2021년 공공 빅데이터 분석 청년 인재양성 데이터 분석 전문교육과정

데이터기반 행정으로 국민의 삶의질을 개선하라!

데이터딘십 해커톤 제 4회

분석 결과 보고서

전주시 아동급식카드 가맹점 실태파악 및 정책 보완 재고

참여조: 전라권 153조

참여자: 최성렬(조장)
김경민
김대근
김선희
조예진
황산하

씨에스리 컨소시엄
CSLEE kpc 한국생산성본부

2021년 공공 빅데이터 분석청년 인재양성 - 실무형프로젝트 결과보고서

목 차

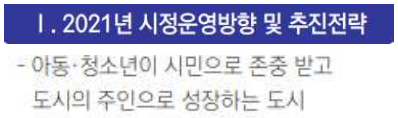
그림 목차

표 목차

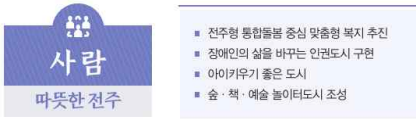
1. 분석 개요

1.1 분석 배경 및 개요

1.1.1 아동급식관련 복지 정책 현황



[그림 1-1] 전주시 2021 시정과제 (전주시청)

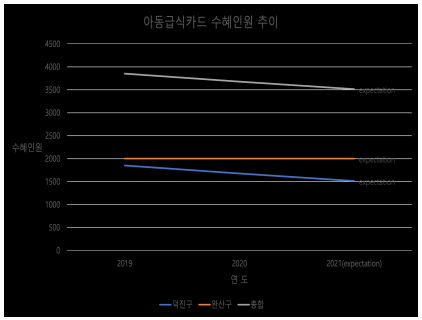


[그림 1-2] 전주시 2020 시정과제 (전주시청)

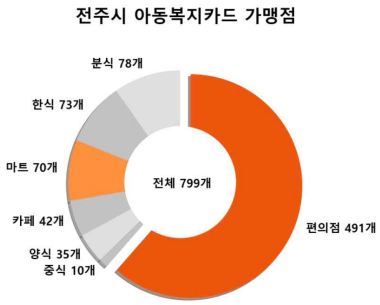
□ 전주시는 아동의 삶 증진을 주요 시정과제로 선정하여 복지 확대에 많은 노력을 기울이고 있음

□ 2021년 한 끼 당 급식단가를 5500원->6000원으로 인상하여 양질의 식사 제공 추진

1.1.2 아동급식카드 사용 및 가맹 현황



[그림1-3] 전주시 아동급식카드 수혜인원 추이



[그림 1-4] 전주시 아이급식카드 가맹점

□ 2021년(추정) 전주시 내 아동급식카드 수혜 인원은 3500명 정도

□ 2021년 4월 기준 아동급식카드를 사용할 수 있는 전주시 내 가맹점은 799개소

1.1.3 아동급식카드 가맹점 확대의 필요성과 현 정책의 문제점

□ 결식 아동들이 이용할 수 있는 음식점은 아동급식카드 가맹점으로 한정되어 있고 가맹점의 분포는 특정지역에만 편향되어 확대 필요

□ 주 활동 반경이 가맹점 밀집지역과 거리가 먼 수혜 아동들은 음식점 선택에 제약을 크게 받아 활동 반경을 고려한 해결방안 요망

□ 가맹점 중 편의점과 마트의 비중이 70%를 차지함. 이 중 편의점 매출 비율은 식사 품목 보다 음료와 간식 비율이 높아 양질의 영양소 섭취가 가능한 선택지 필요

□ 현 정책의 한 끼 당 지원금액이 물가를 반영하지 못하여 적절한 지원금액 산정 필요

1.2 분석 목적 및 방향

□ 정책 수혜자인 아동의 행동반경을 기반으로 공간적, 통계적 분석을 통해 실태 점검 및 해결방안 도출

□ 특정지역에 편향된 가맹점의 분포를 점검하고 이를 통해 취약지역을 선정, 이후 전체 가맹점 확대 시의 변화를 분석하여 가맹점 확대 필요성 재고

□ 전체 가맹점 확대 시 효율성 분석

□ 기존 정책과 가맹점 확대 시 유의미한 한 끼 당 지원금액을 비교하고 적절 지원금액 산출

1.3 분석 결과 활용 방안

□ 아동 활동반경내 접근 가능한 음식점 개수의 변화 추이로 정책 방향을 재고함

□ 전주시 복지 카드 가맹점 현황 데이터를 이용하여 현행 정책의 취약점을 공간적으로 분석하여 취약지역 도출

□ 군집분석을 기반으로 한 접근성 지수 상위, 하위지역의 유사성을 분석하고, 이를 통해 시범사업지역을 제안함

□ 우수 모델을 전주시에 적용하는 경우와 사용한도를 증가시키는 경우의 효과를 통계적으로 검증함. 해당 시뮬레이션 결과를 이용하여 정책 제안

2. 분석 데이터

2.1 분석 데이터 목록

[표 1] 분석 데이터 목록

구분	분석 데이터	기간	제공기관
환경 변수	전국 초중등학교 위치 표준데이터	2021.03	공공데이터포털 (한국교원대학교) (https://www.data.go.kr)
	전주시 법정동별 초등학생 인구수	2021.04	국토정보플랫폼 (행정안전부) (http://map.ngii.go.kr)
	전주시 법정동별 중학생 인구수	2021.04	국토정보플랫폼 (행정안전부) (http://map.ngii.go.kr)
	전주시 법정동별 고등학생 인구수	2021.04	국토정보플랫폼 (행정안전부) (http://map.ngii.go.kr)
	전주시 법정동별 유아 인구수	2021.04	국토정보플랫폼 (행정안전부) (http://map.ngii.go.kr)
	전주시 법정동별 유소년 인구수	2021.04	국토정보플랫폼 (행정안전부) (http://map.ngii.go.kr)
	전주시 법정동별 공동건물 개수	2021.04	국토정보플랫폼 (행정안전부) (http://map.ngii.go.kr)
	전주시 법정동별 단독건물 개수	2021.04	국토정보플랫폼 (행정안전부) (http://map.ngii.go.kr)
	전주시 개별공시지가	2020.11	공공데이터포털 (전라북도 전주시) (https://www.data.go.kr)
접근성 변수	전주시 음식점 정보	2021.02	공공데이터포털 (전라북도 전주시) (https://www.data.go.kr)
	전주시 음식점 메뉴 정보	2021.02	공공데이터포털 (전라북도 전주시) (https://www.data.go.kr)
	전주시 아이복지카드 가맹점 정보	2021.07	크롤링 (https://www.purmeecard.com/public.do?request=merchantSelectFormNew)

1)

1) 법정동별 인구수 데이터 정제 과정에서 법정동 코드 자료를 이용하여 법정동 이름을 지정함(행정표준코드관리시스템), 전주시 법정동 SHP 파일은 전주시 법정동 별 인구수 데이터에서 추출함

2.2 데이터 상세 설명

[표 2] 분석 데이터 상세 설명

구분	분석 데이터	데이터 형식	생성주기
공공 데이터	전국 초중등학교 위치 표준데이터	CSV	수시
	전주시 법정동별 초등학생 인구수	SHP	6개월
	전주시 법정동별 중학생 인구수	SHP	6개월
	전주시 법정동별 고등학생 인구수	SHP	6개월
	전주시 법정동별 유아 인구수	SHP	6개월
	전주시 법정동별 유소년 인구수	SHP	6개월
	전주시 법정동별 공동건물 개수	SHP	월간
	전주시 법정동별 단독건물 개수	SHP	월간
	전주시 개별공시지가	CSV	연간
	전주시 음식점 정보	CSV	1회
	전주시 음식점 메뉴 정보	CSV	1회
웹 크롤링 데이터	전주시 아이복지카드 가맹점 정보	CSV	수시

2.2.1 전국 초중등학교 위치 표준데이터 [.CSV]

□ 공공데이터포털(한국교원대학교)에서 제공하는 자료로 전국 초등・중등・고등학교 위치 정보 표준 데이터 학교명, 학교급구분, 운영상태, 주소, 위도, 경도 등을 포함

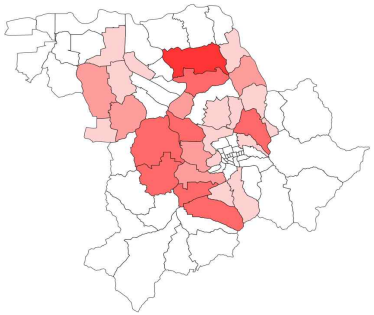
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	학교ID	학교명	학교급구분	설립일자	설립형	본교분	운영상	소재지지번	소재지도로	시도교육청	교육지원청	운영상	생성일자	변경일자	위도	경도	데이터기준일	제공기관코	제공기관명	
2	B0000133	진주외국어	고등학교	1970-12-26	사립	본교	운영	경상남도 진	경상남도 진	9010000	경상남도교	9051000	경상남도진	2013-11-29	2020-09-22	35.16966	128.2817	2021-03-25	7001220	한국교원대학교
3	B0000229	용남고등학	고등학교	1991-03-09	공립	본교	운영	충청남도 논	충청남도 논	8140000	충청남도교	8222000	충청남도논	2013-11-29	2020-09-22	36.29431	127.2448	2021-03-25	7001220	한국교원대학교
4	B0000109	장곡중학교	중학교	2001-03-06	공립	본교	운영	경상북도 경	경상북도 경	8750000	경상북도교	8941000	경상북도경	2013-11-29	2020-09-22	36.07346	128.4151	2021-03-25	7001220	한국교원대학교
5	B0000112	대정중학교	중학교	1946-11-08	공립	본교	운영	제주특별자	제주특별자	9290000	제주특별자	9299000	제주특별자	2013-11-29	2020-09-22	33.21996	126.2532	2021-03-25	7001220	한국교원대학교
6	B0000081	안천중학교	중학교	1991-03-01	공립	본교	운영	서울특별시	서울특별시	7010000	서울특별시	7041000	서울특별시	2013-11-29	2020-09-22	37.45952	126.8875	2021-03-25	7001220	한국교원대학교
7	B0000033	대전동광초	초등학교	1936-04-13	공립	본교	운영	대전광역시	대전광역시	7430000	대전광역시	7441000	대전광역시	2013-11-29	2020-09-22	36.33735	127.4496	2021-03-25	7001220	한국교원대학교
8	B0000127	봉일천고등	고등학교	2007-03-06	공립	본교	운영	경기도 파주	경기도 파주	7530000	경기도교육	7681000	경기도파주	2013-11-29	2020-09-22	37.74699	126.8163	2021-03-25	7001220	한국교원대학교
9	B0000095	안성중학교	중학교	1939-03-24	공립	본교	운영	경기도 안성	경기도 안성	7530000	경기도교육	7761000	경기도안성	2013-11-29	2020-09-22	37.01331	127.2705	2021-03-25	7001220	한국교원대학교
10	B0000021	서울잠일초	초등학교	1976-03-12	공립	본교	운영	서울특별시	서울특별시	7010000	서울특별시	7130000	서울특별시	2013-11-29	2020-09-22	37.51474	127.0816	2021-03-25	7001220	한국교원대학교
11	B0000084	태중대중학	중학교	1974-12-21	공립	본교	운영	부산광역시	부산광역시	7150000	부산광역시	7171000	부산광역시	2013-11-29	2020-09-22	35.07367	129.0649	2021-03-25	7001220	한국교원대학교

[그림 2-1] 전국 학교 데이터 상세

2.2.2 전주시 법정동별 초등·중등·고등학생 인구수, 유아·유소년 인구수, 공동·단독건물수 [.SHP]

□ 국토교통부 국토정보플랫폼에서 제공하는 자료로 법정동별 초등·중등·고등학생 인구수, 유아·유소년 인구수, 공동·단독건물수 정보를 포함. 생성 주기는 인구수 6개월, 건물수 1개월

□ 유형별로 상이한 파일이나 형태가 동일하므로 본 장에서는 초등학생 인구수와 공동건물수 데이터를 대표로 나타냄

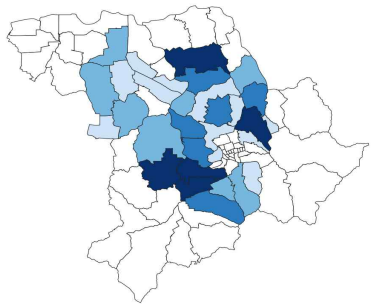


[그림 2-2] 전주시 법정동별 초등학생 인구수 Choropleth Map

	gid	lbl	val
1	45111141	3210.00	3210.00000000...
2	45111110	17.00	17.00000000000...
3	45111122	43.00	43.00000000000...
4	45111102	2.00	2.00000000000...
5	45113121	2601.00	2601.00000000...
6	45113116	1018.00	1018.00000000...
7	45113136	4.00	4.00000000000...
8	45111144	6.00	6.00000000000...
9	45113114	312.00	312.00000000...
10	45111142	2111.00	2111.00000000...

[그림 2-3] 전주시 법정동별 초등학생 인구수 속성테이블 데이터 상세[2)]

2) 속성테이블 gid열은 법정동 코드, val열은 초등학생 인구수를 나타냄



[그림 2-4] 전주시 법정동별 공동건물 수 데이터 상세

	gid	lbl	val
1	45111141	323.00	323.00000000...
2	45111110	1.00	1.00000000000...
3	45111122	12.00	12.00000000...
4	45113121	273.00	273.00000000...
5	45113116	114.00	114.00000000...
6	45113114	69.00	69.00000000...
7	45111142	144.00	144.00000000...
8	45113129	14.00	14.00000000...
9	45113109	49.00	49.00000000...
10	45111124	22.00	22.00000000...

[그림 2-5] 전주시 법정동별 공동건물 수 속성테이블 데이터 상세[3)]

2.2.3 전주시 개별공시지가 [.CSV]

□ 공공데이터포털(전라북도 전주시)에서 제공하는 자료로 전주시 내 법정동별 공시지가, 토지구분, 기준년도, 기준일, 표준지여부 등의 항목을 포함

	A	B	C	D	E	F	G	H	I	J	K	L
1	지번주소코	시군구코드	시군구	읍면동코드	읍명동	지번구분	본번	부번	기준년도	기준월	공시지가	표준지여부
2	4.51E+18	45111	완산구	101	중앙동1가	1	1	1	2020	1	599,700	N
3	4.51E+18	45111	완산구	101	중앙동1가	1	1	4	2020	1	1,257,000	N
4	4.51E+18	45111	완산구	101	중앙동1가	1	1	5	2020	1	1,120,000	Y
5	4.51E+18	45111	완산구	101	중앙동1가	1	1	7	2020	1	605,800	N
6	4.51E+18	45111	완산구	101	중앙동1가	1	2	1	2020	1	1,097,000	N
7	4.51E+18	45111	완산구	101	중앙동1가	1	2	3	2020	1	1,097,000	N
8	4.51E+18	45111	완산구	101	중앙동1가	1	3	3	2020	1	1,108,000	N
9	4.51E+18	45111	완산구	101	중앙동1가	1	3	5	2020	1	2,361,000	N
10	4.51E+18	45111	완산구	101	중앙동1가	1	4	1	2020	1	1,465,000	N
11	4.51E+18	45111	완산구	101	중앙동1가	1	5	3	2020	1	396,500	N

[그림 2-6] 전주시 공시지가 데이터 상세

2.2.4 전주시 음식점 정보 [.CSV]

□ 공공데이터포털(전라북도 전주시)에서 제공하는 자료로 전주시 내 일반·휴게음식점을 대상으로 조사한 식당ID, 식당명, 업종, 위도, 경도, 다국어 정보 등의 자료를 포함

3) 속성테이블 gid열은 법정동 코드, val열은 공동건물 수를 나타냄

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1	식당ID	식당명	업종(메뉴	테이크아	예약가능	선결제여	식당 대표	도로명주	지번주소	정규휴업	식당면적	식당위도	식당경도	대표메뉴	어워드_글	어워드_로	트립어드	네이버 인	KTI 인덱	온라인화	다국어메	다국어 지	다국어메	
2	1606	피자애 송	피자	Y	Y	N	063-252-4	전라북도	전라북도	연중무휴	0	35.86165	127.1263	불고기피자, 고구마피자			0	0	0	0 Y	Y	Y	zh-hans	http://red
3	2819	일등낙지	낙지	Y	Y	N	063-222-5	전라북도	전주시 완산		0	35.81612	127.1083	낙삼볶음, 불낙볶음, 연포탕, 낙			0	0	5	2.34 Y	Y	Y	zh-hans	http://red
4	2999	비공식	일식	N	N	N		전라북도	전주시 완산		0	35.81855	127.1393				0	0	0.68	1.6 Y	Y	Y	zh-hans	http://red
5	3110	짝태포차	맥주	Y	Y	N		전라북도	전라북도	일요일	0	35.81046	127.1067	짝태,짝태	면보사,골	뱅이무침	0	0	0	0 Y	Y	Y	zh-hans	http://red
6	4465	엘디아	카페	N	N	N	063-253-0	전라북도	전주시 덕진		0	35.86646	127.1336				0	0	0	0 Y	Y	Y	zh-hans	http://red
7	4578	자담치킨	치킨	Y	N	N	063-276-9	전라북도	전주시 덕진		135.21	35.86201	127.1144				0	0	0	0 Y	Y	Y	zh-hans	http://red
8	5127	뷸른통갈비 앤 치킨		N	N	N	063-272-9	전라북도	전주시 덕진		0	35.85716	127.1188				0	0	0	0 Y	Y	Y	zh-hans	http://red
9	6050	라임그린카페		Y	Y	N	063-271-2	전라북도	전라북도	첫째주 일	0	35.86248	127.126	아메리카노			0	0	0	0 Y	Y	Y	zh-hans	http://red
10	6119	객주	맥주	Y	Y	N		전라북도	전라북도	전주시 완산	43.68	35.81831	127.1498	시래기국밥, 막걸비			0	0	0	0 Y	Y	Y	zh-hans	http://red
11	6427	아웃닭 전	치킨	Y	Y	N	063-255-9	전라북도	전라북도	연중무휴	0	35.84255	127.128	치킨			0	0	0	0 Y	Y	Y	zh-hans	http://red

[그림 2-7] 전주시 음식점 데이터 상세

2.2.5 전주시 음식점 정보 [.CSV]

□ 공공데이터포털(전라북도 전주시)에서 제공하는 자료로 전주시 내 일반·휴게음식점을 대상으로 조사한 식당ID, 식당명, 메뉴명, 메뉴가격, 다국어 메뉴명 등의 정보를 포함

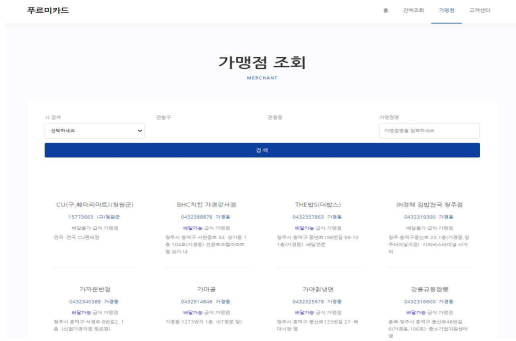
	A	B	C	D	E	F	G	H	I	J	K	L
1	식당ID	식당명	메뉴ID	메뉴명	메뉴가격	메뉴태그정	메뉴명-로(	메뉴태그-(	메뉴태그-(	메뉴태그-중국어(간체)		
2	294795	청년다방 ㅈ	70135	불향차돌서	23500	옵션 : 세트	Bulhyang (	Etc : Set	その他 : セ	其他 : 套餐		
3	294795	청년다방 ㅈ	70136	불향차돌서	26500	옵션 : 세트	Bulhyang (	Etc : Set	その他 : セ	其他 : 套餐		
4	294795	청년다방 ㅈ	70137	통큰오짱서	23500	옵션 : 세트	Tongkeun	Etc : Set	その他 : セ	其他 : 套餐		
5	294795	청년다방 ㅈ	70138	통큰오짱서	26500	옵션 : 세트	Tongkeun	Etc : Set	その他 : セ	其他 : 套餐		
6	294795	청년다방 ㅈ	70139	치믈랫세트	24000	옵션 : 세트	Chimeullet	Etc : Set	その他 : セ	其他 : 套餐		
7	294795	청년다방 ㅈ	70140	치믈랫세트	27000	옵션 : 세트	Chimeullet	Etc : Set	その他 : セ	其他 : 套餐		
8	294795	청년다방 ㅈ	70141	깻잎순대서	22500	주재료 : 돠	Kkaennip (	Ingredient	主食材 : 豚	主料 : 猪肉,血肠,芝麻 / 其他 : 套餐		
9	294795	청년다방 ㅈ	70142	깻잎순대서	25500	주재료 : 돠	Kkaennip (	Ingredient	主食材 : 豚	主料 : 猪肉,血肠,芝麻 / 其他 : 套餐		
10	294795	청년다방 ㅈ	70143	빨간크림서	23000	옵션 : 세트	Ppalgan Cı	Etc : Set	その他 : セ	其他 : 套餐		
11	294795	청년다방 ㅈ	70144	빨간크림서	26000	옵션 : 세트	Ppalgan Cı	Etc : Set	その他 : セ	其他 : 套餐		

[그림 2-8] 전주시 음식점 메뉴 데이터 상세

2.2.6 전주시 음식점 정보 [.CSV]

□ 아동급식카드[4] 사이트(https://www.purmeecard.com/index5.jsp)에서 제공하는 전주시 아동급식카드 가맹점 정보를 웹 크롤링 방식을 통해 추출

4) 푸르미 코리아 : 지자체와 협약하여 아동급식카드를 운영하는 대행사



[그림 2-9] 푸르미 코리아 사이트

	A	B	C	D
1	No		m_name	address
2	1	0	CU (구,훼미리마	전라북도 전주시 덕진구 금암1동 기린대로 398
3	2	1	금암면옥	전주시 덕진구 기린대로 400-3
4	3	2	대성고기백화점	전라북도 전주시 덕진구 태진로 137-4
5	4	3	또또분식	전주시 덕진구 삼송3길 14
6	5	4	롯데리아 전주터	전주시 덕진구 가리내로 21
7	6	5	바른카츠	전주시 덕진구 삼송5길 14-23
8	7	6	본도시락 전북대	전라북도 전주시 덕진구 조경단로 83
9	8	7	본설렁탕령탕 전	전라북도 전주시 덕진구 전주천동로 490
10	9	8	슈퍼.Super	전주시 덕진구 금암동 전주천동로 493
11	10	9	신김밥천국	전주시 덕진구 조경단로 107

[그림 2-10] 아동급식카드 가맹점 웹 크롤링 데이터 상세

2.3 데이터 정제 방안

2.3.1 전주시 법정동별 초·중·고등학생 인구수, 총인구수, 유소년·유아인구수, 공동·단독건물수 데이터

2.3.1.1 원본 데이터

[표 3] 2.3.1.1 원본 데이터

이름	설명	출처
전주시 법정동별 초등학교 인구수	전주시를 법정동별로 나눠 각 법정동의 초·중·고등학생 인구수, 총인구수, 유소년·유아인구수, 공동·단독건물 수를 나타냄	국토정보 플랫폼
전주시 법정동별 중학교 인구수		
전주시 법정동별 고등학교 인구수		
전주시 법정동별 총인구수		
전주시 법정동별 유소년 인구수		
전주시 법정동별 유아 인구수		
전주시 법정동별 공동건물 수		
전주시 법정동별 단독건물 수		

2.3.1.2 정제 과정

□ 속성테이블의 법정동 코드를 기준으로 초등학교 인구수, 중학교 인구수, 고등학교 인구수, 총인구수, 유소년 인구수, 유아 인구수, 공동건물수, 단독건물수 데이터를 합친 후 CSV 파일로 저장

□ 행정 표준 코드관리시스템에서 제공하는 법정동 코드 목록을 이용하여 전주시 법정동 코드 목록을 추출

□ 전주시 법정동 코드 목록을 이용하여 앞서 저장한 CSV 파일의 법정동 코드를 통해 법정동명 지정

	A	B	C	D	E	F	G	H	I	J
1		법정동이름	공동건물수	단독건물수	총인구	유소년인구	고등학생수	중학생수	초등학생수	유아인
2	0	중앙동1가	1	29	226	20	7	8	15	
3	1	중앙동2가	0	9	116	4	1	1	2	
4	2	중앙동3가	0	8	109	6	2	0	5	
5	3	중앙동4가	0	58	388	17	10	4	16	
6	4	경원동1가	2	21	223	5	2	2	2	
7	5	경원동2가	2	46	188	7	3	3	2	
8	6	경원동3가	2	143	539	13	8	6	6	
9	7	풍남동1가	3	78	239	8	5	3	3	
10	8	풍남동2가	0	78	152	7	0	3	3	
11	9	풍남동3가	1	268	485	30	6	11	17	

[그림 2-11] 전주시 법정동별 인구수 정제 데이터 상세

2.3.2 전주시 개별공시지가 데이터

2.3.2.1 원본 데이터

[표 4] 2.3.2.1 원본 데이터

이름	설명	출처
전라북도 전주시 개별공시지가 20201031	전주시 법정동별 공시지가 자료가 아닌 지번주소별 공시지가 자료	공공 데이터 포털

2.3.2.2 정제 과정

□ 전주시 법정동별 공시지가 자료가 필요하므로 지번 주소별로 제시되어있는 공시지가를 이용하여 법정동별 공시지가 평균을 구함

□ 법정동별 공시지가 평균을 2.3.1 전주시 법정동별 데이터에 병합

	A	B	C	D	E	F	G	H	I	J	K
1		법정동이름	공동건물수	단독건물수	총인구	유소년인구	고등학생수	중학생수	초등학생수	유아인구수	공시지가
2	0	중앙동1가	1	29	226	20	7	8	15	4	678351.6
3	1	중앙동2가	0	9	116	4	1	1	2	1	1128669
4	2	중앙동3가	0	8	109	6	2	0	5	1	1958812
5	3	중앙동4가	0	58	388	17	10	4	16	1	508617.2
6	4	경원동1가	2	21	223	5	2	2	2	1	1655583
7	5	경원동2가	2	46	188	7	3	3	2	4	847563.6
8	6	경원동3가	2	143	539	13	8	6	6	4	798861.5
9	7	풍남동1가	3	78	239	8	5	3	3	4	1011271
10	8	풍남동2가	0	78	152	7	0	3	3	3	930801.3
11	9	풍남동3가	1	268	485	30	6	11	17	8	1648011

[그림 2-12] 전주시 법정동별 인구수 및 공시지가 정제 데이터 상세

2.3.3 전주시 음식점 데이터

2.3.3.1 원본 데이터

[표 5] 2.3.3.1 원본 데이터

이름	설명	출처
전라북도 전주시 음식점 정보	전주시 일반·휴게음식점을 대상으로 조사한 식당ID, 식당명, 업종, 위도, 경도, 다국어 정보 등을 나타냄	공공 데이터 포털

2.3.3.2 정제 과정

□ 필요 없는 열을 제거한 후 식당ID, 식당명, 업종, 위도, 경도 열만 남김

2.3.4 전주시 음식점 메뉴 데이터

2.3.4.1 원본 데이터

[표 6] 2.3.4.1 원본 데이터

이름	설명	출처
전라북도 전주시 음식점 메뉴 정보	전주시 일반·휴게음식점을 대상으로 조사한 식당ID, 식당명, 메뉴ID, 메뉴명, 메뉴가격, 다국어 정보 등을 나타냄	공공 데이터 포털

2.3.4.2 정제 과정

☐ 불필요한 열을 소거하여 식당ID, 식당명, 메뉴ID, 메뉴명, 메뉴가격 열을 추출

☐ 전주시 음식점 데이터와 식당ID 열을 기준으로 병합

☐ 아동이 이용할 수 없는 업종(술집, 포장마차) 제외

☐ 메뉴명 중에서 주류와 사리 등 불필요한 메뉴 제거

	A	B	C	D	E	F	G	H	I
1		res_ID	res_name	kind	lat(y)	lon(x)	menu_ID	menu_name	price
2	0	464069	원조중앙찹쌀호떡	카페	35.82485	127.1433	502814	뽕잎냉면	5000
3	1	545534	비어드파파 전주효자점	카페	35.82131	127.1041	87544	버블흑당카페라	5300
4	2	545534	비어드파파 전주효자점	카페	35.82131	127.1041	87545	버블밀크티	5000
5	3	545534	비어드파파 전주효자점	카페	35.82131	127.1041	87546	버블흑당밀크티	5300
6	4	545534	비어드파파 전주효자점	카페	35.82131	127.1041	87547	샘플러세트	15000
7	5	545534	비어드파파 전주효자점	카페	35.82131	127.1041	87548	오코세트	14000
8	6	545534	비어드파파 전주효자점	카페	35.82131	127.1041	87549	오도세트	17000
9	7	545534	비어드파파 전주효자점	카페	35.82131	127.1041	87550	우리둘이세트	16500
10	8	545534	비어드파파 전주효자점	카페	35.82131	127.1041	87551	오쿠세트	15000
11	9	545534	비어드파파 전주효자점	카페	35.82131	127.1041	87552	치즈케익세트(5	9000

[그림 2-13] 전주시 음식점 메뉴 정제 데이터 상세

2.3.5 전주시 아이복지카드 가맹점 데이터

2.3.5.1 원본 데이터

[표 7] 2.3.5.1 원본 데이터

이름	설명	출처
전주시 아이급식카드 가맹점 데이터	푸르미 코리아 사이트에서 제공하는 전주시 아이복지카드 가맹점 정보를 웹 크롤링을 통해 추출한 데이터	푸르미카드 사이트 (https://www.purmeecard.com/index5.jsp)

2.3.5.2 정제 과정

☐ Geocoder 프로그램을 통해 가맹점의 위도, 경도 좌표를 추출

☐ 가맹점 이름 정보를 통해 한식, 양식, 중식, 일식, 아시안, 분식, 뷔페, 카페로 업종분류

☐ 각 가맹점의 메뉴 및 가격 정보의 경우 전주시 음식점 메뉴 데이터를 이용하여 병합함. 이때 가맹점 데이터의 식당명과 전주시 음식점 메뉴 데이터의 식당명 간 단어 유사도와 좌표를 사용한 유클리디안 거리를 이용

	A	B	C	D	E	F
1		m_name	address	lon(x)	lat(y)	kind
2	0	파리바게뜨 전주	전주시 덕진구 기린	127.1321	35.83796	카페
3	1	빵아저씨	전북 전주시 완산구	127.1525	35.82286	카페
4	2	브래드#	전라북도 전주시 덕	127.1099	35.84303	카페
5	3	샌앤잼	전주시 덕진구 경동	127.1103	35.84303	카페
6	4	소풍가는날	전주시 완산구 천경	127.1531	35.80857	카페
7	5	파리바게뜨 삼익	전주시 완산구 용리	127.1245	35.79719	카페
8	6	파리바게뜨 전주	전주시 완산구 거미	127.1171	35.7928	카페
9	7	뚜레쥬르 흑석골	전주시 완산구 장승	127.1497	35.80383	카페
10	8	뚜레쥬르 전주서	전주시 완산구 당신	127.1141	35.83199	카페
11	9	밸리하우스서신점	전북 전주시 완산구	127.1164	35.83481	카페

[그림 2-14] 전주시 아이복지카드 가맹점 정제 데이터 상세

3. 분석 프로세스

□ 분석 프로세스의 경우 데이터 수집, 데이터 정제, 탐색적 데이터 분석, 정책 시뮬레이션 총 '네 단계'로 진행함

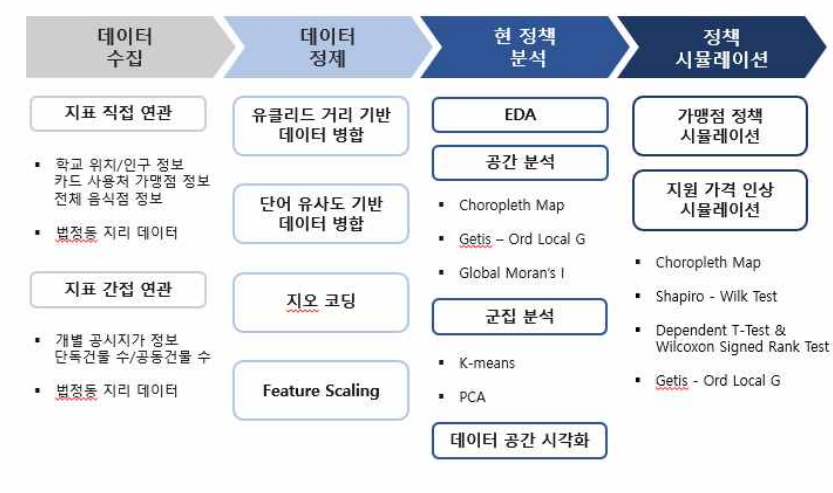
3.1 분석 프로세스

3.1.1 데이터 수집 및 가공

□ 행동패턴을 고려한 아동의 접근성과 관련된 학교 위치, 아동급식카드 가맹점, 전체 음식점 데이터는 웹 크롤링 또는 관련 사이트에서 수집함. 이후 전주시 법정동 지리 데이터와 결합하여 접근성 지수[5]의 형식으로 가공

□ 분석 전, 아동 복지와 간접적인 관련이 있을 것으로 판단한 공시지가, 아동 연령 그룹별 수, 단독건물 수 등 데이터는 관련 사이트에서 수집함 이후 전주시 지리 데이터와 결합 및 scaling 등을 통해 모델 기반 분석에 사용

3.1.2 전체 분석 및 통계기반 시뮬레이션 과정 개괄



[그림3-1] 분석 프로세스

5) 현황조사자료(4) 접근성 지수 공식 참조

3.1.3 결과 도출 및 시각화

□ 취약지역 도출과 군집분석을 통한 실효성 검증 결과를 시각화하고, 정책 시뮬레이션 결과를 바탕으로 전주시 아동급식카드 개선 정책 제안 시 활용

3.2 분석 내용 및 방법

□ 전주시 아동급식카드 현황을 파악하기 위해 분석의 주요 요소인 초·중·고등학교, 가맹 음식점, 전체 음식점 데이터를 이용하여 공간 분포를 시각적으로 확인하고 특성을 파악. 또한, 음식점 메뉴 데이터의 가격 정보를 이용하여 아동이 음식점에서 제한된 금액으로 실질적으로 급식카드를 이용할 수 있는지 확인이 필요

□ 전주시 현 아동급식카드 정책의 취약지역을 분석하기 위해 아동 행동 패턴을 고려한 '접근성 지수'를 정의한 후, 이를 공간 시각화

□ 군집분석을 실시해 취약단계 비교를 통한 정책 실효성 검증

□ 타 지역의 정책을 전주시에 적용할 경우의 효과성을 시뮬레이션한 후, 전주시 정책변경의 타당성을 통계적으로 검증

3.2.1 요인 도출

3.2.1.1 요인 근거

□ 요인들은 논문, 선행 연구 자료, 기사 등을 참고하여 도출

□ 아동의 활동 반경을 고려하여 가맹 음식점을 전체 음식점으로 확대 시 접근성 지수가 높아짐 (초등학생이 편안하게 이동할 수 있는 거리 400m, 중·고등학생 700m[6])

□ 근린주구이론에 기반하여 아동이 주로 활동하는 지역 주변을 중심으로 확대 시 접근성 지수가 높아짐 (초·중·고등학교 주변 지역)

3.2.1.2 요인 인식

□ 음식점 이용도 요인, 접근성 요인으로 인식 후 논문 및 기사에 근거한 구체적인 초기 요인 도출

□ 음식점 이용도 요인 : 초·중·고등학교 위치, 1식 지원금액, 아동급식카드 수혜인원, 법정동별 개별 공시지가

□ 접근성 요인: 음식점 위치, 음식점 개수, 음식점 메뉴, 아동급식카드 가맹점포

6) "아파트지구개발기본계획에 관한 규정"에서 활동반경을 초등학교에서 400m로 규정 (현황조사자료1 참조)

3.2.1.3 가설 설정

□ 가맹점포 수의 증가는 아동급식카드 사용처의 다양성 증대에 영향을 끼침

□ 한 끼 당 지원금액의 증가는 아동급식카드 사용처의 다양성 증대에 영향을 끼침

□ 전체 음식점 확대 정책은 접근성 지수가 증가함

□ 전체 음식점 확대 정책은 한 끼 당 지원금액 인상에 긍정적인 영향을 끼칠 것임

□ 법정동별 개별 공시지가가 낮은 지역은 아동급식카드 수혜자의 주거지역과 상관이 있음

3.2.2 취약지역 분석

□ 접근성 지수 : 아동 행동 패턴[7]을 고려한 '접근성 지수' 정의

[표 8] 3.2.2 취약지역 분석 방법론

이름	설명
Getis - Ord Local G	- 전체 공간상의 관측값 패턴이 아닌 개별 관측값과 근거리 이웃과의 관계를 통해 군집성 여부를 판별하는 방법 - 통계적 검증법을 통해 Hot spot, Cold spot, Non-significant 세 가지 형태로 공간 관측값을 분리하여 Hot spot analysis를 가능하게 함
Choropleth Map	- 통계적 데이터를 다양한 음영 패턴 혹은 기호를 통해 지리적 영역에 나타내기 위해 사용 -변수가 지리적 영역에 따라 어떻게 변하는지 시각화하거나 지역 내 변동성 수준 표시를 가능하게 함

7) 페리의 근린주구 규모는 어린이들이 걸어서 통학할 수 있는 거리인 반경 약 400m(1/4mile)로, 총 거주인 구는 초등학교 1,000명~1,200명을 기준으로 약 5,000명~6,000명을 제시하였다.(안정근 외, 도시개발계획 과 설계, 보성각, 2001, p.81) (현황조사자료 1 첨부)

3.2.3 군집분석을 통한 정책 실효성 검증

[표 9] 3.2.3 군집분석을 통한 정책 실효성 검증

이름	설명
K-means Clustering	- 비지도학습의 일종으로 일반적으로 데이터를 분류하는 용도로 사용됨 - 그룹 내 거리 제곱합을 최소화 하는 방향으로 데이터를 각 군집에 할당하는 방식으로 작동 - 단순하지만 강력한 방법이지만 군집 개수를 사전에 설정하여야 하고 이상치에 민감한 한계가 존재함
PCA (Principal Component Analysis)	- N 차원 데이터 셋의 공간상 방향벡터에 따른 분산을 기반으로 하여 분산을 가능한 보존하는 방식으로 데이터 셋의 차원을 줄이는 기법 - 고차원 데이터 셋을 이해하기 쉬운 저차원 데이터 셋으로 바꾸는 등 장점이 있는 반면 처리된 데이터의 해석이 어렵고 scaling에 민감하다는 단점이 있음
Point-biserial Correlation	- 한 변수가 명명척도에 의해 이분화된 이분변수이고 다른 한 변수가 연속적인 양적변수일 때 두 변수간의 상관을 구하는 방법 - biserial correlation coefficient 에 비해 계산상 편리하고, 기본가정의 제한이나 해석상 제한이 덜하다는 장점을 가짐

3.2.4 타지역 정책을 전주시에 적용할 경우의 효능성 시뮬레이션

-Simulation 1 : 전체 음식점으로 확대
-Simulation 2 : 한 끼 당 지원 금액을 인상

[그림 3-2] Simulation 1, 2

[표 10] 3.2.4 타지역 정책을 전주시에 적용할 경우의 효능성 시뮬레이션

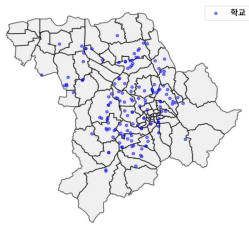
이름	설명
Choropleth Map	- 통계적 데이터를 다양한 음영 패턴 혹은 기호를 통해 지리적 영역에 나타내기 위해 사용 -변수가 지리적 영역에 따라 어떻게 변하는지 시각화하거나 지역 내 변동성 수준 표시를 가능하게 함
Shapiro - Wilk Test	- 빈도주의적 관점에서 데이터의 정규성을 검정하는 통계적 방법
Dependent T - Test	- 데이터가 정규분포를 따를 때 종속적인 두 그룹의 차이를 통계적으로 검증하는 모수적 방법 - 두 그룹의 평균값 차이가 통계적으로 유의한지 검증함
Wilcoxon Signed Rank Test	- 종속적인 두 그룹의 차이를 통계적으로 검증하는 비모수적 방법 - 두 그룹의 중앙값 차이가 통계적으로 유의한지 검증함 - 일반적으로 Dependent T - Test 보다 검증력이 낮은 것으로 알려져 있음
Getis - Ord Local G	- 전체 공간상의 관측값 패턴이 아닌 개별 관측값과 근거리 이웃과의 관계를 통해 군집성 여부를 판별하는 방법 - 통계적 검증법을 통해 Hot spot, Cold spot, Non-significant 세 가지 형태로 공간 관측값을 분리하여 Hot spot analysis를 가능하게 함

4. 분석 결과

4.1 Exploratory Data Analysis

4.1.1 데이터의 공간 분포 시각화

□ 분석 목적 : 분석의 주요 요소인 초·중·고등학교, 가맹 음식점, 전체 음식점의 좌표 위치를 이용하여 공간 분포를 직관적으로 살피고 관련 특징을 도출



[그림4-1] 전주시 내 학교 분포

[그림4-2] 전주시 내 아동급식카드 가맹 분포

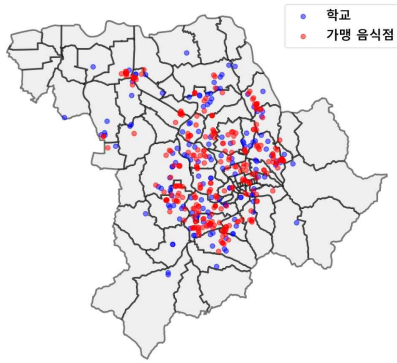
[그림4-3] 전주시 내 전체 음식점 분포

```
# gdf.plot(ax=ax_test)
gdf_emd_jj.plot(ax=ax_test, alpha=0.5,
                edgecolor='k', linewidth=2, color='lightgrey')
gdf_sch_jj.plot(column='num_rest', ax=ax_test, alpha=0.5, color='red',
                label='school', markersize=50)
gdf_rest_jj_frchs.plot(ax=ax_test, alpha=0.7, color='blue',
                       label='franchise rest')
gdf_rest_jj.plot(ax=ax_test, alpha=0.3, color='orange',
                 label='total rest')
ax_test.axis(False)
ax_test.legend()
```

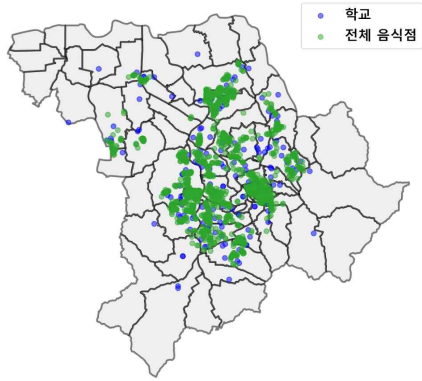
[그림4-4] 핵심코드 1

□ 학교, 가맹 음식점, 전체 음식점 모두 전주시 중심부에 집중해 있고 외곽 지역으로 갈수록 밀도와 개수가 낮아지는 것이 확인됨

□ 특히 중앙지역에서 가맹 음식점과 전체 음식점의 밀도 차이가 뚜렷함



[그림4-5] 전주시 내 학교와 아동급식카드 가맹 음식점 분포



[그림4-6] 전주시 내 학교와 전체 음식점 분포

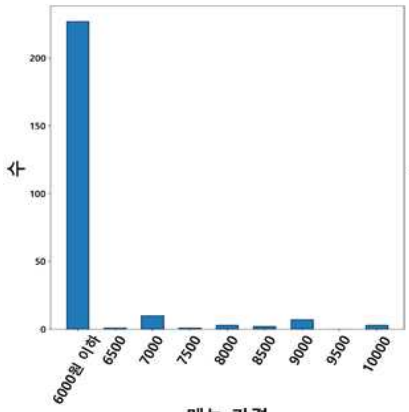
```
gdf_emd_jj.plot(ax=ax_test, alpha=0.5,
                edgecolor='k', linewidth=2, color='lightgrey')
gdf_sch_jj.plot(column='num_rest', ax=ax_test, alpha=0.5, color='red',
                label='school', markersize=50)
gdf_rest_jj_frchs.plot(ax=ax_test, alpha=0.7, color='blue',
                       label='franchise rest')

gdf_emd_jj.plot(ax=ax_test, alpha=0.5,
                edgecolor='k', linewidth=2, color='lightgrey')
gdf_sch_jj.plot(column='num_rest', ax=ax_test, alpha=0.5, color='red',
                label='school', markersize=50)
gdf_rest_jj.plot(ax=ax_test, alpha=0.3, color='orange',
                 label='total rest')
```

[그림4-7] 핵심코드 2

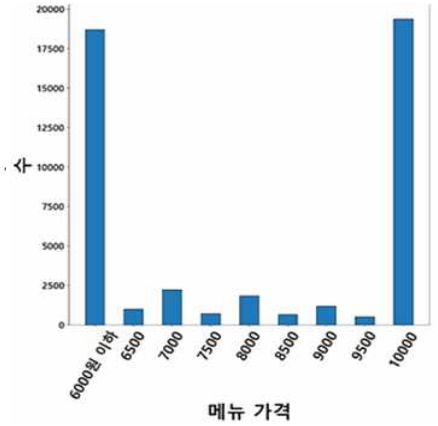
□ 전체 음식점은 가맹 음식점에 비해 중앙 지역에서 학교 근처 밀도가 상대적으로 높음

□ 전체 음식점의 경우도 북서와 남부 일부 지역에서 학교 근처에 가깝게 위치한 경우가 적었음

4.1.2 전주시 음식점 메뉴의 가격 분포



[그림4-8] 전주시 아동급식카드 가맹점 메뉴 가격 분포



[그림4-9] 전주시 내 학교와 전체 음식점 분포

```
#%% Plot - Distribution of menu data
s_menu_jj = df_rest_menu.price
s_menu_jj = s_menu_jj[s_menu_jj <= 10000]
s_menu_jj[s_menu_jj < 6000] = 6000

s_menu_jj_frchs = gdf_rest_jj_frchs.price
s_menu_jj_frchs = s_menu_jj_frchs[s_menu_jj_frchs <= 10000]
s_menu_jj_frchs[s_menu_jj_frchs < 6000] = 6000

cust_tick_label = ['6000원 이하']
for i in np.arange(6500, 10500, 500):
    cust_tick_label.append(str(i))

fig_menu_dist, ax_menu_dist = plt.subplots(1, 2, figsize=(18, 10))

ax_menu_dist[0].hist(s_menu_jj_frchs, edgecolor='k', align='left', bins=9,
                     rwidth=0.6, range=(6000, 10500))
ax_menu_dist[0].set_xticks(np.arange(6000, 10500, 500))
ax_menu_dist[0].set_xticklabels(cust_tick_label, rotation=60,
                                fontsize=1.5*MEDIUM_SIZE)
ax_menu_dist[0].set_xlabel('메뉴 가격', fontsize=1.5*LARGE_SIZE)
ax_menu_dist[0].set_ylabel('수', fontsize=1.5*LARGE_SIZE)

ax_menu_dist[1].hist(s_menu_jj, edgecolor='k', align='left', bins=9,
                     rwidth=0.5, range=(6000, 10500))
ax_menu_dist[1].set_xticks(np.arange(6000, 10500, 500))
ax_menu_dist[1].set_xticklabels(cust_tick_label, rotation=60,
                                fontsize=1.5*MEDIUM_SIZE)
ax_menu_dist[1].set_xlabel('메뉴 가격', fontsize=1.5*LARGE_SIZE)
ax_menu_dist[1].set_ylabel('수', fontsize=1.5*LARGE_SIZE)

fig_menu_dist.tight_layout()
```

[그림 4-10] 핵심코드 3

□ 가맹점[8]의 경우 전체 규모가 전주시 전체 음식점에 비해 현저히 적었으며 6500원 이상 메뉴의 수가 극단적으로 적었음

□ 가맹점의 경우 한 끼 당 지원 가격을 인상해도 아동이 접근 가능한 메뉴 또는 식당 증가율이 크지 않을 것이라 판단됨

□ 전체 음식점의 경우 가격이 10000원인 메뉴가 6000원 이하인 메뉴의 수 보다

8) 10000원을 초과하는 고가 메뉴가 주가되는 음식점과 영양 균형이 좋지 않은 편의점은 제외

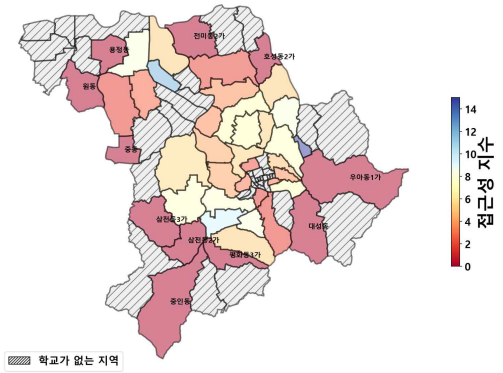
많았으며 6000원 초과 10000원 미만의 메뉴도 적지 않게 확인됨

□ 전체 음식점의 경우 한 끼 당 지원 가격을 인상할 때 아동의 메뉴 및 음식점 접근 다양성이 가맹점의 경우보다 잘 확보될 것으로 예상됨

4.2 취약지역 분석

4.2.1 아동 행동패턴을 고려한 현 정책 하 접근성 지수 시각화

□ 분석 목적 : 아동의 행동 패턴을 고려할 때, 현 정책 하에서 지원 아동의 음식점 접근성 지수를 읍면동 단위로 시각화하여 공간적 분포를 분석하고 취약지역을 파악



[그림4-11] 아동의 행동 패턴을 고려한 접근성 지수 Choropleth map

```
#%% Plot - Mapping number of franchise rests considering child behavior
fig_num_rest_frchs, ax_num_rest_frchs = \
    plt.subplots(1, 1, figsize=(12, 12))

LegendElement = [mpatches.Patch(facecolor='w', hatch='//////',
                                edgecolor='k', label='학교가 없는 지역')]

vmin_frchs_r = gdf_emd_jj_frchs.mean_rest.min()
vmax_frchs_r = gdf_emd_jj_frchs.mean_rest.max();
c_norm_frchs_r = plt.Normalize(vmin=vmin_frchs_r, vmax=vmax_frchs_r)
c_cbar_frchs_r = plt.cm.ScalarMappable(norm=c_norm_frchs_r, cmap=c_cmap)

gdf_emd_jj_frchs.plot(ax=ax_num_rest_frchs, alpha=0.5,
                      edgecolor='k', linewidth=2, column='mean_rest',
                      cmap=c_cmap, legend=False, norm=c_norm_frchs_r)
gdf_emd_jj_frchs[gdf_emd_jj_frchs['mean_rest'].isnull()]\
    .plot(ax=ax_num_rest_frchs, alpha=0.5,
          edgecolor='k', linewidth=2,
          color='lightgrey', legend=False, norm=c_norm_tot_r,
          hatch='//', label='학교')

ax_num_rest_frchs.axis('off')
ax_num_rest_frchs.set_title(
    '활동반경을 고려한 접근가능 가맹음식점 지수 분포')
ax_num_rest_frchs.legend(handles=LegendElement, loc='lower left')

ax_num_rest_frchs_cbar = \
    fig_num_rest_frchs.colorbar(c_cbar_frchs_r, ax=ax_num_rest_frchs,
                                fraction=0.018)
ax_num_rest_frchs_cbar.set_label(label='접근성 지수',
                                 size=25 ,weight='bold')

for idx, r in gdf_emd_jj_frchs.iterrows():
    if (idx in gdf_emd_jj_frchs.mean_rest.nlargest(0).index) or \
        (idx in gdf_emd_jj_frchs.mean_rest.dropna().nsmallest(11).index):
        x, y = r.geometry.centroid.x, r.geometry.centroid.y
        plt.text(x, y, r.ADM_DR_NM, fontsize=10)

save_fig(dir_nm_save_fig, fig_num_rest_frchs,
         '활동반경을 고려한 접근가능 가맹음식점 지수 분포(지역이름 무)',
         close_fig=False)
```

[그림 4-12] 핵심코드 4

□ 4.1.1 결과와 마찬가지로 전주시 중심부에서 벗어날수록 접근성 지수가 하락하는 형태를 보임

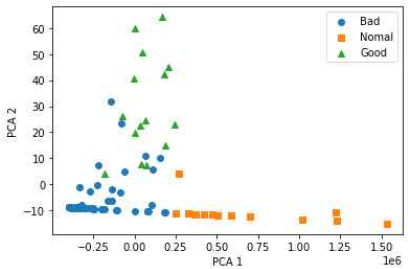
□ 접근성 지수가 가장 높았던 세 지역은 우아동 2가, 여의동 2가, 삼천동 1가로 셋 모두 전주시의 주요 주거지역에 포함됨

□ 학교가 없는 지역들의 경우 관광지, 공장지대 또는 시 외곽도로 근처에 위치한 지역들임

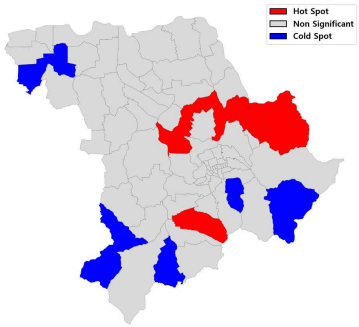
□ 학교가 있으나 근거리에 가맹점이 존재하지 않는 지역은 삼천동 3가, 평화동 3가 등 총 11개가 존재했음. 이는 단순 수치상 취약지역으로 여겨지며 주로 전주시 외곽 지역에 위치함

4.2.2 Hotspot Analysis

□ 분석 목적 : 군집 분석 방법과 공간 통계 모델을 이용하여 취약지역을 파악하고 두 가지 모델을 사용함으로써 분석의 신뢰성 확보



[그림 4-13] K-means 군집 분석을 이용한 현 정책하 취약지역 분류 결과의 PCA를 이용한 시각화



[그림 4-14] Getis-Ord Local G를 이용한 현 정책 하 취약지역 분석 결과

```
gdf_base1, gdf_base2 = load_map_rest_join(pri_thld=6000)
baseline = GdfCompare(gdf_base1, gdf_base2)
baseline.run_local_g()
baseline.run_stats_diff()
baseline.run_global_moran()
baseline_fig = baseline.plot_compare()
```

```
pca = PCA(n_components=2) #2차원 평면에 시각화를 위해 차원축소
pca_transformed = pca.fit_transform(six_bf_merge[['sum_rest','sum_school','mean_rest','공시지가']])

six_bf_merge['pca_x'] = pca_transformed[:,0]
six_bf_merge['pca_y'] = pca_transformed[:,1]

marker0_ind = six_bf_merge[six_bf_merge['cluster'] == 0].index
marker1_ind = six_bf_merge[six_bf_merge['cluster'] == 1].index
marker2_ind = six_bf_merge[six_bf_merge['cluster'] == 2].index

plt.scatter(x = six_bf_merge.loc[marker0_ind, 'pca_x'], y = six_bf_merge.loc[marker0_ind, 'pca_y'], marker = 'o', label= 'Bad')
plt.scatter(x = six_bf_merge.loc[marker1_ind, 'pca_x'], y = six_bf_merge.loc[marker1_ind, 'pca_y'], marker = 's', label= 'Nomal')
plt.scatter(x = six_bf_merge.loc[marker2_ind, 'pca_x'], y = six_bf_merge.loc[marker2_ind, 'pca_y'], marker = '^', label= 'Good')

plt.legend()
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.show()
```

[그림 4-15] 핵심코드 5-1, 5-2

□ K-means 군집 분석의 경우 취약지역의 비율이 다른 두 지역의 비율에 비해 높았음. 이는 현 정책하 가맹점 음식점 수의 절대적 부족에 따른 결과로 판단됨

□ Getis-Ord Local G를 이용한 Hotspot 분석의 경우 총 6곳을 취약지역의 핵심(Cold spot)으로 판단했음. 해당 6개 지역은 K-means 방법에서도 취약지역으로 분류되었기 때문에 두 모델 어느정도 같은 지역을 가리킨다고 볼 수 있음

□ Getis-Ord Local G 결과에 따르면 전주시 중심부 우측이 접근성 지수가 높은 지역이 밀집해있는 곳으로 선정됨. 이는 4.2.1의 결과에 부합하는 형태임

4.2.3 취약지역 관련 변수 상관분석

□ 분석 목적 : 취약지역과 연관성이 있다고 판단되는 변수들과 취약지역의 관계를 통계적 방법을 통해 확인함으로써 취약지역의 환경에 대해 분석

[표 11] Point-Biserial Correlation을 이용한 통계

변수명	Correlation Value	P-value < 0.05
공시지가	0.4636333016	O
총 학교 수	0.7278438546	O
총 식당 수	0.6778813512	O
평균 식당 수	0.3447202561	O

```
print('공시지가 cor :',stats.pointbiserialr(x = six_bf_merge['공시지가'], y = six_bf_merge['cluster']))
print('식당 총 수 cor :',stats.pointbiserialr(x = six_bf_merge['sum_rest'], y = six_bf_merge['cluster']))
print('식당수 평균 cor :',stats.pointbiserialr(x = six_bf_merge['mean_rest'], y = six_bf_merge['cluster']))
print('학교 총 수 cor :',stats.pointbiserialr(x = six_bf_merge['sum_school'], y = six_bf_merge['cluster']))
```

[그림 4-16] 핵심코드 6

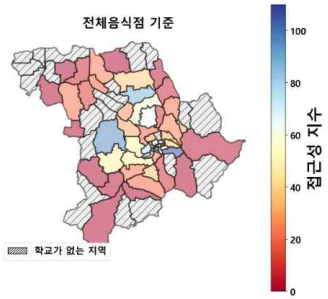
□ 실험한 네 변수 모두 통계적으로 유의미한 차이($p < 0.05$)을 확보했으며 보통 정도의 상관관계를 보였음. 특히, 공시지가와 취약지역간 상관관계가 유의한 것은 아동 복지 측면에서 저소득층 아동이 해당 복지 정책에서의 높은 취약성을 면밀히 검토해야함을 시사함

4.3 타지역 정책을 기반으로 한 전주시 정책변경 효능 시뮬레이션

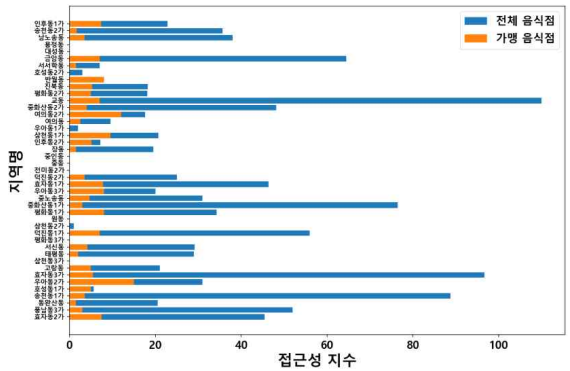
4.3.1 전체 음식점 기반 지원 방식 적용 시뮬레이션

□ 분석 목적 : 타 시도에서 이미 시행되고 있는 전체 음식점 기반 정책을 전주시에서 실시한다 가정할 때, 정책의 공간적 효과를 다양한 관점에서 비교 분석함

[그림4-17] 정책별 지원아동 접근성 지수
가맹음식점 기준



[그림4-18] 정책별 지원아동 접근성 지수
전체음식점 기준



[그림4-19] 행동패턴을 고려한 법정동별 접근성 지수 bar chart

```
### Plot - Mapping number of restaurants (total & franchise mix)
fig_num_rest_mix, ax_num_rest_mix = \
    plt.subplots(1, 2, figsize=(10, 10))

legendElement = [patches.Patch(facecolor='w', hatch='//////',
                               edgecolor='k', label='학교가 없는 지역')]

gdf_cmd_ss_frchs.plot(ax=ax_num_rest_mix[0], alpha=0.5,
                      edgecolor='k', linewidth=2, column='mean_rest',
                      cmap=c_cmap, legend=False, norm=c_norm_tot_r)
gdf_cmd_ss_frchs[gdf_cmd_ss_frchs['mean_rest'].isnull()]\
.plot(ax=ax_num_rest_mix[0], alpha=0.5,
                      edgecolor='k', linewidth=2,
                      color='lightgrey', legend=False, norm=c_norm_tot_r,
                      hatch='//', label='학교가 없는 지역')
ax_num_rest_mix[0].axis('off')
ax_num_rest_mix[0].set_title('가맹음식점 기준')
ax_num_rest_mix[0].legend(handles=legendElement, loc='lower left')

gdf_cmd_ss.plot(ax=ax_num_rest_mix[1], alpha=0.5,
                edgecolor='k', linewidth=2, column='mean_rest',
                cmap=c_cmap, legend=False, norm=c_norm_tot_r)
ax_num_rest_mix[1].axis('off')
ax_num_rest_mix[1].set_title('전체음식점 기준')
gdf_cmd_ss[gdf_cmd_ss['mean_rest'].isnull()]\
.plot(ax=ax_num_rest_mix[1], alpha=0.5,
      edgecolor='k', linewidth=2,
      color='lightgrey', legend=False, norm=c_norm_tot_r,
      hatch='//')
ax_num_rest_mix[1].legend(handles=legendElement, loc='lower left')

ax_num_rest_mix_cbar = \
    fig_num_rest_mix.colorbar(c_cbar_tot_r, ax=ax_num_rest_mix[1],
                              fraction=0.04)
ax_num_rest_mix_cbar.set_label(label='접근성 지수',
                               size=15, weight='bold')

save_fig(dir_re_save_fig, fig_num_rest_mix,
         '음식점별 그룹화 인접동에 가맹음식점 지수 분포(가맹, 전체)',
         close_fig=False)

### Plot - Bar chart of the number of restaurants (total & franchise mix)
fig_num_rest_bar, ax_num_rest_bar = plt.subplots(1, 1, figsize=(12, 8))

ax_num_rest_bar.barh(y=np.arange(len(gdf_cmd_ss_mean_rest.dropna())),
                     width=gdf_cmd_ss_mean_rest.dropna(),
                     label='전체 음식점')

ax_num_rest_bar.barh(y=np.arange(len(gdf_cmd_ss_frchs.mean_rest.dropna())),
                     width=gdf_cmd_ss_frchs.mean_rest.dropna(),
                     label='가맹 음식점')

ax_num_rest_bar.set_ylabel('지역명', fontsize=LARGE_SIZE)
ax_num_rest_bar.set_xlabel('접근성 지수', fontsize=LARGE_SIZE)
ax_num_rest_bar.set_yticks(np.arange(len(gdf_cmd_ss_mean_rest.dropna())))
ax_num_rest_bar.set_yticklabels(
    gdf_cmd_ss_frchs.dropna(subset=['mean_rest']).ADM_DR_NM,
    fontsize=SMALL_SIZE*0.9)
ax_num_rest_bar.legend()

save_fig(dir_re_save_fig, fig_num_rest_bar,
         '음식점별로 그룹화 인접 법정동 지수 분포도 막대 bar 차')
```

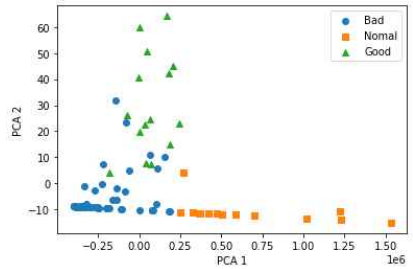
[그림 4-20] 핵심코드 7

□ 전체 음식점 정책을 실시할 경우 각 법정동의 아동급식카드 사용 접근성 지수가 평균 약 13.3배 증가할 것으로 예상됨

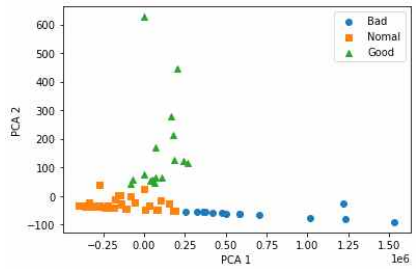
□ 가맹 음식점 정책 하에서 접근성이 낮은 지역(접근성 지수가 중위수 이하)은 전체 음식점 정책 시행 시 접근성 지수가 약 13배 증가할 것으로 예상됨

□ 가맹 음식점 정책 하에서 접근성이 높은 지역(접근성 지수가 중위수 초과)은 전체 음식점 정책 시행 시 접근성 지수가 약 0.65배 증가할 것으로 예상됨

□ 상대적 취약지역들이 높은 지수 상승 폭을 보임. 이는 현 정책하에서 소외된 외곽 지역과 중심지 간 불균형 해소에 도움이 될 것을 시사함



[그림 4-21] K-means 군집 분석을 이용한 취약지역 분류 결과의 PCA를 이용한 시각화 (현 정책 기반)

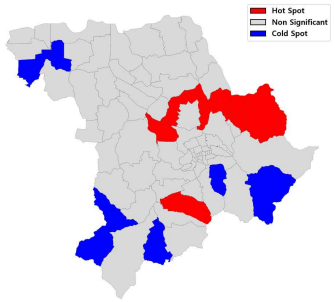


[그림 4-22] K-means 군집 분석을 이용한 취약지역 분류 결과의 PCA를 이용한 시각화 (전체 음식점 확대 정책 기반)

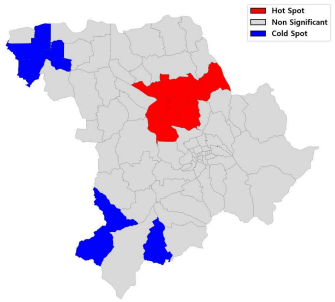
```
### Add Official Price

sc = StandardScaler() #정확한 거리를 산정하기 위한 스케일링

sc.fit(six_bf_merge)
sc.fit(six_af_merge)

kmeans = KMeans(n_clusters=3, init = 'k-means++', max_iter= 300, random_state = 40).fit(sc.transform(six_bf_merge))
six_bf_merge['cluster'] = kmeans.labels_
kmeans_2 = KMeans(n_clusters=3, init = 'k-means++', max_iter= 300, random_state = 40).fit(sc.transform(six_af_merge))
six_af_merge['cluster'] = kmeans_2.labels_

six_bf_merge['target'] = six_bf.hotspot_class
six_merge_result = six_bf_merge.groupby(['target', 'cluster'])['sum_school'].count()
six_af_merge['target'] = six_af.hotspot_class
six_merge_result2 = six_af_merge.groupby(['target', 'cluster'])['sum_school'].count()


six_bf_score = silhouette_score(six_bf_merge, kmeans.labels_,metric='euclidean')
six_af_score = silhouette_score(six_af_merge, kmeans_2.labels_,metric='euclidean')

print(six_merge_result)
print(six_merge_result2)
print('6000 before silhouette score: %.3f' %six_bf_score)
print('6000 after silhouette score: %.3f' %six_af_score)

### Before 6000 Policy Visualization

pca = PCA(n_components=2) #2차원 평면에 시각화를 위해 차원축소
pca_transformed = pca.fit_transform(six_bf_merge[['sum_rest','sum_school','mean_rest','공시지가']])

six_bf_merge['pca_x'] = pca_transformed[:,0]
six_bf_merge['pca_y'] = pca_transformed[:,1]

marker0_ind = six_bf_merge[six_bf_merge['cluster'] == 0].index
marker1_ind = six_bf_merge[six_bf_merge['cluster'] == 1].index
marker2_ind = six_bf_merge[six_bf_merge['cluster'] == 2].index

plt.scatter(x = six_bf_merge.loc[marker0_ind, 'pca_x'], y = six_bf_merge.loc[marker0_ind, 'pca_y'], marker = 'o', label= 'Bad')
plt.scatter(x = six_bf_merge.loc[marker1_ind, 'pca_x'], y = six_bf_merge.loc[marker1_ind, 'pca_y'], marker = 's', label= 'Nomal')
plt.scatter(x = six_bf_merge.loc[marker2_ind, 'pca_x'], y = six_bf_merge.loc[marker2_ind, 'pca_y'], marker = '^', label= 'Good')

plt.legend()
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.show()
```

[그림 4-23] 핵심코드 8

□ K-means 군집 분석 결과에 따르면 취약지역은 74% 감소했으며, 우수지역의 경우 약 7% 증가했음. 이는 전체적으로 접근성 지수가 상향평준화 되었음을 가리키는 결과로 볼 수 있으며 취약지역의 높은 감소율이 고무적임



[그림 4-24] Getis-Ord Local G를 이용한 정책 전 후 Hotspot Analysis(현 정책 기반)



[그림 4-25] Getis-Ord Local G를 이용한 정책 전 후 Hotspot Analysis(전체 음식점 확대 정책 기반)

```
def plot_compare(self):
    import matplotlib.patches as mpatches

    fig, ax = plt.subplots(1, 2, figsize=(18, 9))

    LegendElement = [mpatches.Patch(facecolor='r',
                                    edgecolor='k', label='Hot Spot'),
                     mpatches.Patch(facecolor='lightgrey',
                                    edgecolor='k', label='Non Significant'),
                     mpatches.Patch(facecolor='blue',
                                    edgecolor='k', label='Cold Spot')]

    sig_gdf_before = self.gdf_before.lg_p_sim < 0.05
    hh_jj_frchs = self.gdf_before[(sig_gdf_before==True) &
                                  (self.gdf_before.lg_Zs > 0)].plot(
        ax=ax[0], color='red', edgecolor='k', linewidth=0.1,
        legend=True, categorical=True)
    ns_jj_frchs = self.gdf_before[sig_gdf_before==False].plot(
        ax=ax[0], color='lightgrey', edgecolor='k', linewidth=0.1,
        label='Non Significant')
    ll_jj_frchs = self.gdf_before[(sig_gdf_before==True) &
                                  (self.gdf_before.lg_Zs < 0)].plot(
       ax=ax[0], color='blue', edgecolor='k', linewidth=0.1,
       label='Cold Spot')
    ax[0].axis(False)
    ax[0].legend(handles=LegendElement, loc='upper right')

    sig_gdf_after = self.gdf_after.lg_p_sim < 0.05
    hh_jj = self.gdf_after[(sig_jj==True) &
                           (self.gdf_after.lg_Zs > 0)].plot(
        ax=ax[1], color='red', edgecolor='k', linewidth=0.1,
        label='Hot Spot')
    ns_jj = self.gdf_after[sig_jj==False].plot(
        ax=ax[1], color='lightgrey', edgecolor='k', linewidth=0.1,
        label='Non Significant')
    ll_jj = self.gdf_after[(sig_jj==True) &
                           (self.gdf_after.lg_Zs < 0)].plot(
        ax=ax[1], color='blue', edgecolor='k', linewidth=0.1,
        label='Cold Spot')
    ax[1].axis(False)
    ax[1].legend()
    ax[1].legend(handles=LegendElement, loc='upper right')

    return fig
```

[그림 4-26] 핵심코드 9

□ Getis-Ord Local G 결과의 경우 전주시 우측 하단의 색장동과 동서학동이 Cold spot에서 제외됨. 이는 정책 시뮬레이션 결과인 4.3.1에서 전주시 중앙 아래 측 지역의 접근성 지수가 높게 표시된 것과 흐름을 같이하는 결과이며, 접근성이 높은 인근지역의 혜택을 간접적으로 받음으로서 취약성이 높은 지역들의 밀집성을 감소시킬 수 있는 방향의 가능성을 보여줌

[표 12] 현 정책 기반 Point-Biserial Correlation

현 정책 기반	Correlation Value	P-value < 0.05
공시지가	0.4636333016	O
총 학교 수	0.7278438546	O
총 식당 수	0.6778813512	O
평균 식당 수	0.3447202561	O

```
print('공시지가 cor :',stats.pointbiserialr(x = six_bf_merge['공시지가'], y = six_bf_merge['cluster']))
print('식당 총 수 cor :',stats.pointbiserialr(x = six_bf_merge['sum_rest'], y = six_bf_merge['cluster']))
print('식당수 평균 cor :',stats.pointbiserialr(x = six_bf_merge['mean_rest'], y = six_bf_merge['cluster']))
print('학교 총 수 cor :',stats.pointbiserialr(x = six_bf_merge['sum_school'], y = six_bf_merge['cluster']))
```

[그림 4-27] 핵심코드 10

[표 13] 전체 음식점 정책 시뮬레이션 결과 기반 Point-Biserial Correlation

전체 음식점 정책 시뮬레이션 결과 기반	Correlation Value	P-value < 0.05
공시지가	-0.4127460429	O
총 학교 수	0.6775748527	O
총 식당 수	0.5953756255	O
평균 식당 수	0.6111836645	O

```
print('공시지가 cor :',stats.pointbiserialr(x = six_af_merge['공시지가'], y = six_af_merge['cluster']))
print('식당 총 수 cor :',stats.pointbiserialr(x = six_af_merge['sum_rest'], y = six_af_merge['cluster']))
print('식당수 평균 cor :',stats.pointbiserialr(x = six_af_merge['mean_rest'], y = six_af_merge['cluster']))
print('학교 총 수 cor :',stats.pointbiserialr(x = six_af_merge['sum_school'], y = six_af_merge['cluster']))
```

[그림 4-28] 핵심코드 11

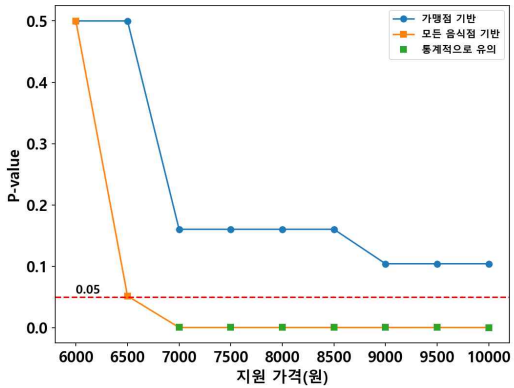
□ 전체 음식점 정책 시뮬레이션 결과의 경우 Point-Biserial Correlation이 이전의 현행 정책하 결과와 마찬가지로 모두 유의하였음

□ 공시지가의 경우 통계적으로 유의하나 현 정책하 결과와는 반대로 취약지역과 음의 상관관계를 보였음. 이는 추가적인 분석이 필요하나, 공시지가가 낮은 지역과 정책 우수지역 간 밀접한 관련성을 가리키는 만큼 새로운 정책이 자본적으로 취약

한 지역의 아동에 대한 접근성 지수가 대폭 증가할 가능성을 시사함

4.3.2 한 끼 당 지원 가격 인상시 정책 효율성 시뮬레이션

□ 분석 목적 : 한 끼 당 지원 가격을 인상할 경우를 상정하고 인상 금액에 따른 지원 아동의 편의성 증가 정도가 기존의 한 끼당 6000원 지원 방식에 비해 통계적으로 유의한지 검증하고, 이를 통해 지원 아동의 실질적 접근성 증가 효과를 얻기 위한 최소한의 지원 금액 인상가를 제시



[그림4-29] 한 끼당 지원 가격, 정책 별 접근성의 기준값과의 차이 통계량 P-value

```
### Get results from price increasing test prior to the new policy
df_pri_inc_pr = pd.DataFrame(index=np.arange(6000, 10500, 500))
df_pri_inc_pr['stats_diff'] = \
    ['t' if x.t_test else 'w' for x in pri_inc_wo_plcy]
df_pri_inc_pr['statistic'] = \
    [x.t_test_s if x.t_test else x.wrst_s for x in pri_inc_wo_plcy]
df_pri_inc_pr['pvals'] = \
    [x.t_test_pval if x.t_test else x.wrst_pval for x in pri_inc_wo_plcy]
df_pri_inc_pr['glo_m'] = \
    [x.glo_m_after[1] for x in pri_inc_wo_plcy]
df_pri_inc_pr['sum_mean_rests_wo_p'] = \
    [x.gdf_after.mean_rest.sum() for x in pri_inc_wo_plcy]
df_pri_inc_pr['avg_mean_rests_wo_p'] = \
    [x.gdf_after.mean_rest.dropna().mean() for x in pri_inc_wo_plcy]
df_pri_inc_pr['med_mean_rests_wo_p'] = \
    [x.gdf_after.mean_rest.dropna().median() for x in pri_inc_wo_plcy]

### price increasing test under condition posterior to the new policy
pri_inc_w_plcy = []
for price in tqdm(np.arange(6000, 10500, 500),
                  desc='Price increase test posterior to the new policy'):
    pri_inc_w_plcy.append(
        get_all_attr(6000, price, mode='after')
        )

### Get results from price increasing test posterior to the new policy
df_pri_inc_po = pd.DataFrame(index=np.arange(6000, 10500, 500))
df_pri_inc_po['stats_diff'] = \
    ['t' if x.t_test else 'w' for x in pri_inc_w_plcy]
df_pri_inc_po['statistic'] = \
    [x.t_test_s if x.t_test else x.wrst_s for x in pri_inc_w_plcy]
df_pri_inc_po['pvals'] = \
    [x.t_test_pval if x.t_test else x.wrst_pval for x in pri_inc_w_plcy]
df_pri_inc_po['glo_m'] = \
    [x.glo_m_after[1] for x in pri_inc_w_plcy]
df_pri_inc_po['sum_mean_rests_w_p'] = \
    [x.gdf_after.mean_rest.sum() for x in pri_inc_w_plcy]
df_pri_inc_po['avg_mean_rests_w_p'] = \
    [x.gdf_after.mean_rest.dropna().mean() for x in pri_inc_w_plcy]
df_pri_inc_po['med_mean_rests_w_p'] = \
    [x.gdf_after.mean_rest.dropna().median() for x in pri_inc_w_plcy]

### Plot - P values based on price increasing
fig_pri_inc_pvals, ax_pri_inc_pvals = plt.subplots(1, 1, figsize=(8, 6))

ax_pri_inc_pvals.plot(df_pri_inc_pr.index, df_pri_inc_pr.pvals, 'o-',
                      label='가맹점 기반', markersize=8)
ax_pri_inc_pvals.plot(df_pri_inc_po.index, df_pri_inc_po.pvals, 's-',
                      label='모든 음식점 기반', markersize=8)
ax_pri_inc_pvals.plot(df_pri_inc_po.index[21], df_pri_inc_po.pvals[21:],
                      'g', label='통계적으로 유의', markersize=8)
ax_pri_inc_pvals.axhline(0.05, linestyle='--', color='r', linewidth=1.5)
ax_pri_inc_pvals.text(6000, 0.055, '0.05', fontsize=12)
ax_pri_inc_pvals.legend()

ax_pri_inc_pvals.set_xlabel('지원 가격(원)')
ax_pri_inc_pvals.set_ylabel('P-value')

save_fig(dir_rm_save_fig, fig_pri_inc_pvals, title='정책별 pvalues',
         close_fig=False)
```

[그림 4-30] 핵심코드 12

□ 모든 음식점 기반 정책의 경우 한 끼 당 지원가격 7000원 부터 기존의 6000원보다 통계적으로 유의미한 차이($p < 0.05$)를 보이는 접근성 증가가 확인됨

□ 가맹점 기반 정책의 경우 금액 인상에 따라 p-value가 감소세를 보이긴 하나, 최대 10000원 까지 증가시켰음에도 불구하고 통계적으로 유의미한 차이($p < 0.05$)를 보이지 않음

□ 지원 가격 인상에도 불구하고 가맹점 기반 정책의 접근성 지수가 유의미하게 증가하지 않는 결과는 4.1.2에서 6000원 초과 10000원 미만의 음식점이 상대적으로 적었던 것과 상응하는 결과로 판단됨

□ 지원 가격 인상 정책이 효과를 보이려면 가맹점 기반의 정책에서 전체 음식점 정책으로의 변경이 우선시 되어야 함을 시뮬레이션을 통해 확인할 수 있었음

5. 활용 방안

5.1 문제점 개선 방안

5.1.1 전주시 내 전체 음식점으로 아동급식카드 사용처 확대

□ 전주시 내 전체 음식점으로 아동급식카드 사용처 확대를 통해 아동들의 접근성을 증진하여 수요자 중심의 복지정책을 실현

□ 전주시 내 전체 음식점으로 아동급식카드 사용처를 확대하여 중심부에 밀집한 급식카드 사용처를 기존 사각지대인 외곽까지 확장

□ 전체적으로 상향되는 경향을 보였지만 다소 소외되는 구역이 발견되어 해당 구역에 대한 정책 수립 시 결정 지표로 사용

5.1.2 적정 급식단가로 인상 추진

□ 전체 음식점 정책을 적용할 경우 지원금액별 효율성을 기반으로 인상안 상정 시 예산 내 최적 인상분을 제안

□ 현 정책과 전체 음식점 정책 각각의 자금 운용 효율성 분석 결과를 의사결정의 보조 수단으로 사용

5.2 업무 활용 방안

5.2.1 전북은행과 연계를 통한 방안 마련

□ 서울시, 경기도, 부산시 등 이미 적용되어 실행되고 있는 우수 사례와 분석 결과를 기반으로 타당성을 검토하고 지역은행과 연계 방안을 마련

□ 은행에서 발행한 급식카드는 전용 단말기 사용의 한계를 극복하고 사용처를 유

연하게 확대할 수 있음

□ 일반 카드와 동일한 디자인 사용으로 결식 우려 아동의 경제적 여건, 신원 노출의 불편함과 낙인감을 해소

□ 전북은행의 재난지원금, 지역화폐 등을 발행해본 경험은 복지정책 실현에 이로운 환경을 제공할 것으로 기대, 지역은행에 사회적 책임감을 제공할 수 있는 환경을 제공

5.2.2 취약지역을 우선적으로 보완하는 방안 제시

□ 분석 데이터를 바탕으로 취약지역으로 선정된 전주시 북서부 지역(성덕동, 남정동 등)과 남부지역(색장동, 용복동 일대 등)에서 수혜 아동들의 급식카드 사용실태를 긴밀히 점검해야 함을 주장

□ 취약지역에는 절대적인 음식점의 수가 부족하므로 아동복지센터, 노인복지센터, 장애인센터, 종교시설 등 취사가 가능한 시설에 협조를 요청하여 단가인상, 가맹점 확대 노력보다는 단기간 내에 실효성이 있는 방안 마련

5.2.1 시범사업지역 선정에 활용

□ 일괄적으로 전체 음식점 확대가 어려울 시, 도출한 취약지역을 중심으로 시범사업지역을 선정하는데 활용

6. 참고 자료

6.1 연구 자료

- 이지은「보행친화 관점에서 본 근린주구 계획모델의 특성 연구」
- 백혜선「국내 주거지 생활권 계획개념 및 사례분석」
- 구미경「도시 저층주거지의 생활권 중심 분석」
- 오상우「취약계층 영양불균형 및 비만 예방관리 방안」
- 안호준「결식우려아동 급식 개선을 위한 실태조사 및 개선방안을 위한 연구」
- 정영태「대구지역 결식우려아동에 대한 급식지원 개선방안 연구」
- 류정희「아동종합실태조사」
- 박금식「결식우려아동에 대한 효율적인 급식지원 방안 연구」
- 전주대학교 산학협력단「2020 전주시 사회조사 보고서」
- 김기헌「2020년 청소년종합실태조사」
- 김희진「학교 밖 청소년 지원센터 운영모형 개발」

- 한국보건사회연구원「빈곤층 아동급식지원제도 실태조사」
- 안정근「도시개발계획과 설계」

6.2 문헌 자료

- Tobigs「클러스터링 실습 (1) (EDA,Sklearn)」
- 세바스찬 리시카「머신러닝교과서 with 파이썬,사이킷런,텐서플로」
- 권철민「파이썬머신러닝완벽가이드」
- 전북교육청「전북교육청, 2021년도 제1회 추경예산안 도의회 제출」보도자료
- 최범규「충북 결식아동 급식지원 시스템 개선해야」노컷뉴스
- 변진경「먹어도 먹는 게 아닌 '아동 흙밥 보고서'」시사in
- 최지영「편의점 지겹지만...기능 못하는 아동급식카드」KBS
- 장수인「코로나19 확산에 문 닫은 초등학교, 비대면 수업 증가 '급식지원 사각지대 우려」전북도민일보
- 김호경「과일은 학교에서만 먹어요...급식없는 방학이 두려운 아이들」동아일보
- 유은영「최선 의원 "결식우려 아동 급식지원 확대해야"」한국농업신문

6.3 관련 법률

- 아동복지법 제 35조(건강한 심신의 보존) 2항, 3항
- 전주시 공공급식 지원 조례 제 3조
- 전주시 취약계층에 대한 급식지원 조례 제 1조, 제 4조, 제 5조

7. 부록

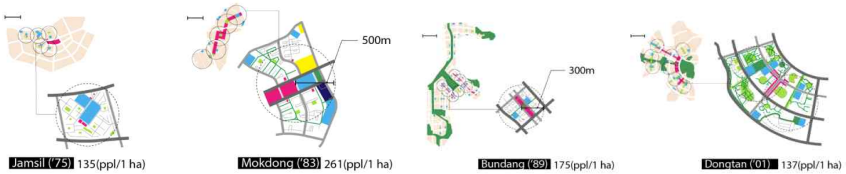
7.1 현황 조사 자료

7.1.1 현황조사자료(1)

□ 단순히 가맹점, 시설이 존재하는 것 뿐만 아니라 실질적 수요자인 아이들이 편하게 접근할 수 있는 곳에 위치해 있는지를 우선으로 정책을 검증해보는 시도가 필요하다.

□“아파트지구개발기본계획에 관한 규정”에서 근린주구반경을 초등학교를 중심으로 400m로 규정하였고, 현행 ‘도시계획시설의 결정, 구조 및 설치기준에 관한 규칙’에서도 초등학교를 근린주구단위로 규정하고 있어, 초등학교를 중심으로 반경 400m를, 중고등학교를 중심으로 700m를 학생들의 이동반경으로 산정하였다.

□ 페리의 근린주구 규모는 어린이들이 걸어서 통학할 수 있는 거리인 반경 약 400m(1/4mile)로, 총 거주인 구는 초등학교 1,000명~1,200명을 기준으로 약 5,000명~6,000명을 제시하였다(안정근 외, 도시개발계획 과 설계, 보성각, 2001, p.81)



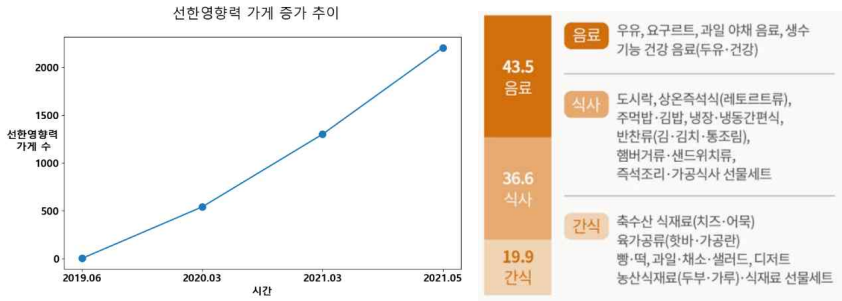
7.1.2 현황조사자료(2)

□ 2019년 진짜파스타(상호명) 부터 시작되어 올해 초 철인7호(상호명) 까지 많은 이들의 가슴을 따스하게 만들었던 선한영향력 가게들의 선행으로 아동급식에 대한 대중의 인식수준은 이전보다 더 높아졌다.

□ 이처럼 지난 2년간 선한영향력 참여 가게의 증가 추이를 보면 최근 3개월간 약 2배의 높은 상승폭을 보인다.(부산일보. 19.07.23, 티브이데일리. 21.02.27)

□ 아동급식카드 사용의 80%를 차지하는 편의점 사용실태를 살펴보면 40%이상이 음료로 구성되었고 36%가 식사, 20%가 간식으로 이루어져 있어 사실상 한끼 식사로는 턱없이 부족한 구성이다.(꿈나무카드 분석보고서, 경희대학교 산학협력단 SK 청년비상 빅리더팀)

□ 이에 대중의 인식수준 개선과 함께 발 맞추어 아동이 실제 섭취하는 영양소에 관심을 가질 필요가 있다.

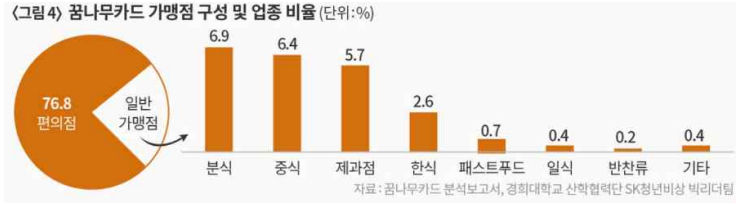


7.1.3 현황조사자료(3)

□ 소득이 낮은 가구에 속해있는 아동의 경우, 또래 아동 보다 부족한 영양소를 섭취하고 있다는 사실이 조사결과를 통해 밝혀졌다. (2018년 아동종합실태조사, 한국보건사회연구원)

□ 또한 실제 섭취량 통계에서도 1~2분위(하위40%)아동이 눈에 띄게 섭취량이 낮게 나타났다. (2017년 취약계층 영양불균형 및 비만 예방관리 방안, 보건복지부,동국대 산학협력단)

□ 이러한 현상은 지원금액 자체의 문제로 볼 수도 있으나 가맹점 중 80%에 달하는 비중을 편의점이 차지하고 있어 다양한 사용처가 구비되지 않았다는 것이 하나의 주 원인으로 볼 수 있다.



7.1.4 현황조사자료(4)

$$a_i = \frac{\sum_j \sum_k g(i,j) f(j,k)}{\sum_j g(i,j)}$$

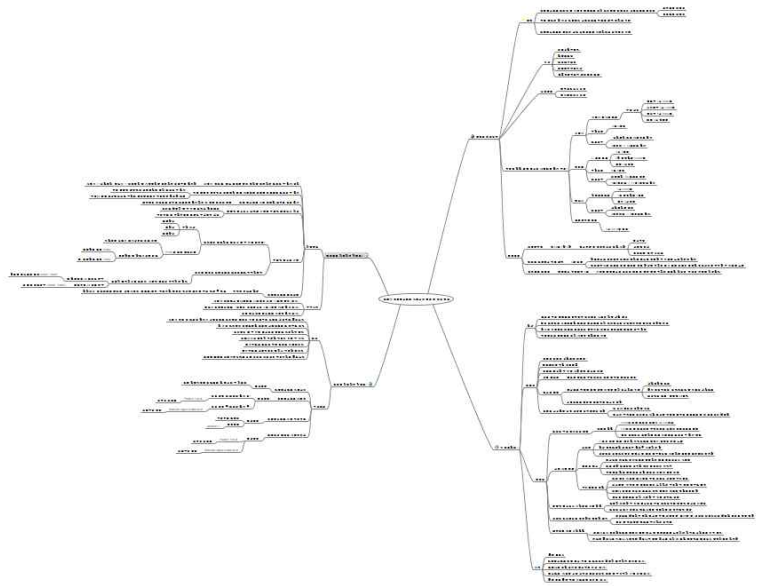
$$where\ f(j,k) = \begin{cases} \mathbf{1}\ \ if\ distance\left(school_j, restaurant_k\right) < S_j \\ \mathbf{0}\ \ else \end{cases}$$

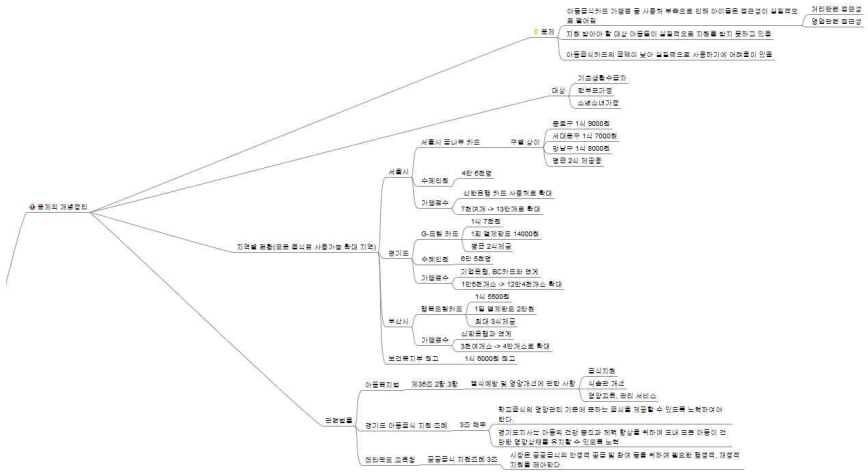
$$\&\ g(i,j) = \begin{cases} \mathbf{1}\ if\ school_j \in area_i \\ \mathbf{0}\ \ else \end{cases}$$

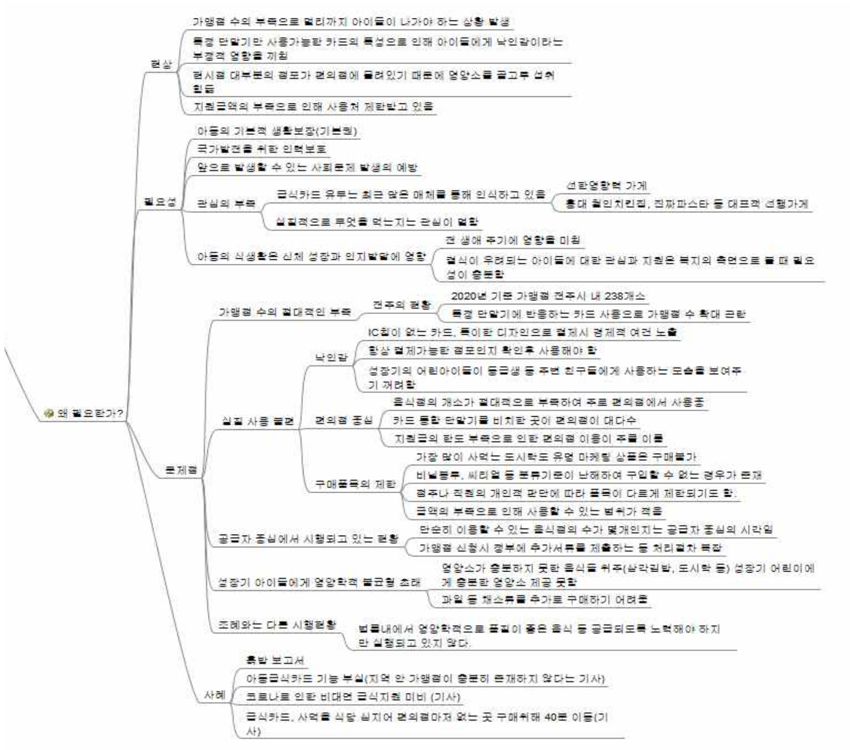
$$S_j = \begin{cases} \mathbf{400}\ \ if\ school_j\ is\ an\ elementary\ school \\ \mathbf{700}\ \ else \end{cases}$$

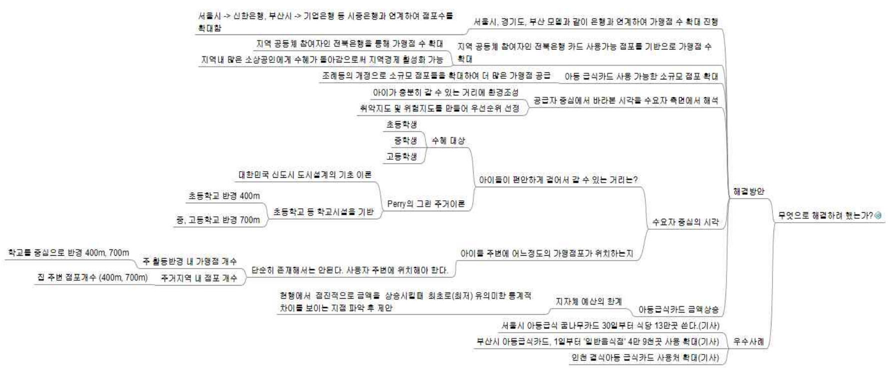
접근성 지수 공식

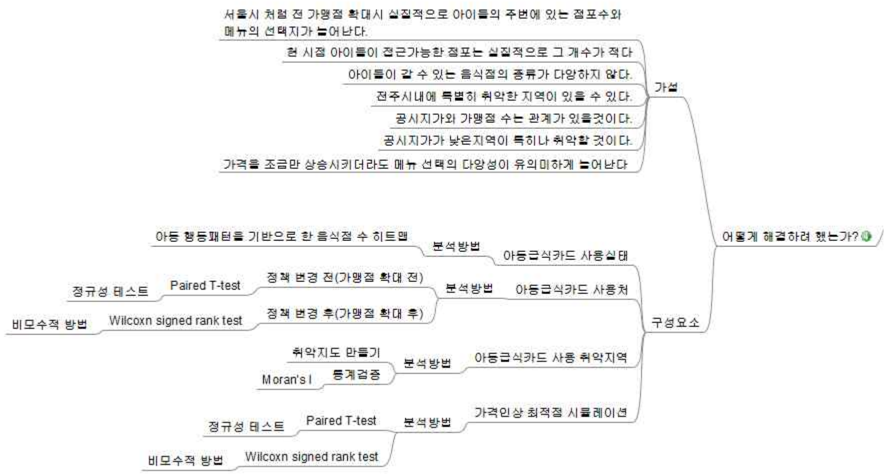
7.2 마인드맵






왜 필요한가?
현상
가맹점 수의 부족으로 멀리까지 아이들이 나가야 하는 상황 발생
특정 단말기만 사용가능한 카드의 특성으로 인해 아이들에게 낙인감이라는 부정적 영향을 끼침
편의점 대부분의 점포가 편의점에 몰려있기 때문에 영양소를 골고루 섭취 힘듦
지원금액의 부족으로 인해 사용처 제한받고 있음
필요성
아동의 기본적 생활보장(기본권)
국가발전을 위한 인력보호
앞으로 발생할 수 있는 사회문제 발생의 예방
관심의 부족
급식카드 유무는 최근 많은 매체를 통해 인식하고 있음
선한영향력 가게
홍대 철인7호치킨집, 진짜파스타 등 대표적 선행가게
실질적으로 무엇을 먹는지는 관심이 없음
아동의 식생활은 신체 성장과 인지발달에 영향
전 생애 주기에 영향을 미침
결식이 우려되는 아이들에 대한 관심과 지원은 복지의 측면으로 볼 때 필요성이 충분함
문제점
가맹점 수의 절대적인 부족
전주의 현황
2020년 기준 가맹점 전주시 내 238개소
특정 단말기에 반응하는 카드 사용으로 가맹점 수 확대 곤란
실질 사용 불편
낙인감
IC칩이 없는 카드, 특이한 디자인으로 결제시 경제적 여건 노출
항상 결제가능한 점포인지 확인후 사용해야 함
성장기의 어린아이들이 동급생 등 주변 친구들에게 사용하는 모습을 보여주기 꺼려함
편의점 중심
음식점의 개소가 절대적으로 부족하여 주로 편의점에서 사용중
카드 통합 단말기를 비치한 곳이 편의점이 대다수
지원금의 한도 부족으로 인한 편의점 이용이 주를 이룸
구매품목의 제한
가장 많이 사먹는 도시락도 유명 마케팅 상품은 구매불가
비빔등류, 씨리얼 등 분류기준이 난해하여 구입할 수 없는 경우가 존재
점주나 직원의 개인적 판단에 따라 품목이 다르게 제한되기도 함.
금액의 부족으로 인해 사용할 수 있는 범위가 적음
공급자 중심에서 시행되고 있는 현황
단순히 이용할 수 있는 음식점의 수가 몇개인지는 공급자 중심의 시각임
가맹점 신청시 정부에 추가서류를 제출하는 등 처리절차 복잡
성장기 아이들에게 영양학적 불균형 초래
영양소가 충분하지 못한 음식들 위주(삼각김밥, 도시락 등) 성장기 어린이에게 충분한 영양소 제공 못함
과일 등 채소류를 추가로 구매하기 어려움
조례와는 다른 시행현황
법률내에서 영양학적으로 품질이 좋은 음식 등 공급되도록 노력해야 하지만 실행되고 있지 않다.
사례
홍보 보고서
아동급식카드 기능 부실(지역 안 가맹점이 충분히 존재하지 않다는 기사)
코로나로 인한 비대면 급식지원 미비 (기사)
급식카드, 사먹을 식당 심지어 편의점마저 없는 곳 구매위해 40분 이동(기사)


무엇으로 해결하려 했는가?
해결방안
서울시, 경기도, 부산 모델과 같이 은행과 연계하여 가맹점 수 확대 진행
서울시 -> 신한은행, 부산시 -> 기업은행 등 시중은행과 연계하여 점포수를 확대함
지역 공동체 참여자인 전북은행 카드 사용가능 점포를 기반으로 가맹점 수 확대
지역 공동체 참여자인 전북은행을 통해 가맹점 수 확대
지역내 많은 소상공인에게 수혜가 돌아감으로써 지역경제 활성화 가능
아동 급식카드 사용 가능한 소규모 점포 확대
조례등의 개정으로 소규모 점포들을 확대하여 더 많은 가맹점 공급
공급자 중심에서 바라본 시각을 수요자 측면에서 해석
아이가 충분히 갈 수 있는 거리에 환경조성
취약지도 및 위험지도를 만들어 우선순위 선정
수요자 중심의 시각
아이들이 편안하게 걸어서 갈 수 있는 거리는?
수혜 대상
초등학생
중학생
고등학생
Perry의 그린 주거이론
대한민국 신도시 도시설계의 기초 이론
초등학교 등 학교시설을 기반
초등학교 반경 400m
중, 고등학교 반경 700m
아이들 주변에 어느정도의 가맹점포가 위치하는지
단순히 존재해서는 안된다. 사용자 주변에 위치해야 한다.
주 활동반경 내 가맹점 개수
학교를 중심으로 반경 400m, 700m
주거지역 내 점포 개수
집 주변 점포개수 (400m, 700m)
아동급식카드 금액상승
지자체 예산의 한계
현행에서 점진적으로 금액을 상승시킬때 최초로(최저) 유의미한 통계적 차이를 보이는 지점 파악 후 제안
우수사례
서울시 아동급식 꿈나무카드 30일부터 식당 13만곳 쓴다.(기사)
부산시 아동급식카드, 1일부터 '일반음식점' 4만 9천곳 사용 확대(기사)
인천 결식아동 급식카드 사용처 확대(기사)


어떻게 해결하려 했는가?
가설
서울시 처럼 전 가맹점 확대시 실질적으로 아이들의 주변에 있는 점포수와 메뉴의 선택지가 늘어난다.
현 시점 아이들이 접근가능한 점포는 실질적으로 그 개수가 적다
아이들이 갈 수 있는 음식점의 종류가 다양하지 않다.
전주시내에 특별히 취약한 지역이 있을 수 있다.
공시지가와 가맹점 수는 관계가 있을것이다.
공시지가가 낮은지역이 특히나 취약할 것이다.
가격을 조금만 상승시키더라도 메뉴 선택의 다양성이 유의미하게 늘어난다
구성요소
아동급식카드 사용실태
분석방법
아동 행동패턴을 기반으로 한 음식점 수 히트맵
아동급식카드 사용처
분석방법
정책 변경 전(가맹점 확대 전)
정책 변경 후(가맹점 확대 후)
Paired T-test
정규성 테스트
Wilcoxn signed rank test
비모수적 방법
아동급식카드 사용 취약지역
분석방법
취약지도 만들기
통계검증
Moran's I
가격인상 최적점 시뮬레이션
분석방법
Paired T-test
정규성 테스트
Wilcoxn signed rank test
비모수적 방법

7.3 분석 상세코드

(전처리)

1.모듈 import

```
import pandas as pd
import geopandas as gpd

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

import requests
import re
import html
from bs4 import BeautifulSoup
```

2. Web Crowling

```
# 1) html 불러오는 함수 정의

def get_html(url):
    _html = ""
    resp = requests.get(url)
    if resp.status_code == 200:
        _html = resp.text
    return _html
```

```
# 2)-1 가맹점 이름 크롤링

merchant_name = []
url = 'https://www.purmeecard.com/public.do?request=merchantSelect&gu=&name=&si=08&curPage='

for i in range(1,29):
    n_url = f'{url}{i}'
    html = get_html(n_url)
    soup = BeautifulSoup(html, 'html.parser')
    b = soup.find_all('h3')
    tmp = []
    for i in b:
        tmp.append(i.string)
    merchant_name.append(tmp)
```



```
# 2)-2 크롤링한 데이터 -> 데이터프레임
m_list = [name for names in merchant_name for name in names]
merchant = pd.DataFrame(m_list)
merchant.columns = ['m_name']
merchant.head()
```

	m_name
0	CU (구,훼미리마트)(전주금암점)
1	금암면옥
2	대성고기백화점
3	또또분식
4	롯데리아 전주터미널

```
# 3)-1 주소 크롤링
merchant_address = []
url = 'https://www.purmeecard.com/public.do?request=merchantSelect&gu=&name=&si=08&curPage='

for i in range(1,29):
    n_url = f'{url}{i}'
    html = get_html(n_url)
    soup = BeautifulSoup(html, 'html.parser')
    b = soup.find_all('p',{'class' : 'text-left'})
    tmp = []
    for i in b:
        tmp.append(i.string)
    merchant_address.append(tmp)
```

```
# 3)-2 주소 전처리
ma_list = [name for names in merchant_address for name in names]
ma_list

# 텍스트 처리
result = []
for i in ma_list:
    tmp = i.replace("\xa0\xa0","")
    result.append(tmp)

ma = pd.DataFrame(result)
ma.columns = ['address']
```

```
# 4) 주소랑 가맹점이름 concat
merchant_list = pd.concat([merchant,ma], axis = 1)
#merchant_list.to_csv('가맹점list.csv', encoding = 'cp949'
```

**가맹점list.CSV 파일 전처리

- geocoder 프로그램으로 가맹점의 좌표를 따왔다
- CU에 훼미리 마트 등 방해되는 요소들 처리

```
merchant_list = pd.read_csv('가맹점 좌표.csv', encoding = 'euc-kr')
```

```
merchant_list = merchant_list.drop(['No','Unnamed: 1','상태'], axis = 1)
merchant_list.columns = ['m_name','address', 'lon(x)','lat(y)']
merchant_list
```

	m_name	address	lon(x)	lat(y)
0	CU (전주금암점)	전라북도 전주시 덕진구 금암1동 기린대로 398	127.132497	35.837687
1	금암면옥	전주시 덕진구 기린대로 400-3	127.132278	35.838063
2	대성고기백화점	전라북도 전주시 덕진구 태진로 137-4	127.134912	35.833377
3	또또분식	전주시 덕진구 삼송3길 14	127.135447	35.842415
4	롯데리아 전주터미널	전주시 덕진구 가리내로 21	127.133605	35.834042
...	...	...	...	...
306	본죽&비빔밥 전주전북도청점	전라북도 전주시 완산구 홍산남로 45	127.107015	35.816051
307	아딸(감탄떡볶이) 신시가지 문학점	전북 전주시 완산구 문학대4길 31	127.105257	35.826899
308	암스(서부신시가지점)	전북 전주시 완산구 문학대4길 28-4	127.104703	35.826620
309	이삭토스트	전북 전주시 완산구 문학로 19	127.101724	35.826926
310	홍산김밥	전북 전주시 완산구 마전들로 36	127.099512	35.820840

311 rows × 4 columns

```
# 4)-1 가맹점 업종분류 : 가맹점 중 편의점만 추출

cu = merchant_list[merchant_list['m_name'].str.contains('CU')]
emart24 = merchant_list[merchant_list['m_name'].str.contains('이마트24')]

cv = pd.concat([cu,emart24], axis=0)
cv = cv[~cv['m_name'].str.contains('완산구')]
cv = cv[~cv['m_name'].str.contains('덕진구')] #주소가 확실하지 않으므로 제거
cv['kind'] = '편의점'
```



```
# 4)-2 가맹점 이름별로 업종분류하기
a = merchant_list[(merchant_list['m_name'].str.contains('카페')) | (merchant_list['m_name'].str.contains('레드'))|
                  (merchant_list['m_name'].str.contains('파리바')) | (merchant_list['m_name'].str.contains('우레쥬르'))|
                  (merchant_list['m_name'].str.contains('브')) | (merchant_list['m_name'].str.contains('샌맨점')) | (merchant_list['m_name'].str.contains('빵')) |
                  (merchant_list['m_name'].str.contains('하우스')) | (merchant_list['m_name'].str.contains('아띠')) | (merchant_list['m_name'].str.contains('봄폼'))|
                  (merchant_list['m_name'].str.contains('커피')) | (merchant_list['m_name'].str.contains('소풍'))|(merchant_list['m_name'].str.contains('풍샌'))]
a['kind'] = '카페'

b = merchant_list[(merchant_list['m_name'].str.contains('분식')) |(merchant_list['m_name'].str.contains('길')) | (merchant_list['m_name'].str.contains('꼬치')) |
                  (merchant_list['m_name'].str.contains('김밥')) |(merchant_list['m_name'].str.contains('아딸')) |(merchant_list['m_name'].str.contains('감탄'))|
                  (merchant_list['m_name'].str.contains('암스')) |(merchant_list['m_name'].str.contains('이삭')) | (merchant_list['m_name'].str.contains('신포'))|
                  (merchant_list['m_name'].str.contains('봉구스')) | (merchant_list['m_name'].str.contains('랩토'))| (merchant_list['m_name'].str.contains('김밥'))|
                  (merchant_list['m_name'].str.contains('떡볶이'))|(merchant_list['m_name'].str.contains('우동'))]
b['kind'] = '분식'

c = merchant_list[(merchant_list['m_name'].str.contains('피자')) | (merchant_list['m_name'].str.contains('오구몽가')) | (merchant_list['m_name'].str.contains('롯데리아'))|
                  (merchant_list['m_name'].str.contains('맘스')) | (merchant_list['m_name'].str.contains('한스'))]
c['kind'] = '양식'

d = merchant_list[(merchant_list['m_name'].str.contains('각|청혼|짜장|짬뽕|반점|빈|중식|정무문'))]
d['kind'] = '중식'

e = merchant_list[merchant_list['m_name'].str.contains('슈퍼|마트|마켓|쇼핑|전주푸드|굿모닝')]
e['kind'] = '마트'

f = merchant_list[(merchant_list['m_name'].str.contains('돈|마시루'))]
f['kind'] = '일식'

list1 = [a.index, b.index, c.index, d.index, e.index, f.index, cv.index]
list1 = [name for names in list1 for name in names] #리스트안의 리스트 해제

idx = list(set(merchant_list.index) - set(list1))
g = merchant_list.loc[idx] #한식
g['kind'] = '한식' #분류안된 가맹점은 다 한식으로

merchant_list_k = pd.concat([a,b,c,d,e,f,g,cv], axis = 0).reset_index().drop(['index'],axis =1) # 하나의 데이터프레임으로 합치기
merchant_list_k
```

```
merchant_list_k[merchant_list_k['m_name'].duplicated().values] #실행X, 중복확인
```

	m_name	address	lon(x)	lat(y)	kind
106	본김밥카페	전주 덕진구 소라로 181	127.125451	35.852243	분식
119	이삭토스트	전북 전주시 완산구 문학로 19	127.101724	35.826926	분식
187	롯데리아 롯데마트전주송천점	전라북도 전주시 덕진구 송천중앙로 82	127.120460	35.854574	마트
193	홈마트	전북 전주시 덕진구 가재미3길 15	127.160661	35.828653	마트
219	롯데리아 롯데마트전주점	전북 전주시 완산구 우전로 240	127.102511	35.816333	마트

```
# 4)-3 중복처리 - 중복아닌 메뉴 분리 후 중복메뉴 삭제
dup = merchant_list_k.drop_duplicates('m_name')
again = merchant_list_k.iloc[[193,119]]

##가맹점 마지막으로 concat
merchant_final = pd.concat([dup,again], axis =0).reset_index().drop(['index'],axis =1)
merchant_final
```

3. 전주시 식당, 메뉴 정리

```
menu_df = pd.read_csv("C:/Users/user/Documents/pb/jeonju_menu.csv", encoding = 'utf-8')
res_df = pd.read_csv("C:/Users/user/Documents/pb/jeonju_res.csv", encoding = 'utf-8')
```

```
# 1)-1 전처리 - 필요없는 열 제거
menu_df = menu_df.drop(['메뉴명-로마자', '메뉴태그-영어', '메뉴태그-일본어', '메뉴태그-중국어(간체)'], axis=1)
menu_df.head()
```

```python
# 안쓰는 칼럼 제거
res_df = res_df.drop(['테이크아웃여부', '예약가능여부', '선결제여부', '식당 대표전화번호',
       '도로명주소', '지번주소', '정규휴업일', '식당면적', '어워드_글로벌',
       '어워드_로컬', '트립어드바이저 인기도', '씨트립 인기도', '네이버 인기도', 'RTI 인덱스', '온라인화 진행 여부',
       '다국어메뉴판 제공 여부', '다국어 지원언어', '다국어메뉴판 주소', '수용태세 지수', '좌석수(입식)',
       '좌석수(좌식)','반려동물출입가능여부', '주차가능여부', '식당근처랜드마크정보', '랜드마크위도',
       '랜드마크경도', '식당 홈페이지', '등록일시', '식당명-로마자', '업종정보명-영어', '업종정보명-일본어',
       '업종정보명-중국어(간체)', '도로명주소-영어', '도로명주소-일본어','베리어프리여부', '도로명주소-중국어(간체)', '지번주소-영어',
       '지번주소-일본어', '지번주소-중국어(간체)', '영업시간정보','대표메뉴', '휴식시간정보'], axis = 1)

res_df.columns = ['식당ID', '식당명', '업종', '위도', '경도']
res_df.head(3)
```

	식당ID	식당명	업종	위도	경도
0	1606	피자애 송천점	피자	35.861649	127.126325
1	2819	일등낙지 전주본점	낙지	35.816124	127.108326
2	2999	비공식	일식	35.818546	127.139333

```python
# 식당ID 중복여부 확인 ; menudata는 식당의 메뉴마다 데이터가 있으므로 식당ID의 unique_value 개수로 확인
print(len(res_df))
print(len(menu_df['식당ID'].unique()))
```

```
3018
3018
```

```python
# 1)-2 merge - 식당과 메뉴데이터 식당ID 기준으로
df_merge = pd.merge(res_df, menu_df, on='식당ID')

menu_drop = df_merge['업종'].isin(['카페', '맥주', '와인', '술집', '포장마차'])
df_merge = df_merge[~menu_drop]

# 중복되는 열 삭제
df_merge = df_merge.drop('식당명_y', axis=1)

# 열이름 변경
df_merge.columns = [['식당ID', '식당명', '업종', '위도', '경도', '메뉴ID', '메뉴명', '메뉴가격', '메뉴태그정보']]
df_merge = df_merge.reset_index().drop(['index'], axis = 1)
df_merge = df_merge.drop(['메뉴태그정보'], axis= 1)
df_merge
```

```python
df_merge.isnull().sum() #결측치 확인 실행 X
```

```
식당ID          0
식당명           0
업종        16760
위도            0
경도            0
메뉴ID          0
메뉴명           0
메뉴가격          0
메뉴태그정보     7390
dtype: int64
```

```python
# 1)-3 merge로 병합하면 Series가 나오지 않으므로 분리해서 다시 결합

a = pd.Series(df_merge['식당ID'].values.squeeze())
b = pd.Series(df_merge['식당명'].values.squeeze())
c = pd.Series(df_merge['업종'].values.squeeze())
d = pd.Series(df_merge['위도'].values.squeeze())
e = pd.Series(df_merge['경도'].values.squeeze())
f = pd.Series(df_merge['메뉴ID'].values.squeeze())
g = pd.Series(df_merge['메뉴명'].values.squeeze())
h = pd.Series(df_merge['메뉴가격'].values.squeeze())

df_reshape = pd.concat([a,b,c,d,e,f,g,h], axis = 1)
df_reshape.columns = ['res_ID','res_name','kind','lat(y)','lon(x)','menu_ID','menu_name','price']
df_reshape
```


2021년 공공 빅데이터 분석청년 인재양성 - 실무형프로젝트 결과보고서

```python
# 1)-4 전처리 - Price (df_reshape에서 prcie 0인것을 확인 후 살릴 것을 찾음)

price0 = df_reshape.loc[df_reshape['price'] == 0] # 인터넷 서핑 결과 : 돈가스집, 정명기김밥 제육덮밥 7000원, 본토홍반점, BHC 전주에코시티점, 필스수제버거, 로띠팜
possible = price0[price0['res_name'].str.contains('돈가스집|정명기김밥|필스|로띠')]
df_reshape_2 = df_reshape.loc[~(df_reshape['price'] == 0)]
```

```python
# 결측치 처리

possible['price'][(possible['res_name'] == '돈가스집')] = 8500 # 메뉴 평균가
possible['price'][(possible['res_name'] == '정명기김밥')] = 7000 #해당 메뉴가격 7000원
possible['price'][(possible['res_name'] == '필스수제버거')] = 3900 #과일은 의미없다 판단 에그샌드위치 가격으로 통일
possible['price'][(possible['res_name'] == '로띠팜')] = 7700 #메뉴 평균가

possible #결측치 처리한 데이터셋 (1)
```

```python
# 1) -5 주류 및 추가,사리메뉴 ouT !

## 1 주류관련 단어, 추가,사리관련단어 포함된 메뉴 제거
menu_drop_1 = df_reshape_2['menu_name'].str.contains('뽕주|콜라|사이다|마운틴|소주|맥주|모주|막걸리|환타|\
치즈모듬|추가|사리|대포|청하|백세주|산사춘|참이슬|하이트|좋은데이|처음처럼|매화수|복분자|심술|진로|테라|\
국순당|클라우드|이슬톡톡|토닉|버드와이저|호가든|카프리|하이네켄|볼랑|코젤|한라산|대선|잎새주|아사히|칭따오|고량')
df_reshape_3 = df_reshape_2[~menu_drop_1]

## 2 카스테라라는 메뉴가 있기 때문에 카스와 테라는 찾아서 정확하게 일치하는 데이터만 제거
menu_drop_2 = df_reshape_3['menu_name'].isin(['카스/하이트/하얼빈','카스','카스(330ml)','카스300ml','카스(500cc)','카스(1,700cc)',
                                              '카스(3,000cc)','카스(500ml)','맥스','맥스(330ml)','테라','테라(2000cc)','테라(3000cc)',
                                              '테라(500ml)','테라(3병)','테라3병'])
df_reshape_3 = df_reshape_3[~menu_drop_2]

## 3 술집관련된 이름을 가진 식당 제거
menu_drop_3 = df_reshape_3['res_name'].str.contains('호프|가맥|시장집|주점|맥주|까투리|제주|닭발|풍미|족발|마시로이|아프로|황제|만족|육회|오뎅|뚱보집|초장집|도시어부') #술집과 관련된 매장 삭제
df_reshape_3 = df_reshape_3[~menu_drop_3]

## 4 #1500원이하 메뉴 삭제(사리가 대부분)
df_reshape_3 = df_reshape_3.loc[~(df_reshape_3['price'] <= 1500)]

# 5 그럼에도 불구하고 업종분류 결측치 처리
kind_na = df_reshape_3[df_reshape_3['kind'].isnull()] #업종분류가 결측치인것만 새로운 데이터 셋
df_reshape_3 = df_reshape_3[~(df_reshape_3['kind'].isnull())] #이를 제외한 데이터셋 (2)

df_reshape_3
```

```python
# 2)-1 이름으로 업종 분류 하기

a = kind_na[(kind_na['res_name'].str.contains('카페|와플|제과|도넛|원조중앙찹쌀|비어드파파|돌돌|초코|마카롱|꽈배기|던킨|설빙|달달|호떡당|명과|캘리|로띠팜|도너츠|쥬씨\
와플|더리터'))]
a['kind'] = '카페'

b = kind_na[(kind_na['res_name'].str.contains('분식|튀김|box'))]
b['kind'] = '분식'

c = kind_na[(kind_na['res_name'].str.contains('피자|르엘리스|파스타|맘스터치|아이마미따|돈스엔스파'))]
c['kind'] = '양식'

d = kind_na[(kind_na['res_name'].str.contains('소호|짜장|짬뽕|반점|중식|우리밀|마라탕|소림사|향|송천궁|청하|수타|신대유성|백자원|장가계|루|사천성|중화요리|중화정|\
산돌|츠츠허허|중식|양세마리|마라|호남성|태풍화|설궁|동보성|향후항|라화|깐쇼|장수원'))]
d['kind'] = '중식'

e = kind_na[(kind_na['res_name'].str.contains('치킨|페리카나|닭|통닭|BBQ|지코바|후라이드|오꾸꼬|가마꿉|솜키친'))]
e['kind'] = '치킨'

f = kind_na[(kind_na['res_name'].str.contains('돈카츠|마시루|초밥|스시|쿠우쿠우|돈까스|돈쿡|꼬치|카츠|성자동|어락|동경|돈페플러스|황금손가락|유쿄니|진까|호천당\
소바|돈갓|까스|멘야케이|고자루|돈기부여|돈가스|가츠연'))]
f['kind'] = '일식'

g = kind_na[(kind_na['res_name'].str.contains('정육|꽃순이|육감만족|삼겹살|원조|구이|조개|바다|김돈이|고깃|해물|숯불|돼지|대게|고기|\
낙지|갈비|곰탕|고궁|갈비|연와미당|비빔밥|신날개|21세기꿀호떡|빨간맛도리|미가옥|전주부림온|개성집|장어|오징어|해장국|명작|도마|아구|동태|\
국밥|보쌈|평화돈|돈벗|쭈꾸미|고마담|찜닭|두부|갈매기|도시락|삼미당|소곰집|순대국|냉삼이|가본|한솔|참돈|화로로|돌판집|대방|이차돌|오겹살\
우리돈'))]
g['kind'] = '한식'

h = kind_na[(kind_na['res_name'].str.contains('쿠우쿠우|부림온|해요'))]
h['kind'] = '뷔페'

df_final = pd.concat([a,b,c,d,e,f,g,h,possible,df_reshape_3], axis = 0).reset_index().drop(['index'],axis =1) #(1),(2) 데이터셋도 같이 합침
df_final
```

```python
df_final['kind'].unique() #업종분류 확인, 실행 X
```

```
array(['카페', '분식', '양식', '중식', '치킨', '일식', '한식', '뷔페', '돈가스', '김밥', '햄버거',
       nan, '피자', '낙지', '곰탕', '케이크전문', '파스타', '육류', '과일주스', '한정식', '차',
       '커리', '곱창,막창,양', '우동', '생선회', '소고기구이', '베트남음식', '죽', '만두', '돼지고기',
       '닭갈비', '초밥', '해물', '태국음식', '찌개,전골', '아이스크림', '베이커리', '이탈리아음식',
       '샌드위치', '순대국', '샤브샤브', '도시락', '퓨전음식', '두부요리집', '국밥', '닭요리', '막국수',
       '라멘', '족발', '보리밥', '양꼬치', '찜닭', '국수', '도너츠', '백반', '아귀찜', '샐러드',
       '오리요리', '장어', '음식점', '한식뷔페', '주꾸미', '해장국', '브런치', '감자탕', '쌈밥',
       '칼국수', '아시아음식', '추어탕', '핫도그', '냉면', '푸드코트', '복어요리', '사철,영양탕', '백숙',
       '떡볶이', '양갈비', '와플', '매운탕', '패밀리레스토랑', '호두과자', '닭발', '빙수', '오뎅',
       '곱창', '생선구이', '보쌈', '설렁탕', '토스트', '순대', '해물탕', '전복요리', '닭강정', '전',
       '게요리', '삼계탕'], dtype=object)
```

```python
# kind 정리
change_value_dict = {'카페' : '카페', '분식' : '분식', '양식':'양식', '중식':'중식', '치킨':'치킨', '일식':'일식', '한식':'한식', '뷔페':'뷔페', '돈가스':'일식', '김밥':'분식', '햄버거':'양식',
                     '피자':'양식', '낙지':'한식', '곱탕':'한식', '케이크전문':'카페', '파스타':'양식', '육류':'한식', '과일주스':'카페', '한정식':'한식', '차':'카페','커리':'아시안', '곱창,막창,양':'한
                     '우동':'일식', '생선회':'한식', '소고기구이':'한식', '베트남음식':'아시안', '죽':'한식', '만두':'중식', '돼지고기':'한식','닭갈비':'한식', '초밥':'일식', '해물':'한식', '태국음식':
                     '찌개,전골':'한식', '아이스크림':'카페', '베이커리':'카페', '이탈리아음식':'양식','샌드위치':'카페', '순대국':'한식', '샤브샤브':'한식', '도시락':'한식', '퓨전음식':'한식', '두부요
                     '국밥':'한식', '닭요리':'한식', '막국수':'한식','라멘':'일식', '족발':'한식', '보리밥':'한식', '양꼬치':'중식', '찜닭':'한식', '국수':'한식', '도너츠':'카페', '백반':'한식', '아귀
                     '샐러드':'양식','오리요리':'한식', '장어':'한식', '음식점':'한식', '한식뷔페':'뷔페', '주꾸미':'한식', '해장국':'한식', '브런치':'양식', '감자탕':'한식', '쌈밥':'한식','칼국수':'한
                     '아시아음식':'아시안', '추어탕':'한식', '핫도그':'양식', '냉면':'한식', '푸드코트':'분식', '복어요리':'한식', '사철,영양탕':'한식', '백숙':'한식','떡볶이':'분식', '양갈비':'아시안'
                     '와플':'카페', '매운탕':'한식', '패밀리레스토랑':'양식', '호두과자':'카페', '닭발':'한식', '빙수':'카페', '오뎅':'한식', '곱창':'한식', '생선구이':'한식', '보쌈':'한식', '설렁탕':'
                     '토스트':'분식', '순대':'한식', '해물탕':'한식', '전복요리':'한식', '닭강정':'한식', '전':'한식','게요리':'한식', '삼계탕':'한식' }

df_final = df_final.replace({'kind':change_value_dict})
df_final.head()
```

(분석)

```python
# -*- coding: utf-8 -*-
"""
Created on Mon Aug  9 09:51:59 2021

script for spatial analysis (Spatial Autocorrelation and Hotspot Analysis)

author : Daegun Kim
"""

#%% Import packages
import esda
import pandas as pd
import geopandas as gpd
from geopandas import GeoDataFrame
import libpysal as lps
import numpy as np
import matplotlib.pyplot as plt
from shapely.geometry import Point

#%% Fix random seed for reproduction
def fix_random_seed(seed=42):
    import random
    import numpy as np
    import os

    random.seed(seed)
    np.random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
    try:
        tf.random.set_seed(seed)
    except:
        pass

fix_random_seed()

#%% Custom functions
def save_fig(save_dir, figure, title=None, close_fig=True):
    dir_org = os.getcwd()
    os.chdir(save_dir)

    figure.savefig('{}.{}'.format(title, fig_file_ext),
                   dpi=high_dpi)
    if close_fig:
        plt.close(figure)

    os.chdir(dir_org)


def get_all_attr(price, price2=None, min_th_dist=1.1, mode='policy'):
    from eda import load_map_rest_join

    if mode == 'policy':
        gdf_base1, gdf_base2 = load_map_rest_join(pri_thld=price)
    elif mode == 'before':
        gdf_base1, _ = load_map_rest_join(pri_thld=price)
        gdf_base2, _ = load_map_rest_join(pri_thld=price2)
    elif mode == 'after':
        _, gdf_base1 = load_map_rest_join(pri_thld=price)
        _, gdf_base2 = load_map_rest_join(pri_thld=price2)
    else:
        print("mode should be policy or before or after")
        raise NotImplementedError

    gdf_set = GdfCompare(gdf_base1, gdf_base2, min_th_dist=min_th_dist)
    gdf_set.run_local_g()
    gdf_set.run_stats_diff()
    gdf_set.run_global_moran()

    return gdf_set

#%% Custom classes
class GdfCompare:
    def __init__(self, gdf1, gdf2, min_th_dist=1.1):
        self.gdf_before = gdf1
        self.gdf_after = gdf2
        self.alternative = 'less'
        self.t_test = False
        self.wrst = False
        self.min_thld_distance = min_th_dist

    def run_stats_diff(self):
        import math
        from scipy.stats import shapiro, ttest_rel

        sw_jj_frchs = shapiro(self.gdf_before.mean_rest)
        sw_jj = shapiro(self.gdf_after.mean_rest)
```

```python
    def run_stats_diff(self):
        import math
        from scipy.stats import shapiro, ttest_rel

        sw_jj_frchs = shapiro(self.gdf_before.mean_rest)
        sw_jj = shapiro(self.gdf_after.mean_rest)

        if sw_jj_frchs.pvalue < 0.05 and sw_jj.pvalue < 0.05:
            t_test = ttest_rel(self.gdf_before.mean_rest,
                               self.gdf_after.mean_rest,
                               alternative=self.alternative)
            # H0 : Mean values are identical
            # H1 : Mean of a is less than the mean of b
            if math.isnan(t_test.pvalue):
                wrst = wilcoxon(self.gdf_before.mean_rest,
                                self.gdf_after.mean_rest,
                                alternative=self.alternative,
                                zero_method='zsplit')
                self.wrst = True
                self.wrst_s = wrst.statistic
                self.wrst_pval = wrst.pvalue
            else:
                self.t_test = True
                self.t_test_s = t_test.statistic
                self.t_test_pval = t_test.pvalue
        else:
            wrst = wilcoxon(self.gdf_before.mean_rest,
                            self.gdf_after.mean_rest,
                            alternative=self.alternative,
                            zero_method='wilcox')
            # H0 : Two related paired samples come from the same distribution
            # H1 : Median of a is less than the median of b
            # print('Wilcoxon Signed Rank Test was done')
            self.wrst = True
            self.wrst_s = wrst.statistic
            self.wrst_pval = wrst.pvalue


    def run_global_moran(self):
        wq_before =  lps.weights.Queen.from_dataframe(self.gdf_before)
        wq_before.transform = 'r'

        wq_after =  lps.weights.Queen.from_dataframe(self.gdf_after)
        wq_after.transform = 'r'

        y_before = self.gdf_before.mean_rest

        y_after = self.gdf_after.mean_rest

        mi_before = esda.Moran(y_before, wq_before)
        mi_after = esda.Moran(y_after, wq_after)

        self.glo_m_before = [mi_before.I, mi_before.p_sim]
        self.glo_m_after = [mi_after.I, mi_after.p_sim]


    def run_local_g(self):
        import libpysal

        # Get Getis Ord Local
        cent_jj = self.gdf_before.geometry.centroid
        xys = pd.DataFrame({'X': cent_jj.x, 'Y': cent_jj.y})
        min_wt_jj_thld = libpysal.weights.util.min_threshold_distance(xys)\
            *self.min_thld_distance
        wt_jj = libpysal.weights.DistanceBand(xys, threshold=min_wt_jj_thld)

        lg_gdf_before = esda.getisord.G_Local(self.gdf_before.mean_rest,
                                              wt_jj, transform='r')
        self.lg_gdf_before = lg_gdf_before

        lg_gdf_after = esda.getisord.G_Local(self.gdf_after.mean_rest,
                                             wt_jj, transform='r')
        self.lg_gdf_after = lg_gdf_after

        self.gdf_before['lg_p_sim'] = lg_gdf_before.p_sim
        self.gdf_before['lg_Zs'] = lg_gdf_before.Zs

        self.gdf_after['lg_p_sim'] = lg_gdf_after.p_sim
        self.gdf_after['lg_Zs'] = lg_gdf_after.Zs

        # Classify the hotspot classes
        self.gdf_before['hotspot_class'] = np.nan
        sig_gdf_before = self.gdf_before.lg_p_sim < 0.05

        self.gdf_before.loc[sig_gdf_before==False, 'hotspot_class'] = 0
```



```python
    def plot_compare(self):
        import matplotlib.patches as mpatches

        fig, ax = plt.subplots(1, 2, figsize=(18, 9))

        LegendElement = [mpatches.Patch(facecolor='r',
                                        edgecolor='k', label='Hot Spot'),
                         mpatches.Patch(facecolor='lightgrey',
                                        edgecolor='k', label='Non Significant'),
                         mpatches.Patch(facecolor='blue',
                                        edgecolor='k', label='Cold Spot')]

        sig_gdf_before = self.gdf_before.lg_p_sim < 0.05
        hh_jj_frchs = self.gdf_before[(sig_gdf_before==True) &
                                      (self.gdf_before.lg_Zs > 0)].plot(
            ax=ax[0], color='red', edgecolor='k', linewidth=0.1,
            legend=True, categorical=True)
        ns_jj_frchs = self.gdf_before[sig_gdf_before==False].plot(
            ax=ax[0], color='lightgrey', edgecolor='k', linewidth=0.1,
            label='Non Significant')
        ll_jj_frchs = self.gdf_before[(sig_gdf_before==True) &
                                      (self.gdf_before.lg_Zs < 0)].plot(
           ax=ax[0], color='blue', edgecolor='k', linewidth=0.1,
           label='Cold Spot')
        ax[0].axis(False)
        ax[0].legend(handles=LegendElement, loc='upper right')

        sig_gdf_after = self.gdf_after.lg_p_sim < 0.05
        hh_jj = self.gdf_after[(sig_jj==True) &
                               (self.gdf_after.lg_Zs > 0)].plot(
            ax=ax[1], color='red', edgecolor='k', linewidth=0.1,
            label='Hot Spot')
        ns_jj = self.gdf_after[sig_jj==False].plot(
            ax=ax[1], color='lightgrey', edgecolor='k', linewidth=0.1,
            label='Non Significant')
        ll_jj = self.gdf_after[(sig_jj==True) &
                               (self.gdf_after.lg_Zs < 0)].plot(
            ax=ax[1], color='blue', edgecolor='k', linewidth=0.1,
            label='Cold Spot')
        ax[1].axis(False)
        ax[1].legend()
        ax[1].legend(handles=LegendElement, loc='upper right')

        return fig

#%% Main script
if __name__ == '__main__':

    #% Import modules
    import os
    import time
    import sys
    import matplotlib as mpl
    import geopandas as gpd
    import multiprocessing
    import seaborn as sns
    import math
    import pyproj
    import re
    import libpysal
    import pickle
    import matplotlib.patches as mpatches

    from tqdm import tqdm
    from matplotlib import pyplot as plt
    from shapely.geometry import Polygon, LineString, Point
    from scipy.stats import shapiro, ttest_rel, wilcoxon

    from eda import load_map_rest_join

    tot_run_time_start = time.time()

    #%% Make dir for saving figures
    dir_nm_save_fig = 'Spatial_Analysis_plot'
    try:
        os.mkdir(dir_nm_save_fig)
    except FileExistsError:
        pass

    #%% Plot style
    plt.style.use('default')
    # possible options :
    # seaborn, dark_background, bmh, ggplot, fivethirtyeight...
```

```python
#%% Sample Test
# ---------------------------------------------------------------------------
gdf_base1, gdf_base2 = load_map_rest_join(pri_thld=6000)
baseline = GdfCompare(gdf_base1, gdf_base2)
baseline.run_local_g()
baseline.run_stats_diff()
baseline.run_global_moran()
baseline_fig = baseline.plot_compare()

save_fig(dir_nm_save_fig, baseline_fig, title='Baseline Hotspot Analysis')

#%% Difference check when applyging new policy by baseline
gdf_before_low_med = \
    baseline.gdf_before[baseline.gdf_before.mean_rest <
                        baseline.gdf_before.mean_rest.median()]

#%% price increasing test under condition prior to new policy
pri_inc_wo_plcy = []
for price in tqdm(np.arange(6000, 10500, 500),
                  desc='Price increase test prior to the new policy'):
    pri_inc_wo_plcy.append(
        get_all_attr(6000, price, mode='before')
        )

#%% Get results from price increasing test prior to the new policy
df_pri_inc_pr = pd.DataFrame(index=np.arange(6000, 10500, 500))
df_pri_inc_pr['stats_diff'] = \
    ['t' if x.t_test else 'w' for x in pri_inc_wo_plcy]
df_pri_inc_pr['statistic'] = \
    [x.t_test_s if x.t_test else x.wrst_s for x in pri_inc_wo_plcy]
df_pri_inc_pr['pvals'] = \
    [x.t_test_pval if x.t_test else x.wrst_pval for x in pri_inc_wo_plcy]
df_pri_inc_pr['glo_m'] = \
    [x.glo_m_after[1] for x in pri_inc_wo_plcy]
df_pri_inc_pr['sum_mean_rests_wo_p'] = \
    [x.gdf_after.mean_rest.sum() for x in pri_inc_wo_plcy]
df_pri_inc_pr['avg_mean_rests_wo_p'] = \
    [x.gdf_after.mean_rest.dropna().mean() for x in pri_inc_wo_plcy]
df_pri_inc_pr['med_mean_rests_wo_p'] = \
    [x.gdf_after.mean_rest.dropna().median() for x in pri_inc_wo_plcy]

#%% price increasing test under condition posterior to the new policy
pri_inc_w_plcy = []
for price in tqdm(np.arange(6000, 10500, 500),
                  desc='Price increase test posterior to the new policy'):
    pri_inc_w_plcy.append(
        get_all_attr(6000, price, mode='after')
        )

#%% Get results from price increasing test posterior to the new policy
df_pri_inc_po = pd.DataFrame(index=np.arange(6000, 10500, 500))
df_pri_inc_po['stats_diff'] = \
    ['t' if x.t_test else 'w' for x in pri_inc_w_plcy]
df_pri_inc_po['statistic'] = \
    [x.t_test_s if x.t_test else x.wrst_s for x in pri_inc_w_plcy]
df_pri_inc_po['pvals'] = \
    [x.t_test_pval if x.t_test else x.wrst_pval for x in pri_inc_w_plcy]
df_pri_inc_po['glo_m'] = \
    [x.glo_m_after[1] for x in pri_inc_w_plcy]
df_pri_inc_po['sum_mean_rests_w_p'] = \
    [x.gdf_after.mean_rest.sum() for x in pri_inc_w_plcy]
df_pri_inc_po['avg_mean_rests_w_p'] = \
    [x.gdf_after.mean_rest.dropna().mean() for x in pri_inc_w_plcy]
df_pri_inc_po['med_mean_rests_w_p'] = \
    [x.gdf_after.mean_rest.dropna().median() for x in pri_inc_w_plcy]

#%% Plot - P values based on price increasing
fig_pri_inc_pvals, ax_pri_inc_pvals = plt.subplots(1, 1, figsize=(8, 6))

ax_pri_inc_pvals.plot(df_pri_inc_pr.index, df_pri_inc_pr.pvals, 'o-',
                      label='[illegible]', markersize=8)
ax_pri_inc_pvals.plot(df_pri_inc_po.index, df_pri_inc_po.pvals, 's-',
                      label='[illegible]', markersize=8)
ax_pri_inc_pvals.plot(df_pri_inc_po.index[2:], df_pri_inc_po.pvals[2:],
                      's', label='[illegible]', markersize=8)
ax_pri_inc_pvals.axhline(0.05, linestyle='--', color='r', linewidth=1.5)
ax_pri_inc_pvals.text(6000, 0.055, '0.05', fontsize=12)
ax_pri_inc_pvals.legend()

ax_pri_inc_pvals.set_xlabel('[illegible]')
ax_pri_inc_pvals.set_ylabel('P-value')

save_fig(dir_nm_save_fig, fig_pri_inc_pvals, title='[illegible] pvalues',
         close_fig=False)
```

```python
def load_map_rest_join(pri_thld=6000):
    """
    Parameters
    ----------
    pri_thld : TYPE, optional
        DESCRIPTION. The default is 6000.

    Returns
    -------
    gdf_emd_jj_frchs : GeoDataFrame
        GeoDataFrame based on eup_myeon_dong of JeonJu merged with child meal
        card franchise restaurants in JeonJu
    gdf_emd_jj : GeoDataFrame
        GeoDataFrame based on eup_myeon_dong of JeonJu merged with every
        restaurant in JeonJu
    """
    # Import packages
    import os
    import geopandas as gpd
    import pyproj

    # Load preprocessed restaurants data of Jeonju
    dir_org = os.getcwd()
    os.chdir('data')

    df_rest_menu = pd.read_csv('J_rnm.csv', encoding='utf-8', index_col=0)

    os.chdir(dir_org)

    # Load map shp data
    dir_org = os.getcwd()
    os.chdir('data/SHP파일모음/BUBJUNG')

    # gdf_emd = gpd.read_file('Z_SOP_BND_ADM_DONG_PG.shp', encoding='euc-kr')
    gdf_emd = gpd.read_file('BUBJUNG.shp', encoding='utf-8')
    gdf_emd = gdf_emd.rename(columns={'name':'ADM_DR_NM'})

    os.chdir(dir_org)

    # Load school data
    dir_org = os.getcwd()
    os.chdir('data/SHP파일모음/학교shp')

    gdf_sch = gpd.read_file('school.shp', encoding='euc-kr')

    os.chdir(dir_org)

    # Load Jeonju franchise restaurants data
    dir_org = os.getcwd()
    os.chdir('data/SHP파일모음/전주시_가맹점shp')

    df_rest_jj_frchs = pd.read_csv('jeonju_franchise.csv', encoding='euc-kr',
                                   index_col=1).iloc[:, 1:]

    os.chdir(dir_org)

    # Menu data preprocess by prices of menu
    df_rest_jj = df_rest_menu[df_rest_menu.price <= pri_thld]
    df_rest_jj = df_rest_jj.groupby('res_name').max().reset_index()
    df_rest_jj = df_rest_jj.rename(columns={'lat(y)':'위도',
                                            'lon(x)':'경도'})

    #% Join the price column of df_rest_jj to franchise based on str simility
    df_rest_jj_frchs[['price', 'menu', 'name']] = np.nan
```

```python
    #% Join the price column of df_rest_jj to franchise based on str simility
    df_rest_jj_frchs[['price', 'menu', 'name']] = np.nan

    for idx, row in df_rest_jj_frchs.iterrows():
        nm_org = df_rest_jj_frchs.at[idx, 'm_name']
        str_sims = df_rest_jj.res_name.apply(lambda x : str_sim(nm_org, x))

        best_sim = np.max(str_sims)
        best_sim_idx = np.argmax(str_sims)
        if best_sim > 0.8:
            df_rest_jj_frchs.loc[idx, 'name'] = \
                df_rest_jj.at[best_sim_idx, 'res_name']
            df_rest_jj_frchs.loc[idx, 'price'] = \
                df_rest_jj.at[best_sim_idx, 'price']
            df_rest_jj_frchs.loc[idx, 'menu'] = \
            df_rest_jj.at[best_sim_idx, 'menu_name']

    #% Assign price manually
    list_res_names = ['마트', 'CU', '롯데리아', '맘탄', '본도시락',
                      '본죽', '맘스', '뚜레쥬르', '롯데슈퍼', '이삭',
                      '봉구스밥버거', '신포우리만두']
    list_menu_names = ['라면', '도시락', '핫크리스피버거', '허브말수육', '흑미밥',
                       '본죽', '해산물팔보밥', '정통크라상', '라면',
                       '햄치즈포테이토', '햄치즈', '해물부추만두']
    list_prices = [6000, 6000, 6000, 5500, 5900, 6000, 6000, 2300, 6000, 4500,
                   5000, 6000]

    for res_name, menu, price in zip(list_res_names,
                                     list_menu_names,
                                     list_prices):
        df_rest_jj_frchs.loc[
            df_rest_jj_frchs.m_name.str.contains(res_name)==True,
            'price'] = price
        df_rest_jj_frchs.loc[
            df_rest_jj_frchs.m_name.str.contains(res_name)==True,
            'menu'] = menu

    df_rest_jj_frchs = df_rest_jj_frchs.dropna(subset=['price'])

    # Remove convenience store
    df_rest_jj_frchs = \
        df_rest_jj_frchs[~df_rest_jj_frchs.m_name.str.contains('CU')]

    # Select map data of Jeonju and copy it for franchies restaurants
    gdf_emd_jj = deepcopy(gdf_emd)
    gdf_emd_jj_frchs = deepcopy(gdf_emd)

    # Select school data of Jeonju
    gdf_sch_jj = \
        gdf_sch[gdf_sch['소재지도로'].str.contains('전라북도 전주시')]

    # Change epsg value of restaurants in Jeonju
    in_proj = pyproj.Proj(init='epsg:4326')
    out_proj = pyproj.Proj(init='epsg:5181')
    out_proj_str = 'epsg:5181'

    df_rest_jj['x'], df_rest_jj['y'] = pyproj.transform(
        in_proj, out_proj,
        df_rest_jj['경도'].tolist(),
        df_rest_jj['위도'].tolist())
```



```python
    # Change epsg value of franchise restaurants in Jeonju
    in_proj = pyproj.Proj(init='epsg:4326')
    out_proj = pyproj.Proj(init='epsg:5181')
    out_proj_str = 'epsg:5181'

    df_rest_jj_frchs['x'], df_rest_jj_frchs['y'] = pyproj.transform(
        in_proj, out_proj,
        df_rest_jj_frchs['경도'].tolist(),
        df_rest_jj_frchs['위도'].tolist())

    gdf_rest_jj_frchs = gpd.GeoDataFrame()

    gdf_rest_jj_frchs = gpd.GeoDataFrame(
            df_rest_jj_frchs.loc[:, :],
            geometry=gpd.points_from_xy(df_rest_jj_frchs.loc[:, 'x'],
                                        df_rest_jj_frchs.loc[:, 'y']),
            crs=out_proj_str
            )

    # Calculate summation of all restaurants near school
    gdf_sch_jj['num_rest'] = np.zeros(len(gdf_sch_jj.index))

    for idx, row in gdf_sch_jj.iterrows():
        sch_class = row.at['학교급구분']
        if sch_class == '초등학교':
            dist = 400
        elif sch_class == '중학교':
            dist = 700
        else:
            dist = 700

        geo_sch = row.at['geometry']
        num_rest = 0

        for geo_rest in gdf_rest_jj.geometry:
            if geo_sch.distance(geo_rest) <= dist:
                num_rest += 1

        gdf_sch_jj.at[idx, 'num_rest'] = num_rest

    # Calculate summation of franchise restaurants near school
    gdf_sch_jj_frchs = deepcopy(gdf_sch_jj)
    gdf_sch_jj_frchs['num_rest'] = np.zeros(len(gdf_sch_jj_frchs.index))

    for idx, row in gdf_sch_jj_frchs.iterrows():
        sch_class = row.at['학교급구분']
        if sch_class == '초등학교':
            dist = 400
        elif sch_class == '중학교':
            dist = 700
        else:
            dist = 700

        geo_sch = row.at['geometry']
        num_rest = 0

        for geo_rest in gdf_rest_jj_frchs.geometry:
            if geo_sch.distance(geo_rest) <= dist:
                num_rest += 1

        gdf_sch_jj_frchs.at[idx, 'num_rest'] = num_rest
```

```python
    # Calculate total restaurants near school in terms of district
    gdf_emd_jj['sum_rest'] = np.zeros(len(gdf_emd_jj.index))
    gdf_emd_jj['sum_school']= np.zeros(len(gdf_emd_jj.index))

    for idx, row in gdf_emd_jj.iterrows():
        geo_emd = row.geometry
        num_rest = 0
        num_sch = 0

        for _, row_s in gdf_sch_jj.iterrows():
            if geo_emd.intersects(row_s.geometry):
                gdf_emd_jj.at[idx, 'sum_school'] += 1
                gdf_emd_jj.at[idx, 'sum_rest'] += row_s.num_rest

    gdf_emd_jj['mean_rest'] = gdf_emd_jj.sum_rest/gdf_emd_jj.sum_school
    gdf_emd_jj.mean_rest = gdf_emd_jj.mean_rest.fillna(np.finfo(float).eps)

    # Calculate franchise restaurants near school in terms of district
    gdf_emd_jj_frchs['sum_rest'] = np.zeros(len(gdf_emd_jj_frchs.index))
    gdf_emd_jj_frchs['sum_school']= np.zeros(len(gdf_emd_jj_frchs.index))

    for idx, row in gdf_emd_jj_frchs.iterrows():
        geo_emd = row.geometry
        num_rest = 0
        num_sch = 0

        for _, row_s in gdf_sch_jj_frchs.iterrows():
            if geo_emd.intersects(row_s.geometry):
                gdf_emd_jj_frchs.at[idx, 'sum_school'] += 1
                gdf_emd_jj_frchs.at[idx, 'sum_rest'] += row_s.num_rest

    gdf_emd_jj_frchs['mean_rest'] = \
        gdf_emd_jj_frchs.sum_rest/gdf_emd_jj_frchs.sum_school
    gdf_emd_jj_frchs.mean_rest = \
        gdf_emd_jj_frchs.mean_rest.fillna(np.finfo(float).eps)

    return gdf_emd_jj_frchs, gdf_emd_jj
```

```
#%% Import module
from sklearn.preprocessing import scale
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy import stats
from sklearn.metrics import silhouette_score ,silhouette_samples
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
#%% pickle data Load
import pickle
from Spatial_Analysis import GdfCompare

with open('gdf_6000_won', 'rb') as f :
    a = pickle.load(f)
with open('gdf_7000_won', 'rb') as f :
    b = pickle.load(f)
with open('법정동_지도_정보_추가', 'rb') as f:
    c = pickle.load(f)

six_bf = a.gdf_before
six_af = a.gdf_after

six_bf_nc = six_bf[['sum_rest', 'sum_school', 'mean_rest']]
six_af_nc = six_af[['sum_rest', 'sum_school', 'mean_rest']]

merge = pd.merge(six_bf, c.loc[:,['ADM_DR_NM','공시지가','초등학생수','중학생수','고등학생수']], how ='inner', on= 'ADM_DR_NM')
merge_2 = pd.merge(six_af, c.loc[:,['ADM_DR_NM','공시지가','초등학생수','중학생수','고등학생수']], how ='inner', on= 'ADM_DR_NM')

six_bf_merge = merge[['sum_rest', 'sum_school','mean_rest','공시지가']]
six_af_merge = merge_2[['sum_rest', 'sum_school','mean_rest', '공시지가']]
```

```
#%% Add Official Price

sc = StandardScaler() #정확한 거리를 산정하기 위한 스케일링

sc.fit(six_bf_merge)
sc.fit(six_af_merge)

kmeans = KMeans(n_clusters=3, init = 'k-means++', max_iter= 300, random_state = 40).fit(sc.transform(six_bf_merge))
six_bf_merge['cluster'] = kmeans.labels_
kmeans_2 = KMeans(n_clusters=3, init = 'k-means++', max_iter= 300, random_state = 40).fit(sc.transform(six_af_merge))
six_af_merge['cluster'] = kmeans_2.labels_

six_bf_merge['target'] = six_bf.hotspot_class
six_merge_result = six_bf_merge.groupby(['target', 'cluster'])['sum_school'].count()
six_af_merge['target'] = six_af.hotspot_class
six_merge_result2 = six_af_merge.groupby(['target', 'cluster'])['sum_school'].count()


six_bf_score = silhouette_score(six_bf_merge, kmeans.labels_,metric='euclidean')
six_af_score = silhouette_score(six_af_merge, kmeans_2.labels_,metric='euclidean')

print(six_merge_result)
print(six_merge_result2)
print('6000 before silhouette score: %.3f' %six_bf_score)
print('6000 after silhouette score: %.3f' %six_af_score)
```

```
#%% Before 6000 Policy Visualization

pca = PCA(n_components=2) #2차원 평면에 시각화를 위해 차원축소
pca_transformed = pca.fit_transform(six_bf_merge[['sum_rest','sum_school','mean_rest','공시지가']])

six_bf_merge['pca_x'] = pca_transformed[:,0]
six_bf_merge['pca_y'] = pca_transformed[:,1]

marker0_ind = six_bf_merge[six_bf_merge['cluster'] == 0].index
marker1_ind = six_bf_merge[six_bf_merge['cluster'] == 1].index
marker2_ind = six_bf_merge[six_bf_merge['cluster'] == 2].index

plt.scatter(x = six_bf_merge.loc[marker0_ind, 'pca_x'], y = six_bf_merge.loc[marker0_ind, 'pca_y'], marker = 'o', label= 'Bad')
plt.scatter(x = six_bf_merge.loc[marker1_ind, 'pca_x'], y = six_bf_merge.loc[marker1_ind, 'pca_y'], marker = 's', label= 'Nomal')
plt.scatter(x = six_bf_merge.loc[marker2_ind, 'pca_x'], y = six_bf_merge.loc[marker2_ind, 'pca_y'], marker = '^', label= 'Good')

plt.legend()
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.show()

print('공시지가 cor :',stats.pointbiserialr(x = six_bf_merge['공시지가'], y = six_bf_merge['cluster']))
print('식당 총 수 cor :',stats.pointbiserialr(x = six_bf_merge['sum_rest'], y = six_bf_merge['cluster']))
print('식당수 평균 cor :',stats.pointbiserialr(x = six_bf_merge['mean_rest'], y = six_bf_merge['cluster']))
print('학교 총 수 cor :',stats.pointbiserialr(x = six_bf_merge['sum_school'], y = six_bf_merge['cluster']))
```



```
#%% After 6000 Policy Visualization

pca = PCA(n_components=2)
pca_transformed = pca.fit_transform(six_af_merge[['sum_rest', 'sum_school', 'mean_rest','공시지가']])

six_af_merge['pca_x'] = pca_transformed[:,0]
six_af_merge['pca_y'] = pca_transformed[:,1]

marker0_ind = six_af_merge[six_af_merge['cluster'] == 0].index
marker1_ind = six_af_merge[six_af_merge['cluster'] == 1].index
marker2_ind = six_af_merge[six_af_merge['cluster'] == 2].index

plt.scatter(x = six_af_merge.loc[marker0_ind, 'pca_x'], y = six_af_merge.loc[marker0_ind, 'pca_y'], marker = 'o', label= 'Bad')
plt.scatter(x = six_af_merge.loc[marker1_ind, 'pca_x'], y = six_af_merge.loc[marker1_ind, 'pca_y'], marker = 's', label= 'Nomal')
plt.scatter(x = six_af_merge.loc[marker2_ind, 'pca_x'], y = six_af_merge.loc[marker2_ind, 'pca_y'], marker = '^', label= 'Good')

plt.legend()
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.show()

print('공시지가 cor :',stats.pointbiserialr(x = six_af_merge['공시지가'], y = six_af_merge['cluster']))
print('식당 총 수 cor :',stats.pointbiserialr(x = six_af_merge['sum_rest'], y = six_af_merge['cluster']))
print('식당수 평균 cor :',stats.pointbiserialr(x = six_af_merge['mean_rest'], y = six_af_merge['cluster']))
print('학교 총 수 cor :',stats.pointbiserialr(x = six_af_merge['sum_school'], y = six_af_merge['cluster']))
```

```
#%% Silhouette Sample Distribution : Before 6000 Policy
f, axes = plt.subplots(1, 5, sharex=True, sharey=True)
f.set_size_inches(15, 3)
for i, ax in enumerate(axes):
    sil_samples = silhouette_samples(six_bf_merge, kmeans.labels_,metric='euclidean')
    sil_score = silhouette_score(six_bf_merge, kmeans.labels_,metric='euclidean')
    ax.plot(sorted(sil_samples), color="red",linestyle='dashed', linewidth=2)
    ax.set_title("silhouette_score: {}".format(round(sil_score, 2)))
```

```
#%% Silhouette Sample Distribution : After 6000 Policy
f, axes = plt.subplots(1, 5, sharex=True, sharey=True)
f.set_size_inches(15, 3)
for i, ax in enumerate(axes):
    sil_samples = silhouette_samples(six_af_merge, kmeans_2.labels_,metric='euclidean')
    sil_score = silhouette_score(six_af_merge, kmeans_2.labels_,metric='euclidean')
    ax.plot(sorted(sil_samples), color="red",linestyle='dashed', linewidth=2)
    ax.set_title("silhouette_score: {}".format(round(sil_score, 2)))
```

```
#%% Vulnerable Location List
six_bf_idx = six_bf_merge[six_bf_merge['cluster'] == 0 ].index
six_af_idx = six_af_merge[six_af_merge['cluster'] == 0 ].index

vul_location_bf = merge.loc[six_bf_idx, 'ADM_DR_NM']
vul_location_af = merge.loc[six_af_idx, 'ADM_DR_NM']

(len(vul_location_bf) - len(vul_location_af)) /len(vul_location_bf) * 100 #취약지역 증감률

six_bf_idx_2 = six_bf_merge[six_bf_merge['cluster'] == 2 ].index
six_af_idx_2 = six_af_merge[six_af_merge['cluster'] == 2 ].index

good_location_bf = merge.loc[six_bf_idx_2, 'ADM_DR_NM']
good_location_af = merge.loc[six_af_idx_2, 'ADM_DR_NM']

(len(good_location_af) - len(good_location_bf)) /len(good_location_bf) * 100 #좋은 지역 증감률
```