', font=('Times', 13))
K = format(K,'.2f')
self.label14 = Label(self.master)
self.label14['text'] = 'K=',str(K),'kN/m'
self.label14['font']=('Times', 11)
self.label14.place(x=425, y=500)
self.label15 = Label(self.master)
self.label15['text'] = 'Amount/span:',str(format(Y1[0],'.0f'))
self.label15['font']=('Times', 11)
```

```
        self.label15.place(x=425, y=525)
        self.label16 = Label(self.master)
        self.label16['text'] = 'μ=',str(format(Y1[1],'.2f'))
        self.label16['font']=('Times', 11)
        self.label16.place(x=425, y=550)
        self.label17 = Label(self.master)
        self.label17['text'] = 'Shearkey design:'
        self.label17['font']=('Times', 13)
        self.label17.place(x=585, y=470)
        Y1[6] = format(Y1[6],'.1f')
        self.label18 = Label(self.master)
        self.label18['text'] = 'Gap=',str(Y1[6]),'cm'
        self.label18['font']=('Times', 11)
        self.label18.place(x=585, y=500)
        Y1[4] = format(Y1[4],'.1f')
        self.label19 = Label(self.master)
        self.label19['text'] = 'Strength:',str(format(Y1[4],'.1f')),'MPa'
        self.label19['font']=('Times', 11)
        self.label19.place(x=585, y=525)
        Y1[5] = format(Y1[5],'.5f')
        self.label20 = Label(self.master)
        self.label20['text'] = 'Ductile:',str(format(Y1[5],'.4f')),'m'
        self.label20['font']=('Times', 11)
        self.label20.place(x=585, y=550)
        messagebox.showinfo('Alert', 'Optimization complete')
...
#GUI impoet
root = Tk()
root.geometry("800x720+10+10")
root.title('Multi-objective optimization program for seismic performance of highway bridges')
app = Application(master=root)
root.mainloop()
```