학술제 - 사용한 코드 정리

- 사용한 라이브러리

```
# 데이터 프레임
import numpy as np
import pandas as pd

# 사용한 기법(최소-최대 정규화, PCA, K-means 군집화, 엘보우 메소드)
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer

# 그래프 시각화, 한글 호환
import seaborn as sns
from matplotlib import font_manager, rc
import matplotlib.pyplot as plt

# 좌표계 변환
import pyproj

# 지도 시각화
import folium
from folium import Circle, Marker
```

- 기본 설정

```
# 데이터프레임 row 생략 없이 출력
pd.set_option('display.max_rows', None)

# 시각화 할때 한글 호환
from matplotlib import font_manager, rc
font_path = 'C:/Users/KJY/.ipython/나눔 글꼴/나눔고딕/NanumFontSetup_TTF_GOTHIC/NanumGothic.ttf'
font = font_manager.FontProperties(fname=font_path).get_name()
rc('font', family=font)
```

- 데이터 불러오기

```
# 주유소 면적, 공시지가, 범위 내 행정동 정리
gas_station = pd.read_csv('C:/Users/KJY/Desktop/학술제/주유소 별 범위 데이터 최종.csv')

# 이상치를 제거한 140개의 주유소 데이터
df1= pd.read_csv('C:/Users/KJY/Desktop/학술제/완전 필터링 된 주유소 데이터.csv')

# 동별 데이터 정리
dong_value = pd.read_csv("C:/Users/KJY/desktop/학술제/동별 데이터.csv")

# 구별 데이터 정리
df2 = pd.read_csv('C:/Users/KJY/Desktop/학술제/구별 데이터.csv')
```

데이터 프레임 정리

- **주유소 3km범위 안 포함되는 행정동을 모두 구했고, 주유소 별 데이터를 정리함**

```
dong_df = gas_station.iloc[0: ,13:]; dong_df.head() # 14번째 열부터 범위 내 행정동 데이터
dong_df = dong_df.transpose() # 행, 열 전환을 통해 병합을 가능하게 함
dong_df.reset_index(drop=True) # 인덱스 초기화, 기존 인덱스 삭제
```

	0	1	2	3	4	5	6	7	8	9	...	31	32	33	34	35	36	37	38	39	40
0	강남구 역삼2동	강남구 도곡1동	강남구 도곡2동	강남구 대치1동	강남구 개포1동	강남구 개포2동	강남구 개포4동	서초구 내곡동	서초구 양재1동	서초구 양재2동	...	.	.	.	.	.	.	.	.	.	.
1	강남구 삼성1동	강남구 삼성2동	강남구 대치1동	강남구 대치2동	강남구 대치4동	강남구 역삼2동	강남구 도곡2동	강남구 개포1동	강남구 개포2동	강남구 일원2동	...	.	.	.	.	.	.	.	.	.	.
2	강남구 청담동	강남구 논현2동	강남구 역삼1동	강남구 역삼2동	강남구 도곡1동	강남구 도곡2동	강남구 개포1동	강남구 개포2동	강남구 일원2동	강남구 개포4동	...	.	.	.	.	.	.	.	.	.	.
3	강남구 삼성1동	강남구 삼성2동	강남구 역삼1동	강남구 역삼2동	강남구 도곡1동	강남구 도곡2동	강남구 개포1동	강남구 개포2동	강남구 일원2동	강남구 개포4동	...	.	.	.	.	.	.	.	.	.	.
4	서초구 서초1동	서초구 서초2동	서초구 서초3동	서초구 서초4동	서초구 양재1동	강남구 도곡1동	강남구 도곡2동	강남구 개포1동	강남구 개포2동	강남구 개포4동	...	.	.	.	.	.	.	.	.	.	.

```
dong_value = pd.read_csv("C:/Users/KJY/desktop/학술제/동별 데이터.csv") # 동별 데이터
dong_value.head()
dong_value.columns = ['시', '구', '동', '행정동', '시,구.동', '1인가구', '인구수',
                      '종사자수', '유통점합계', '아파트', '단독, 연립, 다세대', '비주거용',
                      '자동차 등록대수', '소득'] # 열 이름 재지정
```

	시	구	동	행정동	시,구.동	1인가구	인구수	종사자수	유통점합계	아파트	단독, 연립, 다세대	비주거용	자동차 등록대수	소득
0	NaN	동별(2)	동별(3)	NaN	NaN	1인가구 (가구)	인구수	종사자수	유통점합계	아파트	단독,연립,다세대 주택	비주거용 건물	자동차 등록대수	소득데이터
1	NaN	소계	소계	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	서울특별시	종로구	사직동	종로구 사직동	서울특별시 종로구 사직동	1487	9326	56013	80	1591	1139	48	8317	5038049
3	NaN	NaN	삼청동	종로구 삼청동	서울특별시 종로구 삼청동	380	2495	4978	27	0	700	30	788	3591723
4	NaN	NaN	부암동	종로구 부암동	서울특별시 종로구 부암동	1129	9443	3762	22	121	2912	37	3079	3622510

첫 번째 데이터 프레임의 각 열과 두 번째 데이터프레임의 행정동 열을 병합하면 주유소 별 데이터를 구할 수 있다.

```
num_col = col[5:]; num_col # 수치형 자료의 열만 가져옴

df_check=[]
df_gas=[]
for i in range(len(dong_df.columns)):
    df = pd.merge(dong_df[i].dropna(), dong_value,left_on = i, right_on = '행정동', how='inner')
    df_check.append(df)
```

```
    df_num = df[num_col].astype(float)
    df_sum = pd.DataFrame(df_num.sum()).transpose()
    df_gas.append(df_sum)
    df_gas[i]['소득'] = df_gas[i]['소득'] / len(df_check[i]) # 소득은 평균값을 이용해줄거다.

# 리스트 내 데이터 프레임을 병합해 하나로 만들어 줌
merged_df = pd.concat(df_gas, ignore_index=True)
merged_df
merged_df.head()
```

	1인가구	인구수	종사자수	유통점합계	아파트	단독, 연립, 다세대	비주거용	자동차 등록대수	소득
0	25788.0	293966.0	291239.0	720.0	65444.0	19380.0	845.0	132754.0	5.030322e+06
1	51845.0	506852.0	433436.0	1422.0	117118.0	46463.0	910.0	200910.0	4.913616e+06
2	64855.0	466917.0	688269.0	1557.0	94649.0	56190.0	1639.0	188151.0	4.669938e+06
3	57678.0	513354.0	592328.0	1529.0	111082.0	49395.0	1208.0	211698.0	4.953976e+06
4	48661.0	431380.0	673124.0	1308.0	94164.0	33722.0	1173.0	206503.0	5.179666e+06

- **면적, 제곱미터당 공시 지가 데이터 추가**

```
area_df = gas_station[['소재지면적', '공시지가']]
gas_station_num = pd.concat([merged_df, area_df], axis=1)
```

	1인가구	인구수	종사자수	유통점합계	아파트	단독, 연립, 다세대	비주거용	소득	소재지면적	공시지가	1인당 자동차 등록대수
0	25788.0	293966.0	291239.0	720.0	65444.0	19380.0	845.0	5.030322e+06	991	3248	0.451596
1	51845.0	506852.0	433436.0	1422.0	117118.0	46463.0	910.0	4.913616e+06	1288	4613	0.396388
2	64855.0	466917.0	688269.0	1557.0	94649.0	56190.0	1639.0	4.669938e+06	1008	4613	0.402965
3	57678.0	513354.0	592328.0	1529.0	111082.0	49395.0	1208.0	4.953976e+06	1529	4613	0.412382
4	48661.0	431380.0	673124.0	1308.0	94164.0	33722.0	1173.0	5.179666e+06	2126	3760	0.478703

- **논문과 동일하게 1인당 자동차 등록대수를 변수로 이용**

```
gas_station_num['1인당 자동차 등록대수'] = gas_station_num['자동차 등록대수'] / gas_station_num['인구수']
gas_station_num.drop('자동차 등록대수', axis=1, inplace = True) # 기존 열은 제거
gas_station_num.head()
```

	1인가구	인구수	종사자수	유통점합계	아파트	단독, 연립, 다세대	비주거용	소득	소재지면적	공시지가	1인당 자동차 등록대수
0	25788.0	293966.0	291239.0	720.0	65444.0	19380.0	845.0	5.030322e+06	991	3248	0.451596
1	51845.0	506852.0	433436.0	1422.0	117118.0	46463.0	910.0	4.913616e+06	1288	4613	0.396388
2	64855.0	466917.0	688269.0	1557.0	94649.0	56190.0	1639.0	4.669938e+06	1008	4613	0.402965
3	57678.0	513354.0	592328.0	1529.0	111082.0	49395.0	1208.0	4.953976e+06	1529	4613	0.412382
4	48661.0	431380.0	673124.0	1308.0	94164.0	33722.0	1173.0	5.179666e+06	2126	3760	0.478703

- **사용하지 않는 행 제거 후 기초통계량 확인**

```
# 0을 가지고 있는 행, 면적 말도안되게 큰 거 제거
# 면적 5000이상은 이상치라 생각하고 제거함
gas_station_clean = gas_station_num[(gas_station_num['1인가구']!= 0) & (gas_station_num['소재지면적']<5000)]
gas_station_clean.reset_index(drop=True, inplace=True)
gas_station_clean.describe()
```

	1인가구	인구수	종사자수	유통점합계	아파트	단독, 연립, 다세대	비주거용	소득	소재지면적	공시지가	1인당 자동차 등
count	140.000000	140.000000	140.000000	140.000000	140.000000	140.000000	140.000000	1.400000e+02	140.000000	140.000000	140.0
mean	66599.314286	463256.264286	283966.192857	1576.928571	87514.557143	56781.242857	1376.464286	3.453518e+06	1462.978571	4426.871429	0.3
std	27456.501884	150271.105197	189794.922966	654.436290	31193.699148	24501.923589	669.641399	6.980231e+05	501.335403	1034.833910	0.0
min	4329.000000	30316.000000	34394.000000	98.000000	3044.000000	4781.000000	63.000000	2.570427e+06	977.000000	1510.000000	0.2
25%	51015.750000	389270.000000	140319.750000	1230.750000	70547.000000	40883.500000	979.000000	2.880472e+06	1064.000000	3689.000000	0.2
50%	65371.000000	492529.000000	206860.500000	1561.000000	87666.500000	52590.000000	1331.000000	3.221215e+06	1320.000000	4613.000000	0.3
75%	84758.500000	566728.250000	396408.500000	1936.250000	107473.500000	76049.500000	1756.750000	3.788511e+06	1660.000000	4840.250000	0.3
max	155249.000000	758770.000000	887989.000000	3425.000000	156131.000000	111169.000000	3061.000000	5.179666e+06	3778.000000	8476.000000	0.4

```
gas_station_clean.shape

# (140, 11)
```

- **csv파일 내에서 주유소를 필터링 후 도로명주소, 사업장명 불러옴**

```
df1= pd.read_csv('C:/Users/KJY/Desktop/학술제/완전 필터링 된 주유소 데이터.csv')
df1 = df1[['도로명주소', '사업장명']]
df1['도로명주소'] = df1['도로명주소'].apply(lambda x: x.split(' ')[0] + ' ' + x.split(' ')[1])
# 병합해야 하기 때문에 시•구 형태로 만들어줌
df1.head()
```

- **합친 후 열 순서 변경**

```
gas_station_index= gas_station_index[['도로명주소', '사업장명', '1인가구', '인구수',
                                      '종사자수', '유통점합계', '아파트','단독, 연립, 다세대',
                                      '비주거용', '소득', '소재지면적', '공시지가', '1인당 자동차 등록대수']]
gas_station_index.head()
```

	도로명주소	사업장명	1인가구	인구수	종사자수	유통점합계	아파트	단독, 연립, 다세대	비주거용	소득	소재지면적	공시지가	1인당 자동차 등록대수
0	서울특별시 강남구	대양산업(주)개포주유소	25788.0	293966.0	291239.0	720.0	65444.0	19380.0	845.0	5030322	991	3248	0.451596
1	서울특별시 강남구	지에스칼텍스(주)학여울주유소	51845.0	506852.0	433436.0	1422.0	117118.0	46463.0	910.0	4913616	1288	4613	0.396388
2	서울특별시 강남구	㈜소모 센트럴주유소	64855.0	466917.0	688269.0	1557.0	94649.0	56190.0	1639.0	4669937	1008	4613	0.402965
3	서울특별시 강남구	은마주유소	57678.0	513354.0	592328.0	1529.0	111082.0	49395.0	1208.0	4953975	1529	4613	0.412382
4	서울특별시 강남구	에이치디현대오일뱅크(주)직영 유진주유소	48661.0	431380.0	673124.0	1308.0	94164.0	33722.0	1173.0	5179665	2126	3760	0.478703

- **구별 데이터 불러오기**

```
df2 = pd.read_csv('C:/Users/KJY/Desktop/학술제/구별 데이터.csv')
df2.head()
```

	Unnamed: 0	Unnamed: 1	1인가구 (20-39세)	유동인구	지방세	1인당 지방세
0	종로구	서울특별시 종로구	13678	1801392	1175671	7.644702
1	중구	서울특별시 중구	11406	2274160	1776265	13.478302
2	용산구	서울특별시 용산구	20884	1757579	1187660	5.005205
3	성동구	서울특별시 성동구	22783	1583287	787894	2.692072
4	광진구	서울특별시 광진구	39617	1785534	485028	1.375470

- **구별 데이터 기존 데이터프레임에 병합**

```
gas_station_total = pd.merge(gas_station_index, df2, left_on='도로명주소', right_on = 'Unnamed: 1', how='left')
gas_station_total.drop(['Unnamed: 0', 'Unnamed: 1'], axis=1, inplace = True) # 불필요열 제거
gas_station_total.head()
```

- **분석을 위해 수치형 데이터만 가져옴**

```
gas_station_total_num = gas_station_total.iloc[:, 2:]
gas_station_total_num.head()
```

	1인가구	인구수	종사자수	유통점합계	아파트	단독, 연립, 다세대	비주거용	소득	소재지면적	공시지가	1인당 자동차 등록대수	1인가구 (20-39세)	유동인구	지방세	1인당 지방세
0	25788.0	293966.0	291239.0	720.0	65444.0	19380.0	845.0	5030322	991	3248	0.451596	36502	5659671	4617990	8.586817
1	51845.0	506852.0	433436.0	1422.0	117118.0	46463.0	910.0	4913616	1288	4613	0.396388	36502	5659671	4617990	8.586817
2	64855.0	466917.0	688269.0	1557.0	94649.0	56190.0	1639.0	4669937	1008	4613	0.402965	36502	5659671	4617990	8.586817
3	57678.0	513354.0	592328.0	1529.0	111082.0	49395.0	1208.0	4953975	1529	4613	0.412382	36502	5659671	4617990	8.586817
4	48661.0	431380.0	673124.0	1308.0	94164.0	33722.0	1173.0	5179665	2126	3760	0.478703	36502	5659671	4617990	8.586817

데이터 전처리

```
scaler = MinMaxScaler() # 최대-최소 정규화
normalized_data = scaler.fit_transform(gas_station_total_num)
```

	1인가구	인구수	종사자수	유통점합계	아파트	단독, 연립, 다세대	비주거용	소득	소재지면적	공시지가	1인당 자동차 등록대수	1인가구 (20-39세)	유동인구	지방세	1인당 지방세
0	0.142188	0.361931	0.300898	0.186955	0.407611	0.137224	0.260841	0.942764	0.004998	0.249498	0.860204	0.297857	1.0	1.0	0.613565
1	0.314842	0.654174	0.467484	0.397956	0.745158	0.391792	0.282522	0.898036	0.111032	0.445449	0.625117	0.297857	1.0	1.0	0.613565
2	0.401047	0.599353	0.766025	0.438533	0.598385	0.483222	0.525684	0.804645	0.011067	0.445449	0.653121	0.297857	1.0	1.0	0.613565
3	0.353492	0.663100	0.653628	0.430117	0.705729	0.419352	0.381921	0.913504	0.197072	0.445449	0.693223	0.297857	1.0	1.0	0.613565
4	0.293745	0.550569	0.748282	0.363691	0.595217	0.272033	0.370247	1.000000	0.410211	0.322997	0.975628	0.297857	1.0	1.0	0.613565

<표 4-5> 회전된 성분행렬

변수	성분				
	1	2	3	4	5
소비	.943	-.022	.171	-.007	-.166
지방세	.937	-.022	.143	.102	-.138
1인당 자동차 등록대수	.924	-.051	-.184	-.049	-.041
종사자수	.916	.043	.240	-.082	.146
1인당 지방세 부담액	.911	-.075	.276	-.194	.043
소득	.857	-.208	.286	.137	-.214
단독·연립·다세대 주택	-.756	.213	.164	.301	-.107
20~39세 1인 가구	-.200	.890	.266	.079	.054
주택이외 거처	.243	.821	-.253	-.102	.094
1인 가구	-.437	.785	.206	.286	.128
유동인구	.103	.142	.764	.002	.350
임대료	.071	.198	.677	.093	-.386
공시지가	.522	-.128	.564	-.358	.393
면적	.410	-.282	.510	-.171	-.040
아파트	.185	-.088	-.071	0.915	.153
인구	-.414	.313	.027	0.933	.007
경쟁 유통점	-.149	.205	.061	.175	.800

추출 방법: 주성분 분석, 회전 방법: Kaiser 정규화가 있는 배리맥스

KMO / Bartlett 구형성 검정 : 0.745 / 0.000

- **mfc에 영향을 미치는 요인에 따라 데이터프레임 재생성**

1요인: 경제 특성 [소비, 지방세, 1인당 자동차 등록대수, 종사자 수, 1인당 지방세 부담액, 소득, 단독·연립·다세대 주택]

2요인: 1인 가구 특징 [1인 가구 특성]

3요인: 임대료 특성 [유동인구, 임대료, 공시지가, 면적 변수]

4요인: 인구 특성 [인구, 아파트]

5요인: 경쟁특성 [경쟁 유통점]

PCA

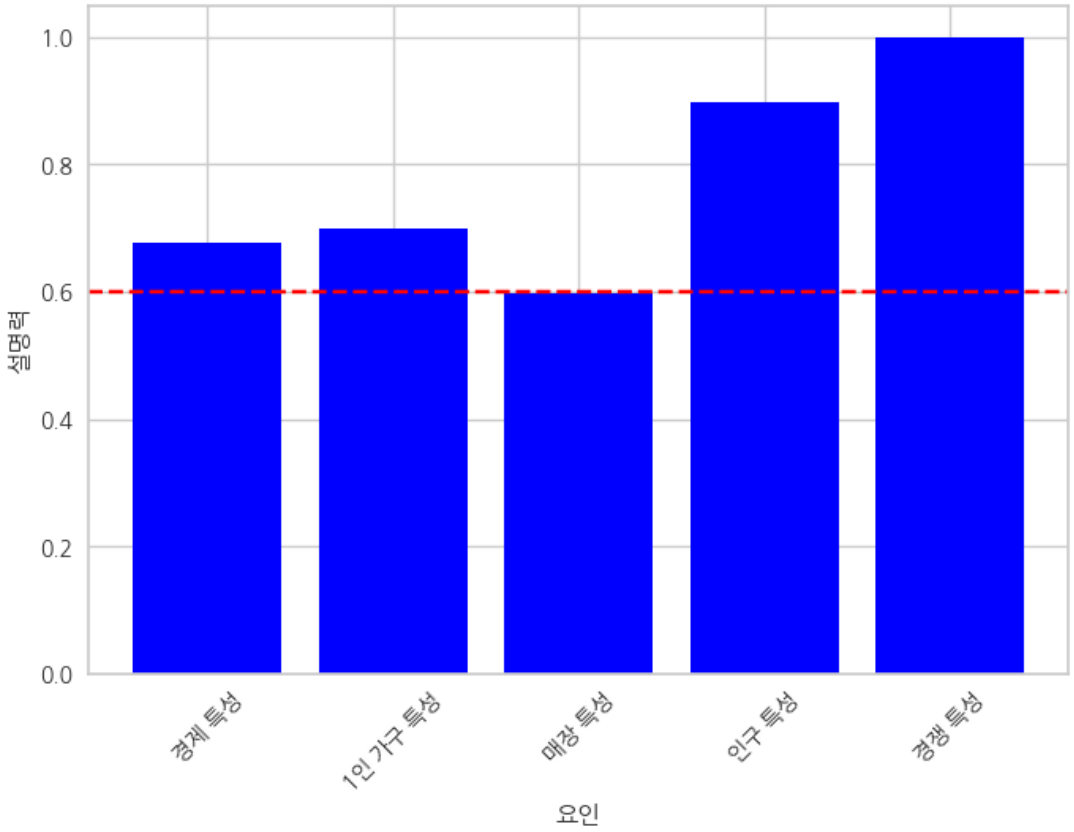
- 요인에 따라 데이터 프레임 재생성

```
economy_factor = ['소득', '종사자수', '단독, 연립, 다세대', '1인당 자동차 등록대수', '1인당 지방세', '지방세']
_1households_factor = ['1인가구', '비주거용', '1인가구 (20-39세)']
area_factor = ['소재지면적', '공시지가', '유동인구']
pop_factor = ['인구수', '아파트']
compet_factor = ['유통점합계']

economy_df = scaled_df[economy_factor]
_1households_df = scaled_df[_1households_factor]
area_df = scaled_df[area_factor]
pop_df = scaled_df[pop_factor]
compet_df = scaled_df[compet_factor]
df_list = [economy_df, _1households_df, area_df, pop_df, compet_df]
```

- **각 하나의 열로 PCA를 진행 후 병합**

```
transformed_data = []
pca = PCA(n_components = 1)

# Fit the PCA model to the data
for i in range(len(df_list)):
    pca.fit(df_list[i])
    data = pca.transform(df_list[i])
    transformed_data.append(data)
    print(pca.explained_variance_ratio_)

# [0.67780881]
# [0.69896507]
# [0.5987327]
# [0.89880173]
# [1.]
```

- **PCA 후 각 열에 대한 설명력 시각화**



5개 요인 모두 0.6보다 크므로 유의하다고 판단하였다.

```
pc_df = []
for i in range(len(transformed_data)):
    pca_df = pd.DataFrame(transformed_data[i])
    pc_df.append(pca_df)
gas_station_pca = pd.concat(pc_df, axis=1)

gas_station_pca.columns = ['경제 특성', '1인 가구 특성', '매장 특성', '인구 특성', '경쟁 특성']
gas_station_pca
```

	경제 특성	1인 가구 특성	매장 특성	인구 특성	경쟁 특성
0	1.148583	-0.302858	0.627049	0.266746	-0.257568
1	1.034417	-0.178335	0.659879	-0.178322	-0.046567
2	1.081856	0.065107	0.645740	-0.036279	-0.005990
3	1.127375	-0.076695	0.672049	-0.157018	-0.014406
4	1.348082	-0.123041	0.691052	0.000704	-0.080832

군집화

군집화 방법은 K-means 방법을 선택했다.

- **실루엣 계수**

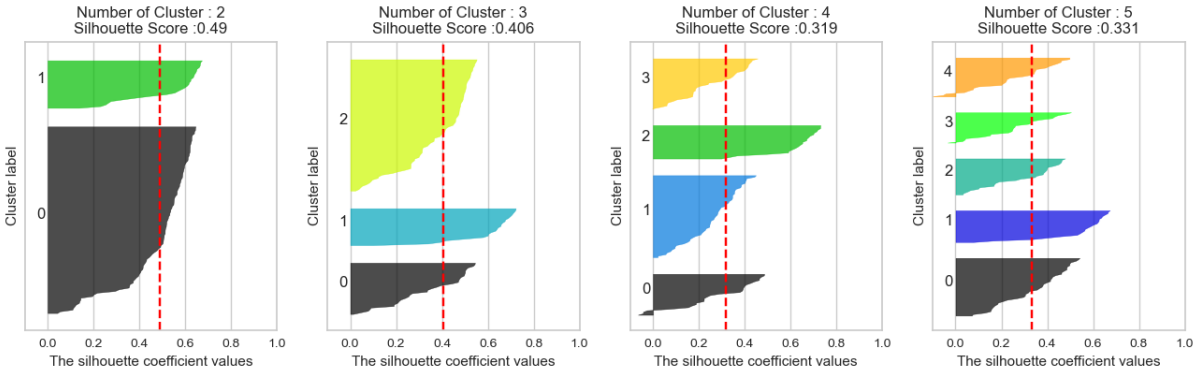
$$s(i) = \frac{(b(i) - a(i))}{(\max(a(i), b(i))}$$

- 각 군집 간의 거리가 얼마나 **효율적으로 분리**되어 있는지
- 실루엣 분석은 개별 데이터가 가지는 군집화 지표인 실루엣 계수를 기반으로 한다.
- 실루엣 계수는 해당 데이터가 **같은 군집 내 데이터와 얼마나 가깝게** 군집화 되어있는지 **다른 군집과는 얼마나 멀리** 분리되어 있는지 나타내는 지표

- 일반적으로 0.3정도가 넘으면 클러스터링이 되었다고 판단.

- **실루엣 계수 확인을 통해 군집의 수 결정**

```
import numpy as np
# make_blobs 을 통해 clustering 을 위한 4개의 클러스터 중심의 500개 2차원 데이터 셋 생성
from sklearn.datasets import make_blobs
X = np.array(gas_station_[['경제 특성', '1인 가구 특성', '매장 특성', '인구 특성', '경쟁 특성']])
# cluster 개수를 2개, 3개, 4개, 5개 일때의 클러스터별 실루엣 계수 평균값을 시각화
visualize_silhouette([ 2, 3, 4, 5], X)
```



k = 5일 때, 실루엣 계수도 상대적으로 높고, 군집 간 크기가 균등하므로 군집의 개수를 5개로 설정해줬다.

- **K=5로 군집화 진행**

```
k = 5

# 그룹 수, random_state 설정
model = KMeans(n_clusters = k, random_state = 10)

# 정규화된 데이터에 학습
model.fit(gas_station_pca)

# 클러스터링 결과 각 데이터가 몇 번째 그룹에 속하는지 저장
gas_station_pca['label'] = model.fit_predict(gas_station_pca)

gas_station_pca['label'].value_counts()

# 1    82
# 2    33
# 0    25
```

- **좌표 정보를 이용해서 지도에 시각화**

```
df_name_location = pd.read_csv('C:/Users/KJY/Desktop/학술제/완전 필터링 된 주유소 데이터.csv')
df_name_location = df_name_location[['사업장명', '좌표정보(X)', '좌표정보(Y)']]
df_name_location.head()

# 하나의 데이터프레임 생성
gas_station_= pd.concat([gas_station_pca, df_name_location], axis=1)
gas_station_ = gas_station_[['사업장명', '경제 특성', '1인 가구 특성', '매장 특성',
                             '인구 특성', '경쟁 특성', 'label', '좌표정보(X)', '좌표정보(Y)' ]]
```

- **시각화를 위한 좌표계 변경**

```
def project_array(coord, p1_type, p2_type):
    """
    좌표계 변환 함수
    - coord: x, y 좌표 정보가 담긴 NumPy Array
    - p1_type: 입력 좌표계 정보 ex) epsg:5179
    - p2_type: 출력 좌표계 정보 ex) epsg:4326
    """
    p1 = pyproj.Proj(init=p1_type)
    p2 = pyproj.Proj(init=p2_type)
    fx, fy = pyproj.transform(p1, p2, coord[:, 0], coord[:, 1])
    return np.dstack([fx, fy])[0]

coord = np.array(gas_station_ [['좌표정보(X)', '좌표정보(Y)']])
coord[:, 0]

p1_type = "epsg:5174"
p2_type = "epsg:4326"

# project_array() 함수 실행
result = project_array(coord, p1_type, p2_type)
result

gas_station_ ['위도'] = result[:, 1]
gas_station_ ['경도'] = result[:, 0]


gas_station_.head()
```

	사업장명	경제 특성	1인 가구 특성	매장 특성	인구 특성	경쟁 특성	label	좌표정보(X)	좌표정보(Y)	위도	경도
0	대양산업(주)개포주유소	1.148583	-0.302858	0.627049	0.266746	-0.257568	2	204487.0267	441119.1570	37.472241	127.051513
1	지에스칼텍스(주)학여울주유소	1.034417	-0.178335	0.659879	-0.178322	-0.046567	2	206780.9577	444026.0854	37.498419	127.077472
2	㈜소모 벤트럴주유소	1.081856	0.065107	0.645740	-0.036279	-0.005990	2	205159.7593	444358.1611	37.501421	127.059141
3	은마주유소	1.127375	-0.076695	0.672049	-0.157018	-0.014406	2	205879.9922	444208.9362	37.500073	127.067285
4	에이치디현대오일뱅크(주)직영 유진주유소	1.348082	-0.123041	0.691052	0.000704	-0.080832	2	203697.4855	442359.3130	37.483419	127.042594

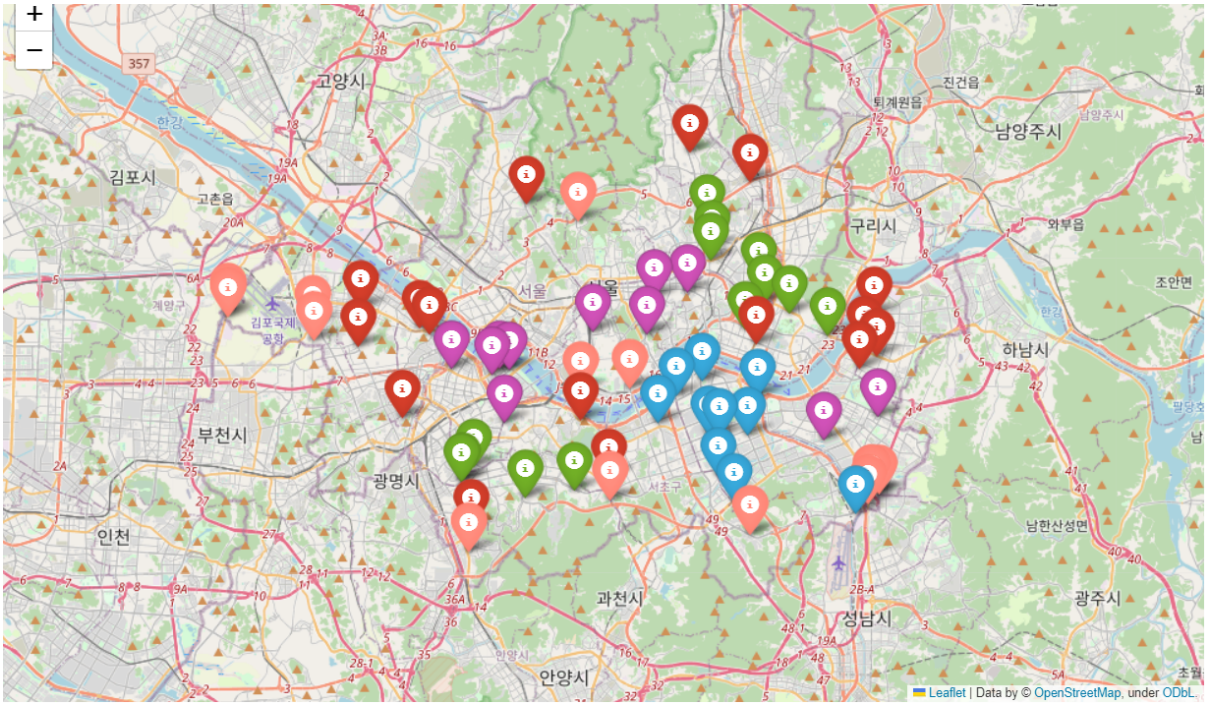
- **스코어링에서 한 번 이라도 뽑힌 애들만 필터링**

```
score = pd.read_csv('C:/Users/KJY/Desktop/학술제/ScoringResult.csv')
gas_station_score = gas_station_.iloc[list(score['item']),:]
```

	index	사업장명	경제 특성	1인 가구 특성	매장 특성	인구 특성	경쟁 특성	label	좌표정보(X)	좌표정보(Y)	위도	경도	count
0	136	서울석유(주) 장충주유소	0.502132	0.388659	-0.034924	-0.111870	0.430439	3	200551.8661	451166.7415	37.562780	127.007034	49
1	68	삼미상사(주) 장안킹셀프주유소	-0.233897	0.597863	-0.171920	-0.130616	0.477328	2	205664.3244	451975.0360	37.570046	127.064905	46
2	81	남선석유(주)현릉로주유소	0.613554	-0.485562	0.311042	0.737769	-0.409657	4	205242.6809	439473.9313	37.457414	127.060044	45
3	135	세화주유소	0.300746	0.515280	-0.049920	-0.219687	0.555477	3	202185.7257	451383.5571	37.564731	127.025527	41
4	9	지에스칼텍스(주) 삼성로주유소	1.219969	0.114218	0.672821	-0.131016	0.095603	1	203184.1089	444431.8739	37.502094	127.036798	40

- **시각화**

```
# 해당위치를 시작으로 지도 시각화
center = [37.5, 127]
m_cluster = folium.Map(location=center, zoom_start=10)
m_cluster

# 마커의 색 지정
col_lst = ['red', 'blue', 'green', 'purple', 'lightred']

# 지도에 시각화
for i in range(len(gas_station_score)):
    latitude = gas_station_score.loc[i, '위도']
    longitude = gas_station_score.loc[i, '경도']

    # 클러스터 번호에 따라 색상을 선택 (나머지 연산을 사용)
    cluster_idx = gas_station_score['label'][i]
    color_idx = cluster_idx % len(col_lst)
    color = col_lst[color_idx]

    marker = Marker([latitude, longitude], popup='Center Marker', icon=folium.Icon(color=color))


    marker.add_to(m_score)
```



11월 6일에 한 거

거리, 미처리 수요지에 대한 시각화

- **사용 라이브러리**

```
from haversine import haversine
import openpyxl
import pandas as pd
import numpy as np
import pyproj
import folium
from folium import Circle, Marker, PolyLine
```

- **사용한 데이터**

```
# 행정동 센터 데이터
welfare = pd.read_csv('C:/Users/KJY/Desktop/학술제/행정동센터.csv')

# 위에서 구한 최종 선정된 주유소 데이터
gas_station = pd.read_csv('C:/Users/KJY/실습 폴더 모음/최종 선정된 주유소.csv', encoding='euc-kr')
```

- **좌표계 변환**

```
coord = np.array(gas_station[['좌표정보(X)', '좌표정보(Y)']])
```

```
def project_array(coord, p1_type, p2_type):
    """
    좌표계 변환 함수
    - coord: x, y 좌표 정보가 담긴 NumPy Array
    - p1_type: 입력 좌표계 정보 ex) epsg:5179
    - p2_type: 출력 좌표계 정보 ex) epsg:4326
    """
    p1 = pyproj.Proj(init=p1_type)
    p2 = pyproj.Proj(init=p2_type)
    fx, fy = pyproj.transform(p1, p2, coord[:, 0], coord[:, 1])
    return np.dstack([fx, fy])[0]

p1_type = "epsg:5174"
p2_type = "epsg:4326"

# project_array() 함수 실행
result = project_array(coord, p1_type, p2_type)
result

gas_station['위도'] = result[:, 1]
gas_station['경도'] = result[:, 0]
```

- **주유소와 행정동센터 직선거리 구하기**

```
distance_compare_list = []

for i in range(gas_station.shape[0]):
    for j in range(welfare.shape[0]):
        gas= (gas_station['위도'][i], gas_station['경도'][i])
        dong = (welfare['위도'][j], welfare['경도'][j])
        result = haversine(gas, dong, unit='m')
        distance_compare_list.append(result)
```

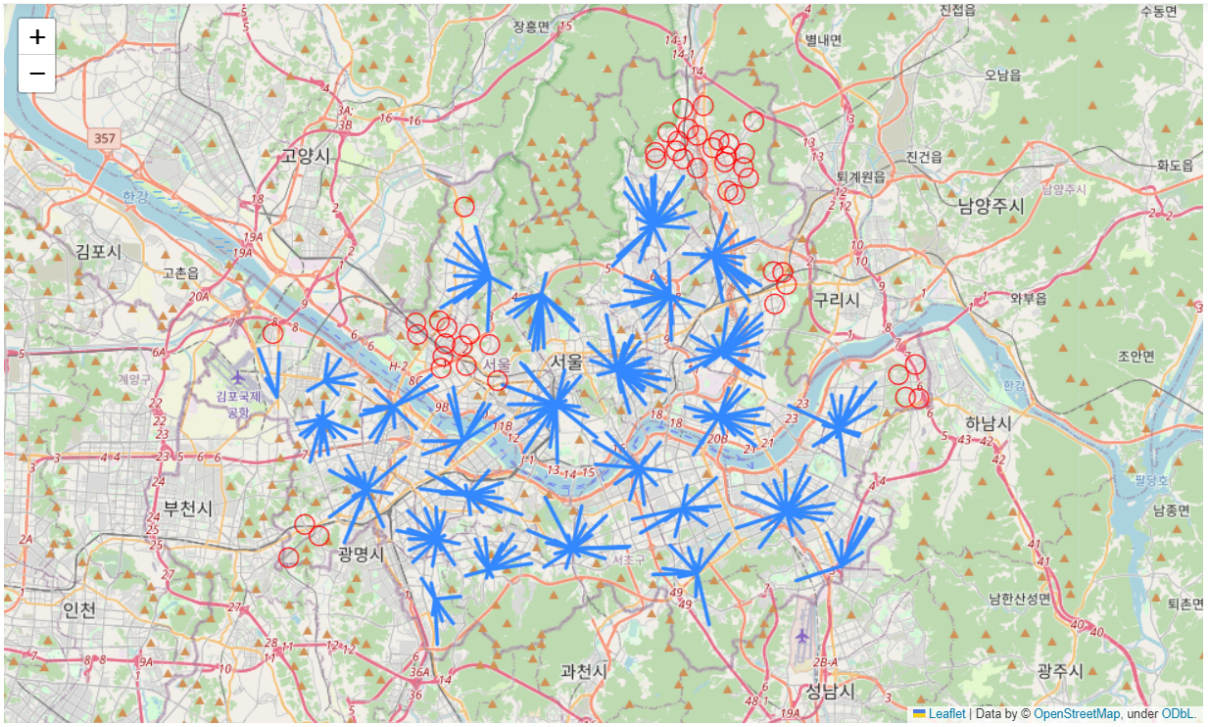
- **주유소를 행으로, 행정동센터를 열로 직선거리행렬 생성**

```
distance_matrix = pd.DataFrame(np.array(np.array(distance_compare_list).reshape(len(gas_station), len(welfare))))
distance_matrix.to_csv('주유소, 복지센터 거리.csv', encoding='euc-kr')
```

- **최적화 후 지도 시각화**

```
center = [37.5, 127]
m2 = folium.Map(location=center, zoom_start=10)

# 3km 기준
df1 = pd.read_csv('C:/Users/KJY/Desktop/학술제/OPT100.csv', encoding='euc-kr'); df1.head()

# 미처리 수요지
non_demand = pd.read_csv('C:/Users/KJY/Desktop/학술제/미처리.csv', encoding = 'euc-kr')

# 리스트화 시켜서 출발노드, 도착노드를 모두 연결
start1 = gas_station.loc[df1['출발']][['위도', '경도']]
end1 = welfare.loc[df1['도착']][['위도', '경도']]

start_list1 = np.array(start1).tolist()
end_list1 = np.array(end1).tolist()

for i in range(len(start_list1)):
```

```
    lines1 = [start_list1[i], end_list1[i]]
    folium.PolyLine(locations = lines1, tooltip = 'PolyLine').add_to(m2)

# 미처리 수요지 원으로 시각화
non_loc = welfare.loc[non_demand['미처리']][['위도', '경도']]
non_list = np.array(non_loc).tolist()

for i in range(len(non_list)):
    circle = Circle(
        location=(non_list[i][0], non_list[i][1]),
        radius=500,  # 3 km in meters
        color='red',
        fill=True,
        fill_opacity=0.01,
        weight = 1)
    circle.add_to(m2)
```



데이터 출처

소득	**스마트치안 빅데이터 플랫폼**
편의점 수	**서울 열린데이터 광장 서울시 생계형사업 분포현황**
슈퍼마켓	**서울 열린데이터 광장 서울시 생계형사업 분포현황**
식료품점	**서울시 생계형사업 분포현황**

데이터 변수의 단위

1인 가구	가구
20-39세 1인 가구	가구
유동인구	명
인구수	명
종사자수	명
유통점합계	개
아파트	호
단독, 연립, 다세대	호
비주거용	호
자동차 등록대수	대
소득	원
소재지면적	제곱미터
공시지가	만원/제곱미터
지방세	만원
1인당 지방세	만원