正常代码

import numpy as np # 导入numpy包并将其命名为np

##定义正向化的函数

def positivization(x,type,i):

# x：需要正向化处理的指标对应的原始向量

# typ：指标类型（1：极小型，2：中间型，3：区间型）

# i：正在处理的是原始矩阵的哪一列

if type == 1: #极小型

print("第",i,"列是极小型，正向化中...")

posit\_x = x.max(0)-x

print("第",i,"列极小型处理完成")

print("--------------------------分隔--------------------------")

return posit\_x

elif type == 2: #中间型

print("第",i,"列是中间型")

best = int(input("请输入最佳值："))

m = (abs(x-best)).max()

posit\_x = 1-abs(x-best)/m

print("第",i,"列中间型处理完成")

print("--------------------------分隔--------------------------")

return posit\_x

elif type == 3: #区间型

print("第",i,"列是区间型")

a,b = [int(l) for l in input("按顺序输入最佳区间的左右界，并用逗号隔开：").split(",")]

m = (np.append(a-x.min(),x.max()-b)).max()

x\_row = x.shape[0] #获取x的行数

posit\_x = np.zeros((x\_row,1),dtype=float)

for r in range(x\_row):

if x[r] < a:

posit\_x[r] = 1-(a-x[r])/m

elif x[r] > b:

posit\_x[r] = 1-(x[r]-b)/m

else:

posit\_x[r] = 1

print("第",i,"列区间型处理完成")

print("--------------------------分隔--------------------------")

return posit\_x.reshape(x\_row)

## 第一步：从外部导入数据

#注：保证表格不包含除数字以外的内容

x\_mat = np.loadtxt('river.csv', encoding='UTF-8-sig', delimiter=',') # 推荐使用csv格式文件

## 第二步：判断是否需要正向化

n, m = x\_mat.shape

print("共有", n, "个评价对象", m, "个评价指标")

judge = int(input("指标是否需要正向化处理，需要请输入1，不需要则输入0："))

if judge == 1:

position = np.array([int(i) for i in input("请输入需要正向化处理的指标所在的列，例如第1、3、4列需要处理，则输入1,3,4").split(',')])

position = position-1

typ = np.array([int(j) for j in input("请按照顺序输入这些列的指标类型（1：极小型，2：中间型，3：区间型）格式同上").split(',')])

for k in range(position.shape[0]):

x\_mat[:, position[k]] = positivization(x\_mat[:, position[k]], typ[k], position[k])

print("正向化后的矩阵：", x\_mat)

## 第三步：对正向化后的矩阵进行标准化

tep\_x1 = (x\_mat \* x\_mat).sum(axis=0) # 每个元素平方后按列相加

tep\_x2 = np.tile(tep\_x1, (n, 1)) # 将矩阵tep\_x1平铺n行

Z = x\_mat / ((tep\_x2) \*\* 0.5) # Z为标准化矩阵

print("标准化后的矩阵为：", Z)

## 第四步：计算与最大值和最小值的距离，并算出得分

tep\_max = Z.max(0) # 得到Z中每列的最大值

tep\_min = Z.min(0) # 每列的最小值

tep\_a = Z - np.tile(tep\_max, (n, 1)) # 将tep\_max向下平铺n行,并与Z中的每个对应元素做差

tep\_i = Z - np.tile(tep\_min, (n, 1)) # 将tep\_max向下平铺n行，并与Z中的每个对应元素做差

D\_P = ((tep\_a \*\* 2).sum(axis=1)) \*\* 0.5 # D+与最大值的距离向量

D\_N = ((tep\_i \*\* 2).sum(axis=1)) \*\* 0.5

S = D\_N / (D\_P + D\_N) # 未归一化的得分

std\_S = S / S.sum(axis=0)

sorted\_S = np.sort(std\_S, axis=0)

print(std\_S) # 打印标准化后的得分

## std\_S.to\_csv(std\_S.csv) 结果输出到std\_S.csv文件

案例：

某一教育评估机构对5个研究生院进行评估。该机构选取了4个评价指标：人均专著、生师比、科研经费、逾期毕业率。采集数据如表所示。



解释：人均专著和科研经费是效益性指标，预期毕业率是成本型指标，生师比是区间型指标，最优范围是[5,6]，最差下限2，最差上限12. 4个指标权重采用专家打分的结果，分别为0.2,0.3,0.4和0.1。

实现代码：

import numpy as np

import pandas as pd

#TOPSIS方法函数

def Topsis(A1):

W0=[0.2,0.3,0.4,0.1] #权重矩阵

W=np.ones([A1.shape[1],A1.shape[1]],float)

for i in range(len(W)):

for j in range(len(W)):

if i==j:

W[i,j]=W0[j]

else:

W[i,j]=0

Z=np.ones([A1.shape[0],A1.shape[1]],float)

Z=np.dot(A1,W) #加权矩阵

#计算正、负理想解

Zmax=np.ones([1,A1.shape[1]],float)

Zmin=np.ones([1,A1.shape[1]],float)

for j in range(A1.shape[1]):

if j==3:

Zmax[0,j]=min(Z[:,j])

Zmin[0,j]=max(Z[:,j])

else:

Zmax[0,j]=max(Z[:,j])

Zmin[0,j]=min(Z[:,j])

#计算各个方案的相对贴近度C

C=[]

for i in range(A1.shape[0]):

Smax=np.sqrt(np.sum(np.square(Z[i,:]-Zmax[0,:])))

Smin=np.sqrt(np.sum(np.square(Z[i,:]-Zmin[0,:])))

C.append(Smin/(Smax+Smin))

C=pd.DataFrame(C,index=['院校' + i for i in list('12345')])

return C

#标准化处理

def standard(A):

#效益型指标

A1=np.ones([A.shape[0],A.shape[1]],float)

for i in range(A.shape[1]):

if i==0 or i==2:

if max(A[:,i])==min(A[:,i]):

A1[:,i]=1

else:

for j in range(A.shape[0]):

A1[j,i]=(A[j,i]-min(A[:,i]))/(max(A[:,i])-min(A[:,i]))

#成本型指标

elif i==3:

if max(A[:,i])==min(A[:,i]):

A1[:,i]=1

else:

for j in range(A.shape[0]):

A1[j,i]=(max(A[:,i])-A[j,i])/(max(A[:,i])-min(A[:,i]))

#区间型指标

else:

a,b,lb,ub=5,6,2,12

for j in range(A.shape[0]):

if lb <= A[j,i] < a:

A1[j,i]=(A[j,i]-lb)/(a-lb)

elif a <= A[j,i] < b:

A1[j,i]=1

elif b <= A[j,i] <= ub:

A1[j,i]=(ub-A[j,i])/(ub-b)

else: #A[i,:]< lb or A[i,:]>ub

A1[j,i]=0

return A1

#读取初始矩阵并计算

def data(file\_path):

data=pd.read\_excel(file\_path).values

A=data[:,1:]

A=np.array(A)

#m,n=A.shape[0],A.shape[1] #m表示行数,n表示列数

return A

#权重

A=data('研究生院评估数据.xlsx')

A1=standard(A)

C=Topsis(A1)

print(C)