10182100359-郑佳辰-第七周实验练习题

2021 年 4 月 15 日

1 Week7 Multicollinearity

1.1 背景描述

资产评估：科学的大众评估是将线性回归方法应用于资产评估问题的一种技术。科学大规模评估的目的是根据选定的建筑物的物理特性和对建筑物支付的税费 (地方、学校、县) 预测房屋的销售价格。这些数据最初是由 Narula 和 Wellington(1977) 提出的，我们观察 24 个观测值。

由此我们构造了 24 个观测的 9 个变量，具体请见下表：

1.2 数据描述

变量名	变量含义	变量类型	变量取值范围
(自变量) X1	数千美元税收 (当地，县，学校)	continuous variable	$\mathbb{R}^+$
(自变量) X2	浴室的数量	continuous variable	$\mathbb{R}^+$
(自变量) X3	批量 (千平方英尺)	continuous variable	$\mathbb{R}^+$
(自变量) X4	居住面积 (千平方英尺)	continuous variable	$\mathbb{R}^+$
(自变量) X5	车库档位数量	continuous variable	$\mathbb{R}^+$
(自变量) X6	房间的数量	continuous variable	$\mathbb{R}^+$
(自变量) X7	卧室的数量	continuous variable	$\mathbb{R}^+$
(自变量) X8	住宅年龄 (年)	continuous variable	$\mathbb{R}^+$
(自变量) X9	壁炉的数量	continuous variable	$\mathbb{R}^+$
(因变量) Y	数千美元房子的售价	continuous variable	$\mathbb{R}^+$

1.3 问题

注：这里使用 α=0.05 的显著性水平

1. 判断变量间是否具有多重共线性.
2. 如果存在多重共线性，如何消除多重共线性/选择变量.

1.4 解决方案

Q1:

多重共线性是指自变量 $x_1, x_2, ..., x_p$ 之间不完全线性相关但是相关性很高的情况。此时，虽然可以得到最小二乘估计，但是精度很低。随着自变量之间相关性增加，最小二乘估计结果的方差会增大。

[1]:
```
# Import standard packages
import numpy as np
import pandas as pd
import scipy.stats as stats
import matplotlib.pyplot as plt
import math

# Import additional packages
from itertools import combinations
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor

alpha = 0.05
p = 9
n = 24

x = pd.read_csv('Project7.csv')
x.insert(0, 'intercept', np.ones(len(x)))
data = x.values * 1
df = pd.DataFrame(data)
print(df)

X = data[:,0:p+1]
Y = data[:,-1]
```

```
     0      1    2      3      4    5    6    7     8    9     10
0   1.0  4.918  1.0  3.472  0.998  1.0  7.0  4.0  42.0  0.0  25.9
1   1.0  5.021  1.0  3.531  1.500  2.0  7.0  4.0  62.0  0.0  29.5
```

2	1.0	4.543	1.0	2.275	1.175	1.0	6.0	3.0	40.0	0.0	27.9
3	1.0	4.557	1.0	4.050	1.232	1.0	6.0	3.0	54.0	0.0	25.9
4	1.0	5.060	1.0	4.455	1.121	1.0	6.0	3.0	42.0	0.0	29.9
5	1.0	3.891	1.0	4.455	0.988	1.0	6.0	3.0	56.0	0.0	29.9
6	1.0	5.898	1.0	5.850	1.240	1.0	7.0	3.0	51.0	1.0	30.9
7	1.0	5.604	1.0	9.520	1.501	0.0	6.0	3.0	32.0	0.0	28.9
8	1.0	5.828	1.0	6.435	1.225	2.0	6.0	3.0	32.0	0.0	35.9
9	1.0	5.300	1.0	4.988	1.552	1.0	6.0	3.0	30.0	0.0	31.5
10	1.0	6.271	1.0	5.520	0.975	1.0	5.0	2.0	30.0	0.0	31.0
11	1.0	5.959	1.0	6.666	1.121	2.0	6.0	3.0	32.0	0.0	30.9
12	1.0	5.050	1.0	5.000	1.020	0.0	5.0	2.0	46.0	1.0	30.0
13	1.0	8.246	1.5	5.150	1.664	2.0	8.0	4.0	50.0	0.0	36.9
14	1.0	6.697	1.5	6.902	1.488	1.5	7.0	3.0	22.0	1.0	41.9
15	1.0	7.784	1.5	7.102	1.376	1.0	6.0	3.0	17.0	0.0	40.5
16	1.0	9.038	1.0	7.800	1.500	1.5	7.0	3.0	23.0	0.0	43.9
17	1.0	5.989	1.0	5.520	1.256	2.0	6.0	3.0	40.0	1.0	37.9
18	1.0	7.542	1.5	5.000	1.690	1.0	6.0	3.0	22.0	0.0	37.9
19	1.0	8.795	1.5	9.890	1.820	2.0	8.0	4.0	50.0	1.0	44.5
20	1.0	6.083	1.5	6.727	1.652	1.0	6.0	3.0	44.0	0.0	37.9
21	1.0	8.361	1.5	9.150	1.777	2.0	8.0	4.0	48.0	1.0	38.9
22	1.0	8.140	1.0	8.000	1.504	2.0	7.0	3.0	3.0	0.0	36.9
23	1.0	9.142	1.5	7.326	1.831	1.5	8.0	4.0	31.0	0.0	45.8

数据预处理：

[2]:
```
# 对自变量 X 进行标准化
# 自变量 X 的均值
X_mean = []
for i in range(p):
    X_mean.append(np.mean(X[:, i+1]))

# 自变量 X 的标准差
X_L = []
for i in range(p):
    X_L.append(sum((X[:, i+1] - X_mean[i]) ** 2))

# 对自变量 X 标准化 (截距项不用标准化)
```

```
X_std = X * 1.0
X_std[:,1:p+1] = (X[:,1:p+1] - X_mean) / np.sqrt(X_L)
```

做多元线性回归分析

先后对未经标准化和已标准化的数据进行回归分析，得到的 $\hat{\beta}$ 分别如表所示。

[3]:
```
# Do the multiple linear regression
# OLS(endog,exog=None,missing='none',hasconst=None) (endog:因变量, exog= 自变量)
model = sm.OLS(Y, X).fit()
Y_hat = model.fittedvalues
model.summary()
```

[3]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.851
Model:                            OLS   Adj. R-squared:                  0.756
Method:                 Least Squares   F-statistic:                     8.898
Date:                Thu, 15 Apr 2021   Prob (F-statistic):           0.000202
Time:                        22:31:15   Log-Likelihood:                -53.735
No. Observations:                  24   AIC:                             127.5
Df Residuals:                      14   BIC:                             139.3
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         15.3104      5.961      2.568      0.022       2.526      28.095
x1             1.9541      1.038      1.882      0.081      -0.273       4.181
x2             6.8455      4.335      1.579      0.137      -2.453      16.144
x3             0.1376      0.494      0.278      0.785      -0.923       1.198
x4             2.7814      4.395      0.633      0.537      -6.645      12.207
x5             2.0508      1.385      1.481      0.161      -0.919       5.020
x6            -0.5559      2.398     -0.232      0.820      -5.699       4.587
x7            -1.2452      3.423     -0.364      0.721      -8.587       6.096
```

```
x8            -0.0380      0.067     -0.565      0.581      -0.182       0.106
x9             1.7045      1.953      0.873      0.398      -2.485       5.894
==============================================================================
Omnibus:                        1.516   Durbin-Watson:                   1.857
Prob(Omnibus):                  0.469   Jarque-Bera (JB):                1.100
Skew:                           0.262   Prob(JB):                        0.577
Kurtosis:                       2.091   Cond. No.                         470.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[4]:
```
# Do the multiple linear regression
# OLS(endog,exog=None,missing='none',hasconst=None) (endog:因变量, exog= 自变量)
model_std = sm.OLS(Y, X_std).fit()
Y_std_hat = model_std.fittedvalues
model_std.summary()
```

[4]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.851
Model:                            OLS   Adj. R-squared:                  0.756
Method:                 Least Squares   F-statistic:                     8.898
Date:                Thu, 15 Apr 2021   Prob (F-statistic):           0.000202
Time:                        22:31:15   Log-Likelihood:                -53.735
No. Observations:                  24   AIC:                             127.5
Df Residuals:                      14   BIC:                             139.3
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         34.6292      0.607     57.065      0.000      33.328      35.931
```

```
x1            14.8258      7.878      1.882      0.081      -2.070      31.722
x2             7.9045      5.006      1.579      0.137      -2.832      18.641
x3             1.2966      4.658      0.278      0.785      -8.694      11.287
x4             3.6852      5.823      0.633      0.537      -8.803      16.174
x5             5.9459      4.014      1.481      0.161      -2.664      14.556
x6            -2.3585     10.173     -0.232      0.820     -24.178      19.461
x7            -3.3719      9.269     -0.364      0.721     -23.253      16.509
x8            -2.5590      4.529     -0.565      0.581     -12.273       7.155
x9             3.6157      4.143      0.873      0.398      -5.271      12.502
==============================================================================
Omnibus:                        1.516   Durbin-Watson:                   1.857
Prob(Omnibus):                  0.469   Jarque-Bera (JB):                1.100
Skew:                           0.262   Prob(JB):                        0.577
Kurtosis:                       2.091   Cond. No.                         23.7
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

预判变量间是否存在多重共线性

判断变量间是否存在多重共线性可以使用相关系数进行直观判定，也可以采用方差扩大因子法或特征值判定法。

方法 1：直观判定法

[5]:
```
# 相关系数
r = df.corr()
r
```

[5]:

	0	1	2	3	4	5	6
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	1.000000	0.651267	0.689212	0.734274	0.458556	0.640616
2	NaN	0.651267	1.000000	0.412956	0.728592	0.224022	0.510310
3	NaN	0.689212	0.412956	1.000000	0.571552	0.204664	0.392124
4	NaN	0.734274	0.728592	0.571552	1.000000	0.358884	0.678861

```
5  NaN  0.458556  0.224022  0.204664  0.358884  1.000000  0.589387
6  NaN  0.640616  0.510310  0.392124  0.678861  0.589387  1.000000
7  NaN  0.367113  0.426401  0.151609  0.574335  0.541299  0.870388
8  NaN -0.437101 -0.100748 -0.352751 -0.139087 -0.020169  0.124266
9  NaN  0.146683  0.204124  0.305995  0.106561  0.101618  0.222222
10 NaN  0.873912  0.709777  0.647636  0.707766  0.461468  0.528444
```

```
                7         8             9         10
0             NaN       NaN           NaN       NaN
1   3.671126e-01 -0.437101  1.466825e-01  0.873912
2   4.264014e-01 -0.100748  2.041241e-01  0.709777
3   1.516093e-01 -0.352751  3.059946e-01  0.647636
4   5.743353e-01 -0.139087  1.065612e-01  0.707766
5   5.412988e-01 -0.020169  1.016185e-01  0.461468
6   8.703883e-01  0.124266  2.222222e-01  0.528444
7   1.000000e+00  0.313511 -5.797951e-17  0.281520
8   3.135114e-01  1.000000  2.257796e-01 -0.397403
9  -5.797951e-17  0.225780  1.000000e+00  0.266878
10  2.815200e-01 -0.397403  2.668783e-01  1.000000
```

1. 当与因变量之间的简单相关系数绝对值很大的自变量在回归方程中没有通过显著性检验时，可初步判断存在严重的多重共线性。

[6]:
```
r_xy = np.array(r.iloc[1:p+1][p+1])
print('因变量和每个自变量之间的相关系数: \n', r_xy)

judge_xy = True
for i in range(p):
    if (abs(r_xy[i]) >= 0.5) & (model_std.pvalues[i+1] >= alpha):
        judge_xy = False
        print('自变量 %d 与因变量之间的简单相关系数为: %.4f, tPal: %.4f.' %␣
↪(i+1, r_xy[i], model_std.pvalues[i+1]))

if judge_xy:
    print('\n自变量之间不存在多重共线性。')
else:
    print('\n自变量之间存在多重共线性。')
```

```
因变量和每个自变量之间的相关系数:
 [ 0.87391169  0.7097771   0.64763642  0.70776562  0.46146792  0.52844361
  0.28151997 -0.39740338  0.26687833]
自变量 1 与因变量之间的简单相关系数为: 0.8739, tPal: 0.0808.
自变量 2 与因变量之间的简单相关系数为: 0.7098, tPal: 0.1367.
自变量 3 与因变量之间的简单相关系数为: 0.6476, tPal: 0.7848.
自变量 4 与因变量之间的简单相关系数为: 0.7078, tPal: 0.5370.
自变量 6 与因变量之间的简单相关系数为: 0.5284, tPal: 0.8200.

自变量之间存在多重共线性。
```

2. 在自变量的相关矩阵中，当自变量间的相关系数较大时会出现多重共线性的问题。

[7]:
```python
judge_xx = True
for (i, j) in combinations(range(1, p+1), 2):
    if(r.iloc[i][j] >= 0.7):
        judge_xx = False
        print('变量 (%d,%d) 之间相关系数较大，为: %.4f'% (i, j, r.iloc[i][j]))

if judge_xx:
    print('\n自变量之间不存在多重共线性。')
else:
    print('\n自变量之间存在多重共线性。')
```

```
变量 (1,4) 之间相关系数较大，为: 0.7343
变量 (2,4) 之间相关系数较大，为: 0.7286
变量 (6,7) 之间相关系数较大，为: 0.8704

自变量之间存在多重共线性。
```

可以看出，自变量 1、2、3、4、6 与因变量之间的相关系数高且没有通过显著性检验，且变量 (1,4)、(2,4)、(6,7) 之间的相关系数较大，这说明自变量之间存在多重共线性。

方法 2：方差扩大因子法

1. 计算自变量 x_j 的方差扩大因子 $\mathsf{VIF_j}$，$j = 1, \cdots, p$.

[8]:
```python
# 法 1:
c = np.dot(X_std.T, X_std)
C = np.linalg.inv(c)  # 求逆
```

```
C_list = []
for i in range(p):
    C_list.append(C[i + 1][i + 1])

# 法 2:
vif = [variance_inflation_factor(X_std[:,1:p + 1], i) for i in range(p)]

print('C 主对角线元素  方差扩大因子: ')
for i in range(p):
    print('%d. %.4f        %.4f' % (i+1, C_list[i], vif[i]))
```

```
C 主对角线元素  方差扩大因子:
1. 7.0219        7.0219
2. 2.8355        2.8355
3. 2.4549        2.4549
4. 3.8363        3.8363
5. 1.8234        1.8234
6. 11.7109        11.7109
7. 9.7218        9.7218
8. 2.3211        2.3211
9. 1.9424        1.9424
```

2. 通过 $\mathrm{VIF_j}$ 的大小判断自变量之间是否存在多重共线性.
   如果 VIF 值大于 10 说明共线性很严重，这种情况需要处理，如果 VIF 值在 5 以下不需要处理，如果 VIF 介于 5~10 之间视情况而定。

[9]:
```
thres_vif = 5
for i in range(p):
    if vif[i] >= thres_vif:
        print('自变量 x%d 与其余自变量之间存在多重共线性，其中 VIF 值为: %.4f' %␣
↪(i + 1, vif[i]))
```

```
自变量 x1 与其余自变量之间存在多重共线性，其中 VIF 值为: 7.0219
自变量 x6 与其余自变量之间存在多重共线性，其中 VIF 值为: 11.7109
自变量 x7 与其余自变量之间存在多重共线性，其中 VIF 值为: 9.7218
```

方法 3：特征值判定法

1. 计算自变量 x_j 的条件数 $\kappa_j = \sqrt{\frac{\lambda_1}{\lambda_j}}$, $j = 1, \cdots, p$.

[10]:
```
corr = np.corrcoef(X_std[:,1:p+1], rowvar = 0) # 相关系数矩阵
w, v = np.linalg.eig(corr) # 特征值 & 特征向量

kappa = []
for i in range(p):
    kappa.append(np.sqrt(max(w) / w[i]))
    print('特征值%d: %.4f, kappa%d: %.4f' %(i + 1, w[i], i + 1, kappa[i]))
```

```
特征值 1: 4.2142, kappa1: 1.0000
特征值 2: 1.7062, kappa2: 1.5716
特征值 3: 1.1450, kappa3: 1.9185
特征值 4: 0.7997, kappa4: 2.2955
特征值 5: 0.4616, kappa5: 3.0216
特征值 6: 0.0426, kappa6: 9.9503
特征值 7: 0.2805, kappa7: 3.8760
特征值 8: 0.1610, kappa8: 5.1156
特征值 9: 0.1892, kappa9: 4.7198
```

2. 通过 κ_p 的大小判断自变量之间是否存在多重共线性以及多重共线性的严重程度.
   记 $\kappa = \lambda_{max}/\lambda_{min}$，从实际应用的角度，一般若 $\kappa < 100$，则认为多重共线性的程度很小，若是 $100 <= \kappa <= 1000$，则认为存在一般程度上的多重共线性，若是 $\kappa > 1000$，则就认为存在严重的多重共线性.
   $\kappa >= c_\kappa$ 时，自变量之间存在多重共线性，c_κ 常见取值为 10，100，1000.

[11]:
```
thres_kappa = 10
if np.max(kappa) >= thres_kappa:
    print('设计矩阵 X 存在多重共线性，其中 kappa 值为：%.4f' % np.max(kappa))
else:
    print('设计矩阵 X 不存在多重共线性，其中 kappa 值为：%.4f' % np.max(kappa))
```

```
设计矩阵 X 不存在多重共线性，其中 kappa 值为：9.9503
```

由上述结果可以看出，虽然直观上看特征值 6 接近 0，但是根据 kappa 值判断的结果并没有超过预设的上限。结合其他两种方法，可以大致判断，设计矩阵存在一定多重共线性，但是程度并不高。

Q2:

消除多重共线性：1. 剔除不重要的解释变量 2. 增大样本量

方法 1：剔除不重要的解释变量

在剔除时，我们可以以方差扩大因子为标准，去掉容易导致高共线性的自变量。也可以采用上一节中讲过的自变量选择的方法来选择合适的自变量，下面使用了后退法来选择自变量。

[12]:
```
# 利用 VIF 删除导致高共线性的变量
col = list(range(X_std.shape[1]))
dropped = True
while dropped:
    dropped = False
    vif_drop = [variance_inflation_factor(X_std[:,col], i) for i in␣
↪range(X_std[:,col].shape[1])]
    maxvif = max(vif_drop)
    maxix = vif_drop.index(maxvif)
    if maxvif > thres_vif:
        del col[maxix]
        dropped = True

    if dropped:
        print('剔除剩余变量中第 %d 列变量: ' % maxix, '剩余变量: ', col)

        # 利用 AIC、BIC 准则做变量选择的一个参考
        X_std_vif = X_std[:, col]
        model_vif = sm.OLS(Y, X_std_vif).fit()
        print('此时模型的 AIC 值为: %.4f'% model_vif.aic)
```

剔除剩余变量中第 6 列变量: 剩余变量: [0, 1, 2, 3, 4, 5, 7, 8, 9]
此时模型的 AIC 值为: 125.5623

[13]:
```
# Do the multiple linear regression
X_std_vif = X_std[:, col]
model_vif = sm.OLS(Y, X_std_vif).fit()
model_vif.summary()
```

[13]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.851
```

```
Model:                            OLS   Adj. R-squared:                  0.771
Method:                 Least Squares   F-statistic:                     10.68
Date:                Thu, 15 Apr 2021   Prob (F-statistic):           5.94e-05
Time:                        22:31:19   Log-Likelihood:                -53.781
No. Observations:                  24   AIC:                             125.6
Df Residuals:                      15   BIC:                             136.2
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         34.6292      0.587     58.955      0.000      33.377      35.881
x1            13.7810      6.254      2.204      0.044       0.451      27.111
x2             8.2433      4.634      1.779      0.096      -1.635      18.121
x3             1.3870      4.493      0.309      0.762      -8.189      10.963
x4             3.6215      5.630      0.643      0.530      -8.378      15.621
x5             5.9877      3.882      1.543      0.144      -2.286      14.262
x6            -5.1787      4.857     -1.066      0.303     -15.531       5.174
x7            -2.5804      4.383     -0.589      0.565     -11.923       6.762
x8             3.1554      3.520      0.896      0.384      -4.347      10.658
==============================================================================
Omnibus:                        1.705   Durbin-Watson:                   1.835
Prob(Omnibus):                  0.426   Jarque-Bera (JB):                1.110
Skew:                           0.217   Prob(JB):                        0.574
Kurtosis:                       2.039   Cond. No.                         12.5
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[14]:
```
# 后退法
col0 = list(range(X_std.shape[1]))
col1 = col0 * 1
dropped1 = True
```

```
aic_model = sm.OLS(Y, X_std).fit().aic
while dropped1:
    X_std_aic = X_std[:, col1]
    model_aic = sm.OLS(Y, X_std_aic).fit().aic
    aic = []
    for i in range(len(col1)):
        col2 = col1 * 1
        del col2[i]
        aic.append(sm.OLS(Y, X_std[:, col2]).fit().aic)
    minaic = min(aic[1:len(aic)])
    minaic_rank = aic.index(minaic)
    minaic_ix = col1[minaic_rank]
    if minaic < model_aic:
        del col1[minaic_rank]
    else:
        dropped1 = False
    if dropped1:
        print('剔除剩余变量中第 %d 列变量: ' % minaic_ix, '剩余变量: ', col1)
        print('此时模型的 AIC 值为: %.4f'% minaic)
```

```
剔除剩余变量中第 6 列变量:  剩余变量:  [0, 1, 2, 3, 4, 5, 7, 8, 9]
此时模型的 AIC 值为: 125.5623
剔除剩余变量中第 3 列变量:  剩余变量:  [0, 1, 2, 4, 5, 7, 8, 9]
此时模型的 AIC 值为: 123.7143
剔除剩余变量中第 8 列变量:  剩余变量:  [0, 1, 2, 4, 5, 7, 9]
此时模型的 AIC 值为: 122.3525
剔除剩余变量中第 4 列变量:  剩余变量:  [0, 1, 2, 5, 7, 9]
此时模型的 AIC 值为: 121.2717
剔除剩余变量中第 9 列变量:  剩余变量:  [0, 1, 2, 5, 7]
此时模型的 AIC 值为: 120.3662
```

[15]:
```
# Do the multiple linear regression
X_std_aic = X_std[:, col1]
model_aic = sm.OLS(Y, X_std_aic).fit()
model_aic.summary()
```

[15]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.832
Model:                            OLS   Adj. R-squared:                  0.797
Method:                 Least Squares   F-statistic:                     23.54
Date:                Thu, 15 Apr 2021   Prob (F-statistic):           3.87e-07
Time:                        22:31:21   Log-Likelihood:                -55.183
No. Observations:                  24   AIC:                             120.4
Df Residuals:                      19   BIC:                             126.3
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         34.6292      0.553     62.587      0.000      33.471      35.787
x1            18.3019      3.964      4.617      0.000      10.005      26.599
x2             9.7675      3.845      2.540      0.020       1.719      17.816
x3             5.9737      3.547      1.684      0.109      -1.451      13.398
x4            -5.9995      3.493     -1.717      0.102     -13.311       1.312
==============================================================================
Omnibus:                        2.014   Durbin-Watson:                   2.005
Prob(Omnibus):                  0.365   Jarque-Bera (JB):                1.096
Skew:                          -0.083   Prob(JB):                        0.578
Kurtosis:                       1.966   Cond. No.                         9.94
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[16]:
```
# 方差扩大因子法
print('方差扩大因子: ')
```

```
vif_aic = [variance_inflation_factor(X_std_aic, i) for i in range(X_std_aic.
 ↪shape[1])]
for i in range(X_std_aic.shape[1] - 1):
    print('%.4f' % vif_aic[i + 1])

print('\n')
for i in range(X_std_aic.shape[1] - 1):
    if vif_drop[i] >= thres_vif:
        print('自变量 x%d 与其余自变量之间存在多重共线性，其中 VIF 值为: %.4f' %␣
 ↪(i + 1, vif_drop[i + 1]))
```

方差扩大因子:

2.1389

2.0124

1.7127

1.6611

[17]:
```
# 特征值判定法
corr_ = np.corrcoef(X_std_aic[:,1:p+1], rowvar = 0) # 相关系数矩阵
w_, v_ = np.linalg.eig(corr_) # 特征值 & 特征向量

kappa_ = []
for i in range(X_std_aic.shape[1] - 1):
    kappa_.append(np.sqrt(w_[0] / w_[i]))
    print('特征值%d: %.4f, kappa%d: %.4f' %(i + 1, w_[i], i + 1, kappa_[i]))

if max(kappa_) >= thres_kappa:
    print('\n设计矩阵 X 存在多重共线性，其中 kappa 值为: %.4f' % max(kappa_))
else:
    print('\n设计矩阵 X 不存在多重共线性，其中 kappa 值为: %.4f' % max(kappa_))
```

特征值 1: 2.3389, kappa1: 1.0000

特征值 2: 0.8724, kappa2: 1.6374

特征值 3: 0.2427, kappa3: 3.1041

特征值 4: 0.5460, kappa4: 2.0697

```
设计矩阵 X 不存在多重共线性，其中 kappa 值为：3.1041
```

综上，剔除原始数据的第 3、4、6、8 和 9 个变量后，方差扩大因子和 kappa 值都有一定程度的减小，可以一定程度上消除变量间的多重共线性。

方法 2：增大样本量

这里不用增大样本量，原因有二：

其一，这个数据集中数据量是充足的，而且不是因为样本量过少而导致的多重共线性问题，更多是因为这个变量之间的相关性很强造成的；

其二，增加变量的方法，只是在于采集数据时，如果样本量过小可能会产生多重共线性的问题，因此需要采集足够多的样本。在实际分析阶段，往往无法增加样本量。