python进行因子分析

版权

一、基本思想

通过一个例子说明:

假设一个同学数学、物理、化学、生物都考了满分，那么可以认为这个同学的理性思维比较强。此时，我们所说的**理性思维**就是一个因子，在这个因子的作用下，偏理科的成绩才会这么高。

什么是 因子分析 ？就是假设现有全部自变量x的出现是因为某个潜在变量的作用，这个潜在变量就是所谓的因子。在这个因子的作用下，x能够被察觉到。

因子分析就是将存在某些相关性的变量提炼为较少的几个因子，用这几个因子去表示原本的变量，也可以根据因子对变量进行分类。

因子分析本质上也是**降维**的过程，和**主成分分析（PCA）**算法类似。

二、用途

因子分析和PCA有很多类似之处。因子分析主要用来描述一组测量到的变量中的一些更基本、但又无法直接测量到的隐形变量。因子分析法也可以用来综合评价。

主要思路是利用研究指标之间存在的一定的相关性，从而推理出是否存在某些潜在的共性因子，而这些共性因子在不同程度上共同影响着研究指标。因子分析可以在许多变量中找出隐藏的具有代表性的因子，将共同本质的变量归入一个因子，可以减少变量的数目。

三、相关概念

①因子荷载：每个原始变量和每个因子之间的相关系数，反映了变量相对于因子的重要性。

②公因子方差（变量共同度）：每个变量所包含的信息能够被因子所解释的程度。所有变量的共同度都在60%以上，可以认为所提取的因子对各个变量的解释能力可以接受。

③因子旋转：因子分析的结果需要每个因子都有实际意义，在原始变量和因子之间的相关系数可能无法明显的表示出因子的含义，可以对因子荷载进行旋转。（当有多个因子的时候，因子荷载就构成一个矩阵，成为因子荷载矩阵）。

④因子得分：用来评价每个案例在每个因子上的分值，可以用于替代原始变量进行其他统计分析，也可以看成是降维后的结果。

⑤最大方差法：能够使每个变量尽可能在一个因子上有较高荷载，在其余因子的荷载较小。

⑥巴特利特和KMO检验：使用KMO取样适切性量数（KMO检验统计量）对比变量间简单相关系数和偏相关系数的指标。Kaiser给出的KMO度量标准：0.5一下表示极不适合，0.6表示不太合适，0.7表示一般，0.8表示适合，0.9以上表示非常合适。

⑦总方差解释：通过分析所提取的因子数量，以及所提取的因子对变量的累计方差贡献值。累计方差贡献值达到60%及以上，因子对变量的解释能力可接受；达到80%及以上，说明因子对变量的解释能力极好。

三、因子分析实现步骤:

---

①对数据样本进行标准化处理

②计算样本的相关矩阵R

③求相关矩阵R的特征值、特征向量

④根据系统要求的贡献度确定主因子个数

⑤计算因子荷载矩阵A

⑥确定因子模型

四、实例详解

用python进行因子分析主要用到:

factor_analyzer.analyze（重点）

factor_analyzer.factor_analyzer

1.导入库

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib as mpl
from factor_analyzer import FactorAnalyzer
from factor_analyzer.factor_analyzer import calculate_kmo
from factor_analyzer.factor_analyzer import calculate_bartlett_sphericity
import scipy.cluster.hierarchy as shc
```

2.读取数据

```
#读取数据
df = pd.read_excel("数据.xls", index_col=0).reset_index(drop=True)
df = df.astype('float32')
print(df)
#去除空值
df.dropna(inplace=True)
#查询数据缺失值情况
print(df.isnull().sum())
```

输出:

	Hg(ng/m3)	Rn(Bq/m3)	H2(ppm)	CH4(ppm)	CO2(ppm)	H2S(ppb)
0	5.967000	6297.152832	77.333000	1.193	8097.663086	0.338
1	5.328000	3603.061035	7.900000	0.307	4516.178223	0.384
2	23.483999	8843.578125	12.522000	0.429	5224.597168	0.389
3	4.040000	15265.429688	43.351002	1.396	8854.173828	0.409
4	8.826000	7859.134766	21.313999	0.824	3428.552979	0.447
...	...	...	...	...	...	...

数据缺失值情况:

```
[4063 rows x 6 columns]
Hg(ng/m3)    0
Rn(Bq/m3)    0
H2(ppm)      0
CH4(ppm)     0
CO2(ppm)     0
H2S(ppb)     0
dtype: int64
```


3.充分性检测

在进行因子分析前，需要先进行充分性检测，检验相关特征阵中各个变量间的相关性，是否为单位矩阵，即检验各个变量是否各自独立。

3.1 Bartlett's球状检验

检验总体变量的相关矩阵是否是单位矩阵（相关系数矩阵对角线的所有元素皆为1，所有非对角线上的元素均为零），即检验各个变量是否独立。

如果不是单位矩阵，说明原变量之间存在相关性，可以进行因子分析；反之，原变量之间没有相关性，不适合进行主成分分析。

```
#Bartlett's球状检验
#检验总体变量的相关矩阵是否是单位矩阵（相关系数矩阵对角线的所有元素均为1，所有 非对角线上的元素均为零）；检验各个变量是
#如果不是单位矩阵，说明原变量之间存在相关性，可以进行因子分析；反之，原变量之间不存在相关性，数据不适合进行主成分分析
chi_square_value, p_value = calculate_bartlett_sphericity(df)
print("bartlett球状检验参数：\n卡方值为：{}，p值为：{}".format(chi_square_value, p_value))
```

输出：

卡方值为：1574.0350099858335，p值为：0.0

3.2 KMO检验

检验变量间的相关性和偏相关性，取值在0~1之间。KMO统计量越接近1，变量间相关性越强，偏相关性越弱，因子分析效果越佳。通常取值从0.6开始进行因子分析。

```
#KMO检验
#检验变量间相关性和偏相关性，取值在0~1之间；KOM统计量越接近1，变量相关性越强，偏相关性越弱，因子分析效果越好
#通常取值从0.6开始进行因子分析
kmo_all, kmo_model = calculate_kmo(df)
print("KMO检验参数：\n", kmo_model)
```

4.选择因子个数

方法：计算相关矩阵的特征值，进行降序排列。

4.1特征值和特征向量

```
#构建因子分析模型
fa = FactorAnalyzer(8, rotation=None)
#训练模型
fa.fit(df)

#得到特征值featValue、特征向量featVec
featValue, featVec = fa.get_eigenvalues()
print("特征值: \n{}\n特征向量: \n{}".format(featValue, featVec))
```

输出：

```
特征值:
[1.64217163 1.06181242 0.97023108 0.96351187 0.89944124 0.46283181]
特征向量:
[ 1.15174996e+00  1.61266940e-01  1.17284055e-01  4.38624595e-02
  1.78805473e-02 -6.15886880e-07]
```


4.2可视化

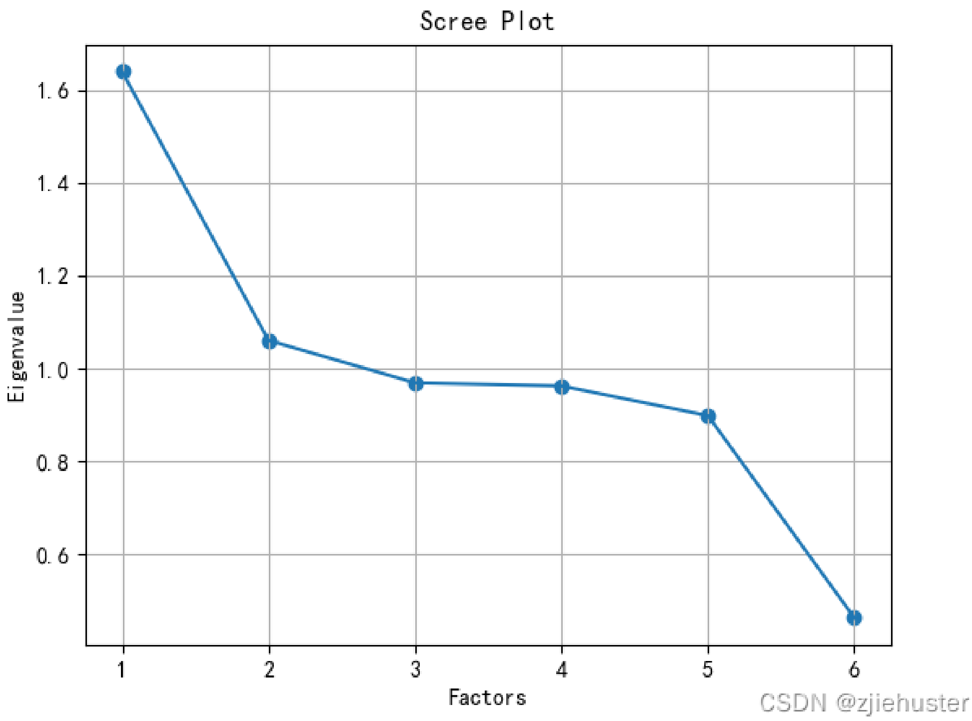
将特征值和因子个数的变化绘制成图形。

```
#同样的数据绘制散点图和折线图
plt.scatter(range(1, df.shape[1] + 1), featValue)
plt.plot(range(1, df.shape[1] + 1), featValue)

plt.title("Scree Plot")
plt.xlabel("Factors")
plt.ylabel("Eigenvalue")

mpl.rcParams['font.sans-serif'] = ['SimHei']  # 指定默认字体
mpl.rcParams['axes.unicode_minus'] = False  # 解决保存图像是负号'-'显示为方块的问题
plt.grid()  # 显示网格
plt.show()  # 显示图形
```

输出：



有图可见，选择3~5个因子即可。

5.因子旋转

5.1建立因子分析模型

采用方差最大化因子旋转方式。

```
#因子旋转
#选择方式：varimax 方差最大化
#选择固定因子为4
fa_four = FactorAnalyzer(4, rotation='varimax')
fa_four.fit(df)
```

5.2查看每个变量的公因子方差、旋转后的特征值、成分矩阵和因子方差

```
#查看每个变量的公因子方差数据
pd.DataFrame(fa_four.get_communalities(), index=df.columns)
print("每个变量的公因子方差数据:\n", pd.DataFrame(fa_four.get_communalities(), index=df.columns))

#查看旋转后的特征值
pd.DataFrame(fa_four.get_eigenvalues())
print("旋转后的特征值:\n", pd.DataFrame(fa_four.get_eigenvalues()))

#查看成分矩阵
#变量个数*因子个数
```

输出：

```
旋转后的特征值:
          0         1         2         3         4         5
0  1.642172  1.061812  0.970231  0.963512  0.899441  0.462832
1  1.141669  0.216925  0.155267  0.048182  0.000002 -0.000006
成分矩阵:
                  0         1         2         3
Hg(ng/m3) -0.006255 -0.009968  0.056096  0.258442
Rn(Bq/m3)  0.045483 -0.036610  0.359210  0.121793
H2(ppm)    0.090386  0.458749 -0.016013 -0.017190
CH4(ppm)   0.706120  0.088725  0.086616  0.002683
CO2(ppm)   0.111462  0.118098  0.142795 -0.012351
H2S(ppb)   0.723822  0.162763  0.117717 -0.018625
因子方差:
 (array([1.04522497, 0.26020186, 0.17418516, 0.08242763]), array([0.17420416, 0.04336698, 0.02903086, 0.01373794]), array([0.17420416, 0.21757114, 0.246602  , 0.26033993]))
```

5.3隐藏变量可视化

为了更直观的观察每个隐藏变量和哪些特征的关系比较大，进行可视化展示，为了方便取上面相关系数绝对值。然后利用热力图将稀疏矩阵绘制出来。

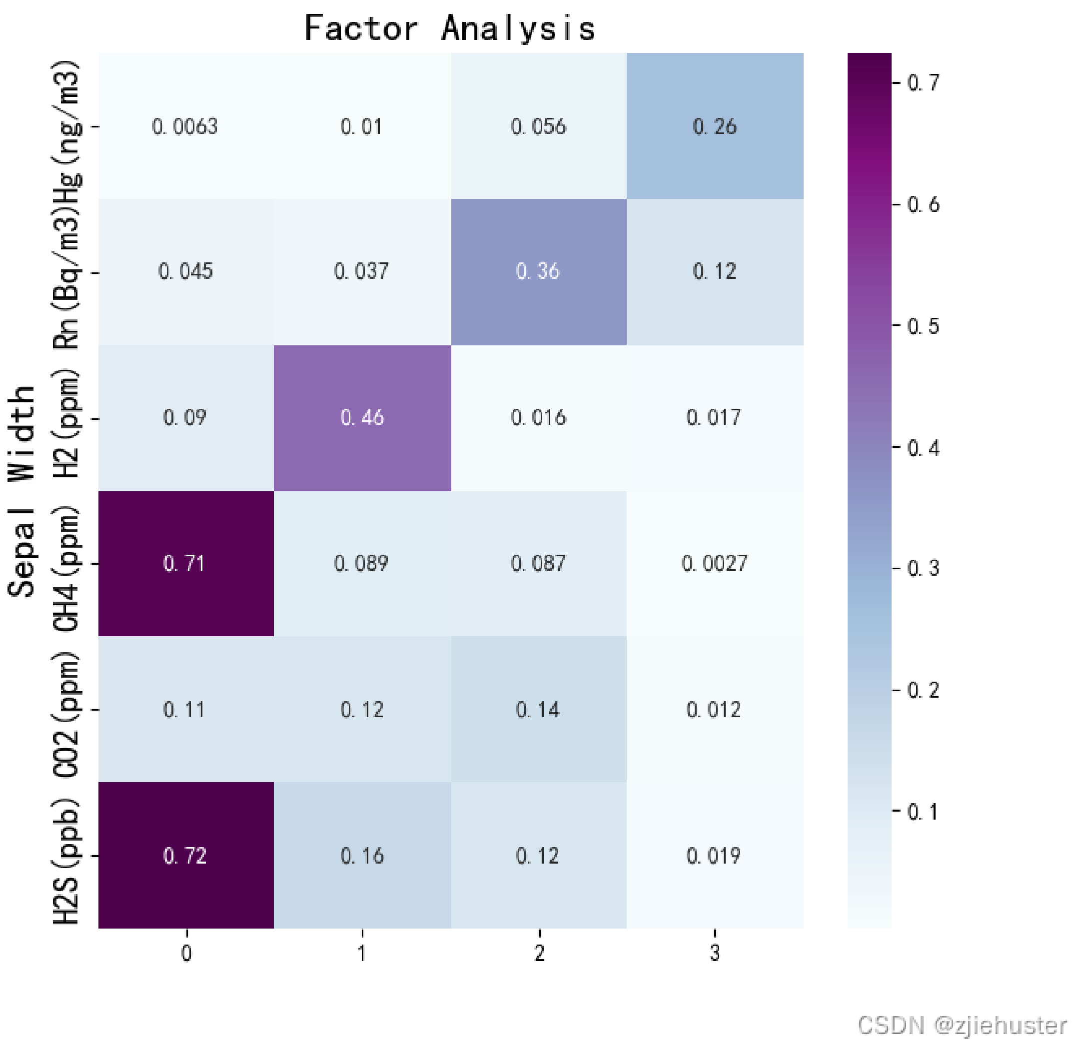
```
#隐藏变量可视化
df1 = pd.DataFrame(np.abs(fa_two.loadings_), index=df.columns)
print("隐藏变量可视化：\n", df1)

#绘图
plt.figure(figsize=(7, 7))
ax = sns.heatmap(df1, annot=True, cmap="BuPu")
#设置y轴字体大小
ax.yaxis.set_tick_params(labelsize=15)
plt.title("Factor Analysis", fontsize="xx-large")
# 设置y轴标签
plt.ylabel("Sepal Width", fontsize="xx-large")
# 显示图片
plt.show()
```

输出:

```
隐藏变量可视化:
                  0         1         2         3
Hg(ng/m3)  0.006255  0.009968  0.056096  0.258442
Rn(Bq/m3)  0.045483  0.036610  0.359210  0.121793
H2(ppm)    0.090386  0.458749  0.016013  0.017190
CH4(ppm)   0.706120  0.088725  0.086616  0.002683
CO2(ppm)   0.111462  0.118098  0.142795  0.012351
H2S(ppb)   0.723822  0.162763  0.117717  0.018625
```


热力图:



5.4转成新变量

由于采用较为合适的4个因子，可以将原始数据转换成4个新的特征，转换方式如下：

```
#由于采用较为合适的4个因子，可以将原始数据转换成4个新的特征
df2 = pd.DataFrame(fa_four.transform(df))
print("转换后数据：\n", df2)
```

输出：

```
转换后数据：
            0         1         2         3
0   -0.062173  0.457414  0.170544 -0.042968
1   -0.043846 -0.105713 -0.105476 -0.065398
2   -0.056594 -0.086865  0.236039  0.103637
3   -0.065842  0.192393  0.733331  0.114597
4   -0.061169 -0.072885  0.090915  0.032510
...       ...       ...       ...       ...
```