In [42]:

```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
from sklearn import metrics
from IPython.display import display
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set(style='darkgrid',context='notebook',font_scale=1.5) # 设置背景
from pylab import *
mpl.rcParams['font.sans-serif'] = ['SimHei']#显示中文
import statsmodels.api as sm
import statsmodels.formula.api as smf
# 这部分是超参数提前设置
sns.set(style='darkgrid')
warnings.filterwarnings('ignore')
```

1 数据准备

In [2]:

```
df = pd.read_excel('Flat prices_after_processing.xlsx',index_col=[0])
df.head()
```

Out[2]:

	month	town	flat_type	storey_range	floor_area_sqm	flat_model	lease_commence_date	rer
0	2022-01-17	0	1	11	44.0	3	1979	
1	2022-01-17	0	2	2	67.0	10	1978	
2	2022-01-17	0	2	2	67.0	10	1980	
3	2022-01-17	0	2	5	68.0	10	1980	
4	2022-01-17	0	2	2	67.0	10	1980	

2 描述性统计

In [3]:

```
print (f"Train has {df.shape[0]} rows and {df.shape[1]} columns")
```

```
Train has 42070 rows and 9 columns
```

In [4]:

```
df.describe().T
```

Out[4]:

	count	mean	std	min	25%	50%
town	42070.0	13.829784	7.972728	0.0	8.0	14.0
flat_type	42070.0	3.150131	0.925828	0.0	2.0	3.0
storey_range	42070.0	8.666675	5.810077	2.0	5.0	8.0
floor_area_sqm	42070.0	98.108907	24.207332	31.0	82.0	96.0
flat_model	42070.0	6.746446	3.712077	0.0	3.0	6.0
lease_commence_date	42070.0	1993.140789	12.027778	1966.0	1984.0	1993.0
remaining_lease	42070.0	895.236748	144.542941	565.0	789.0	895.0
resale_price	42070.0	442552.703682	153525.477850	160000.0	332000.0	410000.0	51

In [5]:

```
df.describe()
```

Out[5]:

	town	flat_type	storey_range	floor_area_sqm	flat_model	lease_commen
count	42070.000000	42070.000000	42070.000000	42070.000000	42070.000000	42070
mean	13.829784	3.150131	8.666675	98.108907	6.746446	1993
std	7.972728	0.925828	5.810077	24.207332	3.712077	12
min	0.000000	0.000000	2.000000	31.000000	0.000000	1966
25%	8.000000	2.000000	5.000000	82.000000	3.000000	1984
50%	14.000000	3.000000	8.000000	96.000000	6.000000	1993
75%	21.000000	4.000000	11.000000	113.000000	10.000000	2002
max	25.000000	6.000000	50.000000	249.000000	18.000000	2016

In [6]:

```
df.info()
```

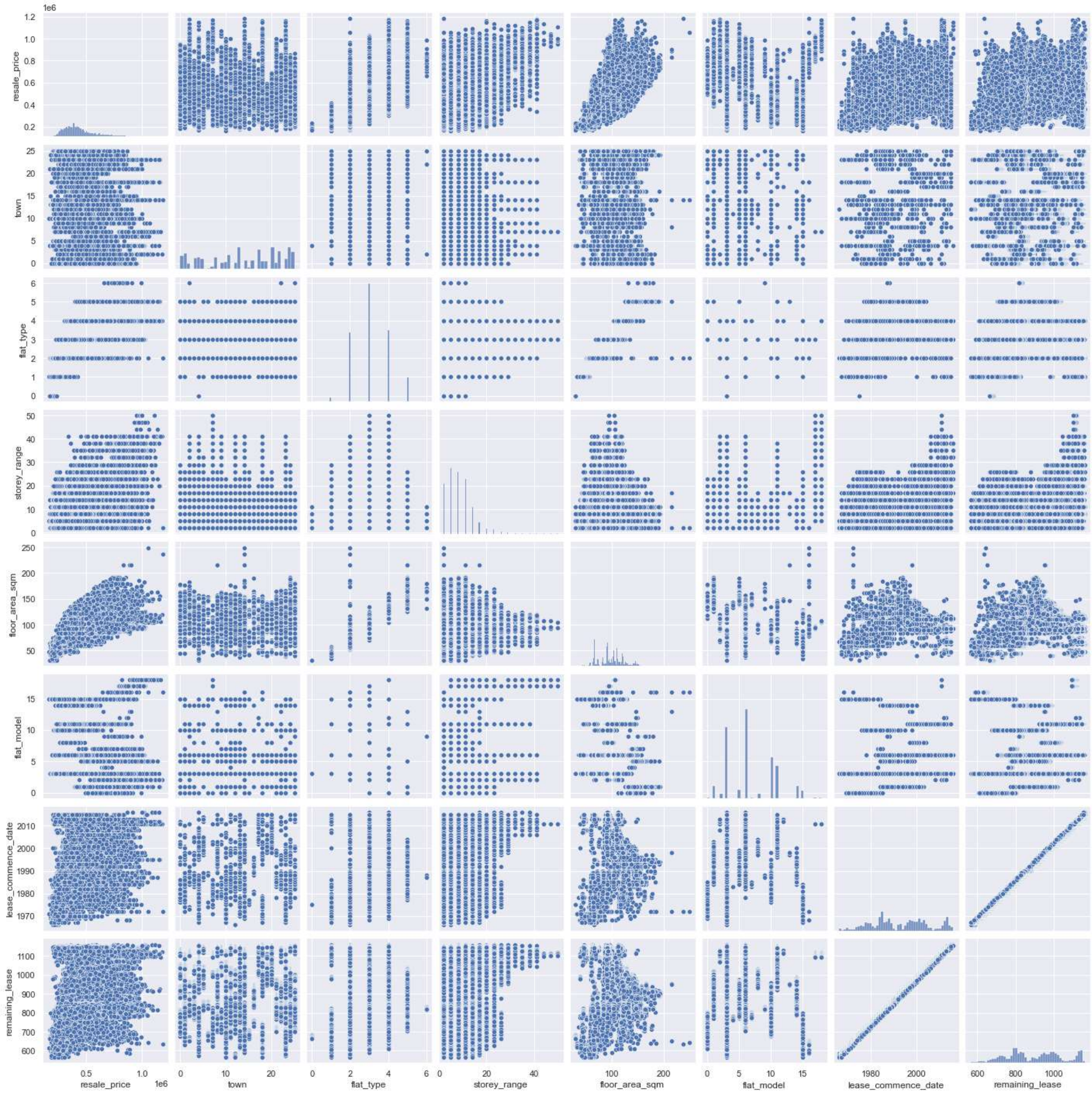
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 42070 entries, 0 to 42069
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   month                42070 non-null  datetime64[ns]
 1   town                 42070 non-null  int64
 2   flat_type            42070 non-null  int64
 3   storey_range         42070 non-null  int64
 4   floor_area_sqm       42070 non-null  float64
 5   flat_model           42070 non-null  int64
 6   lease_commence_date  42070 non-null  int64
 7   remaining_lease      42070 non-null  int64
 8   resale_price         42070 non-null  float64
dtypes: datetime64[ns](1), float64(2), int64(6)
memory usage: 3.2 MB
```

3 相关性

3.1 两两变量之间的散点图

In [8]:

```
sns.set()
cols = ['resale_price','town','flat_type','storey_range','floor_area_sqm','flat_model','lease_commen
sns.pairplot(df[cols], size = 2.5)
plt.show()
```



3.2 皮尔森相关系数

In [9]:

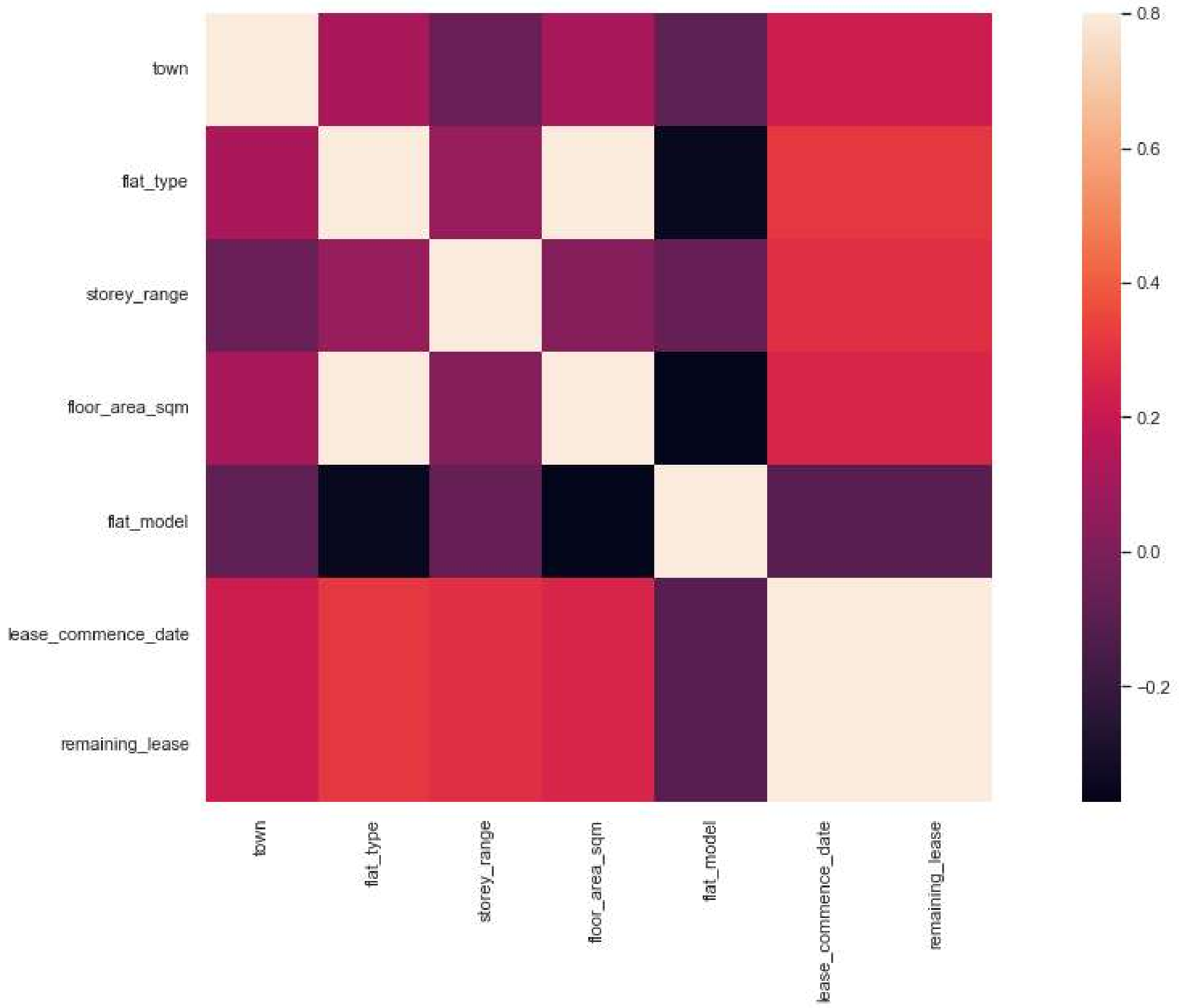
```
df2=df[['town','flat_type','storey_range','floor_area_sqm','flat_model','lease_commence_date','remai
```

In [10]:

```
corrmat = df2.corr()
f, ax = plt.subplots(figsize=(20, 9))
sns.heatmap(corrmat, vmax=0.8, square=True)
```

Out[10]:

<AxesSubplot:>



In [15]:

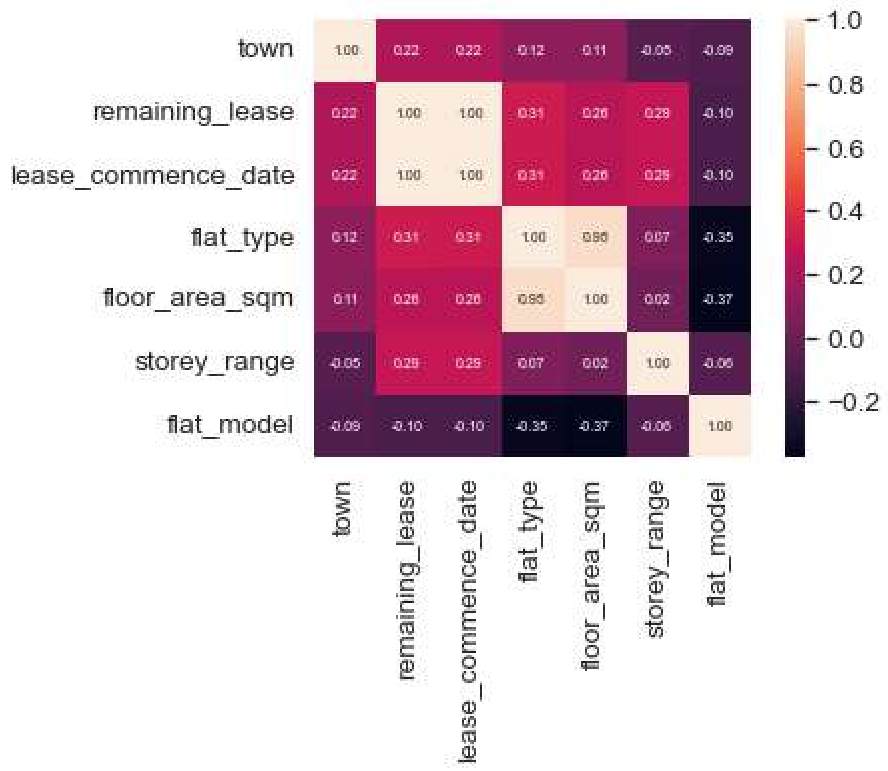
```
#多重共线性：简单相关系数法
#皮尔森系数
corr_p=df2.corr(method='pearson')
corr_p
```

Out[15]:

	town	flat_type	storey_range	floor_area_sqm	flat_model	lease_co
town	1.000000	0.124476	-0.053850	0.114799	-0.085075	
flat_type	0.124476	1.000000	0.068306	0.948714	-0.347036	
storey_range	-0.053850	0.068306	1.000000	0.024992	-0.062983	
floor_area_sqm	0.114799	0.948714	0.024992	1.000000	-0.372321	
flat_model	-0.085075	-0.347036	-0.062983	-0.372321	1.000000	
lease_commence_date	0.222382	0.313356	0.290303	0.257753	-0.103717	
remaining_lease	0.222395	0.314535	0.289421	0.259276	-0.103679	

In [14]:

```
k  = 7 # 关系矩阵中将显示7个特征
cols = corrmat.nlargest(k, 'town')['town'].index
cm = np.corrcoef(df[cols].values.T)
sns.set(font_scale=1.25)
hm = sns.heatmap(cm, cbar=True, annot=True, \
                 square=True, fmt='.2f', annot_kws={'size': 7}, yticklabels=cols.values, xticklabel
plt.show()
```



4 回归分析

4.1 多元线性回归1

4.1.1 多元线性回归

In [17]:

```
#多元线性回归1
from statsmodels.stats.anova import anova_lm
from statsmodels.formula.api import ols
import pandas as pd
model = ols("resale_price~town+flat_type+storey_range+floor_area_sqm+flat_model+lease_commence_date+
data = model.fit()
data.summary()
```

Out[17]:

OLS Regression Results

Dep. Variable:	resale_price	**R-squared:**	0.563
Model:	OLS	**Adj. R-squared:**	0.562
Method:	Least Squares	**F-statistic:**	7726.
Date:	Sat, 26 Nov 2022	**Prob (F-statistic):**	0.00
Time:	23:02:18	**Log-Likelihood:**	-5.4469e+05
No. Observations:	42070	**AIC:**	1.089e+06
Df Residuals:	42062	**BIC:**	1.089e+06
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.844e+06	1.5e+06	1.226	0.220	-1.11e+06	4.79e+06
town	-3106.7664	64.413	-48.232	0.000	-3233.017	-2980.516
flat_type	2.501e+04	1742.966	14.349	0.000	2.16e+04	2.84e+04
storey_range	8313.5399	90.380	91.985	0.000	8136.394	8490.686
floor_area_sqm	2939.0644	66.330	44.310	0.000	2809.057	3069.072
flat_model	-831.9114	144.238	-5.768	0.000	-1114.620	-549.203
lease_commence_date	-988.6295	784.212	-1.261	0.207	-2525.701	548.442
remaining_lease	199.0320	65.254	3.050	0.002	71.132	326.932

Omnibus:	7002.999	**Durbin-Watson:**	0.472
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	12503.401
Skew:	1.070	**Prob(JB):**	0.00
Kurtosis:	4.598	**Cond. No.**	6.65e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.65e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

4.1.2 多重共线性分析

In [16]:

```
#多重共线性：计算方差膨胀因子VIF
import numpy as np
import pandas as pd
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif = pd.DataFrame()
vif["VIF Factor"] = [variance_inflation_factor(df2.values, i) for i in range(df2.shape[1])]
vif["features"] = df2.columns
vif
```

Out[16]:

	VIF Factor	features
0	4.312358	town
1	133.536980	flat_type
2	3.625558	storey_range
3	183.115463	floor_area_sqm
4	5.031079	flat_model
5	72.015936	lease_commence_date
6	53.501651	remaining_lease

4.1.3 方差分析

In [19]:

```
anova_lm(data)#方差分析
```

Out[19]:

	df	sum_sq	mean_sq	F	PR(>F)
town	1.0	6.369100e+12	6.369100e+12	617.544984	2.430793e-135
flat_type	1.0	4.170694e+14	4.170694e+14	40438.860553	0.000000e+00
storey_range	1.0	1.055563e+14	1.055563e+14	10234.687864	0.000000e+00
floor_area_sqm	1.0	1.908425e+13	1.908425e+13	1850.399775	0.000000e+00
flat_model	1.0	2.883590e+11	2.883590e+11	27.959161	1.245205e-07
lease_commence_date	1.0	9.296236e+12	9.296236e+12	901.358847	5.870707e-196
remaining_lease	1.0	9.594828e+10	9.594828e+10	9.303102	2.289085e-03
Residual	42062.0	4.338098e+14	1.031358e+10	NaN	NaN

4.2 多元线性回归2

4.2.1 多元线性回归（变量合并后）

In [18]:

```
#多元线性回归2
from statsmodels.stats.anova import anova_lm
from statsmodels.formula.api import ols
import pandas as pd
model2 = ols("resale_price~town+flat_type+storey_range+flat_model+remaining_lease", data=df)
data2 = model2.fit()
data2.summary()
```

Out[18]:

OLS Regression Results

Dep. Variable:	resale_price	**R-squared:**	0.542
Model:	OLS	**Adj. R-squared:**	0.542
Method:	Least Squares	**F-statistic:**	9956.
Date:	Sat, 26 Nov 2022	**Prob (F-statistic):**	0.00
Time:	23:05:21	**Log-Likelihood:**	-5.4565e+05
No. Observations:	42070	**AIC:**	1.091e+06
Df Residuals:	42064	**BIC:**	1.091e+06
Df Model:	5		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	3.183e+04	3649.667	8.720	0.000	2.47e+04	3.9e+04
town	-3120.5258	65.898	-47.354	0.000	-3249.688	-2991.364
flat_type	9.757e+04	612.159	159.386	0.000	9.64e+04	9.88e+04
storey_range	7904.0087	91.967	85.944	0.000	7723.752	8084.266
flat_model	-1795.2308	145.873	-12.307	0.000	-2081.145	-1509.316
remaining_lease	100.6838	3.950	25.492	0.000	92.943	108.425

Omnibus:	7560.545	**Durbin-Watson:**	0.487
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	15167.482
Skew:	1.087	**Prob(JB):**	0.00
Kurtosis:	4.981	**Cond. No.**	6.55e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.55e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

4.2.2 多重共线性分析

In [23]:

```
#多重共线性：计算方差膨胀因子VIF
import numpy as np
import pandas as pd
from statsmodels.stats.outliers_influence import variance_inflation_factor
df3=df[['town','flat_type','storey_range','flat_model','remaining_lease']]
vif = pd.DataFrame()
vif["VIF Factor"] = [variance_inflation_factor(df3.values, i) for i in range(df3.shape[1])]
vif["features"] = df3.columns
vif
```

Out[23]:

	VIF Factor	features
0	4.298114	town
1	13.884428	flat_type
2	3.587749	storey_range
3	3.963618	flat_model
4	26.182656	remaining_lease

4.2.3 方差分析

In [20]:

```
anova_lm(data2)#方差分析
```

Out[20]:

	df	sum_sq	mean_sq	F	PR(>F)
town	1.0	6.369100e+12	6.369100e+12	589.953172	2.005214e-129
flat_type	1.0	4.170694e+14	4.170694e+14	38632.058677	0.000000e+00
storey_range	1.0	1.055563e+14	1.055563e+14	9777.403633	0.000000e+00
flat_model	1.0	1.438257e+12	1.438257e+12	133.222008	8.996175e-31
remaining_lease	1.0	7.015876e+12	7.015876e+12	649.862377	2.892975e-142
Residual	42064.0	4.541205e+14	1.079594e+10	NaN	NaN

4.3 残差分析

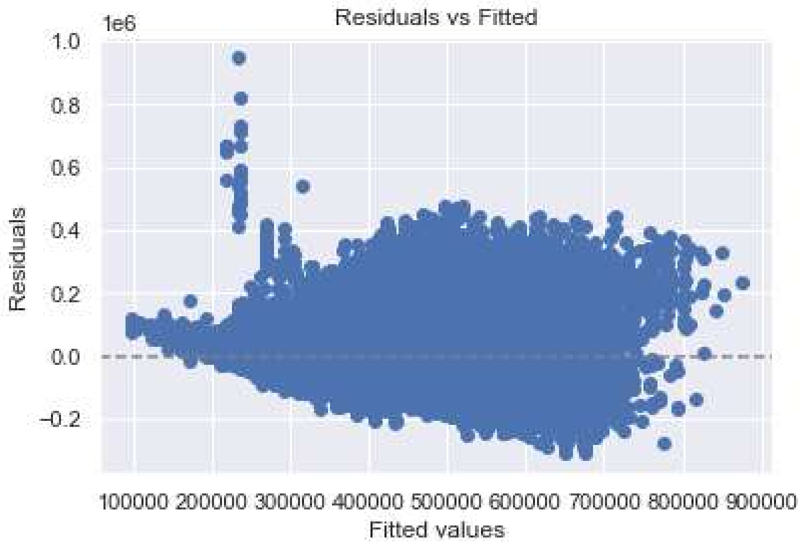
In [29]:

```
results = pd.DataFrame({'index': df['resale_price'], # y实际值
             'resids': data2.resid, # 残差
             'std_resids':data2.resid_pearson, # 方差标准化的残差
             'fitted': data2.predict() # y预测值
             })
```

4.3.1 残差拟合图

In [44]:

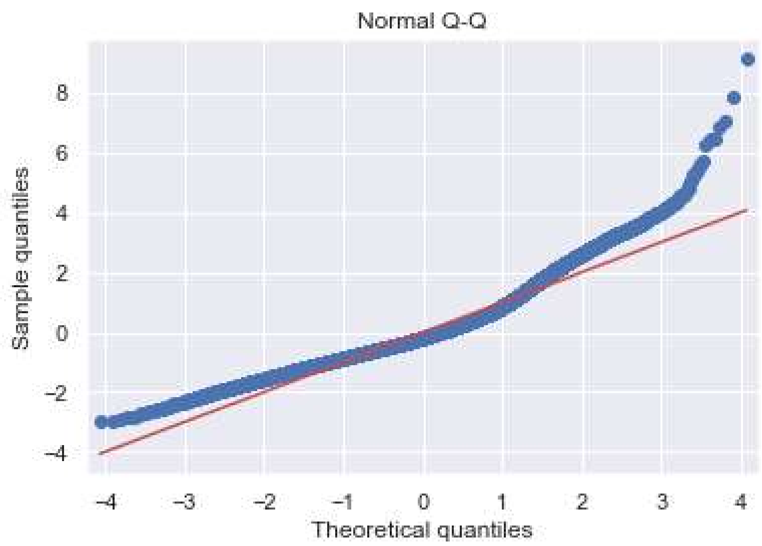
```
# 残差拟合图：横坐标是拟合值，纵坐标是残差。
residsvfitted = plt.plot(results['fitted'], results['resids'],  'o')
l = plt.axhline(y = 0, color = 'grey', linestyle = 'dashed') # 绘制y=0水平线
plt.xlabel('Fitted values')
plt.ylabel('Residuals')
plt.title('Residuals vs Fitted')
plt.show(residsvfitted)
```



4.3.2 残差QQ图

In [43]:

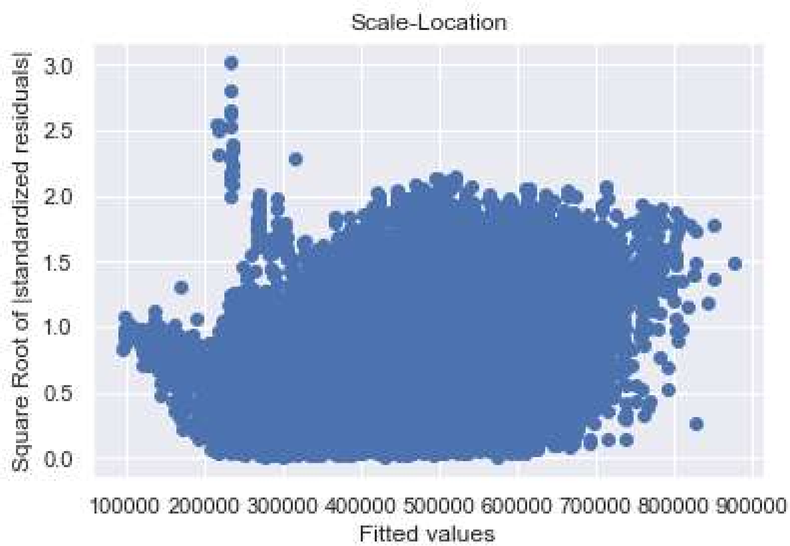
```
## q-q plot
# 残差QQ图：用来描述残差是否符合正态分布。
qqplot = sm.qqplot(results['std_resids'], line='s')
plt.xlabel('Theoretical quantiles')
plt.ylabel('Sample quantiles')
plt.title('Normal Q-Q')
plt.show(qqplot)
```



4.3.3 标准化的残差对拟合值图

In [45]:

```
## scale-location
# 标准化的残差对拟合值：对标准化残差平方根和拟合值作图，横坐标是拟合值，纵坐标是标准化后的残差平方根
scalelocplot = plt.plot(results['fitted'], abs(results['std_resids'])**.5,  'o')
plt.xlabel('Fitted values')
plt.ylabel('Square Root of |standardized residuals|')
plt.title('Scale-Location')
plt.show(scalelocplot)
```



In [41]:

```
## residuals vs. leverage
# 标准化残差对杠杆值:通常用Cook距离度量的回归影响点。
residsvlevplot = sm.graphics.influence_plot(data2, criterion = 'Cooks', size = 0.1)
plt.xlabel('Obs.number')
plt.ylabel("Cook's distance")
plt.title("Cook's distance")
plt.show(residsvlevplot)
plt.close()
```