데이터기반 행정으로 국민의 삶의 질을 개선하라!

데이턴십 해커톤 제 4회

웰빙 시니어 라이프를 위한 노인 친화형 다세대 놀이터 제안과 최적 입지 선정

- 서울특별시를 중심으로

분석 결과보고서

참여조 : 서울2 수목2반 56조

참여자 : 최수진(조장)

김다희

김혜인

박진원

오윤정

유시은

씨에스리 컨소시엄

CSLEE kpc 한국생산성본부

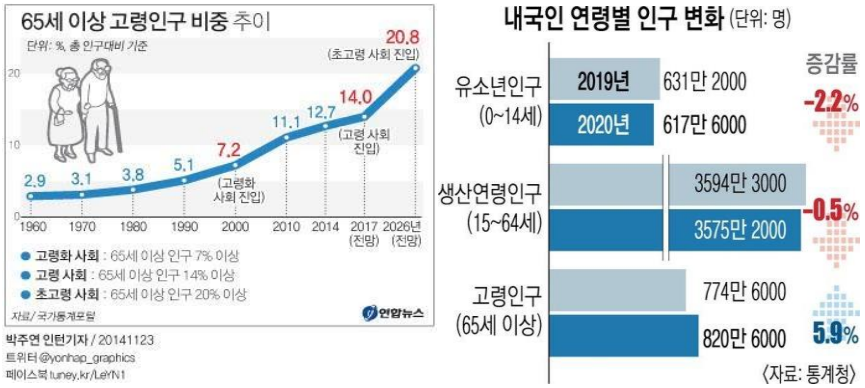
목 차

1. 분석 개요

1.1. 분석 배경 및 개요

❑ 고령 인구 비중의 증가 속도가 빨라짐에 따라 고령 사회에서 초고령 사회로의 진입이 임박함

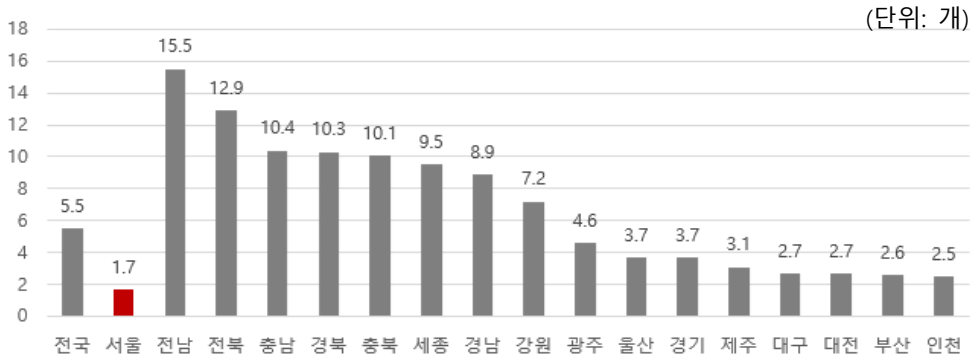
- 65 세 이상 인구 비중에 따라 고령화 사회(7% 이상), 고령 사회(14% 이상), 초고령 사회(20% 이상)로 분류함
- 대한민국은 2017 년 65 세 이상 비중이 14.2%를 기록하며 고령 사회에 진입함. 고령 인구 비중은 2018 년에 0.6%, 2019 년에 0.7%, 2020 년에 0.9% 증가하며 고령 인구 비중의 증가 폭이 확대됨
- '20 년 통계청 인구주택총조사에 따르면, 65 세 이상 고령 인구는 인구의 16.4%인 820 만 6000 명으로, 전년 대비 46 만 명 증가함



<그림 1> 65 세 이상 고령인구 비중
출처: 연합뉴스. "65 세 이상 고령인구 비중 추이". 2014.11.23.

❑ **초고령 사회에 진입하면서 고령 인구층의 일상생활 서비스 지원과 이들이 자립할 수 있는 프로그램을 제공하는 장소가 서울에 시급함**

- ‘20 년 통계청의 노인 천 명당 노인 여가복지시설 수는 전국 평균 5.5 개임에 반해서, 서울은 1.7 개로 전국에서 가장 낮은 수치임



<그림 2> 노인 천 명당 노인 여가복지시설 수
출처: 통계청. 2020 년 노인 천명당 노인 여가복지시설 수

❑ 다수의 고령 인구가 1 개 이상의 만성질병을 앓고 있으나, 노인 건강을 증진시키고 질병을 예방할 수 있는 복지시설이 미비함

- ‘20 년 보건복지부 노인실태조사에 따르면 응답자의 84%가 1 개 이상의 만성질병을 앓고 있으며, 연령이 높을수록 유병률이 증가함

(단위:%, 명, 개)

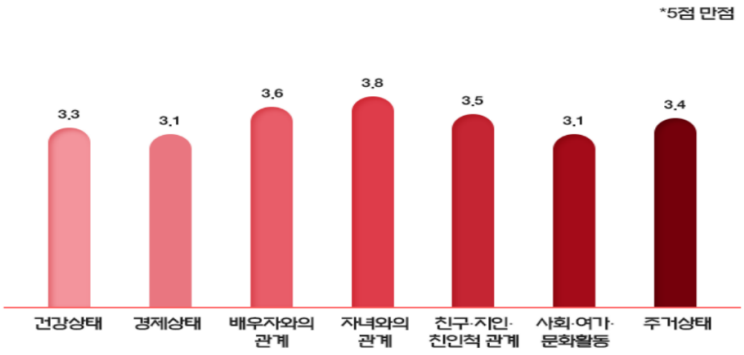
특성	없음	1개	2개	3개 이상	계(명)	평균(개)
전체	16.0	29.2	27.1	27.8	100.0 (10,097)	1.9
65-69세	24.4	33.4	24.0	18.2	100.0 (3,344)	1.5
70-74세	16.3	29.4	28.0	26.4	100.0 (2,342)	1.9
75-79세	10.2	28.4	28.5	32.9	100.0 (2,292)	2.1
80-84세	9.2	24.6	28.6	37.6	100.0 (1,475)	2.3
85세 이상	7.0	20.0	31.0	42.1	100.0 (644)	2.6

<표 1>‘ 20 년 노인의 연령별 만성질병(의사진단 기준) 개수
출처: 보건복지부 노인실태조사

- ‘16~’20 년 보건복지부 노인 복지시설 현황에 따르면 건강과 관련된 시설은 ‘노인요양시설’과 ‘노인요양공동생활가정’으로, 노인이 주체적으로 참여할 수 있는 시설은 미비함

❑ 서울시 노인의 우울지수가 증가하였고 15%의 노인이 우울증상을 보였으며, 사회·여가·문화 활동 및 건강상태에 대한 만족도가 낮음

- ‘20 년 서울시 노인실태조사에 따르면 서울시 어르신 노후생활 만족도는 5 점 만점 중 사회·여가·문화 활동(3.1 점), 경제상태(3.1 점), 건강상태(3.3 점) 순으로 낮음



<그림 3> 서울시 어르신 노후생활 만족도
출처: 서울시복지재단. 2020 년 서울시 노인실태조사

❑ 국민체육진흥법에 노인체육을 직접 규율하는 조항이 명시되어 있으나 정책을 구체적으로 규율하기에는 미비하며, 노인체육을 협소한 틀로 규정함

- 문화체육관광부와 보건복지부를 중심으로 노인체육 정책을 시행하고 있지만, 협력과 협업보다는 행정편의주의적 정책으로 기존 정책을 반복하는 형태만 지속되고 있는 상황임

국민체육진흥법
<시행 2021.06.09>[법률 제 17580, 2020.12.08. 일부개정]
제 1 조: 이 법은 국민체육을 진흥하여 국민의 체력을 증진하고, 체육활동으로 연대감을 높이며, 공정한 스포츠 정신으로 체육인인권을 보호하고, 국민의 행복과 자긍심을 높여 건강한 공동체의 실현에 이바지함을 목적으로 한다.

제 10 조의 2(노인 체육의 진흥) ① 국가와 지방자치단체는 노인 체육진흥에 필요한 시책을 마련하여야 한다.

② 국가와 지방자치단체는 노인 건강의 유지 및 증진을 위한 맞춤 체육활동 프로그램을 운영하거나 그 운영에 필요한 비용 및 시설을 지원할 수 있다.

<표 2> 국민체육진흥법 제 1 조 및 제 10 조의 2

❑ 해외 사례의 경우 스페인 바르셀로나와 독일 뉘른베르크 등 유럽 주요 도시에서는 노인을 위한 공간이 성공적으로 자리매김함

- 스페인의 경우 어린이 놀이터 인근에 노인 놀이터를 설치하여 양쪽 세대 모두 소외되지 않도록 하였고, 독일의 경우는 더 나아가 지역 사회에 세대간 통합을 도모하는 공간을 조성하는데 심혈을 기울임
- **이를 참고하여 노인 놀이터를 세대간 통합을 이룰 수 있는 <노인 친화형 다세대 놀이터>로 발전시킬 필요가 있음**

1.2. 분석 목적 및 방향

1.2.1. 분석 목적

❑ 본 연구는 노인비율 뿐만 아니라 이용 편리성, 지역 주민 접근성 등을 고려한 노인 친화형 다세대 놀이터의 최적 입지를 선정하고자 함

- 노인 친화형 다세대 놀이터의 주 연령층은 노인(만 65 세 이상)으로 하되, 누구나 이용할 수 있도록 전 세대를 대상으로 함
- 이후 언급되는 노인 놀이터는 노인 친화형 다세대 놀이터로 간주함

1.2.2. 분석 방향

❑ SNS 와 뉴스 댓글 크롤링을 통해 노인 전용 복지에 대한 여론을 워드 클라우드로 분석 및 시각화 함

❑ 노인들의 생활을 전반적으로 고려하여 서울시 노인 놀이터의 최적 입지를 선정하기 위해 다음의 <그림 4>과 같은 프로세스로 진행됨

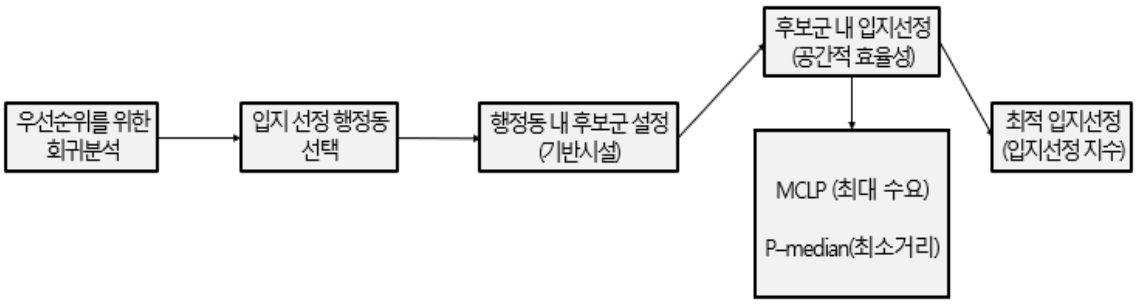
❑ 노인거주인구 및 생활인구, 기존 노인시설(개수, 수용인원) 등을 고려하여 노인 놀이터 입지 선정 행정동을 선택함

- 행정동을 고르는 우선순위를 구하기 위해 회귀분석을 진행함
- 우선순위 1) 노인 거주 및 생활 인구가 많지만 노인 시설이 적은 행정동, 2) 노인 거주 및 생활 인구가 많고 접근성 및 편리성이 좋은 행정동

❑ 클러스터링을 통해 기반시설을 중점으로 행정동 내 후보군을 설정함

❑ 공간적 효율성을 고려하여 MCLP 및 P-Median 방법을 사용하여 후보군 내 입지 후보지를 선정함

❑ 이후 고도 데이터(DEM) 및 입지선정지수를 활용하여 서울시 노인 놀이터 최적 입지를 선정함



<그림 4> 분석 프로세스

1.3. 분석 결과 활용 방안

1.3.1. 정책적 활용

❑ (노인 복지) 초고령화 사회에 대비하여 노인들에게 가장 실효성이 있는 노인복지정책으로 활용 가능함

❑ (노인 교육) 노인 놀이터에서 진행하는 다양한 프로그램을 통해 노인을 위한 교육의 장을 마련할 수 있음

❑ (그린 뉴딜 연계) 노인 놀이터와 한국형 그린 뉴딜 사업의 연계를 통한

성공적인 포용 사회로의 도약에 이바지함

1.3.2. 경제적 활용

- ❑ (일자리 창출) 노인 놀이터 사업과 관련한 새로운 일자리를 창출함
- ❑ (의료비 감소) 일상적인 생활 체육 활동으로 노인의 질병 예방 및 신체적·정신적 건강 증진을 통해 노인 의료비가 감소함
- ❑ (지역 경제 활성화) 노인들의 건강 증진을 통한 지역 경제 이바지 및 공동체 사회로의 연결이 가능함

1.3.3. 사회적 활용

- ❑ (인적 네트워크) 운동과 여가를 동시에 즐길 수 있는 공간 내에서 활발한 인적 네트워크를 형성할 수 있음
- ❑ (세대 통합) 노인만의 전용 공간이 아닌 어린이와 젊은이도 함께 이용하는 세대 통합형 공원으로 활용 가능함
- ❑ (인식 제고) 여론 분석을 통해 현재 노인 놀이터의 개선 방향을 파악하여 다세대 놀이터로의 전환 및 긍정적 인식을 제고함

1.3.4. 분석 활용

- ❑ 직접 개발한 맞춤형 입지선정지수를 활용해 공간적 효율성을 고려한 접근성 및 효율성 높은 입지 선정이 가능함
- ❑ 서울시 외 다른 지역에서의 노인 놀이터 최적 입지 선정 시 활용 가능함
- ❑ 추후 노인 관련 사업 시행 시 활용할 수 있는 레퍼런스를 제공함

2. 분석 데이터

2.1. 분석 데이터 목록

2.1.1. 데이터 활용 개요

❑ 지역주민 접근성(버스 정류장, 지하철역, 도로데이터, 주택단지)

+ 노인비율 (행정동별 거주 고령 인구 수, 행정동별 시간대별 노인생활인구)

+ 시설위치 (노인 여가복지시설, 경로당, 노인교실)

+ 이용 편리성 (주차장 유무, DEM 고도)

→ 입지후보군 (공원, 공공체육시설)

2.1.2. 활용 데이터 목록표

구분	분석 데이터	기간	제공기관
크롤링	SNS 와 뉴스 댓글	2017.01-2021.07	유튜브 https://www.youtube.com/ 다음 뉴스 https://news.daum.net/
행정동 기준 데이터	센서스용 행정구역경계 (읍면동)	2020.06	통계지리정보서비스 https://sgis.kostat.go.kr/
	행정구역코드	2021.06	행정안전부 https://www.mois.go.kr//
입지후보군	전국체육시설현황 데이터	2021.07	문화 빅데이터 플랫 https://www.bigdata-culture.kr/
	전국도시공원정보 표준데이터	2021.08	공공데이터포털 http://data.seoul.go.kr/
노인비율	서울시 고령자현황 (동별) 통계	2021.08	서울열린데이터광장 http://data.seoul.go.kr/
	행정동별 서울생활인구	2020.06-2021.07	서울열린데이터광장 http://data.seoul.go.kr/

	(내국인)		
시설 위치	서울특별시 노인 여가복지시설 목록	2021.01-2021.08	서울열린데이터광장 http://data.seoul.go.kr/
	서울특별시 경로당 현황(2020)	2018.11-2020.11	서울열린데이터광장 http://data.seoul.go.kr/
지역주민 접근성	서울특별시 버스노선별 정류소 정보	2021.05	서울열린데이터광장 http://data.seoul.go.kr/
	노선별 지하철역 위치 정보	2021.08	카카오 API https://developers.kakao.com/
	서울시 주택관리 현황	2020.02	서울열린데이터광장 http://data.seoul.go.kr/
	서울도로데이터	2021.07	국가공간정보포털 http://www.nsdi.go.kr/
이용 편리성	서울시 수치표고모델(DEM)	2021.05	국가공간정보포털 http://www.nsdi.go.kr/lxportal/
	공영주차장 위치 정보	2021.08	카카오 API https://developers.kakao.com/

2.2. 데이터 상세 설명

2.2.1. 분석데이터 상세목록표

활용구분	분석 데이터	데이터 형식	생성주기
공공 데이터	서울시 고령자현황 (동별) 통계	CSV	정기(분기)
	행정동별 서울생활인구(내국인)	CSV	매일
	서울특별시 노인 여가복지시설 목록	CSV	매일
	서울특별시 경로당 현황(2020)	XLSX	수시

	서울특별시 버스노선별 정류소 정보	XLSX	비정기 (수시)
	서울시 주택관리 현황	XLSX	비정기 (수시)
민간 데이터	SNS와 뉴스 댓글	비정형	수시
	센서스용 행정구역경계(읍면동)	SHP	연간
	행정구역코드	XLSX	비정기 (수시)
	전국체육시설현황 데이터	CSV	월간
	전국도시공원정보표준데이터	CSV	비정기
	노선별 지하철역 위치 정보	JSON	수시
	서울도로데이터	SHP	월간
	서울시 수치표고모델(DEM)	TIF	비정기
	공영주차장 위치 정보	JSON	수시

2.2.2. 데이터 목록 상세 설명

1) 서울시 고령자 현황 (동별) 통계[.CSV]

❑ 서울 열린데이터 광장에서 제공하는 데이터로, 행정동별 전체인구, 65세 이상 인구 수에 대한 정보로 구성됨

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	기간	자치구	동	전체인구	전체인구	전체인구	65세이상	65세이상	65세이상	65세이상	65세이상	65세이상	65세이상	65세이상	65세이상 인구
2	기간	자치구	동	전체인구	전체인구	전체인구	합계	합계	합계	내국인	내국인	내국인	외국인	외국인	외국인
3	기간	자치구	동	계	남자	여자	계	남자	여자	계	남자	여자	계	남자	여자
4	2021.2/4	합계	합계	9,795,426	4,756,178	5,039,248	1,576,971	698,990	877,981	1,568,769	694,646	874,123	8,202	4,344	3,858
5	2021.2/4	종로구	소계	155,106	75,009	80,097	27,605	12,334	15,271	27,386	12,206	15,180	219	128	91

2) 행정동별 서울생활인구 (내국인)[.CSV]

❑ 국가통계포털에서 제공하는 데이터로, 행정동 코드와 각 시간대의 연령층별 생활인구에 대한 데이터로 구성됨 (여자 컬럼도 동일한 형태)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	기준일ID	시간대구분	자치구코드	총생활인구	남자0세부	남자10세부	남자15세부	남자20세부	남자25세부	남자30세부	남자35세부	남자40세부	남자45세부	남자50세부	남자55세부	남자60세부	남자65세부	남자70세
2	20210731	0	11110	199416.2	4808.098	2891.757	4556.8	8047.396	8578.714	6950.262	7887.273	7442.993	8944.923	8094.926	8397.225	6179.214	4521.47	9189.404
3	20210731	0	11140	170495.8	3839.075	1541.467	2216.78	5613.007	7963.798	8118.392	10065.26	8067.325	7953.839	6799.892	6446.156	4953.252	3522.053	6808.2
4	20210731	0	11170	250019.6	7066.924	3513.787	5348.386	7674.531	9944.953	11455.46	13722.04	11113.92	11126.56	9446.039	8924.528	6801.531	4987.21	9321.719
5	20210731	0	11200	313111.3	10686.78	4502.107	6454.701	9777.792	12939.37	12641.87	15600.44	12888.95	13301.14	11012.1	11923.75	8822.617	6166.104	12257.25

3) 서울특별시 노인여가복지시설 목록[.CSV]

❑ 서울 열린데이터 광장에서 제공하는 데이터로, 시설명, 시설유형, 시설종류, 시설장명의 정보로 구성됨

	A	B	C	D	E	F	G	H	I
1	시설명	시설코드	시설종류명(시설유형)	시설종류상세명(시설종류)	자치구(시)구분	시설장명	시군구코드	시군구명	시설주소
2	성북구립 상월곡실버복지센터	A0495	(노인) 노인복지관(소규모)	노인여가복지시설	자치구	김경희	1129000000	성북구	서울특별시 성북구 화랑로18길 6 (상월곡동)
3	서초구립양재노인종합복지관	A0724	(노인) 노인복지관	노인여가복지시설	자치구	전경아	1165000000	서초구	서울특별시 서초구 강남대로30길 73-7양재노인종합복지관 (양재동)
4	강서구립봉제산노인복지센터	A0922	(노인) 노인복지관(소규모)	노인여가복지시설	자치구	서순애	1150000000	강서구	서울특별시 강서구 초록마을로15길 12
5	시립중랑노인종합복지관	A1022	(노인) 노인복지관	노인여가복지시설	자치구	조희정	1126000000	중랑구	서울특별시 중랑구 겸재로9길 45(면목동)

4) 서울특별시 경로당 현황(2020)[.XLSX]

❑ 서울 열린데이터 광장에서 제공하는 데이터로, 사업장명, 소재지 전체주소, 인허가일자, 영업상태명, 업소 정원 등의 정보로 구성됨

	A	B	C	D	E	F
1	연번	경로당 명	주 소	회원수	경로당 개설일	비고 (휴지/휴지시작연도)
2	1	청 운	종로구 필운대로 86 1층(신교동)	22	1997	
3	2	세종마을	종로구 자하문로11길33(통인동)	49	2019	
4	3	사 직	종로구사직로9길 15-25(사직동)	70	1993	
5	4	북 정	종로구 삼청로4길 22(삼청동)	42	1974	
6	5	삼 청	종로구 삼청로5길 30(팔판동)	29	1983	

5) 서울특별시 노선별 정류소 정보[.XLSX]

❑ 서울 열린 데이터광장에서 제공하는 데이터로, 노선 별로 정차하는 정류소 명과 정류소 좌표 등으로 구성됨

	A	B	C	D	E	F	G	H
1	노선ID	노선명	순번	NODE_ID	ARS-ID	정류소명	X좌표	Y좌표
2	100100124	0017	1	102000271	03689	청암자이아파트	126.9465174884	37.5343626071
3	100100124	0017	2	102000204	03298	청암동강변삼성아파트	126.9493037945	37.5339606747
4	100100124	0017	3	102000227	03321	청심경로당	126.9504485667	37.5337439118
5	100100124	0017	4	102000210	03304	원효2동주민센터	126.9509043881	37.5342784061

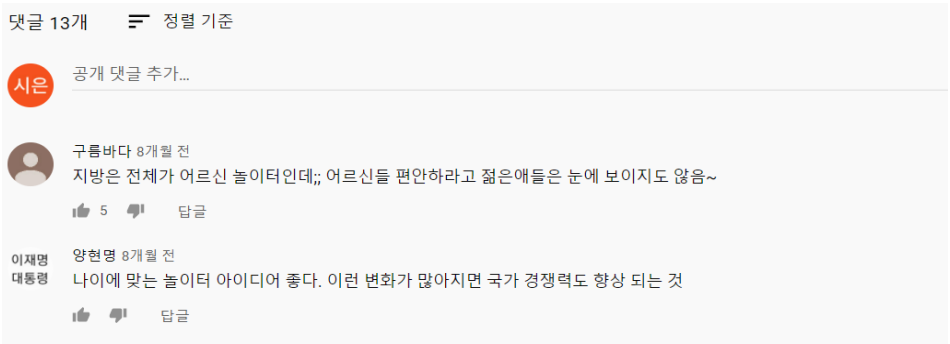
6) 서울시 주택관리 현황[.XLSX]

❑ 서울 열린 데이터광장에서 제공하는 데이터로, 서울시 주택 단지의 단지명, 주택유형, 세대 수 등에 대한 정보로 구성됨

	A	B	C	D	E	F	G	H
1	구명	단지명	주택유형	세대수	입주개시일	주소	우편번호	전화번호
2	강남구	대치1	영구임대	1,623	1992-01-15	개포로109	6335	
3	강남구	수서1-1단	영구,공공	2,214	1992-11-01	양재대로5	6341	
4	강남구	수서6	영구임대	1,508	1992-12-01	광평로56길	6368	
5	강남구	신사삼지리	재건축	63	2009-03-20	압구정로2	6027	

7) SNS와 뉴스 댓글(유튜브, 다음 뉴스)

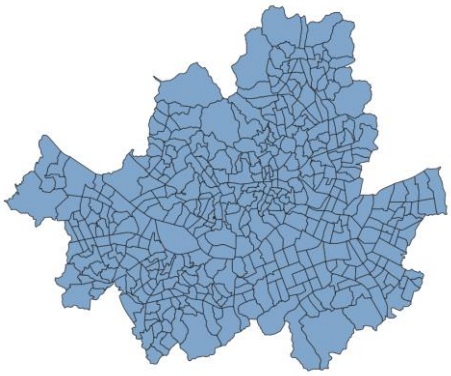
❑ 2017년부터 최근까지 노인 놀이터를 검색했을 때 뜨는 관련 동영상과 뉴스에 달린 댓글로 구성됨



<그림 5> '노인 놀이터'에 대한 유튜브 댓글

8) 센서스용 행정구역경계(읍면동) [.SHP]

❑ 통계지리정보서비스에서 제공하는 데이터로, 서울시의 행정동 경계값을 담고 있음



<그림 6> 서울시 행정구역경계.shp

9) 행정구역코드[.XLSX]

❑ 행정안전부에서 제공하는 데이터로, 행정동코드, 시도명, 읍면동명 등의 정보를 담고 있음

	A	B	C	D	E	F
1	행정동코드	시도명	시군구명	읍면동명	생성일자	말소일자
2	1100000000	서울특별시			19880423	
3	1111000000	서울특별시	종로구		19880423	
4	1111051500	서울특별시	종로구	청운효자동	20081101	
5	1111053000	서울특별시	종로구	사직동	19880423	

10) 전국체육시설현황 데이터[.CSV]

❑ 문화 빅데이터 플랫폼에서 제공하는 전국 체육시설에 대한 데이터로 시설명, 구분, 주소, 면적, 좌표 등의 정보가 담겼으며, 그 중 서울특별시의 데이터를 사용함

	A	B	C	D	E	F	G
1	자치구	분류	시설명	주소	위도	경도	전화번호
2	중랑구	구기체육관	묵동다목적체육관	중랑구 숙선옹주로 66	37.61442	127.0842	949-5577
3	중랑구	골프연습장	중랑청소년수련관 골프연습장	중랑구 용마산로 217	37.57317	127.0858	490-0114
4	중랑구	수영장	망우청소년수련관 수영장	중랑구 송림길 156	37.60588	127.1088	492-7942
5	중랑구	수영장	중랑문화체육관 수영장	중랑구 사가정로72길 47	37.57924	127.0959	436-9200

11) 전국도시공원정보표준데이터[.CSV]

❑ 공공 데이터 포털에서 제공하는 전국 도시공원에 대한 데이터로 공원명, 구분, 주소, 면적, 좌표 등의 정보가 담겼으며, 그 중 서울특별시의 데이터를 사용함

	A	B	C	D	E	F	G	H
1	관리번호	공원명	공원구분	소재지도로	소재지지번	위도	경도	공원면적
2	11680-000	학동공원	어린이공원		서울특별시	37.51165	127.0388	1498.5
3	11680-000	신사은행나	어린이공원		서울특별시	37.52111	127.0216	1240.1
4	11680-000	신사목련공	어린이공원		서울특별시	37.5229	127.0171	1001.1
5	11680-000	신사무궁화	어린이공원		서울특별시	37.52413		1641.8

12) 노선별 지하철역 위치 정보[.JSON]

❑ 카카오 API의 기능 중 키워드로 장소 검색을 이용하여 서울시 안에 있는 지하철역 위치 정보를 가져와 사용함

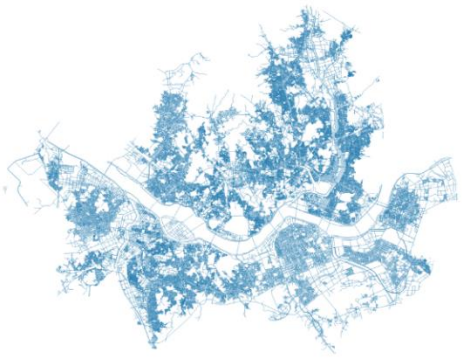
'교통,수송 > 지하철,전철 > 수도권5호선', 'distance': '', 'id': '21160840', 'phone': '02-6311-5331', 'place_name': '광화문역 5호선'
'교통,수송 > 지하철,전철 > 수도권3호선', 'distance': '', 'id': '21160823', 'phone': '02-6110-3271', 'place_name': '경복궁역 3호선'
'교통,수송 > 지하철,전철 > 수도권3호선', 'distance': '', 'id': '21160638', 'phone': '02-6110-3261', 'place_name': '독립문역 3호선
통,수송 > 지하철,전철 > 수도권5호선', 'distance': '', 'id': '21160751', 'phone': '02-6311-5321', 'place_name': '서대문역 5호선', '
,수송 > 지하철,전철 > 수도권1호선', 'distance': '', 'id': '21160570', 'phone': '02-6110-1321', 'place_name': '시청역 1호선', 'plac
교통,수송 > 지하철,전철 > 수도권1호선', 'distance': '', 'id': '21160613', 'phone': '02-6110-1311', 'place_name': '종각역 1호선', '
'교통,수송 > 지하철,전철 > 수도권3호선', 'distance': '', 'id': '21160552', 'phone': '02-6110-3281', 'place_name': '안국역 3호선',

13) 서울도로데이터[.SHP]

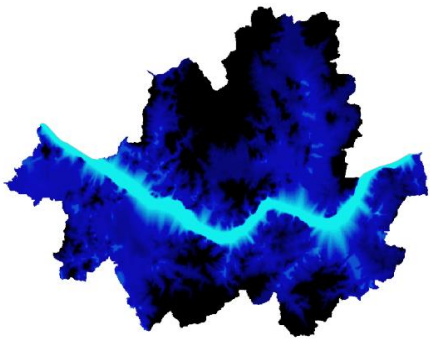
❑ 국가정보공간포털에서 제공하는 데이터로 서울시의 도로 이름, 도로 폭, 길이 등에 대한 정보가 담겨있음

14) 서울시 수치표고모델[.TIF]

❑ 국가공간정보포털에서 제공하는 서울시의 고도 정보를 담고 있음



<그림 7> 서울도로데이터.shp



<그림 8> 서울시 수치표고모델.tif

15) 공영주차장 위치 정보[.JSON]

❑ 카카오 API의 기능 중 키워드로 장소 검색을 이용하여 서울시 안에 있는 공영 주차장 정보를 가져와 사용함

'phone': '02-2290-6566', 'place_name': '세종로공영주차장', 'place_url': 'http://place.map.kakao.com/20590715',
'phone': '02-2264-4895', 'place_name': '적선노외공영주차장', 'place_url': 'http://place.map.kakao.com/26535499',
81', 'phone': '', 'place_name': '사직공영주차장', 'place_url': 'http://place.map.kakao.com/2006105181', 'road_ad
place_name': '센터포인트 주차장', 'place_url': 'http://place.map.kakao.com/26864776', 'road_address_name': '',
'', 'place_name': '콘코디언빌딩 주차장', 'place_url': 'http://place.map.kakao.com/590385661', 'road_address_name
02-724-0234', 'place_name': '서울역사박물관 주차장', 'place_url': 'http://place.map.kakao.com/20517644', 'road_a
place_name': '토속촌유료주차장', 'place_url': 'http://place.map.kakao.com/23289330', 'road_address_name': '서울
723-1942', 'place_name': '꼬레아주차장', 'place_url': 'http://place.map.kakao.com/561641169', 'road_address_name
'place_name': '삼원주차장', 'place_url': 'http://place.map.kakao.com/716781258', 'road_address_name': '', 'x':
, 'place_name': '백송주차장', 'place_url': 'http://place.map.kakao.com/20516505', 'road_address_name': '서울 종

2.3. 데이터 정제 방안

2.3.1. [크롤링] 댓글 워드 클라우드

❑ '노인 놀이터' 키워드에 관한 SNS와 뉴스 댓글을 수집함

- '노인 놀이터'에 대한 게시물과 댓글이 적었기 때문에 그 중 댓글 데이터를 찾을 수 있었던 사이트로 SNS - 유튜브, 뉴스 - 다음을 선정함

❑ raw 데이터

- 이름: SNS와 뉴스 댓글(유튜브, 다음 뉴스)
- 출처: 유튜브, 다음 뉴스

❑ 정제 과정

1) 댓글 데이터 token화

- 약 200개 댓글 데이터를 크롤링하여 csv 파일로 저장함
- 유튜브와 다음 뉴스 댓글 문장들을 띄어쓰기 및 단어 단위로 token 화함

2) 주요 키워드와 의미 없는 데이터 전처리

- token화 된 데이터 SejongDic를 활용하여 명사만 추출한 뒤 2글자 이상의 글자들만 count함

노인	놀이터	아이들	우리	들이	어르신	미래	생각	운동	운동기구
98	90	52	35	22	22	21	18	18	17

<그림 9> 댓글 데이터 전처리 전

- 검색 키워드였던 '노인', '놀이터'와 '들이', '무엇', '누구', '우리' 등

의미 없는 데이터는 삭제함

아이들	어르신	미래	생각	운동	사회	운동기구	공원	나라	아파트	
44	22	21	18	18	17	17	15	15	15	

<그림 10> 댓글 데이터 전처리 후

3) 워드 클라우드로 시각화

- wordcloud 와 RColorBrewer library를 활용하여 적절한 scale, min.freq, color 등의 파라미터를 설정하여 워드 클라우드로 시각화 함

<그림 11> 전처리 전 wordcloud 모습

<그림 12> 전처리 후 wordcloud 모습

2.3.2. [행정동 기준 데이터[서울시 읍면동 코드 매핑

❑ 가장 최근 데이터가 2020년 6월 기준이므로 2021년 6월 기준인 행정동 코드와 일치시키기 위한 작업을 수행함

❑ raw 데이터

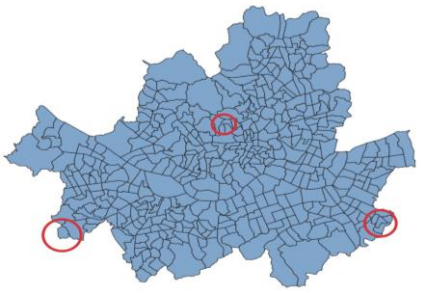
- 이름: 센서스용 행정구역경계(읍면동).shp
- 출처: 통계지리정보서비스

❑ 정제 과정

1) 2021년 6월 기준 425개 읍면동 데이터로 업데이트함

- 오류2동 분리 후 항동 추가, 거여2동 분리 후 위례동 추가함

- 명륜3가동 → 가회동에 포함함
- 신당1동 → 신당동, 신당2동 → 다산동, 신당3동 → 약수동, 신당4동 → 청구동, 신당6동 → 동화동, 공릉1.3동 → 공릉1동 읍면동 이름 및 코드를 변경함
- 혜화동, 답십리1동, 오류2동, 장지동 읍면동 코드를 수정함



<그림 13> 최종 서울 읍면동(4326).shp

	A	B	C
1	읍면동코드	읍면동	읍면동_영어
2	1101053	사직동	Sajik-dong
3	1101054	삼청동	Samcheong-dong
4	1101055	부암동	Buam-dong
5	1101056	평창동	Pyeongchang-dong
6	1101057	무악동	Muak-dong
7	1101058	교남동	Gyonam-dong
8	1101060	가회동	Gahoe-dong
9	1101061	종로1·2·3·4가동	Jongno 1·2·3·4-ga-dong
10	1101063	종로5·6가동	Jongno 5·6-ga-dong

<그림 14> 최종 읍면동(4326).csv

2.3.3. [행정동 기준 데이터[서울시 행정동 코드 매핑

❑ 모든 데이터를 행정동별로 구분할 수 있도록 기준이 되는 데이터로써 읍면동과 행정동 코드를 병합하여 사용함

❑ raw 데이터

- 이름: 행정구역코드.xlsx
- 출처: 통계지리정보서비스

❑ 정제 과정

1) 2021년 6월 데이터로 수정된 읍면동, 행정동 데이터 병합

- 최종 서울 읍면동(4326).csv와 행정구역코드.xlsx의 시군구, 행정동(읍면동) 컬럼을 merge하여 데이터 무결성 확인 및 수정함 (예: 번제1동 →번1동)

	A	B	C	D
1	행정동코드	시도명	시군구명	읍면동명
2	11110530	서울특별시	종로구	사직동
3	11110540	서울특별시	종로구	삼청동
4	11110550	서울특별시	종로구	부암동
5	11110560	서울특별시	종로구	평창동

	A	B	C	D
1	시군구	행정동	행정동코드	읍면동코드
2	종로구	사직동	11110530	1101053
3	종로구	삼청동	11110540	1101054
4	종로구	부암동	11110550	1101055
5	종로구	평창동	11110560	1101056

2.3.4. [입지후보군[체육시설

❑ raw 데이터

- 이름: 전국체육시설현황 데이터
- 출처: 문화 빅데이터 플랫폼

❑ 정제 과정

1) 서울 특별시 데이터 추출

- ‘전국’ 체육시설에 대한 데이터이므로 이 중 분석단위인 서울에 대한 데이터만 추출함

2) 열(columns) 정리

- 분석에 필요 없다고 판단되는 열을 지운 뒤 ‘분류’, ‘시설명’, ‘x좌표’, ‘y좌표’ 열만 가져와 중복값 처리, 결측값 처리 등 기본적인 전처리를 진행함

3) 행정동 열(columns) 추가

- raw 데이터의 주소 정보가 정확하지 않아 카카오 API를 이용해 좌표값을 바탕으로 주소를 새로 받아오고 행정동 열을 추가함

2.3.5. [입지후보군[공원

❑ raw 데이터

- 이름: 전국체육시설현황 데이터
- 출처: 문화 빅데이터 플랫폼

❑ 정제 과정

1) 서울 특별시 데이터 추출

- 역시 '전국' 공원에 대한 데이터이므로 이 중 분석단위인 서울에 대한 데이터만 추출함

2) 열(columns) 정리

- 분석에 필요 없다고 판단되는 열을 지운 뒤 '공원구분', '공원명', '경도', '위도' 열만 가져와 기본적인 전처리를 진행함

3) 행정동 열(columns) 추가

- raw 데이터의 주소 정보가 정확하지 않아 카카오 API를 이용해 좌표값을 바탕으로 주소를 새로 받아오고 행정동 열을 추가해 공공 체육시설 데이터와 양식을 맞춤

4) 입지후보군 데이터 병합

- 위에서 정제한 공공 체육시설 데이터와 공원데이터를 병합한 결과 총 2757개의 입지후보군 데이터가 완성됨
- 이때 공공 체육시설 1005개, 공원 데이터 1752개로 이루어짐

	A	B	C	D	E	F	G
1		분류	시설명	법정동	행정동	행정동_경도	행정동_위도
2	0	간이운동장	돌 뫼	석관동	석관동	127.061437	37.6129824
3	1	간이운동장	동 방	장위동	장위1동	127.0437373	37.61404624
4	2	간이운동장	동 산	하월곡동	월곡1동	127.0357756	37.61055309
5	3	간이운동장	밤 골	하월곡동	월곡1동	127.0357756	37.61055309

2.3.6. [노인비율] 65세 이상 노인거주 인구수

❑ 노인 비율 측정 지표 첫번째로 노인의 기준을 법적 기준인 65세 이상으로 잡은 뒤, 서울시 행정동별 거주 인구 수와 비율을 계산함

❑ raw 데이터

- 이름: 서울시 고령자현황 (동별) 통계.csv
- 출처: 서울열린데이터광장

❑ 정제 과정

1) 컬럼 추출

- 남녀 구분 없이 내국인 데이터만 추출함

2) 컬럼 추가

- 65세 이상 거주 인구/전체 거주 인구를 계산하여 65세 이상 거주 인구 비율 컬럼을 추가함

3) 행정동 컬럼 확인

- 2021년 6월 행정동을 기준으로 데이터 무결성을 확인함

	A	B	C	D	E
1	자치구	동	전체인구	거주인구수	거주인구비율
2	종로구	사직동	9813	1834	0.1869
3	종로구	삼청동	2803	641	0.2287
4	종로구	부암동	10069	1763	0.1751
5	종로구	평창동	18491	3407	0.1843

2.3.7. [노인비율] 노인 생활인구 수(시간대별)

❑ 노인 비율 측정 지표 두번째로 노인의 기준을 법적 기준인 65세 이상으로 잡은 뒤, 서울시 행정동별 시간대별 생활 인구 수와 비율을 계산함

❑ raw 데이터

- 이름: 행정동별 서울생활인구(내국인).csv
- 출처: 서울열린데이터광장

❑ 정제 과정

1) 열(column) 추출

- 행정동별 서울생활인구수 데이터에서 고령인구를 중심으로 살펴보기

위해 65세 이상의 인구로 정제 작업을 진행함

2) 행(row) 추출

- 2020년 8월부터 2021년 7월까지 최근 1년 데이터만 추출하여 평균값을 계산함

3) 행정동 컬럼 확인

- 2021년 6월 행정동을 기준으로 데이터 무결성을 확인함

4) 열(column) 추가

- 65세 이상 생활인구수/전체 생활인구수를 하여 65세이상 생활 인구 비율 열을 추가함

5) 시간대 구분

- 시간대별로 노인 생활인구수를 파악하기 위하여 시간대를 절기학*에 따라 새벽, 아침, 낮, 밤, 저녁으로 구분하여 평균값을 계산함

* 절기학에서 0~5시는 새벽, 5~9시는 아침, 9~17시는 낮, 17~21시는 저녁, 21~24시는 밤을 의미함

	A	B	C	D	E	F	G	H	I	J	K	L
1	행정동코드	행정동	새벽생활인구수	새벽생활인구비율	아침생활인구수	아침생활인구비율	낮생활인구수	낮생활인구비율	저녁생활인구수	저녁생활인구비율	밤생활인구수	밤생활인구비율
2	11110530	사직동	875.8710102	0.062637544	1104.030338	0.056339828	1437.354717	0.042983869	1144.439123	0.043178746	922.2042533	0.055170903
3	11110540	삼청동	298.8502078	0.067798609	319.4143924	0.062407779	358.0212188	0.047103993	279.7014002	0.049051517	269.7568445	0.063469419
4	11110550	부암동	1035.308758	0.066652406	1028.421995	0.066544525	925.1506194	0.064248383	975.8579412	0.067551528	1048.022034	0.067685629
5	11110560	평창동	1379.958999	0.068910249	1439.653999	0.070473776	1406.473729	0.073160791	1373.420901	0.073164181	1403.274736	0.070445722

2.3.8. [시설 위치[노인 여가복지시설,aqt

❑ 노인시설 지표 중 첫번째로, 여가복지시설에는 복지관, 시니어아카데미, 노인교실 등이 존재하며, 행정동별 개수를 계산함

❑ raw 데이터

- 이름: 서울특별시 노인 여가복지시설 목록
- 출처: 서울열린데이터광장

❑ 정제 과정

1) 경위도 좌표값 추가

- 주소값의 이상치 값을 변경하여 올바른 주소로 변경한 후, 카카오 API를 이용하여 각 단지명에 경도와 위도 값을 추가함

2) 행정동 열(column) 추가

- 다른 데이터와 행정동을 기준으로 매핑하여 분석을 진행하기 위하여 본 데이터도 Q-gis에서 좌표값을 이용해 행정동 정보를 추가함

<그림 15> 노인여가복지시설.shp

3) 행정동별 시설 개수

- -행정동별로 노인여가복지시설 개수를 구함

2.3.9. [시설 위치[경로당

❑ 노인시설 지표 중 두번째로, 행정동별 경로당 개수를 계산함

❑ raw 데이터

- 이름: 서울특별시 경로당 현황(2020).xlsx
- 출처: 서울열린데이터광장

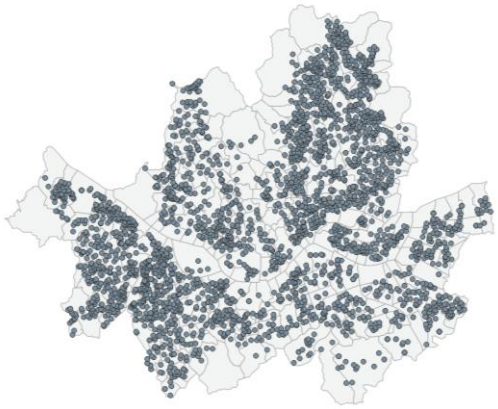
❑ 정제 과정 (여가복지시설과 동일)

1) 경위도 좌표값 추가

- 주소값의 이상치 값을 변경하여 올바른 주소로 변경한 후, 카카오 API를 이용하여 각 단지명에 경도와 위도 값을 추가함

2) 행정동 열(column) 추가

- 다른 데이터와 행정동을 기준으로 매핑하여 분석을 진행하기 위하여 본 데이터도 Q-gis에서 좌표값을 이용해 행정동 정보를 추가함



<그림 16> 경로당.shp

3) 행정동별 시설 개수

- 행정동별로 경로당 개수를 구함

4) 행정동 별 총 노인 시설 개수

- 노인여가복지시설과 경로당 수를 합쳐 행정동 별 총 노인 시설 개수 데이터를 만듦

	A	B	C	D	E
1	읍면동코드	읍면동	경로당수	복지시설수	총노인시설수
2	1101053	사직동	0	0	0
3	1101054	삼청동	2	0	2
4	1101055	부암동	1	0	1
5	1101056	평창동	6	0	6

2.3.10. [지역주민 접근성] 버스

❑ 접근성을 측정 지표 중 첫번째로 서울특별시 행정동별 버스 정류장에 정

류하는 버스 수를 계산함

❑ raw 데이터

- 이름: 서울특별시 버스노선별 정류소 정보.xlsx
- 출처: 서울열린데이터광장

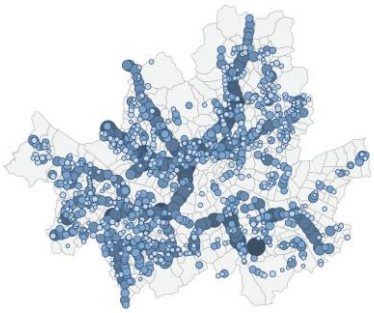
❑ 정제 과정

1) 정류소별 정차하는 버스 개수

- 서울특별시 노선별 정류소 정보 데이터에서 정류소명을 기준으로 해당 정류소를 지나가는 정류버스의 수를 계산함

2) 행정동 열(column) 추가

- 다른 데이터와 행정동을 기준으로 매핑하여 분석을 진행하기 위하여 본 데이터도 Q-gis에서 좌표값을 이용해 행정동 정보를 추가함



<그림 17> 버스.shp

	A	B	C	D	E	F	G
1	정류소명	X좌표	Y좌표	정류버스수	행정동	행정동_경도	행정동_위도
2	서울역버스	126.9728	37.55548	58	회현동	126.9793567	37.55729359
3	현대아파트	127.0529	37.48228	55	개포4동	127.0516222	37.47885718
4	노량진역	126.9447	37.51389	52	노량진1동	126.9420162	37.5123068
5	연희104고	126.9258	37.56636	52	연희동	126.9352308	37.57391001
6	미아사거리	127.03	37.61377	48	송천동	127.023868	37.6183082
7	신도림역	126.8894	37.50976	46	신도림동	126.8806102	37.50778091
8	종로2가	126.9855	37.57009	45	종로1.2.3.4	126.9902873	37.57443554
9	논현역	127.023	37.50722	45	반포1동	127.0133771	37.50509061
10	갈월동	126.9717	37.54953	43	남영동	126.973368	37.54284398

<그림 18> 버스.csv

2.3.11. [지역주민 접근성] 지하철

❑ 접근성을 측정 지표 중 두번째로 서울특별시 행정동별 지하철 개수를 환승역에 가중치를 두어 계산함

❑ raw 데이터

- 이름: 노선별 지하철역 위치 정보.json
- 출처: 카카오API

❑ 정제 과정

1) 서울시 행정동별 지하철역의 위치 데이터 수집

- 공공데이터는 1호선~4호선 데이터만 제공함
- 카카오 API의 키워드로 장소 검색을 이용하여 전철역명, 전철역코드(id), 주소 등의 데이터를 직접 수집함

```
1 result = getLatLng()
```

```
사직동   7 개 주차장 데이터 수집  완료 7
삼청동   13 개 주차장 데이터 수집  완료 20
부암동   9 개 주차장 데이터 수집  완료 29
```

2) 환승역과 일반역 구분

- 노선이 겹칠 경우 환승역=1, 아닐 경우=0으로 코딩하여 환승역 데이터를 추가함

```
1 환승역 = data8.groupby(['전철역명'])['검색키워드'].count().to_frame()
2 환승역 = 환승역[환승역['검색키워드'] >= 2].drop('신촌역', axis=0)   # 신촌역은 환승역이라고 볼 수 없으므로 제외
```

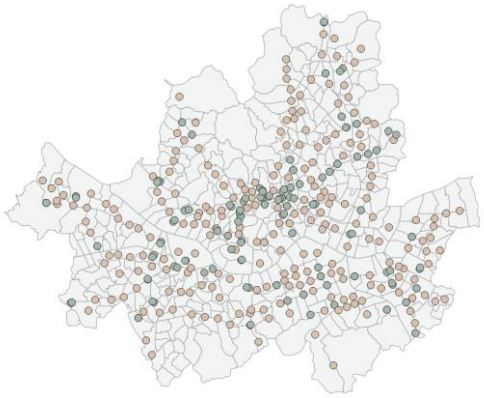
```
1 # 환승역이면 1, 아니면 0
2 data8['환승역'] = 0
3 for i in range(len(data8)):
4     if data8['전철역명'][i] in 환승역.index:
5         data8['환승역'][i] = 1
```

```
1 data8.head()
```

	전철역코드	전철역명	전철명명(영문)	호선	외부코드	검색키워드	주소	위도	경도	환승역
0	2622	망원역	Mangwon	6호선	621	망원역 6호선	서울 마포구 월드컵로 지하 77	37.5560572923314	126.910033925408	0
1	2623	합정역	Hapjeong	6호선	622	합정역 6호선	서울 마포구 양화로 지하 45	37.5491226824276	126.913544566078	1
2	2624	상수역	Sangsu	6호선	623	상수역 6호선	서울 마포구 독막로 지하 85	37.5477716665856	126.922409857908	0
3	2625	광흥창역	Gwangheungchang	6호선	624	광흥창역 6호선	서울 마포구 독막로 지하 165	37.5474860706906	126.931939240491	0
4	2626	대흥역	Daeheung	6호선	625	대흥역 6호선	서울 마포구 대흥로 지하 85	37.5476683544726	126.942465251709	0

3) 행정동 열(column) 추가

- 다른 데이터와 행정동을 기준으로 매핑하여 분석을 진행하기 위하여 본 데이터도 Q-gis에서 좌표값을 이용해 행정동 정보를 추가함



<그림 19> 지하철.shp: 환승역=초록색, 일반역=연갈색

4) 환승역 가중치 부여

- 환승역일 경우 교통이 편리하다는 점을 고려하여 환승역이 있는 경우는 2배로 계산하여 행정동별로 지하철의 유무와 환승역*2로 지하철의 접근성을 계산함

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	field_1	전철역코드	전철역명	전철명명(E	호선	외부코드	검색키워드	주소	위도	경도	환승역	H/B	행정동	행정동_경.	행정동_위	행정동별_	행정동별_환승역가중치부여(환승역*2)
2	142	340	가락시장역	Garak Mar	3호선	350	가락시장역	서울 송파-	37.49274	127.1185	1	H	가락1동	127.1073	37.49975	2	5
3	355	2817	송파역	Songpa	8호선	816	송파역 8호	서울 송파-	37.49969	127.1122	0	H	가락1동	127.1073	37.49975	2	5
4	356	2818	가락시장역	Garak Mar	8호선	817	가락시장역	서울 송파-	37.49274	127.1185	1	H	가락1동	127.1073	37.49975	2	5
5	207	2559	개롱역	Gaerong	5호선	P553	개롱역 5호	서울 송파-	37.49781	127.1352	0	H	가락2동	127.1266	37.4987	0	1

2.3.12. [지역주민 접근성] 주택 세대 수

❑ 접근성을 측정 지표 중 세번째로 서울특별시 행정동별 주택단지의 세대수 평균을 계산함

❑ raw 데이터

- 이름: 서울시 주택관리 현황.xlsx
- 출처: 서울열린데이터광장

❑ 정제 과정

1) 경위도 좌표값 추가

- 주소값의 이상치 값을 변경하여 올바른 주소로 변경한 후, 카카오 API를 이용하여 각 단지명에 경도와 위도 값을 추가함

주소값 이상치 값 변경

```
# 주택
# 이상치 탐색
print(house.loc[51])
print(house.loc[102])
print(house.loc[243])
print(house.loc[338])

# 올바른 주소로 바꾸기
df.loc[51]['주소'] = '강동구 고덕로 333'
df.loc[102]['주소'] = '강서구 강서로 515'
df.loc[243]['주소'] = '동작구 사당동 177-1 구립 학수경로당'
df.loc[338]['주소'] = '서초구 잠원로 60'
```

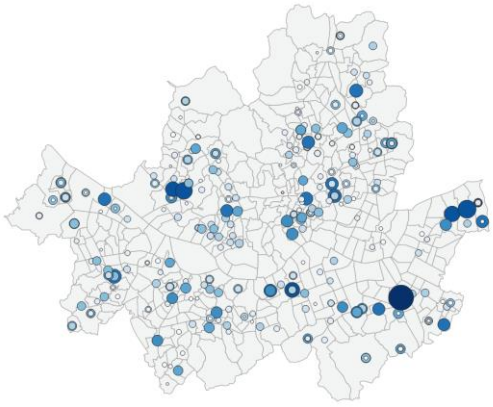
	B	C	D	E	F	G	H	I	J	K
	구명	단지명	주택유형	세대수	입주개시일	주소	우편번호	전화번호	경도	위도
	강남구	대치1	영구임대	1623	1992-01-15	개포로109	6335		37.49575	127.0751
	강남구	수서1-1단지	영구공공	2214	1992-11-01	양재대로5	6341		37.49315	127.091
	강남구	수서6	영구임대	1508	1992-12-01	광평로56길	6368		37.48798	127.1056
	강서구	보람더하임	재건축	202	2008-04-01	양천로 66(	7554		37.55152	126.8705
	강남구	신사삼지래미안	재건축	63	2009-03-20	압구정로2	6027		37.5201	127.0196

<그림 20> 변경된 주소값

<그림 21> 주택_좌표추가.csv

2) 행정동 열(column) 추가

- 다른 데이터와 행정동을 기준으로 매핑하여 분석을 진행하기 위하여 본 데이터도 Q-gis에서 좌표값을 이용해 행정동 정보를 추가함



<그림 22> 주택.shp

	A	B	C	D	E	F	G	H	I	J	K	L
1	구명	단지명	주택유형	세대수	주소	우편번호	경도	위도	H/B	행정동	행정동_경도	행정동_위도
2	강남구	대치1	영구임대	1623	개포로109	6335	127.0751	37.49575	H	일원2동	127.0736996	37.49217318
3	강남구	수서1-1단	영구공공	2214	양재대로5	6341	127.091	37.49315	H	일원1동	127.0880692	37.49187863
4	강남구	수서6	영구임대	1508	광평로56길	6368	127.1056	37.48798	H	수서동	127.105004	37.48888786
5	강서구	보람더하은	재건축	202	양천로 66(	7554	126.8705	37.55152	H	염창동	126.8709853	37.55373624

<그림 23> 주택_행정동추가.csv

3) 세대수별 가중치 부여

- 단지마다 세대수의 차이가 있으므로 행정동별 세대수 평균을 계산함

4) 버스, 지하철, 주택 데이터 행정동 기준 병합

- 신사동과 같이 관악구와 강남구에 동시에 존재하여 중복되는 경우가 있기 때문에 행정동 이름만으로는 key값이 될 수 없음. 따라서, 행정동, 행정동_경도, 행정동_위도를 모두 key값으로 하여 병합함

- 이후 행정동별 구분을 위해 위도와 경도 컬럼 대신 행정구 컬럼을 추가함

	A	B	C	D	E
1	행정구	행정동	버스수	환승역가중치부여(환승역*2)	주택세대수
2	강남구	개포1동	80	0	1469.5
3	강남구	개포2동	108	1	3277
4	강남구	개포4동	141	0	1469.5
5	강남구	논현1동	95	1	1469.5

2.3.13. [지역주민 접근성] 도로

❑ 도로 데이터는 데이터 형식이 다르기 때문에 최종 입지 선정 후 검증 데이터로 활용함

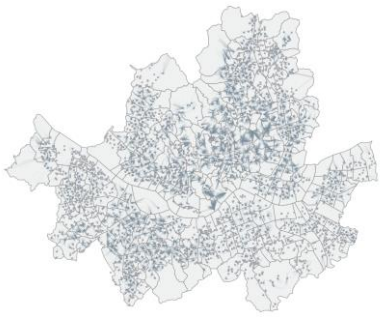
❑ raw 데이터

- 이름: 서울도로데이터.shp
- 출처: 국가공간정보포털

❑ 정제 과정

1) 도로 수, 최단거리

- Q-gis를 이용해 입지 후보 지점에서 가장 가까운 도로의 하위 도로와 연결하여 거리 및 하위 도로 개수와 도로까지의 최단거리를 계산함



<그림 24> 후보지-도로거리.shp

	A	B	C
1	후보지	연결개수	최소거리
2	홍릉	647	514955.1
3	보림	554	408208.7
4	선농단	471	261107
5	반딧불	370	450001.1

<그림 25> 후보지-도로거리.csv

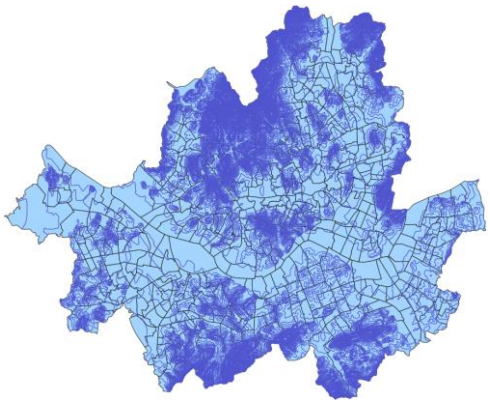
2.3.14. [이용 편리성[고도

❑ 시설의 이용 편리성을 측정하기 위해 서울시에서 제공하는 고도데이터를

활용하여 등고선으로 활용함

❑ raw 데이터

- 이름: 서울시 수치표고모델(DEM).tif
- 설명: 서울시 고도 데이터
- 출처: 국가공간정보포털



<그림 26> 등고선.shp

2.3.15. [이용 편리성[주차장

❑ 이용편리성의 두번째 지표로 행정동별 주차장 개수를 계산함

❑ raw 데이터

- 이름: 공영주차장 위치 정보.json
- 출처: 카카오 API

❑ 정제 과정

1) 서울시 행정동별 공영주차장의 위치 데이터 수집(지하철역과 동일)

2) 행정동 열(column) 추가

- 다른 데이터와 행정동을 기준으로 매핑하여 분석을 진행하기 위하여 본 데이터도 Q-gis에서 좌표값을 이용해 행정동 정보를 추가함

3) 행정동별 주차장 개수

- 행정동별로 주차장 개수를 구함

	A	B	C
1	읍면동코드	읍면동	주차장수
2	1101053	사직동	12
3	1101054	삼청동	2
4	1101055	부암동	1
5	1101056	평창동	4

3. 분석 프로세스

3.1. 분석 프로세스

전체 프로세스는 분석 방법론 'CRISP-DM'을 참고하여 '데이터 준비', '분석 및 모델링', '모델 평가 및 검증', '결과 도출 및 시각화'로 진행함

3.1.1. 데이터 준비

❑ 노인 놀이터가 입지할 수 있는 곳으로 공원 및 체육관을 선정하여 입지 후보군을 생성함

❑ 최적 입지 선정 기준을 노인 비율, 노인 시설, 접근성, 이용 편리성으로 나누어 확인함

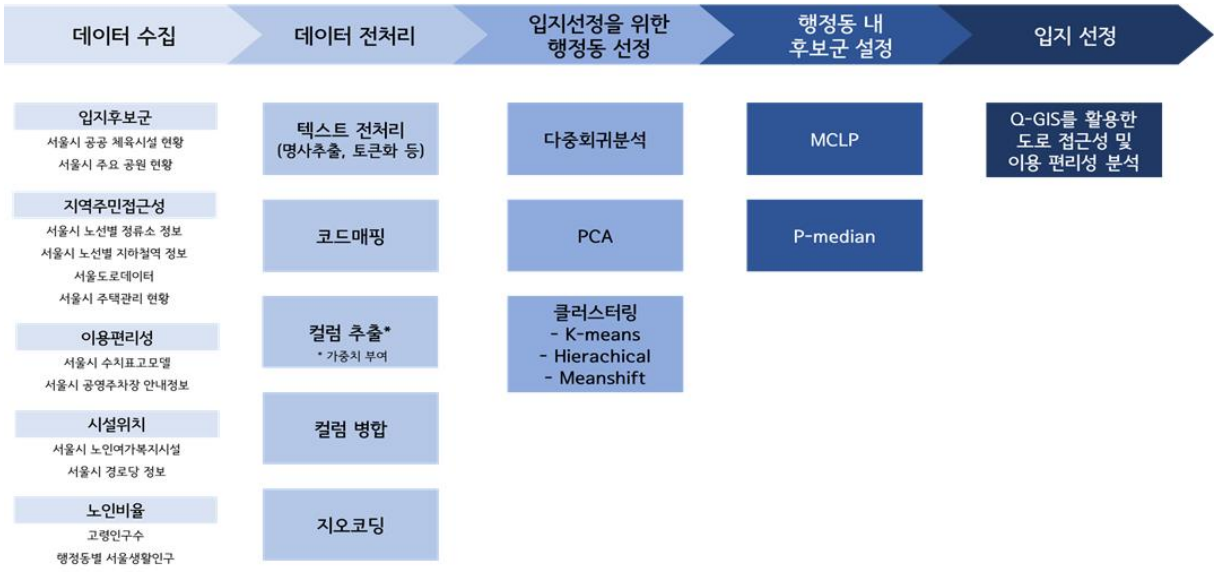
- [노인 비율] 서울시 노인 거주 인구 수/비율과 시간대별 생활인구 수/비율 데이터로 행정동별 노인 비율을 측정함
- [노인 시설] 행정동별 노인여가복지시설과 경로당 총 개수를 계산함
- [접근성 1] 행정동별 버스, 지하철, 주택 데이터를 활용하여 서울시 교통수단 이용률 별로 가중치*를 주어 접근성 지표를 계산함

  * 2014 년 자동차이용실태조사(국가교통 DB,2014) 자료 참고

- [접근성 2] 입지 후보군에서 가장 가까운 도로까지의 최단 거리와 인접한 도로 개수를 계산함

- [이용 편리성 1] 행정동별 주차장 개수로 이용 편리성을 계산함
- [이용 편리성 2] 서울시의 고도를 등고선으로 시각화하여 경사가 가파른 지역을 확인함

3.1.2. 분석 및 모델링



<그림 27> 분석 및 모델링 프로세스

- ❑ 현황 분석을 통해 노인들만 이용하는 '노인 놀이터' 대신 누구나 쉽게 이용할 수 있는 노인 친화형 다세대 놀이터 입지 선정을 계획함
- ❑ 노인 거주 인구 수/비율과 시간대별 노인 생활인구 수/비율 간의 상관분석을 실시하고 다중회귀분석을 통해 노인 비율과 노인 시설 수 간의 관계성을 파악함
- ❑ 노인 비율이 높지만 관련 시설이 부족한 행정동을 타겟으로 잡아 접근성 지표와 함께 클러스터링한 뒤 비교함
- ❑ '노인 친화형 다세대 놀이터' 입지 선정 모델 (MCLP/P-Median)을 개발하여 후보 행정동에 적용함

3.1.3. 결과 도출 및 시각화

❑ 이용 편리성을 확인하기 위하여 고도 데이터를 활용해 경사도가 심한 지역은 제외함

❑ 입지 선정 지수를 개발하여 '노인 친화형 다세대 놀이터' 서울시 최적 입지 top5 선정 후 시각화 함

3.2. 분석 내용 및 방법

3.2.1. 1차 입지 후보 행정동 선정 (다중회귀)

❑ 사용 목적

- 거주인구 수(비율)와 생활인구 수(비율)에 비해 노인시설이 작게 분포하는 행정동을 찾아 노인 놀이터 입지 후보군으로 선정하기 위함

❑ 변수 설정

- 독립변수: 행정동별 거주인구 수, 거주인구 비율, 시간대별* (새벽, 아침, 낮, 저녁, 밤) 생활인구와 생활인구비율

  * 시간대별은 절기학을 기준으로 구분 : 새벽(0~5 시), 아침(5~9 시), 낮(9~17 시), 저녁(17~21 시), 밤(21~24 시)

- 종속변수 : 행정동별 노인시설 수 (경로당 수 + 노인복지시설 수)

❑ 가설 설정

- 노인 거주 인구 수/비율이 많은 곳에 노인 시설도 많을 것임
- 노인 생활 인구 수/비율이 많은 곳에 노인 시설도 많을 것임

❑ 분석 과정

- 다중공선성 확인 : 독립 변수들 간 상관관계가 높아 다중공선성 문제를 확인함

- 단계적 변수 선택법을 통해 유의한 독립 변수 선정 : 거주인구수, 낮생활인구비율
- 다중회귀분석을 실행함
  - 독립변수 : 노인 거주인구 수, 노인 낮 생활인구 비율
  - 종속변수 : 노인시설 수

3.2.2. 2차 입지 후보 행정동 선정 (클러스터링)

❑ 변수 설정

- 회귀분석에서 활용된 거주인구 수, 낮생활인구비율, 총노인시설 수에 접근성지표(버스+지하철+주택)와 이용편리성지표(주차장)를 추가함

❑ 차원 축소

- 차원의 저주와 과적합 문제를 해결하기 위해 변수들 간의 상관 분석을 실시하여 가장 높은 값을 갖는 변수 제거하는 등 3 차원 데이터로 차원을 축소함

❑ 분석 과정

- 회귀분석 결과로 나온 행정동을 주요 타겟으로 설정함
- K-means, Hierarchical, Mean Shift Clustering 알고리즘을 사용하여 결과를 비교함
- 후보 행정동이 주로 포함된 군집 파악 후, 최종 입지 행정동으로 선정함

3.2.3. 최적입지선정 모델 개발(MCLP/P-Median)

❑ 후보지와 수요지점 정의

- 후보지: 서울시 공원 데이터와 공공 체육시설 데이터를 이용해 노인 놀이터의 입지 후보군으로 형성함
- 수요지점: 버스, 지하철, 주택, 주차장 데이터를 이용해 노인 놀이터로의 접근성을 수요지점으로 형성함

❑ 거리행렬(Distance Matrix) 생성

- 클러스터링 결과 정해진 행정동 내 후보지와 인근 행정동을 포함한 수요지점 간의 **거리행렬(Distance Matrix)** 생성함
- 이때, 지구가 구형임을 고려하여 **Haversine 공식**을 이용해 직선거리가 아닌 곡선거리로 실제 이동거리를 도출함

❑ 가중치 계산

- 평가기준을 계층화하여 요인들의 중요도를 산출하는 AHP 분석을 통해 수요지점별 **선호도***를 반영한 가중치를 계산함

  * 2016 년 한국교통연구원의 전국 여객 기종점 통행량 조사 중 서울시 통행수단 분포를 참고하여 주택, 버스, 지하철, 주차장으로 우선순위를 계층화한 후 AHP 분석을 실시함

- 수집한 데이터의 특성상 버스의 수요지점이 지하철, 주차장, 주택 데이터에 비해 약 20 배 많으므로 이러한 **데이터 불균형**을 해소하기 위한 가중치를 계산함

$$\frac{\text{전체 수요지점 개수} - \text{해당 수요지점 개수}}{\text{전체 수요지점 개수}}$$

- 통행수단 선호도에 대한 가중치와 데이터 불균형에 대한 가중치를 곱하여 **최적 후보지 선정 가중치**를 생성함

❑ MCLP

수정된 MCLP 목적함수	설명
$Maximize \sum_{i \in I} w_i y_i \quad \ldots(1)$ $\sum_{j \in N_i} x_j \geq y_i \quad \ldots(2)$ $\sum_{j \in J} x_j = K, \;\; x_j, y_i \in \{0,1\} \quad \ldots(3)$	- I : 수요지점 집합 - J : 노인놀이터 후보지 집합 - K : 설치해야 하는 노인놀이터 수 - x : $\begin{cases} 1, \text{노인놀이터가 후보지 } j\text{에 설치되는 경우} \\ 0, \text{그렇지 않은 경우} \end{cases}$ - y : $\begin{cases} 1, \text{수요지점 } i\text{가 후보지 반경 내 포함되는 경우} \\ 0, \text{그렇지 않은 경우} \end{cases}$ - w : 수요지점 i에 대한 가중치

(1) : 목적함수, 반경 내 가중화 된 수요지점을 최대한 많이 포함하는 후보지 도출
(2) : 수요지점 i는 후보지 반경 내에서 적어도 하나 이상의 후보지에 포함
(3) : 설치할 노인놀이터 K개

- 반경 내 가장 많은 수요지점을 포함하는 후보지를 최적 입지로 선정하는 **MCLP 알고리즘**에서 가중치를 반영하여 주어진 후보군 중 최적 후보지를 선정함

❑ P-Median

수정된 P-median 목적함수	설명
$Maximize \sum_{i \in I} \sum_{j \in J} w_i D_{ij} \quad \ldots(1)$ $\sum_{j \in J} D_{ij} = 1, \;\; D_{ij} \leq x_j \quad \ldots(2)$ $\sum_{j \in J} x_j = K, \;\; D_{ij}, x_j \in \{0,1\} \quad \ldots(3)$	- I : 수요지점 집합 - J : 노인놀이터 후보지 집합 - K : 설치해야 하는 노인놀이터 수 - x : $\begin{cases} 1, \text{노인놀이터가 후보지 } j\text{에 설치되는 경우} \\ 0, \text{그렇지 않은 경우} \end{cases}$ - D : $\begin{cases} 1, \text{수요지점 } i\text{가 후보지 } j\text{와 최단거리인 경우} \\ 0, \text{그렇지 않은 경우} \end{cases}$ - w : 수요지점 i에 대한 가중치

(1) : 목적함수, 가중화 된 최단거리 수요지점을 최대한 많이 포함하는 후보지 도출
(2) : 각 수요지점은 반드시 하나의 후보지에만 할당
(3) : 설치할 노인놀이터 K개

- 공간적 효율성을 고려하여 모든 수요지점과 가중화 된 거리의 합을 최소화하는 **P-Median 모델에서 착안**하여 주어진 후보군 중 최적 후보지를 선정함

❑ 최적 입지 선정

- MCLP 와 P-Median 두 모델에서 모두 최적 후보지로 선정된 지점에 대하여 최종적으로 고도 데이터와 입지선정지수를 활용하여 **서울시 노인놀이터 Top5 최적 입지를 선정함**

4. 분석 결과

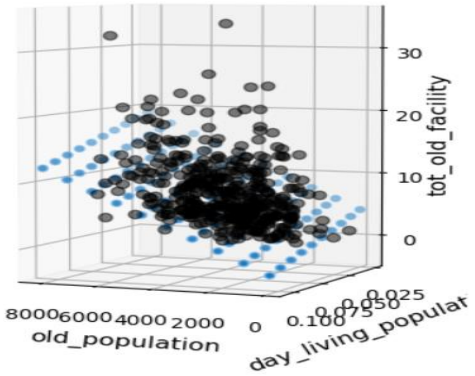
4.1. 다중회귀분석 결과

4.1.1. 단계적 변수 선택법

❑ 전체 독립변수 중 단계적 변수 선택법을 통해 유의한 독립 변수 추출 코드 실행 결과, '거주인구 수'와 '낮 생활 인구 비율'이 유의한 독립변수로 선정됨

4.1.2. 다중회귀분석

❑ 앞서 선택된 두 가지 독립변수(old_population, day_living_population_ratio)와, 경로당 수, 노인 복지시설 수, 노인교실 수를 더해 만든 '기존시설(tot_old_facility)'을 종속변수로 두어 다중회귀분석을 실행함



<그림 28> 다중회귀분석 결과 그래프

❑ 다중회귀분석 실행 결과, 수정된 결정 계수 값은 0.757 으로 모델 설명력이 뛰어난 것을 확인할 수 있었고, 회귀식은 다음과 같음

$$Y = 0.0019 * X_{\text{거주인구 수}} + 6.844 * X_{\text{낮생활인구비율}}$$

- 유의수준 5%에서 p-value 가 0.0000 으로 0.05 보다 작으므로 귀무가설을 기각함. 따라서 회귀계수 값이 양수이므로 노인 거주 인구수가 많은 곳에 노인 시설도 많음
- 유의수준 5%에서 p-value 가 0.509 로 0.05 보다 크므로 귀무가설을 채택함. 따라서 낮생활인구비율은 총 노인시설 수를 설명하지 못함
- 모델 검정 결과, 유의수준 5%에서 p-value 가 0.0000 으로 0.05 보다 작아 귀무가설을 기각하므로 모델이 유의함

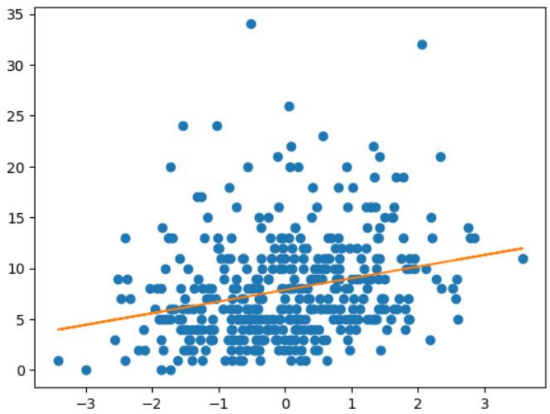
```
Call:
lm(formula = 총노인시설수 ~ 0 + 거주인구수 + 낮생활인구비율,
    data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-13.6080  -2.8008  -0.4455   2.6689  25.3880

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
거주인구수     1.960e-03  1.595e-04  12.283   <2e-16 ***
낮생활인구비율 6.844e+00  1.036e+01   0.661    0.509
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.633 on 421 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.7584,    Adjusted R-squared:  0.7573
F-statistic: 660.8 on 2 and 421 DF,  p-value: < 2.2e-16
```

4.1.3. PCA 실행



<그림 29> 차원축소 값과 기존시설 수의 산점도

- x 축은 독립변수(노인 거주인구 수, 노인 낮생활인구 비율)의 1 차원 차원축소 값, y 축은 기존시설 수를 나타냄. 회귀식을 기준으로 좌표가 다양하게 분포함을 확인할 수 있음. 이 중 회귀식 아래 좌표의 경우, 노인거주인구, 노인 낮 생활인구비율에 비해 노인 시설이 부족하다는 것을 뜻하므로, 회귀식 아래쪽으로 멀리 떨어진 점을 후보동으로 선정함
- 회귀식에서 회귀식 아래 위치한 좌표의 거리 최대값이 7.6 이므로, 회귀식과 좌표간 거리가 6 이상인 좌표 14 개를 추출하여 후보 행정동으로 선정함

4.2. 클러스터링 결과

4.2.1. 차원 축소

- 상관계수 행렬 생성 : 과적합을 방지하기 위해 독립 변수들 간에 가장 높은 상관계수를 갖는 '거주인구 수' 요인을 제거함

	거주인구수	낮생활인구비율	총노인시설수	접근성지표	이용편리지표
거주인구수	1.000000	0.325639	0.449034	0.263270	0.060493
낮생활인구비율	0.325639	1.000000	-0.028263	-0.237683	-0.109605
총노인시설수	0.449034	-0.028263	1.000000	0.248215	0.001985
접근성지표	0.263270	-0.237683	0.248215	1.000000	0.270585
이용편리지표	0.060493	-0.109605	0.001985	0.270585	1.000000

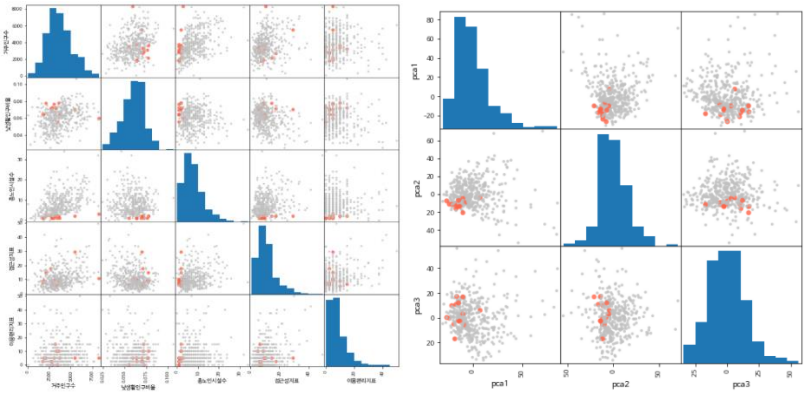
- PCA : 차원의 저주를 막기 위해 3 차원의 데이터로 차원을 축소함

	낮생활인구비율	총노인시설수	접근성지표	이용편리지표
0	22.4	0.0	20.7	60.0
1	27.5	5.9	3.2	10.0
2	48.5	2.9	14.0	5.0
3	59.4	17.6	18.6	20.0
4	52.5	5.9	1.3	0.0

	pca1	pca2	pca3
0	26.912113	-45.634861	3.936217
1	-11.742670	-17.033084	-19.474290
2	-19.277146	-11.857675	-5.197152
3	-8.597259	-4.784628	14.692669
4	-30.483787	-8.059042	-4.788926

4.2.2. 데이터 분포 및 후보행정동 위치 파악

❑ 데이터를 시각화 하여 전체 데이터에서 후보 행정동의 분포를 파악한 결과, 일정 지점에 후보 행정동이 몰려 있음을 확인함

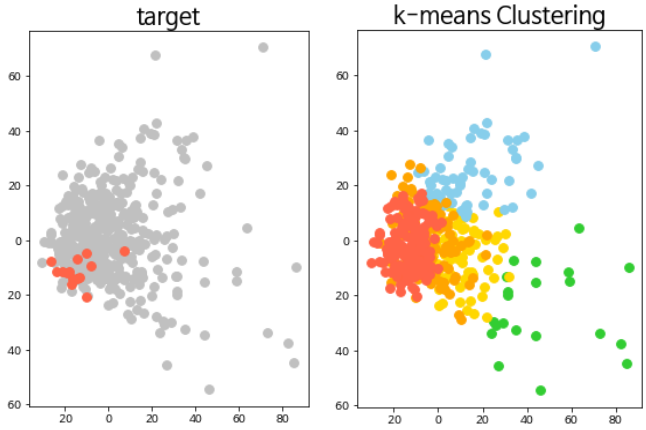


<그림 30> 행정동 분포 그래프

4.2.3. 클러스터링 결과 비교

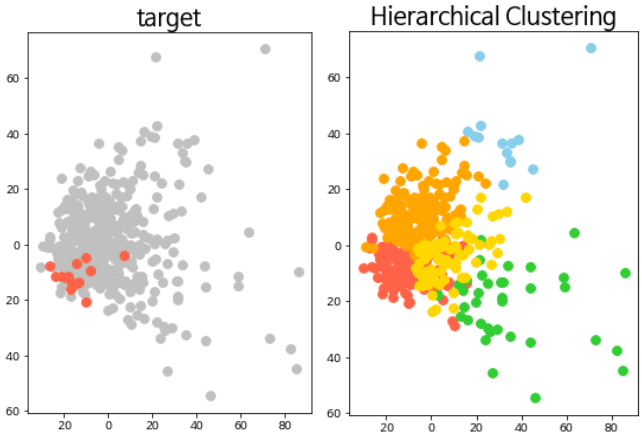
모두 5 개의 군집으로 클러스터링을 진행함

❑ K-means Clustering : 후보 행정동과 K-means Clustering 모델의 결과를 비교함. 왼쪽의 빨간색 군집과 겹치는 후보 행정동을 추출함



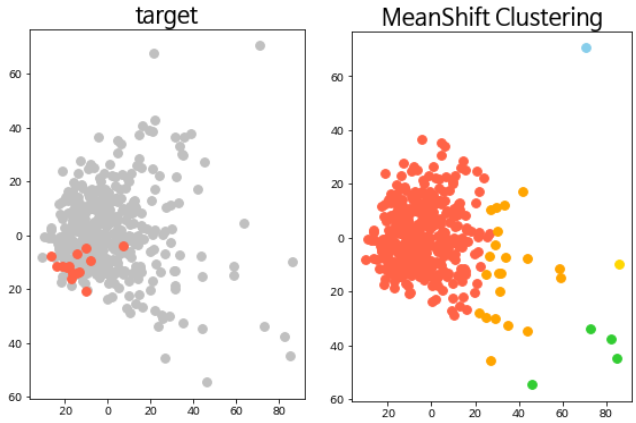
<그림 31> K-means Clustering 행정동 추출 결과

❑ Hierarchical Clustering: 후보 행정동과 Hierarchical Clustering 모델의 결과를 비교함. 왼쪽의 빨간색 군집과 겹치는 후보 행정동을 추출함



<그림 32> Hierarchical Clustering 행정동 추출 결과

❑ Mean Shift Clustering: 후보 행정동과 Mean Shift Clustering 모델의 결과를 비교함. 왼쪽의 빨간색 군집과 겹치는 후보 행정동을 추출함



<그림 33> Mean Shift Clustering 행정동 추출 결과

4.2.4. 최종 행정동 선택

❑ Clustering Model 선택 : 특정 군집에 후보 행정동을 가장 많이 포함하는 모델은 Mean Shift 이지만 해당 군집의 크기가 너무 크기 때문에 과대 적합이 발생했다고 판단하여, 그 다음으로 후보 행정동을 많이 포함하는 Hierarchical Clustering Model 을 선정함

❑ 따라서 최종 최적 입지 대상 행정동은 '**부암동, 중곡 3 동, 망우 3 동, 돈암 1 동, 장위 2 동, 역촌동, 홍제 3 동, 반포본동, 일원 1 동, 오륜동, 잠실 7 동, 성내 3 동**'으로 총 12 개임

4.3. MCLP/P-Median 결과

4.3.1. 거리행렬 생성

```
# 후보지와 수요지점 간 거리행렬 생성
havers = []
for i in 후보지_points:
    site = []
    for j in 버스_points:
        site.append(haversine(i,j)*1000)
    havers.append(site)

location = list(입지후보지['시설명'])
location2 = list(버스['정류소명'])

havers_D = dict(zip(location, [dict(zip(location2, i)) for i in havers]))
D = pd.DataFrame(havers_D)
D.head(2)
```

	역말어린이공원	연서어린이공원	자투리(녹번)마을마당	백련	신사	대조어린이공원	길마어린이공원	궁말어린이공원	호연어린이공원
녹번역	2324.833082	2040.304448	2169.371567	1768.302079	2201.173593	1837.613729	2325.923573	2095.420987	2041.596088
불광역 3.6 호선	1579.651018	1383.162101	1635.395359	991.912871	1517.369056	1307.778092	1581.678649	1309.360781	1385.581880

❑ Haversine 공식으로 후보지와 수요지점 좌표 간의 거리를 계산한 값에 1000 을 곱하여 미터(m) 단위의 거리행렬을 생성함

❑ 버스, 지하철, 주차장, 주택 수요지점에 따라 행정동별 4 개의 거리행렬이 생성되었으며, 각 행렬은 (수요지점 수) X (후보지 수) 로 구성됨

4.3.2. 가중치 부여

(1) 가중치 산정 결과

	주택	주차장	버스	지하철
	0.392	0.136	0.278	0.193

(2) 비교 행렬

	주택	주차장	버스	지하철
주택	1	2	2	2
주차장	0.5	1	0.5	0.5
버스	0.5	2	1	2
지하철	0.5	2	0.5	1

❑ 통행수단별 선호도를 반영한 가중치를 계산하기 위해 주택, 버스, 지하철, 주차장 우선순위에 따라 3 점 척도의 AHP 분석을 실시함

❑ AHP 분석을 실시한 결과, 버스, 지하철, 주차장, 주택 수요지점에 각각 0.278, 0.193, 0.136, 0.392 의 가중치가 도출됨

```
# 통행수단별 선호도에 대한 가중치 * 데이터 불균형 해소를 위한 가중치
m1 = 0.278 * (전체수요지점수-버스수요지점수) / 전체수요지점수
m2 = 0.193 * (전체수요지점수-지하철수요지점수) / 전체수요지점수
m3 = 0.136 * (전체수요지점수-주차장수요지점수) / 전체수요지점수
m4 = 0.392 * (전체수요지점수-주택수요지점수) / 전체수요지점수
```

❑ 앞서 도출한 가중치와 데이터 불균형 해소를 위한 가중치를 곱하여 각 수요지점에 대한 행정동별 최적 후보지 선정 가중치를 생성함

4.3.3. MCLP 결과

```
# 반경 내 속하면 1, 아니면 0
for i in [D1, D2, D3, D4]:
    mask1 = i<=radius
    i[mask1]=1
    i[~mask1]=0
```

❑ 앞서 생성한 수요지점의 거리행렬에서 300m* 이하인 경우 1, 그렇지 않으면 0 값을 부여하여 후보지로부터 300m 반경 내 속하는 수요지점을 구분함

* 조현주, 이순주(2019)에 따르면 보행적 측면에서 노인친화형 공원의 유치거리가 300m 로 도출됨

```
# 목적함수 수정
m.setObjective(quicksum(i for i in [m1*x1[a] for a in range(A)] + [m2*x2[b] for b in range(B)] + ₩
                        [m3*x3[c] for c in range(C)] + [m4*x4[d] for d in range(D)]), GRB.MAXIMIZE)

m.setParam('OutputFlag', 0)
m.optimize()
print('Optimal coverage points: %g' % m.objVal)

solution = []
if m.status == GRB.Status.OPTIMAL:
    for v in m.getVars():
        if v.x==1 and v.varName[0]=="y":
            solution.append(int(v.varName[1:]))
opt_sites = sites[solution]
```

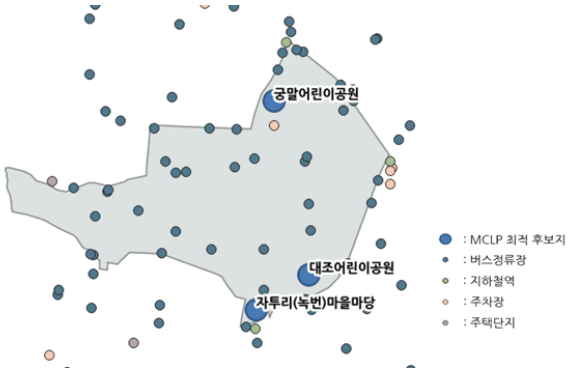
```
opts_sites, mobjVal = mclp(버스_points, 지하철_points, 주차장_points, 주택_points, points, 3, 300)
opts_sites
```

```
  Optimal coverage points: 1.76976

array([[126.915551,  37.599552],
       [126.918478,  37.601073],
       [126.916513,  37.608641]])
```

❑ 행정동별 가중치를 반영하여 기존의 목적함수를 수정한 결과, 목적함수에 따른 점수가 가장 높은 1~3 개의 최적 입지 후보지가

선정됨. 12 개의 행정동에 MCLP 를 적용하여 총 21 개의 최적 입지 후보지를 도출함



<그림 34> MCLP - '역촌동'의 최적 입지 후보지

4.3.4. P-Median 결과

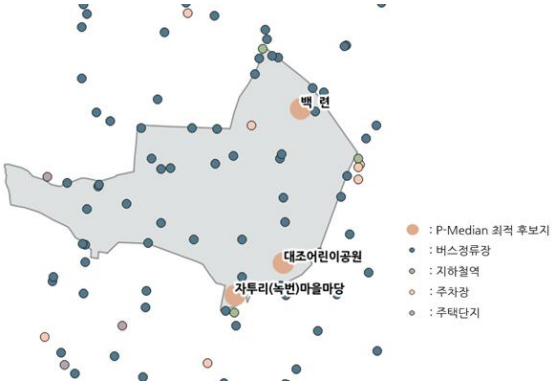
```
# 최단거리인 경우 가중치 부여, 아니면 0
idx = D1.index
col = D1.columns
lidx = len(idx)
lcol = len(col)

for i in range(lidx):
    for j in range(lcol):
        if D1.loc[idx[i], col[j]] == 최소값[i]:
            D1.loc[idx[i], col[j]] = m1
        else:
            D1.loc[idx[i], col[j]] = 0
```

```
D_final = pd.concat([D1, D2, D3, D4])
D_final.sum()
```

```
역말어린이공원         1.302492
연서어린이공원         0.048297
자투리(녹번)마을마당    3.715636
백   련           6.097530
신   사           0.096593
대조어린이공원         4.781216
길마어린이공원         1.764364
궁말어린이공원         1.374072
호연어린이공원         0.048297
dtype: float64
```

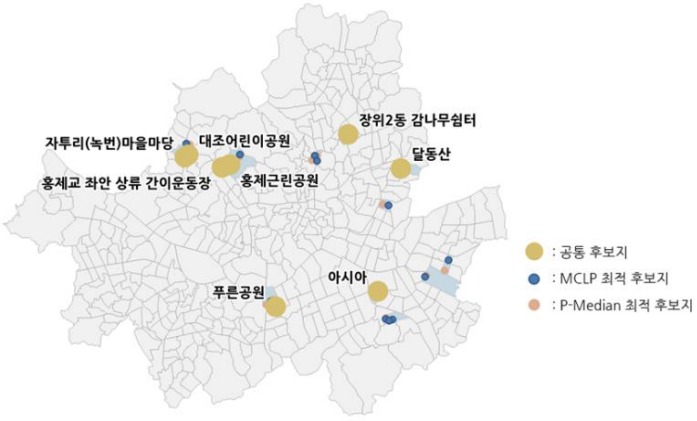
- ❑ 앞서 생성한 수요지점의 거리행렬에서 각 수요지점과 최단거리에 위치한 후보지의 경우 행정동별 가중치를, 그렇지 않으면 0 값을 부여함
- ❑ 후보지마다 부여된 값을 합산하여 점수가 가장 높은 1~3 개의 최적 입지 후보지를 선정함. MCLP 와 마찬가지로 12 개의 행정동에 P-Median 을 적용하여 총 21 개의 최적 입지 후보지가 도출됨



<그림 35> P-Median - '역촌동'의 최적 입지 후보지

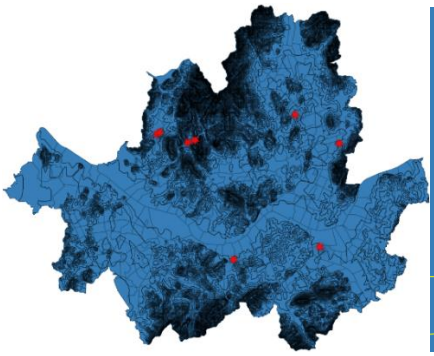
4.4. 최종 입지 선정 결과

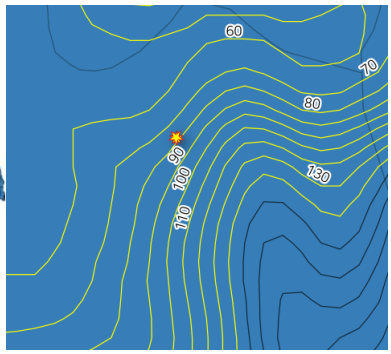
4.4.1. MCLP와 P-Median 결과 비교



<그림 36> MCLP 와 P-Median 의 공통 후보지

❑ MCLP 와 P-Median 에서 도출된 최적 입지 후보지를 비교한 결과, '자투리(녹번) 마을마당', '대조어린이공원', '홍제교 좌안 상류 간이운동장', '홍제근린공원', '장위 2 동 감나무쉼터', '달동산', '푸른공원', '아시아' 8 개의 후보지가 공통적으로 선정됨





<그림 37> 최적 입지 후보지의 고도 확인 결과

❑ 고도 데이터를 활용하여 8 개의 최적 입지 후보지의 이용 편리성을 확인한 결과, 경사도가 너무 가파른 '홍제근린공원'이 제외됨

4.4.2. 입지선정지수를 이용한 최종 입지 선정

❑ p-median 모델의 행정동별 가중치와 hierarchical clustering 의 행정동별 가중치를 고려하여 5 개의 최종 입지를 선정함. 선정된 상위 5 개의 입지는 **'아시아, 홍제교 좌안 상류 간이운동장, 달동산, 푸른공원, 장위 2 동 감나무쉼터'**임

	시설명	도로최소거리(m)	모델가중치	행정동가중치	입지선정지수
0	아시아	3.829486e+06	14.968368	27.35	80.3
1	홍제교 좌안 상류 간이운동장	4.005076e+05	7.554322	26.90	43.3
2	달동산	1.605864e+06	10.491134	23.15	40.5
3	푸른공원	8.976606e+06	6.123496	19.60	40.5
4	장위2동 감나무쉼터	8.724606e+05	10.618108	23.50	39.6
5	자투리(녹번)마을마당	3.873575e+06	3.715636	20.45	17.5
6	대조어린이공원	2.584589e+05	4.781216	20.45	6.8

<그림 38> 5 개의 최종 입지 좌표

5. 활용 방안

5.1. 문제점 개선 방안

5.1.1. 급속도로 증가하는 노인 수 대비 부족한 노인 시설 수

❑ (기존) 초고령화 사회로의 진입이 임박했으나 이를 대비할 만한 노인 복지 및 노인 시설은 미비함

❑ (개선 및 적용) 입지 선정 결과를 활용해 기존 시설인 공원 및 체육시설을 최적 입지로 선정하여 새로운 부지에 시설을 설립하는 것에 비해 부족한 노인 시설 수를 빠르게 확충함

5.1.2. 노인을 위한 융합형 복지시설 부족

❑ (기존) 노인복지시설은 의료복지 혹은 여가복지 등 특정 목적에 따라 설립되어 있으며, 노인이 자립적으로 이용할 수 있는 시설이 미비함

❑ (개선) 노인놀이터의 도입으로 사회 여가 문화 활동과 동시에 신체적, 정신적 건강상태를 증진시킬 수 있음

❑ (적용) 현장조사 및 주민의견을 수렴하여 내년 상반기 내에 노인 놀이터 사업을 실시함

5.1.3. 노인 전용 놀이터에 대한 부정적 여론

❑ (기존) 웹 크롤링을 통해 발견한 노인 놀이터에 대한 여론이 부정적임

❑ (개선) ‘노인 놀이터’에 대한 부정적 인식을 완화하고 노인 전용 시설이 아닌 다세대를 위한 통합의 공간 마련을 위하여 지역주민 접근성과 이용 편리성을 고려함

❑ (적용) 노인 친화형 다세대 놀이터의 도입으로 세대 간의 소통의 장 및 모든 세대가 이용 가능한 공간을 확보함

5.2. 업무 활용 방안

5.2.1. 노인 친화형 시설 입지를 위한 구체적 변인 다양화

❑ (기존 방식) 노인시설은 노인인구 수에 비례하여 입지하는 경우가 다수임

* 회귀 분석 결과 노인 인구 수가 많은 곳에 노인시설이 많은 것이 일반적

❑ (개선) 노인 거주 및 생활인구 비율, 지역주민 접근성, 이용 편리성을 고려하여, 노인 계층뿐만 아니라 다세대가 편리하게 이용할 수 있는 최적 입지를 선정함

❑ (활용) 노인 친화적이면서 국민 모두의 삶을 개선시킬 수 있는 공공 정책으로 활용 가능함

5.2.2. 서울시 행정동별 접근성 및 이용 편리성 지표 개발

❑ (접근성) 서울시 버스 정류장 위치 및 해당 정류장을 지나는 버스 개수, 환승역을 고려한 지하철역 위치, 주택 세대수를 고려한 접근성을

평가하여, 접근성이 좋은 위치에 공공시설을 설립하여 공익성을 최대화할 수 있음

- (이용 편리성) 주차장 여부 및 고도 데이터를 활용하여 공공시설의 주변 환경을 파악함으로써 이용 편리성을 극대화하여 공공시설 이용 시의 불편함을 해소함

5.2.3. 입지선정지수 개발

- (타 지역 확장) 분석에서 활용한 입지선정지수를 타 지역구의 특색에 맞춘 지수로 활용하여 타 지역구의 노인 놀이터 최적 입지 선정 시에 활용 가능함

6. 참고자료

고민정, 이건웅, 김상헌. (2018). 시니어를 위한 노인놀이터의 개념과 유형. 춘계 종합학술대회 논문집. 한국콘텐츠학회. 147-148.

* 기존의 노인복지기관과 노인문화복지관은 증대되는 노인의 건강과 삶의 질에 대한 이들의 욕구를 채워주는 데 한계가 있으므로, 새로운 니즈를 충족하는 새로운 개념의 노인놀이터가 필요함

고민정. (2019). 액티브 시니어 시대의 노인놀이터 도입 방안 연구. 상명대학교 대학원. 박사학위논문.

* 해외 선진국을 중심으로 급속히 보급되고 있는 노인놀이터의 한국 도입을 주제로, 한국형 노인놀이터 도입의 구체적인 방안과 모델을 제시하기 위해서 조례나 법적 장치, 예산 관련 재정적인 부분을 고려해야 하며, 노인놀이터 도입을 앞당기기 위해 실버 세대에 대한 관심과 배려가 필요함

문소현, 이건학. (2020). 지역 제한 P-median 모델을 이용한 서울시 주거복지센터 입지 분석 및 모델링. 대한지리학회지. 55(2). 197-206.

* 서울시 현행 주거복지센터의 입지적 특성을 분석한 뒤, 주거 복지의 수혜 가구를 대상으로 공간적 효율성과 형평성을 동시에 고려한 최적 주거복지센터의 입지를 제시함. 공공 시설의 지역별 균등 입지를 효율적으로 고려할 수 있는 지역 제한 P-median 모델(RCPMP)을 활용함

오민수 외 6 인. (2017). 종합사회복지관 건립 적정 입지 선정 연구.

* 광주시 내 종합사회복지관 설치를 위한 최적의 입지를 선정하기 위한 연구로, 복지관의 설치 형태 등을 모색하여 제시함. 경기도 내 유사사례 참고 및 이용가능권역 주민대상 의견 또한 수렴하여

최적 위치를 제시함. 연구 방법으로는 문헌분석, 주민대상 인식조사, Rawls 모형을 활용한 입지선정 모델을 구성하여 사용함

〈표 Ⅱ-12〉 종합사회복지관 입지선정 분석틀

구분		복지관 입지결정인자	데이터 구축방법
입지선정인자	공간적 형평성	인구밀도	행정동별 인구자료
		복지대상자 수	저소득, 아동, 노인, 장애인 수
	유관기관 연계성	집중도	입지계수
	지역주민 접근성	도로접근도	1. 후보지별 가장 가까운 주민센터 직선거리 및 진입 도로 갯수
			2. 후보지별 가장 먼 주민센터 직선거리 및 진입 도로 갯수
			3. 후보지별 가장 가까운 보건소 직선거리| 및 진입 도로 갯수
		대중교통수단과의 거리	버스정류소 위치와 후보지 간 비교
		대중교통최대통행 거리	1. 후보지 정류소와와 각 행정동별 가장 먼 버스정류소 간 거리 총합
			2. 행정동별 주민센터를 출발점으로 후보지 간 버스정류소 거리 총합
		행정 및 보건시설과의 접근성	1:100 수치지도의 토지피복도, 보건시설, 행정시설 위치 및 거리 파악
	지역주민 참여성	주민 인식 조사	아통장 및 읍면동지역사회보장협의체 대상 주민조사
배제지역인자		하천, 상수원보호구역 이격거리	Daum 거리지도 이용
		집중도	입지계수

이건웅, 고민정. (2019). 인간다운 삶. 공존의 시대. 코리아 매니페스토. 57-80.

* 기존의 노인복지관, 경로당, 노인교실 등 이외의 새로운 노인 여가복지시설로 노인놀이터를 신설하는 정책을 제안함. 또한 노인의 개념과 노인문화복지시설의 개념을 분석해 보고, 관련 해외 사례를 제시하며, 기존 어린이공원 및 어린이놀이터에 비추어 노인놀이터의 개념과 유형을 제시함

이형숙, 안준석, 전승훈. (2011). 도시 노인들의 걷기활동 참여에 영향을 주는 물리적 환경요인 분석. 한국조경학회지 39(2). 65-72.

* 걷기 운동장소로 선호되고 있는 곳은 도보권 내의 공원이나 산책로, 아파트 단지 등이 많았으며, 유료시설의 체육시설이나 실내 운동기구 이용은 상대적으로 낮았음. 걷기운동 빈도에 유의적인 영향을 미치는 물리적 환경 요인으로는 공원접근성, 교통안전, 신호등, 가로등이 있었으며, 소득이 상대적으로 적은 노인들은 걷기운동에 대한 선호도가 보다 높은 것으로 파악됨

이유진. 최명섭. (2018). 노인 인구 밀집지역의 시공간적 분포와 결정요인분석: 서울 생활인구 빅데이터의 활용. 서울도시연구. 19(4). 149-168.

* 서울 생활인구 빅데이터를 활용해 이항로짓 모형으로 노인 밀집지역의 특성을 분석한 결과, 일상활동 수요시설의 접근성과 대중교통 접근성이 평일 주간의 주된 노인 밀집 요인으로 확인되며 야간에만 노인 인구가 밀집한 지역은 대체로 주택환경이 불량하고 일상활동 수요시설의 접근성과 대중교통 접근성이 낮음. 따라서 이러한 지역의 노인의 사회적 배제를 막고 일상활동 수요를 효율적으로 충족하기 위한 필요시설과 서비스를 제공하는 지역으로 이동성을 높이는 노력이 필요함

임경호. 김아연. (2013). 노인의 실외 놀이공간 실태분석: 노인친화형 공원 및 시니어 놀이터 사례를 중심으로. 춘계학술대회 논문집. 한국조경학회. 26-30.

* 지역과 세대의 조화를 이루지 못하는 놀이공간은 노인들을 더욱 고립시키는 역할을 할 수 있으므로, 아동, 장애인, 지역주민 모두가 이용 가능한 시설 및 공간 배치로 노인의 놀이공간 활성화를 유도해야 함

조현주, 이순주. (2019). 보행적 측면에서 노인친화형 공원의 유치거리 도출 및 녹지서비스 지역 평가: 보행자 측면 중심으로. 한국조경학회지. 47(1). 1-9.

* 신체적 변화를 고려한 노인친화형 공원의 유치거리를 도출해 본 결과, 노인친화형 공원의 유치거리는 300m 로 도출

7. 부록

1. 데이터 전처리 코드

```
#!/usr/bin/env python

# coding: utf-8



import pandas as pd

import numpy as np



# ## 0. 행정동 기준데이터

HJD = pd.read_csv('/content/drive/MyDrive/ᄋᆡᆸᄌᆞᄋᆡᄉᆞᄀᆡᄂᆞᄌᆞᄀᆞᄋᆞᄃᆞᄀᆡᄋᆞᄐᆞᄀᆡ/ᄂᆞᄂᆞᄋᆞᄂᆞᄉᆞᄀᆡᄋᆞᄒᆞᄉᆞᆯᄆᆞᄋᆡᄎᆞᄀᆞᄀᆞᄌᆞᄀᆞ/ᄒᆞᄀᆞᄋᆞᄌᆞᄀᆞᄋᆞᄃᆞᄂᆞᄋᆞᄏᆞᄂᆞᄃᆞᄂᆞᄆᆞᄀᆡᄑᆞᄋᆡᄋᆞ.csv',encoding='cp949')

HJD.rename(columns={'시군구':'자치구'},inplace=True)

HJD.head()

print(HJD.dtypes)

# 형변환

HJD['행정동'] = HJD['행정동'].astype("string")

HJD['자치구'] = HJD['자치구'].astype("string")

print(HJD.dtypes)

HJD['자치행정동'] = HJD['자치구'] + " " + HJD['행정동']

HJD.head()

HJD.dtypes

# ## 1. 노인 거주 인구 데이터
```

```
sum(노인인구.duplicated(['자치구','행정동']))
```

```
노인인구 =
pd.read_csv('/content/drive/MyDrive/ᄋᆡᄇᆞᄌᆞ
ᆡᄉᆞᄒᆡᄂᆞᄌᆞᄒᆞᆼᄃᆞᄒᆡᄋᆞᄐᆞᄒᆡ/ᄂᆞᄉᆞᄋᆡᄂᆞᄉᆞᄒᆡᆼ
ᄒᆞᄀᆞᄅᆞᄆᆞ ᆡᄉᆞᄏᆞᄀᆞᄌᆞᄀᆞᄒᆞ/ᄃᆞᄉᆞᄋᆞᄇᆞᄒᆞᄅᆞᄂᆞᄉᆞᄋᆞ ᆡ
ᄂᆞᄀᆞᄒᆞᄌᆞᄋᆞ ᆡᄂᆞᄀᆞᄂᆞᄉᆞᄂᆞᄉᆞᄌᆞᄒᆞ
ᄋᆞ.csv',encoding='cp949')

노인인구.rename(columns={'동':'행정동
'},inplace=True)

노인인구 = 노인인구[['자치구','행정동','거주인
구수','거주인구비율']]

노인인구.head()
```

```
노인인구['자치행정동'] = (노인인구['자치구'] + "
" + 노인인구['행정동']).astype("string")

노인인구.head()
```

```
# 최종 데이터에 추가

df_all = pd.merge(HJD,노인인구, how = "left",
on = "자치행정동")

df_all
```

```
노인인구[노인인구['행정동'] == '종로5.6가동']
```

```
# 중복값확인
```

```
df_all['df_diff'] = df_all['자치구_x'] != df_all['자치
구_y']

df_all['df_diff2'] = df_all['행정동_x'] != df_all['행
```

```
정동_y']                                    정동','자치구_x','행정동_x','거주인구수','거주인
                                            구비율']]
                                            df_all.rename(columns={'자치구_x':'자치구
                                            '},inplace=True)
                                            df_all.rename(columns={'행정동_x':'행정동
                                            '},inplace=True)

df_all['df_diff'].sum()

                                            print(len(df_all))
                                            df_all.head()
df_all['df_diff2'].sum()

                                            df_all.dtypes

                                            # ## 2. 생활인구수 data: 2020년 7월부터
                                            2021년 7월까지 data 평균값

                                            # ### 1) 데이터 불러오기
df_all = df_all[['행정동코드','읍면동코드','자치행
```

```
생활인구 = pd.read_csv('/content/drive/MyDrive/ᄋᆡᆸᄌᆞᆺᄀᆡᄂᆞᄌᆞᄋᆞ데ᄋᆞ트ᄀᆡ/ᄂᆞᄉᆞᄋᆞᄂᆞᄉᆞᄒᆡᆼᄒᆞᆨᄀᆞᆯᄆᆞᄋᆡᆺᄀᆞᄌᆞᄀᆞᆫᄃᆞᄇᆡᄇᆞᄅᆞᆯᄉᆞᄒᆡᆼᄒᆞᆨᄀᆞᆯᄋᆞᄋᆡᄂᆞᄀᆞᄂᆞᄉᆞᄂᆞᄉᆞᄒᆡᆼᄒᆞᆨᄀᆞᆯᄋᆞᄋᆡᄂᆞᄀᆞᄂᆞᄒᆞᄒᆡᆼᄌᆞᄀᆞᄃᆞᄉᆞᄋᆞᄎᆞᄂᆞᄀᆞᅡ.csv',encoding='cp949')

생활인구.rename(columns={'행정동명':'행정동'},inplace=True)

생활인구['노인(65세이상)생활인구비율'] = 생활인구['노인(65세이상)생활인구수']/생활인구['총생활인구수평균']

생활인구 = 생활인구[['행정동코드','행정동','시간대구분','노인(65세이상)생활인구수','노인(65세이상)생활인구비율']]

생활인구.head()
```

```
# ### 2) 형변환

# 형변환

생활인구['행정동'] = 생활인구['행정동'].astype("string")

생활인구['시간대구분'] = 생활인구['시간대구분'].astype("category")

생활인구.dtypes
```

```
# ### 3) 결측치 및 이상치 탐색

# 서울시 구로구 항동 데이터 X

# -> 구로구: 신도림동 구로1동 구로2동 구로3동 구로4동 구로5동 고척1동 고척2동 개봉2동 개봉3동 오류1동 오류2동 수궁동 가리봉동 개봉1동

# 평균값으로 결측치 대체

생활인구_구로 = 생활인구.query('행정동.str.contains("신도림|구로|고척|개봉|오류|수궁|가리봉")', engine='python')

생활인구_구로
```

```
생활인구_항동 = 생활인구_구로.groupby(['시간대구분'], as_index=False).mean()
생활인구_항동['행정동'] = "항동"
생활인구_항동['행정동코드'] =11530800
생활인구_항동 = 생활인구_항동[['행정동코드','행정동','시간대구분','노인(65세이상)생활인구수','노인(65세이상)생활인구비율']]
생활인구_항동
```

```
# 항동 데이터 추가
생활인구 = 생활인구.append(생활인구_항동)
생활인구.reset_index(drop=True,inplace=True)
생활인구
```

```
# 중복값 확인
sum(생활인구.duplicated(['행정동코드','행정동','시간대구분']))
```

```
# ### 4) 시간대별 데이터 분할
```

```
# 시간대별로 데이터 나누기
생활인구_새벽 = 생활인구[생활인구['시간대구분'] == "새벽"]
생활인구_아침 = 생활인구[생활인구['시간대구분'] == "아침"]
생활인구_낮 = 생활인구[생활인구['시간대구분'] == "낮"]
생활인구_저녁 = 생활인구[생활인구['시간대구분'] == "저녁"]
생활인구_밤 = 생활인구[생활인구['시간대구분'] == "밤"];생활인구_밤.head()
```

```
# 새벽
생활인구_새벽.rename(columns={'노인(65세이상)생활인구수':'새벽생활인구수'},inplace=True)
생활인구_새벽.rename(columns={'노인(65세이상)생활인구비율':'새벽생활인구비율
```

```
'},inplace=True)

생활인구_새
벽.reset_index(inplace=True,drop=True)

생활인구_새벽 = 생활인구_새벽[['행정동코드','
행정동','새벽생활인구수','새벽생활인구비율']]


# 아침

생활인구_아침.rename(columns={'노인(65세이
상)생활인구수':'아침생활인구수'},inplace=True)

생활인구_아침.rename(columns={'노인(65세이
상)생활인구비율':'아침생활인구비율
'},inplace=True)

생활인구_아
침.reset_index(inplace=True,drop=True)

생활인구_아침 = 생활인구_아침[['행정동코드','
행정동','아침생활인구수','아침생활인구비율']]


# 낮

생활인구_낮.rename(columns={'노인(65세이상)
생활인구수':'낮생활인구수'},inplace=True)

생활인구_낮.rename(columns={'노인(65세이상)
생활인구비율':'낮생활인구비율'},inplace=True)

생활인구_
낮.reset_index(inplace=True,drop=True)

생활인구_낮 = 생활인구_낮[['행정동코드','행정
동','낮생활인구수','낮생활인구비율']]


# 저녁

생활인구_저녁.rename(columns={'노인(65세이
상)생활인구수':'저녁생활인구수'},inplace=True)

생활인구_저녁.rename(columns={'노인(65세이
상)생활인구비율':'저녁생활인구비율
'},inplace=True)

생활인구_저
녁.reset_index(inplace=True,drop=True)

생활인구_저녁 = 생활인구_저녁[['행정동코드','
행정동','저녁생활인구수','저녁생활인구비율']]


# 밤

생활인구_밤.rename(columns={'노인(65세이상)
생활인구수':'밤생활인구수'},inplace=True)

생활인구_밤.rename(columns={'노인(65세이상)
생활인구비율':'밤생활인구비율'},inplace=True)

생활인구_
밤.reset_index(inplace=True,drop=True)

생활인구_밤 = 생활인구_밤[['행정동코드','행정
동','밤생활인구수','밤생활인구비율']]


# 확인

print(len(생활인구_새벽),len(생활인구_아
침),len(생활인구_낮),len(생활인구_저녁),len(생활
인구_밤))
```

```
생활인구_새벽.head()
```

```
생활인구_밤.head()
```

```
# ### 5) 최종 데이터에 merge
```

```
# 최종 데이터 merge

df_all2 = pd.merge(df_all,생활인구_새벽, how = "left", on = "행정동코드")

df_all2
```

```
# 데이터 무결성 확인

df_all2['df_diff'] = df_all2['행정동_x'] != df_all2['행정동_y']

df_all2['df_diff'].sum()
```

```
df_all2 = df_all2[['행정동코드','읍면동코드','자치행정동','자치구','행정동_x','거주인구수','거주인구비율','새벽생활인구수','새벽생활인구비율']]

df_all2.rename(columns={'행정동_x':'행정동'},inplace=True)

df_all2.head()
```

```
# 아침

df_all3 = pd.merge(df_all2,생활인구_아침, how = "left", on = "행정동코드")

## 데이터 무결성 확인

df_all3['df_diff'] = df_all3['행정동_x'] != df_all3['행정동_y']

if df_all3['df_diff'].sum() == 0:

    df_all3 = df_all3[['행정동코드','읍면동코드','자치행정동','자치구','행정동_x','거주인구수','거
```

```
주인구비율','새벽생활인구수','새벽생활인구비율','아침생활인구수','아침생활인구비율']]
    df_all3.rename(columns={'행정동_x':'행정동'},inplace=True)
    print(df_all3.head())
else:
    print("데이터 무결성 오류")


# 낮
df_all4 = pd.merge(df_all3,생활인구_낮, how = "left", on = "행정동코드")
## 데이터 무결성 확인
df_all4['df_diff'] = df_all4['행정동_x'] != df_all4['행정동_y']
if df_all4['df_diff'].sum() == 0:
    df_all4 = df_all4[['행정동코드','읍면동코드','자치행정동','자치구','행정동_x','거주인구수','거주인구비율','새벽생활인구수','새벽생활인구비율','아침생활인구수','아침생활인구비율','낮생활인구수','낮생활인구비율']]
    df_all4.rename(columns={'행정동_x':'행정동'},inplace=True)
    print(df_all4.head())
else:
    print("데이터 무결성 오류")


# 저녁
df_all5 = pd.merge(df_all4,생활인구_저녁, how = "left", on = "행정동코드")
## 데이터 무결성 확인
df_all5['df_diff'] = df_all5['행정동_x'] != df_all5['행정동_y']
if df_all5['df_diff'].sum() == 0:
    df_all5 = df_all5[['행정동코드','읍면동코드','자치행정동','자치구','행정동_x','거주인구수','거주인구비율','새벽생활인구수','새벽생활인구비율','아침생활인구수','아침생활인구비율','낮생활인구수','낮생활인구비율','저녁생활인구수','저녁생활인구비율']]
    df_all5.rename(columns={'행정동_x':'행정동'},inplace=True)
    print(df_all5.head())
else:
    print("데이터 무결성 오류")


# 밤
df_all6 = pd.merge(df_all5,생활인구_밤, how = "left", on = "행정동코드")
## 데이터 무결성 확인
df_all6['df_diff'] = df_all6['행정동_x'] != df_all6['행정동_y']
if df_all6['df_diff'].sum() == 0:
    df_all6 = df_all6[['행정동코드','읍면동코드','자치행정동','자치구','행정동_x','거주인구수','거주인구비율','새벽생활인구수','새벽생활인구비율','아침생활인구수','아침생활인구비율','낮생활인구수','낮생활인구비율','저녁생활인구수','저녁생
```

```
활인구비율','밤생활인구수','밤생활인구비율']]
    df_all6.rename(columns={'행정동_x':'행정동
'},inplace=True)
    print(df_all6.head())
else:
    print("데이터 무결성 오류")
```

```
df_all6
```

```
# ## 3. 노인시설데이터
#
# 오류2동 분리 후 항동 추가
#
# 신당1동 -> 신당동
# 신당2동 -> 다산동
# 신당3동 -> 약수동
# 신당4동 -> 청구동
# 신당6동 -> 동화동
# 공릉1.3동 -> 공릉1동
#
```

```
# 명륜3가동 -> 가회동에 포함
#
# 혜화동,답십리1동,오류2동,장지동 코드 수정
#
# 위례동 추가

# ### 1) 데이터 불러오기
```

```
노인시설 =
pd.read_csv('/content/drive/MyDrive/ᄋᆡᄇᄌᆞ
ᆡᄉᆞᄒᆡᄂᆞᄌᆞᄒᆡᄋᆞ데ᄋᆞᄋᆡᄐᆞᄒᆡ/ᄂᆞᄉᆞᄋᆞᄋᆡᄂᆞᄉᆞᄇᆡᄋᆞ
ᄒᆞᄉᆞᄅᆞᄆᆞᄋᆡᄉᆞᄀᆞᄒᆡᄌᆞᄀᆞ/ᄂᆞᄉᆞᄋᆞᄋᆡᄂᆞᄇᆞᄉᆞᄀᆞᄌᆞᄋᆡᄉᆞ
ᄋᆡᄉᆞᄒᆡᄅᆞᄋᆞᄀᆡᄎᆞᄋᆡ/ᄂᆞᄉᆞᄋᆞᄋᆡᄂᆞᄉᆞᄋᆡᄉᆞᄒᆡᄅᆞᄎᆞᄀᆡ
ᄌᆞᄉᆞᄋᆞ.csv',encoding='cp949')
노인시설
```

```
# ### 2) 형변환
```

```
# 형변환
```

```
print(노인시설.dtypes)

노인시설['읍면동'] = 노인시설['읍면동'].astype("string")

노인시설.dtypes

노인시설.isnull().sum()

# ### 3) 최종 데이터에 합치기

# 최종 데이터 merge

df_all7 = pd.merge(df_all6,노인시설, how = "left", on = "읍면동코드")

df_all7

# 데이터 무결성 확인

df_all7['df_diff'] = df_all7['행정동'] != df_all7['읍면동']

df_all7['df_diff'].sum()

df_all7 = df_all7[['행정동코드','읍면동코드','자치행정동','자치구','행정동','거주인구수','거주인구비율','새벽생활인구수','새벽생활인구비율','아침생활인구수','아침생활인구비율','낮생활인구수','낮생활인구비율','저녁생활인구수','저녁생활인구비율','밤생활인구수','밤생활인구비율','총노인시설수']]

df_all7.head()

df_all7.to_csv("/content/drive/MyDrive/ㅇㅣㅂㅈㅣㅅㅓㄴㅈㅓㅇㄷㅐㅇㅣㅌㅓ/회귀분석데이터.csv",encoding='cp949')
```

'},inplace=True)

접근성.rename(columns={'환승역가중치부여(환승역*2)':'지하철수'},inplace=True)

접근성.head()

df_all7.isnull().sum()

2) 형변환

형변환

4. 접근성 지표

#

접근성['자치구'] = 접근성['자치구'].astype("string")

접근성['행정동'] = 접근성['행정동'].astype("string")

1) 데이터 불러오기: 버스, 지하철, 주택

접근성.dtypes

접근성 = pd.read_csv('/content/drive/MyDrive/ㅇㅣㅂㅈㅣㅅㅓㄴㅈㅓㅇㄷㅔㅇㅣㅌㅓ/ㅈㅣㅇㅕㄱㅈㅜㅁㅣㄴㅈㅓㅂㄱㅡㄴㅅㅓㅇ/ㅂㅓㅅㅡ,ㅈㅣㅎㅏㅊㅓㄹ,ㅈㅜㅌㅐㄱㅅㅔㄷㅐㅅㅜ_ㅇㅘㄴㅅㅓㅇㅂㅗㄴ.csv',encoding='cp949')

접근성['자치행정동'] = 접근성['자치구'] + " " + 접근성['행정동']

접근성.rename(columns={'행정구':'자치구

접근성.head()

```
접근성_sc = 접근성[['자치행정동','버스수','지하철수','주택세대수']]

x_data = 접근성_sc.drop(['자치행정동'], axis=1)
y_data = 접근성_sc['자치행정동']

x_data
```

```
len(접근성['자치행정동'].unique()) # 개수확인
```

```
접근성.isnull().sum()
```

```
from sklearn.preprocessing import minmax_scale

x_data = minmax_scale(x_data)

x_data
```

```
접근성[접근성['자치행정동'] == "종로구 가회동"]
```

```
# ### 3) 정규화
```

```
def column(matrix, i):
    return [row[i] for row in matrix]
```

```
접근성_sc['버스수'] = [row[0] for row in x_data]
접근성_sc['지하철수'] = [row[1] for row in x_data]
접근성_sc['주택세대수'] = [row[2] for row in x_data]
```

```
접근성_sc
```

```
접근성_sc['접근성지표'] = (접근성_sc['버스수'] * 28.5) + (접근성_sc['지하철수'] * 19.2) + (접근성_sc['주택세대수'] * 27.6)
접근성_sc = 접근성_sc[['자치행정동','접근성지표']]
접근성_sc
```

```
접근성_sc[접근성_sc['접근성지표']==0]
```

```
접근성_sc.isnull().sum()
```

```
접근성_sc['접근성지표'].min()
```

```
접근성_sc['접근성지표'].max()
```

4) 데이터 합치기

최종 데이터 merge

df_all8 = pd.merge(df_all7,접근성_sc, how = "left", on = "자치행정동")

df_all8

df_all8.isnull().sum()

5. 이용편리성 지표

1) 데이터 불러오기

이용편리 =

pd.read_csv('/content/drive/MyDrive/ㅇㅣㅂㅈㅣㅅㅓㄴㅈㅓㅇㄷㅐㅇㅣㅌㅓ/ㅇㅣㅇㅛㅇㅍㅕㄴㄹㅣㅅㅓㅇ/ㅈㅜㅊㅏㅈㅏㅇ/ㅈㅜㅊㅏㅈㅏㅇ_ㅊㅚㅈㅗㅇㄱㅐㅅㅜ.csv',encoding='cp949')

이용편리.rename(columns={'읍면동':'행정동'},inplace=True)

이용편리.head()

2) 형변환

#

형변환

이용편리['행정동'] = 이용편리['행정동'].astype("string")

이용편리.dtypes

3) 정규화

```
from sklearn.preprocessing import minmax_scale

sc = minmax_scale(이용편리['주차장수'])
sc = sc*50

이용편리['이용편리지표'] = sc
이용편리 = 이용편리[['읍면동코드','이용편리지표']]
이용편리

이용편리.isnull().sum()

# ### 4)데이터 합치기

# 최종 데이터 merge
df_all9 = pd.merge(df_all8,이용편리, how = "left", on = "읍면동코드")
df_all9

df_all9.isnull().sum()

# 데이터 내보내기
df_all9.to_csv("/content/drive/MyDrive/ㅇㅣㅂㅈㅣㅅㅓㄴㅈㅓㅇㄷㅐㅇㅣㅌㅓ/클러스터링데이터.csv",encoding='cp949')
```

```
reg = pd.read_csv("/content/drive/MyDrive/ᄋᆡᆸᄌᆞ ᆡᄉᆞ ᅥᄂᆞᄌᆞ ᅥᆼᄃᆞ ᅦᄋᆞ ᆡᄐᆞ ᅥ/회귀분석데이터.csv",encoding='cp949')

reg
```

```
x_data = reg[['거주인구수','낮생활인구비율']]

y= reg[['총노인시설수']]
```

```
import numpy as np

import statsmodels.API as sm

import statsmodels.formula.API as smf

import pandas as pd
```

```
result = smf.ols(formula = '총노인시설수 ~ 거주인구수 + 낮생활인구비율', data = reg).fit()

result.summary()
```

3. 다중회귀

```
#!/usr/bin/env python

# coding: utf-8


# In[1]:


import pandas as pd

import numpy as np

import statsmodels.API as sm

import matplotlib.pyplot as plt

from statsmodels.formula.API import ols

from mpl_toolkits.mplot3d import Axes3D

from sklearn import linear_model


# # 1. 데이터 불러오기
```

```
df = pd.read_csv('C:/Users/hyein/Desktop/대외활동/공빅/프로젝트/data/회귀분석데이터.csv',encoding='cp949')
```

```
df.head()
```

```
df.columns
```

```
df = df[['행정동','거주인구수',
        '거주인구비율', '새벽생활인구수', '새벽생활인구비율', '아침생활인구수', '아침생활인구비율', '낮생활인구수',
        '낮생활인구비율', '저녁생활인구수', '저녁생활인구비율', '밤생활인구수', '밤생활인구비율','총노인시설수']]
```

```
df
```

```
df.columns
```

```
df.info()
```

```
#print(df[df['총노인시설수'].isnull()]['행정동'])
```

```
#print('₩n')
```

```
#print(df[df['거주인구수'].isnull()]['행정동'])
```

```
# null값 처리
```

```
df = df.fillna(0)
```

'낮생활인구수', '낮생활인구비율', '저녁생활인구수', '저녁생활인구비율', '밤생활인구수', '밤생활인구비율'] ## 설명 변수 리스트

2. 변수 선택

y = df['총노인시설수'] ## 반응 변수

1) 단계적 변수 선택법

selected_variables = [] ## 선택된 변수들

sl_enter = 0.05

sl_remove = 0.05

sv_per_step = [] ## 각 스텝별로 선택된 변수들

adjusted_r_squared = [] ## 각 스텝별 수정된 결정계수

steps = [] ## 스텝

step = 0

while len(variables) > 0:

remainder = list(set(variables) - set(selected_variables))

pval = pd.Series(index=remainder) ## 변수의 p-value

기존에 포함된 변수와 새로운 변수 하나씩 돌아가면서

선형 모형을 적합한다.

for col in remainder:

X = df[selected_variables+[col]]

X = sm.add_constant(X)

model = sm.OLS(y,X).fit()

전진 단계별 선택법

variables = ['거주인구수', '거주인구비율', '새벽생활인구수', '새벽생활인구비율', '아침생활인구수', '아침생활인구비율',

```
        pval[col] = model.pvalues[col]


    min_pval = pval.min()
    if min_pval < sl_enter: ## 최소 p-value 값이 기준 값보다 작으면 포함

selected_variables.append(pval.idxmin())
        ## 선택된 변수들에대해서
        ## 어떤 변수를 제거할지 고른다.
        while len(selected_variables) > 0:
            selected_X = df[selected_variables]
            selected_X = sm.add_constant(selected_X)
            selected_pval = sm.OLS(y,selected_X).fit().pvalues[1:] ## 절편항의 p-value는 뺀다
            max_pval = selected_pval.max()
            if max_pval >= sl_remove: ## 최대 p-value값이 기준값보다 크거나 같으면 제외
                remove_variable = selected_pval.idxmax()

selected_variables.remove(remove_variable)
            else:
                break


        step += 1
        steps.append(step)
        adj_r_squared = sm.OLS(y,sm.add_constant(df[selected_variables])).fit().rsquared_adj

adjusted_r_squared.append(adj_r_squared)

sv_per_step.append(selected_variables.copy())
    else:
        break



selected_variables
```

```
from matplotlib import font_manager, rc
font_path = "C:/Windows/Fonts/NGULIM.TTF"
font = font_manager.FontProperties(fname=font_path).get_name()
rc('font', family=font)
```

```
# ## 2) 전진 선택법

## 전진 선택법
variables = ['거주인구수', '거주인구비율', '새벽생활인구수', '새벽생활인구비율', '아침생활인구수', '아침생활인구비율',
             '낮생활인구수', '낮생활인구비율', '저녁생활인구수', '저녁생활인구비율', '밤생활인구수', '밤생활인구비율'] ## 설명 변수 리스트

y = df['총노인시설수'] ## 반응 변수
selected_variables = [] ## 선택된 변수들
sl_enter = 0.05

sv_per_step = [] ## 각 스텝별로 선택된 변수들
adjusted_r_squared = [] ## 각 스텝별 수정된 결정계수
steps = [] ## 스텝
step = 0

fig = plt.figure(figsize=(10,10))
fig.set_facecolor('white')

font_size = 15
plt.xticks(steps,[f'step {s}₩n'+'₩n'.join(sv_per_step[i]) for i,s in enumerate(steps)], fontsize=12)
plt.plot(steps,adjusted_r_squared, marker='o')

plt.ylabel('Adjusted R Squared',fontsize=font_size)
plt.grid(True)
plt.show()
```

```
while len(variables) > 0:
    remainder = list(set(variables) - set(selected_variables))
    pval = pd.Series(index=remainder) ## 변수의 p-value
    ## 기존에 포함된 변수와 새로운 변수 하나씩 돌아가면서
    ## 선형 모형을 적합한다.
    for col in remainder:
        X = df[selected_variables+[col]]
        X = sm.add_constant(X)
        model = sm.OLS(y,X).fit()
        pval[col] = model.pvalues[col]

    min_pval = pval.min()
    if min_pval < sl_enter: ## 최소 p-value 값이 기준 값보다 작으면 포함

selected_variables.append(pval.idxmin())

        step += 1
        steps.append(step)
        adj_r_squared = sm.OLS(y,sm.add_constant(df[selected_variables])).fit().rsquared_adj
adjusted_r_squared.append(adj_r_squared)

sv_per_step.append(selected_variables.copy())
    else:
        break

selected_variables
```

```
fig = plt.figure(figsize=(10,10))
fig.set_facecolor('white')

font_size = 15
plt.xticks(steps,[f'step {s}₩n'+'₩n'.join(sv_per_step[i]) for i,s in enumerate(steps)], fontsize=12)
plt.plot(steps,adjusted_r_squared, marker='o')
```

```
plt.ylabel('Adjusted R Squared',fontsize=font_size)
plt.grid(True)
plt.show()
```

```
selected_variables = variables ## 초기에는 모든 변수가 선택된 상태
sl_remove = 0.05

sv_per_step = [] ## 각 스텝별로 선택된 변수들
adjusted_r_squared = [] ## 각 스텝별 수정된 결정계수
steps = [] ## 스텝
step = 0
```

3) 후진소거법

```
## 후진 소거법
variables = ['거주인구수', '거주인구비율', '새벽생활인구수', '새벽생활인구비율', '아침생활인구수', '아침생활인구비율',
             '낮생활인구수', '낮생활인구비율', '저녁생활인구수', '저녁생활인구비율', '밤생활인구수', '밤생활인구비율'] ## 설명 변수 리스트

y = df['총노인시설수'] ## 반응 변수

while len(selected_variables) > 0:
    X = sm.add_constant(df[selected_variables])
    p_vals = sm.OLS(y,X).fit().pvalues[1:] ## 절편항의 p-value는 뺀다
    max_pval = p_vals.max() ## 최대 p-value
    if max_pval >= sl_remove: ## 최대 p-value 값이 기준값보다 크거나 같으면 제외
        remove_variable = p_vals.idxmax()

selected_variables.remove(remove_variable)
```

```
            step += 1
            steps.append(step)
            adj_r_squared = sm.OLS(y,sm.add_constant(df[selected_variables])).fit().rsquared_adj

adjusted_r_squared.append(adj_r_squared)

sv_per_step.append(selected_variables.copy())
        else:
            break



selected_variables
```

```
fig = plt.figure(figsize=(10,10))
fig.set_facecolor('white')

font_size = 15
plt.xticks(steps,[f'step {s}₩n'+'₩n'.join(sv_per_step[i]) for i,s in enumerate(steps)], fontsize=12)
plt.plot(steps,adjusted_r_squared, marker='o')

plt.ylabel('Adjusted R Squared',fontsize=font_size)
plt.grid(True)
plt.show()
```

3. 다중회귀 분석

1) 3D 시각화

참고사이트 : https://lovelydiary.tistory.com/346

```
x_data = df[['거주인구수', '낮생활인구비율']]
#x_data = sm.add_constant(x_data)
target = df[['총노인시설수']]

multi_model = sm.OLS(target, x_data)
fitted_multi_model = multi_model.fit()
fitted_multi_model.summary()
```

```
dff = df[['거주인구수', '낮생활인구비율','총노인시설수']]
```

In[39]:

```
X = df[['거주인구수', '낮생활인구비율']].values.reshape(-1,2)
Y = df['총노인시설수']

# 시각화를 위한 데이터 준비

x = X[:, 0] #Horsepower 열 값만 array로 가져옴
y = X[:, 1] #MPG.city 열 값만 array로 가져옴
z = Y

#각 독립변수들의 range 지정: 데이터 중 최소값, 최대값, 그 구간을 10으로 설정
x_pred = np.linspace(df['거주인구수'].min(), df['
```

```
거주인구수'].max(), 10)

y_pred = np.linspace(df['낮생활인구비율'].min(), df['낮생활인구비율'].max(), 10)

xx_pred, yy_pred = np.meshgrid(x_pred, y_pred)

model_viz = np.array([xx_pred.flatten(), yy_pred.flatten()]).T


#sklearn의 linear_model.LinearRegression()함수로 다중선형회귀식을 만들고,

ols = linear_model.LinearRegression()

model = ols.fit(X, Y)


#위에서 만든 model_viz 값을 넣어 다중선형회귀식의 예측값을 얻음

predicted = model.predict(model_viz)


# 해당 다중선형회귀식의 결정계수 R^2을 구함: 회귀식의 실제 데이터에 대한 설명력을 확인


r2 = model.score(X, Y)


#다중선형회귀 그래프 그리기


plt.style.use('default')


fig = plt.figure(figsize=(10, 5))

ax1 = fig.add_subplot(131, projection='3d')

ax2 = fig.add_subplot(132, projection='3d')

ax3 = fig.add_subplot(133, projection='3d')


axes = [ax1, ax2, ax3]


for ax in axes:

    ax.plot(x, y, z, color='k', zorder=15, linestyle='none', marker='o', alpha=0.5) #검은색 마커들

    ax.scatter(xx_pred.flatten(), yy_pred.flatten(), predicted, s=20, edgecolor='#70b3f0') #파란색 마커들

    ax.set_xlabel('old_population', fontsize=12) #해당 축을 설명하는 라벨

    ax.set_ylabel('day_living_population', fontsize=12)

    ax.set_zlabel('tot_old_facility', fontsize=12)

    ax.locator_params(nbins=6, axis='x') #해당 축의 구간 개수 (쪼개진 구간도 포함)

    ax.locator_params(nbins=4, axis='y')

    ax.locator_params(nbins=4, axis='z')


# 높이와 방위각을 조절해서 보고 싶은 위치를 조정할 수 있음
```

```
# elevation (높이), azimuth (방위각)
ax1.view_init(elev=27, azim=112)
ax2.view_init(elev=4, azim=114)
ax3.view_init(elev=60, azim=165)

#fig.suptitle('$R^2 = %.2f$' % r2, fontsize=20)

fig.tight_layout()
```

```
9834

# In[24]:

dff.describe()

# 데이터 스케일링 필요

# In[25]:

x = df[['거주인구수', '낮생활인구비율']].values
y = df[['총노인시설수']]

# In[26]:

from sklearn.preprocessing import StandardScaler
```

```
# ## 2) 2D 시각화
# 참고사이트 : https://m.blog.naver.com/tjdrud1323/22172025

# In[27]:
```

```
x = StandardScaler().fit_transform(x)
features=['거주인구수', '낮생활인구비율']
pd.DataFrame(x,columns=features)
```

PCA 실행

In[28]:

```
from sklearn.decomposition import PCA
pca = PCA(n_components=1) # 주성분을 몇개로 할지 결정
printcipalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(data=printcipalComponents, columns = ['principal component1'])
```

In[29]:

```
principalDf.head()
```

In[30]:

```
principalDf['행정동']=df['행정동']
principalDf['총노인시설']=df['총노인시설수']
principalDf
```

In[31]:

```
principalDf.columns=['pca_x', '행정동', '총노인시설']
```

In[32]:

```
x = principalDf['pca_x']
y = principalDf['총노인시설']
```

In[33]:

```
from sklearn.linear_model import LinearRegression
line_fitter = LinearRegression()
line_fitter.fit(x.values.reshape(-1,1), y)
```

```
# In[34]:
```

```
plt.plot(x, y, 'o')
plt.plot(x,line_fitter.predict(x.values.reshape(-1,1)))
plt.show()
```

```
# ### 회귀식에서 멀리 떨어진 동 찾기
```

```
# 회귀식에서 아래쪽으로 멀리 떨어진 동 = 노인인구수, 낮생활인구비율에 비해 노인시설이 적은 것을 뜻함
```

```
# In[35]:
```

```
ff = pd.DataFrame(line_fitter.predict(x.values.reshape(-1,1))-y)
ff['행정동']=df['행정동']
ff
```

```
# In[36]:
```

```
f1 = ff[ff['총노인시설']>0]
f1
```

```
# In[37]:
```

```
f1.describe()
```

In[38]:

```
f2 = f1[f1['총노인시설']>6]['행정동'].tolist()
print('최종 후보 동 : {}'.format(f2))
```

2. 클러스터링

```
#!/usr/bin/env python
# coding: utf-8

# 한글 깨짐
get_ipython().system('sudo apt-get install -y fonts-nanum')
get_ipython().system('sudo fc-cache -fv')
get_ipython().system('rm ~/.cache/matplotlib -rf')
# 설치후 런타임 재시작

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.rc('font', family='NanumBarunGothic')

# 경고 메시지 지우기
import warnings

def warn_func():
    warn_message = 'warn_func() is deprecated, use new_function() instead'
    warnings.warn(warn_message)

warn_func()

# 데이터 불러오기
hjd_df = pd.read_csv('/content/drive/MyDrive/
```

ㅇㅣㅂㅈㅣㅅㅓㄴㅈㅓㅇㄷㅔㅇㅣㅌㅓ
/clustering/ㅋㅡㄹㄹㅓㅅㅡㅌㅓㄹㅣㅇㄷㅔㅇㅣ
ㅌㅓ.csv', encoding='cp949')

hjd_df

hjd_df_plot = hjd_df[['거주인구수','낮생활인구비율','총노인시설수','접근성지표','이용편리지표']]

결측값 확인

hjd_df.isnull().sum()

데이터 분포 확인

color_wheel = {1: "silver",

2: "tomato"}

colors = hjd_df["target"].map(lambda x: color_wheel.get(x + 1))

ax = pd.plotting.scatter_matrix(hjd_df_plot, color=colors, alpha=0.8, s=(hjd_df['target']+0.5)*100, figsize=(12, 12), diagonal='hist')

target 행정동 지정

hjd_df['target'] = 0

target_d = ['부암동', '중곡3동', '전농1동', '망우3동', '돈암1동', '장위2동', '역촌동', '홍제3동', '서림동', '반포본동', '일원1동', '오륜동', '잠실7동', '성내3동']

for i in range(0,len(target_d)):

hjd_df.loc[hjd_df['행정동'] == target_d[i],'target'] = 1

hjd_df[hjd_df['target']==1]

상관계수 행렬

```
hjd_df_cor = hjd_df[['거주인구수','낮생활인구비율','총노인시설수','접근성지표','이용편리지표']]

corr = hjd_df_cor.corr(method = 'pearson')

corr

# 거주인구수 제거

clu_data = hjd_df[['낮생활인구비율','총노인시설수','접근성지표','이용편리지표']]

clu_data.head()

# 정규화

from sklearn.preprocessing import minmax_scale

clu_data_n = minmax_scale(clu_data)

clu_data_n

clu_data['낮생활인구비율'] = [row[0] for row in clu_data_n]

clu_data['총노인시설수'] = [row[1] for row in clu_data_n]

clu_data['접근성지표'] = [row[2] for row in clu_data_n]

clu_data['이용편리지표'] = [row[3] for row in clu_data_n]

clu_data.head()

clu_data = (clu_data*100).round(1)
```

```
clu_data['pca1'] = [row[0] for row in df_pca]
clu_data['pca2'] = [row[1] for row in df_pca]
clu_data['pca3'] = [row[2] for row in df_pca]
```

```
clu_data.head()
```

```
clu_data.shape
```

```
clu_data_pca = clu_data[['pca1','pca2','pca3']]
clu_data_pca.head()
```

```
# ### pca
```

```
from sklearn.decomposition import PCA
pca = PCA(n_components=3)
df_pca = pca.fit_transform(clu_data)
df_pca
```

```
# 차원축소후 분포 확인
color_wheel = {1: "silver",
               2: "tomato"}
colors = hjd_df['target'].map(lambda x: color_wheel.get(x + 1))
ax = pd.plotting.scatter_matrix(clu_data_pca, color=colors, alpha=0.8, s=(hjd_df['target']+0.5)*100, figsize=(8, 8), diagonal='hist')
```

```
warnings.filterwarnings(action='ignore')
```

```
# ## 1. K-means
```

```
# X = clu_data.iloc[:, [0,1,2]].astype("int")
X = clu_data_pca.iloc[:, [0,1,2]].astype("int")
y = hjd_df['target'] # y 값은 군집에서 사용하지 않는다
```

```
from sklearn.cluster import KMeans

model = KMeans(5)
model.fit(X)
```

```
labels = model.labels_
# labels
```

```
plot_df = X
plot_df['y'] = y
plot_df['p'] = labels
plot_df
```

```
plot_df1 = plot_df[plot_df['y']==0]
plot_df2 = plot_df[plot_df['y']==1]
```

```
plot_df_c1 = plot_df[plot_df['p']==0]
plot_df_c2 = plot_df[plot_df['p']==1]
plot_df_c3 = plot_df[plot_df['p']==2]
plot_df_c4 = plot_df[plot_df['p']==3]
plot_df_c5 = plot_df[plot_df['p']==4]
```

```
                                                    c="tomato", s=60)

                                                    # plt.colorbar()



plt.figure(figsize=[12,6])


plt.subplot(1,2,1)

plt.title('target', fontsize=20)                    hjd_df['kmeansP'] = plot_df['p']


plt.scatter(plot_df1['pca1'],       plot_df1['pca2'],
c="silver", s=60)

plt.scatter(plot_df2['pca1'],       plot_df2['pca2'],
c="tomato", s=60)

# plt.colorbar()
                                                    hjd_df[hjd_df['target']==1]


plt.subplot(1,2,2)

plt.title('k-means Clustering', fontsize=20)

plt.scatter(plot_df_c1['pca1'],  plot_df_c1['pca2'],
c="limegreen", s=60)

plt.scatter(plot_df_c2['pca1'],  plot_df_c2['pca2'],
c="gold", s=60)                                     len(hjd_df[hjd_df['kmeansP']==4])

plt.scatter(plot_df_c3['pca1'],  plot_df_c3['pca2'],
c="skyblue", s=60)

plt.scatter(plot_df_c4['pca1'],  plot_df_c4['pca2'],
c="orange", s=60)

plt.scatter(plot_df_c5['pca1'],  plot_df_c5['pca2'],
```

```
len(hjd_df[hjd_df['target']==1])
```

```
hjd_final = hjd_df.loc[(hjd_df['target']==1) & (hjd_df['kmeansP']==4)]

hjd_final.reset_index(drop=True,inplace=True)

hjd_final = hjd_final[['행정동코드','읍면동코드','자치행정동','자치구','행정동']]

hjd_final
```

```
clu_data_s[(clu_data_s['자치행정동'] == hjd_final['자치행정동'][0]) | (clu_data_s['자치행정동'] == hjd_final['자치행정동'][1]) |(clu_data_s['자치행정동'] == hjd_final['자치행정동'][2]) |(clu_data_s['자치행정동'] == hjd_final['자치행정동'][3]) |(clu_data_s['자치행정동'] == hjd_final['자치행정동'][4]) |(clu_data_s['자치행정동'] == hjd_final['자치행정동'][5]) |(clu_data_s['자치행정동'] == hjd_final['자치행정동'][6]) |(clu_data_s['자치행정동'] == hjd_final['자치행정동'][7]) |(clu_data_s['자치행정동'] == hjd_final['자치행정동'][8]) |(clu_data_s['자치행정동'] == hjd_final['자치행정동'][9]) |(clu_data_s['자치행정동'] == hjd_final['자치행정동'][10]) |(clu_data_s['자치행정동'] == hjd_final['자치행정동'][11]) ]
```

```
# ## 2. Hierachical
```

```
X = clu_data.iloc[:,4:]

X
```

```
data = X.values
```

```
import scipy.cluster.hierarchy as shc
```

```
plt.figure(figsize=(10, 7))
plt.title("Hierachical Clustering")
dend = shc.dendrogram(shc.linkage(data, method='ward'))
```

```
from sklearn.cluster import AgglomerativeClustering

cluster = AgglomerativeClustering(n_clusters=5, affinity='euclidean', linkage='ward')
cluster.fit_predict(data)
```

```
plot_df = X
plot_df['y'] = y
plot_df['p'] = cluster.fit_predict(data)
plot_df
```

```
plot_df1 = plot_df[plot_df['y']==0]
plot_df2 = plot_df[plot_df['y']==1]

plot_df_c1 = plot_df[plot_df['p']==0]
plot_df_c2 = plot_df[plot_df['p']==1]
plot_df_c3 = plot_df[plot_df['p']==2]
plot_df_c4 = plot_df[plot_df['p']==3]
plot_df_c5 = plot_df[plot_df['p']==4]
```

```
plt.figure(figsize=[12,6])

plt.subplot(1,2,1)
plt.title('target', fontsize=20)

plt.scatter(plot_df1['pca1'], plot_df1['pca2'], c="silver", s=60)
plt.scatter(plot_df2['pca1'], plot_df2['pca2'],
```

```
c="tomato", s=60)
# plt.colorbar()

plt.subplot(1,2,2)
plt.title('Hierarchical Clustering', fontsize=20)
plt.scatter(plot_df_c1['pca1'], plot_df_c1['pca2'], c="tomato", s=60)
plt.scatter(plot_df_c2['pca1'], plot_df_c2['pca2'], c="limegreen", s=60)
plt.scatter(plot_df_c3['pca1'], plot_df_c3['pca2'], c="orange", s=60)
plt.scatter(plot_df_c4['pca1'], plot_df_c4['pca2'], c="gold", s=60)
plt.scatter(plot_df_c5['pca1'], plot_df_c5['pca2'], c="skyblue", s=60)
# plt.colorbar()

hjd_df['hierarchicalP'] = cluster.fit_predict(data)
hjd_df

hjd_df[hjd_df['target']==1]

hjd_final = hjd_df.loc[(hjd_df['target']==1) & (hjd_df['hierarchicalP']==0)]
hjd_final.reset_index(drop=True,inplace=True)
hjd_final = hjd_final[['행정동코드','읍면동코드','자치행정동','자치구','행정동']]
hjd_final

hjd_final.to_csv("/content/drive/MyDrive/ㅇㅣㅂㅈㅣ선정데이터/clustering/최종행정동.csv",encoding='cp949')
```

```
                                                    from sklearn.cluster import MeanShift


                                                    meanshift= MeanShift(bandwidth=25.974)
                                                    cluster_labels = meanshift.fit_predict(X)
# ## 3. meanshift                                  print('cluster labels 유형:',
                                                    np.unique(cluster_labels))



X = clu_data.iloc[:,4:]
X
                                                    import pandas as pd


                                                    clusterDF = X
                                                    clusterDF['y'] = y
from sklearn.cluster import estimate_bandwidth
                                                    # estimate_bandwidth()로 최적의 bandwidth
                                                    계산
bandwidth = estimate_bandwidth(X)
                                                    best_bandwidth = estimate_bandwidth(X)
print('bandwidth 값:', round(bandwidth,3))


                                                    meanshift= MeanShift(bandwidth=25.974)
                                                    cluster_labels = meanshift.fit_predict(X)
                                                    print('cluster labels 유
                                                    형:',np.unique(cluster_labels))
```

```
clusterDF
```

```
plot_df = X
# plot_df['y'] = y
plot_df['p'] = cluster_labels
plot_df
```

```
plot_df1 = plot_df[plot_df['y']==0]
plot_df2 = plot_df[plot_df['y']==1]

plot_df_c1 = plot_df[plot_df['p']==0]
plot_df_c2 = plot_df[plot_df['p']==1]
plot_df_c3 = plot_df[plot_df['p']==2]
plot_df_c4 = plot_df[plot_df['p']==3]
plot_df_c5 = plot_df[plot_df['p']==4]
```

```
plt.figure(figsize=[10,6])

plt.subplot(1,2,1)
plt.title('target', fontsize=20)

plt.scatter(plot_df1['pca1'], plot_df1['pca2'], c="silver", s=60)
plt.scatter(plot_df2['pca1'], plot_df2['pca2'], c="tomato", s=60)
# plt.colorbar()

plt.subplot(1,2,2)
plt.title('MeanShift Clustering', fontsize=20)
plt.scatter(plot_df_c1['pca1'], plot_df_c1['pca2'], c="tomato", s=60)
plt.scatter(plot_df_c2['pca1'], plot_df_c2['pca2'], c="limegreen", s=60)
plt.scatter(plot_df_c3['pca1'], plot_df_c3['pca2'],
```

```
c="orange", s=60)

plt.scatter(plot_df_c4['pca1'],  plot_df_c4['pca2'],
c="gold", s=60)

plt.scatter(plot_df_c5['pca1'],  plot_df_c5['pca2'],
c="skyblue", s=60)


# plt.colorbar()
```

```
# ## 4. ranking 선정
```

```
hjd_df['MSP'] = plot_df['p']
```

```
clu_data_s=clu_data

clu_data_s['자치행정동'] = hjd_df['자치행정동']
```

```
hjd_final = hjd_df.loc[(hjd_df['target']==1) &
(hjd_df['MSP']==0)]

hjd_final.reset_index(drop=True,inplace=True)

hjd_final = hjd_final[['행정동코드','읍면동코드','
자치행정동','자치구','행정동']]

hjd_final
```

```
clu_data_s['score'] = (clu_data_s['낮생활인구비
율']+clu_data_s['총노인시설수']+clu_data_s['접
근성지표']+clu_data_s['이용편리지표'])/4

clu_data_s
```

```
rank_df.iloc[:,1:]
```

```
clu_data_s[clu_data_s['자치행정동']=='송파구
잠실7동']
```

```
rank_df_n = minmax_scale(rank_df.iloc[:,1:])
rank_df_n
```

```
rank_df = pd.read_csv('/content/drive/MyDrive/
ㅇ ㅣㅂㅈ ㅣㅅ ㅓㄴㅈ ㅓㅇㄷ ㅐㅇ ㅣㅌ ㅓ/행정동랭
킹.csv', encoding='cp949')
rank_df = rank_df.iloc[:7]
rank_df
```

```
sc=[]
for i in range(0,len(rank_df_n)):
sc.append((rank_df_n[i].mean()*100).round(1))
sc
```

```
rank_df =rank_df[['시설명','도로최소거리(m)','모
델가중치','행정동가중치']]
```

```
(rank_df_n[0].mean()*100).round(1)
```

```
rank_df['입지선정지수'] = sc
rank_df = rank_df.sort_values(by=['입지선정지수'],axis=0, ascending=False)
rank_df.reset_index(drop=True,inplace=True)
```

```
rank_df
```

```
# In[155]:
```

```
import numpy as np
from scipy.spatial import distance_matrix
from gurobipy import *
#from scipy.spatial import ConvexHull
#from shapely.geometry import Polygon, Point
from numpy import random
import time
import pandas as pd
from itertools import zip_longest  # zip 함수 길이 다를 경우
from haversine import haversine
```

```
# In[156]:
```

```
#행정동 = '장위2동' # 3개 -> 1
#행정동 = '돈암1동' # 5개 -> 2
#행정동 = '망우3동' # 4개 -> 1
행정동 = '중곡3동' # 2개 -> 1
```

4. MCLP

```
#!/usr/bin/env python
# coding: utf-8
```

```
# In[157]:


입지후보지 = pd.read_csv('수정_최종후보입지.csv', encoding='EUC-KR')

입지후보지 = 입지후보지[입지후보지['행정동'] == 행정동]


버스 = pd.read_csv('입지4_버스.csv', encoding='utf-8')

#버스 = 버스[버스['행정동'].isin(행정동리스트)]

버스_points = np.array([list(i) for i in zip(버스['X좌표'], 버스['Y좌표'])])


지하철 = pd.read_csv('입지4_지하철.csv', encoding='utf-8')

#지하철 = 지하철[지하철['행정동'].isin(행정동리스트)]

지하철_points = np.array([list(i) for i in zip(지하철['경도'], 지하철['위도'])])


주차장 = pd.read_csv('입지4_주차장.csv', encoding='utf-8')

#주차장 = 주차장[주차장['행정동'].isin(행정동리스트)]

주차장_points = np.array([list(i) for i in zip(주차장['경도'], 주차장['위도'])])


주택 = pd.read_csv('입지4_주택.csv', encoding='utf-8')

#주택 = 주택[주택['행정동'].isin(행정동리스트)]

주택_points = np.array([list(i) for i in zip(주택['위도'], 주택['경도'])])  # 위도 경도 바뀜


# In[158]:


X = list(버스['X좌표']) + list(지하철['경도']) + list(주차장['경도']) + list(주택['위도'])  # 주택 위도 경도 이름 바뀜

Y = list(버스['Y좌표']) + list(지하철['위도']) + list(주차장['위도']) + list(주택['경도'])


# In[159]:


points = np.array([list(i) for i in zip(X, Y)])


# In[160]:
```

```
전체w = points.shape[0]
버스w = 버스.shape[0]
지하철w = 지하철.shape[0]
주차장w = 주차장.shape[0]
주택w = 주택.shape[0]


# In[161]:


# 가중치 by AHP 분석(데이터 불균형 * 노인 통행수단 선호도)
m1 = (전체w-버스w)/전체w * 0.278
m2 = (전체w-지하철w)/전체w * 0.136
m3 = (전체w-주차장w)/전체w * 0.193
m4 = (전체w-주택w)/전체w * 0.392


# In[162]:


# haversine -> meter 단위로 수정
def mclp3(버스_points, 지하철_points, 주차장_points, 주택_points, points, K, radius):
    """
    Solve maximum covering location problem
    Input:
        points: 버스정류장, 지하철역 위치 좌표 등 (기타 인근에 있으면 좋은 시설 좌표)
        K: 배치할 노인놀이터의 수
        radius: 반경 (노인들이 이동하기 적합한 거리)
        M: generate_candidate_sites 함수에서 생성할 random 좌표 수 (임의의 노인놀이터 수)
        the ConvexHull wrapped by the polygon
    Return:
        opt_sites: locations K optimal sites, Numpy array in shape of [K,2]
        f: the optimal value of the objective function
    """

    print('    Number of points %g' % points.shape[0])
    print('    K %g' % K)
    print('    Radius %g' % radius)

    start = time.time()
    sites = np.array([list(i) for i in zip(입지후보지['x좌표'], 입지후보지['y좌표'])])
```

```
    J               =               sites.shape[0]
# 후보지 수


    # 수요지점 수
    A = 버스_points.shape[0]
    B = 지하철_points.shape[0]
    C = 주차장_points.shape[0]
    D = 주택_points.shape[0]


    # 후보지와 수요지점 간 거리 계산
    D1 = []
    for i in 버스_points:
        site = []
        for j in sites:
            site.append(haversine(i, j)*1000)
        D1.append(site)
    D1 = np.array(D1)


    D2 = []
    for i in 지하철_points:
        site = []
        for j in sites:
            site.append(haversine(i, j)*1000)
        D2.append(site)
    D2 = np.array(D2)
    D3 = []
    for i in 주차장_points:
        site = []
        for j in sites:
            site.append(haversine(i, j)*1000)
        D3.append(site)
    D3 = np.array(D3)


    D4 = []
    for i in 주택_points:
        site = []
        for j in sites:
            site.append(haversine(i, j)*1000)
        D4.append(site)
    D4 = np.array(D4)


    for i in [D1, D2, D3, D4]:
        mask1 = i<=radius
        i[mask1]=1
# 반경 내 속하면 1, 아니면 0
        i[~mask1]=0


    m = Model()
    x1, x2, x3, x4 = {}, {}, {}, {}
```

```
    y = {}


    # 수요지점 변수 추가
    for i in range(A):
        x1[i] = m.addVar(vtype=GRB.BINARY, name="x1%d" % i)
    for i in range(B):
        x2[i] = m.addVar(vtype=GRB.BINARY, name="x2%d" % i)
    for i in range(C):
        x3[i] = m.addVar(vtype=GRB.BINARY, name="x3%d" % i)
    for i in range(D):
        x4[i] = m.addVar(vtype=GRB.BINARY, name="x4%d" % i)


    for j in range(J):
        y[j] = m.addVar(vtype=GRB.BINARY, name="y%d" % j)       # 후보지 변수 추가


    m.update()
    m.addConstr(quicksum(y[j] for j in range(J)) == K)          # 후보지 제약 조건


    # 수요지점 제약 조건
    for i in range(A):
        m.addConstr(quicksum(y[j] for j in np.where(D1[i]==1)[0]) >= x1[i])
    for i in range(B):
        m.addConstr(quicksum(y[j] for j in np.where(D2[i]==1)[0]) >= x2[i])
    for i in range(C):
        m.addConstr(quicksum(y[j] for j in np.where(D3[i]==1)[0]) >= x3[i])
    for i in range(D):
        m.addConstr(quicksum(y[j] for j in np.where(D4[i]==1)[0]) >= x4[i])


    # 목적함수 수정
    res=[]
    for a,b,c,d in zip_longest(range(A),range(B),range(C),range(D), fillvalue=0):
        w1=m1;w2=m2;w3=m3;w4=m4
        if a==b==c==d==0:
            w1=m1;w2=m2;w3=m3;w4=m4
        else:
            if b==0:
                w2=0
            if c==0:
                w3=0
            if d==0:
                w4=0
```

```
        res.append(w1*x1[a] + w2*x2[b] + w3*x3[c] + w4*x4[d])

    m.setObjective(quicksum(i for i in res),GRB.MAXIMIZE)

    m.setParam('OutputFlag', 0)
    m.optimize()
    end = time.time()
    print('----- Output -----')
    print('  Running time : %s seconds' % float(end-start))
    print('  Optimal coverage points: %g' % m.objVal)

    solution = []
    if m.status == GRB.Status.OPTIMAL:
        for v in m.getVars():
            # print v.varName,v.x
            if v.x==1 and v.varName[0]=="y":
                solution.append(int(v.varName[1:]))
    opt_sites = sites[solution]
    return opt_sites,m.objVal
```

```
# In[163]:


opts_sites, mobjVal = mclp3(버스_points, 지하철_points, 주차장_points, 주택_points, points, 1, 300)

opts_sites
```

```
# In[164]:


후보지 = pd.DataFrame(opts_sites, columns=['경도','위도'])

후보지.to_csv('중곡3동_후보지.csv', index=False, encoding='cp949')

수요지점 = pd.DataFrame(points, columns=['경도','위도'])

수요지점.to_csv('중곡3동_수요지점.csv', index=False, encoding='cp949')

버스_수요지점 = pd.DataFrame(버스_points, columns=['경도','위도'])

버스_수요지점.to_csv('중곡3동_버스수요지점.csv', index=False, encoding='cp949')
```

```
지하철_수요지점 = pd.DataFrame(지하철_points, columns=['경도','위도'])

지하철_수요지점.to_csv('중곡3동_지하철수요지점.csv', index=False, encoding='cp949')

주차장_수요지점 = pd.DataFrame(주차장_points, columns=['경도','위도'])

주차장_수요지점.to_csv('중곡3동_주차장수요지점.csv', index=False, encoding='cp949')

주택_수요지점 = pd.DataFrame(주택_points, columns=['경도','위도'])

주택_수요지점.to_csv('중곡3동_주택수요지점.csv', index=False, encoding='cp949')
```

5. P-median

```
#!/usr/bin/env python

# coding: utf-8

# In[1]:

import numpy as np

import pandas as pd

# In[27]:

#행정동 = '장위2동' # 3개 -> 1

#행정동 = '돈암1동' # 5개 -> 2

행정동 = '망우3동' # 4개 -> 1

#행정동 = '중곡3동' # 2개 -> 1

# In[28]:

입지후보지 = pd.read_csv('수정_최종후보입지.csv', encoding='EUC-KR')

입지후보지 = 입지후보지[입지후보지['행정동'] == 행정동]

후보지_points = np.array([list(i) for i in zip(입지후보지['x좌표'], 입지후보지['y좌표'])])

버스 = pd.read_csv('입지3_버스.csv', encoding='utf-8')

#버스 = 버스[버스['행정동'].isin(행정동리스트)]

버스_points = np.array([list(i) for i in zip(버스['X좌표'], 버스['Y좌표'])])
```

```
지하철 = pd.read_csv('입지3_지하철.csv', encoding='utf-8')

#지하철 = 지하철[지하철['행정동'].isin(행정동리스트)]

지하철_points = np.array([list(i) for i in zip(지하철['경도'], 지하철['위도'])])


주차장 = pd.read_csv('입지3_주차장.csv', encoding='utf-8')

#주차장 = 주차장[주차장['행정동'].isin(행정동리스트)]

주차장_points = np.array([list(i) for i in zip(주차장['경도'], 주차장['위도'])])


주택 = pd.read_csv('입지3_주택.csv', encoding='utf-8')

#주택 = 주택[주택['행정동'].isin(행정동리스트)]

주택_points = np.array([list(i) for i in zip(주택['위도'], 주택['경도'])])  # 위도 경도 바뀜
```

```
# In[29]:


X = list(버스['X좌표']) + list(지하철['경도']) + list(주차장['경도']) + list(주택['위도'])  # 주택 위도 경도 이름 바뀜

Y = list(버스['Y좌표']) + list(지하철['위도']) + list(주차장['위도']) + list(주택['경도'])


# In[30]:


points = np.array([list(i) for i in zip(X, Y)])


# In[31]:


전체w = points.shape[0]

버스w = 버스.shape[0]

지하철w = 지하철.shape[0]

주차장w = 주차장.shape[0]

주택w = 주택.shape[0]


# In[32]:


# 가중치 by AHP 분석 (데이터 불균형 * 통행수단 선호도)
```

m1 = (전체w-버스w)/전체w * 0.278

m2 = (전체w-지하철w)/전체w * 0.193

m3 = (전체w-주차장w)/전체w * 0.136

m4 = (전체w-주택w)/전체w * 0.392

In[33]:

```
from haversine import haversine

havers = []
for i in 후보지_points:
    site = []
    for j in 버스_points:
        site.append(haversine(i,j))
    havers.append(site)
```

버스

In[34]:

```
from pulp import *

# 후보지와 버스정류장 간 거리행렬 생성
location = list(입지후보지['시설명'])
location2 = list(버스['정류소명'])

havers_D = dict(zip(location, [dict(zip(location2, i)) for i in havers]))
D = havers_D
```

In[35]:

```
#col = list(D.keys())
D = pd.DataFrame(D)
D.head()
```

In[36]:

```
# #col = D.columns
최소값 = list(D.min(axis=1))
```

```
# In[37]:


# 버스
idx = D.index
col = D.columns
lidx = len(idx)
lcol = len(col)

for i in range(lidx):
    for j in range(lcol):
        if D.loc[idx[i], col[j]] == 최소값[i]:
            D.loc[idx[i], col[j]] = m1
        else:
            D.loc[idx[i], col[j]] = 0


# # 지하철

# In[38]:


from pulp import *

# 후보지와 버스정류장 간 거리행렬 생성
location = list(입지후보지['시설명'])
location2 = list(지하철['전철역명'])

havers_D2 = dict(zip(location, [dict(zip(location2, i)) for i in havers]))
D2 = havers_D2


# In[39]:


#col = list(D.keys())
D2 = pd.DataFrame(D2)
D2.head()


# In[40]:


# #col = D.columns
최소값 = list(D2.min(axis=1))


# In[41]:
```

```
# 지하철
idx = D2.index
col = D2.columns
lidx = len(idx)
lcol = len(col)

for i in range(lidx):
    for j in range(lcol):
        if D2.loc[idx[i], col[j]] == 최소값[i]:
            D2.loc[idx[i], col[j]] = m2
        else:
            D2.loc[idx[i], col[j]] = 0


# # 주차장

# In[42]:


from pulp import *

# 후보지와 버스정류장 간 거리행렬 생성
location = list(입지후보지['시설명'])
location2 = list(주차장['주차장이름'])
havers_D3 = dict(zip(location, [dict(zip(location2, i)) for i in havers]))
D3 = havers_D3


# In[43]:


#col = list(D.keys())
D3 = pd.DataFrame(D3)
D3.head()


# In[44]:


# #col = D.columns
최소값 = list(D3.min(axis=1))


# In[45]:


# 주차장
```

```
idx = D3.index
col = D3.columns
lidx = len(idx)
lcol = len(col)

for i in range(lidx):
    for j in range(lcol):
        if D3.loc[idx[i], col[j]] == 최소값[i]:
            D3.loc[idx[i], col[j]] = m3
        else:
            D3.loc[idx[i], col[j]] = 0
```

```
# # 주택
```

```
# In[46]:
```

```
from pulp import *
```

```
# 후보지와 버스정류장 간 거리행렬 생성
location = list(입지후보지['시설명'])
location2 = list(주택['단지명'])

havers_D4 = dict(zip(location, [dict(zip(location2, i)) for i in havers]))
D4 = havers_D4
```

```
# In[47]:
```

```
#col = list(D.keys())
D4 = pd.DataFrame(D4)
D4.head()
```

```
# In[48]:
```

```
# #col = D.columns
최소값 = list(D4.min(axis=1))
```

```
# In[49]:
```

```
# 주택
idx = D4.index
col = D4.columns
```

```
lidx = len(idx)
lcol = len(col)


for i in range(lidx):
    for j in range(lcol):
        if D4.loc[idx[i], col[j]] == 최소값[i]:
            D4.loc[idx[i], col[j]] = m4
        else:
            D4.loc[idx[i], col[j]] = 0



# # Final


# In[50]:


D_final = pd.concat([D, D2, D3, D4])
```

```
# In[51]:


D_final.sum()


# * (3개 중 1위) 장위2동 - 장위2동 감나무쉼터 - 127.051625  37.614919
# * (4개 중 1위) 돈암1동 - 돈암1동 마을마당 - 127.021175  37.597746
# * (4개 중 2위) 돈암1동 - 새소리 - 127.025559  37.600827
# * (3개 중 1위) 망우3동 - 달동산 - 127.0952  37.59217
# * (2개 중 1위) 중곡3동 - 배나무터공원 - 127.0802718  37.56833955
```