迴歸分析 期末報告

順序：第八位

姓名：温宏岳

題目：World Happiness Expanatory Data Analysis(世界幸福指數數據分析)

參考資料：https://www.kaggle.com/datasets/mathurinache/world-happiness-report-20152021?select=2021.csv (https://www.kaggle.com/datasets/mathurinache/world-happiness-report-20152021?select=2021.csv)

1、資料匯入與預處理

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from collections import Counter
from pandas import DataFrame

import sklearn
import statsmodels.api as sm
from statsmodels.compat import lzip
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from statsmodels.sandbox.regression.predstd import wls_prediction_std
from sklearn.model_selection import train_test_split
from patsy import dmatrices
from statsmodels.stats.outliers_influence import variance_inflation_factor

import scipy.stats as stats
from sklearn.metrics import mean_squared_error

import seaborn as sns
import matplotlib.mlab as mlab

from mlxtend.plotting import plot_sequential_feature_selection as plot_sfs
import matplotlib.pyplot as plt

sns.set_style("whitegrid")
sns.set_context("paper")
df = pd.read_csv("2021a.csv")
```

In [2]:

```
df.head()
```

Out[2]:

	CountryName	RegionalIndicator	LadderScore	StandardErrorOfLadderScore	upperwhisker	lowerwhisker	LoggedGDPPerCapita	SocialSupport	HealthyLi
0	Finland	Western Europe	7.842	0.032	7.904	7.780	10.775	0.954	
1	Denmark	Western Europe	7.620	0.035	7.687	7.552	10.933	0.954	
2	Switzerland	Western Europe	7.571	0.036	7.643	7.500	11.117	0.942	
3	Iceland	Western Europe	7.554	0.059	7.670	7.438	10.878	0.983	
4	Netherlands	Western Europe	7.464	0.027	7.518	7.410	10.932	0.942	

In [3]:

```
df.info()
print('\n')
print('len = ' ,len(df))
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149 entries, 0 to 148
Data columns (total 20 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   CountryName                       149 non-null    object 
 1   RegionalIndicator                 149 non-null    object 
 2   LadderScore                       149 non-null    float64
 3   StandardErrorOfLadderScore        149 non-null    float64
 4   upperwhisker                      149 non-null    float64
 5   lowerwhisker                      149 non-null    float64
 6   LoggedGDPPerCapita                149 non-null    float64
 7   SocialSupport                     149 non-null    float64
 8   HealthyLifeExpectancy             149 non-null    float64
 9   FreedomToMakeLifeChoices          149 non-null    float64
 10  Generosity                        149 non-null    float64
 11  PerceptionsOfCorruption           149 non-null    float64
 12  LadderScoreInDystopia             149 non-null    float64
 13  ExplainedbyLogGDPpercapita        149 non-null    float64
 14  ExplainedbySocialsupport          149 non-null    float64
 15  ExplainedbyHealthylifeexpectancy  149 non-null    float64
 16  ExplainedbyFreedomtomakelifechoices  149 non-null    float64
 17  ExplainedbyGenerosity             149 non-null    float64
 18  ExplainedbyPerceptionsofcorruption  149 non-null    float64
 19  Dystopiaresidual                  149 non-null    float64
dtypes: float64(18), object(2)
memory usage: 23.4+ KB


len =  149
```

a. 檢查資料是否有缺失值

In [4]:

```
df.isnull().sum(axis=0)
```

Out[4]:

```
CountryName                            0
RegionalIndicator                      0
LadderScore                            0
StandardErrorOfLadderScore             0
upperwhisker                           0
lowerwhisker                           0
LoggedGDPPerCapita                     0
SocialSupport                          0
HealthyLifeExpectancy                  0
FreedomToMakeLifeChoices               0
Generosity                             0
PerceptionsOfCorruption                0
LadderScoreInDystopia                  0
ExplainedbyLogGDPpercapita             0
ExplainedbySocialsupport               0
ExplainedbyHealthylifeexpectancy       0
ExplainedbyFreedomtomakelifechoices    0
ExplainedbyGenerosity                  0
ExplainedbyPerceptionsofcorruption     0
Dystopiaresidual                       0
dtype: int64
```

b. Drop 無關值

In [5]:

```
df = df.drop(['StandardErrorOfLadderScore','CountryName','RegionalIndicator','upperwhisker','lowerwhisker',\
              'ExplainedbyLogGDPpercapita','ExplainedbySocialsupport','ExplainedbyHealthylifeexpectancy',\
              'ExplainedbyFreedomtomakelifechoices',\'ExplainedbyGenerosity','ExplainedbyPerceptionsofcorruption',\
              'Dystopiaresidual'], axis = 1)
df.head()
```

Out[5]:

	LadderScore	LoggedGDPPerCapita	SocialSupport	HealthyLifeExpectancy	FreedomToMakeLifeChoices	Generosity	PerceptionsOfCorruption	LadderScor
0	7.842	10.775	0.954	72.0	0.949	-0.098	0.186	
1	7.620	10.933	0.954	72.7	0.946	0.030	0.179	
2	7.571	11.117	0.942	74.4	0.919	0.025	0.292	
3	7.554	10.878	0.983	73.0	0.955	0.160	0.673	
4	7.464	10.932	0.942	72.4	0.913	0.175	0.338	

2、複迴歸模型

score會透過R^2來判定我們模型的精準程度，如果訓練集的分數很高，但測試集的分數卻很低，那就是過度擬和

In [6]:

```
model = LinearRegression()
X, y = df[['LoggedGDPPerCapita','SocialSupport','HealthyLifeExpectancy','FreedomToMakeLifeChoices','Generosity',\
           'PerceptionsOfCorruption','LadderScoreInDystopia']], df['LadderScore']

model.fit(X, y)
print('score = ',model.score(X, y))
```

```
score =  0.7558471374226855
```

a. 從理論公式推導

In [7]:

```
yhat = model.predict(X)
SS_Residual = sum((y-yhat)**2)
SS_Total = sum((y-np.mean(y))**2)
r_squared1 = 1 - (float(SS_Residual))/SS_Total
adjusted_r_squared1 = 1 - (1-r_squared1)*(len(y)-1)/(len(y)-X.shape[1]-1)

print(r_squared1, adjusted_r_squared1)
```

```
0.7558471374226854 0.7437260733231024
```

b. 使用 sklearn linear_model 計算

雖然無法直接從文檔中找到任何計算adjusted R^2方式的函數

In [8]:

```
result2_rsquared= model.score(X, y)
result2_rsquared_adj = 1 - (1-model.score(X, y))*(len(y)-1)/(len(y)-X.shape[1]-1)

print(result2_rsquared, result2_rsquared_adj)
```

```
0.7558471374226855 0.7437260733231026
```

c. 使用 statsmodels 計算

In [9]:

```
import statsmodels.formula.api as sm
result3 = sm.ols(formula="LadderScore~ LoggedGDPPerCapita + SocialSupport + HealthyLifeExpectancy + \
                          FreedomToMakeLifeChoices + Generosity + PerceptionsOfCorruption + LadderScoreInDystopia"\
                          , data=df).fit()

print(result3.rsquared, result3.rsquared_adj)
```

```
0.7558471374226855 0.7455308192856158
```

d. 比較

In [10]:

```
from tabulate import tabulate
# creating a DataFrame
dict = {'Model' : ['a', 'b', 'c'],
        'R^2' : [r_squared1, result2_rsquared, result3.rsquared],
        'Adj R^2' : [adjusted_r_squared1, result2_rsquared_adj, result3.rsquared_adj]}

dataframe = pd.DataFrame(dict)

# displaying the DataFrame
dataframe.style
print(tabulate(dataframe, headers = 'keys', tablefmt = 'pretty'))
```

```
+---+-------+--------------------+--------------------+
|   | Model |        R^2         |      Adj R^2       |
+---+-------+--------------------+--------------------+
| 0 |   a   | 0.7558471374226854 | 0.7437260733231024 |
| 1 |   b   | 0.7558471374226855 | 0.7437260733231026 |
| 2 |   c   | 0.7558471374226855 | 0.7455308192856158 |
+---+-------+--------------------+--------------------+
```

=> 由上述3種方式得知調整後的R^2最好的是c的模型，也就是使用 statsmodels 的模型，因此在之後的模型皆以此方式去fit。

3、OLS 線性關係判斷

線性回歸模型，顧名思義，首先要保證自變量與因變量之間存在線性關係。關於線性關係的判斷，我們可以通過圖形或Pearson相關係數來識別 一般情況下的評判標準，

當Pearson相關係數 低於0.4 ，則表明變量之間存在弱相關關係；

當Pearson相關係數在 0.4~0.6之間 ，則說明變量之間存在中度相關關係；

當相關係數在 0.6以上時 ，則反映變量之間存在強相關關係。

a. 創建線性迴歸最小平方法模型(使用原始資料)

In [11]:

```
import statsmodels.formula.api as smf
import statsmodels.api as sm

model = sm.OLS(y,X)
results = model.fit()

results.summary()
```

Out[11]:

OLS Regression Results

Dep. Variable:	LadderScore	**R-squared:**	0.756
Model:	OLS	**Adj. R-squared:**	0.746
Method:	Least Squares	**F-statistic:**	73.27
Date:	Tue, 13 Dec 2022	**Prob (F-statistic):**	5.06e-41
Time:	02:26:41	**Log-Likelihood:**	-116.50
No. Observations:	149	**AIC:**	247.0
Df Residuals:	142	**BIC:**	268.0
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
LoggedGDPPerCapita	0.2795	0.087	3.219	0.002	0.108	0.451
SocialSupport	2.4762	0.668	3.706	0.000	1.155	3.797
HealthyLifeExpectancy	0.0303	0.013	2.274	0.024	0.004	0.057
FreedomToMakeLifeChoices	2.0105	0.495	4.063	0.000	1.032	2.989
Generosity	0.3644	0.321	1.134	0.259	-0.271	0.999
PerceptionsOfCorruption	-0.6051	0.291	-2.083	0.039	-1.179	-0.031
LadderScoreInDystopia	-0.9207	0.259	-3.548	0.001	-1.434	-0.408

Omnibus:	12.908	**Durbin-Watson:**	1.614
Prob(Omnibus):	0.002	**Jarque-Bera (JB):**	13.688
Skew:	-0.667	**Prob(JB):**	0.00107
Kurtosis:	3.650	**Cond. No.**	1.05e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.05e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

b. LadderScore與自變量之間的相關係數

In [12]:

```
df.corrwith(df['LadderScore'])
```

Out[12]:

```
LadderScore                 1.000000
LoggedGDPPerCapita          0.789760
SocialSupport               0.756888
HealthyLifeExpectancy       0.768099
FreedomToMakeLifeChoices    0.607753
Generosity                 -0.017799
PerceptionsOfCorruption    -0.421140
LadderScoreInDystopia            NaN
dtype: float64
```

=> 經過對比發現，LadderScore與Generosity之間的為弱相關關係，可以不考慮將該變量納入模型。當然，變量之間不存在線性關係並不代表不存在任何關係，可能是二次函數關係、對數關係等，所以一般還需要進行檢驗和變量轉換。

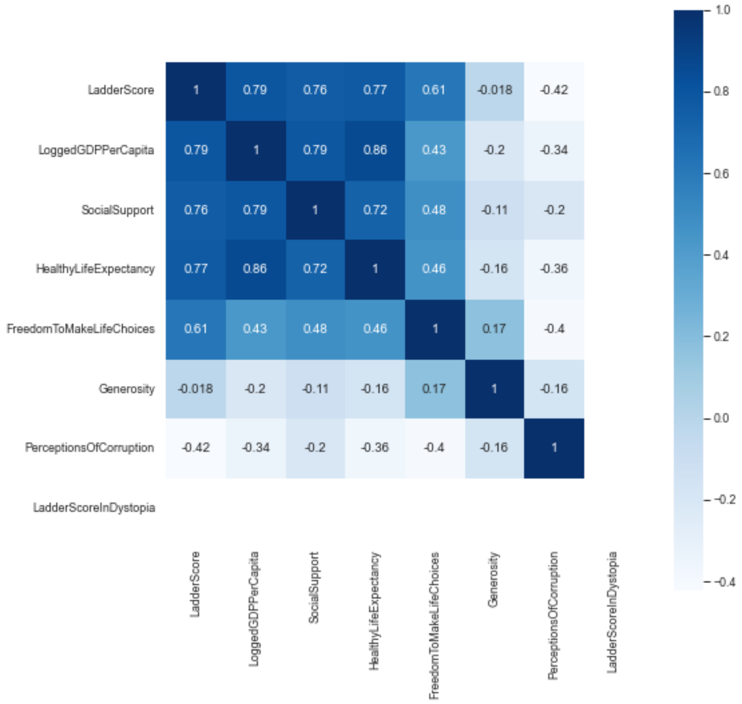
=> 相關係數較大的是Logged GDP per capita、Healthy life expectancy、Social support、Freedom to make life choices。

4、多重共線性判斷

a. 相關性(使用heatmap)

In [13]:

```
plt.figure(figsize=(8,8))
sns.heatmap(df.corr(), annot=True, vmax=1, square=True, cmap='Blues')

plt.show()
```



b. Boxplot

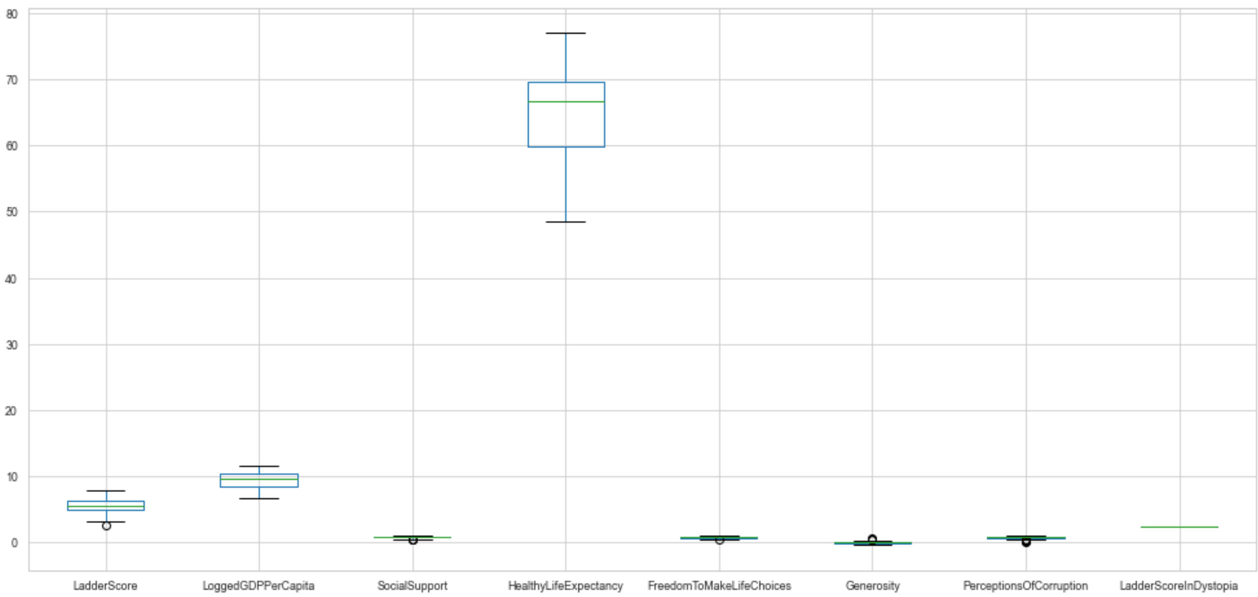
極端異常值，即超出四分位數差3倍距離的異常值，用實心點表示；較爲溫和的異常值，即處於1.5倍-3倍四分位數差之間的異常值，用空心點表示。

In [14]:

```
df.boxplot(figsize=(17,8))
```

Out[14]:

<AxesSubplot:>



c. 一開始建立模型-模型顯著性和參數顯著性判斷(使用切分資料做訓練(0.8)和預測(0.2))

In [15]:

```
Train,Test = train_test_split(df, train_size = 0.8, random_state=0)
fit1 = smf.ols('LadderScore~ LoggedGDPPerCapita + SocialSupport + HealthyLifeExpectancy + FreedomToMakeLifeChoices + \
                Generosity + PerceptionsOfCorruption + LadderScoreInDystopia'
               , data = Train).fit()

fit1.summary()
```

Out[15]:

OLS Regression Results

Dep. Variable:	LadderScore	**R-squared:**	0.752
Model:	OLS	**Adj. R-squared:**	0.739
Method:	Least Squares	**F-statistic:**	56.63
Date:	Tue, 13 Dec 2022	**Prob (F-statistic):**	1.14e-31
Time:	02:26:43	**Log-Likelihood:**	-97.408
No. Observations:	119	**AIC:**	208.8
Df Residuals:	112	**BIC:**	228.3
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.3658	0.108	-3.389	0.001	-0.580	-0.152
LoggedGDPPerCapita	0.2185	0.111	1.968	0.052	-0.002	0.438
SocialSupport	2.8474	0.799	3.565	0.001	1.265	4.430
HealthyLifeExpectancy	0.0424	0.016	2.689	0.008	0.011	0.074
FreedomToMakeLifeChoices	1.6991	0.568	2.990	0.003	0.573	2.825
Generosity	0.3891	0.358	1.088	0.279	-0.320	1.098
PerceptionsOfCorruption	-0.5927	0.333	-1.779	0.078	-1.253	0.067
LadderScoreInDystopia	-0.8889	0.262	-3.389	0.001	-1.409	-0.369

Omnibus:	7.976	**Durbin-Watson:**	2.318
Prob(Omnibus):	0.019	**Jarque-Bera (JB):**	7.660
Skew:	-0.592	**Prob(JB):**	0.0217
Kurtosis:	3.375	**Cond. No.**	7.23e+17

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 9.97e-31. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [16]:

```
names = ['Lagrange multiplier statistic', 'p-value',
         'f-value', 'f p-value']

# Get the test result
test_result = sm.stats.diagnostic.het_breuschpagan(fit1.resid, fit1.model.exog)

lzip(names, test_result)
```

Out[16]:

```
[('Lagrange multiplier statistic', 13.146280708740417),
 ('p-value', 0.0686234890786861),
 ('f-value', 2.3182675255960543),
 ('f p-value', 0.037918093587370354)]
```

=> 通過上面結果我們清楚看到：

如果用p-value 來看，The p-value = 0.0686234890786861 大於 α = 0.05，因此 not reject H_0。

但由7個回歸係數的t統計量p值除了Generosity、PerceptionsOfCorruption、LoggedGDPPerCapita其餘的都<0.05，說明剩下的迴歸係數較顯著，因此需要Drop掉P值最大的Generosity重新建模，來處理他造成的強多重共線性問題。

d. 第一次重新建模

In [17]:

```
fit2 = smf.ols('LadderScore~ LoggedGDPPerCapita + SocialSupport + HealthyLifeExpectancy + FreedomToMakeLifeChoices + \
                PerceptionsOfCorruption + LadderScoreInDystopia'
                , data = Train.drop('Generosity', axis = 1)).fit()

fit2.summary()
```

Out[17]:

OLS Regression Results

Dep. Variable:	LadderScore	**R-squared:**	0.749
Model:	OLS	**Adj. R-squared:**	0.738
Method:	Least Squares	**F-statistic:**	67.61
Date:	Tue, 13 Dec 2022	**Prob (F-statistic):**	2.35e-32
Time:	02:26:43	**Log-Likelihood:**	-98.033
No. Observations:	119	**AIC:**	208.1
Df Residuals:	113	**BIC:**	224.7
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.3448	0.106	-3.244	0.002	-0.555	-0.134
LoggedGDPPerCapita	0.1996	0.110	1.818	0.072	-0.018	0.417
SocialSupport	2.9098	0.797	3.649	0.000	1.330	4.490
HealthyLifeExpectancy	0.0413	0.016	2.623	0.010	0.010	0.072
FreedomToMakeLifeChoices	1.8121	0.559	3.241	0.002	0.704	2.920
PerceptionsOfCorruption	-0.6498	0.329	-1.973	0.051	-1.302	0.003
LadderScoreInDystopia	-0.8379	0.258	-3.244	0.002	-1.350	-0.326

Omnibus:	8.416	**Durbin-Watson:**	2.282
Prob(Omnibus):	0.015	**Jarque-Bera (JB):**	8.142
Skew:	-0.606	**Prob(JB):**	0.0171
Kurtosis:	3.419	**Cond. No.**	7.22e+17

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 9.98e-31. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [18]:

```
names = ['Lagrange multiplier statistic', 'p-value',
         'f-value', 'f p-value']

# Get the test result
test_result = sm.stats.diagnostic.het_breuschpagan(fit2.resid, fit2.model.exog)

lzip(names, test_result)
```

Out[18]:

```
[('Lagrange multiplier statistic', 12.658862253462022),
 ('p-value', 0.048784687876246846),
 ('f-value', 2.690306808735979),
 ('f p-value', 0.02457361031936004)]
```

=>通過上面結果我們清楚看到：

如果用p-value 來看，The p-value = 0.048784687876246846 小於 α = 0.05，因此 reject H_0。

但剩下6個回歸係數的t統計量p值除了LoggedGDPPerCapita、PerceptionsOfCorruption、其餘的都<0.05，說明剩下的迴歸係數較顯著，因此需要Drop掉P值最大的LoggedGDPPerCapita繼續重新建模。

e. 第二次重新建模

In [19]:

```
fit3 = smf.ols('LadderScore~ SocialSupport + HealthyLifeExpectancy + FreedomToMakeLifeChoices + PerceptionsOfCorruption + \
                LadderScoreInDystopia'
                , data = Train.drop('LoggedGDPPerCapita', axis = 1)).fit()

fit3.summary()
```

Out[19]:

OLS Regression Results

Dep. Variable:	LadderScore	**R-squared:**	0.742
Model:	OLS	**Adj. R-squared:**	0.733
Method:	Least Squares	**F-statistic:**	82.02
Date:	Tue, 13 Dec 2022	**Prob (F-statistic):**	1.22e-32
Time:	02:26:43	**Log-Likelihood:**	-99.749
No. Observations:	119	**AIC:**	209.5
Df Residuals:	114	**BIC:**	223.4
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.3122	0.106	-2.951	0.004	-0.522	-0.103
SocialSupport	3.7287	0.665	5.610	0.000	2.412	5.045
HealthyLifeExpectancy	0.0607	0.012	5.195	0.000	0.038	0.084
FreedomToMakeLifeChoices	1.5791	0.550	2.872	0.005	0.490	2.668
PerceptionsOfCorruption	-0.7708	0.326	-2.366	0.020	-1.416	-0.125
LadderScoreInDystopia	-0.7586	0.257	-2.951	0.004	-1.268	-0.249

Omnibus:	6.199	**Durbin-Watson:**	2.225
Prob(Omnibus):	0.045	**Jarque-Bera (JB):**	5.773
Skew:	-0.524	**Prob(JB):**	0.0558
Kurtosis:	3.257	**Cond. No.**	7.15e+17

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 9.98e-31. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [20]:

```
names = ['Lagrange multiplier statistic', 'p-value',
         'f-value', 'f p-value']

# Get the test result
test_result = sm.stats.diagnostic.het_breuschpagan(fit3.resid, fit3.model.exog)

lzip(names, test_result)
```

Out[20]:

```
[('Lagrange multiplier statistic', 13.21341343978292),
 ('p-value', 0.02145885728468162),
 ('f-value', 3.559830175817713),
 ('f p-value', 0.008941679488950775)]
```

=> 如果用p-value 來看，The p-value = 0.02145885728468162 小於 α = 0.05，因此 reject H_0。

通過模型反饋的結果我們可知，模型是通過顯著性檢驗的，即剩下6個迴歸係數的t統計量p值遠遠小於0.05這個閾值的，說明需要拒絕原假設(即認為模型的所有回歸係數都不全為0)。

f. 比較

In [21]:

```
from math import sqrt
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

y_predict1 = fit1.predict(exog = Test)

mae1 = mean_absolute_error(Test['LadderScore'],y_predict1)
mse1 = mean_squared_error(Test['LadderScore'],y_predict1)
rmse1 = sqrt(mean_squared_error(Test['LadderScore'],y_predict1))
r2_score1 = r2_score(Test['LadderScore'],y_predict1)
```

In [22]:

```
y_predict2 = fit2.predict(exog = Test)

mae2 = mean_absolute_error(Test['LadderScore'],y_predict2)
mse2 = mean_squared_error(Test['LadderScore'],y_predict2)
rmse2 = sqrt(mean_squared_error(Test['LadderScore'],y_predict2))
r2_score2 = r2_score(Test['LadderScore'],y_predict2)
```

In [23]:

```
y_predict3= fit3.predict(exog = Test)

mae3 = mean_absolute_error(Test['LadderScore'],y_predict3)
mse3 = mean_squared_error(Test['LadderScore'],y_predict3)
rmse3 = sqrt(mean_squared_error(Test['LadderScore'],y_predict3))
r2_score3 = r2_score(Test['LadderScore'],y_predict3)
```

In [24]:

```
from tabulate import tabulate
# creating a DataFrame
dict = {'Model' : ['一開始建立模型', '第一次重新建模', '第二次重新建模'],
        'MSE' : [mse1, mse2, mse3],
        'RMSE' : [rmse1, rmse2, rmse3],
        'R2_score' : [r2_score1, r2_score2, r2_score3],
        'AIC' : [fit1.aic, fit2.aic, fit3.aic],
        'BIC' : [fit1.bic, fit2.bic, fit3.bic]}

dataframe = pd.DataFrame(dict)

# displaying the DataFrame
dataframe.style
print(tabulate(dataframe, headers = 'keys', tablefmt = 'pretty'))
```

```
+---+----------------+---------------------+---------------------+--------------------+--------------------+---------------
-----+
|   |     Model      |         MSE         |        RMSE         |      R2_score      |        AIC         |        BIC
|
+---+----------------+---------------------+---------------------+--------------------+--------------------+---------------
-----+
| 0 | 一開始建立模型 | 0.21186796214884354 | 0.4602911710524584  | 0.7547648927487713 | 208.81508934894586 | 228.26895380072
656 |
| 1 | 第一次重新建模 | 0.2122926966195769  | 0.46075231591341664 | 0.7542732667264735 | 208.06519458409602 | 224.73993554276
518 |
| 2 | 第二次重新建模 | 0.2676812064772173  | 0.5173791708961787  | 0.6901616048326304 | 209.49718702713366 | 223.39280449269
13  |
+---+----------------+---------------------+---------------------+--------------------+--------------------+---------------
-----+
```

=> 對於連續多變量預測效果的好壞,我們可以信賴於RMSE(均方根誤差,即真實值與預測值的均方根)來衡量,如果這個值越小就說明模型越優秀,即預測出來的值會越接近於真實值且很明顯

=> 所以當我們把Generosity、LoggedGDPPerCapita移除掉反而會讓RMSE、AIC值變大，BIC、R2_score值變小。

=> 因此模型1的RMSE相比於模型2、3會小一些,模型會更符合實際。

g. VIF

如果自變量X與其他自變量共線性強，那麼迴歸方程的R2就會較高，導致VIF也高。一般，有自變量VIF值大於10，則說明存在嚴重多重共線性，可以選擇刪除該變量或者用其他類似但VIF低的變量代替。

可以看到AT的方差膨脹因子大於10，可以刪除該變量。

(3)多重共線性的處理方法

多重共線性對於線性回歸是種災難，並且我們不可能完全消除，而只能利用一些方法來減輕它的影響。對於多重共線性的處理方式，有以下幾種思路：

1)提前篩選變量：可以利用相關檢驗來或變量聚類的方法。注意：決策樹和隨機森林也可以作為提前篩選變量的方法，但是它們對於多重共線性幫助不大，因為如果按照特徵重要性排序，共線性的變量很可能都排在前面。

2)子集選擇：包括逐步回歸和最優子集法。因為該方法是貪婪算法，理論上大部分情況有效，實際中需要結合第一種方法。

3)收縮方法：正則化方法，包括嶺回歸和LASSO回歸。LASSO回歸可以實現篩選變量的功能。

4)維數縮減：包括主成分迴歸(PCR)和偏最小平方法迴歸(PLS)方法。

In [25]:

```
y, X = dmatrices('LadderScore~ LoggedGDPPerCapita + SocialSupport + HealthyLifeExpectancy + FreedomToMakeLifeChoices + \
                  Generosity + PerceptionsOfCorruption + LadderScoreInDystopia'
                  , data = df, return_type='dataframe')

vif = pd.DataFrame()
vif["VIF Factor"]=[variance_inflation_factor(X.values,i) for i in range(X.shape[1])]

vif["feature"]=X.columns
vif
```

```
C:\Users\willy\anaconda3\lib\site-packages\statsmodels\regression\linear_model.py:1736: RuntimeWarning: divide by zero enco
untered in double_scalars
  return 1 - self.ssr/self.centered_tss
```

Out[25]:

	VIF Factor	feature
0	0.000000	Intercept
1	5.104890	LoggedGDPPerCapita
2	2.972200	SocialSupport
3	4.099348	HealthyLifeExpectancy
4	1.585807	FreedomToMakeLifeChoices
5	1.180982	Generosity
6	1.367122	PerceptionsOfCorruption
7	0.000000	LadderScoreInDystopia

=> 結果顯示,所有自變量的VIF均低於10,說明自變量之間並不存在多重共線性的隱患。

5、強影響點診斷

Cook's D統計量、DFFITS統計量、DFBETAS統計量

In [26]:

```
#離群點檢驗
outliers = fit1.get_influence()
#高槓杆值點（帽子矩陣）
leverage = outliers.hat_matrix_diag
#dffts值
dffits = outliers.dffits[0]
#學生化殘差
resid_stu = outliers.resid_studentized_external
#cook距離
cook = outliers.cooks_distance[0]
#covratio值
covratio = outliers.cov_ratio

#將上面的幾種異常值檢驗統計量與原始數據集合
contat1 = pd.concat([pd.Series(leverage, name = 'leverage'),pd.Series(dffits, name = 'dffits'),

pd.Series(resid_stu,name ='resid_stu'),pd.Series(cook,name = 'cook'),
pd.Series(covratio, name = 'covratio'),], axis = 1)
df_outliers = pd.concat([df,contat1], axis = 1)

df_outliers.head()
```

Out[26]:

	LadderScore	LoggedGDPPerCapita	SocialSupport	HealthyLifeExpectancy	FreedomToMakeLifeChoices	Generosity	PerceptionsOfCorruption	LadderScor
0	7.842	10.775	0.954	72.0	0.949	-0.098	0.186	
1	7.620	10.933	0.954	72.7	0.946	0.030	0.179	
2	7.571	11.117	0.942	74.4	0.919	0.025	0.292	
3	7.554	10.878	0.983	73.0	0.955	0.160	0.673	
4	7.464	10.932	0.942	72.4	0.913	0.175	0.338	

a. 計算異常值數量的比例

In [27]:

```
outliers_ratio = sum(np.where((np.abs(df_outliers.resid_stu)>2),1,0))/df_outliers.shape[0]

print('異常值數量的比例 = ',outliers_ratio)
```

異常值數量的比例 = 0.046979865771812O8

b. 刪除異常值

In [28]:

```
df_outliers = df_outliers.loc[np.abs(df_outliers.resid_stu)<=2,]
```

In [29]:

```
data1 = df_outliers.drop(['leverage', 'dffits', 'resid_stu', 'cook', 'covratio'], axis = 1)
```

In [30]:

```
outliers_Train,outliers_Test = train_test_split(data1 , train_size = 0.8, random_state=0)
```

c. 第三次重新建模

In [31]:

```
fit4 = sm.formula.ols('LadderScore~ LoggedGDPPerCapita + SocialSupport + HealthyLifeExpectancy + FreedomToMakeLifeChoices + \
                      Generosity + PerceptionsOfCorruption +LadderScoreInDystopia'\
                      , data = outliers_Train).fit()

fit4.summary()
```

Out[31]:

OLS Regression Results

Dep. Variable:	LadderScore	**R-squared:**	0.760
Model:	OLS	**Adj. R-squared:**	0.743
Method:	Least Squares	**F-statistic:**	43.38
Date:	Tue, 13 Dec 2022	**Prob (F-statistic):**	1.91e-23
Time:	02:26:44	**Log-Likelihood:**	-41.998
No. Observations:	89	**AIC:**	98.00
Df Residuals:	82	**BIC:**	115.4
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.1028	0.108	-0.953	0.343	-0.317	0.112
LoggedGDPPerCapita	0.3202	0.089	3.580	0.001	0.142	0.498
SocialSupport	1.9628	0.762	2.577	0.012	0.448	3.478
HealthyLifeExpectancy	0.0026	0.014	0.187	0.852	-0.025	0.030
FreedomToMakeLifeChoices	2.6519	0.531	4.992	0.000	1.595	3.709
Generosity	0.3410	0.323	1.054	0.295	-0.303	0.985
PerceptionsOfCorruption	-0.6485	0.270	-2.403	0.018	-1.185	-0.112
LadderScoreInDystopia	-0.2498	0.262	-0.953	0.343	-0.771	0.272

Omnibus:	2.560	**Durbin-Watson:**	1.915
Prob(Omnibus):	0.278	**Jarque-Bera (JB):**	2.553
Skew:	-0.375	**Prob(JB):**	0.279
Kurtosis:	2.645	**Cond. No.**	2.81e+17

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 5.19e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

d. 第三次重新建模與上述3種模型做比較

In [32]:

```
y_predict4= fit4.predict(exog = outliers_Test)

mae4 = mean_absolute_error(outliers_Test['LadderScore'],y_predict4)
mse4 = mean_squared_error(outliers_Test['LadderScore'],y_predict4)
rmse4 = sqrt(mean_squared_error(outliers_Test['LadderScore'],y_predict4))
r2_score4 = r2_score(outliers_Test['LadderScore'],y_predict4)
```

In [33]:

```
from tabulate import tabulate
# creating a DataFrame
dict = {'Model' : ['一開始建立模型', '第一次重新建模', '第二次重新建模', '第三次重新建模'],
        'MSE' : [mse1, mse2, mse3, mse4],
        'RMSE' : [rmse1, rmse2, rmse3, rmse4],
        'R2_score' : [r2_score1, r2_score2, r2_score3, r2_score4],
        'AIC' : [fit1.aic, fit2.aic, fit3.aic, fit4.aic],
        'BIC' : [fit1.bic, fit2.bic, fit3.bic, fit4.bic]}

dataframe = pd.DataFrame(dict)

# displaying the DataFrame
dataframe.style
print(tabulate(dataframe, headers = 'keys', tablefmt = 'pretty'))
```

```
+---+----------------+--------------------+--------------------+--------------------+--------------------+--------------
-----+
|   |     Model      |        MSE         |        RMSE        |      R2_score      |        AIC         |        BIC   
|
+---+----------------+--------------------+--------------------+--------------------+--------------------+--------------
-----+
| 0 | 一開始建立模型 | 0.21186796214884354 | 0.4602911710524584  | 0.7547648927487713 | 208.81508934894586 | 228.26895380072
656 |
| 1 | 第一次重新建模 | 0.2122926966195769  | 0.46075231591341664 | 0.7542732667264735 | 208.06519458409602 | 224.73993554276
518 |
| 2 | 第二次重新建模 | 0.2676812064772173  | 0.5173791708961787  | 0.6901616048326304 | 209.49718702713366 | 223.39280449269
13  |
| 3 | 第三次重新建模 | 0.30662190512232085 |  0.553734507794413  | 0.6069360030908807 | 97.99555405655951  | 115.41600864468
45  |
+---+----------------+--------------------+--------------------+--------------------+--------------------+--------------
-----+
```

= > 通過將異常值刪除後重新建模的話會造成反效果: AIC和BIC均變大，同時RMSE也有提升了，也驗證了前面boxplot所呈現的，還有上述異常值數量的比例也是極低的。因此，對於此資料的了解會讓我們不會去fit錯誤的model，而造成反效果。

6、ANOVA Table

In [34]:

```
#!pip install researchpy
import pandas as pd
import researchpy as rp
rp.summary_cont(df['LadderScore'])
```

Out[34]:

	Variable	N	Mean	SD	SE	95% Conf.	Interval
0	LadderScore	149.0	5.5328	1.0739	0.088	5.359	5.7067

In [35]:

```
fit11 = smf.ols('LadderScore~ LoggedGDPPerCapita + SocialSupport + HealthyLifeExpectancy + FreedomToMakeLifeChoices + \
               Generosity + PerceptionsOfCorruption + LadderScoreInDystopia'
               , data = Train).fit()

result = sm.stats.anova_lm(fit11, type=2)
print('一開始建立模型')
print(result)
```

```
一開始建立模型
                             df     sum_sq    mean_sq           F  \
LoggedGDPPerCapita          1.0  88.128918  88.128918  275.600284
SocialSupport               1.0   8.650668   8.650668   27.052717
HealthyLifeExpectancy       1.0   4.597534   4.597534   14.377592
FreedomToMakeLifeChoices    1.0   5.644001   5.644001   17.650145
Generosity                  1.0   0.613651   0.613651    1.919034
PerceptionsOfCorruption     1.0   1.011886   1.011886    3.164410
LadderScoreInDystopia       1.0   0.189438   0.189438    0.592419
Residual                  112.0  35.814327   0.319771         NaN

                                PR(>F)
LoggedGDPPerCapita        5.696124e-32
SocialSupport             9.024823e-07
HealthyLifeExpectancy     2.427715e-04
FreedomToMakeLifeChoices  5.355140e-05
Generosity                1.687160e-01
PerceptionsOfCorruption   7.797251e-02
LadderScoreInDystopia     4.431056e-01
Residual                           NaN
```

In [36]:

```
fit22 = smf.ols('LadderScore~ LoggedGDPPerCapita + SocialSupport + HealthyLifeExpectancy + FreedomToMakeLifeChoices + \
                PerceptionsOfCorruption + LadderScoreInDystopia'
                , data = Train.drop('Generosity', axis = 1)).fit()

result2 = sm.stats.anova_lm(fit22, type=2)
print('第一次重新建模')
print(result2)
```

```
第一次重新建模
                             df     sum_sq    mean_sq           F  \
LoggedGDPPerCapita          1.0  88.128918  88.128918  275.155235
SocialSupport               1.0   8.650668   8.650668   27.009031
HealthyLifeExpectancy       1.0   4.597534   4.597534   14.354375
FreedomToMakeLifeChoices    1.0   5.644001   5.644001   17.621643
PerceptionsOfCorruption     1.0   1.247321   1.247321    3.894374
LadderScoreInDystopia       1.0   0.207096   0.207096    0.646593
Residual                  113.0  36.192543   0.320288         NaN

                                PR(>F)
LoggedGDPPerCapita        4.653080e-32
SocialSupport             9.084253e-07
HealthyLifeExpectancy     2.444682e-04
FreedomToMakeLifeChoices  5.394397e-05
PerceptionsOfCorruption   5.088927e-02
LadderScoreInDystopia     4.230218e-01
Residual                           NaN
```

In [37]:

```
fit33 = smf.ols('LadderScore~ SocialSupport + HealthyLifeExpectancy + FreedomToMakeLifeChoices + PerceptionsOfCorruption + \
                LadderScoreInDystopia'
                , data = Train.drop('LoggedGDPPerCapita', axis = 1)).fit()

result3 = sm.stats.anova_lm(fit33, type=2)
print('第二次重新建模')
print(result3)
```

```
第二次重新建模
                             df     sum_sq    mean_sq           F  \
SocialSupport               1.0  85.702821  85.702821  262.274276
HealthyLifeExpectancy       1.0  14.892938  14.892938   45.576500
FreedomToMakeLifeChoices    1.0   4.784159   4.784159   14.640847
PerceptionsOfCorruption     1.0   1.829523   1.829523    5.598846
LadderScoreInDystopia       1.0   0.214852   0.214852    0.657506
Residual                  114.0  37.251543   0.326768         NaN

                                PR(>F)
SocialSupport             2.449714e-31
HealthyLifeExpectancy     6.457762e-10
FreedomToMakeLifeChoices  2.128068e-04
PerceptionsOfCorruption   1.965953e-02
LadderScoreInDystopia     4.191314e-01
Residual                           NaN
```

In [38]:

```
fit44 = sm.formula.ols('LadderScore~ LoggedGDPPerCapita + SocialSupport + HealthyLifeExpectancy + FreedomToMakeLifeChoices + \
                       Generosity + PerceptionsOfCorruption +LadderScoreInDystopia'\
                       , data = df_outliers).fit()

result4 = sm.stats.anova_lm(fit44, type=2)
print('第三次重新建模')
print(result4)
```

```
第三次重新建模
                             df     sum_sq    mean_sq           F  \
LoggedGDPPerCapita          1.0  39.479700  39.479700  210.213134   
SocialSupport               1.0   4.534560   4.534560   24.144663   
HealthyLifeExpectancy       1.0   0.930714   0.930714    4.955666   
FreedomToMakeLifeChoices    1.0   6.595014   6.595014   35.115733   
Generosity                  1.0   0.380142   0.380142    2.024097   
PerceptionsOfCorruption     1.0   2.201754   2.201754   11.723433   
LadderScoreInDystopia       1.0   0.434498   0.434498    2.313524   
Residual                  105.0  19.719836   0.187808         NaN   

                                PR(>F)  
LoggedGDPPerCapita        8.165657e-27  
SocialSupport             3.305200e-06  
HealthyLifeExpectancy     2.814180e-02  
FreedomToMakeLifeChoices  3.992819e-08  
Generosity                1.577851e-01  
PerceptionsOfCorruption   8.813401e-04  
LadderScoreInDystopia     1.312588e-01  
Residual                           NaN  
```

7、Best subset selection、Forward selection、Backward selection

Using AIC & BIC & Adj r squared

a. 轉換資料label的名稱方便做selection

In [39]:

```
df.columns = ['y','V1','V2','V3','V4','V5','V6','V7']
df
```

Out[39]:

	y	V1	V2	V3	V4	V5	V6	V7
0	7.842	10.775	0.954	72.000	0.949	-0.098	0.186	2.43
1	7.620	10.933	0.954	72.700	0.946	0.030	0.179	2.43
2	7.571	11.117	0.942	74.400	0.919	0.025	0.292	2.43
3	7.554	10.878	0.983	73.000	0.955	0.160	0.673	2.43
4	7.464	10.932	0.942	72.400	0.913	0.175	0.338	2.43
...	...	...	...	...	...	...	...	...
144	3.512	7.926	0.787	48.700	0.715	-0.131	0.915	2.43
145	3.467	9.782	0.784	59.269	0.824	-0.246	0.801	2.43
146	3.415	7.676	0.552	61.400	0.897	0.061	0.167	2.43
147	3.145	7.943	0.750	56.201	0.677	-0.047	0.821	2.43
148	2.523	7.695	0.463	52.493	0.382	-0.102	0.924	2.43

149 rows × 8 columns

In [40]:

```
col_V = ['V' + str(i) for i in range(1,8)]
X = df[col_V]
X.head()
```

Out[40]:

	V1	V2	V3	V4	V5	V6	V7
0	10.775	0.954	72.0	0.949	-0.098	0.186	2.43
1	10.933	0.954	72.7	0.946	0.030	0.179	2.43
2	11.117	0.942	74.4	0.919	0.025	0.292	2.43
3	10.878	0.983	73.0	0.955	0.160	0.673	2.43
4	10.932	0.942	72.4	0.913	0.175	0.338	2.43

In [41]:

```
col_y = ['y']
y = df[col_y]
y.head()
```

Out[41]:

	y
0	7.842
1	7.620
2	7.571
3	7.554
4	7.464

In [42]:

```
X_train,X_test,y_train,y_test = train_test_split(X, y, train_size = 0.8, random_state=0)
```

In [43]:

```
data = pd.concat([X_train,y_train],axis = 1)
data.head()
```

Out[43]:

	V1	V2	V3	V4	V5	V6	V7	y
27	10.623	0.880	73.800	0.693	-0.084	0.866	2.43	6.483
97	7.686	0.690	55.160	0.697	0.424	0.746	2.43	5.051
96	9.629	0.983	62.409	0.877	0.273	0.888	2.43	5.066
69	9.400	0.935	62.500	0.708	0.116	0.856	2.43	5.677
18	11.023	0.920	68.200	0.837	0.098	0.698	2.43	6.951

b. Best subset selection

In [44]:

```
def fit_linear_reg(X,y):
    #Fit linear regression model and return RSS and R squared values
    X = sm.add_constant(X)
    model = sm.OLS(y,X).fit()
    return model.ssr,model.rsquared,model


from tqdm import tnrange, tqdm_notebook
from itertools import combinations

def run_subset_selection(X,y):
    #Initialization variables
    RSS_list, R_squared_list, feature_list = [],[],[]
    aic_list,bic_list,adj_r_squared_list = [],[],[]
    numb_features = []

    #Looping over k = 1 to k = 11 features in X
    for k in tnrange(1,len(X_train.columns) + 1, desc = 'Loop...'):
        best_features = None
        best_RSS = None
        best_r2 = 0
        best_model = None

        #Looping over all possible combinations:
        for combo in combinations(X_train.columns,k):
            tmp_result = fit_linear_reg(X[list(combo)],y)   #Store temp result
            r2 = tmp_result[1]
            if r2 > best_r2:
                best_features = combo
                best_RSS = tmp_result[0]
                best_r2 = tmp_result[1]
                best_model = tmp_result[2]

        RSS_list.append(best_RSS)
        R_squared_list.append(best_r2)
        feature_list.append(best_features)
        numb_features.append(len(best_features))
        aic_list.append(best_model.aic)
        bic_list.append(best_model.bic)
        adj_r_squared_list.append(best_model.rsquared_adj)

    #Store in DataFrame
    df = pd.DataFrame({'numb_features': numb_features,'RSS': RSS_list, 'R_squared':R_squared_list,
                      'AIC':aic_list,'BIC':bic_list,'adj_r2':adj_r_squared_list,'features':feature_list})
    return df
```

In [45]:

```
best_subset_results = run_subset_selection(X_train,y_train)
```

```
C:\Users\willy\AppData\Local\Temp\ipykernel_3396\2503146914.py:18: TqdmDeprecationWarning: Please use `tqdm.notebook.trange
` instead of `tqdm.tnrange`
  for k in tnrange(1,len(X_train.columns) + 1, desc = 'Loop...'):

Loop...:   0%|          | 0/7 [00:00<?, ?it/s]
```

In [46]:

```
best_subset_results
```

Out[46]:

	numb_features	RSS	R_squared	AIC	BIC	adj_r2	features
0	1	56.332067	0.610053	252.712085	258.270332	0.606721	(V1,)
1	2	43.220108	0.700818	223.182071	231.519442	0.695660	(V1, V4)
2	3	39.081067	0.729470	213.202611	224.319105	0.722412	(V2, V3, V4)
3	4	37.251543	0.742134	209.497187	223.392804	0.733086	(V2, V3, V4, V6)
4	5	36.192543	0.749465	208.065195	224.739936	0.738379	(V1, V2, V3, V4, V6)
5	6	35.814327	0.752083	208.815089	228.268954	0.738802	(V1, V2, V3, V4, V5, V6)
6	7	35.814327	0.752083	208.815089	228.268954	0.738802	(V1, V2, V3, V4, V5, V6, V7)

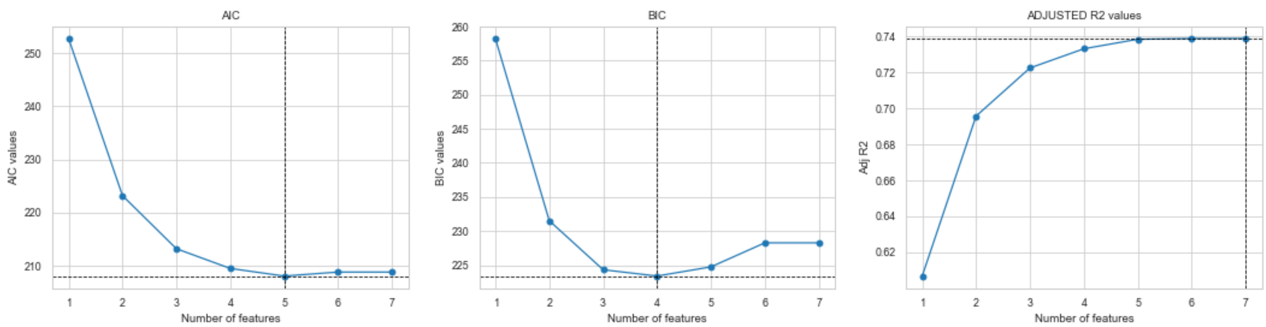
In [47]:

```
print('The best model according to AIC   is model having features - ',
      best_subset_results.sort_values('AIC',ascending=True)['numb_features'].values[0])
print('The best model according to BIC   is model having features - ',
      best_subset_results.sort_values('BIC',ascending=True)['numb_features'].values[0])
print('The best model according to adjR2 is model having features - ',
      best_subset_results.sort_values('adj_r2',ascending=False)['numb_features'].values[0],
     )
```

```
The best model according to AIC   is model having features -  5
The best model according to BIC   is model having features -  4
The best model according to adjR2 is model having features -  7
```

In [48]:

```
def plot_results(df):
    fig,(a1,a2,a3) = plt.subplots(1,3,figsize = (18,4))
    a1.plot(df['numb_features'],df['AIC'],marker = 'o')
    a1.axhline(y = min(df['AIC']),linestyle = 'dashed',linewidth = 0.8,color = 'black')
    a1.axvline(x =  df.sort_values('AIC',ascending=True)['numb_features'].values[0],color = 'black',
               linestyle = 'dashed',linewidth = 0.8)
    a1.set_title('AIC')
    a1.set_xlabel('Number of features')
    a1.set_ylabel('AIC values')

    a2.plot(df['numb_features'],df['BIC'],marker = 'o')
    a2.axhline(y = min(df['BIC']),linestyle = 'dashed',linewidth = 0.8,color = 'black')
    a2.axvline(x =  df.sort_values('BIC',ascending=True)['numb_features'].values[0],color = 'black',
               linestyle = 'dashed',linewidth = 0.8)
    a2.set_title('BIC')
    a2.set_xlabel('Number of features')
    a2.set_ylabel('BIC values')

    a3.plot(df['numb_features'],df['adj_r2'],marker=  'o')
    a3.axhline(y = max(df['adj_r2']),linestyle = 'dashed',linewidth = 0.8,color = 'black')
    a3.axvline(x =  df.sort_values('adj_r2',ascending=False)['numb_features'].values[0],color = 'black',
               linestyle = 'dashed',linewidth = 0.8)

    a3.set_title('ADJUSTED R2 values')
    a3.set_xlabel('Number of features')
    a3.set_ylabel('Adj R2')

plot_results(best_subset_results)
```



In [49]:

```
AIC_F = best_subset_results.sort_values('AIC',ascending=True)['numb_features'].values[0]
BIC_F = best_subset_results.sort_values('BIC',ascending=True)['numb_features'].values[0]
adj_r2_F = best_subset_results.sort_values('adj_r2',ascending=False)['numb_features'].values[0]

print('Features choosen by AIC    is',list(best_subset_results['features'][AIC_F - 1]))
print('Features choosen by BIC    is',list(best_subset_results['features'][BIC_F - 1]))
print('Features choosen by adj_r2 is',list(best_subset_results['features'][adj_r2_F - 1]))
```

```
Features choosen by AIC    is ['V1', 'V2', 'V3', 'V4', 'V6']
Features choosen by BIC    is ['V2', 'V3', 'V4', 'V6']
Features choosen by adj_r2 is ['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7']
```

c. Forward stepwise selection

In [50]:

```
def forward_stepwise_selection(data,target):
    total_features = [[]]
    list_r2 = []
    list_adj_r2 = []
    list_aic,list_bic = [],[]
    len_features = []
    remaining_features = [col for col in data.columns if not col == target]
    for i in range(1,len(data.columns)):
        best_score = 0;best_feature = None
        best_model = None
        for feature in remaining_features:

            X = total_features[i-1] + [feature]
            model = sm.OLS(data[target],sm.add_constant(data[X])).fit()
            score = model.rsquared
#             print('For len {}, feature - {}, score is {}'.format(i,feature,score))

            if score > best_score:
                best_score = score
                best_feature = feature
                best_model = model

        total_features.append(total_features[i-1] + [best_feature])
        remaining_features.remove(best_feature)
        list_r2.append(best_model.rsquared)
        list_adj_r2.append(best_model.rsquared_adj)
        list_aic.append(best_model.aic)
        list_bic.append(best_model.bic)
        len_features.append(len(total_features[-1]))

    return pd.DataFrame({'numb_features': len_features, 'R_squared':list_r2,
                    'AIC':list_aic,'BIC':list_bic,'adj_r2':list_adj_r2,'features':total_features[1:]})
```

In [51]:

```
result_fwd = forward_stepwise_selection(data,'y')
```

In [52]:

```
result_fwd
```

Out[52]:

	numb_features	R_squared	AIC	BIC	adj_r2	features
0	1	0.610053	252.712085	258.270332	0.606721	[V1]
1	2	0.700818	223.182071	231.519442	0.695660	[V1, V4]
2	3	0.723775	215.681701	226.798195	0.716569	[V1, V4, V2]
3	4	0.740831	210.097262	223.992879	0.731737	[V1, V4, V2, V3]
4	5	0.749465	208.065195	224.739936	0.738379	[V1, V4, V2, V3, V6]
5	6	0.752083	208.815089	228.268954	0.738802	[V1, V4, V2, V3, V6, V5]
6	7	0.752083	208.815089	228.268954	0.738802	[V1, V4, V2, V3, V6, V5, V7]

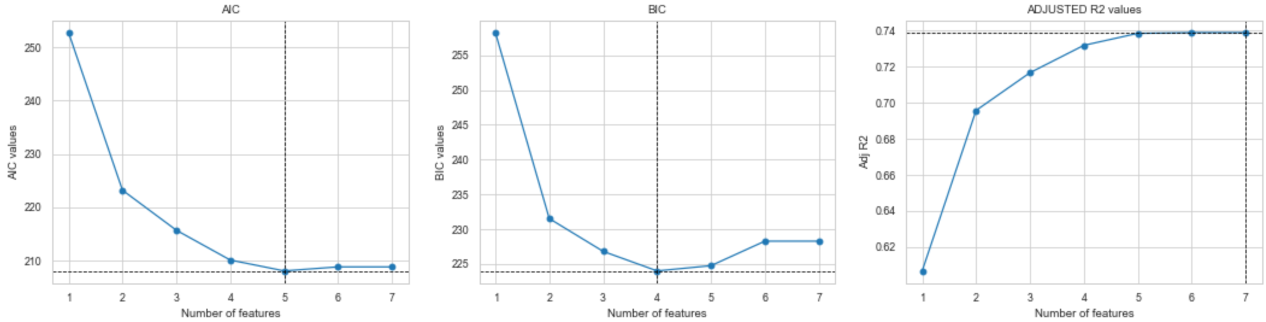
In [53]:

```
print('The best model according to AIC   is model having features - ',
      result_fwd.sort_values('AIC',ascending=True)['numb_features'].values[0])
print('The best model according to BIC   is model having features - ',
      result_fwd.sort_values('BIC',ascending=True)['numb_features'].values[0])
print('The best model according to adjR2 is model having features - ',
      result_fwd.sort_values('adj_r2',ascending=False)['numb_features'].values[0],
     )
```

```
The best model according to AIC   is model having features -  5
The best model according to BIC   is model having features -  4
The best model according to adjR2 is model having features -  7
```

In [54]:

```
plot_results(result_fwd)
```



In [55]:

```
AIC_F = result_fwd.sort_values('AIC',ascending=True)['numb_features'].values[0]
BIC_F = result_fwd.sort_values('BIC',ascending=True)['numb_features'].values[0]
adj_r2_F = result_fwd.sort_values('adj_r2',ascending=False)['numb_features'].values[0]

print('Features choosen by AIC    is',list(result_fwd['features'][AIC_F - 1]))
print('Features choosen by BIC    is',list(result_fwd['features'][BIC_F - 1]))
print('Features choosen by adj_r2 is',list(result_fwd['features'][adj_r2_F - 1]))
```

```
Features choosen by AIC    is ['V1', 'V4', 'V2', 'V3', 'V6']
Features choosen by BIC    is ['V1', 'V4', 'V2', 'V3']
Features choosen by adj_r2 is ['V1', 'V4', 'V2', 'V3', 'V6', 'V5', 'V7']
```

d. Backward stepwise selection

In [56]:

```
def backward_stepwise_selection(data, target):
    features =  [col for col in data.columns if not col == target]
    total_features = []
    list_r2 = []
    list_adj_r2 = []
    list_aic,list_bic = [],[]
    len_features = []
    while(len(features)>0):
        features_with_constant = sm.add_constant(data[features])
        model = sm.OLS(data[target], features_with_constant).fit()
        max_p_value = model.pvalues[1:].max()
        total_features.append(features.copy())
        list_r2.append(model.rsquared)
        list_aic.append(model.aic)
        list_bic.append(model.bic)
        len_features.append(len(total_features[-1]))
        list_adj_r2.append(model.rsquared_adj)
        excluded_feature = model.pvalues[1:].idxmax()
        features.remove(excluded_feature)
    return pd.DataFrame({'numb_features': len_features, 'R_squared':list_r2,
                         'AIC':list_aic,'BIC':list_bic,'adj_r2':list_adj_r2,'features':total_features})
```

In [57]:

```
result_bwd = backward_stepwise_selection(data,'y')
```

In [58]:

```
result_bwd
```

Out[58]:

	numb_features	R_squared	AIC	BIC	adj_r2	features
0	7	0.752083	208.815089	228.268954	0.738802	[V1, V2, V3, V4, V5, V6, V7]
1	6	0.749465	208.065195	224.739936	0.738379	[V1, V2, V3, V4, V6, V7]
2	5	0.740831	210.097262	223.992879	0.731737	[V1, V2, V3, V4, V7]
3	4	0.723775	215.681701	226.798195	0.716569	[V1, V2, V4, V7]
4	3	0.700818	223.182071	231.519442	0.695660	[V1, V4, V7]
5	2	0.700818	223.182071	231.519442	0.695660	[V1, V4]
6	1	0.610053	252.712085	258.270332	0.606721	[V1]

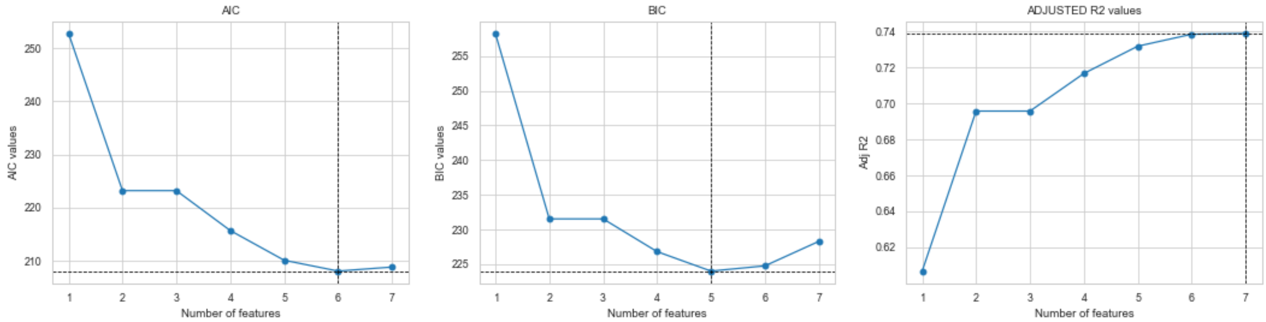
In [59]:

```
print('The best model according to AIC   is model having features - ',
      result_bwd.sort_values('AIC',ascending=True)['numb_features'].values[0])
print('The best model according to BIC   is model having features - ',
      result_bwd.sort_values('BIC',ascending=True)['numb_features'].values[0])
print('The best model according to adjR2 is model having features - ',
      result_bwd.sort_values('adj_r2',ascending=False)['numb_features'].values[0],
     )
```

```
The best model according to AIC   is model having features -  6
The best model according to BIC   is model having features -  5
The best model according to adjR2 is model having features -  7
```

In [60]:

```
plot_results(result_bwd)
```



In [61]:

```
AIC_F = result_bwd.sort_values('AIC',ascending=True)['numb_features'].values[0]
BIC_F = result_bwd.sort_values('BIC',ascending=True)['numb_features'].values[0]
adj_r2_F = result_bwd.sort_values('adj_r2',ascending=False)['numb_features'].values[0]

print('Features choosen by AIC    is',list(result_bwd['features'][AIC_F - 5]))
print('Features choosen by BIC    is',list(result_bwd['features'][BIC_F - 3]))
print('Features choosen by adj_r2 is',list(result_bwd['features'][adj_r2_F - 7]))
```

```
Features choosen by AIC    is ['V1', 'V2', 'V3', 'V4', 'V6', 'V7']
Features choosen by BIC    is ['V1', 'V2', 'V3', 'V4', 'V7']
Features choosen by adj_r2 is ['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7']
```

e. Selection of Model 比較

In [62]:

```
best_subset_AIC = best_subset_results.sort_values('AIC',ascending=True)['AIC'].values[0]
best_subset_BIC = best_subset_results.sort_values('BIC',ascending=True)['BIC'].values[0]
best_subset_adj_r2 = best_subset_results.sort_values('adj_r2',ascending=False)['adj_r2'].values[0]

fwd_AIC = result_fwd.sort_values('AIC',ascending=True)['AIC'].values[0]
fwd_BIC = result_fwd.sort_values('BIC',ascending=True)['BIC'].values[0]
fwd_adj_r2 = result_fwd.sort_values('adj_r2',ascending=False)['adj_r2'].values[0]

bwd_AIC = result_bwd.sort_values('AIC',ascending=True)['AIC'].values[0]
bwd_BIC = result_bwd.sort_values('BIC',ascending=True)['BIC'].values[0]
bwd_adj_r2 = result_bwd.sort_values('adj_r2',ascending=False)['adj_r2'].values[0]
```

In [63]:

```
from tabulate import tabulate
# creating a DataFrame
dict = {'Selection of Model' : ['bsb ', 'fwd ', 'bwd '],
        'AIC' : [best_subset_AIC, fwd_AIC, bwd_AIC],
        'BIC' : [best_subset_BIC, fwd_BIC, bwd_BIC],
        'Adj R Squared' : [best_subset_adj_r2, fwd_adj_r2, bwd_adj_r2]}

dataframe = pd.DataFrame(dict)

# displaying the DataFrame
dataframe.style
print(tabulate(dataframe, headers = 'keys', tablefmt = 'pretty'))
```

```
+---+--------------------+--------------------+--------------------+--------------------+
|   | Selection of Model |        AIC         |        BIC         |   Adj R Squared    |
+---+--------------------+--------------------+--------------------+--------------------+
| 0 |        bsb         |  208.065194584096  | 223.39280449269134 | 0.7388017805999297 |
| 1 |        fwd         | 208.06519458409602 | 223.99287934062863 | 0.7388017805999297 |
| 2 |        bwd         | 208.06519458409596 | 223.9928793406286  | 0.7388017805999297 |
+---+--------------------+--------------------+--------------------+--------------------+
```

Using AIC criterion :

Best subset is model having features is 5 : 'V1', 'V2', 'V3', 'V4', 'V6'

Forward is model having features is 5 : 'V1', 'V4', 'V2', 'V3', 'V6'

Backward is model having features is 6 : 'V1', 'V2', 'V3', 'V4', 'V6', 'V7'

Best subset and Forward are the same.

Using BIC criterion :

Best subset is model having features is 4 : 'V2', 'V3', 'V4', 'V6'

Forward is model having features is 4 : 'V1', 'V4', 'V2', 'V3

Backward is model having features is 5 : 'V1', 'V2', 'V3', 'V4', 'V7'

They are not the same.

Using Adj R Squared criterion :

Best subset is model having features is 7 : 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7'

Forward is model having features is 7 : 'V1', 'V4', 'V2', 'V3', 'V6', 'V5', 'V7'

Backward is model having features is 7 : 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7'

They are the same.

因此，
AIC最小的選擇是Best subset，the model is having features is 5 : 'V1', 'V2', 'V3', 'V4', 'V6';
BIC最小的選擇是Best subset，the model is having features is 4 : 'V2', 'V3', 'V4', 'V6';
Adj R Squared最大的選擇是都一樣，the model is having features is 7 : 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7'

f. Best subset using AIC (第四次重新建模)

In [64]:

```
fit5 = smf.ols('LadderScore~ LoggedGDPPerCapita + SocialSupport + HealthyLifeExpectancy + FreedomToMakeLifeChoices + \
               PerceptionsOfCorruption'
               , data = Train.drop('LadderScoreInDystopia', axis = 1)).fit()

fit5.summary()
```

Out[64]:

OLS Regression Results

Dep. Variable:	LadderScore	**R-squared:**	0.749
Model:	OLS	**Adj. R-squared:**	0.738
Method:	Least Squares	**F-statistic:**	67.61
Date:	Tue, 13 Dec 2022	**Prob (F-statistic):**	2.35e-32
Time:	02:26:52	**Log-Likelihood:**	-98.033
No. Observations:	119	**AIC:**	208.1
Df Residuals:	113	**BIC:**	224.7
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-2.3808	0.734	-3.244	0.002	-3.835	-0.927
LoggedGDPPerCapita	0.1996	0.110	1.818	0.072	-0.018	0.417
SocialSupport	2.9098	0.797	3.649	0.000	1.330	4.490
HealthyLifeExpectancy	0.0413	0.016	2.623	0.010	0.010	0.072
FreedomToMakeLifeChoices	1.8121	0.559	3.241	0.002	0.704	2.920
PerceptionsOfCorruption	-0.6498	0.329	-1.973	0.051	-1.302	0.003

Omnibus:	8.416	**Durbin-Watson:**	2.282
Prob(Omnibus):	0.015	**Jarque-Bera (JB):**	8.142
Skew:	-0.606	**Prob(JB):**	0.0171
Kurtosis:	3.419	**Cond. No.**	1.21e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.21e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

g. Best subset using BIC (第五次重新建模)

In [65]:

```
fit6 = smf.ols('LadderScore~ SocialSupport + HealthyLifeExpectancy + FreedomToMakeLifeChoices + PerceptionsOfCorruption'
               , data = Train.drop('LoggedGDPPerCapita', axis = 1)).fit()

fit6.summary()
```

Out[65]:

OLS Regression Results

Dep. Variable:	LadderScore	**R-squared:**	0.742
Model:	OLS	**Adj. R-squared:**	0.733
Method:	Least Squares	**F-statistic:**	82.02
Date:	Tue, 13 Dec 2022	**Prob (F-statistic):**	1.22e-32
Time:	02:26:52	**Log-Likelihood:**	-99.749
No. Observations:	119	**AIC:**	209.5
Df Residuals:	114	**BIC:**	223.4
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-2.1557	0.731	-2.951	0.004	-3.603	-0.708
SocialSupport	3.7287	0.665	5.610	0.000	2.412	5.045
HealthyLifeExpectancy	0.0607	0.012	5.195	0.000	0.038	0.084
FreedomToMakeLifeChoices	1.5791	0.550	2.872	0.005	0.490	2.668
PerceptionsOfCorruption	-0.7708	0.326	-2.366	0.020	-1.416	-0.125

Omnibus:	6.199	**Durbin-Watson:**	2.225
Prob(Omnibus):	0.045	**Jarque-Bera (JB):**	5.773
Skew:	-0.524	**Prob(JB):**	0.0558
Kurtosis:	3.257	**Cond. No.**	1.08e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.08e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

f. 第四、五次重新建模與上述3種模型做比較

In [66]:

```
y_predict5= fit5.predict(exog = Test)

mae5 = mean_absolute_error(Test['LadderScore'],y_predict5)
mse5 = mean_squared_error(Test['LadderScore'],y_predict5)
rmse5 = sqrt(mean_squared_error(Test['LadderScore'],y_predict5))
r2_score5 = r2_score(Test['LadderScore'],y_predict5)
```

In [67]:

```
y_predict6= fit6.predict(exog = Test)

mae6 = mean_absolute_error(Test['LadderScore'],y_predict6)
mse6 = mean_squared_error(Test['LadderScore'],y_predict6)
rmse6 = sqrt(mean_squared_error(Test['LadderScore'],y_predict6))
r2_score6 = r2_score(Test['LadderScore'],y_predict6)
```

In [68]:

```
fit5 = smf.ols('LadderScore~ LoggedGDPPerCapita + SocialSupport + HealthyLifeExpectancy + FreedomToMakeLifeChoices + \
                PerceptionsOfCorruption'
                , data = Train.drop('LadderScoreInDystopia', axis = 1)).fit()

result5 = sm.stats.anova_lm(fit5, type=2)
print('第四次重新建模')
print(result5)
```

```
第四次重新建模
                             df     sum_sq    mean_sq           F  \
LoggedGDPPerCapita          1.0  88.128918  88.128918  275.155235
SocialSupport               1.0   8.650668   8.650668   27.009031
HealthyLifeExpectancy       1.0   4.597534   4.597534   14.354375
FreedomToMakeLifeChoices    1.0   5.644001   5.644001   17.621643
PerceptionsOfCorruption     1.0   1.247321   1.247321    3.894374
Residual                  113.0  36.192543   0.320288         NaN

                                PR(>F)
LoggedGDPPerCapita        4.653080e-32
SocialSupport             9.084253e-07
HealthyLifeExpectancy     2.444682e-04
FreedomToMakeLifeChoices  5.394397e-05
PerceptionsOfCorruption   5.088927e-02
Residual                           NaN
```

In [69]:

```
fit6 = smf.ols('LadderScore~ SocialSupport + HealthyLifeExpectancy + FreedomToMakeLifeChoices + PerceptionsOfCorruption'
               , data = Train.drop('LoggedGDPPerCapita', axis = 1)).fit()

result6 = sm.stats.anova_lm(fit6, type=2)
print('第五次重新建模')
print(result4)
```

```
第五次重新建模
                             df     sum_sq    mean_sq           F  \
LoggedGDPPerCapita          1.0  39.479700  39.479700  210.213134
SocialSupport               1.0   4.534560   4.534560   24.144663
HealthyLifeExpectancy       1.0   0.930714   0.930714    4.955666
FreedomToMakeLifeChoices    1.0   6.595014   6.595014   35.115733
Generosity                  1.0   0.380142   0.380142    2.024097
PerceptionsOfCorruption     1.0   2.201754   2.201754   11.723433
LadderScoreInDystopia       1.0   0.434498   0.434498    2.313524
Residual                  105.0  19.719836   0.187808         NaN

                                PR(>F)
LoggedGDPPerCapita        8.165657e-27
SocialSupport             3.305200e-06
HealthyLifeExpectancy     2.814180e-02
FreedomToMakeLifeChoices  3.992819e-08
Generosity                1.577851e-01
PerceptionsOfCorruption   8.813401e-04
LadderScoreInDystopia     1.312588e-01
Residual                           NaN
```

8、6種模型做比較

In [70]:

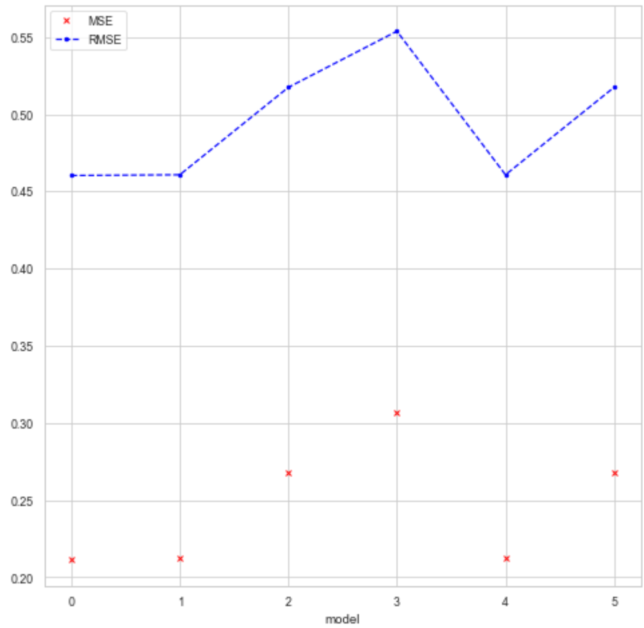
```
from tabulate import tabulate
# creating a DataFrame
dict = {'Model' : ['一開始建立模型', '第一次重新建模', '第二次重新建模', '第三次重新建模', '第四次重新建模', '第五次重新建模'],
        'MSE' : [mse1, mse2, mse3, mse4, mse5, mse6],
        'RMSE' : [rmse1, rmse2, rmse3, rmse4, rmse5, rmse6],
        'R2_score' : [r2_score1, r2_score2, r2_score3, r2_score4, r2_score5, r2_score6],
        'AIC' : [fit1.aic, fit2.aic, fit3.aic, fit4.aic, fit5.aic, fit6.aic],
        'BIC' : [fit1.bic, fit2.bic, fit3.bic, fit4.bic, fit5.bic, fit6.bic]}

dataframe = pd.DataFrame(dict)

# displaying the DataFrame
dataframe.style
print(tabulate(dataframe, headers = 'keys', tablefmt = 'pretty'))
```

	Model	MSE	RMSE	R2_score	AIC	BIC
0	一開始建立模型	0.21186796214884354	0.4602911710524584	0.7547648927487713	208.81508934894586	228.26895380072656
1	第一次重新建模	0.2122926966195769	0.46075231591341664	0.7542732667264735	208.06519458409602	224.73993554276518
2	第二次重新建模	0.2676812064772173	0.5173791708961787	0.6901616048326304	209.49718702713366	223.3928044926913
3	第三次重新建模	0.30662190512232085	0.553734507794413	0.6069360030908807	97.99555405655951	115.4160086446845
4	第四次重新建模	0.21229269661957717	0.4607523159134169	0.7542732667264732	208.06519458409602	224.73993554276518
5	第五次重新建模	0.2676812064772171	0.5173791708961786	0.6901616048326308	209.4971870271337	223.39280449269134

= > 通過這6個模型可知， 一開始建立的模型會更理想一點，具體表現為: RMSE是最小，r2_score是最大的。
而第四次重新建模的模型(Best subset using AIC)會是最接近一開始建立的模型的。

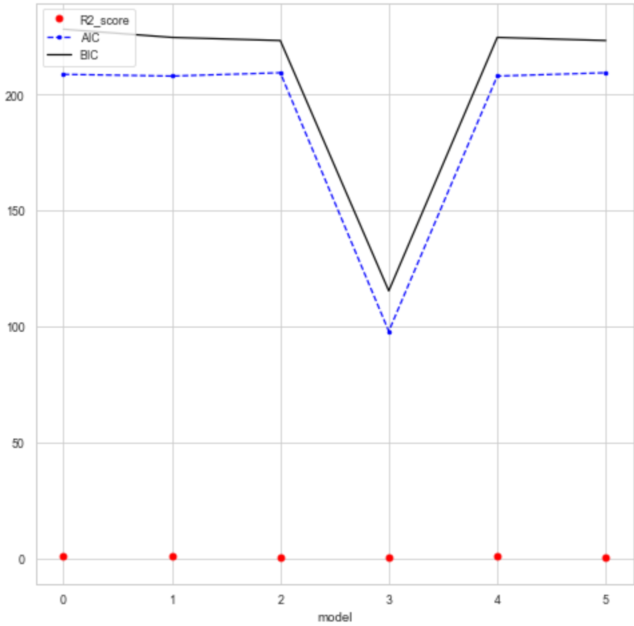
In [71]:

```
#r代表紅色,x代表用'x來表示點,不畫線
plt.figure(figsize=(8,8))
plt.plot([mse1, mse2, mse3, mse4, mse5, mse6], 'rx')
#b代表藍色 ,代表用單個小點表示一個點,表示用虛線(dashedie線
plt.plot([rmse1, rmse2, rmse3, rmse4, rmse5, rmse6],
'b.--')

plt.legend(('MSE', 'RMSE'), loc='upper left')#畫圖例及決定位置
plt.xlabel('model')
plt.grid(True)# 出網格
plt.show()
```



In [72]:

```
#r代表紅色,x代表用'x來表示點,不畫線
plt.figure(figsize=(8,8))
plt.plot([r2_score1, r2_score2, r2_score3, r2_score4, r2_score5, r2_score6], 'ro')
#b代表藍色 ,代表用單個小點表示一個點,表示用虛線(dashedie線
plt.plot([fit1.aic, fit2.aic, fit3.aic, fit4.aic, fit5.aic, fit6.aic],
'b.--')
plt.plot([fit1.bic, fit2.bic, fit3.bic, fit4.bic, fit5.bic, fit6.bic],
'k-')

plt.legend(('R2_score', 'AIC', 'BIC'), loc='upper left')#畫圖例及決定位置
plt.xlabel('model')
plt.grid(True)# 出網格
plt.show()
```



a. 原始資料VS一開始建立模型VS第四次重新建模的模型(Best subset using AIC)

In [73]:

```
results.params
```

Out[73]:

```
LoggedGDPPerCapita          0.279533
SocialSupport               2.476206
HealthyLifeExpectancy       0.030314
FreedomToMakeLifeChoices    2.010465
Generosity                  0.364382
PerceptionsOfCorruption    -0.605092
LadderScoreInDystopia      -0.920666
dtype: float64
```

In [74]:

```
fit1.params
```

Out[74]:

```
Intercept                  -0.365821
LoggedGDPPerCapita          0.218454
SocialSupport               2.847445
HealthyLifeExpectancy       0.042376
FreedomToMakeLifeChoices    1.699141
Generosity                  0.389092
PerceptionsOfCorruption    -0.592698
LadderScoreInDystopia      -0.888946
dtype: float64
```

$$\begin{gathered}LadderScore = -0.365821 + LoggedGDPPerCapita * 0.218454\\+ SocialSupport * 2.847445 + HealthyLifeExpectancy * 0.042376\\+ FreedomToMakeLifeChoices * 1.699141 + Generosity * 0.389092\\+ PerceptionsOfCorruption * (-0.592698) + LadderScoreInDystopia * (-0.888946)\end{gathered}$$

In [75]:

```
fit5.params
```

Out[75]:

```
Intercept                  -2.380819
LoggedGDPPerCapita          0.199556
SocialSupport               2.909797
HealthyLifeExpectancy       0.041288
FreedomToMakeLifeChoices    1.812111
PerceptionsOfCorruption    -0.649820
dtype: float64
```

$$LadderScore = -2.380819 + LoggedGDPPerCapita * 0.199556 \\ + SocialSupport * 2.909797 + HealthyLifeExpectancy * 0.041288 \\ + FreedomToMakeLifeChoices * 1.812111 + PerceptionsOfCorruption * (-0.649820)$$

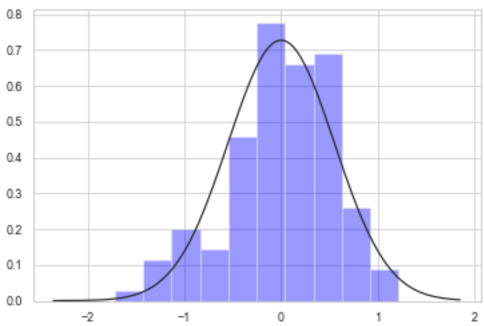
9、殘差診斷(一開始建立模型VS第四次重新建模的模型)

a. KS檢驗

In [76]:

```
residual1 = fit1.resid

sns.distplot(residual1,

             bins = 10,

             kde = False,

             color = 'blue',

             fit = stats.norm)

plt.show()
```

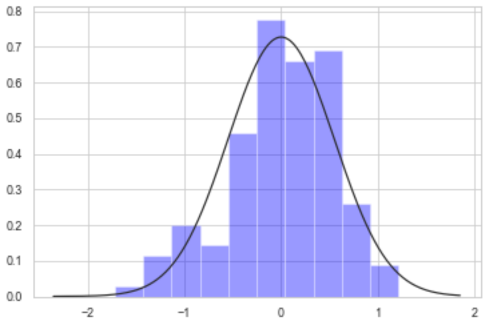
```
C:\Users\willy\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated functio
n and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with sim
ilar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [77]:

```
residual5 = fit5.resid
sns.distplot(residual1,
             bins = 10,
             kde = False,
             color = 'blue',
             fit = stats.norm)
plt.show()
```

```
C:\Users\willy\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated functio
n and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with sim
ilar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

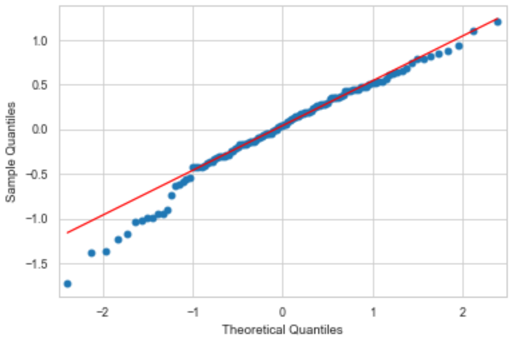


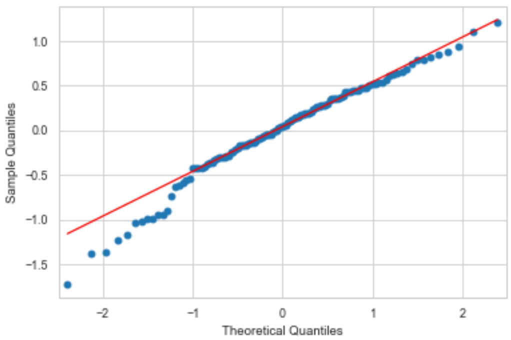
b. QQ圖

In [78]:

```
pq = sm.ProbPlot(residual1)
pq.qqplot(line='q')
```

Out[78]:

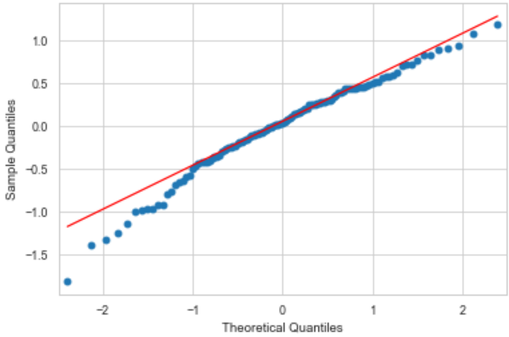


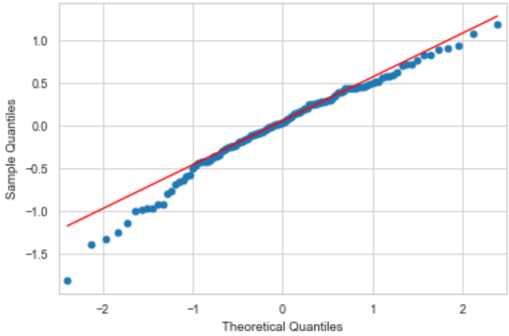


In [79]:

```
pq = sm.ProbPlot(residual5)

pq.qqplot(line='q')
```

Out[79]:

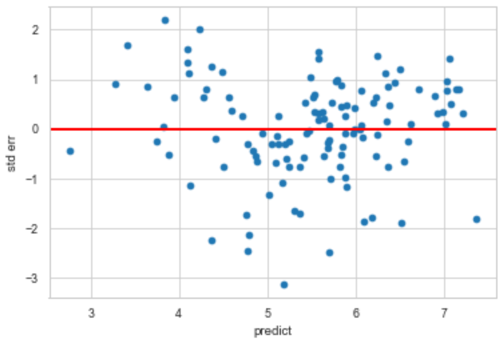




10、標準化殘差檢定圖(一開始建立模型VS第四次重新建模的模型)

In [80]:

```
#標準化殘差與預測值之間的散點圖
plt.scatter(fit1.predict(), (fit1.resid-fit1.resid.mean())/fit1.resid.std())
plt.xlabel('predict')
plt.ylabel('std err')
#添加水平參考線
plt.axhline(y = 0, color = 'r',linewidth = 2)

plt.show()
```



In [81]:

```
#標準化殘差與預測值之間的散點圖
plt.scatter(fit5.predict(), (fit5.resid-fit5.resid.mean())/fit5.resid.std())
plt.xlabel('predict')
plt.ylabel('std err')
#添加水平參考線
plt.axhline(y = 0, color = 'r',linewidth = 2)

plt.show()
```