

다함께 차찾자!

클러스터링을 통한 효율적인 쏘카존 관리 모델 제안

10팀

김정주

전은영

정지원

홍현도

목 차

- 1. 프로젝트 진행 배경 및 목적
- 2. 프로젝트 프로세스(일정, 순서) + 팀원 역할
- 3. 프로젝트 단계별 내용(코드, 이미지)
- 4. 프로젝트 결과 *분석
- 5. 개발 환경 + 참고 문헌

쏘카존을 ‘효율적으로 관리’하려면?

→ 쏘카에 대한 ‘사람들의 수요’를 정확히 파악할 필요가 있다!

쏘카존에 대한 ‘수요를 파악’하려면?

→ 어떤 사람들이 어떤 목적으로 쏘카를 사용하는지 파악할 필요가 있다!

사람들이 다양한 명소(수도권)를 방문할 때 쏘카를 주로 사용할 것이다.

→ But, ‘명소까지 가는 대중교통이 불편하거나, 자차가 없거나... 등의 이유 존재!’

명소를 방문하는 정도에 대해서 ‘Naver 검색량 + 대중교통의 편의성’을 통한 수요 예측

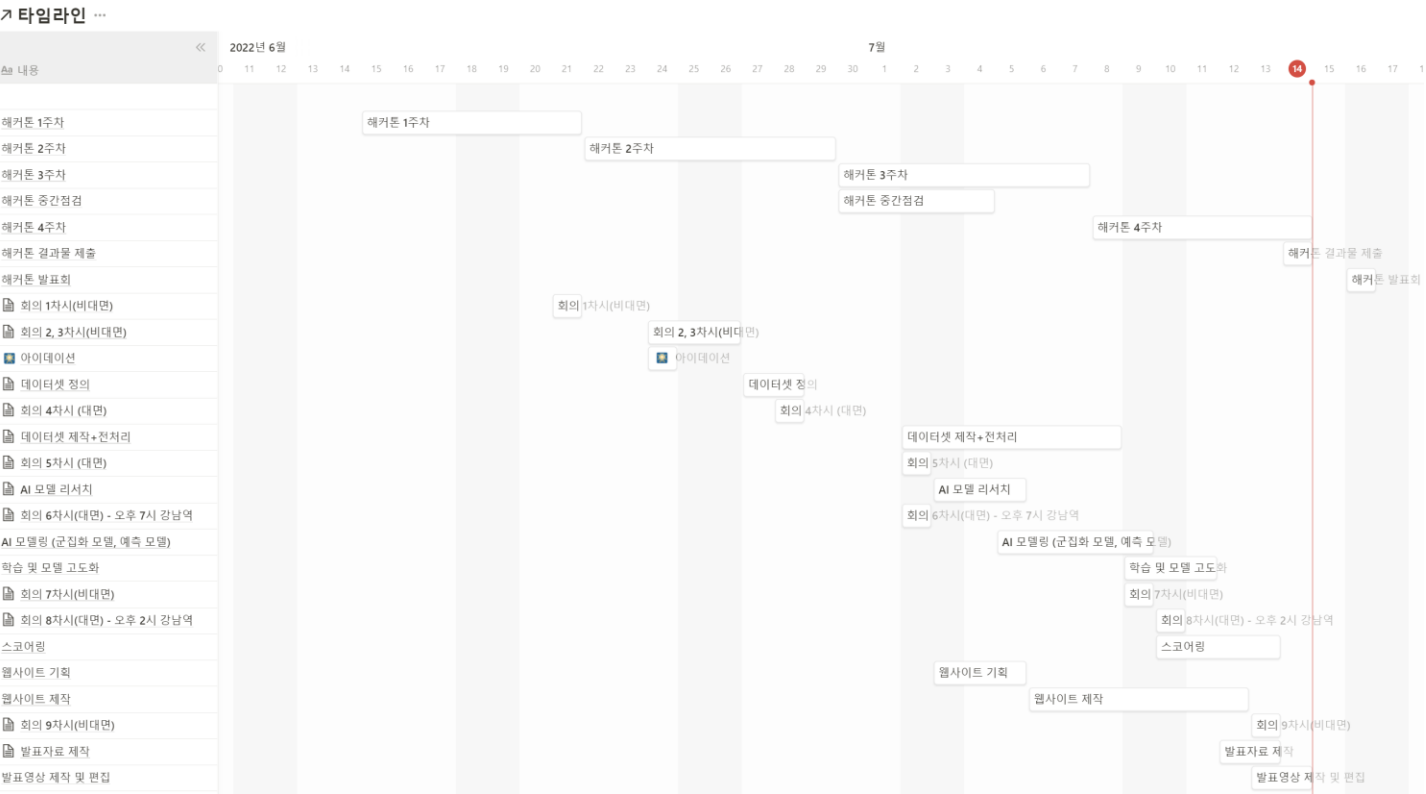
서울에 한정하여 ‘지역별로 인구의 특성(성별, 연령대, 가구수 등)을 통한 클러스터링’

→ 어떤 사람들이 어떤 명소를 주로 방문하는지 파악!

명소를 방문하는 정도에 대해서 Naver 검색량 + 대중교통의 편의성을 통한 수요 예측

지역별 인구 클러스터에 대응하는 쏘카존에 대해 ‘예측한 수요와 실제 사용량과의 차이를 파악’

→ 수요가 많은 곳에 쏘카가 부족하다면, 쏘카를 더 추가해주어야 하는 판단을 내릴 수 있다!



_ 프로세스 진행 일정, 순서

“ 프로젝트 프로세스 _ 일정 및 순서 ”

- 1) 프로젝트 진행 방향 설정
- 2) 활용 오픈데이터셋 선정
- 3) R&R 설정
- 4) 활용 AI모델 리서치 및 선정 관련 논의
- 5) 데이터셋 구축
- 6) AI 모델링/학습/고도화 - 군집화(Clustering)
- 7) 수요 예측 (Scoring)
- 8) 웹 페이지 제작
- 9) 발표자료 및 영상 제작

“ 팀원 역할 ”

- 김 정 주 : 데이터셋 구축 및 전처리, 웹 개발, Scoring 예측
- 전 은 영 : 데이터셋 구축, 크롤링 자동화, Scoring 예측
- 정 지 원 (PM) : 데이터셋 구축 및 전처리, 클러스터링, Scoring 예측
- 홍 현 도 : 데이터셋 구축, Scoring 예측, 발표자료 제작

“ 클러스터링을 통한 효율적인 쏘카존 관리 모델 제안 ”

* 출처 : 팀 내 공유 노션 페이지

3. 단계별 내용 (1)-1 : 데이터셋 구축 ‘ 동별 데이터셋 ’

1. 진행 배경 및 목적 | 2. 프로세스(일정, 순서) + 팀원 역할 | 3. 단계별 내용 (코드, 이미지) | 4. 프로젝트 결과 및 분석 | 5. 개발 환경 + 참고 문헌

평균나이	세대원수 인구수			가구 구성별 인구수						인구수						면허소지자								등록된 차량수	주차장 보급률
	1명	2명	3명+	1인	2인	3인+	외국인	비혈연6인 이상	집단	15세미만	15~19세	20대	30대	40대	50대	60+대	15~19세	20대	30대	40대	50대	60대	70대	80+대	

위치	평균 나이	세대원 1명	세대원 2명	세대원3명 이상	1인가구	2인가구	3인 이상 가구	외국인 인구	비혈연6인 이상가구 인구	15세 미만	15~19세	20대	30대	40대	50대	60대	등록된 차량수	주차장 보급률	면허소지 15~19세	면허소지 20대	면허소지 30대
개포1동	46.8	758.0	646.0	1312.0	550.0	654.0	1120.0	5.0	6.0	664.5	286.0	705.0	630.0	886.5	1006.5	2103.5	3599.0	167.9	2107.0	47107.0	58045.0
개포2동	40.9	1648.0	2199.0	5932.0	1058.0	2081.0	4288.0	59.0	0.0	3424.0	1594.0	3201.0	2968.0	4219.5	4257.0	5004.5	9948.0	167.9	11743.0	213886.0	273458.0
개포4동	42.1	3541.0	2275.0	4221.0	2280.0	2095.0	3423.0	165.0	46.0	2699.0	1107.5	3167.5	3516.5	3771.0	3791.0	4695.5	8563.0	167.9	8159.0	211647.0	323995.0
논현1동	42.7	9526.0	2381.0	2088.0	7121.0	2983.0	2173.0	716.0	76.0	1046.5	481.0	4422.5	5500.5	3596.0	2662.5	4044.0	10529.0	167.9	3544.0	295505.0	506792.0
논현2동	43.0	6474.0	2093.0	2846.0	4618.0	2608.0	2853.0	318.0	31.0	1657.5	691.0	3435.5	4300.0	3622.5	2834.0	4132.0	10295.0	167.9	5091.0	229555.0	396183.0
대치1동	38.6	602.0	1058.0	5596.0	350.0	1147.0	5489.0	38.0	0.0	4707.0	2598.0	2409.0	1461.0	5846.0	3874.5	3381.5	14360.0	167.9	19140.0	160966.0	134610.0
대치2동	41.4	2622.0	2493.0	7793.0	1807.0	2567.0	7598.0	169.0	25.0	4935.0	3281.5	54669.5	3194.0	7010.0	6528.0	7007.0	15069.0	167.9	24175.0	312009.0	294281.0
대치4동	40.7	5594.0	1689.0	2880.0	4269.0	2117.0	2851.0	318.0	56.0	1637.5	1559.0	3343.5	3228.0	3928.5	3141.0	2908.0	10306.0	167.9	11485.0	223407.0	297414.0
도곡1동	41.5	2738.0	1663.0	4138.0	1872.0	1871.0	4039.0	141.0	34.0	2980.0	1089.5	2850.5	3077.0	3768.5	3408.0	4269.0	8974.0	167.9	8026.0	190466.0	283501.0
도곡2동	41.4	2221.0	2455.0	6972.0	1408.0	2676.0	6689.0	171.0	19.0	5229.5	2344.5	3852.5	3376.0	6325.5	5416.0	6488.0	13615.0	167.9	17272.0	257418.0	311050.0
삼성1동	43.5	2044.0	1228.0	2217.0	1433.0	1380.0	2162.0	217.0	19.0	1383.5	684.0	1793.5	1861.0	2065.5	2078.0	2933.5	8378.0	167.9	5039.0	119839.0	171464.0
삼성2동	40.3	5828.0	2480.0	5431.0	4236.0	2765.0	4868.0	293.0	0.0	4175.0	1692.0	3857.5	4792.0	6240.5	4217.0	4704.5	12080.0	167.9	12465.0	257752.0	441514.0
세곡동	41.7	6272.0	4097.0	7559.0	4891.0	4233.0	7615.0	128.0	0.0	6795.0	1995.5	5246.0	6491.5	7751.5	5984.0	9230.0	18782.0	167.9	14701.0	350530.0	598098.0
수서동	51.8	3664.0	2286.0	1820.0	3083.0	2453.0	1841.0	71.0	0.0	852.5	326.5	1496.5	2188.5	1616.5	2139.5	6187.5	5261.0	167.9	2405.0	99994.0	201639.0
신사동	44.4	2732.0	1593.0	2685.0	2047.0	1744.0	2616.0	291.0	15.0	1683.5	871.0	2142.5	2229.0	2765.0	2421.0	3771.5	7876.0	167.9	6417.0	143159.0	205370.0
압구정동	44.3	2787.0	2485.0	4962.0	1822.0	2676.0	4704.0	208.0	44.0	3508.0	1320.5	3023.5	3129.5	4587.0	4008.0	6448.5	12272.0	167.9	9728.0	202026.0	288338.0
역삼1동	40.9	17842.0	3496.0	2899.0	12735.0	4551.0	3070.0	935.0	107.0	1561.0	831.0	7409.5	9839.0	5760.0	3950.0	5140.0	21372.0	167.9	6122.0	495091.0	906522.0
역삼2동	39.0	6635.0	2254.0	6651.0	4735.0	2712.0	6399.0	360.0	47.0	5657.0	2567.0	4403.5	4923.0	8393.0	4867.0	4784.5	15305.0	167.9	18911.0	294235.0	453583.0
일원1동	46.5	3451.0	1827.0	2188.0	2631.0	1839.0	2319.0	147.0	26.0	1207.0	646.0	2124.5	2189.5	2067.0	2367.0	4535.0	4641.0	167.9	4759.0	141956.0	201731.0
일원2동	44.4	2480.0	1870.0	3137.0	1927.0	1899.0	3046.0	66.0	0.0	2080.5	868.0	1844.0	2496.5	3022.0	2546.0	4528.5	6220.0	167.9	6395.0	123213.0	230016.0
일원본동	40.8	1517.0	1677.0	4861.0	923.0	1765.0	4798.0	29.0	0.0	3934.0	1513.5	2525.5	2829.5	4434.5	3683.5	4495.5	8384.0	167.9	11150.0	168750.0	260698.0
청담동	42.2	4477.0	2318.0	4408.0	3203.0	2715.0	4336.0	264.0	13.0	3182.5	1118.5	3606.0	4129.5	4571.0	3822.0	5185.0	11850.0	167.9	8240.0	240947.0	380474.0
강일동	42.1	3377.0	3469.0	6043.0	2081.0	3046.0	6150.0	90.0	0.0	4646.0	1557.0	3334.5	4003.0	4975.0	4377.5	7002.5	12836.0	123.6	11470.0	222806.0	368819.0
도곡1동	39.5	1470.0	1569.0	5176.0	1143.0	1701.0	5508.0	120.0	0.0	5008.0	1370.5	2308.5	3169.5	5340.5	3357.0	4330.5	8722.0	123.6	10097.0	154250.0	292024.0

“ 서울시 동별 데이터셋 제작 _ 오픈 데이터셋 활용 ”

- #1 인구 구성, 차량 관련 특성을 나타낼 수 있는 오픈데이터셋 선정 및 수집
- #2 데이터 전처리

1) 수집 시기가 달라 자치구/동이 다른 경우
→ 가장 오래된 데이터(2020년) 기준으로 자치구/동 목록 통일.

2) 이상치 / 결측치 처리
→ 너무 값이 작은 경우 다른 동의 데이터로 합계 처리.
→ 빈 값은 ‘0’으로 기입.
- #3 경제인구, 운전면허 소지인구 등 기존 데이터를 활용해 원하는 데이터 생성.

- #3 총 117종류의 데이터 중 지역 특성을 대표할 수 있다고 판단되는 27개의 값을 선정하여 클러스터링에 활용.
→ “ 차원의 저주 문제 방지 차원 ”

“ 지역정보 ”
데이터 구성

구 / 동 / 평균 연령 / 연령별 인구 / 세대원수별 인구 / 가구원수별 인구 / 외국인 인구 / 비혈연6인 이상 가구수 / 집단시설 가구수 / 등록된 차량수(연료별) / 주차장 확보율(구별) / 면허소지 인구수(연령별) / ...

3. 단계별 내용 (1)-2 : 데이터셋 구축 ‘ 명소 데이터셋 ’

1. 진행 배경 및 목적 | 2. 프로세스(일정, 순서) + 팀원 역할 | 3. 단계별 내용 (코드, 이미지) | 4. 프로젝트 결과 및 분석 | 5. 개발 환경 + 참고 문헌

“ 서울/경기지역 명소 data set 활용 ”

“ 키워드에 따른 명소 구분 ”

“ 위치 좌표(위도, 경도) 확인 ”

No.	명소 ID	키워드	명칭	시/도	시/군/구	동/면/읍/리	전체 주소	명소_위도	명소_경도
1	1	관광 명소	63스퀘어	서울시	영등포구	여의도동	서울특별시 영등포구 여의도동 60	37.51957	126.9398
2	2	관광 명소	몽촌역사관	서울시	송파구	방이동	서울특별시 송파구 방이동 88-3	37.52442	127.1232
3	3	관광 명소	경복궁	서울시	종로구	세종로	서울특별시 종로구 세종로 1-1	37.57764	126.9780
4	5	관광 명소	광화문광장	서울시	종로구	세종로	서울특별시 종로구 세종로 1-68	37.57243	126.9769
5	6	관광 명소	국립고궁박물관	서울시	종로구	세종로	서울특별시 종로구 세종로 1-57	37.57706	126.9764
6	7	관광 명소	국립기상박물관	서울시	종로구	송월동	서울특별시 종로구 송월동 1-1	37.57136	126.9659
7	8	관광 명소	국립중앙박물관	서울시	용산구	용산동6가	서울특별시 용산구 용산동6가 168-6	37.52304	126.9822
8	9	관광 명소	국립현대미술관	서울시	종로구	소격동	서울특별시 종로구 소격동 165	37.57951	126.9806
9	10	관광 명소	국회의사당	서울시	영등포구	여의도동	서울특별시 영등포구 여의도동 1	37.53097	126.9165
10	11	관광 명소	남산서울타워	서울시	용산구	용산동2가	서울특별시 용산구 용산동2가 산 1-3	37.55112	126.9879
11	12	관광 명소	대한민국역사박물관	서울시	종로구	세종로	서울특별시 종로구 세종로 82-1	37.57392	126.9779
12	13	관광 명소	덕수궁	서울시	중구	정동	서울특별시 중구 정동 5-1	37.56594	126.9750
13	14	관광 명소	도산공원	서울시	강남구	신사동	서울특별시 강남구 신사동 649-9	37.52439	127.0351
14	16	관광 명소	롯데월드 민속박물관	서울시	송파구	잠실동	서울특별시 송파구 잠실동 40-1	37.51125	127.0970
15	17	관광 명소	롯데월드타워	서울시	송파구	신천동	서울특별시 송파구 신천동 29	37.51292	127.1027
16	18	관광 명소	리움 미술관	서울시	용산구	한남동	서울특별시 용산구 한남동 742-1	37.53837	126.9993
17	20	관광 명소	명동성당	서울시	중구	명동2가	서울특별시 중구 명동2가 1-1	37.56331	126.9868
18	21	관광 명소	문화역서울 284	서울시	중구	봉래동2가	서울특별시 중구 봉래동2가 122-28	37.55596	126.9716

* 출처 : 팀 내 공유 구글 스프레드 시트 (공공데이터 포털 / 서울 열린데이터 광장 / 카카오트렌드)

3. 단계별 내용 (1)-2 : 데이터셋 구축 ‘ 명소 데이터셋 ’

1. 진행 배경 및 목적 | 2. 프로세스(일정, 순서) + 팀원 역할 | 3. 단계별 내용 (코드, 이미지) | 4. 프로젝트 결과 및 분석 | 5. 개발 환경 + 참고 문헌

- # Scoring 기준 1 : "Dataset Crawling을 통한 수요 예측"
- 1) ‘성별’에 따른 명소 검색을
- 2) ‘연령대’에 따른 명소 검색을

No.	명소 ID	키워드	명칭	여성_검색율	남성_검색율	여_10대_검색량	여_20대_검색량	여_30대_검색량	여_40대_검색량	여_50대_검색량	여_60대_검색량	남_10대_검색량	남_20대_검색량	남_30대_검색량
1	1	관광 명소	63스퀘어	52	48	12	82	315	432	304	35	0	55	
2	2	관광 명소	몽촌역사관	58	42	0	4	30	223	63	52	0	3	
3	3	관광 명소	경복궁	54	46	1102	2204	9917	46281	34160	17631	0	939	
4	5	관광 명소	광화문광장	31	69	25	75	224	871	846	448	111	388	
5	6	관광 명소	국립고궁박물관	62	38	0	398	1727	5047	3719	2258	80	80	
6	7	관광 명소	국립기상박물관	54	46	0	14	128	825	370	71	0	0	
7	8	관광 명소	국립중앙박물관	62	38	0	3501	21003	73511	47257	29754	0	1073	
8	9	관광 명소	국립현대미술관	73	27	0	11	65	262	404	338	0	4	
9	10	관광 명소	국회의사당	42	58	0	173	864	3630	2680	1124	0	234	
10	11	관광 명소	남산서울타워	50	50	0	82	165	692	412	280	0	16	
11	12	관광 명소	대한민국역사박물관	55	45	0	176	1053	4564	2019	878	0	71	
12	13	관광 명소	덕수궁	61	39	0	1197	3590	14362	12766	7979	0	515	
13	14	관광 명소	도산공원	56	44	0	675	1575	3937	3487	1687	88	177	
14	16	관광 명소	롯데월드 민속박물관	62	38	50	100	702	2859	953	351	0	0	
15	17	관광 명소	롯데월드타워	42	58	0	396	2377	8915	5349	2576	0	271	
16	18	관광 명소	리움 미술관	72	28	0	1673	5578	17292	20081	10599	0	217	
17	20	관광 명소	명동성당	67	33	0	197	1380	6899	6110	5125	0	291	
18	21	관광 명소	문화역서울 284	61	39	0	200	651	1853	1453	801	0	63	

* 출처 : 팀 내 공유 구글 스프레드 시트 (공공데이터 포털 / 서울 열린데이터 광장 / 카카오톡렌드)

3. 단계별 내용 (1)-2 : 데이터셋 구축 ‘ 명소 데이터셋 ’

1. 진행 배경 및 목적 | 2. 프로세스(일정, 순서) + 팀원 역할 | 3. 단계별 내용 (코드, 이미지) | 4. 프로젝트 결과 및 분석 | 5. 개발 환경 + 참고 문헌

Scoring 기준 2 : “ 대중교통 ↔ 명소 간의 거리 ”

1) 가장 가까운 지하철 역 ↔ 명소 간의 거리

2) 가장 가까운 버스정류소 ↔ 명소 간의 거리

				지하철역까지의 거리 계산(이후 병합)				버스정류장까지의 거리 계산(이후 병합)		
No.	명소 ID	키워드	명칭	지하철역	지하철역_위도	지하철역_경도	지하철_거리	버스정류장	버스정류장_위도	버스정류장_경도
1	1	관광 명소	63스퀘어	노량진	37.514219	126.942454	0.6393917898	63빌딩.가톨릭대학교여의도성모병원	37.51916181	126.9377093
2	2	관광 명소	몽촌역사관	강동구청	37.530341	127.120508	0.6998768927	현대토파즈아파트	37.52530533	127.1258611
3	3	관광 명소	경복궁	경복궁	37.575762	126.97353	0.4458521562	경복궁	37.57695016	126.9793378
4	5	관광 명소	광화문광장	광화문	37.571026	126.976669	0.1574397226	KT광화문지사	37.57222644	126.9773321
5	6	관광 명소	국립고궁박물관	경복궁	37.575762	126.97353	0.2912074302	국립고궁박물관.경복궁서문	37.57677167	126.9743664
6	7	관광 명소	국립기상박물관	서대문	37.565773	126.966641	0.6246702775	스위스대사관	37.56998879	126.9658058
7	8	관광 명소	국립중앙박물관	서빙고	37.519594	126.988537	0.67761555	국립중앙박물관용산가족공원	37.52082823	126.979799
8	9	관광 명소	국립현대미술관	안국	37.576477	126.985443	0.5439567133	정독도서관	37.58019153	126.9802007
9	10	관광 명소	국회의사당	국회의사당	37.528105	126.917874	0.3408366268	국회의사당	37.5298981	126.9180242
10	11	관광 명소	남산서울타워	명동	37.560989	126.986325	1.106132002	남산서울타워	37.55124467	126.9907563
11	12	관광 명소	대한민국역사박물관	광화문	37.571026	126.976669	0.3395943364	세종문화회관	37.57394965	126.9768673
12	13	관광 명소	덕수궁	시청	37.564718	126.977108	0.2301838314	덕수궁	37.56558455	126.9767268
13	14	관광 명소	도산공원	압구정로데오	37.527381	127.040534	0.5833093168	신구중학교	37.52564607	127.0335404
14	16	관광 명소	롯데월드 민속박물관	잠실	37.51395	127.102234	0.5506873992	롯데월드	37.51214677	127.0970648
15	17	관광 명소	롯데월드타워	잠실	37.51395	127.102234	0.1216827261	잠실역1번.11번출구	37.51415201	127.1030253
16	18	관광 명소	리움 미술관	한강진	37.539631	127.001725	0.2556900634	한남동새마을금고	37.53684427	127.0003214
17	20	관광 명소	명동성당	명동	37.560989	126.986325	0.2614578008	서울백병원.국가인권위.안중근활동터	37.56435047	126.9878515
18	21	관광 명소	문화역서울 284	서울	37.554648	126.972559	0.1686115756	서울역버스환승센터	37.55544621	126.9723557

* 출처 : 팀 내 공유 구글 스프레드 시트 (공공데이터 포털 / 서울 열린데이터 광장 / 카카오트렌드)

Scoring 기준 2 : “ 대중교통 ↔ 명소 간의 거리 ”

‘ 데이터 전처리 ① ’

; 명소 data set에서 거리 계산을 위한 좌표값만 남기고, 나머지 column들은 drop

Input Feature Optimization

```
[ ] # ① 명소 data set에서 좌표 위치만 남기고 나머지 column들은 drop
place_cord = df_place.drop(['키워드', '명칭', '시/도', '시/군/구', '동/면/읍/리', '전체 주소'],1)
place_cord.head()
```

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument

	명소 ID	명소_위도	명소_경도	지하철역	지하철역_위도	지하철역_경도	지하철역_거리	버스정류장	버스정류장_위도	버스정류장_경도	버스정류장_거리
0	1.0	37.51957	126.9398	여의나루역	37.527098	126.932901	NaN	NaN	NaN	NaN	NaN
1	2.0	37.52442	127.1232	올림픽공원역	37.516078	127.130848	NaN	NaN	NaN	NaN	NaN
2	3.0	37.57764	126.9780	경복궁역	37.575762	126.973530	NaN	NaN	NaN	NaN	NaN
3	4.0	37.59207	127.0555	외대앞역	37.596073	127.063549	NaN	NaN	NaN	NaN	NaN
4	5.0	37.57243	126.9769	경복궁역	37.575762	126.973530	NaN	NaN	NaN	NaN	NaN

3. 단계별 내용 (1)-2 : 데이터셋 구축 ‘명소 데이터셋’

다함께 차찾자!

클러스터링을 통한 효율적인 쓰카존 관리 모델 제안

1. 진행 배경 및 목적 | 2. 프로세스(일정, 순서) + 팀원 역할 | 3. 단계별 내용 (코드, 이미지) | 4. 프로젝트 결과 및 분석 | 5. 개발 환경 + 참고 문헌

Scoring 기준 2 : “대중교통 ↔ 명소 간의 거리”

‘데이터 전처리 ②’

; 명소 위치 ~ 지하철역 위치 거리 계산

“명소에 대한 DataFrame” 과 “지하철역에 대한 DataFrame” 활용.

1) 이 2개의 ‘for문’을 통해, 해당 명소와 가장 가까운 지하철역과의 거리를 계산.

2) 최단 거리이면 그 좌표값을 저장, 아니면 drop후 반복 수행

3) 모든 명소에 대해 반복 후 원본 명소 DataFrame에 저장.

```
[ ] # ② 명소 위치 ~ 지하철역 위치 중 가장 최솟값인 좌표 + 역명 + 거리(km) 계산/선택하여 append
station_name = ''
shortest_x = 0
shortest_y = 0
shortest_dist = 0
```

```
for a in range(max(place_cord['명소 ID'].astype(int))):
    for b in range(max((df_subway['역ID'].astype(int)))):
        place = (place_cord.iloc[a,1], place_cord.iloc[a,2])
        subway = (df_subway.iloc[b,2], df_subway.iloc[b,3])
        dist_temp = haversine(place, subway, unit='km')

        if b == 0:
            station_name = df_subway.iloc[b,1]
            shortest_x = df_subway.iloc[b,2]
            shortest_y = df_subway.iloc[b,3]
            shortest_dist = dist_temp
        else:
            subway_before = (shortest_x, shortest_y)
            if dist_temp < haversine(place, subway_before, unit='km'):
                station_name = df_subway.iloc[b,1]
                shortest_x = df_subway.iloc[b,2]
                shortest_y = df_subway.iloc[b,3]
                shortest_dist = dist_temp
```

```
place_cord.iloc[a,3] = station_name
place_cord.iloc[a,4] = shortest_x
place_cord.iloc[a,5] = shortest_y
place_cord.iloc[a,6] = shortest_dist
```

```
place_cord.head()
```

```
for a in range(max(place_cord['명소 ID'].astype(int))):
    for b in range(max((df_subway['역ID'].astype(int)))):
        place = (place_cord.iloc[a,1], place_cord.iloc[a,2])
        subway = (df_subway.iloc[b,2], df_subway.iloc[b,3])
        dist_temp = haversine(place, subway, unit='km')

        if b == 0:
            station_name = df_subway.iloc[b,1]
            shortest_x = df_subway.iloc[b,2]
            shortest_y = df_subway.iloc[b,3]
            shortest_dist = dist_temp
        else:
            subway_before = (shortest_x, shortest_y)
            if dist_temp < haversine(place, subway_before, unit='km'):
                station_name = df_subway.iloc[b,1]
                shortest_x = df_subway.iloc[b,2]
                shortest_y = df_subway.iloc[b,3]
                shortest_dist = dist_temp
```

Scoring 기준 2 : “대중교통 ↔ 명소 간의 거리”

‘데이터 전처리 ③’

; 명소 위치 ~ 버스정류장 위치 거리 계산

```
[ ] # ② 명소 위치 ~ 지하철역 위치 중 가장 최솟값인 좌표 + 역명 + 거리(km) 계산/선택하여 append
station_name = ''
shortest_x = 0
shortest_y = 0
shortest_dist = 0
```

```
for a in range(max(place_cord['명소 ID'].astype(int))):
    for b in range(max((df_subway['역ID'].astype(int)))):
        place = (place_cord.iloc[a,1], place_cord.iloc[a,2])
        subway = (df_subway.iloc[b,2], df_subway.iloc[b,3])
        dist_temp = haversine(place, subway, unit='km')

        if b == 0:
            station_name = df_subway.iloc[b,1]
            shortest_x = df_subway.iloc[b,2]
            shortest_y = df_subway.iloc[b,3]
            shortest_dist = dist_temp
        else:
            subway_before = (shortest_x, shortest_y)
            if dist_temp < haversine(place, subway_before, unit='km'):
                station_name = df_subway.iloc[b,1]
                shortest_x = df_subway.iloc[b,2]
                shortest_y = df_subway.iloc[b,3]
                shortest_dist = dist_temp
```

```
place_cord.iloc[a,3] = station_name
place_cord.iloc[a,4] = shortest_x
place_cord.iloc[a,5] = shortest_y
place_cord.iloc[a,6] = shortest_dist
```

```
place_cord.head()
```

“앞선 지하철역과의 거리 계산, 좌표 확정 과정과 동일”

```
for a in range(max(place_cord['명소 ID'].astype(int))):
    for b in range(max((df_subway['역ID'].astype(int)))):
        place = (place_cord.iloc[a,1], place_cord.iloc[a,2])
        subway = (df_subway.iloc[b,2], df_subway.iloc[b,3])
        dist_temp = haversine(place, subway, unit='km')

        if b == 0:
            station_name = df_subway.iloc[b,1]
            shortest_x = df_subway.iloc[b,2]
            shortest_y = df_subway.iloc[b,3]
            shortest_dist = dist_temp
        else:
            subway_before = (shortest_x, shortest_y)
            if dist_temp < haversine(place, subway_before, unit='km'):
                station_name = df_subway.iloc[b,1]
                shortest_x = df_subway.iloc[b,2]
                shortest_y = df_subway.iloc[b,3]
                shortest_dist = dist_temp
```

3. 단계별 내용 (1)-2 : 데이터셋 구축 ‘ 명소 데이터셋 ’

1. 진행 배경 및 목적 | 2. 프로세스(일정, 순서) + 팀원 역할 | 3. 단계별 내용 (코드, 이미지) | 4. 프로젝트 결과 및 분석 | 5. 개발 환경 + 참고 문헌

Scoring 기준 2 : “ 대중교통 ↔ 명소 간의 거리 ”

> 두 대중교통 중 더 가까이 있는 대중교통 이용

> “ 최단 거리의 비율 사용 ”

No.	명소 ID	키워드	명칭	버스정류장까지의 거리 계산(이후 병합)					두 대중교통 거리 비교	
				지하철_거리	버스정류장	버스정류장_위도	버스정류장_경도	버스_거리	min_거리	min_거리_비율
1	1	관광 명소	63스퀘어	0.6393917898	63빌딩.가톨릭대학교여의도성모병원	37.51916181	126.9377093	0.1898939526	0.1898939526	0.51%
2	2	관광 명소	몽촌역사관	0.6998768927	현대토파즈아파트	37.52530533	127.1258611	0.2544863082	0.2544863082	0.69%
3	3	관광 명소	경복궁	0.4458521562	경복궁	37.57695016	126.9793378	0.1406490914	0.1406490914	0.38%
4	5	관광 명소	광화문광장	0.1574397226	KT광화문지사	37.57222644	126.9773321	0.04429801244	0.04429801244	0.12%
5	6	관광 명소	국립고궁박물관	0.2912074302	국립고궁박물관.경복궁서문	37.57677167	126.9743664	0.1820562318	0.1820562318	0.49%
6	7	관광 명소	국립기상박물관	0.6246702775	스위스대사관	37.56998879	126.9658058	0.1526984401	0.1526984401	0.41%
7	8	관광 명소	국립중앙박물관	0.67761555	국립중앙박물관용산가족공원	37.52082823	126.979799	0.3245349164	0.3245349164	0.88%
8	9	관광 명소	국립현대미술관	0.5439567133	정독도서관	37.58019153	126.9802007	0.08355141557	0.08355141557	0.23%
9	10	관광 명소	국회의사당	0.3408366268	국회의사당	37.5298981	126.9180242	0.1796387584	0.1796387584	0.49%
10	11	관광 명소	남산서울타워	1.106132002	남산서울타워	37.55124467	126.9907563	0.2521835778	0.2521835778	0.68%
11	12	관광 명소	대한민국역사박물관	0.3395943364	세종문화회관	37.57394965	126.9768673	0.09107421271	0.09107421271	0.25%
12	13	관광 명소	덕수궁	0.2301838314	덕수궁	37.56558455	126.9767268	0.1572481253	0.1572481253	0.42%
13	14	관광 명소	도산공원	0.5833093168	신구중학교	37.52564607	127.0335404	0.1960194543	0.1960194543	0.53%
14	16	관광 명소	롯데월드 민속박물관	0.5506873992	롯데월드	37.51214677	127.0970648	0.09988043547	0.09988043547	0.27%
15	17	관광 명소	롯데월드타워	0.1216827261	잠실역1번.11번출구	37.51415201	127.1030253	0.1399654607	0.1216827261	0.33%
16	18	관광 명소	리움 미술관	0.2556900634	한남동새마을금고	37.53684427	127.0003214	0.1920741798	0.1920741798	0.52%
17	20	관광 명소	명동성당	0.2614578008	서울백병원.국가인권위.안중근활동터	37.56435047	126.9878515	0.1482409332	0.1482409332	0.40%
18	21	관광 명소	문화역서울 284	0.1686115756	서울역버스환승센터	37.55544621	126.9723557	0.08775966488	0.08775966488	0.24%
19	22	관광 명소	보은사	0.267404378	보은사.삼성1파출소.안	37.51406736	127.0504254	0.1733635448	0.1733635448	0.47%

3. 단계별 내용 (1)-3 : 데이터셋 구축 ‘쏘카존 현황 데이터셋’

다함께 차찾자!

클러스터링을 통한 효율적인 쏘카존 관리 모델 제안

1. 진행 배경 및 목적 | 2. 프로세스(일정, 순서) + 팀원 역할 | 3. 단계별 내용 (코드, 이미지) | 4. 프로젝트 결과 및 분석 | 5. 개발 환경 + 참고 문헌

나눔카 거점 데이터 API 활용

- ‘24시간동안 4~5분 간격으로 쏘카존 현황 데이터 저장’

- 해당 데이터를 바탕으로 쏘카존 수요 현황 파악 + (추후) 쏘카존 클러스터링에 활용 가능

```
1 def get_Data():
2     socarCnt = []
3     utc = datetime.datetime.now(datetime.timezone.utc)
4     kst = utc + timedelta(hours=9)
5     nowtime = kst.strftime('%Y-%m-%d_%H%M')
6     try:
7         for id in socarzoneId:
8             url = f'http://openapi.seoul.go.kr:8088/{apikey}/xml/NanumcarCarList/1/5/{id}/so'
9
10            response = requests.get(url)
11            if response.status_code == 200:
12                try:
13                    bs_content = bs(response.content, 'xml')
14                    allcnt = bs_content.find('reservAbleCnt').text
15
16                    socarCnt.append(allcnt)
17                except Exception as e:
18                    socarCnt.append(-100)
19                    print(id, e)
20                    continue
21            else:
22                socarCnt.append(-1)
23
24    except Exception as e:
25        outputdata = pd.DataFrame(socarCnt, columns=[f'{nowtime}'])
26        # 예러시 저장되는 결과파일 (result_err.csv)
27        outputdata.to_csv(f'{garage}//result_err-{nowtime}.csv')
28        print(e)
29        exit()
30
31    outputdata = pd.DataFrame(socarCnt, columns=[f'{nowtime}'])
32    # 결과파일 (result.csv)
33    outputdata.to_csv(f'{garage}//result-{nowtime}.csv', mode='a')
```

Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2022-07-12 02:38	3	1	3	3	1	8	1	1	1	0	2	1	1	0	2
2022-07-12 02:46	3	1	3	3	1	8	1	1	1	0	2	1	1	0	2
2022-07-12 02:51	8	2	3	3	1	8	1	1	1	0	2	1	1	0	2
2022-07-12 02:57	3	1	3	3	1	8	1	1	1	0	2	1	1	0	2
2022-07-12 03:03	3	1	3	3	1	8	1	1	1	0	2	2	1	0	2
2022-07-12 03:08	3	1	3	3	1	8	1	1	1	0	2	2	1	0	2
2022-07-12 03:14	3	1	3	3	1	8	1	1	0	0	2	2	1	0	2
2022-07-12 03:19	3	1	3	3	1	8	1	1	0	0	2	2	1	0	2
2022-07-12 03:25	3	1	3	3	1	8	1	1	0	0	2	2	1	0	2
2022-07-12 03:30	3	1	3	3	1	8	1	1	0	0	2	2	1	0	2
2022-07-12 03:36	3	1	3	2	1	8	2	1	0	0	2	2	1	0	2
2022-07-12 03:41	3	1	3	2	1	8	2	1	0	0	2	2	1	0	2
2022-07-12 03:47	3	1	3	2	1	8	2	1	0	0	2	2	1	0	2
2022-07-12 03:52	3	1	3	3	1	8	2	1	0	0	2	2	1	0	2
2022-07-12 03:58	3	1	3	3	1	8	2	1	0	0	2	2	1	0	2
2022-07-12 04:03	3	1	3	3	1	8	2	1	0	0	2	2	1	0	0
2022-07-12 04:09	3	1	3	3	1	8	2	1	0	0	2	2	1	0	0
2022-07-12 04:14	3	1	4	3	1	8	2	1	0	0	2	2	1	0	0
2022-07-12 04:19	3	1	4	3	1	8	2	1	0	0	2	2	1	0	0
2022-07-12 04:25	3	1	4	3	1	8	2	1	0	0	2	2	1	0	0
2022-07-12 04:30	3	1	4	3	1	8	2	1	0	0	2	2	1	0	0
2022-07-12 04:36	3	1	4	3	1	7	2	1	0	0	2	2	1	0	0
2022-07-12 04:41	3	1	4	3	1	7	2	1	0	0	2	2	1	0	0
2022-07-12 04:47	3	1	4	3	1	7	2	1	0	0	2	2	1	0	0
2022-07-12 04:52	3	1	4	3	1	7	2	1	0	0	2	2	1	0	0
2022-07-12 04:58	3	1	4	3	1	7	2	1	0	0	2	2	1	0	0
2022-07-12 05:03	3	1	4	3	1	7	2	1	0	0	2	2	1	0	0
2022-07-12 05:09	3	1	4	3	1	7	2	1	0	0	2	2	1	0	0

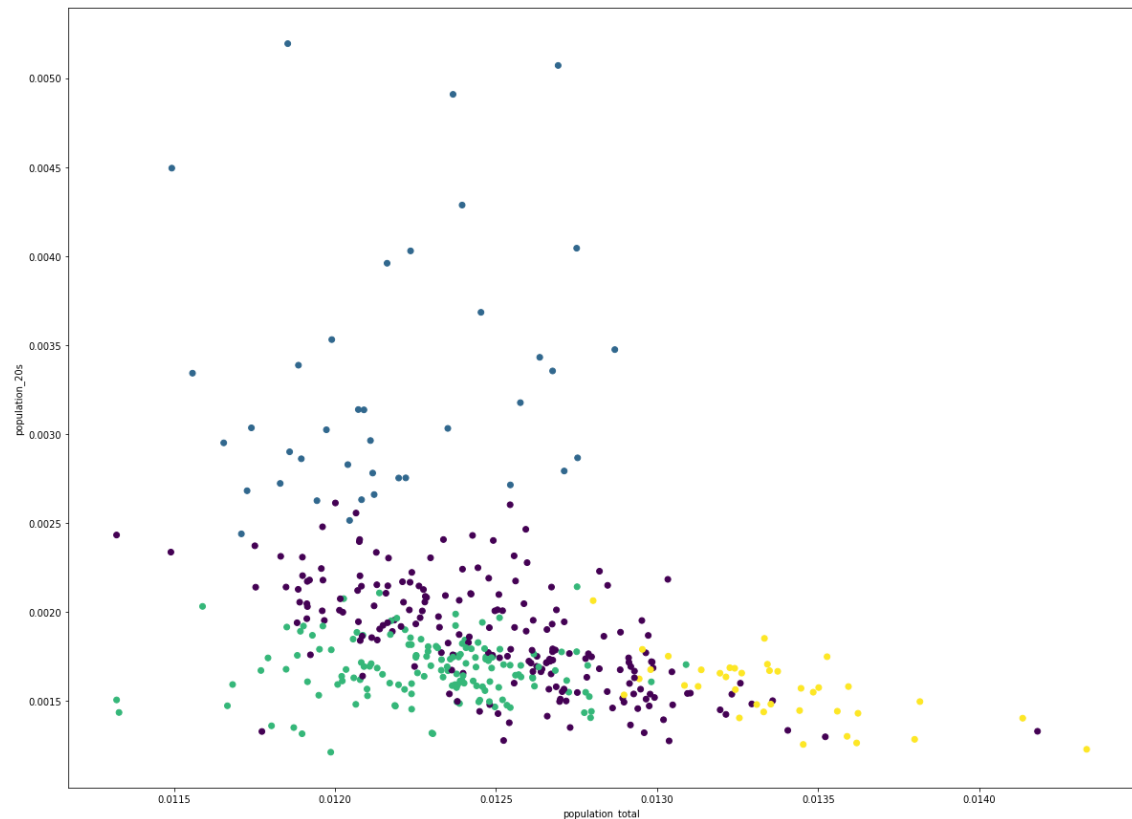
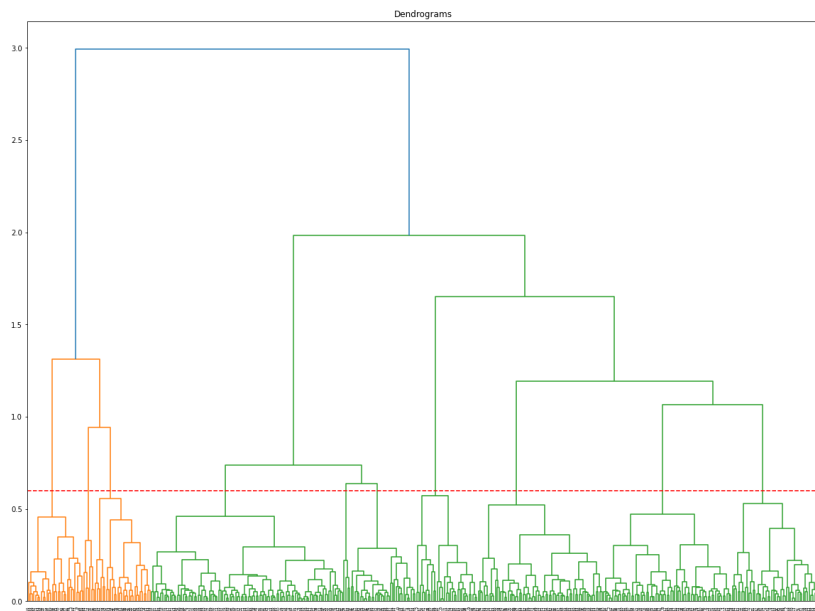
PCA(주요성분분석) + K-Means 클러스터링 활용

- 27개에 달하는 특성(변수)를 담은 동별 데이터셋의 차원 문제 해결을 위한 솔루션

STEP 1. Hierarchical Clustering

- 가장 보편적인 군집화 방법인 K-Means 클러스터링 진행에 앞서 군집 개수를 설정하기 위해 hierarchical clustering 진행.
- 원본 데이터 Normalization 후 클러스터링에 활용
- 생성한 Dendrogram 을 기반으로, 클러스터 개수로 4개가 적합하다 판단

< Dendrogram 예시 >



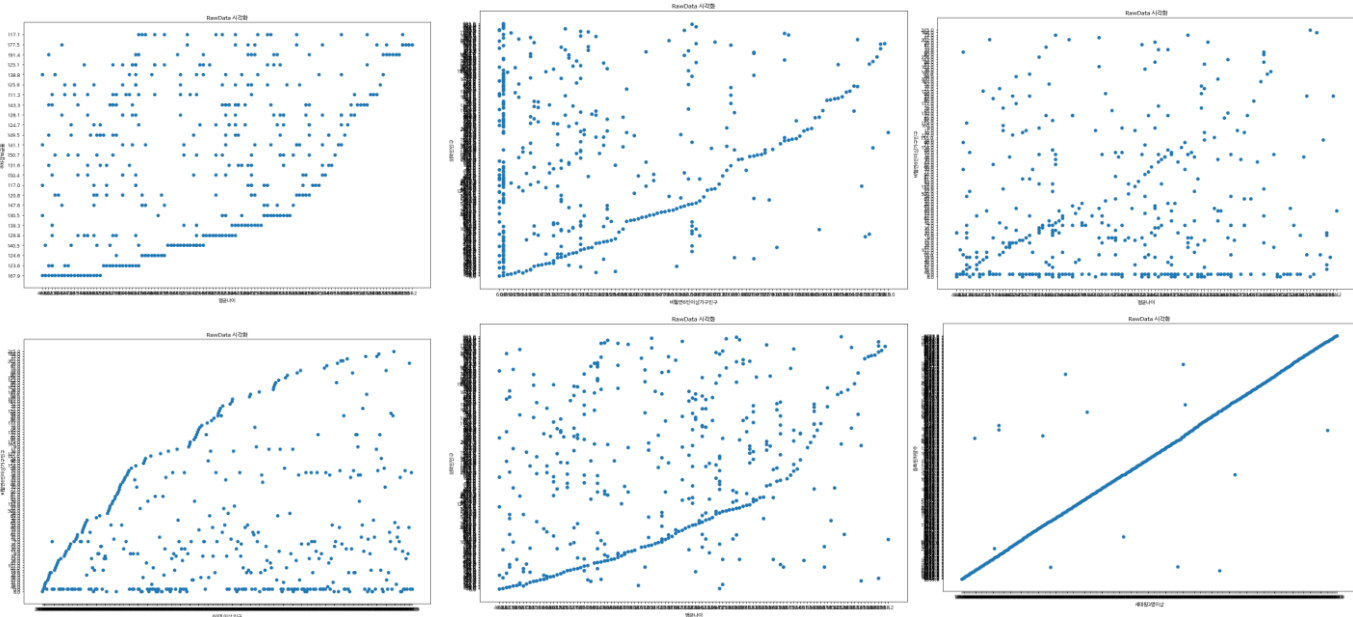
- hierarchical 군집화 수행결과 시각화 하였을 때, 변수 특징별로 구분 가능한 정도의 차이가 컸음.
- 추가로 DBSCAN, KMeans 클러스터링 진행했으나, 만족할만한 결과를 얻지 못함.

PCA(주요성분분석) + K-Means 클러스터링 활용

- 27개에 달하는 특성(변수)를 담은 동별 데이터셋의 차원 문제 해결을 위한 솔루션

STEP 2. 1) Visualize Raw Data

- PCA를 통한 K-Means 클러스터링 진행에 앞서 원본 데이터 시각화 진행
- 변수간 상관관계 살펴보는 과정



2) Standardization

- 데이터의 변수간에 값 차이가 매우 큰 (0~80,000+) 것을 고려하여 Normalization 대신 표준편차를 갖도록 변환해주는 Standardization 시도.

```
1 data_stand_df = pd.DataFrame(dataset_stand_np)
2 dataset_stand_df = data_stand_df.rename(columns=columns)
3 print(dataset_stand_df)
```

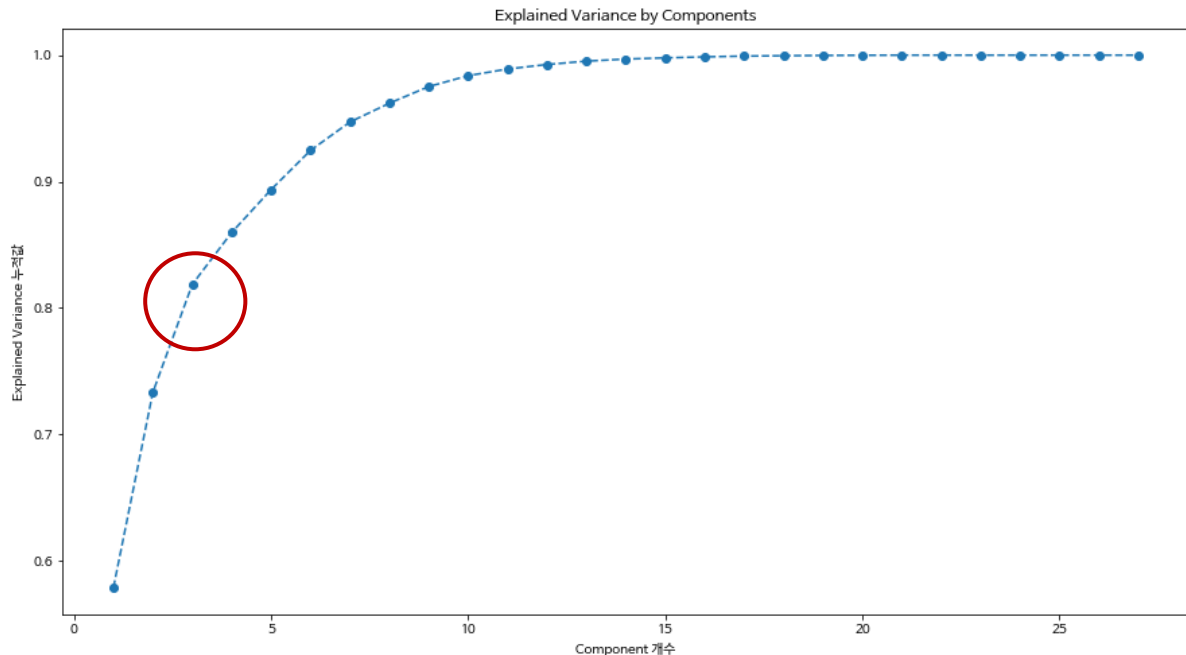
	위치	평균나이	세대원1명	세대원2명	세대원3명 이상		
0	0.836920	-1.382267	-1.797308	-1.294330	-1.294033	-1.869944	-1.485559
1	-1.034071	-1.059289	-0.129389	1.305323	-1.053234	-0.374566	0.357474
2	-0.653531	-0.372326	-0.047765	0.342551	-0.473991	-0.359895	-0.145753
3	-0.463260	1.799610	0.066079	-0.857678	1.820703	0.570655	-0.872960
4	-0.368125	0.692050	-0.243233	-0.431155	0.634250	0.177686	-0.477359
...
419	-0.272990	1.112284	0.150925	-0.802534	0.961792	0.217507	-0.792095
420	-0.019296	-0.232248	1.529938	1.970430	-0.404785	1.377551	2.206907
421	0.614938	-0.500065	0.082189	-0.112669	-0.477309	0.114811	-0.023582
422	0.361244	-0.347286	0.082189	-0.294982	-0.439388	-0.032945	-0.184149
423	1.027190	1.020471	0.870503	-0.264597	1.030998	0.856737	-0.117828

PCA(주요성분분석) + K-Means 클러스터링 활용

- 27개에 달하는 특성(변수)를 담은 동별 데이터셋의 차원 문제 해결을 위한 솔루션

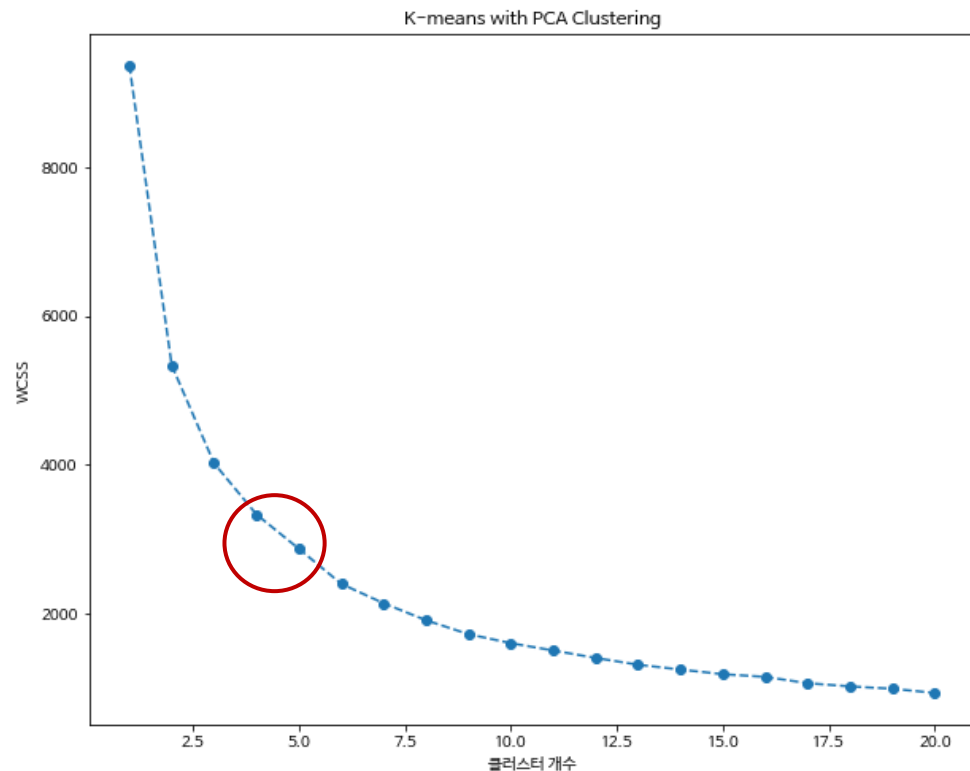
STEP 3. Principal Component Analysis, PCA

- 다차원의 구조에서, 클러스터링에 가장 많은 영향을 끼치는 주요 요소를 추출하는 PCA 적용
- Component 갯수별 Variance 값 확인 후, 원형 데이터가 보존된다고 판단되는 80% 선의 Component 개수(3개) 채택



STEP 4. 클러스터링 with PCA

- Within Cluster Sum of Squares(WCSS) 계산 후 Elbow method를 활용해 클러스터 숫자 선정

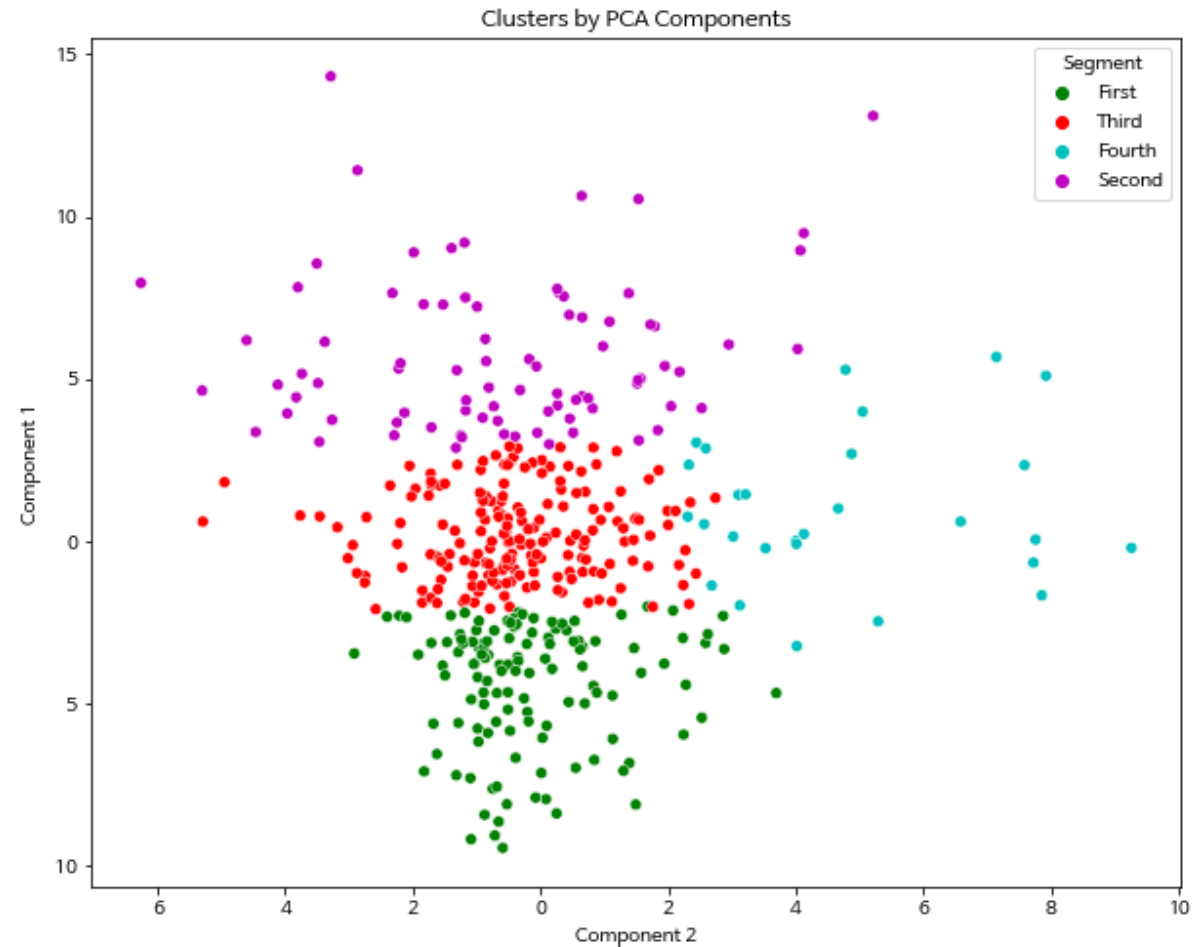
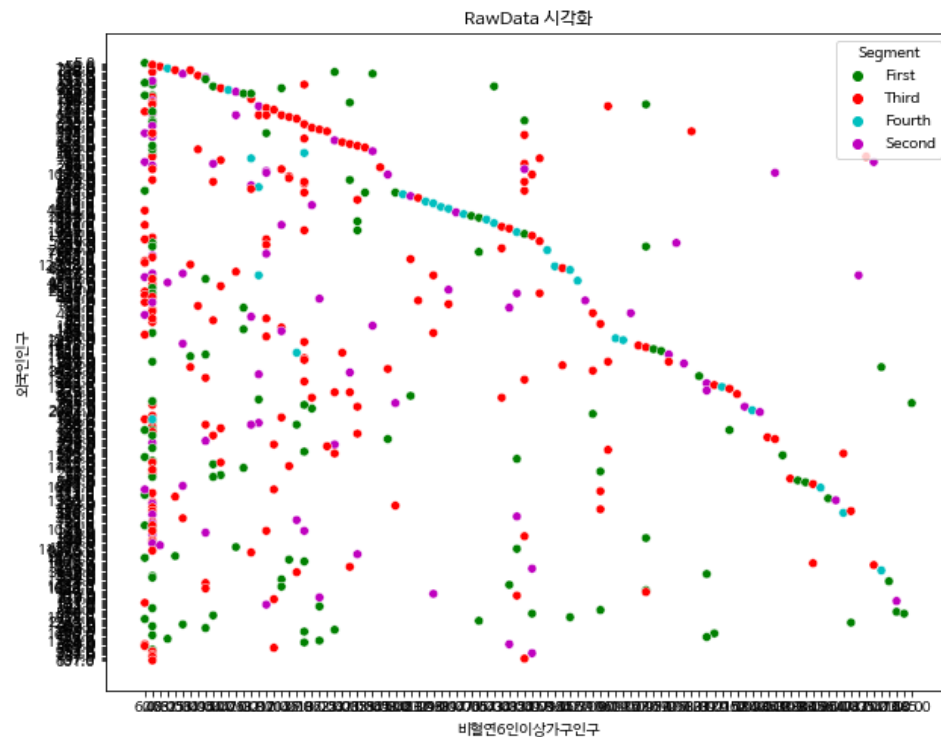


PCA(주요성분분석) + K-Means 클러스터링 활용

- 27개에 달하는 특성(변수)를 담은 동별 데이터셋의 차원 문제 해결을 위한 솔루션

STEP 5. K-means 클러스터링

- 앞서 구한 클러스터 개수(4)에 맞춰 K-Means 군집화 진행.
- 기존 변수 기준으로 시각화한 것보다 PCA 요소 기준으로 시각화한 그래프에서 더 명확하게 군집화가 되었음을 볼 수 있음.



관광명소에 대한 지역별(동별) 쏘카 수요 예측

산정식 = (전체 쏘카수) X {해당 클러스터 수요 / 전체 클러스터 수요}

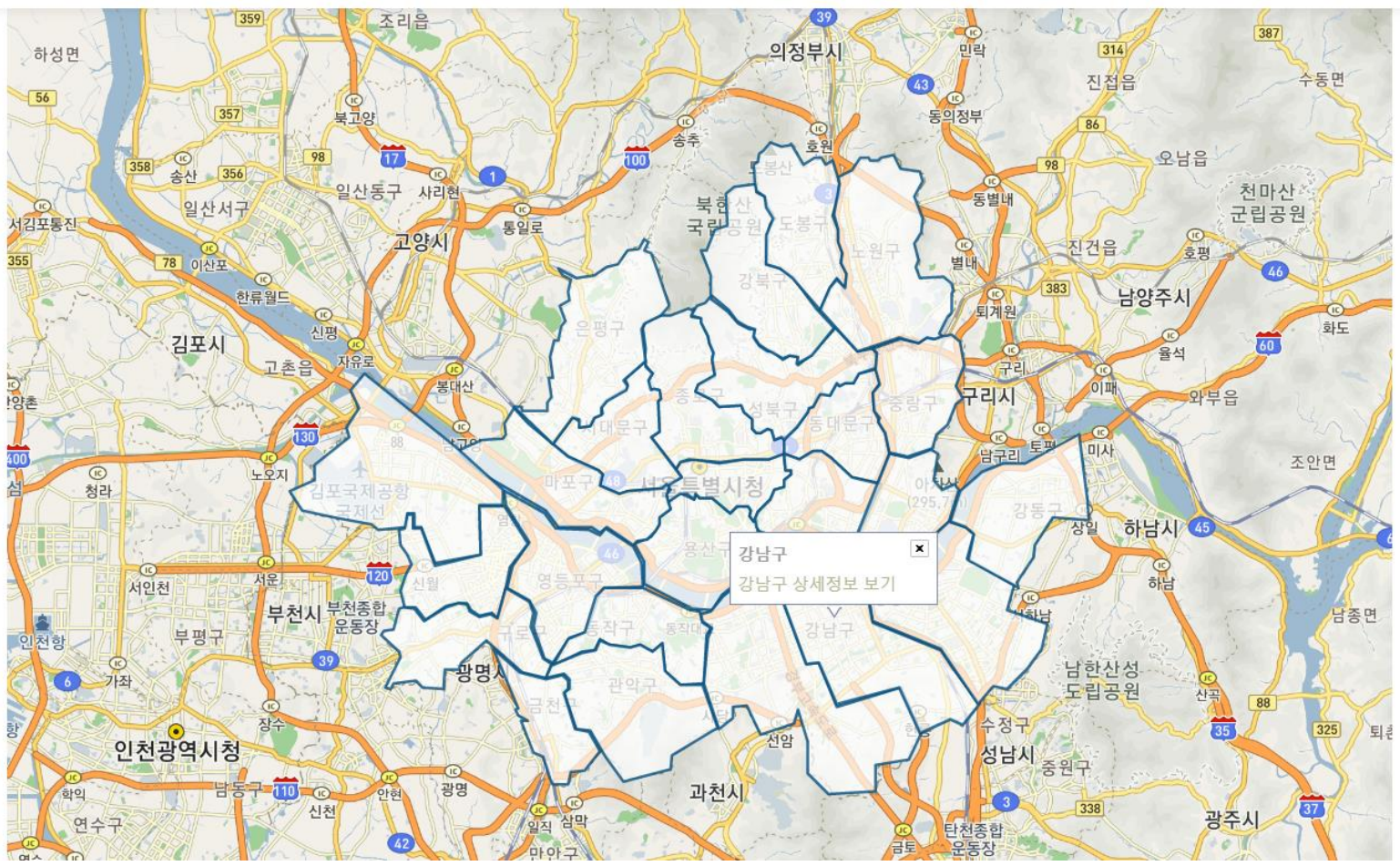
* 해당 클러스터 수요 = 클러스터에 해당하는 **동의 관광명소 수요 합계값

** 동별 관광명소 수요 합계 = [{㉔전체 검색량 중 해당 관광명소의 검색량이 차지하는 비율(M열) X ㉕검색 연령 비율 X ㉖min_거리 비율} X 해당 연령의 동별 인구수]의 연령별 합계값

명소 ID	키워드	명칭	㉖ 연령별 검색율 = (여성 검색율 x 연령별 검색율) + (남성 검색율 x 연령별 검색율)					
			10s	20s	30s	40s	50s	60+s
3	관광 명소	경복궁	0.0054	0.0154	0.0762	0.3832	0.31	0.2198
5	관광 명소	광화문광장	0.0169	0.0576	0.0693	0.2672	0.3952	0.1938
6	관광 명소	국립고궁박물관	0.0038	0.0224	0.1224	0.3382	0.2876	0.2232
7	관광 명소	국립기상박물관	0	0.0054	0.0624	0.5662	0.237	0.1236
8	관광 명소	국립중앙박물관	0	0.0162	0.1048	0.3972	0.2776	0.2042
9	관광 명소	국립현대미술관 서울관	0	0.01	0.0573	0.2346	0.3592	0.3316
10	관광 명소	국회의사당	0	0.02	0.0884	0.3678	0.3158	0.1996
11	관광 명소	남산서울타워	0	0.03	0.11	0.425	0.265	0.165
12	관광 명소	대한민국역사박물관	0	0.0155	0.111	0.457	0.2435	0.1675
13	관광 명소	덕수궁	0	0.0261	0.0822	0.3327	0.3083	0.2468
14	관광 명소	도산공원	0.0044	0.0424	0.1356	0.328	0.2968	0.2028
16	관광 명소	롯데월드 민속박물관	0.0062	0.0124	0.1248	0.5814	0.19	0.0814

STEP 1. 메인화면 → 특정 구 클릭

다함께 차찾자



STEP 2.
해당하는 ‘구’에 속한 ‘동’ 목록 출력
→ 특정 동 선택

강남구

- 개포1동
- 개포2동
- 개포4동
- 논현1동
- 논현2동
- 대치1동
- 대치2동
- 대치4동
- 도곡1동
- 도곡2동
- 삼성1동
- 삼성2동
- 세곡동
- 수서동
- 신사동
- 압구정동
- 역삼1동

STEP 3. 동별 현황 확인

- 선택한 동에 해당하는 인구 클러스터, 방문하는 명소와 그에 따른 쏘카 수요 예측 명시

- 페이지 하단에는 해당 인구 클러스터의 특징적인 요소 출력

클러스터링 정보

서울시 강남구 신사동 Information

2022/07/12 02:38 ~ 2022/07/13 03:25 약 24시간 기준

인구 클러스터

Cluster #1
평균 나이 : 44.928세
등록된 차량수 적음 - 평균 4385.464

방문하는 명소

1위 - 음식점 "마이썸"
2위 - 문화시설 "골든블루마리나"
3위 - 자연명소 "노원우주학교"

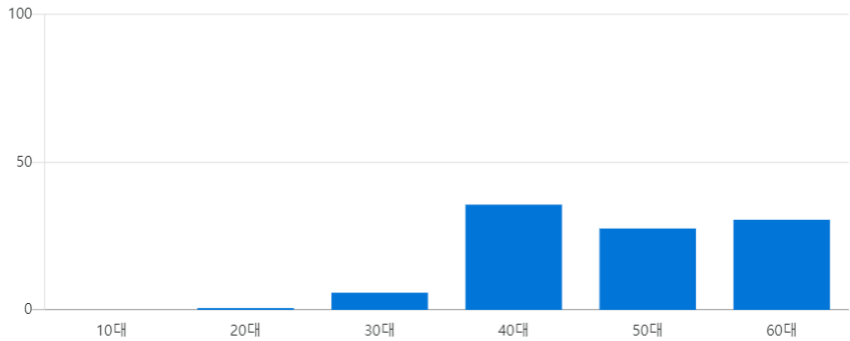
현재 SOCAR Zone 현황 - Target

실제 차량 수 : 8
사용 중인 평균 차량수 : 4.469
사용 가능한 평균 차량수 : 3.531

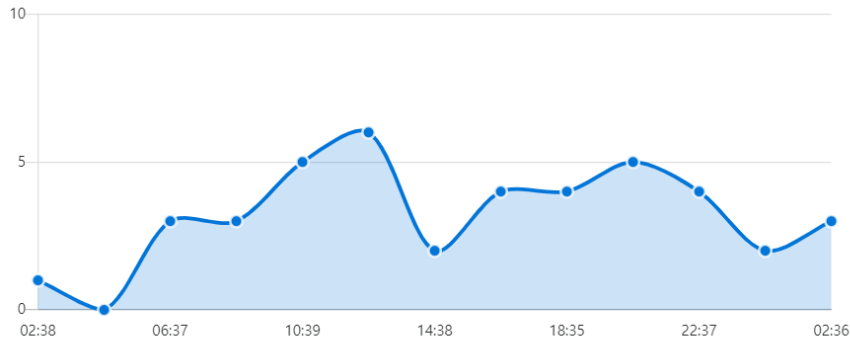
SOCAR Zone Update - Output

쏘카존 수 : 4
예측한 쏘카 수요량 :
추가로 필요한 차량수 :

인구 특성에 따른 쏘카 수요 예측 확률



2시간 단위의 쏘카존 사용 현황



인구 클러스터의 feature

10 entries per page

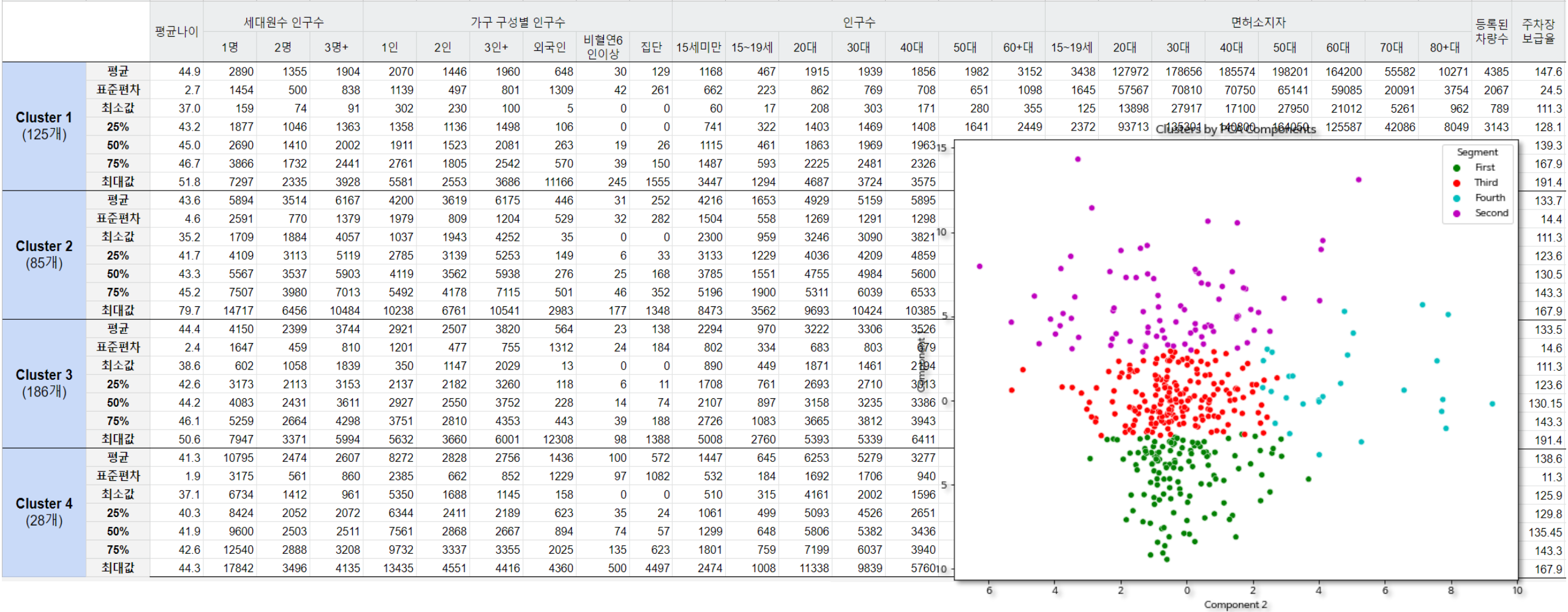
Search...

Cluster	평균 나이	세대원 1명	세대원 2명	세대원 3명	1인 가구	2인 가구	3인 이상 가구	외국인 가구	등록 차량 수
#1 (125개 동)	31 ~ 57.8 Average : 44.928	159 ~ 7297 Average : 2890.048	74 ~ 2335 Average : 1354.776	91 ~ 3928 Average : 1903.888	302 ~ 5581 Average : 2069.616	230 ~ 2553 Average : 1446.368	100 ~ 3686 Average : 1960.472	5 ~ 11166 Average : 647.544	789 ~ 13272 Average : 147.5576

4. 프로젝트 결과 및 분석 - 클러스터링 (동별 데이터셋)

1. 진행 배경 및 목적 | 2. 프로세스(일정, 순서) + 팀원 역할 | 3. 단계별 내용 (코드, 이미지) | 4. 프로젝트 결과 및 분석 | 5. 개발 환경 + 참고 문헌

- 다차원 데이터를 대한 효과적인 클러스터링 진행 (Hierarchical, Kmeans, DBSCAN → PCA + Kmeans)
- 27개 변수에 대한 클러스터별 평균, 표준편차, 최대/최소값 등 확인



관광명소에 대한 클러스터별(동별) 쏘카 수요 예측

- 클러스터에 속한 ‘동’ 리스트 추출 + 해당하는 쏘카존ID 리스트 추출
- 클러스터별 예상되는 쏘카 수요 산정 → 전체 쏘카차량수(2,200대) X 예측 수요(%)
- 클러스터별 배치된 쏘카 차량수 산정 + 예측 차량수와 비교

* ‘관광명소 수요’ 하나만을 기준으로 했을 경우에 해당. 실제 수요는 더 많은 변수와 상황을 고려해야 함.

No	자치구	위치	클러스터	해당 쏘카존ID
1	강남구	개포1동	1	
2	강남구	개포2동	3	105
3	강남구	개포4동	3	104 1198
4	강남구	논현1동	4	277 281 1223
5	강남구	논현2동	3	1222
6	강남구	대치1동	3	
7	강남구	대치2동	2	578 1224 2166
8	강남구	대치4동	3	323 1196
9	강남구	도곡1동	3	903 2366
10	강남구	도곡2동	2	322
11	강남구	삼성1동	1	1242
12	강남구	삼성2동	2	
13	강남구	세곡동	2	
14	강남구	수서동	1	748
15	강남구	신사동	1	777 888 2024 2236
16	강남구	압구정동	3	
17	강남구	역삼1동	4	785 61 62 64 67 205 705 923 937 943 947

클러스터	동	쏘카존 수	쏘카존 ID 리스트	현재 배치된 쏘카 차량수 합계	예측된 수요 (%)	예측 수요에 따른 쏘카존 배정 수량	예측-실제 차량 배치수량 차이 - : 공급과다 + : 공급부족
1	가리봉동, 가양2	112	53, 1974, 360, 2	409	16.63	366	-43
2	가락2동, 강일동,	133	2, 4, 18, 34, 35,	516	31.81	700	184
3	가락1동, 가락본	210	31, 32, 40, 57, 7	861	45.28	996	135
4	가산동, 가양1동,	67	47, 49, 51, 55, 6	462	6.28	138	-324

Keep

- 부족한 시간과 리소스에도 불구하고 결과물을 만들어낸 점
- 각자의 강점을 활용한 역할 배분
- 모델링에서 멈추지 않고, 웹 구현까지 진행하여 실사용자(쏘카존 관리자)가 활용할 수 있는 프로덕트 만들어낸 점

Problem

- 상당 부분의 시간을 데이터셋 구축에 할애하여 모델링과 결과 분석 및 개선에 충분한 시간 확보하지 못함
 - 클러스터링과 수요 예측 결과물과 방법론에 대한 타당성 검토 절차 부재
 - 클러스터링 결과에 대한 세부적인 분석 부재
 - 장기간(최소 한 달) 쏘카존 현황 데이터 기반 클러스터링 진행 X
- 명소에 대한 키워드 검색량 데이터셋 제작 시, 네이버(검색량)와 카카오(성별/연령별 수요 비율)의 결과를 섞어서 활용.
→ 플랫폼의 이용 연령이 편향이 데이터에 반영됨.

Try

- 클러스터링 및 수요예측 결과 분석 및 개선 절차 도입
- PCA와 결합해 다양한 클러스터링 모델 실험 → 클러스터링 고도화
- 장기간의 쏘카존 현황 데이터를 기반, 쏘카존 클러스터링 진행 → 공급/수요 관리 고도화

“ 개발 환경 ; 소통(slack), 버전관리, 데이터베이스, 클라우드 등 ”

소통 : 비대면 _ 디스코드, 구글 Meet / 대면 _ 오프라인 회의

버전 관리 : 각자 맡은 파트 ipynb 파일 생성 및 공용 드라이브에 공유

데이터베이스 : Google Spreadsheet

클라우드 : Google Drive

“ 참고 문헌 ”

논문

; <https://www.nature.com/articles/s41598-022-06767-7>

자료

1) PCA + Kmeans - 1

; <https://365datascience.com/tutorials/python-tutorials/pca-k-means/>

2) PCA + Kmeans - 2

; <https://ranger.uta.edu/~chqding/papers/KmeansPCA1.pdf>

**1부 마침.
감사합니다.**

곧 2부가 이어집니다.