#importing part  
import pandas as pd  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.metrics import confusion\_matrix  
from sklearn.metrics import classification\_report  
from sklearn.metrics import f1\_score  
from sklearn.metrics import accuracy\_score  
  
class BackEnd:  
  
 def load\_dataset(self):  
 # load dataset  
 global dataset  
 dataset= pd.read\_csv('diabetes.csv')  
  
 def remove\_zeros(self):  
 # remove zeros  
 remove\_zero = ['Glucose', 'BloodPressure', 'SkinThickness', 'BMI', 'Insulin' ]  
  
 for column in remove\_zero:  
 dataset[column] = dataset[column].replace(0,np.NaN)  
 mean= int( dataset[column].mean(skipna=True) )  
 dataset[column] = dataset[column].replace(np.NaN, mean)  
  
 def split(self):  
  
 #split dataset  
 self.X = dataset.iloc[:,0:8]  
 self.Y = dataset.iloc[:, 8]  
 self.X\_train, self.X\_test, self.Y\_train, self.Y\_test = train\_test\_split(self.X,self.Y, random\_state=0, test\_size=0.2)  
  
 def Feature\_Scaling(self):  
  
 #Feature scaling  
 global sc\_X  
 sc\_X = StandardScaler()  
 self.X\_train = sc\_X.fit\_transform(self.X\_train)  
 self.X\_test = sc\_X.transform(self.X\_test)  
  
 def model(self):  
 global classifier  
 # define the model : Init K-NN  
 classifier = KNeighborsClassifier(n\_neighbors=42, p=2, metric='euclidean')  
  
 #fit Model  
 classifier.fit(self.X\_train, self.Y\_train)  
  
 # Predict the test set results  
 self.Y\_pred = classifier.predict(self.X\_test)  
  
 #Evaluate the model  
 self.cm = confusion\_matrix(self.Y\_test,self.Y\_pred)  
 self.cr=classification\_report(self.Y\_test,self.Y\_pred)  
  
 def scores(self):  
 print(self.cm)  
 print(self.cr)  
 print("F1 Score : ", f1\_score(self.Y\_test, self.Y\_pred) )  
 print("Accuracy Score : ", accuracy\_score(self.Y\_test, self.Y\_pred) )  
  
 def predictor(b):  
 X\_test1 = b  
 X\_test1 = sc\_X.transform(X\_test1)  
 Y\_pred1 = classifier.predict(X\_test1)  
 return Y\_pred1  
  
 def knn(self):  
 d.load\_dataset()  
 d.remove\_zeros()  
 d.split()  
 d.Feature\_Scaling()  
 d.model()  
 d.scores()  
  
  
d = BackEnd()

from tkinter import \*  
from tkinter import messagebox  
from final import \*  
  
  
class Window(Frame):  
  
 def \_\_init\_\_(self, master=None):  
 Frame.\_\_init\_\_(self, master)  
 self.master = master  
 self.init\_window()  
  
 # Creation of init\_window  
 def init\_window(self):  
 # changing the title of our master widget  
 self.master.title("Diabetes Prediction System")  
  
 # allowing the widget to take the full space of the root window  
 self.pack(fill=BOTH, expand=1)  
  
fields = 'Pregnancies','Glucose','Blood Pressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age'  
  
global a  
global b  
b = []  
a = []  
  
def fetch(entries):  
 for entry in entries:  
 field = entry[0]  
 text = float(entry[1].get())  
 a.append(text)  
 print('%s: "%s"' % (field, text)) # tkinter print with field name  
 b.append(a)  
  
 BackEnd.knn(BackEnd) # calling knn from final  
 res = BackEnd.predictor(b) # calling predictor from final  
 if res == 1:  
 positive()  
 else:  
 negative()  
  
  
def makeform(root, fields):  
 entries = []  
 for i in fields:  
 row = Frame(root)  
 lab = Label(row, width=25, text=i, anchor='w')  
 ent = Entry(row) # get entry  
 row.pack(side=TOP, fill=X, padx=5, pady=5)  
 lab.pack(side=LEFT)  
 ent.pack(side=RIGHT, expand=YES, fill=X)  
 entries.append((i, ent))  
 return entries  
  
  
def positive():  
 messagebox.showinfo(title="Prediction", message="Diabetes is Positive")  
 root.quit()  
  
def negative():  
 messagebox.showinfo(title="Prediction", message="Diabetes is Negative")  
 root.quit()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 root = Tk()  
 ents = makeform(root, fields)  
 root.bind('<Return>', (lambda event, e=ents: fetch(e)))  
 b1 = Button(root, text='Show', command=(lambda e=ents: fetch(e)))  
 b1.pack(side=LEFT, padx=5, pady=5)  
 b2 = Button(root, text='Exit', command=root.quit)  
 b2.pack(side=LEFT, padx=5, pady=5)  
  
app = Window(root)  
root.mainloop()