공공데이터를 활용한 서울특별시 관악구 그린허브 최적 입지 선정 - 딥러닝 및 공간분석 중심

목 차

1. 분석 개요

1.1 분석 배경 및 개요

1.1.1 서울시의 분리수거 정책현황

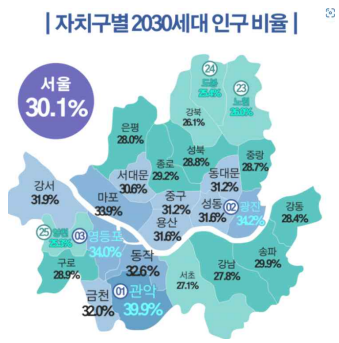
[그림 1-1] 2015 년 서울시 재활용정거장

[그림 1-2] 서울시 2021 년 오늘의 분리수거

□ 2015년 환경정책으로 주택가의 분리수거를 할 수 있도록 '재활용정거장'을 실행하고 현재까지 시행중인 분리시설

□ 2021년 '오늘의 분리수거' 투명 패트병과 캔을 지정된 분리설비에 배출하면 포인트로 환산해서 돌려주는 정책을 시험사업으로 시행하고 있음.

1.1.2 서울시 관악구 2030대 인구현황 및 1인가구 비율



[그림1-3] 서울시 구별 2030대 인구 비율

[그림 1-4] 서울 자치구별 1 인 가구 수

□ 2021년 20~30대 인구 비율이 39.9%로 가장 많은 지역으로 서울시 관악구가 1위로 나타남

□ 2018년 기준 1인 가구 수가 가장 많은 자치구는 112.733세대 관악구로 나타남

1.1.3 새로운 분류 수거함의 설치의 필요성과 현 정책의 문제점

□ 분리수거장이 잘 갖춰져 있는 아파트와 다르게 주택가/원룸촌은 쓰레기 배출 공간이 부족하거나 갖추어져 있지않음

□ 쓰레기가 몰려있는 장소에 무단투기가 빈번하게 발생 되고 있음

□ 관악구가 20~30대 1인 가구 비율이 가장 높으면서, 배달 주문량 또한 높 일회용품의 분리수거가 이루어지지 않은 상태에서 쓰레기 투기의 문제점이 많을 것으로 예상하기에 관악구를 중심으로 문제해결 방법이 필요함

□ 현재 시행중인 재활용정거장은 설치가 되어있는 구역에 비해 설치/지리적요인 등으로 미설치 구역에서 문제점이 발생됨

□ 개인주택 앞에 설치된 경우가 많아 공공적이 아닌 개인 분류함으로 오인하는 문제점이 있음

1.2 분석 목적 및 방향

□ 서울시 관악구 주택가/원룸촌의 무단투기 및 분리되지 않은 쓰레기 주요 발생 지점 분석을 통한 자동 분류 쓰레기통 최적의 입지 선정

□ 수거 시 차량의 효율적인 동선 제시하여 수거업체가 최적경로로 수거할 수 있도록 해야 함

□ 쓰레기 분류시설의 설치 공간 활용에 기존에 미사용 중인 시설물의 위치를 체크 그 지점을 사용할 수 있도록 방향 제시

1.3 분석 결과 활용 방안

□ 1인 가구들의 무단투기의 발생지역과 발생량 데이터가 없어 제고

□ 1인 가구들의 쓰레기 배출장소의 지점 분석을 통해 1인 가구 밀집도, 가로수, 의류 수거함, CCTV 등을 접근성 변수로 설정하게 됨

□ 취약 지역의 접근성 변수(Point)를 이용하여 QGIS를 통한 최적 입지를 선정하여 취약지역 100위를 만듬

□ 선정된 최적 입지를 TSP 알고리즘을 활용한 최단 경로로 수거를 할 수 있도록 루트를 구하고 결괏값을 Qgis로 시각화하였음

□ 시각화된 위치 데이터를 기반으로 쓰레기 분류 수거함 시범사업구역을 제안함

□ 딥러닝을 활용한 자동 분류 쓰레기통의 프로토타입 모델 설계

□ 우수 모델을 타 지역구에 적용하는 경우와 기존 정책 중 '오늘의 분리수거'와의 ver2 등으로 정책에 보완할 수 있는 미래 방향성을 제시할 수 있음

2. 분석 데이터

2.1 분석 데이터 목록

[표 1] 분석 데이터 목록

구분	분석 데이터	생성주기	데이터 소스
환경 변수	서울시 관악구 행정동별 1인 가구 데이터	매년	통계지리정보서비스(SGIS) (https://sgis.kostat.go.kr/view/index?param=0)
접근성 변수	서울시 관악구 한전 가로등 위치 정보	매년	공공데이터 포털 (https://www.data.go.kr/)
	서울시 관악구 다목적 CCTV 위치 정보	매년	관악 열린 데이터 광장 (https://data.gwanak.go.kr)
	서울시 관악구 제로페이 가맹점 현황	매년	공공데이터 포털 (https://www.data.go.kr/)
	서울시 관악구 셀프빨래방 위치 정보	매년	네이버 지도 (https://map.naver.com/)
	서울시 관악구 의류수거함 위치 정보	매년	공공데이터 포털 (https://www.data.go.kr/)
이미지	쓰레기 이미지 정보	실시간	캐글(Kaggle) (https://www.kaggle.com/)

2.2 데이터 상세 설명

[표 2] 분석 데이터 상세 설명

구분	분석 데이터	형식	생성주기
공공 데이터	서울시 관악구 행정동별 1인 가구 데이터	PDF, CSV	매년
	서울시 관악구 한전 가로등 위치 정보	CSV	매년
	서울시 관악구 다목적 CCTV 위치 정보	CSV	매년
	서울시 관악구 제로페이 가맹점 현황	CSV	매년
	서울시 관악구 셀프빨래방 위치 정보	CSV	매년
	서울시 관악구 의류수거함 위치 정보	CSV	매년

2.2.1 서울시 관악구 행정동별 1인 가구 데이터 [.PDF]

□ 통계지리정보서비스(SGIS)에서 제공하는 자료로 서울시 관악구의 행정동별 1인 가구 주소, 순위, 단위(명), 비율 등을 포함

항목	순위	단위 (가구)	비율(%)
합계		129,233.00	100
서울특별시 관악구 청룡동	1	13,435	10.40
서울특별시 관악구 신림동	2	11,942	9.24
서울특별시 관악구 대학동	3	9,718	7.52
서울특별시 관악구 행운동	4	9,674	7.49
서울특별시 관악구 서림동	5	7,735	5.99
서울특별시 관악구 서원동	6	7,607	5.89
서울특별시 관악구 신사동	7	7,282	5.63
서울특별시 관악구 인헌동	8	6,776	5.24
서울특별시 관악구 낙성대동	9	6,341	4.91
서울특별시 관악구 은천동	10	5,723	4.43
서울특별시 관악구 조원동	11	5,581	4.32
서울특별시 관악구 중앙동	12	5,565	4.31
서울특별시 관악구 신원동	13	5,505	4.26
서울특별시 관악구 보라매동	14	5,182	4.01
서울특별시 관악구 미성동	15	4,186	3.24
서울특별시 관악구 난곡동	16	4,155	3.22
서울특별시 관악구 남현동	17	3,855	2.98
서울특별시 관악구 삼성동	18	3,132	2.42
서울특별시 관악구 성원동	19	3,082	2.38
서울특별시 관악구 청림동	20	1,403	1.09
서울특별시 관악구 난향동	21	1,354	1.05

[그림 2-1] 서울시 관악구 행정동별 1인 가구 데이터 상세

□ [그림 2-2]와 같이 서울시 관악구 행정동별 1인 가구 상세 주소를 파악

항목	집계구번호	순위	단위 (명)	비율(%)
합계			294.00	100
(난곡동_1)난우2길 5 부근	1121081010023	1	40	13.61
(난곡동_2)법원단지30길 17 부근	1121081010021	2	30	10.20
(난곡동_3)법원단지20길 28 부근	1121081010013	3	25	8.50
(난곡동_4)법원단지7길 51 부근	1121081010009	4	21	7.14
(난곡동_5)난곡로35가길 19 부근	1121081020004	5	19	6.46
(난곡동_6)법원단지30길 33 부근	1121081010039	6	14	4.76
(난곡동_7)법원단지22길 6 부근	1121081010004	7	12	4.08
(난곡동_8)난곡로26길 95-20 부근	1121081010006	8	11	3.74
(난곡동_9)난곡로24마길 11-2 부근	1121081010010	9	11	3.74
(난곡동_10)법원단지25길 7 부근	1121081010036	10	11	3.74
(난곡동_11)난곡로26마길 31 부근	1121081010038	11	11	3.74
(난곡동_12)난곡로34길 68 부근	1121081010008	12	10	3.40
(난곡동_13)난곡로34길 40-5 부근	1121081010037	13	9	3.06
(난곡동_14)법원단지9길 38 부근	1121081010034	14	8	2.72
(난곡동_15)난곡로35나길 10 부근	1121081020007	15	8	2.72
(난곡동_16)법원단지26길 6 부근	1121081010012	16	7	2.38
(난곡동_17)법원단지7길 70 부근	1121081010014	17	7	2.38
(난곡동_18)난곡로30길 27 부근	1121081010017	18	7	2.38
(난곡동_19)법원단지3길 151-7 부근	1121081010035	19	7	2.38
(난곡동_20)난곡로34길 2 부근	1121081010024	20	6	2.04
(난곡동_21)법원단지14길 2 부근	1121081010016	21	5	1.70
(난곡동_22)난곡로26라길 5 부근	1121081010022	22	5	1.70
(난곡동_23)법원단지16길 19 부근	1121081010029	23	5	1.70
(난곡동_24)	1121081020005	24	5	1.70

[그림 2-2] 서울시 관악구 난곡동 1인 가구 데이터 상세

2.2.2 서울시 관악구 행정동별 1인 가구 데이터 [.CSV]

□ [그림 2-2]에서 파악한 서울시 관악구 행정동별 1인 가구의 주소, 순위, 단위(명), 비율 등을 Excel에 정리하고 CSV 파일로 생성

서울특별시 관악구 주소	행정동	순위	단위(명)	비율
서울특별시 관악구 남부순환로230길 25	낙성대동	1	30	12.66
서울특별시 관악구 남부순환로230길 63	낙성대동	2	27	11.39
서울특별시 관악구 관악로6길 65	낙성대동	3	26	10.97
서울특별시 관악구 봉천로 575	낙성대동	4	19	8.02
서울특별시 관악구 낙성대로 23	낙성대동	5	16	6.75
서울특별시 관악구 관악로12길 83	낙성대동	6	15	6.33
서울특별시 관악구 관악로 110-1	낙성대동	7	14	5.91
서울특별시 관악구 봉천로 568	낙성대동	8	13	5.49
서울특별시 관악구 남부순환로230길 45	낙성대동	9	13	5.49
서울특별시 관악구 낙성대역8길 50-4	낙성대동	10	11	4.64
서울특별시 관악구 남부순환로228길 81	낙성대동	11	11	4.64
서울특별시 관악구 낙성대로 77	낙성대동	12	10	4.22
서울특별시 관악구 관악로10길 37	낙성대동	13	9	3.8
서울특별시 관악구 낙성대로4길 34	낙성대동	14	7	2.95
서울특별시 관악구 남부순환로224길 31	낙성대동	15	6	2.53
서울특별시 관악구 관악로14길 27	낙성대동	16	5	2.11

[그림 2-3] 서울시 관악구 행정동별 1인 가구 데이터 상세

2.2.3 서울시 관악구 한전 가로등 위치 정보 [.CSV]

□ 공공데이터포털에서 제공하는 자료로 관악구 내 행정동별 CCTV의 소재지도로명주소, 지번주소, 위도, 경도 데이터를 포함

보안등위치설치개수	소재지도로명주소	소재지지번주소	위도	경도
낙성대001	1 서울특별시 관악구	서울특별시 관악구 봉천동 857-4	37.48058	126.9533
낙성대002	1 서울특별시 관악구 관악로 168	서울특별시 관악구 봉천동 856-1	37.48044	126.9532
낙성대003	1 서울특별시 관악구 남부순환로224길 25	서울특별시 관악구 봉천동 854-3	37.47988	126.9533
낙성대004	1 서울특별시 관악구 남부순환로224길 29	서울특별시 관악구 봉천동 854-15	37.47952	126.9532
낙성대005	1 서울특별시 관악구 관악로 158	서울특별시 관악구 봉천동 856-6	37.47923	126.953
낙성대008	1 서울특별시 관악구 남부순환로226길 40	서울특별시 관악구 봉천동 1598-7	37.4789	126.9533
낙성대009	1 서울특별시 관악구 관악로 154-11	서울특별시 관악구 봉천동 855-6	37.47894	126.9529
낙성대010	1 서울특별시 관악구 관악로 154	서울특별시 관악구 봉천동 1598-1	37.47885	126.9527
낙성대011	1 서울특별시 관악구 관악로12길 3-12	서울특별시 관악구 봉천동 1598-27	37.47863	126.9528
낙성대012	1 서울특별시 관악구 관악로 148	서울특별시 관악구 봉천동 1598-20	37.47837	126.9527
낙성대013	1 서울특별시 관악구 관악로12길 5	서울특별시 관악구 봉천동 1598-23	37.4782	126.9527
낙성대017	1 서울특별시 관악구 관악로12길 25-6	서울특별시 관악구 봉천동 1604-39	37.47849	126.9541
낙성대018	1 서울특별시 관악구 관악로12길 33	서울특별시 관악구 봉천동 1604-48	37.47839	126.9544
낙성대019	1 서울특별시 관악구 관악로12길 35	서울특별시 관악구 봉천동 1604-52	37.47831	126.9547
낙성대021	1 서울특별시 관악구 관악로14길 45	서울특별시 관악구 봉천동 1611-21	37.47878	126.955
낙성대022	1 서울특별시 관악구	서울특별시 관악구 봉천동 1603-21	37.47889	126.9546
낙성대023	1 서울특별시 관악구 관악로14길 38-10	서울특별시 관악구 봉천동 1604-17	37.47852	126.9544
낙성대025	1 서울특별시 관악구 관악로14길 31	서울특별시 관악구 봉천동 1603-15	37.479	126.9543
낙성대027	1 서울특별시 관악구 관악로14길 25-4	서울특별시 관악구 봉천동 1603-12	37.47918	126.9541
낙성대032	1 서울특별시 관악구 관악로14길 15	서울특별시 관악구 봉천동 1599-4	37.47919	126.9534

[그림 2-4] 서울시 관악구 행정동별 한전 가로등 위치 데이터 상세

2.2.4 서울시 관악구 다목적 CCTV 위치 정보 [.CSV]

□ 관악 열린 데이터 광장에서 제공하는 자료로 관악구 내 행정동별 관리기관명, 소재지번주소, 위도, 경도 등을 포함

번호	관리기관명	지도로명주	소재지지번주소	설치목적구분	카메라대수	카메라화소수	촬영방면정보	보관일수	설치연월	기관전화번	WGS84위도	WGS84경도
1	서울특별시	서울특별시		다목적	1			30		02-879-58	37.48438674445	126.9484749451
2	서울특별시	서울특별시		다목적	1			30		02-879-58	37.48430155792	126.9481865673
3	서울특별시		서울특별시 관악구 대학동 210	다목적	1			30		02-879-58	37.46831654332	126.9456033806
4	서울특별시		서울특별시 관악구 대학동 210	다목적	1			30		02-879-58	37.46824624200	126.9454760493
5	서울특별시		서울특별시 관악구 대학동 210	다목적	1			30		02-879-58	37.46185796308	126.9479584352
6	서울특별시		서울특별시 관악구 대학동 210	다목적	1			30		02-879-58	37.46090920040	126.9481995745
7	서울특별시		서울특별시 관악구 대학동 210	다목적	2			30		02-879-58	37.45796925886	126.9464132911
8	서울특별시		서울특별시 관악구 대학동 210	다목적	1			30		02-879-58	37.45560655638	126.9453963105
9	서울특별시		서울특별시 관악구 청룡동 산 1	다목적	1			30		02-879-58	37.47969721372	126.9372032218
10	서울특별시		서울특별시 관악구 청룡동 산 1	다목적	1			30		02-879-58	37.47983360479	126.9363697153
11	서울특별시		서울특별시 관악구 청룡동 산 1	다목적	1			30		02-879-58	37.48048488826	126.9368633397
12	서울특별시	서울특별시		다목적	1			30		02-879-58	37.47369949282	126.9326582577
13	서울특별시	서울특별시		다목적	1			30		02-879-58	37.47177180804	126.9790536140
14	서울특별시	서울특별시		다목적	1			30		02-879-58	37.48472218765	126.9448622521
15	서울특별시	서울특별시		다목적	1			30		02-879-58	37.49274898075	126.9350922362
16	서울특별시	서울특별시		다목적	1			30		02-879-58	37.47331235370	126.9236363182
17	서울특별시		서울특별시 관악구 난곡동 산 9	다목적	1			30		02-879-58	37.46673249610	126.9237165185
18	서울특별시		서울특별시 관악구 남현동 산 6	다목적	1			30		02-879-58	37.46985455870	126.9778823884
19	서울특별시		서울특별시 관악구 대학동 산 5	다목적	1			30		02-879-58	37.45641695330	126.9461288131
20	서울특별시		서울특별시 관악구 미성동 산 1	다목적	1			30		02-879-58	37.46924977574	126.9126727204
21	서울특별시		서울특별시 관악구 미성동 산 1	다목적	1			30		02-879-58	37.48097851092	126.9164949791
22	서울특별시		서울특별시 관악구 미성동 산 1	다목적	1			30		02-879-58	37.48021998240	126.9158268603
23	서울특별시		서울특별시 관악구 보라매동 신	다목적	1			30		02-879-58	37.49183061814	126.9362613431

[그림 2-5] 서울시 관악구 행정동별 다목적 CCTV 위치 데이터 상세

2.2.5 서울시 관악구 제로페이 가맹점 현황 정보 [.CSV]

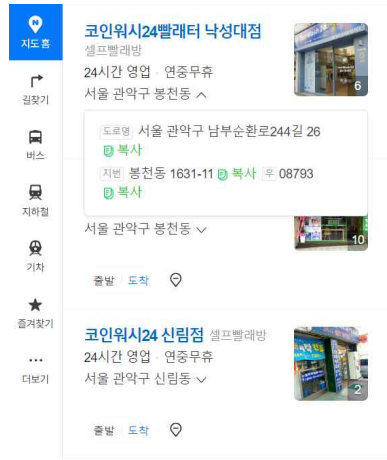
□ 공공데이터포털에서 제공하는 자료로 관악구 내 행정동별 제로페이 사용 가맹점명, 업종, 도로명 주소 등을 포함

가맹점명	업종	행정동	도로명 주소	데이터기준일자
제일미용학	미용학원	조원동	서울특별시	########
정인이네미	일반미용업	서림동	서울특별시	########
개라다이스	애완용 동물	보라매동	서울특별시	########
쉐르빌	여관업	청룡동	서울특별시	########
신데렐라너	미용	행운동	서울특별시	########
명품	화장품, 비	보라매동	서울특별시	########
뮤직박스노	래연습장	신림동	서울특별시	########
부타앤가츠	한식	난곡동	서울시 관	########
삼월부티크	전자상거래	신원동	서울특별시	########
숙고당 관(	한식 일반	청림동	서울특별시	########
프롬팔일팔	커피 전문점	청룡동	서울특별시	########
대감숯불갈	한식	미성동	서울특별시	########
미니스톱 신	편의점	난곡동	서울특별시	########
치요남치킨	치킨	청룡동	서울특별시	########
맛조아한고	과자점업(주	신원동	서울특별시	########
배영만 당(	한식	인헌동	서울특별시	########
막퍼주는집	농수산물	인헌동	서울특별시	########
주식회사 그	휴게음식,	청룡동	서울특별시	########
쁘띠몽드서	실내놀이터	행운동	서울특별시	########
고래가주 식	이자카야	인헌동	서울특별시	########
고기반햄버	한식 일반	신림동	서울특별시	########
프로샵(PR(	전자상거래	청룡동	서울특별시	########
우리호프	호프	서원동	서울시 관	########

[그림 2-6] 서울시 관악구 제로페이 가맹점 현황 데이터 상세

2.2.6 서울시 관악구 셀프빨래방 위치 정보 [.CSV]

□ 네이버 지도에서 제공하는 자료로 관악구 내 행정동별 도로명 주소를 통합 Excel 파일에 정리 후 CSV 파일로 생성



[그림 2-7] 서울시 관악구 셀프빨래방 위치 데이터 상세(네이버 지도)

관악구	서울특별시 관악구 남부순환로244길 26	셀프빨래방
	서울특별시 관악구 호암로 579 1층	셀프빨래방
	서울특별시 관악구 인헌길 71	셀프빨래방
	서울특별시 관악구 관천로12길 30 지1층	셀프빨래방
	서울특별시 관악구 남부순환로187길 30 1층	셀프빨래방
	서울특별시 관악구 난곡로31길 8	셀프빨래방
	서울특별시 관악구 호암로22길 52	셀프빨래방
	서울특별시 관악구 남부순환로151길 54 소노하	셀프빨래방
	서울특별시 관악구 신림로67길 14	셀프빨래방
	서울특별시 관악구 관악로13길 20 1층	셀프빨래방
	서울특별시 관악구 조원로 97-1	셀프빨래방
	서울특별시 관악구 남부순환로161가길 12	셀프빨래방
	서울특별시 관악구 관천로10길 4	셀프빨래방
	서울특별시 관악구 남부순환로181길 30 1층	셀프빨래방
	서울특별시 관악구 남부순환로245길 9 1층	셀프빨래방
	서울특별시 관악구 조원로18길 13	셀프빨래방
	서울특별시 관악구 남부순환로 1790	셀프빨래방
	서울특별시 관악구 문성로 228 1층	셀프빨래방
	서울특별시 관악구 인헌길 125	셀프빨래방
	서울특별시 관악구 신림동7길 8 1층	셀프빨래방
	서울특별시 관악구 청룡10길 18 1층	셀프빨래방
	서울특별시 관악구 봉천로45길 13	셀프빨래방
	서울특별시 관악구 남부순환로 1957 제이원 B0	셀프빨래방
	서울특별시 관악구 남부순환로190길 10	셀프빨래방

[그림 2-8] 서울시 관악구 셀프빨래방 위치 데이터 상세(Excel)

2.2.7 서울시 관악구 의류수거함 위치 정보 [.CSV]

□ 공공데이터포털에서 제공하는 자료로 관악구 내 행정동별 의류수거함 주소를 포함

의류수거함	위치
낙성대동-1	서울특별시 관악구 관악로10길 15
낙성대동-2	서울특별시 관악구 관악로10길 6
낙성대동-3	서울특별시 관악구 관악로10길 77
낙성대동-4	서울특별시 관악구 관악로10길 87
낙성대동-5	서울특별시 관악구 관악로12길 16
낙성대동-6	서울특별시 관악구 관악로12길 55
낙성대동-7	서울특별시 관악구 관악로14가길 28
낙성대동-8	서울특별시 관악구 관악로14길 38-9
낙성대동-9	서울특별시 관악구 관악로6길 104-7
낙성대동-10	서울특별시 관악구 관악로6길 15
낙성대동-11	서울특별시 관악구 관악로6길 15
낙성대동-12	서울특별시 관악구 관악로6길 21
낙성대동-13	서울특별시 관악구 관악로6길 36
낙성대동-14	서울특별시 관악구 관악로6길 49
낙성대동-15	서울특별시 관악구 관악로6길 50
낙성대동-16	서울특별시 관악구 관악로6길 77
낙성대동-17	서울특별시 관악구 관악로8길 32
낙성대동-18	서울특별시 관악구 관악로8길 43
낙성대동-19	서울특별시 관악구 낙성대로3길 11
낙성대동-20	서울특별시 관악구 낙성대로3길 21-4

[그림 2-9] 서울시 관악구 의류수거함 위치 데이터 상세(Excel)

2.3 데이터 정제 방안

2.3.1 서울시 관악구 행정동별 1인 가구 데이터

2.3.1.1 원본 데이터

[표 3] 2.3.1.1 원본 데이터

이름	설명	데이터 출처
서울시 관악구 행정동별 1인 가구 데이터	서울시 관악구 행정동에 따른 1인 가구 단위(명) 및 위치 포함	통계지리정보서비스(SGIS)

2.3.1.2 정제 과정

□ 서울시 관악구 행정동별 1인 가구 CSV 파일을 파이썬으로 불러온 후, geopy.geocoders 패키지를 활용

□ geopy.geocoders 코드 내에 1인 가구의 각 주소지를 입력하여 위도와 경도값으로 변환

□ 변환값을 서울특별시 관악구 행정동별 1인 가구 CSV 파일에 위도와 경도라는 새로운 컬럼으로 추가 및 저장

서울특별시	행정동	순위	단위(명)	비율	위도	경도
서울특별시	낙성대동	1	30	12.66	37.47773	126.9552
서울특별시	낙성대동	2	27	11.39	37.47773	126.9552
서울특별시	낙성대동	3	26	10.97	37.47633	126.9534
서울특별시	낙성대동	4	19	8.02	37.47739	126.9615
서울특별시	낙성대동	5	16	6.75	37.47663	126.9599
서울특별시	낙성대동	6	15	6.33	37.47773	126.9552
서울특별시	낙성대동	7	14	5.91	37.47513	126.9531
서울특별시	낙성대동	8	13	5.49	37.47737	126.9605
서울특별시	낙성대동	9	13	5.49	37.47773	126.9552
서울특별시	낙성대동	10	11	4.64	37.47438	126.9613
서울특별시	낙성대동	11	11	4.64	37.47788	126.9546
서울특별시	낙성대동	12	10	4.22	37.47185	126.9585
서울특별시	낙성대동	13	9	3.8	37.4767	126.9548
서울특별시	낙성대동	14	7	2.95	37.47668	126.9578
서울특별시	낙성대동	15	6	2.53	37.47996	126.9532
서울특별시	낙성대동	16	5	2.11	37.47858	126.9556
서울특별시	난곡동	1	40	13.61	37.47333	126.9198
서울특별시	난곡동	2	30	10.2	37.47522	126.9247
서울특별시	난곡동	3	25	8.5	37.47376	126.9233
서울특별시	난곡동	4	21	7.14	37.47438	126.9202

[그림 2-10] 서울시 관악구 행정동별 1인 가구 정제 데이터 상세

2.3.2 서울시 관악구 한전 가로등 위치 데이터

2.3.2.1 원본 데이터

[표 4] 2.3.2.1 원본 데이터

이름	설명	데이터 출처
서울시 관악구 한전 가로등 위치 데이터	서울시 관악구 행정동에 따른 한전 가로등 위치 포함	공공데이터포털

2.3.2.2 정제 과정

□ 서울시 관악구 한전 가로등 위치 CSV 파일 자체에 위도와 경도값을 가지고 있기 때문에, 정제 과정 없이 바로 활용 가능

보안등위치	설치개수	소재지도로명주소	소재지지번주소	위도	경도
낙성대001	1	서울특별시 관악구	서울특별시 관악구 봉천동 857-4	37.48058	126.9533
낙성대002	1	서울특별시 관악구 관악로 168	서울특별시 관악구 봉천동 856-1	37.48044	126.9532
낙성대003	1	서울특별시 관악구 남부순환로224길 25	서울특별시 관악구 봉천동 854-3	37.47988	126.9533
낙성대004	1	서울특별시 관악구 남부순환로224길 29	서울특별시 관악구 봉천동 854-15	37.47952	126.9532
낙성대005	1	서울특별시 관악구 관악로 158	서울특별시 관악구 봉천동 856-6	37.47923	126.953
낙성대008	1	서울특별시 관악구 남부순환로226길 40	서울특별시 관악구 봉천동 1598-7	37.4789	126.9533
낙성대009	1	서울특별시 관악구 관악로 154-11	서울특별시 관악구 봉천동 855-6	37.47894	126.9529
낙성대010	1	서울특별시 관악구 관악로 154	서울특별시 관악구 봉천동 1598-1	37.47885	126.9527
낙성대011	1	서울특별시 관악구 관악로12길 3-12	서울특별시 관악구 봉천동 1598-27	37.47863	126.9528
낙성대012	1	서울특별시 관악구 관악로 148	서울특별시 관악구 봉천동 1598-20	37.47837	126.9527
낙성대013	1	서울특별시 관악구 관악로12길 5	서울특별시 관악구 봉천동 1598-23	37.4782	126.9527
낙성대017	1	서울특별시 관악구 관악로12길 25-6	서울특별시 관악구 봉천동 1604-39	37.47849	126.9541
낙성대018	1	서울특별시 관악구 관악로12길 33	서울특별시 관악구 봉천동 1604-48	37.47839	126.9544
낙성대019	1	서울특별시 관악구 관악로12길 35	서울특별시 관악구 봉천동 1604-52	37.47831	126.9547
낙성대021	1	서울특별시 관악구 관악로14길 45	서울특별시 관악구 봉천동 1611-21	37.47878	126.955
낙성대022	1	서울특별시 관악구	서울특별시 관악구 봉천동 1603-21	37.47889	126.9546
낙성대023	1	서울특별시 관악구 관악로14길 38-10	서울특별시 관악구 봉천동 1604-17	37.47852	126.9544
낙성대025	1	서울특별시 관악구 관악로14길 31	서울특별시 관악구 봉천동 1603-15	37.479	126.9543
낙성대027	1	서울특별시 관악구 관악로14길 25-4	서울특별시 관악구 봉천동 1603-12	37.47918	126.9541
낙성대032	1	서울특별시 관악구 관악로14길 15	서울특별시 관악구 봉천동 1599-4	37.47919	126.9534

[그림 2-11] 서울시 관악구 한전 가로등 위치 데이터 상세

2.3.3 서울시 관악구 다목적 CCTV 위치 정보

2.3.3.1 원본 데이터

[표 5] 2.3.3.1 원본 데이터

이름	설명	데이터 출처
서울시 관악구 다목적 CCTV 위치 데이터	서울시 관악구 행정동에 따른 다목적 CCTV 위치 포함	관악 열린 데이터 광장

2.3.3.2 정제 과정

□ 서울시 관악구 다목적 CCTV 위치 CSV 파일 자체에 위도와 경도값을 가지고 있기 때문에, 정제 과정 없이 바로 활용 가능

번호	관리기관명	소재지도로명주소	소재지지번주소	설치목적구분	카메라대수	카메라화소수	촬영방면정보	보관일수	설치연월	관리기관전화번호	WGS84위도	WGS84경도
1	서울특별시	서울특별시		다목적	1			30		02-879-58	37.48438674445	126.9484749451
2	서울특별시	서울특별시		다목적	1			30		02-879-58	37.48430155792	126.9481865673
3	서울특별시	서울특별시 관악구 대학동 210		다목적	1			30		02-879-58	37.46831654332	126.9456033806
4	서울특별시	서울특별시 관악구 대학동 210		다목적	1			30		02-879-58	37.46824624200	126.9454760493
5	서울특별시	서울특별시 관악구 대학동 210		다목적	1			30		02-879-58	37.46185796308	126.9479584352
6	서울특별시	서울특별시 관악구 대학동 210		다목적	1			30		02-879-58	37.46090920040	126.9481995745
7	서울특별시	서울특별시 관악구 대학동 210		다목적	2			30		02-879-58	37.45796925886	126.9464132911
8	서울특별시	서울특별시 관악구 대학동 210		다목적	1			30		02-879-58	37.45560655638	126.9453963105
9	서울특별시	서울특별시 관악구 청룡동 산 1		다목적	1			30		02-879-58	37.47969721372	126.9372032218
10	서울특별시	서울특별시 관악구 청룡동 산 1		다목적	1			30		02-879-58	37.47983360479	126.9363697153
11	서울특별시	서울특별시 관악구 청룡동 산 1		다목적	1			30		02-879-58	37.48048488826	126.9368633397
12	서울특별시	서울특별시		다목적	1			30		02-879-58	37.47369949282	126.9326582577
13	서울특별시	서울특별시		다목적	1			30		02-879-58	37.47177180804	126.9790536140
14	서울특별시	서울특별시		다목적	1			30		02-879-58	37.48472218765	126.9448622521
15	서울특별시	서울특별시		다목적	1			30		02-879-58	37.49274898075	126.9350922362
16	서울특별시	서울특별시		다목적	1			30		02-879-58	37.47331235370	126.9236363182
17	서울특별시	서울특별시 관악구 난곡동 산 9		다목적	1			30		02-879-58	37.46673249610	126.9237165185
18	서울특별시	서울특별시 관악구 남현동 산 6		다목적	1			30		02-879-58	37.46985455870	126.9778823884
19	서울특별시	서울특별시 관악구 대학동 산 5		다목적	1			30		02-879-58	37.45641695330	126.9461288131
20	서울특별시	서울특별시 관악구 미성동 산 1		다목적	1			30		02-879-58	37.46924977574	126.9126727204
21	서울특별시	서울특별시 관악구 미성동 산 1		다목적	1			30		02-879-58	37.48097851092	126.9164949791
22	서울특별시	서울특별시 관악구 미성동 산 1		다목적	1			30		02-879-58	37.48021998240	126.9158268603
23	서울특별시	서울특별시 관악구 보라매동 신		다목적	1			30		02-879-58	37.49183061814	126.9362613431

[그림 2-12] 서울시 관악구 다목적 CCTV 위치 데이터 상세

2.3.4 서울시 관악구 제로페이 가맹점 현황

2.3.4.1 원본 데이터

[표 6] 2.3.4.1 원본 데이터

이름	설명	데이터 출처
서울시 관악구 제로페이 가맹점 현황 데이터	서울시 관악구 행정동별 제로페이 가맹점 현황 및 위치 포함	공공데이터포털

2.3.4.2 정제 과정

□ 서울시 관악구 제로페이 가맹점 현황 CSV 파일 내에 있는 업종 컬럼에서 편의점 카테고리만 따로 추출하여 저장

□ 이전 과정을 실시한 서울시 관악구 제로페이 가맹점 현황 CSV 파일을 파이썬으로 불러온 후, geopy.geocoders 패키지를 활용

□ geopy.geocoders 코드 내에 편의점의 각 주소지를 입력하여 위도와 경도값 으로 변환

□ 변환값을 서울특별시 관악구 제로페이 가맹점 현황 CSV 파일에 위도와 경도라는 새로운 컬럼으로 추가 및 파일을 다른 이름으로 CSV 저장

업종	행정동	도로명 주소	데이터기준	위도	경도
편의점	난곡동	서울특별시	########	37.47474	126.9199
편의점	서원동	서울특별시	########	37.48277	126.9309
편의점	남현동	서울특별시	########	37.47586	126.9796
편의점	남현동	서울특별시	########	37.47349	126.9759
편의점	신원동	서울특별시	########	37.48261	126.9226
편의점	은천동	서울특별시	########	37.48779	126.9464
편의점	남현동	서울특별시	########	37.47586	126.9796
편의점	인헌동	서울특별시	########	37.47076	126.9651
편의점	신림동	서울특별시	########	37.47761	126.9338
편의점	서림동	서울특별시	########	37.47598	126.9384
편의점	청룡동	서울특별시	########	37.47624	126.9417
편의점	미성동	서울특별시	########	37.4762	126.9206
편의점	인헌동	서울특별시	########	37.47268	126.9682
편의점	신원동	서울특별시	########	37.48088	126.9278
편의점	청룡동	서울특별시	########	37.47932	126.9458
편의점	삼성동	서울특별시	########	37.47761	126.9338
편의점	은천동	서울특별시	########	37.48158	126.9456
편의점	신사동	서울특별시	########	37.48446	126.9201
편의점	신림동	서울특별시	########	37.48633	126.9285
편의점	행운동	서울특별시	########	37.48105	126.9604
편의점	난곡동	서울특별시	########	37.46819	126.9215
편의점	미성동	서울특별시	########	37.47322	126.9189
편의점	인헌동	서울특별시	########	37.47588	126.961

[그림 2-13] 서울시 관악구 편의점 위치 데이터 상세

2.3.5 서울시 관악구 셀프빨래방 위치 데이터

2.3.5.1 원본 데이터

[표 7] 2.3.5.1 원본 데이터

이름	설명	데이터 출처
서울시 관악구 셀프빨래방 위치 데이터	서울시 관악구 행정동별 셀프빨래방 위치 포함	네이버 지도

2.3.5.2 정제 과정

□ 서울시 관악구 셀프빨래방 위치 CSV 파일을 파이썬으로 불러온 후, geopy.geocoders 패키지를 활용

□ geopy.geocoders 코드 내에 관악구 셀프빨래방의 각 도로명주소를 입력하여 위도와 경도값으로 변환

□ 변환값을 서울특별시 관악구 셀프빨래방 위치 CSV 파일에 위도와 경도라는 새로운 컬럼으로 추가 및 저장

도로명 주소	업종	위도	경도
서울특별시 관악구 남부순환로244길 26	셀프빨래방	37.4765	126.9632
서울특별시 관악구 호암로 579	셀프빨래방	37.45967	126.9227
서울특별시 관악구 인헌길 71	셀프빨래방	37.47237	126.9669
서울특별시 관악구 관천로12길 30	셀프빨래방	37.48576	126.927
서울특별시 관악구 남부순환로187길 30	셀프빨래방	37.48536	126.9335
서울특별시 관악구 난곡로31길 8	셀프빨래방	37.4708	126.9181
서울특별시 관악구 호암로22길 52	셀프빨래방	37.46875	126.9352
서울특별시 관악구 남부순환로151길 54	셀프빨래방	37.48361	126.9161
서울특별시 관악구 신림로67길 14	셀프빨래방	37.48593	126.9282
서울특별시 관악구 관악로13길 20	셀프빨래방	37.47949	126.951
서울특별시 관악구 조원로 97-1	셀프빨래방	37.48374	126.9094
서울특별시 관악구 남부순환로161가길 12	셀프빨래방	37.48482	126.9166
서울특별시 관악구 관천로10길 4	셀프빨래방	37.48525	126.9273
서울특별시 관악구 남부순환로181길 30	셀프빨래방	37.48548	126.9308
서울특별시 관악구 남부순환로245길 9	셀프빨래방	37.47737	126.9653
서울특별시 관악구 조원로18길 13	셀프빨래방	37.48353	126.9124
서울특별시 관악구 남부순환로 1790	셀프빨래방	37.48158	126.9456
서울특별시 관악구 문성로 228	셀프빨래방	37.47771	126.9154
서울특별시 관악구 인헌길 125	셀프빨래방	37.47024	126.9682

[그림 2-14] 서울시 관악구 셀프빨래방 위치 데이터 상세

2.3.6 서울시 관악구 의류수거함 위치 데이터

2.3.6.1 원본 데이터

[표 8] 2.3.6.1 원본 데이터

이름	설명	데이터 출처
서울시 관악구 의류수거함 위치 데이터	서울시 관악구 행정동별 의류수거함 위치 포함	공공데이터포털

2.3.6.2 정제 과정

□ 서울시 관악구 의류수거함 위치 CSV 파일을 파이썬으로 불러온 후, geopy.geocoders 패키지를 활용

□ geopy.geocoders 코드 내에 관악구 의류수거함의 각 주소를 입력하여 위도와 경도값으로 변환

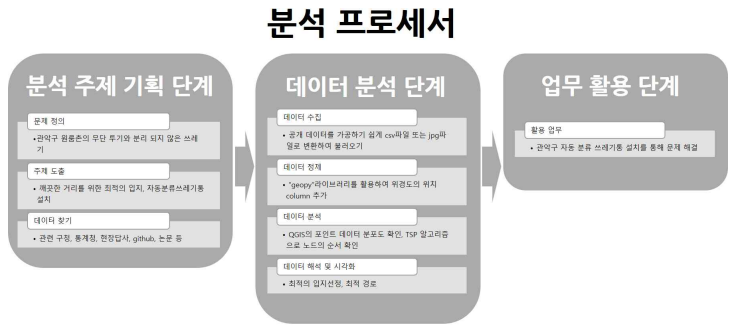
□ 변환값을 서울특별시 관악구 의류수거함 위치 CSV 파일에 위도와 경도라는 새로운 컬럼으로 추가 및 저장

의류수거함	주소	위도	경도
낙성대동-1	서울특별시 관악구 관악로10길 15	37.4767	126.9548
낙성대동-2	서울특별시 관악구 관악로10길 6	37.4767	126.9548
낙성대동-3	서울특별시 관악구 관악로10길 77	37.47653	126.9569
낙성대동-4	서울특별시 관악구 관악로10길 87	37.47658	126.9575
낙성대동-5	서울특별시 관악구 관악로12길 16	37.47683	126.9559
낙성대동-6	서울특별시 관악구 관악로12길 55	37.47773	126.9552
낙성대동-7	서울특별시 관악구 관악로14가길 28	37.47653	126.9555
낙성대동-8	서울특별시 관악구 관악로14길 38-9	37.47858	126.9556
낙성대동-9	서울특별시 관악구 관악로6길 104-7	37.47633	126.9534
낙성대동-10	서울특별시 관악구 관악로6길 15	37.47633	126.9534
낙성대동-11	서울특별시 관악구 관악로6길 15	37.47633	126.9534
낙성대동-12	서울특별시 관악구 관악로6길 21	37.47633	126.9534
낙성대동-13	서울특별시 관악구 관악로6길 36	37.47633	126.9534
낙성대동-14	서울특별시 관악구 관악로6길 49	37.47633	126.9534
낙성대동-15	서울특별시 관악구 관악로6길 50	37.47627	126.9535
낙성대동-16	서울특별시 관악구 관악로6길 77	37.47633	126.9534
낙성대동-17	서울특별시 관악구 관악로8길 32	37.47567	126.9544
낙성대동-18	서울특별시 관악구 관악로8길 43	37.47567	126.9544
낙성대동-19	서울특별시 관악구 낙성대로3길 11	37.47492	126.9606
낙성대동-20	서울특별시 관악구 낙성대로3길 21-4	37.47387	126.9605

[그림 2-15] 서울시 관악구 의류수거함 위치 데이터 상세

3. 분석 프로세스

3.1 분석 프로세스



[그림 3-1] 분석 프로세서 상세

3.2 분석 내용 및 방법

3.2.1 분석 주제 기획 단계

□ 서울시에서 1인 가구가 가장 많은 관악구에서 상습/다발적인 무단투기의 문제점과 분리수거가 이루어지지 않는 문제점을 해결하고자 최적의 입지를 선정하여 자동 분류쓰레기통을 설치하고자 한다. 해당 1인 가구가 밀집되어있는 지역의 파악과 쓰레기 투기가 이루어지는 곳을 파악하기 위해 관악구청의 데이터와 통계청 등의 데이터들을 찾아본다.

3.2.2 데이터 분석 단계

□ 해당 데이터들을 찾고 최적 입지 선정에 필요한 다목적 cctv 위치, 편의점 위치, 셀프 빨래방 위치, 의료수거함 위치, 한전 가로등 위치 데이터가 필요로 하고, 이 데이터들을 QGIS로 취합한 입지 선정 데이터들을 활용하여 TSP 알고리즘과 realcentroid와 QNEAT3 라이브러리를 통해 최적 경로를 시각화한다.

[표 8] 3.2.2 사용 알고리즘 및 플러그인

이름	설명
Geopy - python	해당 데이터의 도로명 주소를 QGIS 상에 표시하기 위해 위도와 경도를 뽑아내는 라이브러리
TSP - python	모든 쓰레기통을 한 번씩 돌아서 오는 최단 경로를 구하는 알고리즘을 통해 노드의 순서를 파악
realcentriod - plug in	QGIS의 격자 중심점 생성
QNEAT3 - plug in	지도 상 노드 간의 최단 거리 계산

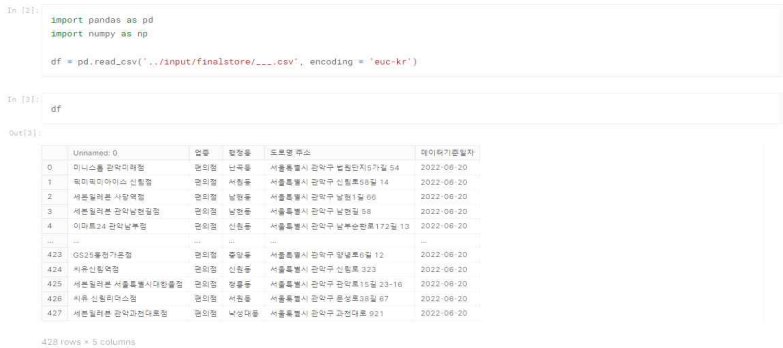
3.2.3 업무 활용 단계

□ 관악구 원룸촌의 협소한 공간을 최대로 활용하면서 개발한 자동 분류쓰레기통을 배치하고, 쓰레기 수거 차량이 들어올 수 있는 최단 경로까지 제시하면서 쾌적한 길거리를 만들어 준다.

3.2.4 분석 과정

□ Q-GIS 적용에 필요한 데이터 전처리, 변수들 간 상관관계 확인을 위한 RandomForest 분석, Q-GIS를 통한 최적입지선정, tsp 알고리즘을 통한 최적 경로 설정 및 시각화의 과정을 설명한다.

3.2.4.1 데이터 전처리

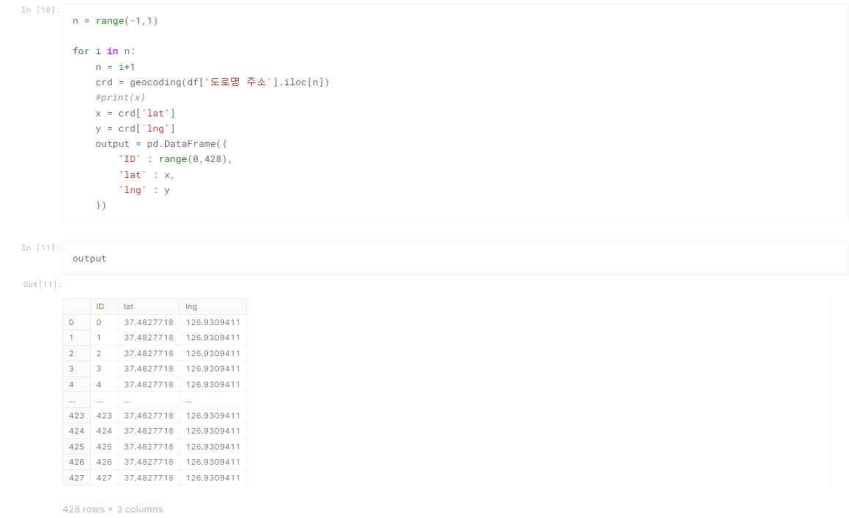


```
In [2]:
import pandas as pd
import numpy as np

df = pd.read_csv('../input/finalstore/___.csv', encoding = 'euc-kr')

In [3]:
df

Out[3]:
```

	Unnamed: 0	업종	행정동	도로명 주소	데이터기준일자
0	미니스톱 관악미래점	편의점	난곡동	서울특별시 관악구 법원단지5가길 54	2022-06-20
1	씨미씨미아이스 신림점	편의점	서림동	서울특별시 관악구 신림로58길 14	2022-06-20
2	세븐일레븐 사당역점	편의점	남현동	서울특별시 관악구 남현1길 66	2022-06-20
3	세븐일레븐 관악남현점	편의점	남현동	서울특별시 관악구 남현길 58	2022-06-20
4	이마트24 관악남부점	편의점	신림동	서울특별시 관악구 남부순환로172길 13	2022-06-20
...	...	...	...	...	...
423	GS25동원가온점	편의점	중앙동	서울특별시 관악구 양녕로6길 12	2022-06-20
424	차유신림역점	편의점	신림동	서울특별시 관악구 신림로 323	2022-06-20
425	세븐일레븐 서울특별시대한출점	편의점	청룡동	서울특별시 관악구 관악로15길 23-16	2022-06-20
426	차유 신림리더스점	편의점	서림동	서울특별시 관악구 문성로38길 67	2022-06-20
427	세븐일레븐 관악과천대로점	편의점	낙성대동	서울특별시 관악구 과천대로 921	2022-06-20

428 rows × 5 columns

[그림 3-2] 서울시 관악구 편의점 위치 데이터 전처리(1)



```
In [10]:
n = range(-1,1)

for i in n:
    n = i+1
    crd = geocoding(df['도로명 주소'].iloc[n])
    #print(x)
    x = crd['lat']
    y = crd['lng']
    output = pd.DataFrame({
        'ID' : range(0,428),
        'lat' : x,
        'lng' : y
    })

In [11]:
output

Out[11]:
```

	ID	lat	lng
0	0	37.4827718	126.9309411
1	1	37.4827718	126.9309411
2	2	37.4827718	126.9309411
3	3	37.4827718	126.9309411
4	4	37.4827718	126.9309411
...	...	...	...
423	423	37.4827718	126.9309411
424	424	37.4827718	126.9309411
425	425	37.4827718	126.9309411
426	426	37.4827718	126.9309411
427	427	37.4827718	126.9309411

428 rows × 3 columns

[그림 3-3] 서울시 관악구 편의점 위치 데이터 전처리(2)

□ [그림 3-2, 3-3]을 통해 서울시 관악구 편의점 위치 데이터에 있는 도로명 주소를 통해 위도, 경도값을 구하는 데이터 전처리 과정을 볼 수 있음

□ 서울시 관악구 1인 가구, 셀프빨래방, 의류수거함 위치 데이터도 같은 전처리 과정을 통해 위도, 경도값을 구할 예정

3.2.4.2 RandomForestRegressor 회귀 분석

```
!pip install xgboost
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: xgboost in c:₩users₩user₩appdata₩roaming₩python₩python39₩site-packages (1.6.1)
Requirement already satisfied: scipy in c:₩programdata₩anaconda3₩lib₩site-packages (from xgboost) (1.7.3)
Requirement already satisfied: numpy in c:₩programdata₩anaconda3₩lib₩site-packages (from xgboost) (1.21.5)
```

```
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.metrics import r2_score, mean_squared_error
```

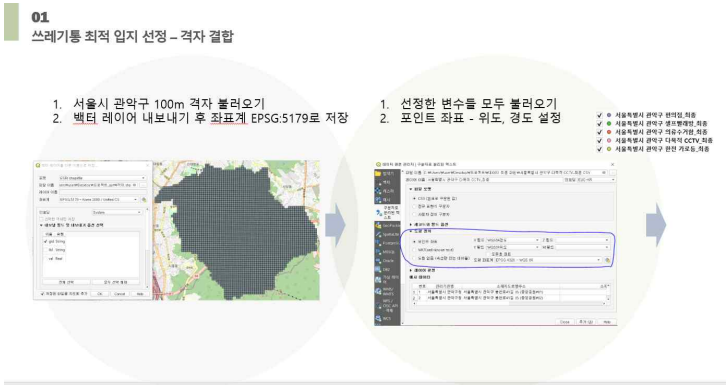
```
model = RandomForestRegressor()
RandomForest = model.fit(X_train, y_train)
output = model.predict(X_train)
print( '결정계수 : ' , model.score(X_train,y_train))
output
```

```
결정계수 :  0.8458468842158592

array([211.3 , 123.64, 269.68, 421.73, 227.53, 326.5 , 408.66, 383.08,
       673.12, 302.16, 254.36, 252.1 , 394.41, 372.55, 293.53, 251.72,
       251.8 , 290.01,  95.65, 246.26])
```

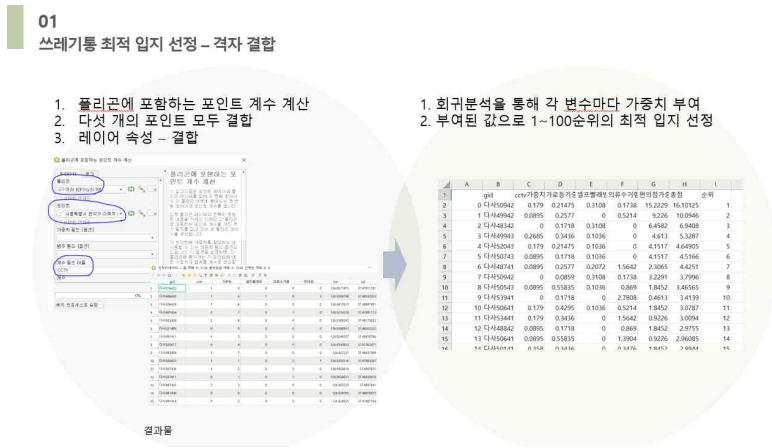
[그림 3-4] RandomForestRegressor 회귀 분석 코드

□ LinearRegression(선형회귀) 방식으로 하면, 결정계수가 0.429로 유의미한 모델이라고 단정짓기에 힘듬. 따라서 분석 모델을 바꾸거나 데이터를 더 세분화 시켜야하므로, [그림 2-18]에 있는 RandomForestRegressor 회귀 분석 모델을 사용해야함.

3.2.4.3 QGIS를 통한 최적입지 선정



[그림 3-5] QGIS 최적입지 선정 과정 中 격자 결합(1)



[그림 3-6] QGIS 최적입지 선정 과정 中 격자 결합(2)

[그림 3-7] QGIS 최적입지 선정 과정 中 격자 결합(3)

□ QGIS에서 서울시 관악구 레이어를 불러오고 관악구 격자를 기준으로, 위도와 경도값을 가진 cctv, 셀프빨래방, 의류수거함, 편의점 CSV 레이어를 불러옴

□ 다섯 개의 포인트를 모두 격자 형태로 결합하고 [그림 3-4]에 있는 회귀 분석 모델을 통해 나온 가중치를 각 변수에 부여한 후 최종적으로 부여된 값 기준으로 1~100 순위의 최적 입지 선정

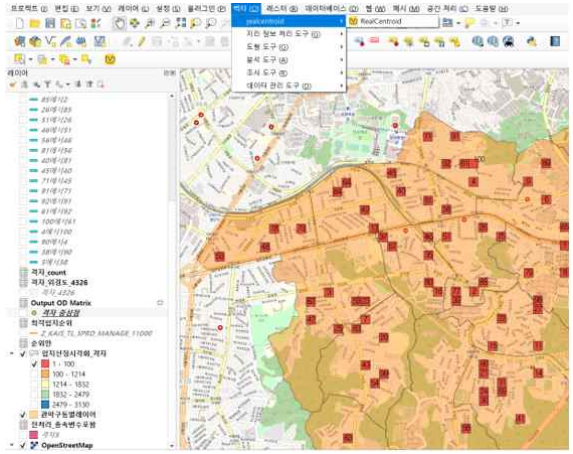
3.2.4.4 TSP 알고리즘

```
#TSP 알고리즘을 이용한 최적 경로 탐색
def main():
    data = create_data_model()
    manager = pywrapcp.RoutingIndexManager(len(data['locations']),
                                           data['num_vehicles'], data['depot'])
    routing = pywrapcp.RoutingModel(manager)
    distance_matrix = compute_euclidean_distance_matrix(data['locations'])

    def distance_callback(from_index, to_index):
        from_node = manager.IndexToNode(from_index)
        to_node = manager.IndexToNode(to_index)
        return distance_matrix[from_node][to_node]

    transit_callback_index = routing.RegisterTransitCallback(distance_callback)
    routing.SetArcCostEvaluatorOfAllVehicles(transit_callback_index)

    search_parameters = pywrapcp.DefaultRoutingSearchParameters()
    search_parameters.first_solution_strategy = (
        routing_enums_pb2.FirstSolutionStrategy.PATH_CHEAPEST_ARC)

    solution = routing.SolveWithParameters(search_parameters)

    if solution:
        print_solution(manager, routing, solution)
```

[그림 3-8] TSP 알고리즘 코드

□ TSP 알고리즘을 통해서 쓰레기 최적 수거 경로를 탐색

3.2.4.5 QGIS를 통한 최적 경로 시각화



[그림 3-9] QGIS RealCentroid 플러그인 설치

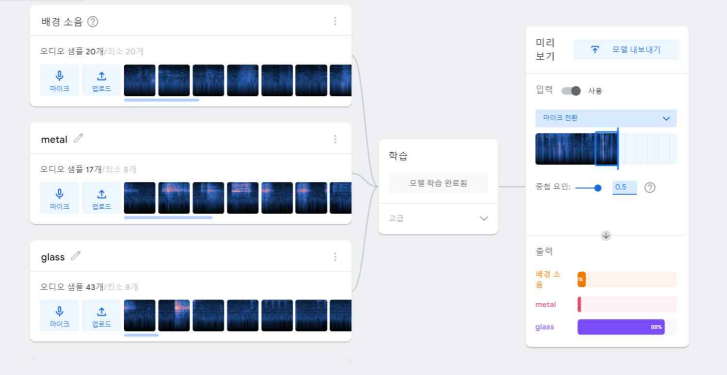
[그림 3-10] QGIS QNEAT3 플러그인 설치

□ QGIS에서 최적경로를 구할 때 활용되는 RealCentroid 플러그인 설치, RealCentroid 플러그인을 통해 격자 중심점 생성

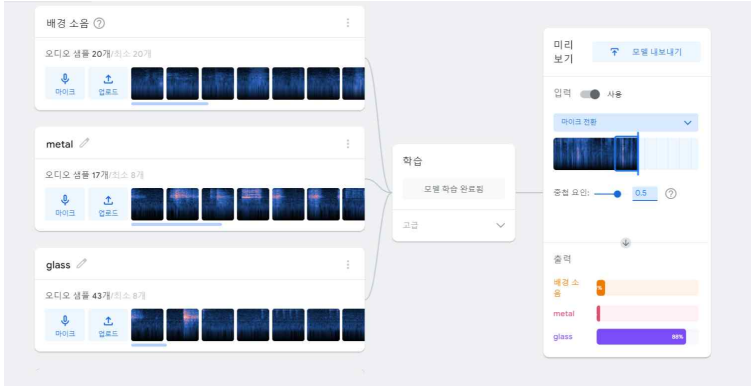
□ QGIS에서 최적경로를 구할 때 활용되는 QNEAT3 플러그인 설치, QNEAT3 플러그인을 통해 점과 점(point to point) 사이의 최적경로 생성

□ 생성된 최적경로 레이어를 성현동- 보라매동- 신림동 - 서원동- 신원동 - 미성동순으로 진행한 후 전체적으로 최종 수거 최적경로를 확인

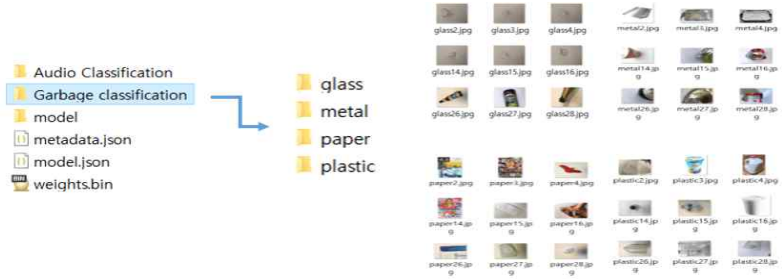
3.2.4.6 딥러닝을 활용한 자동 분류 쓰레기통



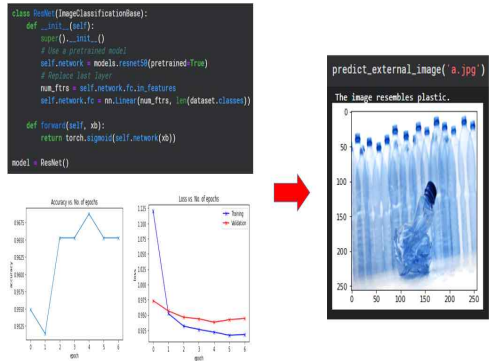
[그림 3-11] 소리 인식 딥러닝 훈련



[그림 3-12] 이미지 인식 딥러닝 훈련



[그림 3-13] 쓰레기 분류 이미지 데이터셋



[그림 3-14] 쓰레기 분류 딥러닝 모델

□ [그림 3-13]에 있는 이미지와 소리 데이터셋을 딥러닝을 적용하여 [그림 3-11, 그림 3-12]처럼 소리와 이미지 데이터를 각각 학습시킴

□ 최종적으로 [그림 3-14]에 있는 쓰레기 분류 딥러닝 모델을 활용하여 쓰레기를 종류에 따라 자동 분류하게 함

4. 분석결과

4.1 데이터 회귀분석

4.1.1 데이터의 회귀분석

☐ LinearRegression의 회귀분석보다 RandomForestRegressor로 분석하였을시 0.81로 높은 정확도가 나타났다.

☐ Permutation Importance 패키지를 이용해 변수별 중요도를 구함

☐ 변수별 중요도를 가중치로 사용하였을 경우 각각 편의점0.4916, 의류수거함 0.1614, 가로등 0.1582, 셀프빨래방 0.0999, CCTV 0.0888으로 값을 나타났으며 가장 높은 가중치로는 편의점으로 선정됨

```
from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor()
RandomForest = model.fit(X_train, y_train)
output = model.predict(X_train)
print( '결정계수 : ' , model.score(X_train,y_train))
```

```
결정계수 :  0.8121920853815363
```

```
# CCTV 개수, 의류수거함 개수, 편의점, 셀프빨래방 개수, 가로등 개수 순 변수 중요도
print(RandomForest.feature_importances_)

import eli5
from eli5.sklearn import PermutationImportance
eli5.show_weights(RandomForest, feature_names = X_train.columns.tolist())
```

```
[0.08877173 0.15822768 0.09994252 0.16142555 0.49163253]
```

Weight	Feature

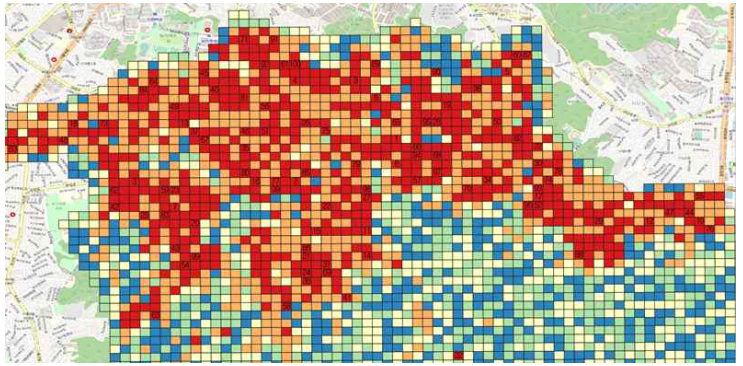
[그림 4-1] RandomForestRegressor 코드와 결과값

4.2 Exploratory Data Analysis

4.2.1 데이터의 공간 분포 시각화

☐ QGIS에서 붉은색으로 진하게 표시되있는 것을 입지 순위 1위~100위로 알 수 있도록 표시되었고 최고의 입지 순위로는 신림동이 선정됨

☐ 붉은색에서 초록색 파란색으로 색이 바뀌는 곳은 총점에서 순위가 100위권 아래로 밀집도가 낮은 지역을 나타나도록 표시하여 밀집도의 차이를 구분



[그림 4-2] 최적의 입지 순위 100위 선정 시각화

4.3 쓰레기 수거 최단 경로 선정

4.3.1 TSP 알고리즘 최단 경로

□ 전체 100개의 그린 허브 중 행정동별로 가장 많은 그린허브를 포함한 신봉그린 업체의 최단 경로로 선정

□ TSP 알고리즘을 사용하여 그린 허브 별 최단 경로를 구하였고 아래와 같다.

```
#TSP 알고리즘을 이용한 최적 경로 탐색
def main():
    data = create_data_model()
    manager = pywrapcp.RoutingIndexManager(len(data['locations']),
                                           data['num_vehicles'], data['depot'])
    routing = pywrapcp.RoutingModel(manager)
    distance_matrix = compute_euclidean_distance_matrix(data['locations'])

    def distance_callback(from_index, to_index):
        from_node = manager.IndexToNode(from_index)
        to_node = manager.IndexToNode(to_index)
        return distance_matrix[from_node][to_node]

    transit_callback_index = routing.RegisterTransitCallback(distance_callback)
    routing.SetArcCostEvaluatorOfAllVehicles(transit_callback_index)

    search_parameters = pywrapcp.DefaultRoutingSearchParameters()
    search_parameters.first_solution_strategy = (
        routing_enums_pb2.FirstSolutionStrategy.PATH_CHEAPEST_ARC)

    solution = routing.SolveWithParameters(search_parameters)

    if solution:
        print_solution(manager, routing, solution)


if __name__ == '__main__':
    main()
```

[그림 4-3] TSP 알고리즘을 통한 신봉그린 업체 최적경로

```
Objective: 10172
신봉그린 최적 경로:
 0 -> 24 -> 22 -> 4 -> 3 -> 23 -> 5 -> 16 -> 1 -> 10 -> 15 -> 20 -> 13 -> 17 -> 21 -> 14 -> 28 -> 26 -> 25 -> 29 -> 2 -> 6 -> 11 -
> 12 -> 9 -> 8 -> 7 -> 30 -> 27 -> 19 -> 18 -> 0
```


4.3.2 QGIS 최적경로 시각화

□ TSP 알고리즘으로 구한 신봉그린의 최적경로를 활용하여 QGIS를 사용 실제 내비게이션과 같은 경로를 시각적 효과를 확인

[그림 4-4] TSP 알고리즘을 통한 신봉그린 업체 최적경로

4.4 딥러닝 이미지분류

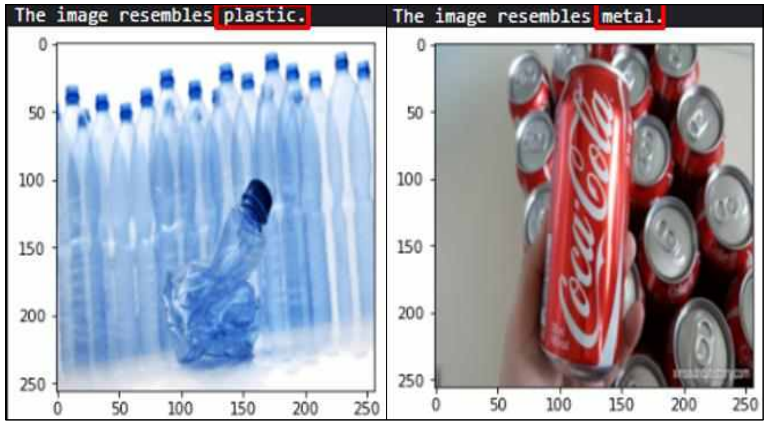
4.4.1 웹페이지 인식 분류

□ 최종적으로 분류되는 것을 아두이노와 라즈베리파이로 실질적으로 구현되는 모습은 구현하기 힘들기 때문에 웹페이지로 인식이 가능하다는 것을 간접적으로 확인할 수 있다.

[그림 4-5] 웹페이지 인식 분류

4.4.2 이미지 자동분류

□ 각각의 플라스틱과 메탈 글라스의 쓰레기를 이미지를 통해 인식하는 것을 확인



5. 활용 방안

5.1 문제점 개선 방안

5.1.1 주택 밀집 지역의 쓰레기 분류시설 문제 해결

□ 분리수거장이 잘 갖춰져 있는 아파트와 다르게 주택가/원룸촌은 쓰레기 배출 공간이 협소하거나 시설이 갖추어져 있지 않아 분류시설을 만들어 해결방안 제시

□ 주택 밀집 지역에는 분류시설이 미비해서 매년 쓰레기 문제가 발생되고 있다. 이러한 문제들을 해결하기 위한 새로운 환경정책이 필요함

□ 서울시의 주택 밀집지역과 1인가구가 가장 많은 지역을 선정하여 취약지역으로 선정한다.

□ 현재 실행중인 정책으로 '쓰레기 정류장'이 있지만 소외되는 구역이 발견되어 해당 구역에 대한 정책 수립 시 결정 지표로 사용하여 설치

5.1.2 주민들의 인식개선

□ 쓰레기 분류 시설이 있는 아파트와 다르게 주택가 쓰레기 분류에 대한 주민들의 인식개선이 필요함

□ 새로운 분리수거시설을 만들어 홍보하고 사용함에 따라 분리수거의 올바른 방법을, 재활용을 촉진하는 방향으로 주민들의 인식을 바꿔줄수 있도록 함

□ 현제 실행중인 환경정책과 더불어 홍보하고 실천하도록 하며 최종적으로는 분류시설이 없어 분리수거를 못한다는 문제점을 해결하고 시민의식 개선에 이바지함

5.2 업무 활용 방안

5.2.1 취약지역을 우선적으로 보완하는 방안 제시

□ 분석 데이터를 바탕으로 취약지역으로 선정된 서울시 관악구 지역(신림동, 청룡동 등)과 쓰레기 분류시설의 사용실태를 긴밀히 점검해야 함을 주장

□ 취약지역에는 절대적인 쓰레기 분류시설 수가 부족하므로 무단투기가 많이 발생되는 지역의 행정동에서도 20-30대 1인 가구가 몰려있는 지역에 분류시설을 우선적으로 설치하고 설치가 필요한 지역이지만 공간이 협소하여 용이하지 않은 구역은 의류 수거함이나 공중전화 등 공간을 지자체와 협조를 구하여 설치할 수 있도록 방안을 제시하여야 함

5.2.2 공공 재활용 기반시설 현대화 일환

□ 서울시, 경기도, 부산시 등 이미 적용되어 실행되고 있는 사례와 분석 결과를 기반으로 기반시설을 현대화하는 방안을 검토

5.2.3 시범사업지역 선정에 활용

□ 모든 지역에 분류시설 확대가 어려울 시, 도출한 취약지역을 중심으로 시범사업 지역을 선정하는데 활용

6. 참고자료(Reference)

6.1 연구 자료

- 박수민「1인가구 밀집지역의 상업업종 입지특성」
- 구동진 and 장준영. (2021). 오픈소스 기반 안면마스크 착용 모니터링 시스템 설계 및 구현. 한국인터넷방송통신학회 논문지, 21(4), 89-96.
- 김동훈, 서길원, 최현, 김현「IoT 센서 기반의 스마트 쓰레기통 개발」
- 김채현, 양라영, 이주현, 장희진, 하수빈, 김웅섭「최적의 생활 폐기물 수거 경로 탐색과 지역별 폐기물 예측에 대한 연구」
- 강민욱, 이상돈 1인 가구 증가에 따른 일회용 플라스틱 배출 실태 분석
- NICHOLAS CHIENG ANAK SALLANG, MOHAMMAD TARIQUL ISLAM, MOHAMMAD SHAHIDUL ISLAM, HASLINA ARSHAD 「A CNN-Based Smart Waste Management System Using TensorFlow Lite and LoRa-GPS Shield in Internet of Things Environment」
- 김태국「사물인터넷 기반의 스마트 휴지통」
- 장준범, 김동진, 박산희, 조다훈「영상 처리 기술을 이용한 재활용 분리기」
- 여상삼, 이상빈, 박진솔「자동 분리수거가 가능한 쓰레기통」

Kim, T.-K. (2020) "IoT(Internet of Things)-based Smart Trash Can," *Journal of The Korea Internet of Things Society*. 사물인터넷학회, 6(1), pp. 17–22. doi: 10.20465/KIOTS.2020.6.1.017.

6.2 문헌 자료

- 박상진, 최기혁, 김민석, 정의태「효율적인 분리수거를 위한 IoT 기반 스마트 리사이클러」
- 이은영, 이태희, 임재경, 임재현, 장윤하, 장혜리「북아현동 쓰레기 무단투기 해결 시스템 개발」
- 양현주「자취촌은 분리수거 무풍지대?…분리수거시설 자체가 없는곳 많아」디지털뉴스국
- 조민정「"구역질 나 못살아"…폭염 속 무단투기 쓰레기 악취 '속수무책'」이데일리
- 이시라「허수아비로 전락한 쓰레기 단속용 CCTV」경북매일
- 이원영「주택가 쓰레기 전쟁…양심까지도 무단투기」
- 송인호「쓰레기와의 전쟁 치르는 수원시, “분리배출 안하면 수거 거부” 초강수…왜?」SBS
- 정단비「1인 가구 많은 행정도 배달 수요 많아..30대 인구 비율 많을수록 인당 누적 이용 금액, 건수 비례」KBS강원
- 한희조「대학가 원룸촌 쓰레기 분리 배출 ‘엉망’」
- 임민영「[GIS] 격자 형식을 점 형식으로 변환하기(.SHP)」
- 김덕배「[BOJ 2098] 외판원 순회 (Python)」

7. 부록

7.1 기획서 현황조사

7.1.1 기사(1)

□ “자취촌은 분리수거 무풍지대?... 분리수거시설 자체가 없는 곳 많아..”

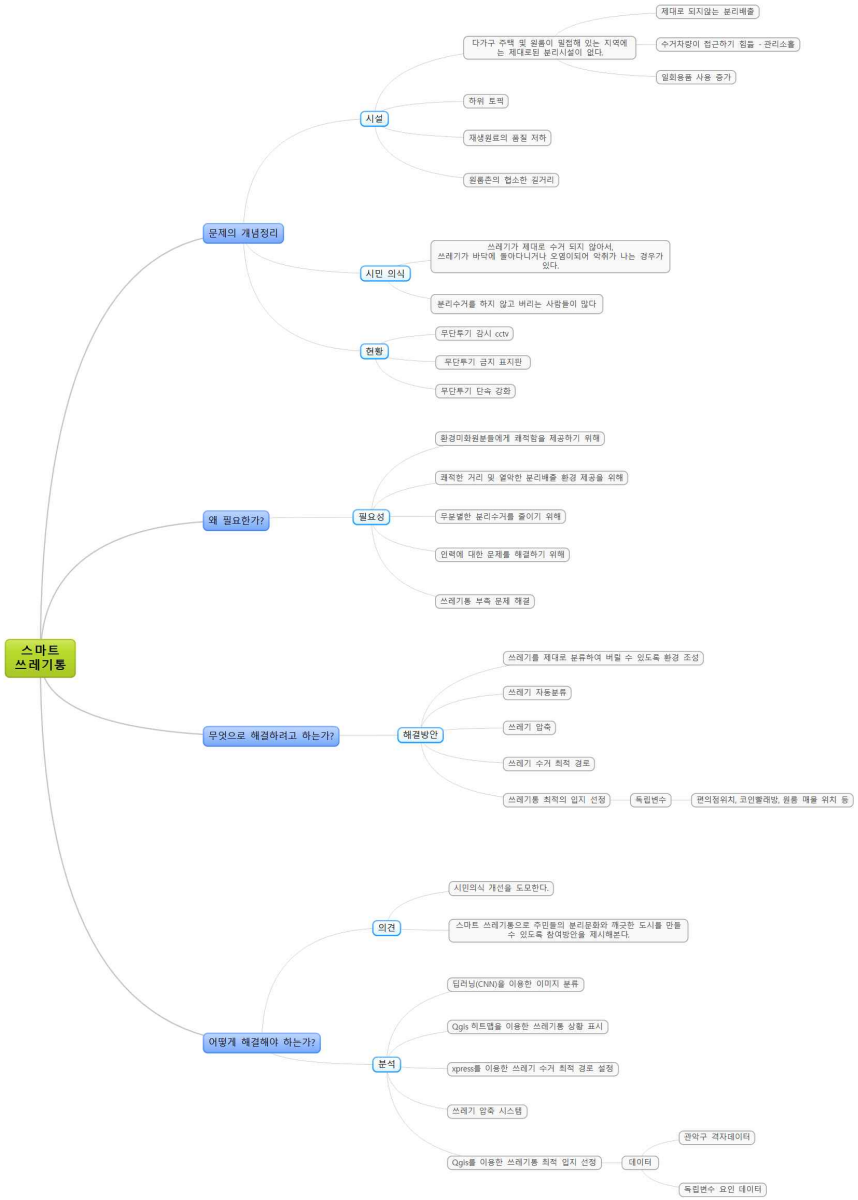
전봇대 곳곳에 쌓여 있는 재활용 쓰레기 봉투, 분리수거 통조차 제대로 비치돼 있지 않은 원룸 건물‥' 서울을 비롯한 전국 곳곳의 자취촌에서 흔히 볼 수 있는 광경이다. 최근 재활용품 업체들의 비닐과 스티로품 수거 중단으로 인해 아파트를 중심으로 이른바 ‘쓰레기 분리수거’ 대란이 일어났다. 원룸 주위는 아파트와 달리 제대로 된 분리수거 시설조차 설치돼 있지 않아, 방치된 쓰레기로 인해 입주민들의 건강마저 위협받는 실정이다.

7.1.2 기사(2)

□ “구역질 나 못 살아.. 폭염 속 무단투기 쓰레기 악취 ‘속수무책’”

서울시 구로구에 사는 A씨는 “건물 앞이 쓰레기장도 아닌데 여름이라 냄새도 이만저만이 아니고 파리도 들끓는다”며 집 앞 쓰레기 배출이라는 기본적인 원칙을 왜 안 지키는 건지 모르겠다고 불만을 드러냈다. 관악구의 경우 지난 6월 한 달간 무단투기가 주로 이뤄지는 주말을 중심으로 남부순환로, 관악로, 봉천로 등을 집중 단속했다.

7.2 마인드 맵



7.3 상세코드

7.3.1 데이터 전처리

□ 서울시 관악구 행정동별 1인 가구 데이터

[1]:
```
pip install geopy
```
```
Requirement already satisfied: geopy in c:\programdata\anaconda3\lib\site-packages (2.2.0)
Requirement already satisfied: geographiclib<2,>=1.49 in c:\programdata\anaconda3\lib\site-packages (from geopy) (1.52)
Note: you may need to restart the kernel to use updated packages.
```

[2]:
```
######## 도로명주소 위도 경도 값으로 바꿔주기 ########
from geopy.geocoders import Nominatim
geo_local = Nominatim(user_agent='South Korea')
# 위도, 경도 반환하는 함수
def geocoding(address):
    geo = geo_local.geocode(address)
    x_y = [geo.latitude, geo.longitude]
    return x_y
```

[3]:
```
import pandas as pd
```

[5]:
```
df = pd.read_csv('서울관악구주소좌표인코딩완료.csv')
```

[6]:
```
df.head()
```

[6]:

	서울특별시 관악구 주소	행정동	순위	단위(명)	비율	위도	경도
0	서울특별시 관악구 남부순환로230길 25	낙성대동	1	30	12.66	NaN	NaN
1	서울특별시 관악구 남부순환로230길 63	낙성대동	2	27	11.39	NaN	NaN
2	서울특별시 관악구 관악로6길 65	낙성대동	3	26	10.97	NaN	NaN
3	서울특별시 관악구 봉천로 575	낙성대동	4	19	8.02	NaN	NaN
4	서울특별시 관악구 낙성대로 23	낙성대동	5	16	6.75	NaN	NaN

[7]:
```
address = df['서울특별시 관악구 주소']
```

[8]:
```
from geopy.geocoders import Nominatim

def geocoding(address):
    geolocoder = Nominatim(user_agent = 'South Korea', timeout=None)
    geo = geolocoder.geocode(address)
    crd = {"lat": str(geo.latitude), "lng": str(geo.longitude)}

    return crd

crd = geocoding("서울특별시 관악구 남현1길 9")
print(crd['lat'])
print(crd['lng'])
```
```
37.4758604
126.9796072
```

[9]:
```
crd = geocoding('서울특별시 관악구 봉천동 41-598')
print(crd)
```
```
{'lat': '37.4837516', 'lng': '126.9480209'}
```

[13]:
```
latitude = []
longitude =[]
#row = 809
#address2 = address[row:]
for i in address:
    crd = geocoding(i)
    #print(i,crd,row,'행')
    latitude.append(crd['lat'])
    longitude.append(crd['lng'])
    #row += 1
```
```
...
```


[63]:
```
df2 = df.drop(columns = ['위도','경도'])
```

[64]:
```
df2
```

[64]:

	서울특별시 관악구 주소	행정동	순위	단위(명)	비율	lat	lng
0	서울특별시 관악구 남부순환로230길 25	낙성대동	1	30	12.66	37.4777259	126.9552387
1	서울특별시 관악구 남부순환로230길 63	낙성대동	2	27	11.39	37.4777259	126.9552387
2	서울특별시 관악구 관악로6길 65	낙성대동	3	26	10.97	37.4763286	126.9533955
3	서울특별시 관악구 봉천로 575	낙성대동	4	19	8.02	37.4773863	126.96145562851146
4	서울특별시 관악구 낙성대로 23	낙성대동	5	16	6.75	37.476625999999996	126.95993953475607
...	...	...	...	...	...	...	...
370	서울특별시 관악구 행운7길 24-10	행운동	21	6	2.08	37.4822909	126.9599622
371	서울특별시 관악구 남부순환로247길 6-3	행운동	22	5	1.74	37.4784602	126.9656647
372	서울특별시 관악구 당곡2가길	보라매동	1	35	15.22	37.4912724	126.9299143
373	서울특별시 관악구 낙성대역14가길	인헌동	20	8	2.62	37.472489	126.9630933
374	서울특별시 관악구 남부순환로125길	조원동	10	19	4.16	37.4810097	126.9060371

375 rows × 7 columns

[67]:
```
df2.to_excel('관악구1인가구밀집위치좌표최종.xlsx', index = False , encoding = 'euc-kr')
```

□ 서울시 관악구 편의점 위치

In [2]:
```
import pandas as pd
import numpy as np

df = pd.read_csv('../input/finalstore/___.csv', encoding = 'euc-kr')
```

In [3]:
```
df
```

Out[3]:

	Unnamed: 0	업종	행정동	도로명 주소	데이터기준일자
0	미니스톱 관악미래점	편의점	난곡동	서울특별시 관악구 법원단지5가길 54	2022-06-20
1	국미국미아이스 신림점	편의점	서림동	서울특별시 관악구 신림로68길 14	2022-06-20
2	세븐일레븐 서당역점	편의점	남현동	서울특별시 관악구 남현1길 66	2022-06-20
3	세븐일레븐 관악남현길점	편의점	남현동	서울특별시 관악구 남현길 58	2022-06-20
4	이마트24 관악남부점	편의점	신림동	서울특별시 관악구 남부순환로172길 13	2022-06-20
...	...	...	...	...	...
423	GS25봉천가온점	편의점	중앙동	서울특별시 관악구 양녕로6길 12	2022-06-20
424	씨유신림역점	편의점	신림동	서울특별시 관악구 신림로 323	2022-06-20
425	세븐일레븐 서울특별시대한솔점	편의점	청룡동	서울특별시 관악구 관악로15길 23-16	2022-06-20
426	씨유 신림미소점	편의점	서림동	서울특별시 관악구 문성로38길 67	2022-06-20
427	세븐일레븐 관악과천대로점	편의점	낙성대동	서울특별시 관악구 과천대로 921	2022-06-20

428 rows × 5 columns

4	이마트24 관악남부점	편의점	신림동	서울특별시 관악구 남부순환로172길 13	2022-06-20
...	...	...	...	...	...
423	GS25봉천가온점	편의점	중앙동	서울특별시 관악구 양녕로6길 12	2022-06-20
424	씨유신림역점	편의점	신림동	서울특별시 관악구 신림로 323	2022-06-20
425	세븐일레븐 서울특별시대한솔점	편의점	청룡동	서울특별시 관악구 관악로15길 23-16	2022-06-20
426	씨유 신림미소점	편의점	서림동	서울특별시 관악구 문성로38길 67	2022-06-20
427	세븐일레븐 관악과천대로점	편의점	낙성대동	서울특별시 관악구 과천대로 921	2022-06-20

428 rows × 5 columns

In [4]:
```
from geopy.geocoders import Nominatim

def geocoding(address):
    geolocoder = Nominatim(user_agent = 'South Korea', timeout=None)
    geo = geolocoder.geocode(address)
    crd = {'lat': str(geo.latitude), 'lng': str(geo.longitude)}

    return crd

crd = geocoding('서울특별시 관악구 남현1길 9')
print(crd['lat'])
print(crd['lng'])
```
```
37.4758604
126.9796072
```

In [10]:
```
n = range(-1,1)

for i in n:
    n = i+1
    crd = geocoding(df['도로명 주소'].iloc[n])
    #print(x)
    x = crd['lat']
    y = crd['lng']
    output = pd.DataFrame({
        'ID' : range(0,428),
        'lat' : x,
        'lng' : y
    })
```

In [11]:
```
output
```

Out[11]:

	ID	lat	lng
0	0	37.4827718	126.9309411
1	1	37.4827718	126.9309411
2	2	37.4827718	126.9309411
3	3	37.4827718	126.9309411
4	4	37.4827718	126.9309411
...	...	...	...
423	423	37.4827718	126.9309411
424	424	37.4827718	126.9309411
425	425	37.4827718	126.9309411
426	426	37.4827718	126.9309411
427	427	37.4827718	126.9309411

428 rows × 3 columns

In [12]:
```
output.pop('ID')
```

Out[12]:
```
0        0
1        1
2        2
3        3
4        4
      ...
423    423
424    424
425    425
426    426
427    427
Name: ID, Length: 428, dtype: int64
```

In [13]:
```
final = pd.concat([df, output], axis = 1)
```

In [14]:
```
final.to_csv('final_store_1.csv', index = False, encoding = 'euc-kr')
```


□ 서울시 관악구 셀프빨래방 위치

In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/store1/store_12345.csv
/kaggle/input/clothesbox/_.csv
/kaggle/input/realfinal/_.csv
/kaggle/input/wtfwtfwtf/___.csv
/kaggle/input/store12345/___.csv
/kaggle/input/finalstore/___.csv
/kaggle/input/selfwash/.csv
/kaggle/input/wtfstore/___.csv
/kaggle/input/finalfinaltt/final_store.csv
```

In [2]:
```
import pandas as pd
import numpy as np

df = pd.read_csv('../input/selfwash/.csv', encoding = 'euc-kr')
```

In [3]:
```
df
```

Out[3]:

	도로명 주소	업종
0	서울특별시 관악구 남부순환로244길 26	셀프빨래방
1	서울특별시 관악구 호암로 579	셀프빨래방
2	서울특별시 관악구 인헌길 71	셀프빨래방
3	서울특별시 관악구 관천로12길 30	셀프빨래방
4	서울특별시 관악구 남부순환로187길 30	셀프빨래방
...	...	...
132	서울특별시 관악구 신림로68길 43	셀프빨래방
133	서울특별시 관악구 신림로17길 42	셀프빨래방
134	서울특별시 관악구 솔밭로 19-1	셀프빨래방
135	서울특별시 관악구 호암로26길 42	셀프빨래방
136	서울특별시 관악구 관악로 209-2	셀프빨래방

137 rows × 2 columns

In [4]:
```
from geopy.geocoders import Nominatim

def geocoding(address):
    geolocoder = Nominatim(user_agent = 'South Korea', timeout=None)
    geo = geolocoder.geocode(address)
    crd = {"lat": str(geo.latitude), "lng": str(geo.longitude)}

    return crd
```

In [5]:
```
# 주소 값 위경도 출력
x = []
y = []

for i in range(-1,136):
    i = i+1
    crd = geocoding(df['도로명 주소'].iloc[i])
    #print(crd)
    x.append(crd['lat'])
    y.append(crd['lng'])



output = pd.DataFrame({
        'lat' : x,
        'lng' : y
    })

output
```

Out[5]:

	lat	lng
0	37.4764994	126.9632036
1	37.4596695	126.9226797
2	37.4723652	126.96686051414687
3	37.4857622	126.9269708
4	37.4853624	126.9334602
...	...	...
132	37.4867434	126.9308639
133	37.4670751	126.935805
134	37.477706	126.9627441
135	37.4704161	126.9355531
136	37.4887393	126.9577365

137 rows × 2 columns

In [6]:
```
final = pd.concat([df, output], axis = 1)
```

In [7]:
```
final
```

Out[7]:

	도로명 주소	업종	lat	lng
0	서울특별시 관악구 남부순환로244길 26	셀프빨래방	37.4764994	126.9632036
1	서울특별시 관악구 호암로 579	셀프빨래방	37.4596695	126.9226797
2	서울특별시 관악구 인헌길 71	셀프빨래방	37.4723652	126.96686051414687
3	서울특별시 관악구 관천로12길 30	셀프빨래방	37.4857622	126.9269708
4	서울특별시 관악구 남부순환로187길 30	셀프빨래방	37.4853624	126.9334602
...	...	...	...	...
132	서울특별시 관악구 신림로68길 43	셀프빨래방	37.4867434	126.9308639
133	서울특별시 관악구 신림로17길 42	셀프빨래방	37.4670751	126.935805
134	서울특별시 관악구 솔밭로 19-1	셀프빨래방	37.477706	126.9627441
135	서울특별시 관악구 호암로26길 42	셀프빨래방	37.4704161	126.9355531
136	서울특별시 관악구 관악로 209-2	셀프빨래방	37.4887393	126.9577365

137 rows × 4 columns

In [8]:
```
final.to_csv('final_wash_777.csv', index = False, encoding = 'euc-kr')
```


□ 서울시 관악구 의류수거함 위치

주소데이터 -> 경위도변환

[1]:
```
pip install geopy
```

```
Requirement already satisfied: geopy in c:\programdata\anaconda3\lib\site-packages (2.2.0)
Requirement already satisfied: geographiclib<2,>=1.49 in c:\programdata\anaconda3\lib\site-packages (from geopy) (1.52)
Note: you may need to restart the kernel to use updated packages.
```

[2]:
```
######## 도로명주소 위도 경도 값으로 바꿔주기 ########
from geopy.geocoders import Nominatim
geo_local = Nominatim(user_agent='South Korea')
# 위도, 경도 반환하는 함수
def geocoding(address):
    geo = geo_local.geocode(address)
    x_y = [geo.latitude, geo.longitude]
    return x_y
```

[3]:
```
import pandas as pd
```

[122]:
```
df = pd.read_excel('train_df.xlsx')
```

[69]:
```
df.head()
```

[69]:

	의류수거함	주소
0	낙성대동-1	서울특별시 관악구 관악로10길 15
1	낙성대동-2	서울특별시 관악구 관악로10길 6
2	낙성대동-3	서울특별시 관악구 관악로10길 77
3	낙성대동-4	서울특별시 관악구 관악로10길 87
4	낙성대동-5	서울특별시 관악구 관악로12길 16

[123]:
```
address = df['주소']
```

[26]:
```
from geopy.geocoders import Nominatim

def geocoding(address):
    geolocoder = Nominatim(user_agent = 'South Korea', timeout=None)
    geo = geolocoder.geocode(address)
    crd = {"lat": str(geo.latitude), "lng": str(geo.longitude)}

    return crd

crd = geocoding("서울특별시 관악구 남현1길 9")
print(crd['lat'])
print(crd['lng'])
```

```
37.4758604
126.9796072
```

[119]:
```
crd = geocoding('서울특별시 관악구 봉천동 41-598')
print(crd)
```

```
{'lat': '37.4839131', 'lng': '126.9480467'}
```

```
latitude = []
longitude =[]
#row = 809
address2 = address[row:]
for i in address:
    crd = geocoding(i)
    #print(i,crd,row,'됨')
    latitude.append(crd['lat'])
    longitude.append(crd['lng'])
    row += 1
```

```
len(latitude)
```

```
887
```

```
df['lat'] = latitude
```

```
df.head()
```

	서울특별시 관악구 주소	행정동	순위	단위(명)	비율	위도	경도	lat
0	서울특별시 관악구 남부순환로230길 25	낙성대동	1	30	12.66	NaN	NaN	37.4777259
1	서울특별시 관악구 남부순환로230길 63	낙성대동	2	27	11.39	NaN	NaN	37.4777259
2	서울특별시 관악구 관악로6길 65	낙성대동	3	26	10.97	NaN	NaN	37.4763286
3	서울특별시 관악구 봉천로 575	낙성대동	4	19	8.02	NaN	NaN	37.4773863
4	서울특별시 관악구 낙성대로 23	낙성대동	5	16	6.75	NaN	NaN	37.476625999999996

```
df['lng'] = longitude
```

```
df.to_excel('의류수거함좌표최종.xlsx', index = False)
```

```
df.head()
```

	의류수거함	주소	lat	lng
0	낙성대동-1	서울특별시 관악구 관악로10길 15	37.4766954	126.9548079
1	낙성대동-2	서울특별시 관악구 관악로10길 6	37.4766954	126.9548079
2	낙성대동-3	서울특별시 관악구 관악로10길 77	37.476532750000004	126.95687032038263
3	낙성대동-4	서울특별시 관악구 관악로10길 87	37.4765783	126.95750096822009
4	낙성대동-5	서울특별시 관악구 관악로12길 16	37.476826450000004	126.9558850229873

7.3.2 회귀분석_RandomForest_변수들의 가중치 값 구하기

In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the curren
```



```
l_csv('회귀분석데이터_취합완료.csv' , encoding = 'euc-kr')
```

CCTV 개수	의류수거함 개수	편의점	빨래방 개수	가로등 개수	1인가구 (명)
113	71	27	7	391	153
83	22	4	1	367	52
109	64	19	4	414	289
88	59	70	17	364	396
45	20	19	4	166	168
74	38	20	6	285	365
86	46	29	17	353	290
66	28	17	8	253	457
56	36	23	15	263	892

In [4]:

```
X_train_id = df['name']
df.pop('name')

df.corr()
```

Out[4]:

	CCTV 개수	의류수거함 개수	편의점	빨래방 개수	가로등 개수	1인가구 (명)
CCTV 개수	1.000000	0.611501	0.378755	0.291895	0.877722	-0.021610
의류수거함 개수	0.611501	1.000000	0.479955	0.237220	0.663679	0.255040
편의점	0.378755	0.479955	1.000000	0.654577	0.379478	0.329459
빨래방 개수	0.291895	0.237220	0.654577	1.000000	0.293406	0.543856
가로등 개수	0.877722	0.663679	0.379478	0.293406	1.000000	0.051582
1인가구 (명)	-0.021610	0.255040	0.329459	0.543856	0.051582	1.000000

In [5]:

```
y_train = df['1인가구 (명)']
df.pop('1인가구 (명)')
X_train = df
```

In [6]:

```
!pip install xgboost
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: xgboost in c:₩users₩user₩appdata₩roaming₩python₩pytho
n39₩site-packages (1.6.1)
Requirement already satisfied: scipy in c:₩programdata₩anaconda3₩lib₩site-packages
(from xgboost) (1.7.3)
Requirement already satisfied: numpy in c:₩programdata₩anaconda3₩lib₩site-packages
(from xgboost) (1.21.5)
```

In [7]:

```
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.metrics import r2_score, mean_squared_error
```

In [8]:

```
model = RandomForestRegressor()
RandomForest = model.fit(X_train, y_train)
output = model.predict(X_train)
print( '결정계수 : ' , model.score(X_train,y_train))
output
```

```
결정계수 :  0.8458468842158592
```

Out[8]:

```
array([211.3 , 123.64, 269.68, 421.73, 227.53, 326.5 , 408.66, 383.08,
       673.12, 302.16, 254.36, 252.1 , 394.41, 372.55, 293.53, 251.72,
       251.8 , 290.01,  95.65, 246.26])
```

In [9]:

```
!pip install eli5
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: eli5 in c:₩users₩user₩appdata₩roaming₩python₩python39
₩site-packages (0.13.0)
Requirement already satisfied: six in c:₩programdata₩anaconda3₩lib₩site-packages (fr
om eli5) (1.16.0)
Requirement already satisfied: scipy in c:₩programdata₩anaconda3₩lib₩site-packages
(from eli5) (1.7.3)
Requirement already satisfied: tabulate>=0.7.7 in c:₩programdata₩anaconda3₩lib₩site-
packages (from eli5) (0.8.9)
Requirement already satisfied: numpy>=1.9.0 in c:₩programdata₩anaconda3₩lib₩site-pac
kages (from eli5) (1.21.5)
Requirement already satisfied: graphviz in c:₩users₩user₩appdata₩roaming₩python₩pyth
on39₩site-packages (from eli5) (0.20.1)
Requirement already satisfied: scikit-learn>=0.20 in c:₩users₩user₩appdata₩roaming₩p
ython₩python39₩site-packages (from eli5) (1.1.1)
Requirement already satisfied: jinja2>=3.0.0 in c:₩users₩user₩appdata₩roaming₩python
₩python39₩site-packages (from eli5) (3.1.2)
Requirement already satisfied: attrs>17.1.0 in c:₩programdata₩anaconda3₩lib₩site-pac
kages (from eli5) (21.4.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:₩programdata₩anaconda3₩lib₩site-
packages (from jinja2>=3.0.0->eli5) (2.0.1)
Requirement already satisfied: joblib>=1.0.0 in c:₩programdata₩anaconda3₩lib₩site-pa
ckages (from scikit-learn>=0.20->eli5) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:₩programdata₩anaconda3₩lib
₩site-packages (from scikit-learn>=0.20->eli5) (2.2.0)
```


In [10]:

```
# CCTV 개수, 의류수거함 개수, 편의점, 빨래방 개수, 가로등 개수 순 변수 중요도

print(RandomForest.feature_importances_)

import eli5
from eli5.sklearn import PermutationImportance
eli5.show_weights(RandomForest, feature_names = X_train.columns.tolist())
```

```
[0.1330305  0.19250986 0.15824802 0.37417276 0.14203886]
```

Out[10]:

Weight	Feature
0.3742 ± 0.5796	빨래방 개수
0.1925 ± 0.4414	의류수거함 개수
0.1582 ± 0.3833	편의점
0.1420 ± 0.4003	가로등 개수
0.1330 ± 0.3445	CCTV 개수

In [11]:

```
'''from sklearn.preprocessing import StandardScaler
cols = ['CCTV 개수', '의류수거함 개수', '편의점', '빨래방 개수', '가로등 개수']

ss = StandardScaler()
X_train[cols] = ss.fit_transform(X_train[cols])
X_train'''
```

Out[11]:

```
"from sklearn.preprocessing import StandardScalerWncols = ['CCTV 개수', '의류수거함
개수', '편의점', '빨래방 개수', '가로등 개수']WnWnss = StandardScaler()WnX_train[col
s] = ss.fit_transform(X_train[cols])WnX_train"
```

In [12]:

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
model.score(X_train, y_train)
```

Out[12]:

```
0.4290170670140122
```

In [13]:

```
import statsmodels.api as sm

x_data = X_train #변수 여러개
target = y_train

# for b0, 상수항 추가
x_data1 = sm.add_constant(x_data, has_constant = "add")

# OLS 검정
multi_model = sm.OLS(target, x_data1)
fitted_multi_model = multi_model.fit()
fitted_multi_model.summary()
```

Out[13]:

OLS Regression Results

Dep. Variable:	1인가구 (명)	**R-squared:**	0.429
Model:	OLS	**Adj. R-squared:**	0.225
Method:	Least Squares	**F-statistic:**	2.104
Date:	Fri, 05 Aug 2022	**Prob (F-statistic):**	0.125
Time:	14:06:32	**Log-Likelihood:**	-126.23
No. Observations:	20	**AIC:**	264.5
Df Residuals:	14	**BIC:**	270.4
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	251.6020	141.414	1.779	0.097	-51.702	554.906
CCTV 개수	-3.7196	3.808	-0.977	0.345	-11.887	4.447
의류수거함 개수	4.2319	2.890	1.464	0.165	-1.967	10.430
편의점	-2.0559	4.057	-0.507	0.620	-10.757	6.645
빨래방 개수	25.2851	10.388	2.434	0.029	3.005	47.565
가로등 개수	-0.0127	1.010	-0.013	0.990	-2.178	2.153

Omnibus:	4.853	**Durbin-Watson:**	2.098
Prob(Omnibus):	0.088	**Jarque-Bera (JB):**	2.758
Skew:	0.842	**Prob(JB):**	0.252
Kurtosis:	3.688	**Cond. No.**	1.27e+03

7.3.3 TSP 알고리즘

In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/twopoint/_.csv
/kaggle/input/tsp-final/3 .csv
```

In [2]:

```
import pandas as pd
import numpy as np

df = pd.read_csv('../input/tsp-final/3 .csv')
df_id = df.pop('ID')
array = np.array(df)
```

In [3]:

```
# !pip install routing
```

In [4]:

```
import math
from ortools.constraint_solver import routing_enums_pb2
from ortools.constraint_solver import pywrapcp

df = pd.read_csv('../input/tsp-final/3 .csv')
df_id = df.pop('ID')
array = np.array(df)

# 거리 행렬을 이용한 유클리디안 거리 구하기
def create_data_model():
    data = {}
    data['locations'] = array
    data['num_vehicles'] = 1
    data['depot'] = 0
    return data

def compute_euclidean_distance_matrix(locations):
    distances = {}
    for from_counter, from_node in enumerate(locations):
        distances[from_counter] = {}
        for to_counter, to_node in enumerate(locations):
            if from_counter == to_counter:
                distances[from_counter][to_counter] = 0
            else:
                distances[from_counter][to_counter] = (int(
                    math.hypot((from_node[0] - to_node[0]),
                               (from_node[1] - to_node[1]))))
    return distances

def print_solution(manager, routing, solution):
    print('Objective: {}'.format(solution.ObjectiveValue()))
    index = routing.Start(0)
    plan_output = '신봉그린 최적 경로:\n'
    route_distance = 0
    while not routing.IsEnd(index):
        plan_output += ' {} ->'.format(manager.IndexToNode(index))
        previous_index = index
        index = solution.Value(routing.NextVar(index))
        route_distance += routing.GetArcCostForVehicle(previous_index, index, 0)
    plan_output += ' {}\n'.format(manager.IndexToNode(index))
    print(plan_output)
    #plan_output += 'Objective: {}m\n\n'.format(route_distance)


#TSP 알고리즘을 이용한 최적 경로 탐색
def main():
    data = create_data_model()
    manager = pywrapcp.RoutingIndexManager(len(data['locations']),
                                           data['num_vehicles'], data['depot'])
    routing = pywrapcp.RoutingModel(manager)
    distance_matrix = compute_euclidean_distance_matrix(data['locations'])

    def distance_callback(from_index, to_index):
        from_node = manager.IndexToNode(from_index)
        to_node = manager.IndexToNode(to_index)
        return distance_matrix[from_node][to_node]

    transit_callback_index = routing.RegisterTransitCallback(distance_callback)
    routing.SetArcCostEvaluatorOfAllVehicles(transit_callback_index)
    search_parameters = pywrapcp.DefaultRoutingSearchParameters()
    search_parameters.first_solution_strategy = (
        routing_enums_pb2.FirstSolutionStrategy.PATH_CHEAPEST_ARC)

    solution = routing.SolveWithParameters(search_parameters)

    if solution:
        print_solution(manager, routing, solution)


if __name__ == '__main__':
    main()
```

```
Objective: 10172
신봉그린 최적 경로:
 0 -> 24 -> 22 -> 4 -> 3 -> 23 -> 5 -> 16 -> 1 -> 10 -> 15 -> 20 -> 13 -> 17 ->
21 -> 14 -> 28 -> 26 -> 25 -> 29 -> 2 -> 6 -> 11 -> 12 -> 9 -> 8 -> 7 -> 30 ->
27 -> 19 -> 18 -> 0
```


7.3.4 딥러닝 알고리즘

In [1]:

```
import os
import torch
import torchvision
from torch.utils.data import random_split
import torchvision.models as models
import torch.nn as nn
import torch.nn.functional as F
```

In [2]:

```
# 파일 불러오기 및 종류 확인
data_dir  = '../input/final-garbage/Garbage classification'

classes = os.listdir(data_dir)
print(classes)
```

```
['metal', 'glass', 'paper', 'trash', 'plastic']
```

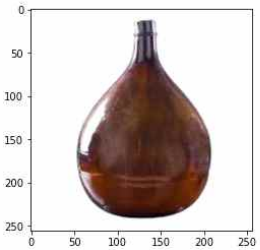
In [3]:

```
# 이미지 사이즈 설정
from torchvision.datasets import ImageFolder
import torchvision.transforms as transforms

transformations = transforms.Compose([transforms.Resize((256, 256)), transforms.ToT
ensor()])

dataset = ImageFolder(data_dir, transform = transformations)
```

In [4]:
```
# 나타낼 때 옵션
import matplotlib.pyplot as plt
%matplotlib inline

def show_sample(img, label):
    print("Label:", dataset.classes[label], "(Class No: "+ str(label) + ")")
    plt.imshow(img.permute(1, 2, 0))
```

In [5]:
```
img, label = dataset[12]
show_sample(img, label)
```

```
Label: glass (Class No: 0)
```



In [6]:
```
random_seed = 42
torch.manual_seed(random_seed)
```

Out[6]:
```
<torch._C.Generator at 0x7f936c78df10>
```

In [7]:
```
# 전체 이미지를 , (train, val, test)로 나눔
train_ds, val_ds, test_ds = random_split(dataset, [4738, 523, 2255])
len(train_ds), len(val_ds), len(test_ds)
```

Out[7]:
```
(4738, 523, 2255)
```

In [8]:
```
from torch.utils.data.dataloader import DataLoader
batch_size = 32
```

In [9]:
```
train_dl = DataLoader(train_ds, batch_size, shuffle = True, num_workers = 4, pin_me
mory = True)
val_dl = DataLoader(val_ds, batch_size*2, num_workers = 4, pin_memory = True)
```


In [10]:
```
from torchvision.utils import make_grid

def show_batch(dl):
    for images, labels in dl:
        fig, ax = plt.subplots(figsize=(12, 6))
        ax.set_xticks([])
        ax.set_yticks([])
        ax.imshow(make_grid(images, nrow = 16).permute(1, 2, 0))
        break
```

In [11]:
```
show_batch(train_dl)
```

In [12]:
```
# 평가지표(accuracy) 생성 및 train, val, test 평가지표 구분

def accuracy(outputs, labels):
    _, preds = torch.max(outputs, dim=1)
    return torch.tensor(torch.sum(preds == labels).item() / len(preds))

class ImageClassificationBase(nn.Module):
    def training_step(self, batch):
        images, labels = batch
        out = self(images)                  # Generate predictions
        loss = F.cross_entropy(out, labels) # Calculate loss
        return loss

    def validation_step(self, batch):
        images, labels = batch
        out = self(images)                    # Generate predictions
        loss = F.cross_entropy(out, labels)   # Calculate loss
        acc = accuracy(out, labels)           # Calculate accuracy
        return {'val_loss': loss.detach(), 'val_acc': acc}

    def validation_epoch_end(self, outputs):
        batch_losses = [x['val_loss'] for x in outputs]
        epoch_loss = torch.stack(batch_losses).mean()   # Combine losses
        batch_accs = [x['val_acc'] for x in outputs]
        epoch_acc = torch.stack(batch_accs).mean()      # Combine accuracies
        return {'val_loss': epoch_loss.item(), 'val_acc': epoch_acc.item()}

    def epoch_end(self, epoch, result):
        print("Epoch {}: train_loss: {:.4f}, val_loss: {:.4f}, val_acc: {:.4f}".format(
            epoch+1, result['train_loss'], result['val_loss'], result['val_acc']))
```

In [13]:
```
# CNN 알고리즘 중 Residual Networks 사용

class ResNet(ImageClassificationBase):
    def __init__(self):
        super().__init__()
        # Use a pretrained model
        self.network = models.resnet50(pretrained=True)
        # Replace last layer
        num_ftrs = self.network.fc.in_features
        self.network.fc = nn.Linear(num_ftrs, len(dataset.classes))

    def forward(self, xb):
        return torch.sigmoid(self.network(xb))

model = ResNet()
```

```
Downloading: "https://download.pytorch.org/models/resnet50-19c8e357.pth" to /root/.cach
e/torch/checkpoints/resnet50-19c8e357.pth
```

```
100% 97.8M/97.8M [00:05<00:00, 18.4MB/s]
```

In [14]:
```
def get_default_device():
    """Pick GPU if available, else CPU"""
    if torch.cuda.is_available():
        return torch.device('cuda')
    else:
        return torch.device('cpu')

def to_device(data, device):
    """Move tensor(s) to chosen device"""
    if isinstance(data, (list,tuple)):
        return [to_device(x, device) for x in data]
    return data.to(device, non_blocking=True)

class DeviceDataLoader():
    """Wrap a dataloader to move data to a device"""
    def __init__(self, dl, device):
        self.dl = dl
        self.device = device

    def __iter__(self):
        """Yield a batch of data after moving it to device"""
        for b in self.dl:
            yield to_device(b, self.device)

    def __len__(self):
        """Number of batches"""
        return len(self.dl)
```


In [15]:
```
device = get_default_device()
device
```

Out[15]:
```
device(type='cuda')
```

In [16]:
```
train_dl = DeviceDataLoader(train_dl, device)
val_dl = DeviceDataLoader(val_dl, device)
to_device(model, device)
```

Out[16]:
```
ResNet(
  (network): ResNet(
    (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=Fals
e)
    (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    (relu): ReLU(inplace=True)
    (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=Fals
e)
    (layer1): Sequential(
      (0): Bottleneck(
        (conv1): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stat
s=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias
=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stat
s=True)
```

In [17]:
```
@torch.no_grad()
def evaluate(model, val_loader):
    model.eval()
    outputs = [model.validation_step(batch) for batch in val_loader]
    return model.validation_epoch_end(outputs)

def fit(epochs, lr, model, train_loader, val_loader, opt_func=torch.optim.SGD):
    history = []
    optimizer = opt_func(model.parameters(), lr)
    for epoch in range(epochs):
        # Training Phase
        model.train()
        train_losses = []
        for batch in train_loader:
            loss = model.training_step(batch)
            train_losses.append(loss)
            loss.backward()
            optimizer.step()
            optimizer.zero_grad()
        # Validation phase
        result = evaluate(model, val_loader)
        result['train_loss'] = torch.stack(train_losses).mean().item()
        model.epoch_end(epoch, result)
        history.append(result)
    return history
```

In [18]:
```
model = to_device(ResNet(), device)
```

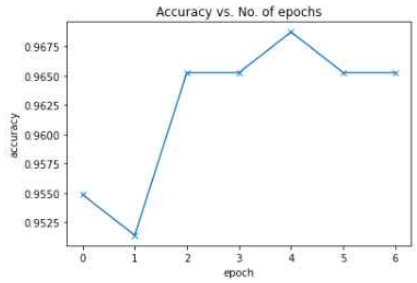
In [19]:
```
evaluate(model, val_dl)
```

Out[19]:
```
{'val_loss': 1.6052757501602173, 'val_acc': 0.16256314516067505}
```

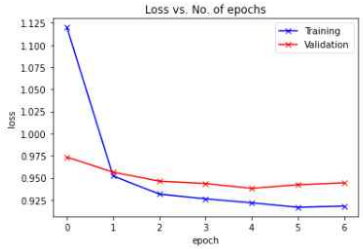
In [20]:
```
# epoch 설정 및 그에 따른 loss, accuracy 추출
# ex) epoch = 5 이면, 전체 데이터셋에 대해 5번 학습을 거친 것
num_epochs = 7
opt_func = torch.optim.Adam
lr = 5.5e-5

history = fit(num_epochs, lr, model, train_dl, val_dl, opt_func)
```

```
Epoch 1: train_loss: 1.1200, val_loss: 0.9734, val_acc: 0.9549
Epoch 2: train_loss: 0.9521, val_loss: 0.9564, val_acc: 0.9514
Epoch 3: train_loss: 0.9316, val_loss: 0.9462, val_acc: 0.9653
Epoch 4: train_loss: 0.9262, val_loss: 0.9435, val_acc: 0.9653
Epoch 5: train_loss: 0.9217, val_loss: 0.9380, val_acc: 0.9688
Epoch 6: train_loss: 0.9167, val_loss: 0.9421, val_acc: 0.9653
Epoch 7: train_loss: 0.9180, val_loss: 0.9443, val_acc: 0.9653
```

In [21]:
```
# 각 epoch별 acuraccy 시각화
def plot_accuracies(history):
    accuracies = [x['val_acc'] for x in history]
    plt.plot(accuracies, '-x')
    plt.xlabel('epoch')
    plt.ylabel('accuracy')
    plt.title('Accuracy vs. No. of epochs');

plot_accuracies(history)
```



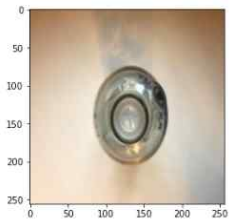
In [22]:
```
# 각 epoch별 loss 시각화
# loss 정답과 예측값 사이의 오차, MSE(평균제곱오차) 사용
def plot_losses(history):
    train_losses = [x.get('train_loss') for x in history]
    val_losses = [x['val_loss'] for x in history]
    plt.plot(train_losses, '-bx')
    plt.plot(val_losses, '-rx')
    plt.xlabel('epoch')
    plt.ylabel('loss')
    plt.legend(['Training', 'Validation'])
    plt.title('Loss vs. No. of epochs');

plot_losses(history)
```



In [23]:
```
def predict_image(img, model):
    # Convert to a batch of 1
    xb = to_device(img.unsqueeze(0), device)
    # Get predictions from model
    yb = model(xb)
    # Pick index with highest probability
    prob, preds  = torch.max(yb, dim=1)
    # Retrieve the class label
    return dataset.classes[preds[0].item()]
```

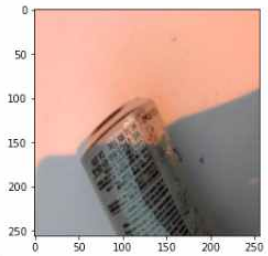
In [24]:
```
# 테스트 데이터셋으로 이미지 테스트

img, label = test_ds[234]
plt.imshow(img.permute(1, 2, 0))
print('Label:', dataset.classes[label], ', Predicted:', predict_image(img, model))
```

```
Label: glass , Predicted: glass
```



In [25]:
```
img, label = test_ds[55]
plt.imshow(img.permute(1, 2, 0))
print('Label:', dataset.classes[label], ', Predicted:', predict_image(img, model))
```

```
Label: metal , Predicted: metal
```



In [26]:
```
# 외부 이미지 크롤링 및 예측
import urllib.request
urllib.request.urlretrieve("https://external-content.duckduckgo.com/iu/?u=http%3A%2F%2Fflexible.img.hani.co.kr%2Fflexible%2Fnormal%2F724%2F482%2Fimgdb%2Foriginal%2F2020%2F0611%2F20200611504284.jpg&f=1&nofb=1", "a.jpg")
urllib.request.urlretrieve("https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Ft1.daumcdn.net%2Fcfile%2Ftistory%2F274C434858F0419C19&f=1&nofb=1", "c.jpg")
urllib.request.urlretrieve("https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fimg1.daumcdn.net%2Fthumb%2FR800x0%2F%3Fscode%3Dmtistory2%26fname%3Dhttps%3A%252F%252Ft1.daumcdn.net%252Fcfile%252Ftistory%252F2624F23D5749972B1B&f=1&nofb=1", "d.jpg")
```

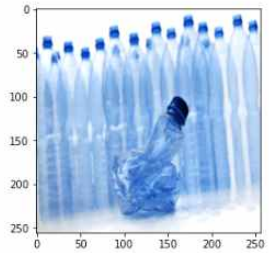
Out[26]:
```
('d.jpg', <http.client.HTTPMessage at 0x7f9336d06e10>)
```

In [27]:
```
loaded_model = model
```

In [28]:
```
from PIL import Image
from pathlib import Path

def predict_external_image(image_name):
    image = Image.open(Path('./' + image_name))

    example_image = transformations(image)
    plt.imshow(example_image.permute(1, 2, 0))
    print("The image resembles", predict_image(example_image, loaded_model) + ".")
```

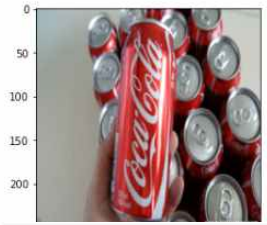

In [29]:
```
predict_external_image('a.jpg')
```

```
The image resembles plastic.
```



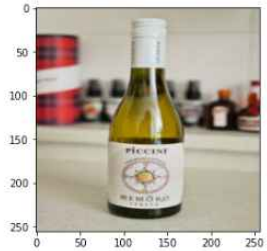
In [30]:
```
predict_external_image('c.jpg')
```

```
The image resembles metal.
```



In [31]:
```
predict_external_image('d.jpg')
```

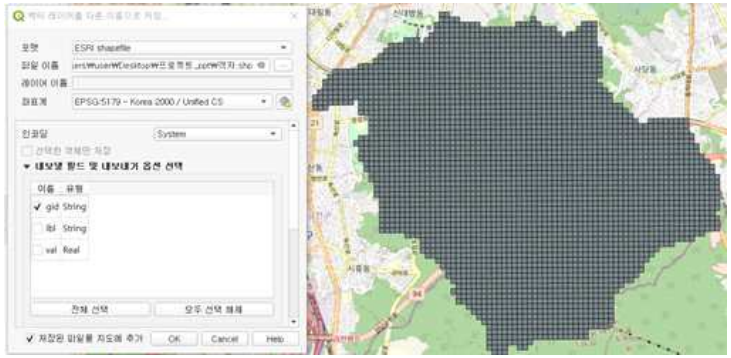
```
The image resembles glass.
```



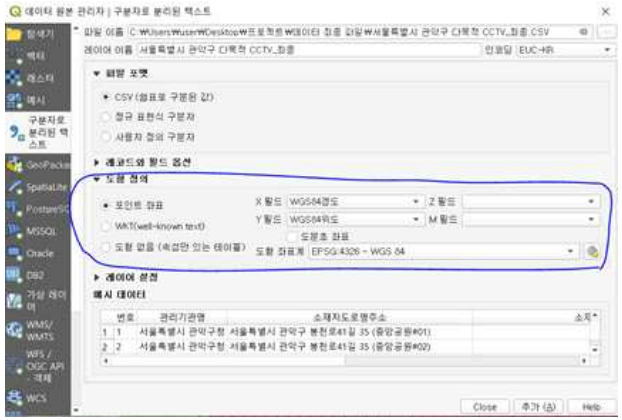
7.4 QGIS 실행화면

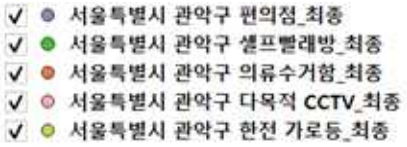
7.4.1 최적 입지 선정

☐ 서울시 관악구 100M 격자 불러오기

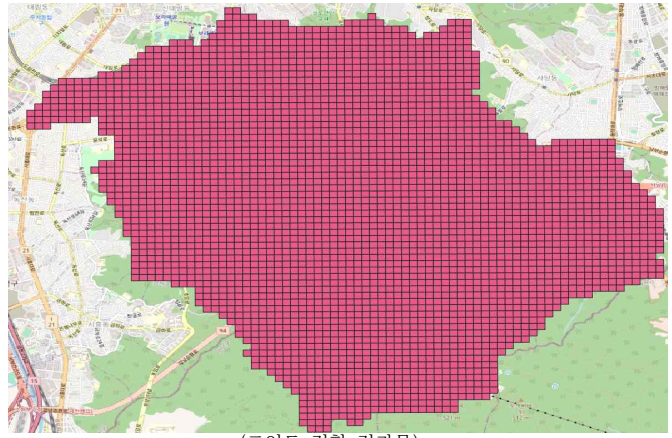


☐ 선정한 변수들을 포인트 좌표로 모두 불러오기





☐ 폴리곤에 포함하는 포인트 개수 계산, 다섯 개의 포인트 모두 결합



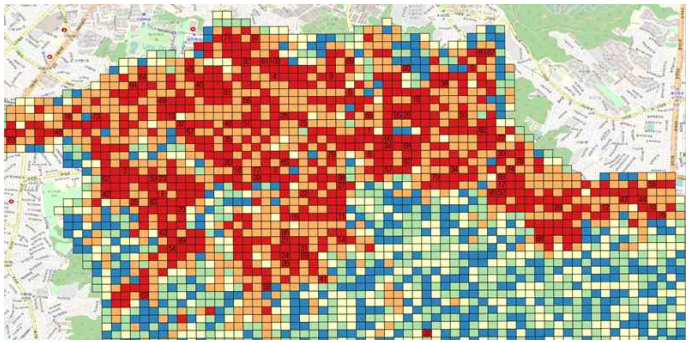
(포인트 결합 결과물)

□ 입지 선정에 필요한 위치(위도, 경도) 결합과 속성 테이블

□ 입지 선정순위에 필요한 변수들의 가중치 값 계산 후 순위 table

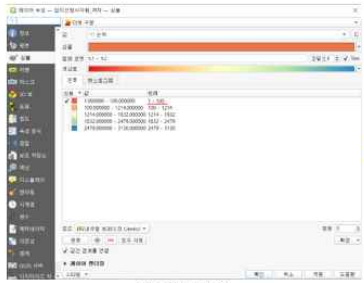
	A	B	C	D	E	F	G	H	I
1		gid	cctv가중치	가로등가중	셀프빨래방	의류수거함	편의점가중	총점	순위
2	0	다사50942	0.179	0.21475	0.3108	0.1738	15.2229	16.10125	1
3	1	다사49942	0.0895	0.2577	0	0.5214	9.226	10.0946	2
4	2	다사48342	0	0.1718	0.3108	0	6.4582	6.9408	3
5	3	다사49943	0.2685	0.3436	0.1036	0	4.613	5.3287	4
6	4	다사52043	0.179	0.21475	0.1036	0	4.1517	4.64905	5
7	5	다사50743	0.0895	0.1718	0.1036	0	4.1517	4.5166	6
8	6	다사48741	0.0895	0.2577	0.2072	1.5642	2.3065	4.4251	7
9	7	다사50942	0	0.0859	0.3108	0.1738	3.2291	3.7996	8
10	8	다사50543	0.0895	0.55835	0.1036	0.869	1.8452	3.46565	9
11	9	다사53941	0	0.1718	0	2.7808	0.4613	3.4139	10
12	10	다사50641	0.179	0.4295	0.1036	0.5214	1.8452	3.0787	11
13	11	다사53441	0.179	0.3436	0	1.5642	0.9226	3.0094	12
14	12	다사48842	0.0895	0.1718	0	0.869	1.8452	2.9755	13
15	13	다사50641	0.0895	0.55835	0	1.3904	0.9226	2.96085	14
16	14	다사50141	0.358	0.3436	0	0.3476	1.8452	2.8944	15

□ 최적 입지 선정 1~100순위



7.4.2 최적 경로

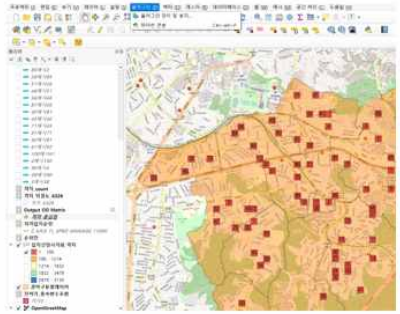
□ 최적 입지 선정 후, 해당 100곳만 추출



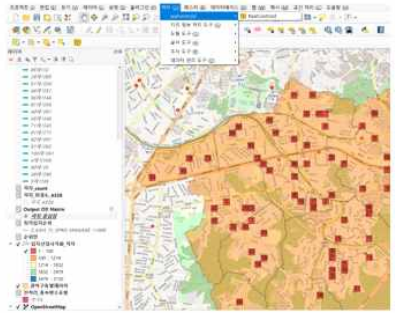
<레이어제거 화면>

<제거 후 결과>

□ 격자에 중심점을 찍기 위한 'Realcentroid' plug-in 설치

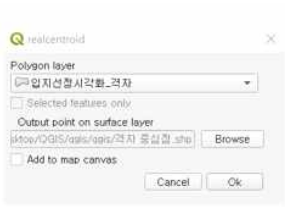


<플러그인 설치 메뉴>



<설치 후 결과>

□ Realcentroid 설정화면에서 최적 입지 격자를 기준으로 중심점 레이어를 생성하여 저장 후, 해당 중심점 레이어 파일을 불러오고 확인

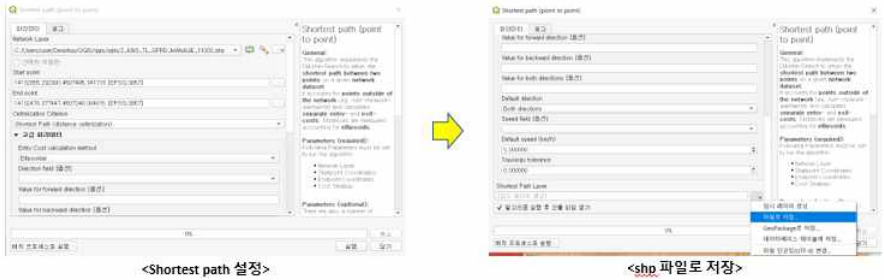


<RealCentroid 설정화면>

<실행 후 결과>

□ 점과 점 사이의 최단 거리를 구하기 위해 QNEAT3 plug-in 설치 이후, 공간 처리 툴박스에서 shortest path 기능 활용

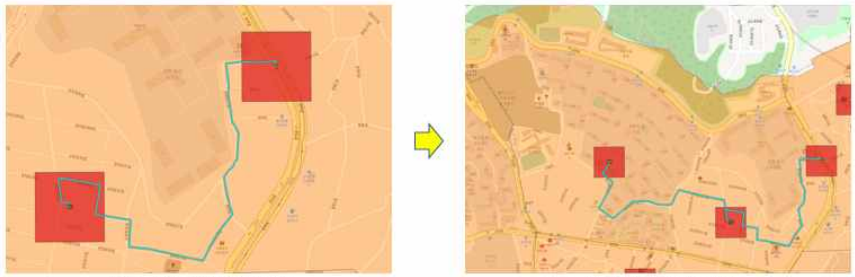
□ network layer는 서울 도로교통망 데이터로 설정 후 시작 점과 끝 점을 격자의 중심점으로 지정하여 설정, 마지막으로 shp 파일로 저장하고 실행



<Shortest path 설정>

<shp 파일로 저장>

□ 생성된 shortest path 레이어를 불러오면 왼쪽 사진처럼 나타남 이를 반복수행



<5번과 38번 사이의 최적 경로>

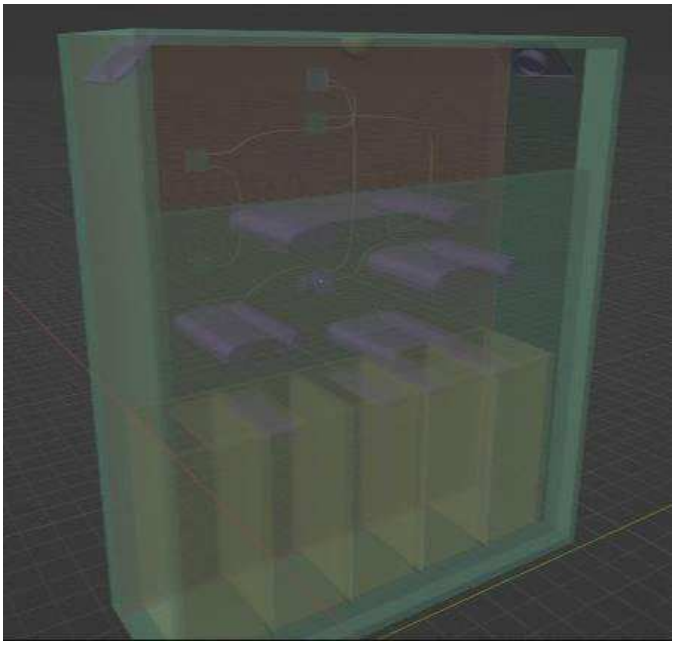
<성현동 수거 최적 경로>

□ 성현동 - 보라매동 - 신림동 - 서원동 - 신원동 - 미성동 순으로 진행한 후 전체적으로 최종 수거 최적경로를 확인

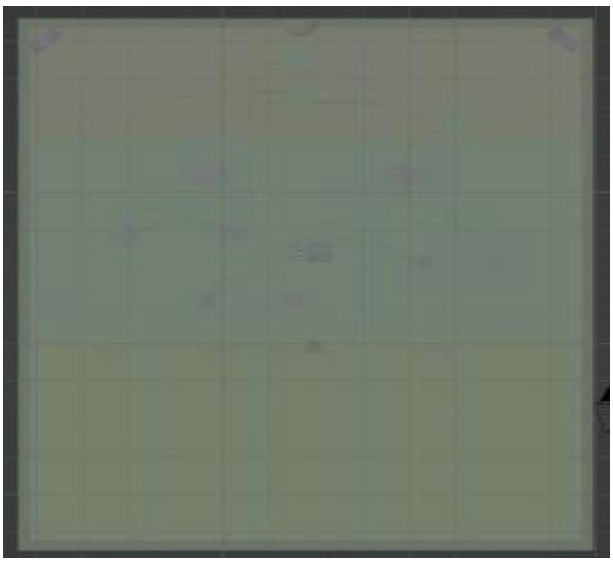
7.5 웹페이지 제작

□ 해당 페이지에서 카메라로 물체 인식 후 분류

7.6 프로토타입



내부



회로도