In [1]:
```
import import_ipynb
from Seline import MLR, mclp, plot_input, plot_result
import pandas as pd

from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns

import numpy as np
```

```
importing Jupyter notebook from Seline.ipynb
```

Load Dataset

In [2]:
```
raw_demand = pd.read_csv('dataset/수요지_데이터셋.csv', index_col=0)
raw_candidate = pd.read_csv('dataset/후보지_정렬2000.csv', index_col=0)
```

In [3]:
```
raw_demand
```

Out[3]:

	cell_x	cell_y	car	population	houses	houses_parking	charger_count	charg
0	127.030786	37.362273	20322.0	13	0	0	0	
1	127.030798	37.360471	20322.0	0	0	0	0	
2	127.031927	37.360475	20322.0	21	0	0	0	
3	127.031932	37.359574	20322.0	18	0	0	0	
4	127.034124	37.370399	20322.0	0	0	0	0	
...	...	...	...	...	...	...	...	
10824	127.421737	37.147743	3607.0	0	0	0	0	
10825	127.421738	37.146841	3607.0	0	0	0	0	
10826	127.421742	37.143236	3607.0	0	0	0	0	
10827	127.421743	37.142334	3607.0	0	0	0	0	
10828	127.424004	37.133321	3607.0	0	0	0	0	

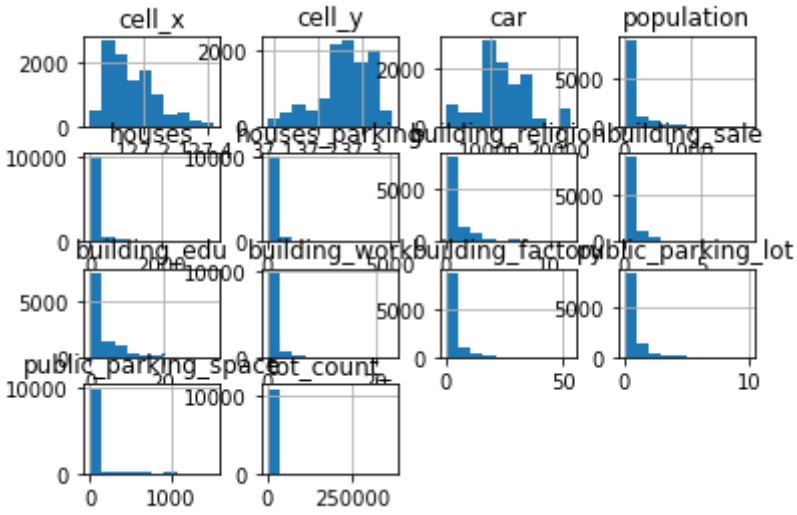
10829 rows × 16 columns

In [4]:
```
raw_lin = raw_demand.drop(["charger_count","charger_value"],axis= 1)
```

In [5]:
```
raw_lin.hist()
raw_lin.describe()
```

Out[5]:

	cell_x	cell_y	car	population	houses	houses_parking	bu
count	10829.000000	10829.000000	10829.000000	10829.000000	10829.000000	10829.000000	
mean	127.165474	37.262857	11842.889379	91.210730	72.585927	93.465971	
std	0.077720	0.058154	3868.637828	176.674292	244.714359	331.130764	
min	127.030786	37.086901	3607.000000	0.000000	0.000000	0.000000	
25%	127.103806	37.233667	9713.000000	0.000000	0.000000	0.000000	
50%	127.147760	37.270556	12192.000000	0.000000	0.000000	0.000000	
75%	127.216422	37.308362	14200.000000	95.000000	0.000000	0.000000	
max	127.424004	37.370399	21707.000000	1876.000000	3511.000000	5082.000000	



In [6]:
```
raw_candidate
```

Out[6]:

	cell_x	cell_y	cnt_cust*charger_val
0	127.075577	37.229037	345469.765400
1	127.072773	37.326377	281137.608500
2	127.078416	37.326397	157946.033400
3	127.083006	37.312893	138673.965000
4	127.089601	37.345366	114740.448000
...	...	...	...
1995	127.124828	37.299512	591.654180
1996	127.127000	37.316645	590.826000
1997	127.132574	37.098522	590.358000
1998	127.131695	37.279703	590.295343
1999	127.131462	37.327476	589.415447

2000 rows × 3 columns

전처리 & 입지선정지수

In [7]:
```
scaler = StandardScaler()
raw_lin.iloc[:,2:] = scaler.fit_transform(raw_lin.iloc[:,2:])
# idx_x = idx_scaled[:,:-1]
# idx_y = idx_scaled[:,-1]
```

In [8]:
```
raw_lin
```

Out[8]:

	cell_x	cell_y	car	population	houses	houses_parking	building_religion
0	127.030786	37.362273	2.191857	-0.442704	-0.296629	-0.282276	0.006470
1	127.030798	37.360471	2.191857	-0.516289	-0.296629	-0.282276	0.006470
2	127.031927	37.360475	2.191857	-0.397420	-0.296629	-0.282276	0.006470
3	127.031932	37.359574	2.191857	-0.414402	-0.296629	-0.282276	0.006470
4	127.034124	37.370399	2.191857	-0.516289	-0.296629	-0.282276	-0.701191
...	...	...	...	...	...	...	...
10824	127.421737	37.147743	-2.128984	-0.516289	-0.296629	-0.282276	-0.701191
10825	127.421738	37.146841	-2.128984	-0.516289	-0.296629	-0.282276	-0.701191
10826	127.421742	37.143236	-2.128984	-0.516289	-0.296629	-0.282276	-0.701191
10827	127.421743	37.142334	-2.128984	-0.516289	-0.296629	-0.282276	-0.701191
10828	127.424004	37.133321	-2.128984	-0.516289	-0.296629	-0.282276	-0.701191

10829 rows × 14 columns

In [9]:
```
idx_x = raw_lin.iloc[:,2:].drop("tot_count_",axis = 1)
```

In [10]:
```
idx_y = raw_lin.iloc[:,-1]
```

In [11]:
```
idx_model = MLR(idx_x, idx_y)
idx_model.summary()
```

```
C:₩Users₩lynn1₩Anaconda3₩lib₩site-packages₩statsmodels₩tsa₩tsatools.py:142: FutureWarning: In a future version of pandas all arguments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

Out[11]:

OLS Regression Results

Dep. Variable:	tot_count_	**R-squared:**	0.039
Model:	OLS	**Adj. R-squared:**	0.038
Method:	Least Squares	**F-statistic:**	40.21
Date:	Fri, 14 Oct 2022	**Prob (F-statistic):**	4.32e-86
Time:	08:24:11	**Log-Likelihood:**	-15149.
No. Observations:	10829	**AIC:**	3.032e+04
Df Residuals:	10817	**BIC:**	3.041e+04
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	2.946e-18	0.009	3.13e-16	1.000	-0.018	0.018
car	0.0087	0.010	0.905	0.365	-0.010	0.028
population	0.1875	0.011	17.115	0.000	0.166	0.209
houses	-0.0725	0.031	-2.311	0.021	-0.134	-0.011
houses_parking	0.0956	0.031	3.098	0.002	0.035	0.156
building_religion	-0.0068	0.010	-0.661	0.508	-0.027	0.013
building_sale	0.0041	0.010	0.403	0.687	-0.016	0.024
building_edu	0.0060	0.010	0.577	0.564	-0.014	0.026
building_work	-0.0134	0.013	-1.012	0.312	-0.039	0.013
building_factory	0.0288	0.010	2.939	0.003	0.010	0.048
public_parking_lot	-0.0120	0.020	-0.615	0.538	-0.050	0.026
public_parking_space	0.0306	0.017	1.855	0.064	-0.002	0.063

Omnibus:	29522.310	**Durbin-Watson:**	2.046
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1194758992.416
Skew:	33.825	**Prob(JB):**	0.00
Kurtosis:	1628.833	**Cond. No.**	8.12

Notes:

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

In [12]:
```
demand_coeff = idx_model.coef()
```

In [13]:
```
raw_lin['ind'] = (demand_coeff[0]*raw_lin['car']+
                  demand_coeff[1]*raw_lin['population']+
                  demand_coeff[2]*raw_lin['houses']+
                  demand_coeff[3]*raw_lin['houses_parking']+
                  demand_coeff[4]*raw_lin['building_religion']+
                  demand_coeff[5]*raw_lin['building_sale']+
                  demand_coeff[6]*raw_lin['building_edu'] +
```

```
                    demand_coeff[7]*raw_lin['building_work'] +
                    demand_coeff[8]*raw_lin['building_factory'] +
                    demand_coeff[9]*raw_demand['public_parking_lot'] +
                    demand_coeff[10]*raw_lin['public_parking_space'])
```

In [14]:
```
raw_lin
```

Out[14]:

	cell_x	cell_y	car	population	houses	houses_parking	building_religion
0	127.030786	37.362273	2.191857	-0.442704	-0.296629	-0.282276	0.006470
1	127.030798	37.360471	2.191857	-0.516289	-0.296629	-0.282276	0.006470
2	127.031927	37.360475	2.191857	-0.397420	-0.296629	-0.282276	0.006470
3	127.031932	37.359574	2.191857	-0.414402	-0.296629	-0.282276	0.006470
4	127.034124	37.370399	2.191857	-0.516289	-0.296629	-0.282276	-0.701191
...	...	...	...	...	...	...	...
10824	127.421737	37.147743	-2.128984	-0.516289	-0.296629	-0.282276	-0.701191
10825	127.421738	37.146841	-2.128984	-0.516289	-0.296629	-0.282276	-0.701191
10826	127.421742	37.143236	-2.128984	-0.516289	-0.296629	-0.282276	-0.701191
10827	127.421743	37.142334	-2.128984	-0.516289	-0.296629	-0.282276	-0.701191
10828	127.424004	37.133321	-2.128984	-0.516289	-0.296629	-0.282276	-0.701191

10829 rows × 15 columns

후보지

In [15]:
```
raw_candidate
```

Out[15]:

	cell_x	cell_y	cnt_cust*charger_val
0	127.075577	37.229037	345469.765400
1	127.072773	37.326377	281137.608500
2	127.078416	37.326397	157946.033400
3	127.083006	37.312893	138673.965000
4	127.089601	37.345366	114740.448000
...	...	...	...
1995	127.124828	37.299512	591.654180
1996	127.127000	37.316645	590.826000
1997	127.132574	37.098522	590.358000
1998	127.131695	37.279703	590.295343
1999	127.131462	37.327476	589.415447

2000 rows × 3 columns

In [16]:
```
## 후보지 데이터 좌표값 가져오기
```

```
X = list(raw_candidate["cell_x"])
Y = list(raw_candidate["cell_y"])

candidate_points = np.array([list(i) for i in zip(X, Y)])
print(candidate_points.shape)
candidate_points
```

Out[16]:
```
(2000, 2)
array([[127.0755771 ,  37.22903674],
       [127.0727728 ,  37.3263772 ],
       [127.0784163 ,  37.32639744],
       ...,
       [127.1325735 ,  37.09852241],
       [127.1316946 ,  37.27970291],
       [127.1314615 ,  37.32747594]])
```

In [17]:
```
## 수요지 데이터 좌표값 가져오기

X = list(raw_lin["cell_x"])
Y = list(raw_lin["cell_y"])

demand_points = np.array([list(i) for i in zip(X, Y)])
print(demand_points.shape)
demand_points
```

Out[17]:
```
(10829, 2)
array([[127.0307864 ,  37.36227345],
       [127.0307977 ,  37.36047073],
       [127.0319269 ,  37.36047521],
       ...,
       [127.4217421 ,  37.1432356 ],
       [127.421743  ,  37.14233418],
       [127.424004  ,  37.13332144]])
```

In [18]:
```
# Plot input data
plot_input(candidate_points)
```

In [19]:
```
plot_input(demand_points)
```

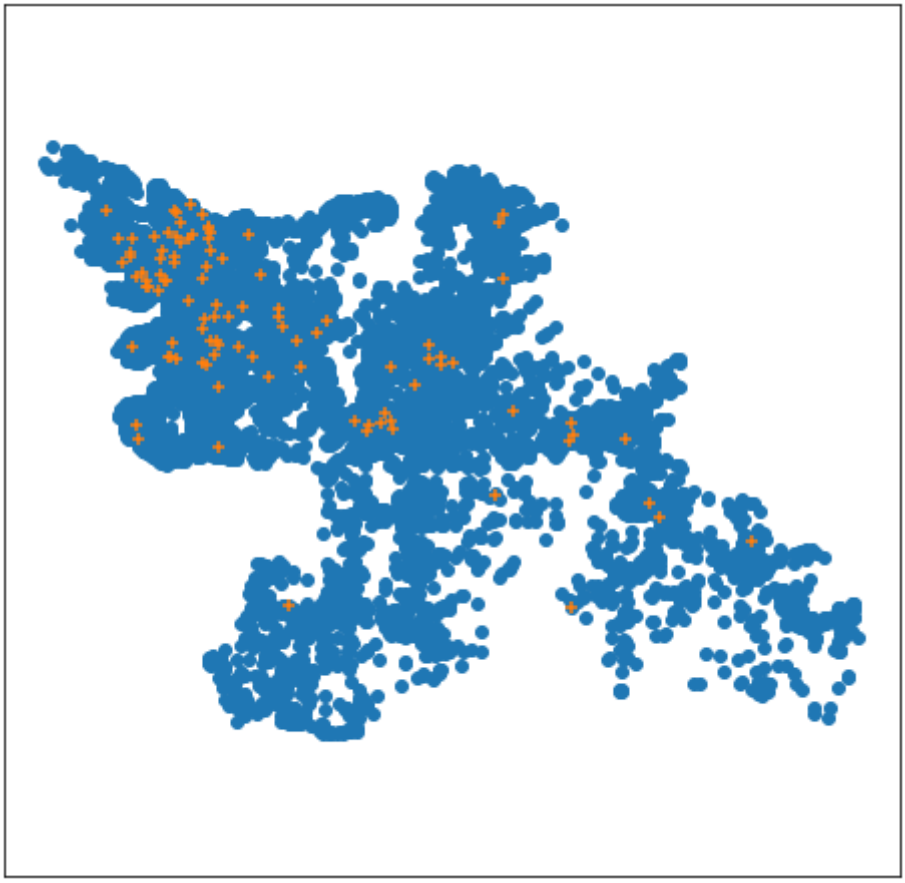
In [20]:
```
# mclp(K, radius, demand, candidate, Weight):
opt_sites, mobjVal= mclp(100,0.310686,demand_points , candidate_points[:1000], raw_l
opt_sites
```

```
----- Configurations -----
  수요지 수 10829
  후보지 수 1000
  K 100
  Radius 0.310686
----- Output -----
  런타임 : 8.961790800094604 seconds
  Optimal coverage points: 746.406
```

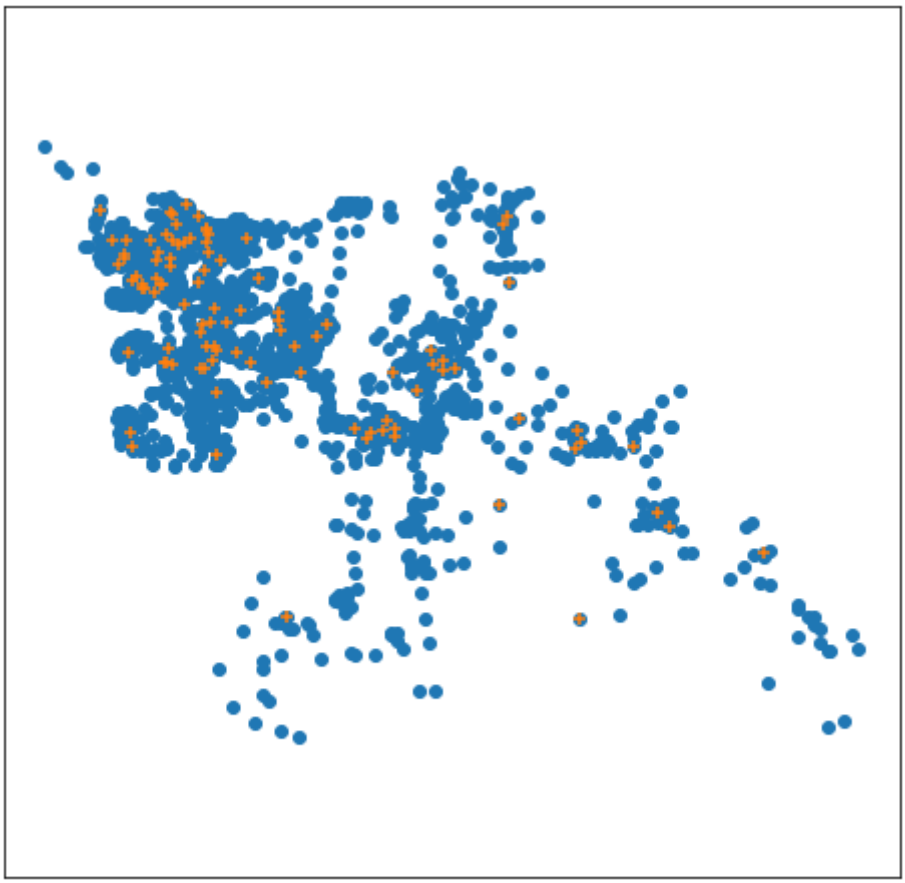
Out[20]:
```
array([[127.0755771 ,  37.22903674],
       [127.0727728 ,  37.3263772 ],
       [127.0942183 ,  37.32645269],
       [127.1452139 ,  37.28334978],
       [127.1803787 ,  37.23747858],
       [127.1149029 ,  37.25441036],
       [127.1860344 ,  37.23298669],
       [127.1306194 ,  37.2688828 ],
       [127.0886877 ,  37.30570166],
       [127.0931576 ,  37.31382962],
       [127.1984314 ,  37.23392031],
       [127.1385618 ,  37.25899194],
       [127.0945329 ,  37.26786332],
       [127.0863717 ,  37.31651026],
       [127.252308  ,  37.30615721],
       [127.1111815 ,  37.32019993],
       [127.1440646 ,  37.2878533 ],
       [127.1079688 ,  37.28683786],
       [127.0864161 ,  37.30839791],
       [127.1009048 ,  37.34270038],
       [127.1349393 ,  37.30855754],
       [127.1927837 ,  37.23661001],
       [127.0808039 ,  37.30296983],
       [127.0964902 ,  37.3237563 ],
       [127.1192455 ,  37.28777607],
       [127.0941553 ,  37.3381705 ],
       [127.1068642 ,  37.28232723],
       [127.2276289 ,  37.26554014],
       [127.0716952 ,  37.31735944],
       [127.1871505 ,  37.23569386],
       [127.1111351 ,  37.32921366],
       [127.0964372 ,  37.33367137],
       [127.1099924 ,  37.33191406],
       [127.0744144 ,  37.23534238],
       [127.0908175 ,  37.32914514],
       [127.1259876 ,  37.29320593],
       [127.1069485 ,  37.26610239],
       [127.0730735 ,  37.27319624],
       [127.1089663 ,  37.31208013],
       [127.068325  ,  37.31464302],
       [127.3111801 ,  37.22874531],
       [127.3281751 ,  37.19091197],
       [127.110025  ,  37.32560446],
       [127.2852297 ,  37.23681375],
       [127.0660005 ,  37.32635256],
       [127.2096219 ,  37.25558194],
       [127.1125691 ,  37.26972651],
       [127.2219931 ,  37.26462564],
       [127.0773835 ,  37.30926739],
       [127.0998667 ,  37.32557054],
       [127.1621472 ,  37.28069514],
       [127.2499616 ,  37.33319404],
       [127.0853273 ,  37.30118298],
       [127.1543292 ,  37.26354602],
       [127.1000243 ,  37.29582521],
       [127.1135726 ,  37.29406744],
       [127.1092088 ,  37.26520848],
       [127.2522045 ,  37.3377057 ],
       [127.1148019 ,  37.27424077],
       [127.3721369 ,  37.17924919],
       [127.0840599 ,  37.32641742],
       [127.0796605 ,  37.30566994],
       [127.0602782 ,  37.3398522 ],
       [127.2481426 ,  37.2015858 ],
```

```
       [127.0922383 ,  37.2750666 ],
       [127.0900167 ,  37.26874916],
       [127.1136647 ,  37.27603986],
       [127.1666355 ,  37.28611623],
       [127.1021145 ,  37.3273809 ],
       [127.1238299 ,  37.27336841],
       [127.2163169 ,  37.27452756],
       [127.1168429 ,  37.31661284],
       [127.2219765 ,  37.2691326 ],
       [127.1150542 ,  37.22466462],
       [127.1292041 ,  37.32746889],
       [127.2854823 ,  37.14757442],
       [127.198417  ,  37.23752589],
       [127.1950203 ,  37.2411228 ],
       [127.0931382 ,  37.3174351 ],
       [127.284128  ,  37.22779769],
       [127.0796756 ,  37.30296582],
       [127.1983128 ,  37.26366627],
       [127.1440434 ,  37.29236021],
       [127.1480987 ,  37.14814757],
       [127.0898161 ,  37.30570558],
       [127.2570275 ,  37.24216767],
       [127.1102809 ,  37.27602877],
       [127.1124767 ,  37.2877541 ],
       [127.0930215 ,  37.33906799],
       [127.0751368 ,  37.30745657],
       [127.109997  ,  37.33101269],
       [127.2863749 ,  37.23050598],
       [127.071685  ,  37.31916217],
       [127.0874805 ,  37.32011968],
       [127.1520156 ,  37.2761588 ],
       [127.0911445 ,  37.26875306],
       [127.3225247 ,  37.19811494],
       [127.1067375 ,  37.30666441],
       [127.1065778 ,  37.33731107],
       [127.2163439 ,  37.26731644]])
```

In [21]:
```
plot_result(demand_points,opt_sites,1)
```



In [22]:
```
plot_result(candidate_points,opt_sites,1)
```



In [23]:
```
df_opt = pd.DataFrame(opt_sites)
df_opt.columns = ["x","y"]
```

In [24]:
```
df_opt.to_csv("최적 입지_100.csv",encoding = "cp949",index = True)
```