因子分析——python

原创 洋洋菜鸟 于 2022-01-18 20:29:49 发布 11418 收藏 236 版权

分类专栏: 机器学习 数学建模 文章标签: python 开发语言 后端

目录

一、起源

因子分析 的**起源**是这样的：1904年英国的一个心理学家发现学生的英语、法语和古典语成绩非常有相关性，他认为这三门课程背后有一个共同的因素驱动，最后将这个因素定义为“语言能力”。

基于这个想法，发现很多相关性很高的因素背后有**共同的因子驱动**，从而定义了**因子分析**，这便是因子分析的由来。

二、基本思想

我们再通过一个更加实际的例子来理解因子分析的基本思想：

现在假设一个同学的数学、物理、化学、生物都考了满分，那么我们可以认为这个学生的理性思维比较强，在这里**理性思维就**是我们所说的一个因子。在这个因子的作用下，偏理科的成绩才会那么高。

到底什么是因子分析？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，这个潜在的变量就是我们说的因子。在这个因子的作用下，x能够被观察到。

> 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。**

因子分子本质上也是**降维**的过程，和**主成分分析（PCA）**算法比较类似。

三、算法用途

因子分析法和主成分分析法有很多类似之处。因子分析的主要目的是用来描述隐藏在一组测量到的变量中的一些更基本的，但又无法直接测量到的隐性变量。因子分析法也可以用来综合评价。

其主要思路是利用研究指标的之间存在一定的相关性，从而推想是否存在某些潜在的共性因子，而这些不同的潜在的共性因子不同程度地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。

四、因子分析步骤

应用因子分析法的主要步骤如下：

- 对所给的数据样本进行**标准化处理**
- 计算样本的**相关矩阵R**
- 求相关矩阵R的**特征值、特征向量**
- 根据系统要求的**累积贡献度**确定主因子的个数

- 计算因子**载荷矩阵A**
- 最终确定因子模型

五、factor_analyzer库

利用Python进行因子分析的核心库是：factor_analyzer

```
pip install factor_analyzer
```

这个库主要有两个主要的模块需要学习：

- factor_analyzer.analyze（重点）
- factor_analyzer.factor_analyzer

官网学习地址：factor_analyzer package — factor_analyzer 0.3.1 documentation

四、实例详解

城市	食品烟酒支出	衣着支出	居住支出	生活用品及服务支出	交通通信支出	教育文化娱乐支出	医疗保健支出	其他用品及服务支出
北京	8070.4	2643	12128	2511	5077.9	4054.7	2629.8	1140.6
天津	8679.6	2114	6187.3	1663.8	3991.9	2643.6	2172.2	892.2
河北	4991.6	1614.4	4483.2	1351.1	2664.1	1991.3	1549.9	460.4
山西	3862.8	1603	3633.8	951.6	2401	2439	1651.6	450.1
内蒙古	6445.8	2543.3	4006.1	1565.1	3045.2	2598.9	1840.2	699.9
辽宁	6901.6	2321.3	4632.8	1558.2	3447	3018.5	2313.6	802.8
吉林	4975.7	1819	3612	1107.1	2691	2367.5	2059.2	534.9
黑龙江	5019.3	1804.4	3352.4	1018.9	2462.9	2011.5	2007.5	468.3
上海	10014.8	1834.8	13216	1868.2	4447.5	4533.5	2839.9	1102.1
江苏	7389.2	1809.5	6140.6	1616.2	3952.4	3163.9	1624.5	736.6
浙江	8467.3	1903.9	7385.4	1420.7	5100.9	3452.3	1691.9	645.3
安徽	6381.7	1491	3931.2	1118.4	2748.4	2233.3	1269.3	432.9
福建	8299.6	1443.5	6530.5	1393.4	3205.7	2461.5	1178.5	492.8
江西	5667.5	1472.2	3915.9	1028.6	2310.6	1963.9	887.4	449.6
山东	5929.4	1977.7	4473.1	1576.5	3002.5	2399.3	1610	526.9
河南	5067.7	1746.6	3753.4	1430.2	1993.8	2078.8	1524.5	492.8
湖北	6294.3	1557.4	4176.7	1163.8	2391.9	2228.4	1792	435.6
湖南	6407.7	1666.4	3918.7	1384.1	2837.1	3406.1	1362.6	437.4
广东	9421.6	1583.4	6410.4	1721.9	4198.1	3103.4	1304.5	870.1
广西	5937.2	886.3	3784.3	1032.8	2259.8	2003	1065.9	299.3
海南	7419.7	859.6	3527.7	954	2582.3	1931.3	1399.8	341
重庆	6883.9	1939.2	3801.1	1466	2573.9	2232.4	1700	434.4
四川	7118.4	1767.5	3756.5	1311.1	2697.6	2008.4	1423.4	577.1
贵州	6010.3	1525.4	3793.1	1270.2	2684.4	2493.5	1050.1	374.6
云南	5528.2	1195.5	3814.4	1135.1	2791.2	2217	1526.7	414.3
西藏	8727.8	1812.5	3614.5	983	2198.4	922.5	585.3	596.5
陕西	5422	1542.2	3681.5	1367.7	2455.7	2474	2016.7	409
甘肃	5777.3	1776.9	3752.6	1329.1	2517.9	2322.1	1583.4	479.9
青海	5975.7	1963.5	3809.4	1322.1	3064.3	2352.9	1750.4	614.9
宁夏	4889.2	1726.7	3770.5	1245.1	3896.5	2415.7	1874	546.6
新疆	6179.4	1966.1	3543.9	1543.8	3074.1	2404.9	1934.8	581.5

数据来源于中国统计年鉴。

1.导入库

```
# 数据处理
```

```
import pandas as pd
import numpy as np

# 绘图
import seaborn as sns
import matplotlib.pyplot as plt
# 因子分析
from factor_analyzer import FactorAnalyzer
```

2.读取数据

```
df = pd.read_csv("D:\桌面\demo.csv",encoding='gbk')
df
```

输出：

:

	城市	食品烟酒支出	衣着支出	居住支出	生活用品及服务支出	交通通信支出	教育文化娱乐支出	医疗保健支出	其他用品及服务支出
0	北京	8070.4	2643.0	12128.0	2511.0	5077.9	4054.7	2629.8	1140.6
1	天津	8679.6	2114.0	6187.3	1663.8	3991.9	2643.6	2172.2	892.2
2	河北	4991.6	1614.4	4483.2	1351.1	2664.1	1991.3	1549.9	460.4
3	山西	3862.8	1603.0	3633.8	951.6	2401.0	2439.0	1651.6	450.1
4	内蒙古	6445.8	2543.3	4006.1	1565.1	3045.2	2598.9	1840.2	699.9
5	辽宁	6901.6	2321.3	4632.8	1558.2	3447.0	3018.5	2313.6	802.8
6	吉林	4975.7	1819.0	3612.0	1107.1	2691.0	2367.5	2059.2	534.9
7	黑龙江	5019.3	1804.4	3352.4	1018.9	2462.9	2011.5	2007.5	468.3
8	上海	10014.8	1834.8	13216.0	1868.2	4447.5	4533.5	2839.9	1102.1
9	江苏	7389.2	1809.5	6140.6	1616.2	3952.4	3163.9	1624.5	736.6
10	浙江	8467.3	1903.9	7385.4	1420.7	5100.9	3452.3	1991.9	645.8

如果不想要城市那一列的话，可以在读取的时候就删除，也可以后面再删

比如，读取时删除

```
df = pd.read_csv("D:\桌面\demo.csv", index_col=0,encoding='gbk').reset_index(drop=True)
df
```

返回：

	食品烟酒支出	衣着支出	居住支出	生活用品及服务支出	交通通信支出	教育文化娱乐支出	医疗保健支出	其他用品及服务支出
0	8070.4	2643.0	12128.0	2511.0	5077.9	4054.7	2629.8	1140.6
1	8679.6	2114.0	6187.3	1663.8	3991.9	2643.6	2172.2	892.2
2	4991.6	1614.4	4483.2	1351.1	2664.1	1991.3	1549.9	460.4
3	3862.8	1603.0	3633.8	951.6	2401.0	2439.0	1651.6	450.1
4	6445.8	2543.3	4006.1	1565.1	3045.2	2598.9	1840.2	699.9

然后我们查询一下，**数据的缺失值情况：**

```
df.isnull().sum()
```

返回：

```
In  [6]: df.isnull().sum()
Out[6]: 食品烟酒支出        0
        衣着支出          0
        居住支出          0
        生活用品及服务支出     0
        交通通信支出        0
        教育文化娱乐支出      0
        医疗保健支出        0
        其他用品及服务支出     0
        dtype: int64
```


然后，我们可以针对的，对数据进行一次处理：

比如**删除无效字段的那一列**

```
#  去掉无效字段
df.drop(["变量名1","变量名2","变量名3"],axis=1,inplace=True)
```

或者，**删除空值**

```
# 去掉空值
df.dropna(inplace=True)
```

3.充分性检测

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

如果不是单位矩阵，说明原变量之间存在相关性，可以进行因子分子；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

```
from factor_analyzer.factor_analyzer import calculate_bartlett_sphericity

chi_square_value, p_value = calculate_bartlett_sphericity(df)
chi_square_value, p_value
```

返回：

```
chi_square_value, p_value
```

```
7]: (223.15491670092194, 2.630088412071237e-32)
```


3.2 KMO检验

检查变量间的相关性和偏相关性，取值在0-1之间；KOM统计量越接近1，变量间的相关性越强，偏相关性越弱，因子分析的效果越好。

通常取值从**0.6**开始进行因子分析

```
#KMO检验
from factor_analyzer.factor_analyzer import calculate_kmo
kmo_all,kmo_model=calculate_kmo(df)
kmo_model
```

返回：

```
kmo_model
```

```
ıt[10]: 0.8260805318560475
```


通过结果可以看到**KMO大于0.6**，也说明变量之间存在相关性，可以进行分析。

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

```
faa = FactorAnalyzer(25,rotation=None)
faa.fit(df)

# 得到特征值ev、特征向量v
ev,v=faa.get_eigenvalues()
print(ev,v)
```

返回：

```
print(ev,v)
```

```
Out[12]: (array([5.39832425, 1.11084719, 0.65029795, 0.28618912, 0.23203656,
        0.13168798, 0.12156876, 0.06904819]),
 array([ 5.29229838e+00,  9.31002303e-01,  5.15848966e-01,  1.54968772e-01,
         1.11950615e-01,  1.50684757e-02,  2.03514953e-03, -1.01104436e-05]))
```

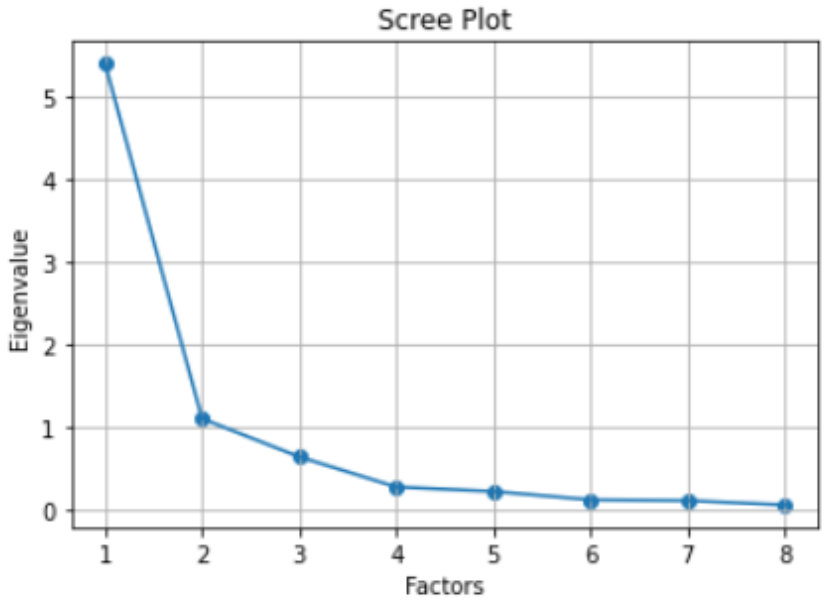

4.2 可视化 展示

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), ev)
plt.plot(range(1, df.shape[1] + 1), ev)

# 显示图的标题和xy轴的名字
# 最好使用英文，中文可能乱码
plt.title("Scree Plot")
plt.xlabel("Factors")
plt.ylabel("Eigenvalue")

plt.grid()  # 显示网格
plt.show()  # 显示图形
```

返回：

```
plt.show()  # 显示图形
```



从上面的图形中，我们明确地看到：选择2或3个因子就可以了

4.3 可视化中显示中文不报错

只需要在画图前，再导入一个库即可，见代码

```
import matplotlib as mpl

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
```

5.因子旋转

5.1 建立因子分析模型

在这里选择，最大方差化因子旋转

```
# 选择方式： varimax 方差最大化
# 选择固定因子为 2 个
faa_two = FactorAnalyzer(2,rotation='varimax')
faa_two.fit(df)
```

返回：

```
faa_two.fit(df)
```

```
Out[16]: FactorAnalyzer(n_factors=2, rotation='varimax', rotation_kwargs={})
```

ratation参数的其他取值情况：

> - varimax (orthogonal rotation)
> - promax (oblique rotation)
> - oblimin (oblique rotation)
> - oblimax (orthogonal rotation)
> - quartimin (oblique rotation)
> - quartimax (orthogonal rotation)
> - equamax (orthogonal rotation)

5.2 查看因子方差-get_communalities()

查看公因子方差

```
# 公因子方差
faa_two.get_communalities()
```

返回：

```
faa_two.get_communalities()
```

```
ut[17]: array([0.77646743, 0.50345232, 0.8366698 , 0.80731788, 0.77272118,
       0.70674152, 0.69364382, 0.88107036])
```


查看每个变量的公因子方差数据

```
pd.DataFrame(faa_two.get_communalities(),index=df.columns)
```

返回：

```
In [18]: pd.DataFrame(faa_two.get_communalities(),index=df.columns)
```

Out[18]:

	0
食品烟酒支出	0.776467
衣着支出	0.503452
居住支出	0.836670
生活用品及服务支出	0.807318
交通通信支出	0.772721
教育文化娱乐支出	0.706742
医疗保健支出	0.693644
其他用品及服务支出	0.881070

5.3 查看旋转后的特征值

```
faa_two.get_eigenvalues()
```

返回：

```
In [19]: faa_two.get_eigenvalues()
Out[19]: (array([5.39832425, 1.11084719, 0.65029795, 0.28618912, 0.23203656,
                 0.13168798, 0.12156876, 0.06904819]),
          array([ 5.16782766,  0.81025609,  0.32606649,  0.04999798,  0.02358588,
                 -0.0707588 , -0.13177043, -0.19712055]))
```



```
pd.DataFrame(faa_two.get_eigenvalues())
```

返回：

```
[21]: pd.DataFrame(faa_two.get_eigenvalues())
```

ut[21]:

	0	1	2	3	4	5	6	7
0	5.398324	1.110847	0.650298	0.286189	0.232037	0.131688	0.121569	0.069048
1	5.167828	0.810256	0.326066	0.049998	0.023586	-0.070759	-0.131770	-0.197121

5.4 查看成分矩阵

查看它们构成的成分矩阵：

```
# 变量个数*因子个数
faa_two.loadings_
```

返回：

```
faa_two.loadings_

Out[22]: array([[0.0275602 , 0.88074279],
                [0.69202166, 0.156711  ],
                [0.48996058, 0.77240432],
                [0.71921422, 0.53856178],
                [0.53504478, 0.69745843],
                [0.62690368, 0.560119  ],
                [0.82100477, 0.13998209],
                [0.6715503 , 0.6558129 ]])
```


如果转成DataFrame格式，index就是我们的变量，columns就是指定的因子factor。转DataFrame格式后的数据：

```
pd.DataFrame(faa_two.loadings_,index=df.columns)
```

返回：

```
In [23]: pd.DataFrame(faa_two.loadings_,index=df.columns)

Out[23]:
```

	0	1
食品烟酒支出	0.027560	0.880743
衣着支出	0.692022	0.156711
居住支出	0.489961	0.772404
生活用品及服务支出	0.719214	0.538562
交通通信支出	0.535045	0.697458
教育文化娱乐支出	0.626904	0.560119
医疗保健支出	0.821005	0.139982
其他用品及服务支出	0.671550	0.655813

5.5 查看因子贡献率

通过理论部分的解释，我们发现每个因子都对变量有一定的贡献，存在某个贡献度的值，在这里查看3个和贡献度相关的指标：

- 总方差贡献：variance (numpy array) – The factor variances
- 方差贡献率：proportional_variance (numpy array) – The proportional factor variances
- 累积方差贡献率：cumulative_variances (numpy array) – The cumulative factor variances

我们来看一下总方差贡献吧

```
faa_two.get_factor_variance()
```

返回：

```
In [24]: faa_two.get_factor_variance()
Out[24]: (array([3.0412938 , 2.93679052]),
          array([0.38016172, 0.36709882]),
          array([0.38016172, 0.74726054]))
```


6.隐藏变量可视化

为了更直观地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的绝对值：

```
df1 = pd.DataFrame(np.abs(faa_two.loadings_),index=df.columns)
print(df1)
```

返回：

```
print(df1)
```

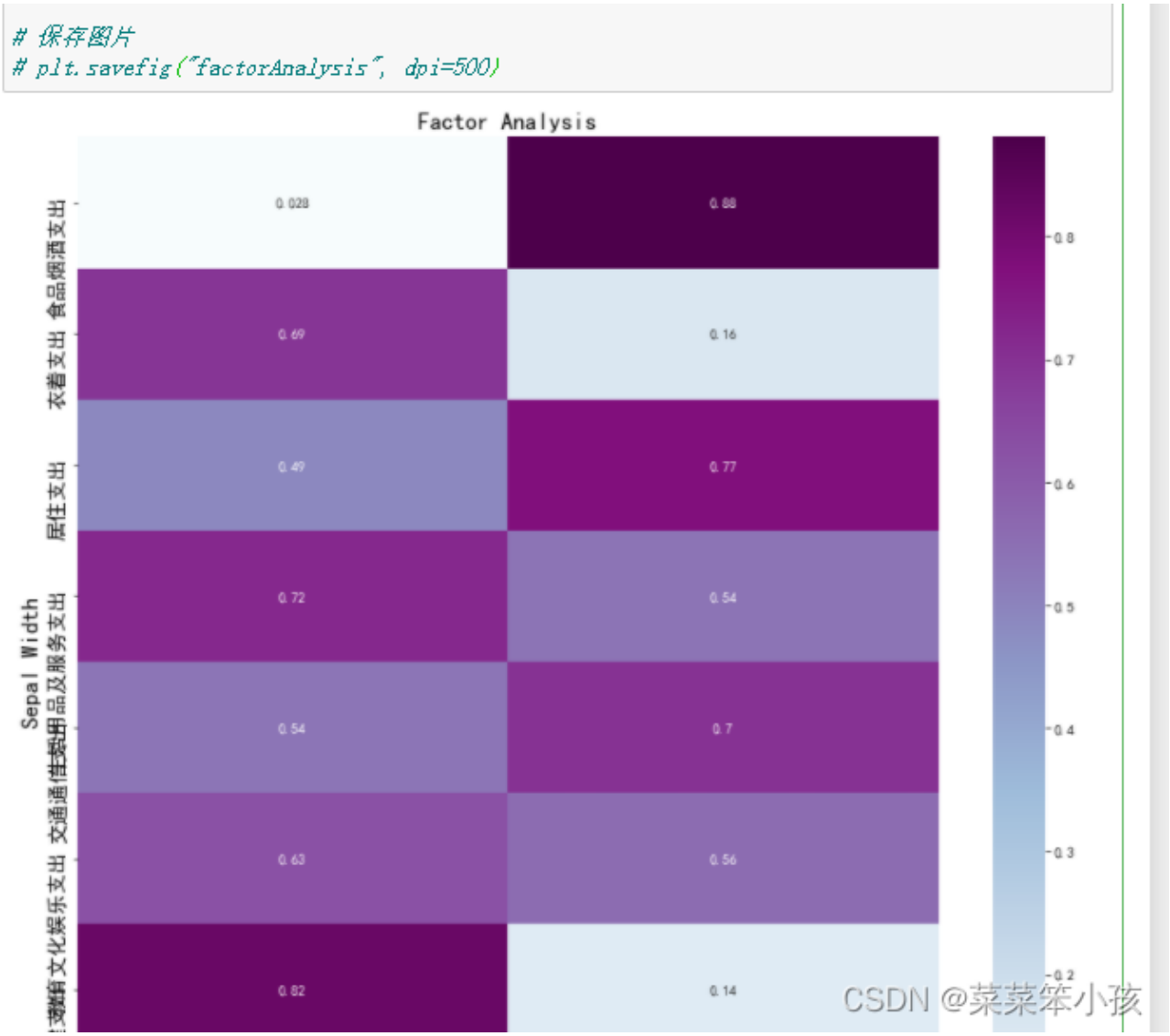
	0	1
食品烟酒支出	0.027560	0.880743
衣着支出	0.692022	0.156711
居住支出	0.489961	0.772404
生活用品及服务支出	0.719214	0.538562
交通通信支出	0.535045	0.697458
教育文化娱乐支出	0.626904	0.560119
医疗保健支出	0.821005	0.139982
其他用品及服务支出	0.671550	0.655813

然后我们通过热力图将系数矩阵绘制出来：

```
# 绘图

plt.figure(figsize = (14,14))
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()

# 保存图片
# plt.savefig("factorAnalysis", dpi=500)
```

返回：



7.转成新变量

上面我们已经知道了2个因子比较合适，可以将原始数据转成2个新的特征，具体转换方式为：

```
faa_two.transform(df)
```

返回：

```
In [28]: faa_two.transform(df)

Out[28]: array([[ 2.7113469 ,  1.66806919],
                [ 0.84317185,  0.88715189],
                [ 0.0800207 , -0.69966236],
                [ 0.14718721, -1.26351123],
                [ 0.80602815, -0.38991662],
                [ 1.32865992, -0.20969206],
                [ 0.49057761, -1.04562994],
                [ 0.10107403, -1.10090616],
                [ 1.31491446,  2.5416871 ],
                [ 0.48281612,  0.72021444],
                [-0.40867177,  1.64122238],
                [-0.85430375, -0.07714331],
                [-1.27085047,  1.20652856],
                [-1.01187755, -0.24960257],
                [ 0.31456746, -0.36610677],
                [ 0.28071457, -0.97800372]
```


转成DataFrame格式后数据展示效果更好：

```
df2 = pd.DataFrame(faa_two.transform(df))
print(df2)
```

返回：

```
In [29]: print(df2)

Out[29]:
```

	0	1
0	2.711347	1.668069
1	0.843172	0.887152
2	0.080021	-0.699662
3	0.147187	-1.263511
4	0.806028	-0.389917

五•、参考资料

1、Factor Analysis：Factor Analysis with Python — DataSklr

2、多因子分析：因子分析(factor analysis)例子–Python | 文艺数学君

3、factor_analyzer package的官网使用手册：factor_analyzer package — factor_analyzer 0.3.1 documentation

4、浅谈主成分分析和因子分析：浅谈主成分分析与因子分析 - 知乎