13

공공데이터 분석

무선청소기 모델별 비교 분석

▪ 데이터 살펴보기

```
# 데이터 불러오기
import pandas as pd
data = pd.read_excel('data/danawa_data.xlsx')
data.head()
```

	카테고리	회사명	제품	가격	사용시간	흡입력
0	핸디/스틱청소기	샤오미	드리미 V10	173900	60.0	220.0
1	핸디/스틱청소기	원더스리빙	다이나킹 Z9	299000	65.0	220.0
2	핸디/스틱청소기	LG전자	코드제로 A9 A978	1005340	80.0	140.0
3	핸디/스틱청소기	델로라	V11 파워 300W	141000	70.0	220.0
4	핸디/스틱청소기	샤오미	드리미 V9	138800	60.0	200.0

```
# columns 데이터 확인
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 241 entries, 0 to 240
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   카테고리    241 non-null    object
 1   회사명     241 non-null    object
 2   제품      241 non-null    object
 3   가격      241 non-null    int64
 4   사용시간    218 non-null    float64
 5   흡입력     129 non-null    float64
dtypes: float64(2), int64(1), object(3)
memory usage: 11.4+ KB
```

무선청소기 모델별 비교 분석

▪ 데이터 살펴보기

```
# 흡입력 기준 정렬
top_list = data.sort_values(["흡입력"],
                            ascending = False)
top_list.head()
```

	카테고리	회사명	제품	가격	사용시간	흡입력
13	핸디/스틱청소기	DIBEA	F20 맥스	222990	50.0	250.0
127	핸디/스틱청소기	DIBEA	X30	259000	50.0	250.0
165	핸디/스틱청소기	DIBEA	TSX-25000A	244470	45.0	250.0
143	핸디/스틱청소기	DIBEA	F20 울트라 맥스	236550	60.0	250.0
152	핸디/스틱청소기	아이룸	RS1	178000	40.0	250.0

```
# 사용시간 기준 정렬
top_list = data.sort_values(["사용시간"],
                            ascending = False)
top_list.head()
```

	카테고리	회사명	제품	가격	사용시간	흡입력
111	핸디/스틱청소기	삼성전자	제트 VS20R9074S2	845990	120.0	200.0
5	핸디/스틱청소기	삼성전자	제트 VS20R9078S2	877880	120.0	200.0
153	핸디/스틱청소기	샤오미	이지에 YE-01	24740	120.0	NaN
16	핸디/스틱청소기	삼성전자	제트 VS20R9078S3	918120	120.0	200.0
76	핸디/스틱청소기	삼성전자	제트 VS20R9074S3	870910	120.0	200.0

무선청소기 모델별 비교 분석

▪ 데이터 살펴보기

```
# 흡입력, 사용시간을 기준으로 정렬
top_list = data.sort_values(["사용시간","흡입력"], ascending = False)
top_list.head()
```

	카테고리	회사명	제품	가격	사용시간	흡입력
5	핸디/스틱청소기	삼성전자	제트 VS20R9078S2	877880	120.0	200.0
16	핸디/스틱청소기	삼성전자	제트 VS20R9078S3	918120	120.0	200.0
76	핸디/스틱청소기	삼성전자	제트 VS20R9074S3	870910	120.0	200.0
109	핸디/스틱청소기	삼성전자	제트 VS20R9077Q3	931100	120.0	200.0
111	핸디/스틱청소기	삼성전자	제트 VS20R9074S2	845990	120.0	200.0

무선청소기 모델별 비교 분석

▪ 데이터 살펴보기

```
# 평균값 정리
price_mean = data['가격'].mean()
suction_mean = data['흡입력'].mean()
use_time_mean = data['사용시간'].mean()
print("가격 평균값", price_mean)
print("흡입력 평균값", suction_mean)
print("사용시간 평균값", use_time_mean)
```

```
가격 평균값 296844.79253112036
흡입력 평균값 151.8294573643411
사용시간 평균값 43.38990825688074
```

```
# 가성비 좋은 제품 탐색
condition_data = data [
    (data['가격'] <= price_mean) &
    (data['흡입력'] >= suction_mean) &
    (data['사용시간'] >= use_time_mean)]
condition_data
```

	카테고리	회사명	제품	가격	사용시간	흡입력
0	핸디/스틱청소기	샤오미	드리미 V10	173900	60.0	220.0
3	핸디/스틱청소기	델로라	V11 파워 300W	141000	70.0	220.0
4	핸디/스틱청소기	샤오미	드리미 V9	138800	60.0	200.0
13	핸디/스틱청소기	DIBEA	F20 맥스	222990	50.0	250.0
18	핸디/스틱청소기	DIBEA	M500 퀀텀	248640	50.0	250.0
42	핸디/스틱청소기	DIBEA	F20 프로	161970	50.0	220.0
73	핸디/스틱청소기	JDL	tech 타이푼 DV-889DC-X	137160	50.0	200.0

무선청소기 모델별 비교 분석

▪ 데이터 시각화

```
# 라이브러리 임포트 및 한글 글꼴 설정
from matplotlib import font_manager, rc
import platform
font_path = ''
if platform.system() == 'Windows':
    font_path = 'c:/Windows/Fonts/malgun.ttf'
    font_name = font_manager.FontProperties(fname = font_path).get_name()
    rc('font', family = font_name)
elif platform.system() == 'Darwin':
    font_path = '/Users/$USER/Library/Fonts/AppleGothic.ttf'
    rc('font', family = 'AppleGothic')
else:
    print('Check your OS system')

%matplotlib inline
```

무선청소기 모델별 비교 분석

▪ 데이터 시각화

```
# 시각화를 위해 null data 정리 하기위해 데이터 확인
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 241 entries, 0 to 240
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   카테고리    241 non-null    object
 1   회사명     241 non-null    object
 2   제품      241 non-null    object
 3   가격      241 non-null    int64
 4   사용시간    218 non-null    float64
 5   흡입력     129 non-null    float64
dtypes: float64(2), int64(1), object(3)
memory usage: 11.4+ KB
```

```
# null data 확인
data.isnull()
```

	카테고리	회사명	제품	가격	사용시간	흡입력
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
...	...	...	...	...	...	...
236	False	False	False	False	False	False
237	False	False	False	False	True	True

```
data['흡입력'].isnull()
```

```
0      False
1      False
2      False
3      False
4      False
       ...
236    False
237     True
238     True
239    False
240    False
Name: 흡입력, Length: 241, dtype: bool
```

무선청소기 모델별 비교 분석

▪ 데이터 시각화

```
# 결측값 없애기
chart_data = data.dropna(axis = 0)
chart_data.head()
```

	카테고리	회사명	제품	가격	사용시간	흡입력
0	핸디/스틱청소기	샤오미	드리미 V10	173900	60.0	220.0
1	핸디/스틱청소기	원더스리빙	다이나킹 Z9	299000	65.0	220.0
2	핸디/스틱청소기	LG전자	코드제로 A9 A978	1005340	80.0	140.0
3	핸디/스틱청소기	델로라	V11 파워 300W	141000	70.0	220.0
4	핸디/스틱청소기	샤오미	드리미 V9	138800	60.0	200.0

```
# null data 확인
chart_data['흡입력'].isnull()
```

```
0      False
1      False
2      False
3      False
4      False
       ...
231    False
235    False
236    False
239    False
240    False
Name: 흡입력, Length: 123, dtype: bool
```

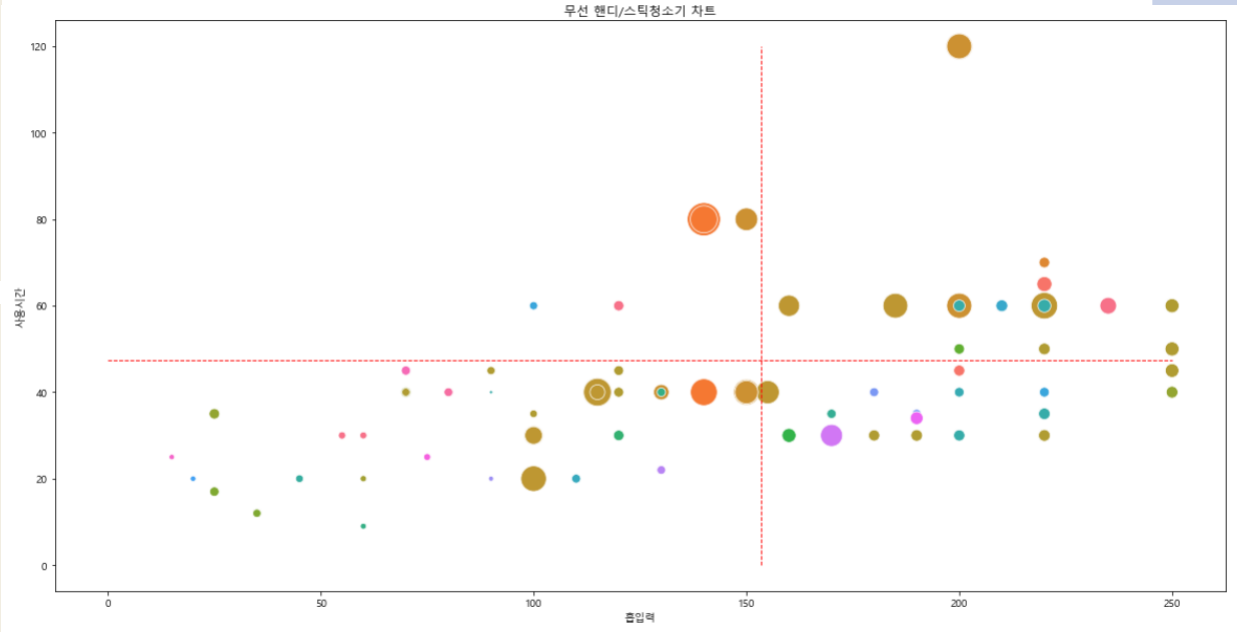
```
chart_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 123 entries, 0 to 240
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   카테고리    123 non-null    object
 1   회사명     123 non-null    object
 2   제품      123 non-null    object
 3   가격      123 non-null    int64
 4   사용시간    123 non-null    float64
 5   흡입력     123 non-null    float64
dtypes: float64(2), int64(1), object(3)
memory usage: 6.7+ KB
```

무선청소기 모델별 비교 분석

▪ 데이터 시각화

```
# 흡입력, 사용시간의 최댓값/최솟값 정리
suction_max = chart_data['흡입력'].max()
suction_mean = chart_data['흡입력'].mean()
use_time_max = chart_data['사용시간'].max()
use_time_mean = chart_data['사용시간'].mean()
```




```
import matplotlib.pyplot as plt
import seaborn as sns

# 청소기 성능 시각화
plt.figure(figsize=(20, 10))
plt.title("무선 핸디/스틱청소기 차트")
sns.scatterplot(x = '흡입력', y = '사용시간', size = '가격', hue = chart_data['회사명'],
                data = chart_data, sizes = (10, 1000), legend = False)
plt.plot([0, suction_max], [use_time_mean, use_time_mean], 'r--', lw = 1 )
plt.plot([suction_mean, suction_mean], [0, use_time_max], 'r--', lw = 1 )
plt.show()
```

무선청소기 모델별 비교 분석

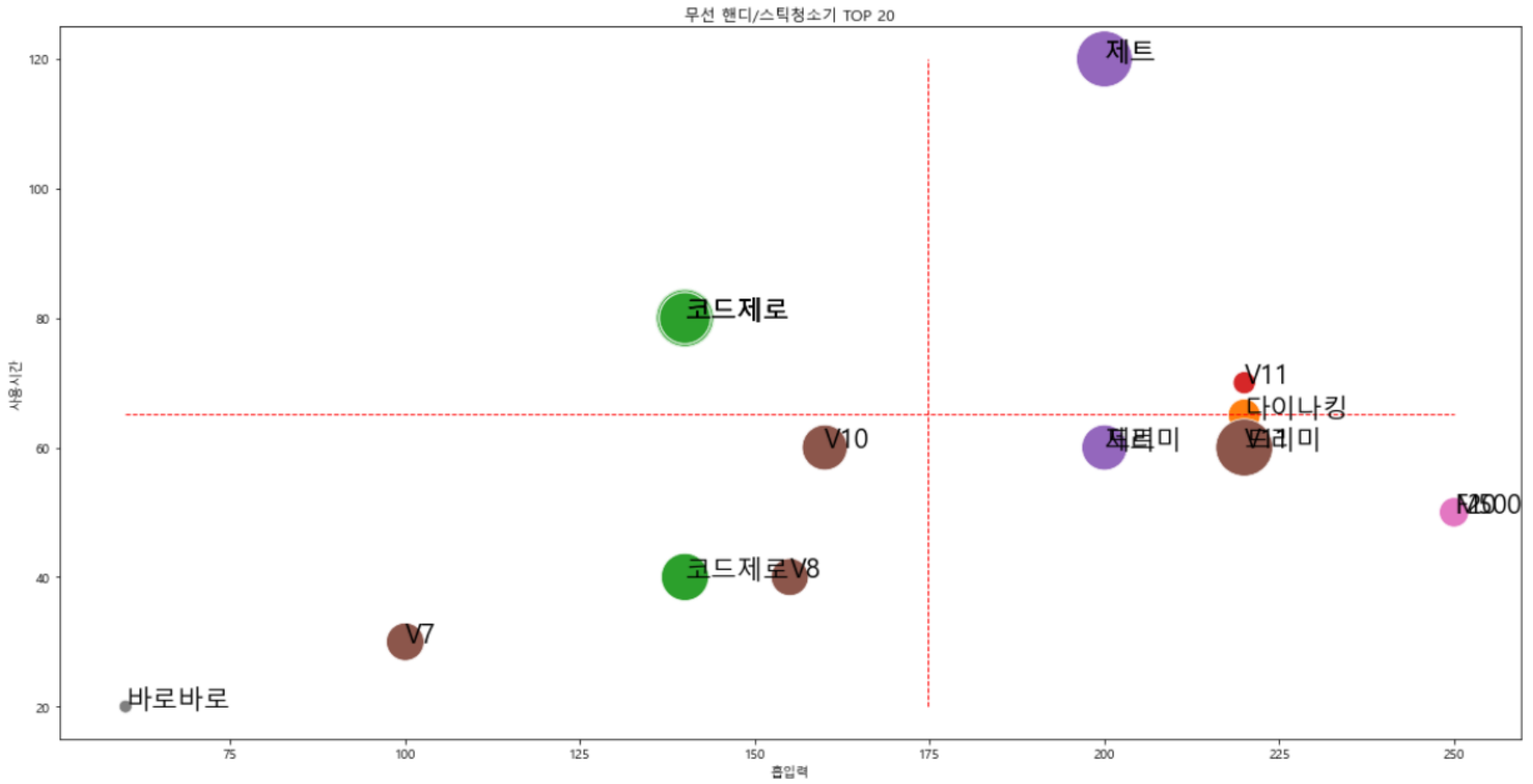
- 데이터 시각화

```
# 인기 제품 20개 선택
chart_data_selected = chart_data[:20]
len(chart_data_selected)
```

```
# 흡입력, 사용시간의 최댓값, 최솟값 구하기
suction_max = chart_data_selected['흡입력'].max()
suction_mean = chart_data_selected['흡입력'].mean()
use_time_max = chart_data_selected['사용시간'].max()
use_time_mean = chart_data_selected['사용시간'].mean()
plt.figure(figsize=(20, 10))
plt.title("무선 핸디/스틱청소기 TOP 20")
sns.scatterplot(x = '흡입력', y = '사용시간', size = '가격',
                hue = chart_data_selected['회사명'],
                data = chart_data_selected, sizes = (100, 2000),
                legend = False)
plt.plot([60, suction_max], [use_time_mean, use_time_mean], 'r--', lw = 1 )
plt.plot([suction_mean, suction_mean], [20, use_time_max], 'r--',  lw = 1 )
for index, row in chart_data_selected.iterrows():
    x = row['흡입력']
    y = row['사용시간']
    s = row['제품'].split(' ')[0]
    plt.text(x, y, s, size=20)
plt.show()
```

무선청소기 모델별 비교 분석

- 데이터 시각화



서울시 구별 CCTV 현황 분석

▪ CCTV 데이터 읽기

```
import pandas as pd
CCTV=pd.read_csv('data/CCTV.csv')
CCTV.head()
```

	기관명	소계	2013년도 이전	2014년	2015년	2016년
0	강남구	2780	1292	430	584	932
1	강동구	773	379	99	155	377
2	강북구	748	369	120	138	204
3	강서구	884	388	258	184	81
4	관악구	1496	846	260	390	613

```
CCTV.rename(columns={'기관명':'구별'}, inplace=True)
CCTV.head()
```

	구별	소계	2013년도 이전	2014년	2015년	2016년
0	강남구	2780	1292	430	584	932
1	강동구	773	379	99	155	377
2	강북구	748	369	120	138	204
3	강서구	884	388	258	184	81
4	관악구	1496	846	260	390	613

서울시 구별 CCTV 현황 분석

▪ 서울 구별 인구 데이터 읽기

```
pop_seoul=pd.read_excel('data/population.xls')
pop_seoul.head()
```

	기간	자치구	세대	인구	인구.1	인구.2	인구.3	인구.4	인구.5	인구.6	인구.7	인구.8	세대당인구	65세이상고령자
0	기간	자치구	세대	합계	합계	합계	한국인	한국인	한국인	등록외국인	등록외국인	등록외국인	세대당인구	65세이상고령자
1	기간	자치구	세대	계	남자	여자	계	남자	여자	계	남자	여자	세대당인구	65세이상고령자
2	2017.1/4	합계	4202888	10197604	5000005	5197599	9926968	4871560	5055408	270636	128445	142191	2.36	1321458
3	2017.1/4	종로구	72654	162820	79675	83145	153589	75611	77978	9231	4064	5167	2.11	25425
4	2017.1/4	중구	59481	133240	65790	67450	124312	61656	62656	8928	4134	4794	2.09	20764

```
# 필요한 데이터 선별
pop_seoul=pd.read_excel('data/population.xls', header=2, usecols=(1,3,6,9,13))
pop_seoul.head()
```

	자치구	계	계.1	계.2	65세이상고령자
0	합계	10197604.0	9926968.0	270636.0	1321458.0
1	종로구	162820.0	153589.0	9231.0	25425.0
2	중구	133240.0	124312.0	8928.0	20764.0
3	용산구	244203.0	229456.0	14747.0	36231.0
4	성동구	311244.0	303380.0	7864.0	39997.0

=

```
pop_seoul=pd.read_excel('data/population.xls', header=2, usecols='B,D,G,J,N')
```

=

```
pop_seoul=pd.read_excel('data/population.xls', header=2)
pop_seoul = pop_seoul[['자치구','계','계.1', '계.2','65세이상고령자']]
```

서울시 구별 CCTV 현황 분석

▪ 서울 구별 인구 데이터 읽기

```
#column명 변경
pop_seoul.columns=['구별','인구수','한국인', '외국인', '고령자']
pop_seoul.head()
```

	구별	인구수	한국인	외국인	고령자
0	합계	10197604.0	9926968.0	270636.0	1321458.0
1	종로구	162820.0	153589.0	9231.0	25425.0
2	중구	133240.0	124312.0	8928.0	20764.0
3	용산구	244203.0	229456.0	14747.0	36231.0
4	성동구	311244.0	303380.0	7864.0	39997.0

=

```
pop_seoul.rename(columns={'자치구':'구별',
                          '계':'인구수',
                          '계.1':'한국인',
                          '계.2':'외국인',
                          '65세이상고령자':'고령자',
                         }, inplace=True)
```

=

```
pop_seoul.rename(columns={pop_seoul.columns[0]:'구별',
                          pop_seoul.columns[1]:'인구수',
                          pop_seoul.columns[2]:'한국인',
                          pop_seoul.columns[3]:'외국인',
                          pop_seoul.columns[4]:'고령자',
                         }, inplace=True)
```

서울시 구별 CCTV 현황 분석

- **CCTV 데이터 읽기**
  - CCTV 데이터에서 CCTV 전체 개수인 소계로 정렬

```
CCTV.sort_values(by='소계',ascending=True).head()
```

	구별	소계	2013년도 이전	2014년	2015년	2016년
9	도봉구	485	238	159	42	386
12	마포구	574	314	118	169	379
17	송파구	618	529	21	68	463
24	중랑구	660	509	121	177	109
23	중구	671	413	190	72	348

```
CCTV.sort_values(by='소계',ascending=False).head()
```

	구별	소계	2013년도 이전	2014년	2015년	2016년
0	강남구	2780	1292	430	584	932
18	양천구	2034	1843	142	30	467
14	서초구	1930	1406	157	336	398
21	은평구	1873	1138	224	278	468
20	용산구	1624	1368	218	112	398

서울시 구별 CCTV 현황 분석

- **CCTV 데이터 읽기**
  - 2014~2016까지 증가율 계산

```
CCTV['최근증가율']=(CCTV['2016년']+CCTV['2015년']+CCTV['2014년'])/
                    (CCTV['2013년도 이전']) *100
CCTV.sort_values(by='최근증가율', ascending=False).head()
```

	구별	소계	2013년도 이전	2014년	2015년	2016년	최근증가율
22	종로구	1002	464	314	211	630	248.922414
9	도봉구	485	238	159	42	386	246.638655
12	마포구	574	314	118	169	379	212.101911
8	노원구	1265	542	57	451	516	188.929889
1	강동구	773	379	99	155	377	166.490765

서울시 구별 CCTV 현황 분석

▪ 서울 구별 인구 데이터 정리

```
pop_seoul.head()
```

	구별	인구수	한국인	외국인	고령자
0	합계	10197604.0	9926968.0	270636.0	1321458.0
1	종로구	162820.0	153589.0	9231.0	25425.0
2	중구	133240.0	124312.0	8928.0	20764.0
3	용산구	244203.0	229456.0	14747.0	36231.0
4	성동구	311244.0	303380.0	7864.0	39997.0

```
# 합계 삭제
pop_seoul.drop([0], inplace=True)
pop_seoul.head()
```

	구별	인구수	한국인	외국인	고령자
1	종로구	162820.0	153589.0	9231.0	25425.0
2	중구	133240.0	124312.0	8928.0	20764.0
3	용산구	244203.0	229456.0	14747.0	36231.0
4	성동구	311244.0	303380.0	7864.0	39997.0
5	광진구	372164.0	357211.0	14953.0	42214.0

▪ 서울의 모든 구 조사

```
pop_seoul['구별'].unique()
```

```
array(['종로구', '중구', '용산구', '성동구', '광진구', '동대문구', '중랑구', '성북구', '강북구',
       '도봉구', '노원구', '은평구', '서대문구', '마포구', '양천구', '강서구', '구로구', '금천구',
       '영등포구', '동작구', '관악구', '서초구', '강남구', '송파구', '강동구', nan],
      dtype=object)
```

```
# nan 삭제하기 위해 non 행 확인
pop_seoul[pop_seoul['구별'].isnull()]
```

	구별	인구수	한국인	외국인	고령자
26	NaN	NaN	NaN	NaN	NaN

```
# NaN 삭제
pop_seoul.drop([26], inplace=True)
pop_seoul.tail()
```

	구별	인구수	한국인	외국인	고령자
21	관악구	525515.0	507203.0	18312.0	68082.0
22	서초구	450310.0	445994.0	4316.0	51733.0
23	강남구	570500.0	565550.0	4950.0	63167.0
24	송파구	667483.0	660584.0	6899.0	72506.0
25	강동구	453233.0	449019.0	4214.0	54622.0

서울시 구별 CCTV 현황 분석

▪ 서울 구별 인구 데이터 정리

▪ 각 구별 전체 인구를 이용해서 구별 외국인비율과 고령자 비율 계산

```
pop_seoul['외국인비율']=(pop_seoul['외국인']/pop_seoul['인구수'])*100
pop_seoul['고령자비율']=(pop_seoul['고령자']/pop_seoul['인구수'])*100
pop_seoul.head()
```

	구별	인구수	한국인	외국인	고령자	외국인비율	고령자비율
1	종로구	162820.0	153589.0	9231.0	25425.0	5.669451	15.615404
2	중구	133240.0	124312.0	8928.0	20764.0	6.700690	15.583909
3	용산구	244203.0	229456.0	14747.0	36231.0	6.038828	14.836427
4	성동구	311244.0	303380.0	7864.0	39997.0	2.526635	12.850689
5	광진구	372164.0	357211.0	14953.0	42214.0	4.017852	11.342849

```
# 계산 결과 각각 확인
pop_seoul.sort_values(by='인구수', ascending=False).head()
pop_seoul.sort_values(by='외국인', ascending=False).head()
pop_seoul.sort_values(by='고령자', ascending=False).head()
pop_seoul.sort_values(by='외국인비율', ascending=False).head()
pop_seoul.sort_values(by='고령자비율', ascending=False).head()
```

서울시 구별 CCTV 현황 분석

▪ 두개의 데이타프레임 합치기

```
result_data=pd.merge(CCTV,pop_seoul, on='구별')#구별 column으로 병합
result_data.head()
```

	구별	소계	2013년도 이전	2014년	2015년	2016년	최근증가율	인구수	한국인	외국인	고령자	외국인비율	고령자비율
0	강남구	2780	1292	430	584	932	150.619195	570500.0	565550.0	4950.0	63167.0	0.867660	11.072217
1	강동구	773	379	99	155	377	166.490765	453233.0	449019.0	4214.0	54622.0	0.929765	12.051638
2	강북구	748	369	120	138	204	125.203252	330192.0	326686.0	3506.0	54813.0	1.061806	16.600342
3	강서구	884	388	258	184	81	134.793814	603772.0	597248.0	6524.0	72548.0	1.080540	12.015794
4	관악구	1496	846	260	390	613	149.290780	525515.0	507203.0	18312.0	68082.0	3.484582	12.955291

```
# 의미 없는 column 제거
result_data.drop(['2013년도 이전','2014년','2015년','2016년'], axis=1, inplace= True)
result_data.head()
```

	구별	소계	최근증가율	인구수	한국인	외국인	고령자	외국인비율	고령자비율
0	강남구	2780	150.619195	570500.0	565550.0	4950.0	63167.0	0.867660	11.072217
1	강동구	773	166.490765	453233.0	449019.0	4214.0	54622.0	0.929765	12.051638
2	강북구	748	125.203252	330192.0	326686.0	3506.0	54813.0	1.061806	16.600342
3	강서구	884	134.793814	603772.0	597248.0	6524.0	72548.0	1.080540	12.015794
4	관악구	1496	149.290780	525515.0	507203.0	18312.0	68082.0	3.484582	12.955291

서울시 구별 CCTV 현황 분석

▪ 두개의 데이타프레임 합치기

```
# 구별 column을 index로 설정
result_data.set_index('구별', inplace=True)
result_data.head()
```

구별	소계	최근증가율	인구수	한국인	외국인	고령자	외국인비율	고령자비율
강남구	2780	150.619195	570500.0	565550.0	4950.0	63167.0	0.867660	11.072217
강동구	773	166.490765	453233.0	449019.0	4214.0	54622.0	0.929765	12.051638
강북구	748	125.203252	330192.0	326686.0	3506.0	54813.0	1.061806	16.600342
강서구	884	134.793814	603772.0	597248.0	6524.0	72548.0	1.080540	12.015794
관악구	1496	149.290780	525515.0	507203.0	18312.0	68082.0	3.484582	12.955291

서울시 구별 CCTV 현황 분석

▪ 두개의 데이타프레임 합치기

```
# 전체 column 간 상관관계 분석
'''상관계수의 절대값이 클수록 두 데이터는 관계가 있음
ex) 상관계수의 절대값이 0.1 이하면 거의 무시, 0.3 이하면 약한 상관관계, 0.7 이하면 뚜렷한 상관관계'''
result_data.corr()
```

	소계	최근증가율	인구수	한국인	외국인	고령자	외국인비율	고령자비율
소계	1.000000	-0.343016	0.306342	0.304287	-0.023786	0.255196	-0.136074	-0.280786
최근증가율	-0.343016	1.000000	-0.093068	-0.082511	-0.150463	-0.070969	-0.044042	0.185089
인구수	0.306342	-0.093068	1.000000	0.998061	-0.153371	0.932667	-0.591939	-0.669462
한국인	0.304287	-0.082511	0.998061	1.000000	-0.214576	0.931636	-0.637911	-0.660812
외국인	-0.023786	-0.150463	-0.153371	-0.214576	1.000000	-0.155381	0.838904	-0.014055
고령자	0.255196	-0.070969	0.932667	0.931636	-0.155381	1.000000	-0.606088	-0.380468
외국인비율	-0.136074	-0.044042	-0.591939	-0.637911	0.838904	-0.606088	1.000000	0.267348
고령자비율	-0.280786	0.185089	-0.669462	-0.660812	-0.014055	-0.380468	0.267348	1.000000

CCTV수량 인구수량 약한 상관 관계임.

```
# 특정 column 간 상관관계 분석
result_data[['소계','고령자비율']].corr()
```

	소계	고령자비율
소계	1.000000	-0.280786
고령자비율	-0.280786	1.000000

```
result_data[['소계','외국인비율']].corr()
```

	소계	외국인비율
소계	1.000000	-0.136074
외국인비율	-0.136074	1.000000

```
result_data[['소계','인구수']].corr()
```

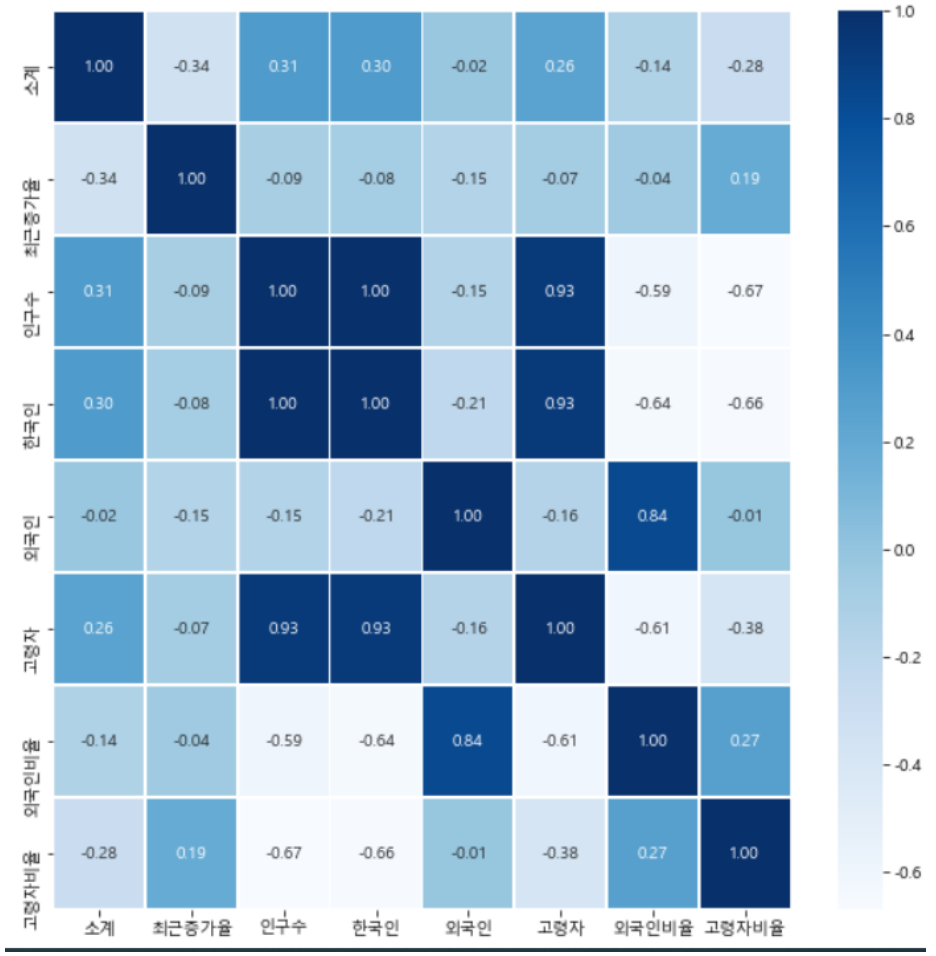
	소계	인구수
소계	1.000000	0.306342
인구수	0.306342	1.000000

서울시 구별 CCTV 현황 분석

▪ 두개의 데이타프레임 합치기

```
# 시각화
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10,10))
sns.heatmap(data = result_data.corr(), annot=True,
            fmt = '.2f', linewidths=.5, cmap='Blues')
```

```
# 한글 폰트 적용
from matplotlib import font_manager, rc
plt.rcParams['axes.unicode_minus']=False

path='c:/Windows/Fonts/malgun.ttf'
font_name=font_manager.FontProperties(fname=path).get_name()
rc('font',family=font_name)
```

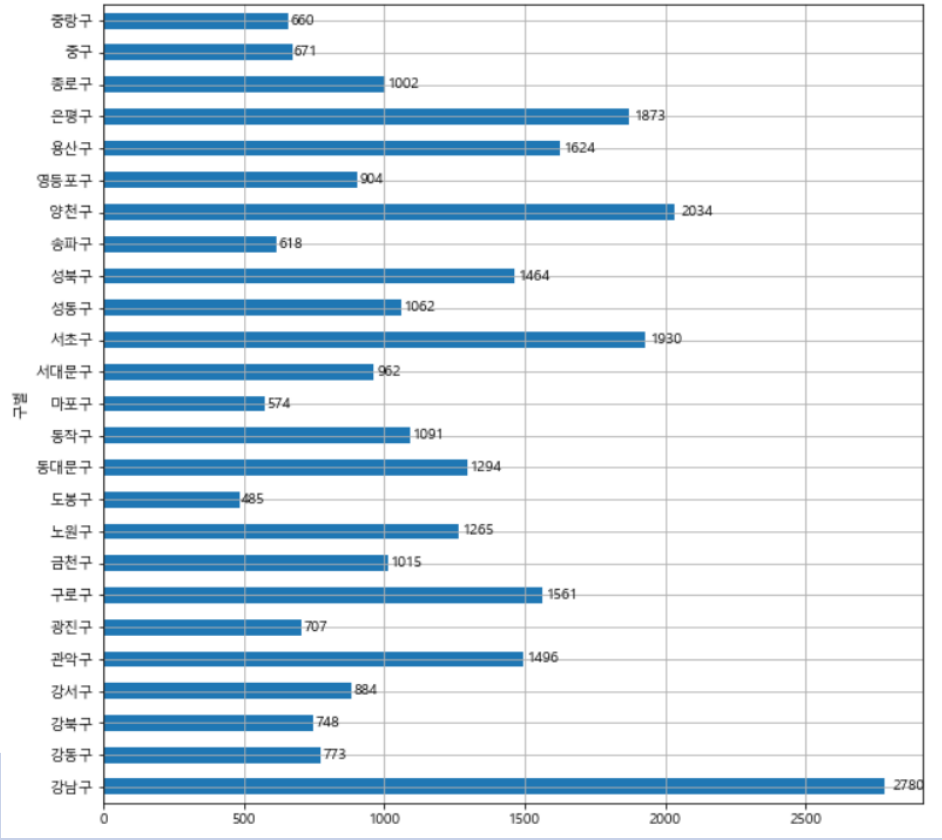


서울시 구별 CCTV 현황 분석

- CCTV 수 막대 그래프

```
result_data['소계'].plot(kind='barh', grid=True,
                         figsize=(10,10))
plt.show()
```


```
ax=data['소계'].plot(kind='barh', grid=True, figsize=(10, 10))
ax.ylable='구별'
for p in ax.patches:
    x, y, width, height = p.get_bbox().bounds
    ax.text(width*1.01, y+height/2, '%.d'%(width), va='center')
```

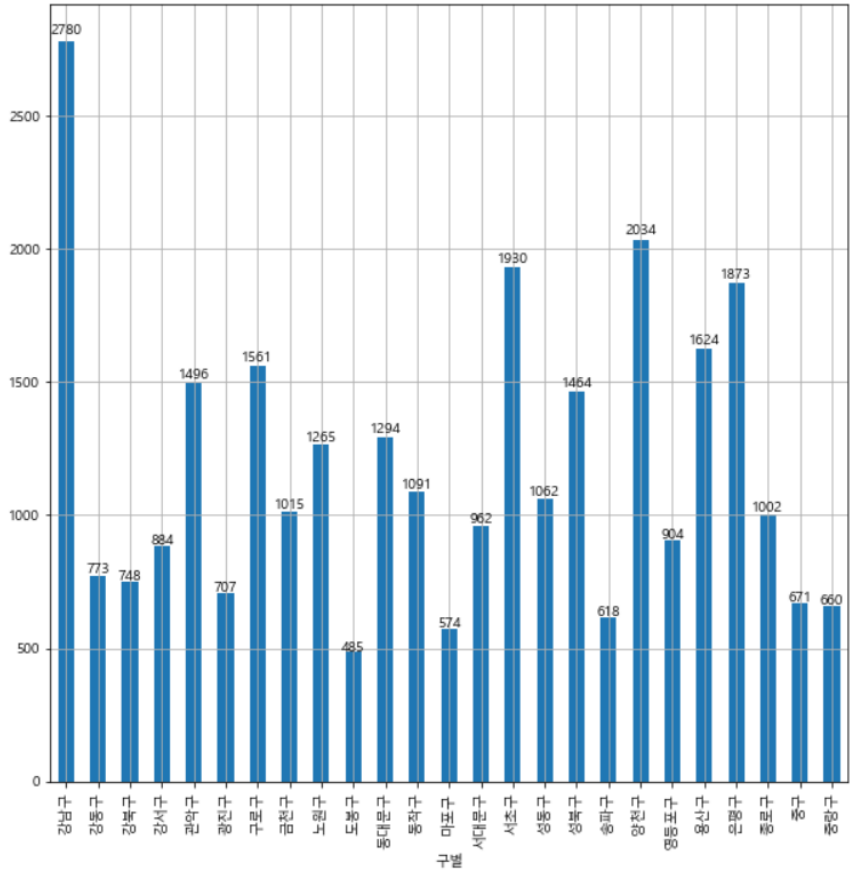


서울시 구별 CCTV 현황 분석

- CCTV수 수직 바 그래프

```
ax=result_data['소계'].plot(kind='bar', grid=True, figsize=(10, 10))
for p in ax.patches:
    left, bottom, width, height = p.get_bbox().bounds
    ax.text(left+width/2, height*1.01,'%d'%(height), ha='center')
```

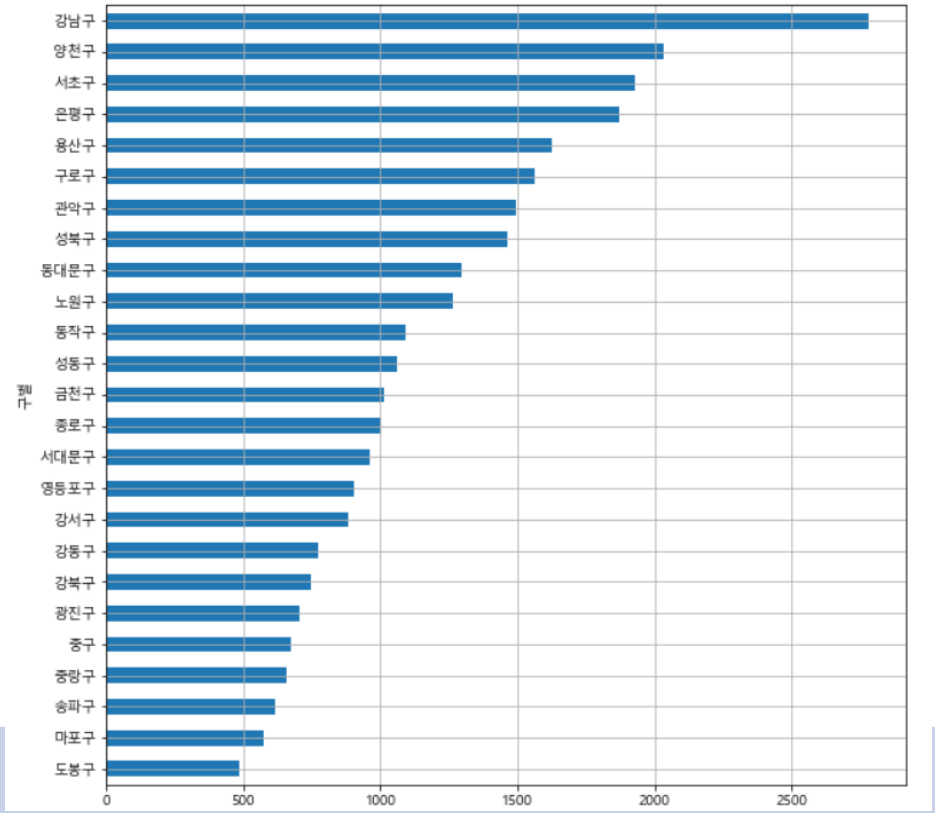
```
ax=result_data['소계'].plot(kind='bar', grid=True, figsize=(10, 10))
for p in ax.patches:
    left, bottom, width, height = p.get_bbox().bounds
    ax.annotate('%d'%(height), (left+width/2, height*1.01), ha='center')
```



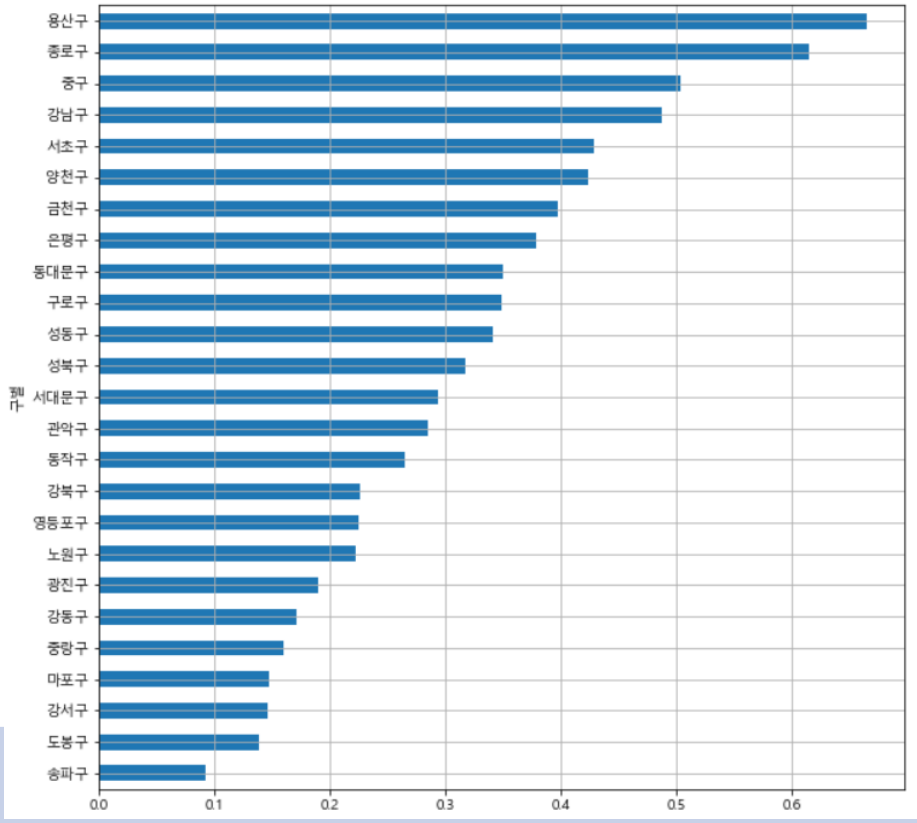
서울시 구별 CCTV 현황 분석

- CCTV수 수직 바 그래프

```
result_data['소계'].sort_values().plot(kind='barh',
                                       grid=True, figsize=(10,10))
plt.show()
```



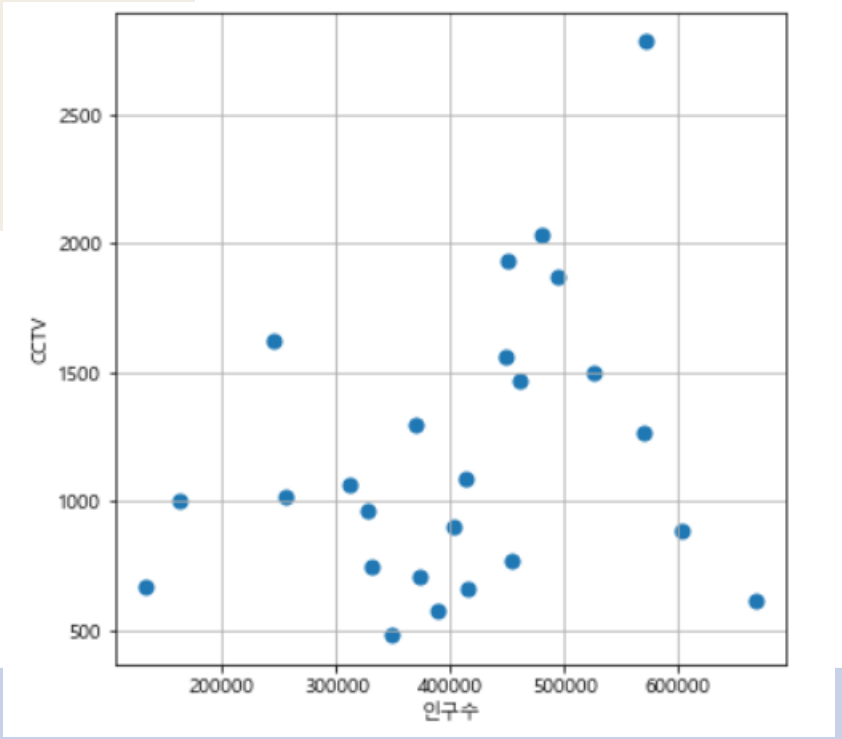

```
# 인구 대비 CCTV 비율 적용하여  정렬.
result_data['CCTV비율'] = result_data['소계'] / result_data['인구수'] *100
result_data['CCTV비율'].sort_values().plot(kind='barh', grid=True,
figsize=(10,10))
plt.show()
```



서울시 구별 CCTV 현황 분석

- CCTV수와 인구수 상관 관계 분석

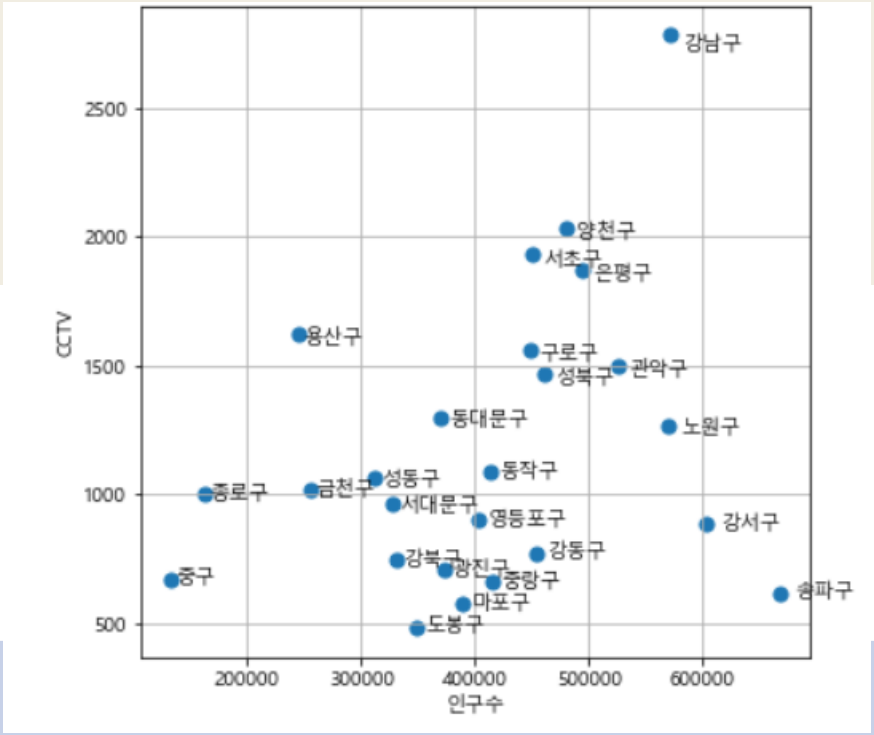
```
# scatter 그래프 적용
plt.figure(figsize=(6,6))
plt.scatter(result_data['인구수'], result_data['소계'], s=50)
plt.xlabel('인구수')
plt.ylabel('CCTV')
plt.grid()
plt.show()
```



서울시 구별 CCTV 현황 분석

- CCTV수와 인구수 상관 관계 분석

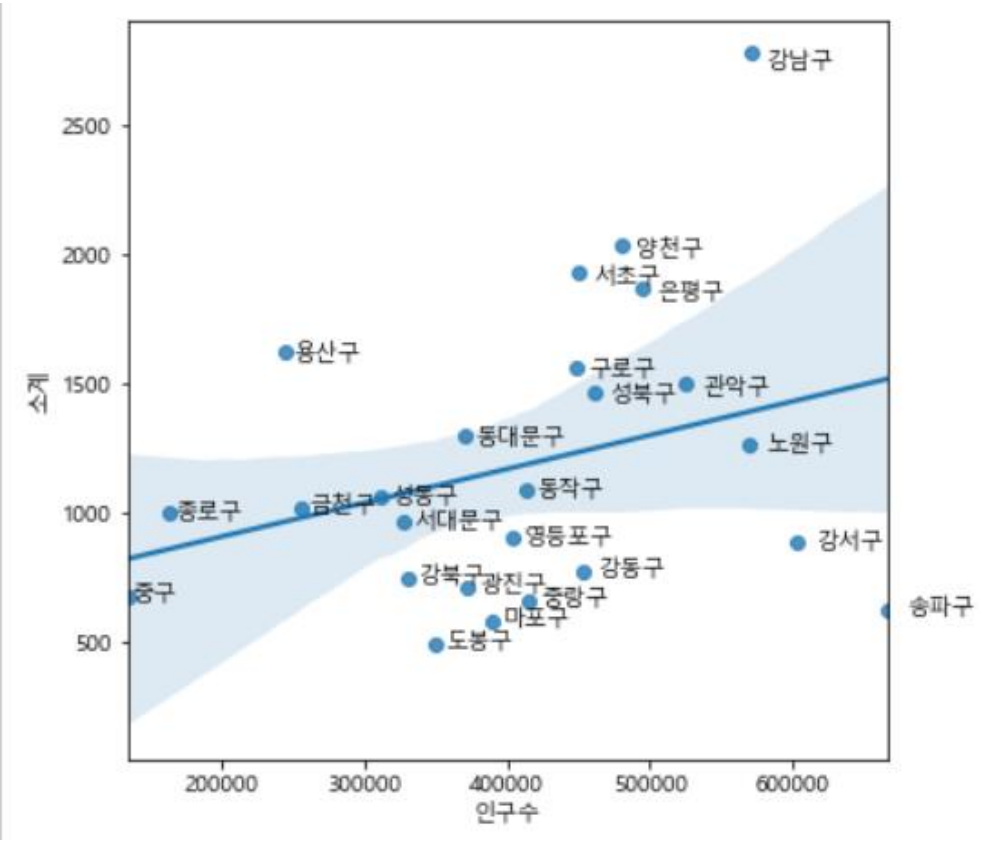
```
plt.figure(figsize=(6,6))
plt.scatter(result_data['인구수'], result_data['소계'], s=50)
for n in range(25): # 구별 레이블 표시
    plt.text(result_data['인구수'][n]*1.02, result_data['소계'][n]*0.98, result_data.index[n], fontsize=10)

plt.xlabel('인구수')
plt.ylabel('CCTV')
plt.grid()
plt.show()
```



서울시 구별 CCTV 현황 분석

- CCTV수와 인구수 회귀선 표시

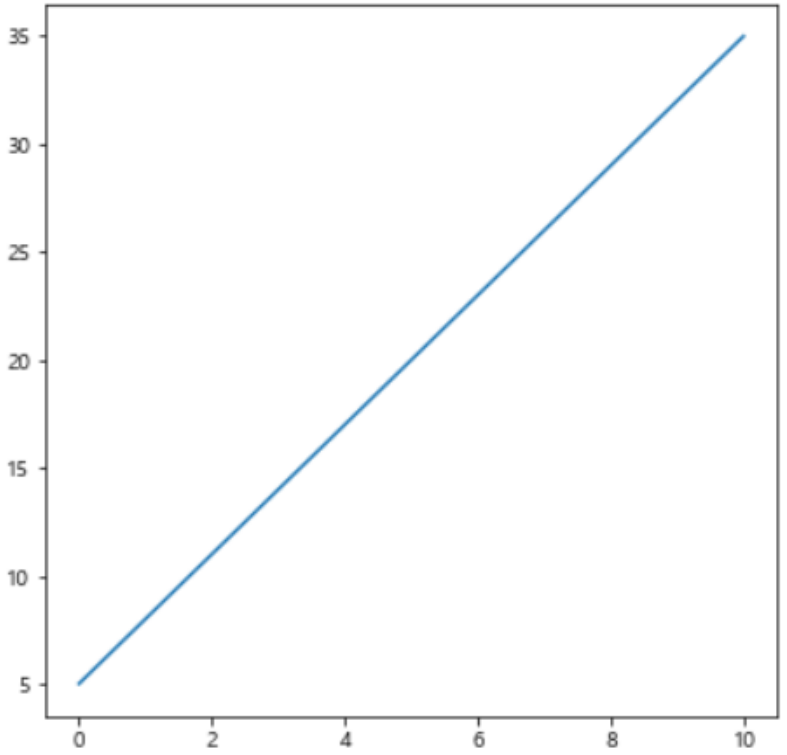
```
#seaborn 라이브러리 적용
import seaborn as sns

fig = plt.figure(figsize=(6, 6))

#  선형회귀선 표시
sns.regplot(x='인구수',      #x축 변수
            y='소계',        #y축 변수
            data=result_data) #데이터
plt.show()
```



서울시 구별 CCTV 현황 분석

- 직선의 방정식을 이용한 회귀선 표시

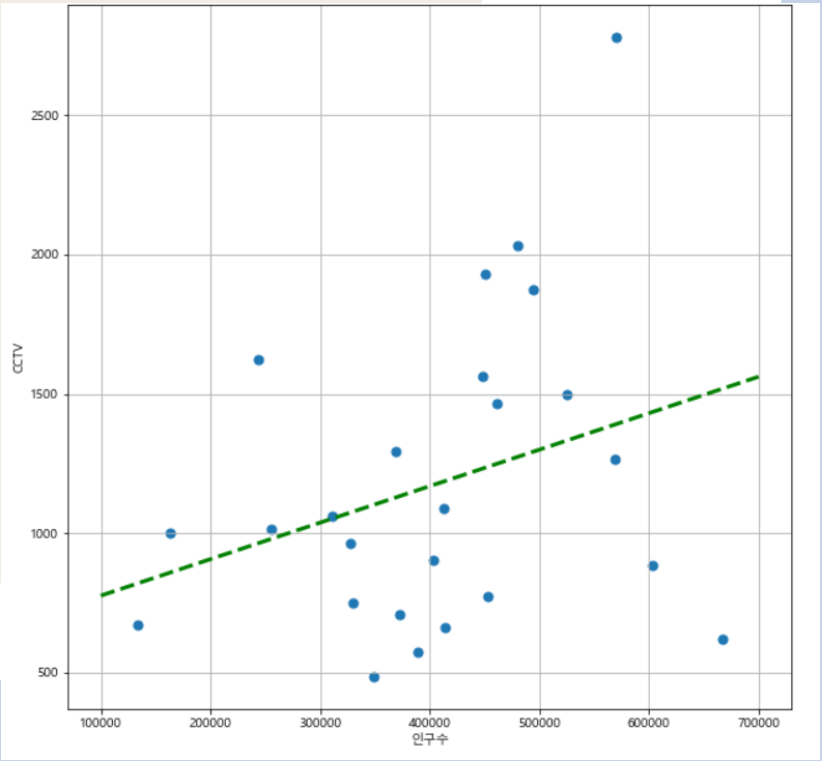
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
t = np.arange(0, 10, 0.01) #x축 설정
y = 3*t + 5  #직선을 그리기 위한 기울기, 절편
plt.figure(figsize=(6,6))
plt.plot(t, y)
plt.show()
```



서울시 구별 CCTV 현황 분석

- 직선의 방정식을 이용한 회귀선 표시

```
import numpy as np

#polyfit 함수로 다항식의 계수 구하기
fp1=np.polyfit(result_data['인구수'],result_data['소계'],1) # 1차 방정식 계수 구하기
f1=np.poly1d(fp1) # 1차 방정식 만들기
fx=np.linspace(100000,700000,100)

plt.figure(figsize=(10,10))
plt.scatter(result_data['인구수'],result_data['소계'],s=50)
plt.plot(fx,f1(fx),ls='dashed', lw=3, color='g') # 직선 그리기
plt.xlabel('인구수')
plt.ylabel('CCTV')
plt.grid()
plt.show()
```



서울시 구별 CCTV 현황 분석

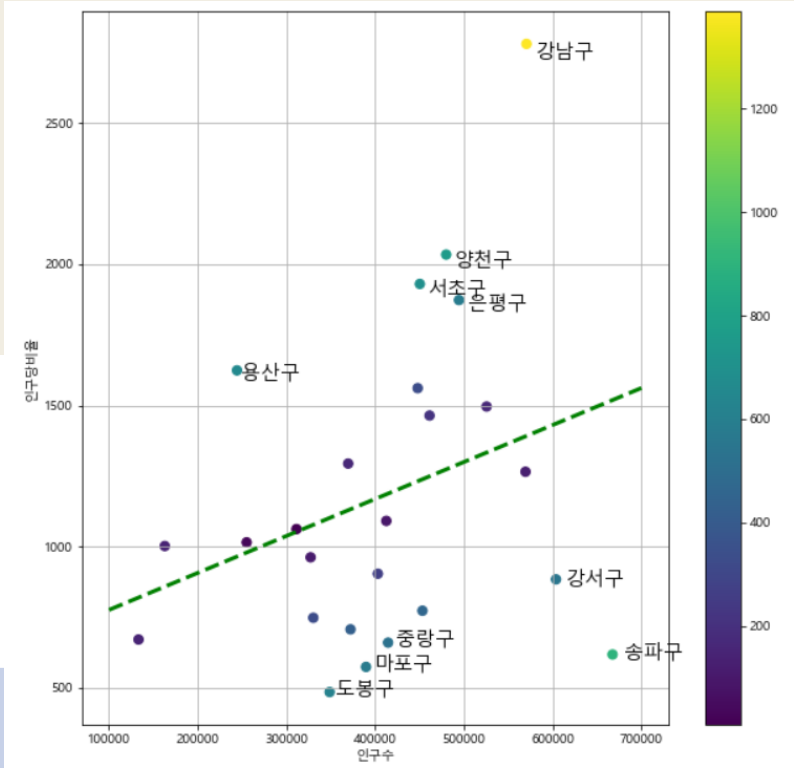
- 회귀선의 의미는 인구수가 300000일 때는 CCTV는 1100 정도여야 한다는 개념임
- 따라서, 값이 멀리 있는 것은 다른 색으로 표시

```
fp1=np.polyfit(result_data['인구수'], result_data['소계'],1)
f1=np.poly1d(fp1)
fx=np.linspace(100000,700000,100)
result_data['오차']=np.abs(result_data['소계']- f1(result_data['인구수']))
df_sort=result_data.sort_values(by='오차', ascending=False)
df_sort.head()
```

구별	소계	최근증가율	인구수	한국인	외국인	고령자	외국인비율	고령자비율	CCTV비율	오차
강남구	2780	150.619195	570500.0	565550.0	4950.0	63167.0	0.867660	11.072217	0.487292	1388.055355
송파구	618	104.347826	667483.0	660584.0	6899.0	72506.0	1.033584	10.862599	0.092587	900.911312
양천구	2034	34.671731	479978.0	475949.0	4029.0	52975.0	0.839413	11.036964	0.423769	760.563512
서초구	1930	63.371266	450310.0	445994.0	4316.0	51733.0	0.958451	11.488308	0.428594	695.403794
용산구	1624	53.216374	244203.0	229456.0	14747.0	36231.0	6.038828	14.836427	0.665020	659.231690

서울시 구별 CCTV 현황 분석

```
plt.figure(figsize=(10,10))
plt.scatter(result_data['인구수'],result_data['소계'],c=result_data['오차'], s=50)
plt.plot(fx,f1(fx),ls='dashed', lw=3, color='g')

for n in range(10):
    plt.text(df_sort['인구수'][n]*1.02, df_sort['소계'][n]*0.98, df_sort.index[n], fontsize=15)

plt.xlabel('인구수')
plt.ylabel('인구당비율')
plt.colorbar()
plt.grid()
plt.show()
```



서울시 범죄 현황 분석

- 데이터 읽기

```
import numpy as np
import pandas as pd

# 살인, 강도, 강간, 절도, 폭력이라는 5대 범죄에 대한 발생 건수와 검거 건수.
df = pd.read_csv('data/crime_in_Seoul.csv', encoding='euc-kr')
df.head()
```

	관서명	살인 발생	살인 검거	강도 발생	강도 검거	강간 발생	강간 검거	절도 발생	절도 검거	폭력 발생	폭력 검거
0	중부서	2	2	3	2	105	65	1,395	477	1,355	1,170
1	종로서	3	3	6	5	115	98	1,070	413	1,278	1,070
2	남대문서	1	0	6	4	65	46	1,153	382	869	794
3	서대문서	2	2	5	4	154	124	1,812	738	2,056	1,711
4	혜화서	3	2	5	4	96	63	1,114	424	1,015	861

서울시 범죄 현황 분석

- 경찰서 주소 경도/위도, 주소 데이터 획득

```
import googlemaps

# 자신의 api 키 입력
gmaps_key='******************'
gmaps = googlemaps.Client(key=gmaps_key)

# Google Maps를 사용하여 '서울중부경찰서' 검색
gmaps.geocode('서울중부경찰서', language='ko')
```


```
[{'address_components': [{'long_name': '2 7',
    'short_name': '2 7',
    'types': ['premise']},
   {'long_name': '수표로',
    'short_name': '수표로',
    'types': ['political', 'sublocality', 'sublocality_level_4']},
   {'long_name': '을지로동',
    'short_name': '을지로동',
    'types': ['political', 'sublocality', 'sublocality_level_2']},
   {'long_name': '중구',
    'short_name': '중구',
    'types': ['political', 'sublocality', 'sublocality_level_1']},
   {'long_name': '서울특별시',
    'short_name': '서울특별시',
    'types': ['administrative_area_level_1', 'political']},
   {'long_name': '대한민국',
    'short_name': 'KR',
    'types': ['country', 'political']},
   {'long_name': '100-032',
    'short_name': '100-032',
    'types': ['postal_code']}],
  'formatted_address': '대한민국 서울특별시 중구 을지로동 수표로 27',
  'geometry': {'location': {'lat': 37.5636465, 'lng': 126.9895796},
   'location_type': 'ROOFTOP',
   'viewport': {'northeast': {'lat': 37.56499548029149,
     'lng': 126.9909285802915},
    'southwest': {'lat': 37.56229751970849, 'lng': 126.9882306197085}}},
  'place_id': 'ChIJc-9q5uSifDURLhQmr5wkXmc',
  'plus_code': {'compound_code': 'HX7Q+FR 대한민국 서울특별시',
   'global_code': '8Q98HX7Q+FR'},
```

서울시 범죄 현황 분석

- 경찰서 이름 추가데이터 읽기

```
station_name=[]

for name in df['관서명']:
    station_name.append('서울' + str(name[:-1]) + '경찰서')

station_name
```

```
['서울중부경찰서',
 '서울종로경찰서',
 '서울남대문경찰서',
 '서울서대문경찰서',
 '서울혜화경찰서',
 '서울용산경찰서',
 '서울성북경찰서',
 '서울동대문경찰서',
 '서울마포경찰서',
 '서울영등포경찰서',
 '서울성동경찰서',
 '서울동작경찰서',
 '서울광진경찰서',
 '서울서부경찰서',
 '서울강북경찰서',
 '서울금천경찰서',
```

서울시 범죄 현황 분석

- 경찰서 이름으로 주소 획득

```
station_address = []
station_lat = []
station_lng = []

for name in station_name:
    tmp = gmaps.geocode(name, language='ko')
    station_address.append(tmp[0].get('formatted_address')) # 경찰서 주소 추가
    tmp_loc = tmp[0].get('geometry')
    station_lat.append(tmp_loc['location']['lat']) # 경찰서 위치의 위도 추가
    station_lng.append(tmp_loc['location']['lng']) # 경찰서 위치의 경도 추가
    print(name + '-->' + tmp[0].get('formatted_address'))
```

```
서울중부경찰서-->대한민국 서울특별시 중구 을지로동 수표로 27
서울종로경찰서-->대한민국 서울특별시 종로구 종로1.2.3.4가동 율곡로 46
서울남대문경찰서-->대한민국 서울특별시 중구 회현동 한강대로 410
서울서대문경찰서-->대한민국 서울특별시 서대문구 충현동 통일로 113
서울혜화경찰서-->대한민국 서울특별시 종로구 인의동 창경궁로 112-16
서울용산경찰서-->대한민국 서울특별시 용산구 원효로1가 백범로 329
서울성북경찰서-->대한민국 서울특별시 성북구 삼선동 보문로 170
```

서울시 범죄 현황 분석

- 주소에서 구 추출하여 구별 column 생성하여 추가

```
gu_name=[]

for name in station_address:
    tmp = name.split()
    tmp_gu = [gu for gu in tmp if gu[-1] == '구'][0]
    gu_name.append(tmp_gu)

df['구별'] = gu_name
df.head()
```

	관서명	살인 발생	살인 검거	강도 발생	강도 검거	강간 발생	강간 검거	절도 발생	절도 검거	폭력 발생	폭력 검거	구별
0	중부서	2	2	3	2	105	65	1,395	477	1,355	1,170	중구
1	종로서	3	3	6	5	115	98	1,070	413	1,278	1,070	종로구
2	남대문서	1	0	6	4	65	46	1,153	382	869	794	중구
3	서대문서	2	2	5	4	154	124	1,812	738	2,056	1,711	서대문구
4	혜화서	3	2	5	4	96	63	1,114	424	1,015	861	종로구

```
# 생성한 데이터프레임 저장
df.to_csv('data/crime_in_Seoul_gu_name.csv', sep=',', encoding='utf-8')
```

서울시 범죄 현황 분석

```
# 저장한 csv 파일 읽기
df_raw =pd.read_csv('data/crime_in_Seoul_gu_name.csv', thousands=',', encoding='utf-8')
df_raw.head()
```

	Unnamed: 0	관서명	살인 발생	살인 검거	강도 발생	강도 검거	강간 발생	강간 검거	절도 발생	절도 검거	폭력 발생	폭력 검거	구별
0	0	중부서	2	2	3	2	105	65	1395	477	1355	1170	중구
1	1	종로서	3	3	6	5	115	98	1070	413	1278	1070	종로구
2	2	남대문서	1	0	6	4	65	46	1153	382	869	794	중구
3	3	서대문서	2	2	5	4	154	124	1812	738	2056	1711	서대문구
4	4	혜화서	3	2	5	4	96	63	1114	424	1015	861	종로구

```
#index_col=0을 사용하여 index 변경
df_raw = pd.read_csv('data/crime_in_Seoul_gu_name.csv', thousands=',', encoding='utf-8', index_col=0)
df_raw.head()
```

	관서명	살인 발생	살인 검거	강도 발생	강도 검거	강간 발생	강간 검거	절도 발생	절도 검거	폭력 발생	폭력 검거	구별
0	중부서	2	2	3	2	105	65	1395	477	1355	1170	중구
1	종로서	3	3	6	5	115	98	1070	413	1278	1070	종로구
2	남대문서	1	0	6	4	65	46	1153	382	869	794	중구
3	서대문서	2	2	5	4	154	124	1812	738	2056	1711	서대문구
4	혜화서	3	2	5	4	96	63	1114	424	1015	861	종로구

서울시 범죄 현황 분석

```
# pivot_table을 이용하여 관서별에서 구별로 변경
crime_anal = pd.pivot_table(df_raw, index='구별',aggfunc=np.sum)
crime_anal.head()
```

구별	강간 검거	강간 발생	강도 검거	강도 발생	살인 검거	살인 발생	절도 검거	절도 발생	폭력 검거	폭력 발생
강남구	349	449	18	21	10	13	1650	3850	3705	4284
강동구	123	156	8	6	3	4	789	2366	2248	2712
강북구	126	153	13	14	8	7	618	1434	2348	2649
강서구	191	262	13	13	8	7	1260	2096	2718	3207
관악구	221	320	14	12	8	9	827	2706	2642	3298

서울시 범죄 현황 분석

```
# 각 검거율 계산하여 column 생성
crime_anal['강간검거율'] = crime_anal['강간 검거'] / crime_anal['강간 발생'] * 100
crime_anal['강도검거율'] = crime_anal['강도 검거'] / crime_anal['강도 발생'] * 100
crime_anal['살인검거율'] = crime_anal['살인 검거'] / crime_anal['살인 발생'] * 100
crime_anal['절도검거율'] = crime_anal['절도 검거'] / crime_anal['절도 발생'] * 100
crime_anal['폭력검거율'] = crime_anal['폭력 검거'] / crime_anal['폭력 발생'] * 100

# 검거 건수는 검거율로 대체할 수 있어서 삭제
del crime_anal['강간 검거']
del crime_anal['강도 검거']
del crime_anal['살인 검거']
del crime_anal['절도 검거']
del crime_anal['폭력 검거']

crime_anal.head()
```

구별	강간 발생	강도 발생	살인 발생	절도 발생	폭력 발생	강간검거율	강도검거율	살인검거율	절도검거율	폭력검거율
강남구	449	21	13	3850	4284	77.728285	85.714286	76.923077	42.857143	86.484594
강동구	156	6	4	2366	2712	78.846154	133.333333	75.000000	33.347422	82.890855
강북구	153	14	7	1434	2649	82.352941	92.857143	114.285714	43.096234	88.637222
강서구	262	13	7	2096	3207	72.900763	100.000000	114.285714	60.114504	84.752105
관악구	320	12	9	2706	3298	69.062500	116.666667	88.888889	30.561715	80.109157

서울시 범죄 현황 분석

```
#검거율이 100 넘는 숫자는 100으로 수정
con_list = ['강간검거율', '강도검거율', '살인검거율', '절도검거율', '폭력검거율']
for column in con_list:
    crime_anal.loc[crime_anal[column] > 100, column] = 100

crime_anal.head()
```

구별	강간 발생	강도 발생	살인 발생	절도 발생	폭력 발생	강간검거율	강도검거율	살인검거율	절도검거율	폭력검거율
강남구	449	21	13	3850	4284	77.728285	85.714286	76.923077	42.857143	86.484594
강동구	156	6	4	2366	2712	78.846154	100.000000	75.000000	33.347422	82.890855
강북구	153	14	7	1434	2649	82.352941	92.857143	100.000000	43.096234	88.637222
강서구	262	13	7	2096	3207	72.900763	100.000000	100.000000	60.114504	84.752105
관악구	320	12	9	2706	3298	69.062500	100.000000	88.888889	30.561715	80.109157

```
# 컬럼명에 발생이라는 단어 삭제.
crime_anal.rename(columns = {'강간 발생':'강간',
                             '강도 발생':'강도',
                             '살인 발생':'살인',
                             '절도 발생':'절도',
                             '폭력 발생':'폭력'},
                            inplace=True)

crime_anal.head()
```

구별	강간	강도	살인	절도	폭력	강간검거율	강도검거율	살인검거율	절도검거율	폭력검거율
강남구	449	21	13	3850	4284	77.728285	85.714286	76.923077	42.857143	86.484594
강동구	156	6	4	2366	2712	78.846154	100.000000	75.000000	33.347422	82.890855
강북구	153	14	7	1434	2649	82.352941	92.857143	100.000000	43.096234	88.637222
강서구	262	13	7	2096	3207	72.900763	100.000000	100.000000	60.114504	84.752105
관악구	320	12	9	2706	3298	69.062500	100.000000	88.888889	30.561715	80.109157

서울시 범죄 현황 분석

```
# 발생 건수 단위가 다름
#발생 건수 정규화
from sklearn import preprocessing

col = ['강간', '강도', '살인', '절도', '폭력']

x = crime_anal[col].values
min_max_scaler = preprocessing.MinMaxScaler()

x_scaled = min_max_scaler.fit_transform(x.astype(float))
crime_anal_norm = pd.DataFrame(x_scaled, columns = col,index = crime_anal.index)

col2 = ['강간검거율', '강도검거율', '살인검거율', '절도검거율', '폭력검거율']
crime_anal_norm[col2] = crime_anal[col2]
crime_anal_norm.head()
```

구별	강간	강도	살인	절도	폭력	강간검거율	강도검거율	살인검거율	절도검거율	폭력검거율
강남구	1.000000	0.941176	0.916667	1.000000	1.000000	77.728285	85.714286	76.923077	42.857143	86.484594
강동구	0.155620	0.058824	0.166667	0.467528	0.437969	78.846154	100.000000	75.000000	33.347422	82.890855
강북구	0.146974	0.529412	0.416667	0.133118	0.415445	82.352941	92.857143	100.000000	43.096234	88.637222
강서구	0.461095	0.470588	0.416667	0.370649	0.614945	72.900763	100.000000	100.000000	60.114504	84.752105
관악구	0.628242	0.411765	0.583333	0.589523	0.647479	69.062500	100.000000	88.888889	30.561715	80.109157

서울시 범죄 현황 분석

```
# CCTV 데이터 활용
result_CCTV = pd.read_csv('data/CCTV_result.csv', encoding='UTF-8', index_col='구별')
crime_anal_norm[['인구수','CCTV']] = result_CCTV[['인구수','소계']]
crime_anal_norm.head()
```

구별	강간	강도	살인	절도	폭력	강간검거율	강도검거율	살인검거율	절도검거율	폭력검거율	인구수	CCTV
강남구	1.000000	0.941176	0.916667	1.000000	1.000000	77.728285	85.714286	76.923077	42.857143	86.484594	570500.0	2780
강동구	0.155620	0.058824	0.166667	0.467528	0.437969	78.846154	100.000000	75.000000	33.347422	82.890855	453233.0	773
강북구	0.146974	0.529412	0.416667	0.133118	0.415445	82.352941	92.857143	100.000000	43.096234	88.637222	330192.0	748
강서구	0.461095	0.470588	0.416667	0.370649	0.614945	72.900763	100.000000	100.000000	60.114504	84.752105	603772.0	884
관악구	0.628242	0.411765	0.583333	0.589523	0.647479	69.062500	100.000000	88.888889	30.561715	80.109157	525515.0	1496

```
# 각 범죄 발생 건수에 대해 합을 구해 '범죄'라는 column 생성
col = ['강간', '강도', '살인', '절도', '폭력']
crime_anal_norm['범죄'] = np.sum(crime_anal_norm[col],axis=1)
crime_anal_norm.head()
```

구별	강간	강도	살인	절도	폭력	강간검거율	강도검거율	살인검거율	절도검거율	폭력검거율	인구수	CCTV	범죄
강남구	1.000000	0.941176	0.916667	1.000000	1.000000	77.728285	85.714286	76.923077	42.857143	86.484594	570500.0	2780	4.857843
강동구	0.155620	0.058824	0.166667	0.467528	0.437969	78.846154	100.000000	75.000000	33.347422	82.890855	453233.0	773	1.286607
강북구	0.146974	0.529412	0.416667	0.133118	0.415445	82.352941	92.857143	100.000000	43.096234	88.637222	330192.0	748	1.641616
강서구	0.461095	0.470588	0.416667	0.370649	0.614945	72.900763	100.000000	100.000000	60.114504	84.752105	603772.0	884	2.333944
관악구	0.628242	0.411765	0.583333	0.589523	0.647479	69.062500	100.000000	88.888889	30.561715	80.109157	525515.0	1496	2.860342

서울시 범죄 현황 분석

```
# 각 검거율 합을 구해 '검거'라는 column 생성
col = ['강간검거율','강도검거율','살인검거율','절도검거율','폭력검거율']
crime_anal_norm['검거'] = np.sum(crime_anal_norm[col], axis=1)
crime_anal_norm.head()
```

구별	강간	강도	살인	절도	폭력	강간검거율	강도검거율	살인검거율	절도검거율	폭력검거율	인구수	CCTV	범죄	검거
강남구	1.000000	0.941176	0.916667	1.000000	1.000000	77.728285	85.714286	76.923077	42.857143	86.484594	570500.0	2780	4.857843	369.707384
강동구	0.155620	0.058824	0.166667	0.467528	0.437969	78.846154	100.000000	75.000000	33.347422	82.890855	453233.0	773	1.286607	370.084431
강북구	0.146974	0.529412	0.416667	0.133118	0.415445	82.352941	92.857143	100.000000	43.096234	88.637222	330192.0	748	1.641616	406.943540
강서구	0.461095	0.470588	0.416667	0.370649	0.614945	72.900763	100.000000	100.000000	60.114504	84.752105	603772.0	884	2.333944	417.767372
관악구	0.628242	0.411765	0.583333	0.589523	0.647479	69.062500	100.000000	88.888889	30.561715	80.109157	525515.0	1496	2.860342	368.622261

서울시 범죄 현황 분석

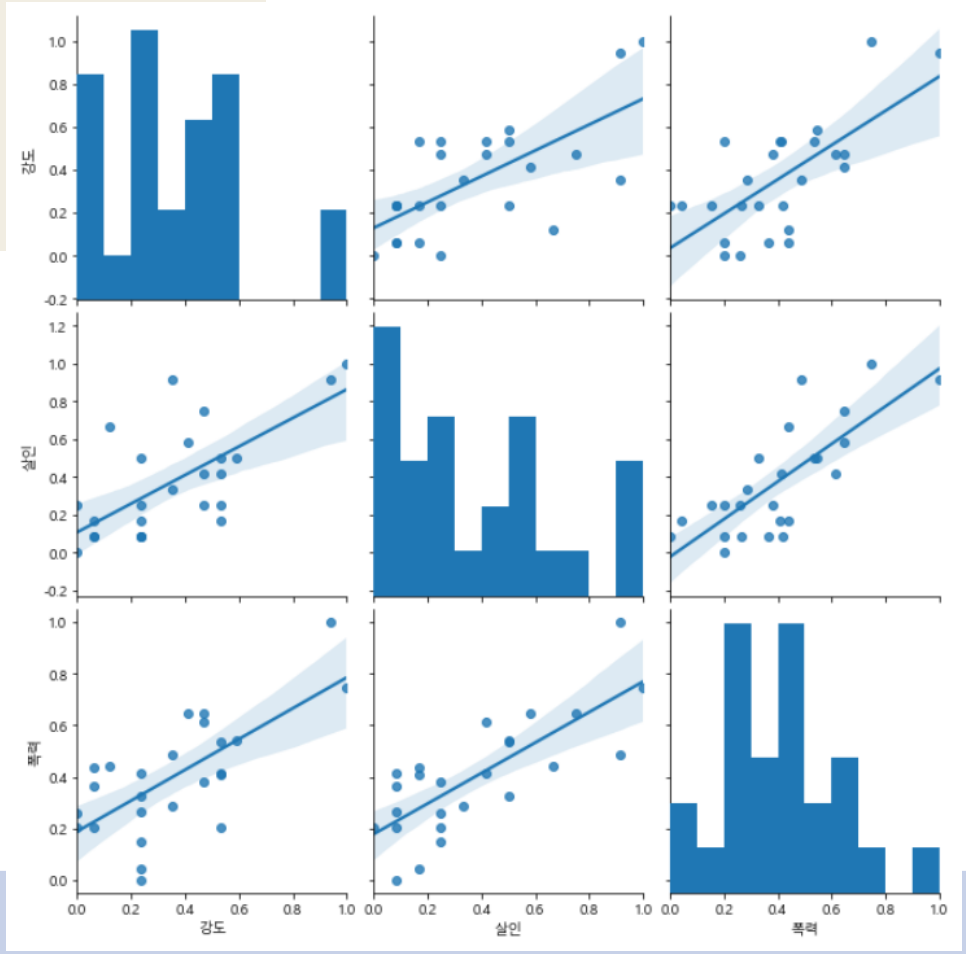
- seaborn을 이용한 시각화

```
# 한글 폰트 적용
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

from matplotlib import font_manager, rc
plt.rcParams['axes.unicode_minus']=False

path='c:/Windows/Fonts/malgun.ttf'
font_name=font_manager.FontProperties(fname=path).get_name()
rc('font',family=font_name)
```

서울시 범죄 현황 분석

- 시각화

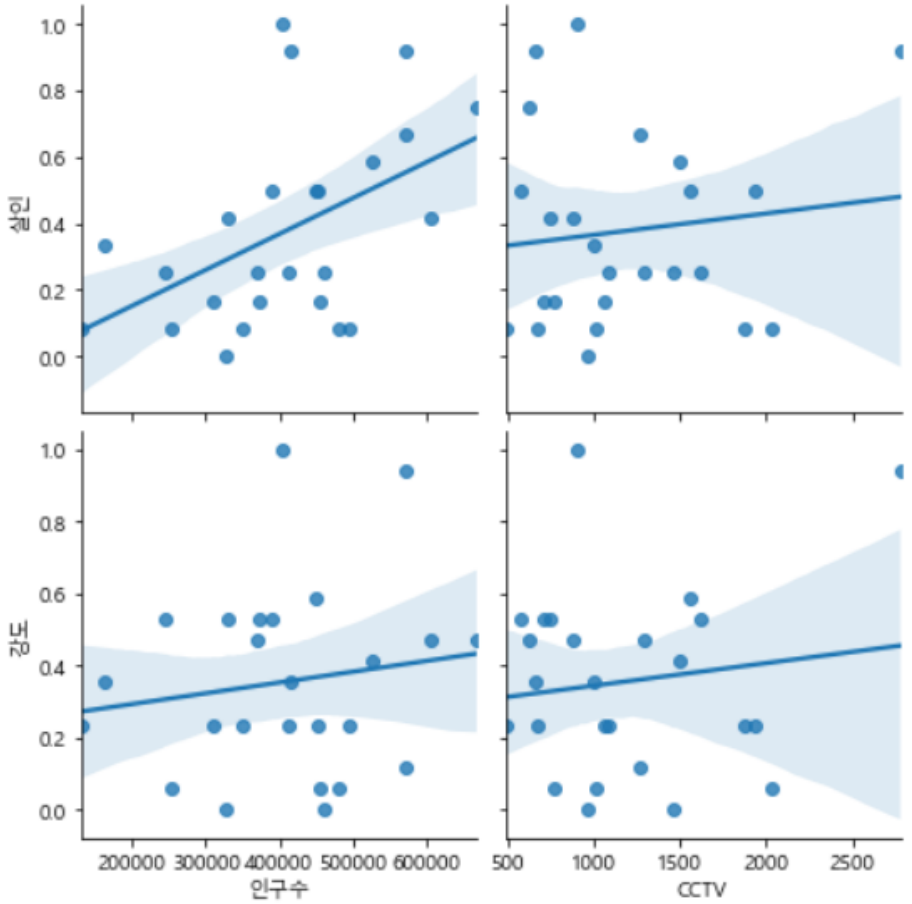
```
# 강도와 폭력, 살인과 폭력, 강도와 살인의 상관관계 확인
sns.pairplot(crime_anal_norm,
             vars=['강도', '살인', '폭력'],
             kind='reg', height=3)
plt.show()
```



서울시 범죄 현황 분석

- 시각화

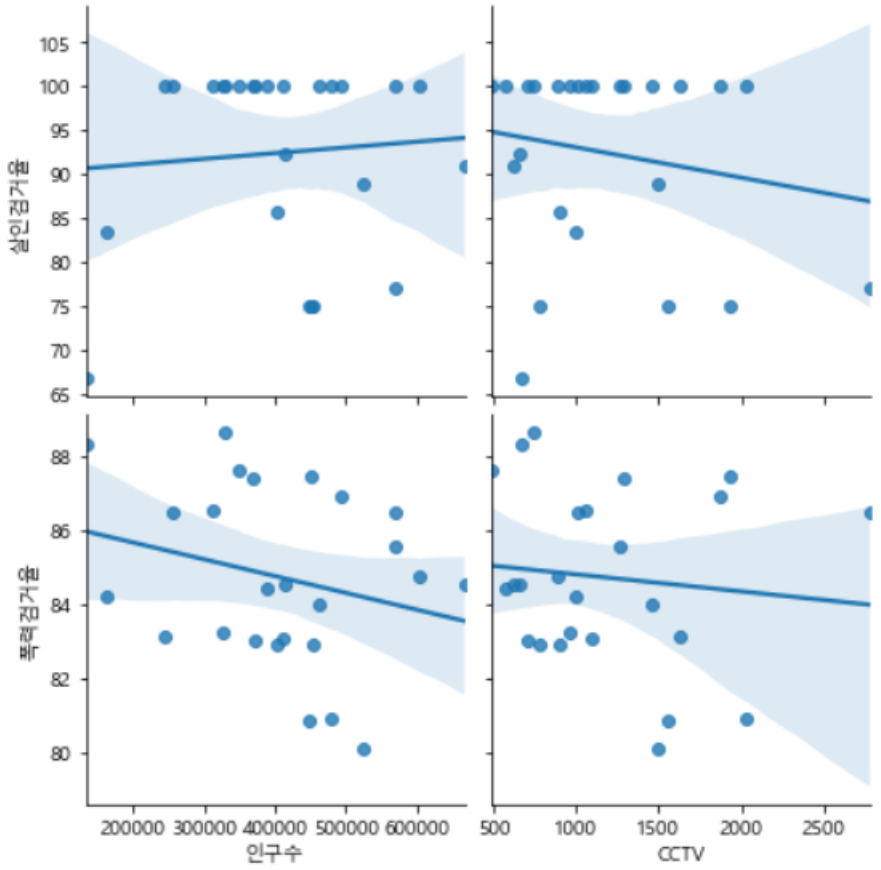
```
# 인구수, CCTV개수와 살인, 강도의 상관관계
sns.pairplot(crime_anal_norm,
             x_vars=['인구수', 'CCTV'],
             y_vars=['살인', '강도'],
             kind='reg', height=3)
plt.show()
```



서울시 범죄 현황 분석

- 시각화

```
 #인구수, CCTV와 살인검거율, 폭력검거율의 상관관계
sns.pairplot(crime_anal_norm,
             x_vars=['인구수', 'CCTV'],
             y_vars=['살인검거율', '폭력검거율'],
             kind='reg', height=3)
plt.show()
```



서울시 범죄 현황 분석

```
# 검거율의 합계인 검거 항목 최고 값을 100으로 한정하고 그 값으로 정렬
tmp_max = crime_anal_norm['검거'].max()
crime_anal_norm['검거'] = crime_anal_norm['검거'] / tmp_max * 100
crime_anal_nrom_sort = crime_anal_norm.sort_values(by='검거', ascending=False)
crime_anal_nrom_sort.head()
```

구별	강간	강도	살인	절도	폭력	강간검거율	강도검거율	살인검거율	절도검거율	폭력검거율	인구수	CCTV	범죄	검거
도봉구	0.000000	0.235294	0.083333	0.000000	0.000000	100.000000	100.0	100.0	44.967074	87.626093	348646.0	485	0.318627	100.000000
금천구	0.141210	0.058824	0.083333	0.180840	0.202717	80.794702	100.0	100.0	56.668794	86.465433	255082.0	1015	0.666924	97.997139
강서구	0.461095	0.470588	0.416667	0.370649	0.614945	72.900763	100.0	100.0	60.114504	84.752105	603772.0	884	2.333944	96.572809
광진구	0.397695	0.529412	0.166667	0.704342	0.406864	91.666667	100.0	100.0	42.200925	83.047619	372164.0	707	2.204979	96.375820
동대문구	0.204611	0.470588	0.250000	0.329386	0.379335	84.393064	100.0	100.0	41.090358	87.401884	369496.0	1294	1.633921	95.444250

서울시 범죄 현황 분석

```
# 범죄 검거 비율 heatmap으로 시각화
target_col = ['강간검거율', '강도검거율',
'살인검거율', '절도검거율', '폭력검거율']

crime_anal_norm_sort =
crime_anal_norm.sort_values(by='검거',
ascending=False)

plt.figure(figsize = (10,10))
sns.heatmap(crime_anal_norm_sort[target_col],
annot=True, fmt='f',linewidths=.5, cmap='RdPu')
plt.title('범죄 검거 비율 (정규화된 검거의 합으로 정렬)')
plt.show()
```

서울시 범죄 현황 분석

```
# 범죄 발생 건수 heatmap 시각화
target_col=['강간','강도','살인','절도','폭력','범죄']

crime_anal_norm['범죄'] = crime_anal_norm['범죄'] / 5
crime_anal_norm_sort = crime_anal_norm.sort_values(by='범죄',
ascending=False)

plt.figure(figsize=(10,10))

sns.heatmap(crime_anal_norm_sort[target_col],
annot=True, fmt='f', linewidth=.5)
plt.title('범죄비율 (정규화된 발생 건수로 정렬)')
plt.show()
```

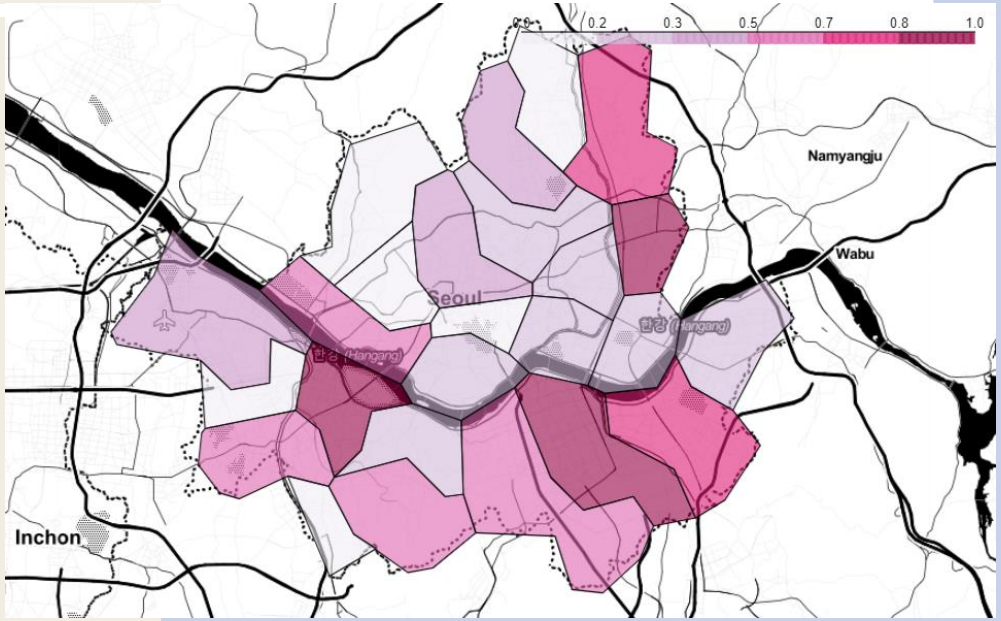

```
# 결과 저장
crime_anal_norm.to_csv('data/crime_in_Seoul_final.csv', sep=',', encoding='utf-8')
```

서울시 범죄 현황 분석

- 지도 시각화

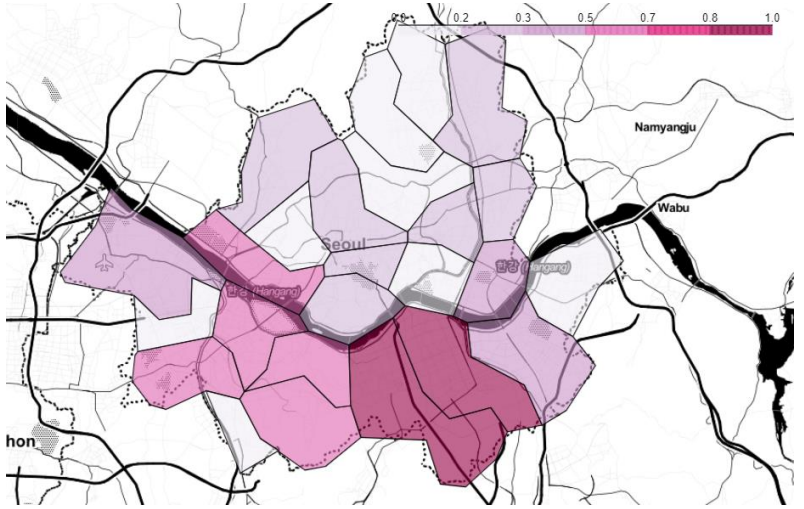
```
import folium
import matplotlib.pyplot as plt


import json
geo_path = 'data/skorea_municipalities_geo_simple.json'
geo_str = json.load(open(geo_path, encoding='utf-8')
```

```
# 서울시 중심 경계선 나타내기. 컬러맵 살인 발생 건수.
map = folium.Map(location=[37.5502, 126.982],
                 zoom_start=11, tiles='Stamen Toner')
map.choropleth(geo_data = geo_str,
               data = crime_anal_norm['살인'],
               columns = [crime_anal_norm.index,
                          crime_anal_norm['살인']],
               fill_color = 'PuRd', #puRd, YlGnBu
               key_on = 'feature.id')
map
```

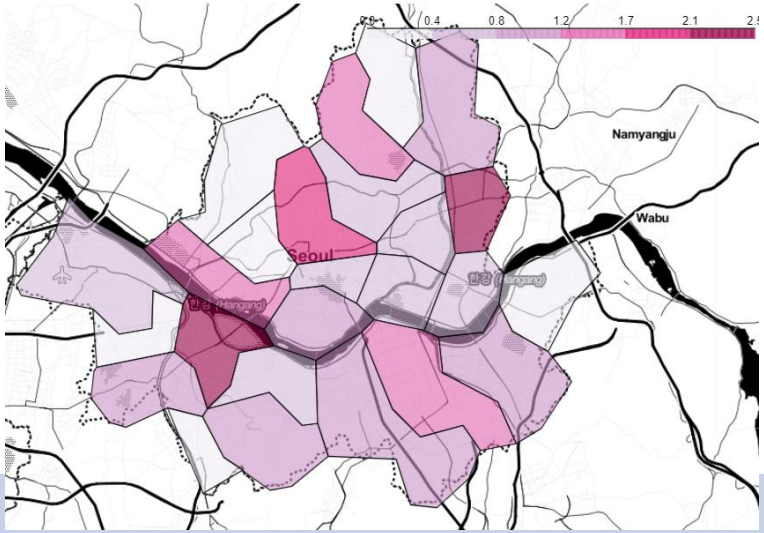


서울시 범죄 현황 분석

```
# 강간 발생 건수
map = folium.Map(location=[37.5502, 126.982], zoom_start=11,
                 tiles='Stamen Toner')
map.choropleth(geo_data = geo_str,
               data = crime_anal_norm['강간'],
               columns = [crime_anal_norm.index,
                          crime_anal_norm['강간']],
               fill_color = 'PuRd', #puRd, YlGnBu
               key_on = 'feature.id')
map
```




```
# 범죄 발생건수
map = folium.Map(location=[37.5502, 126.982], zoom_start=11,
                 tiles='Stamen Toner')
map.choropleth(geo_data = geo_str,
               data = crime_anal_norm['범죄'],
               columns = [crime_anal_norm.index,
                          crime_anal_norm['범죄']],
               fill_color = 'PuRd', #puRd, YlGnBu
               key_on = 'feature.id')
map
```



서울시 범죄 현황 분석

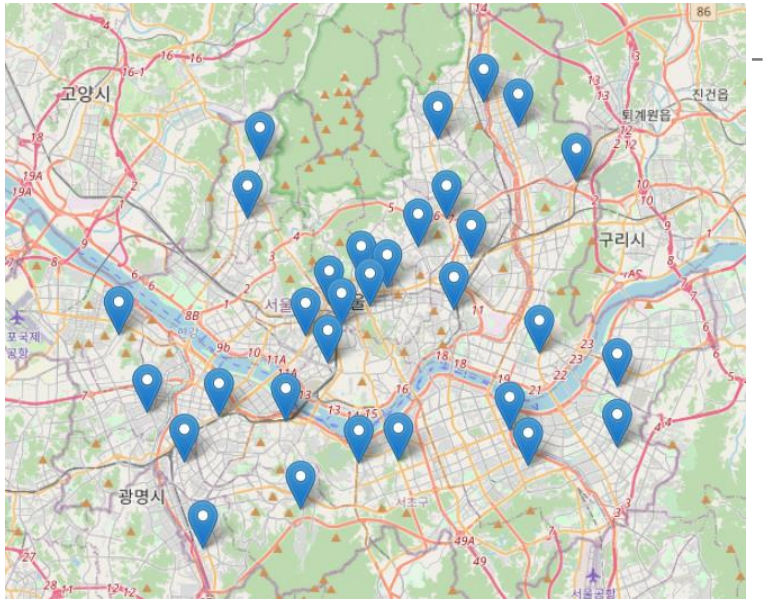
```
# 서울시 경찰서별 검거율과 구별 범죄 발생율을 동시 시각화
# 경찰서의 위도와 경도 정보 이용
df_raw['lat'] = station_lat
df_raw['lng'] = station_lng

col = ['살인 검거', '강도 검거', '강간 검거', '절도 검거', '폭력 검거']
tmp =df_raw[col] / df_raw[col].max()

df_raw['검거'] = np.sum(tmp, axis=1)
```

```
df_raw.head()
```

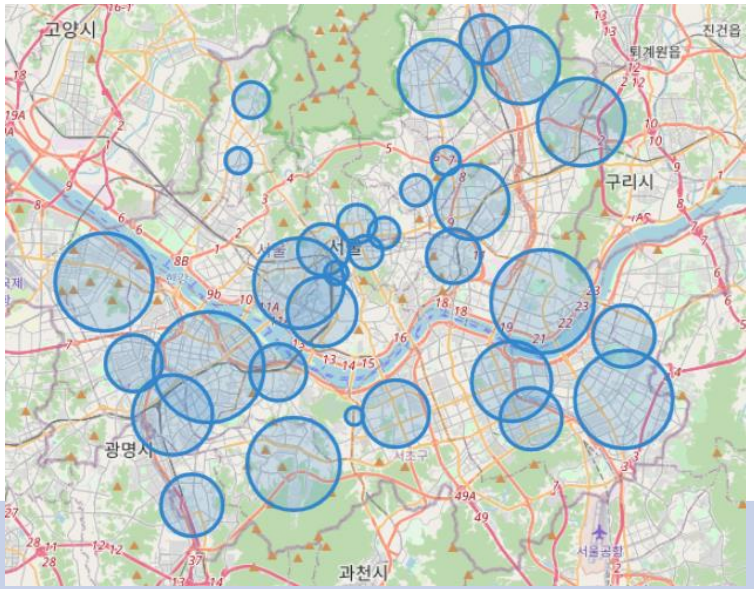
	관서명	살인 발생	살인 검거	강도 발생	강도 검거	강간 발생	강간 검거	절도 발생	절도 검거	폭력 발생	폭력 검거	구별	lat	lng	검거
0	중부서	2	2	3	2	105	65	1395	477	1355	1170	중구	37.563646	126.989580	1.275416
1	종로서	3	3	6	5	115	98	1070	413	1278	1070	종로구	37.575548	126.984747	1.523847
2	남대문서	1	0	6	4	65	46	1153	382	869	794	중구	37.554758	126.973498	0.907372
3	서대문서	2	2	5	4	154	124	1812	738	2056	1711	서대문구	37.564744	126.966770	1.978299
4	혜화서	3	2	5	4	96	63	1114	424	1015	861	종로구	37.571853	126.998914	1.198382

서울시 범죄 현황 분석

```
#경찰서 위치 표시.
map = folium.Map(location=[37.5502, 126.982], zoom_start=11)

for n in df_raw.index:
    folium.Marker([df_raw['lat'][n],
                   df_raw['lng'][n]]).add_to(map)

map
```

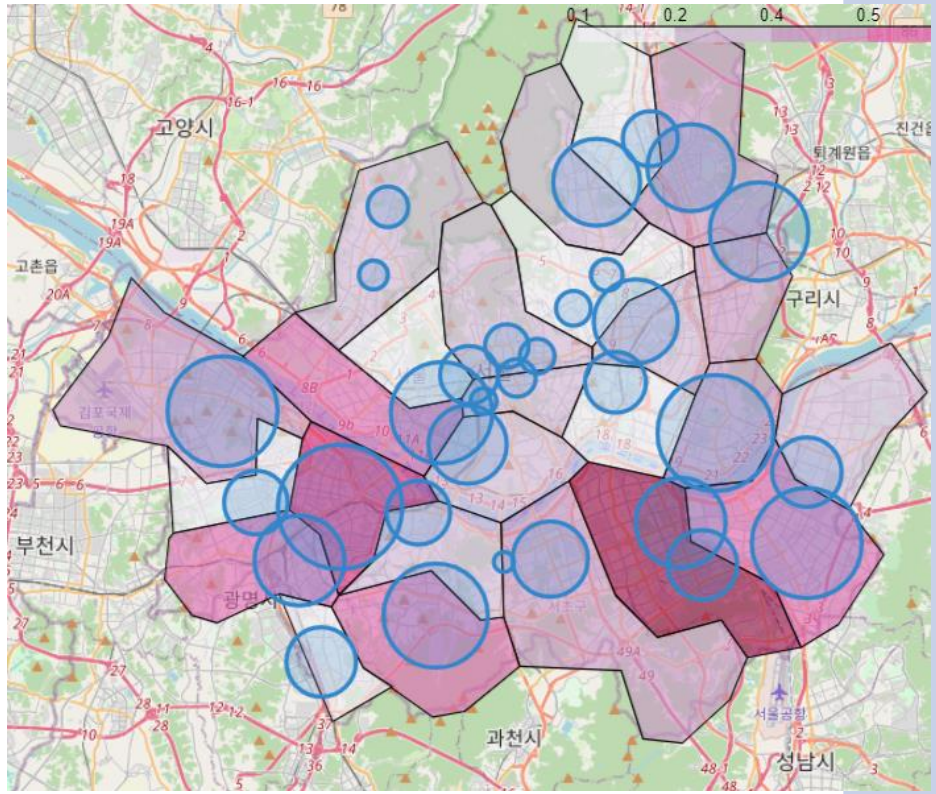



```
# 검거에 10을 곱해서 원 넓이 정함
# 경찰서의 검거율을 원의 넓이로 표현
map =folium.Map(location=[37.5502, 126.982], zoom_start=11)

for n in df_raw.index:
    folium.CircleMarker([df_raw['lat'][n], df_raw['lng'][n]],
                        radius = df_raw['검거'][n]*10,
                        color='#3186cc',
fill_color='#3186cc').add_to(map)
map
```



서울시 범죄 현황 분석

```
# 범죄 발생 건수 추가.
map = folium.Map(location=[37.5502, 126.982], zoom_start=11)

map.choropleth(geo_data = geo_str,
               data = crime_anal_norm['범죄'],
               columns = [crime_anal_norm.index,
                          crime_anal_norm['범죄']],
               fill_color = 'PuRd', #PuRd, YlGnBu
               key_on='feature.id')

for n in df_raw.index:
    folium.CircleMarker([df_raw['lat'][n], df_raw['lng'][n]],
                        radius = df_raw['검거'][n]*10,
                        color='#3186cc',
fill_color='#3186cc').add_to(map)
map
```



감사합니다