

데이터로 같이, 가치 있게(With Value)!

데이턴십 해커톤 제 5회

서울시 LED 바닥형 보행신호등 설치 최적 입지선정

분석 결과보고서

참여조: 8조

참여자: 박찬성, 김수영, 나지수,
박종현, 이건희, 이규희

씨에스리 컨소시엄

CSLEE kpc 한국생산성본부

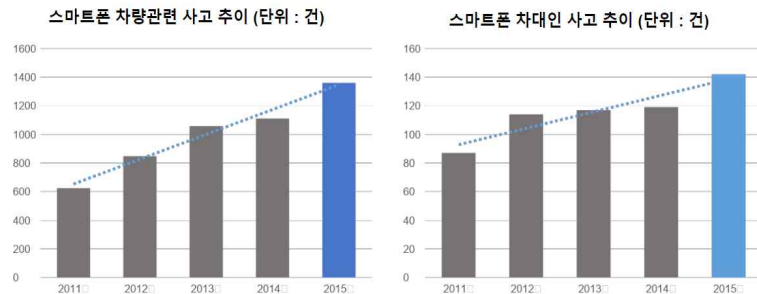
목 차

1. 분석 개요	3
1.1. 분석 배경 및 개요	3
1.2. 분석 목적 및 방향	8
1.3. 분석 결과 활용 방안	10
2. 분석 데이터	11
2.1. 분석 데이터 목록	11
2.2. 데이터 상세 설명	13
2.3. 데이터 정제 방안	15
3. 분석 프로세스	17
3.1. 분석 프로세스	17
3.2. 분석 내용 및 방법	35
4. 분석 결과	40
4.1. 모델링 결과	40
4.2. 최적 입지 선정	41
5. 활용 방안	45
5.1. 문제점 개선 방안	45
5.2. 업무 활용 방안	46
6. 참고자료(Reference)	48
7. 부록	56

1. 분석 개요

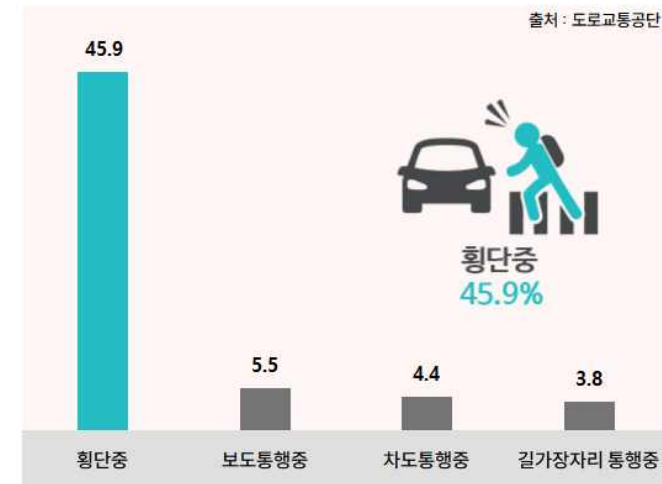
1.1 분석 배경 및 개요

□ 최근 전 세계적으로 스마트폰 보급이 늘어남에 따라 스마트폰 사용에 따른 안전 문제가 끊임없이 제기되고 있다. 특히 보행 중 스마트폰 사용에 따른 교통사고는 지속적으로 늘어나고 있는 실정이며 이에 보행 중 스마트폰을 사용하느라 고개를 숙이고 다니는 사람들을 일컬어 말하는 ‘스몸비족(스마트폰 + 좀비)’이라는 신조어까지 탄생했다. 스몸비족은 성별과 연령대를 가리지 않고 나타나고 있으며 보행자와 운전자 모두의 도로교통 안전에 위협을 가하고 있다.

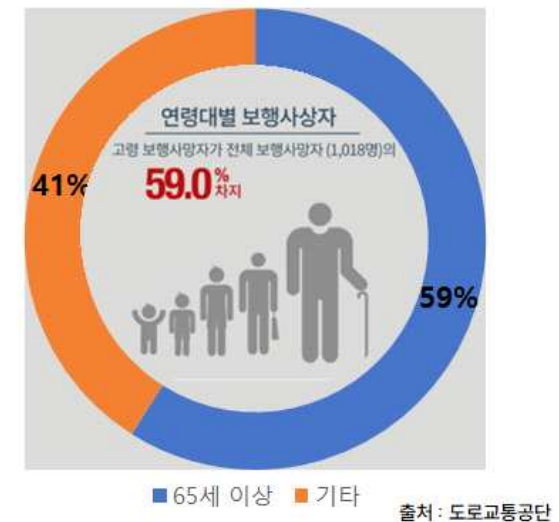


[그림 1-1] 이동 중 스마트폰 사용실태

보행 중 교통사고 가운데 횡단 중 발생하는 사고가 가장 많으며 그중에서도 어린이·노인 등 교통약자가 차지하는 비율이 매우 높다. 교통약자들이 도로를 횡단하다 사고를 많이 당하는 이유는 거리변별력이나 위험도 인지능력이 떨어지기 때문이다. 따라서 보행 취약계층의 교통사고를 줄이기 위해 횡단보도 주변에 보행자 보호 인프라를 확충하여 보행환경을 개선하는 것이 필요하다.

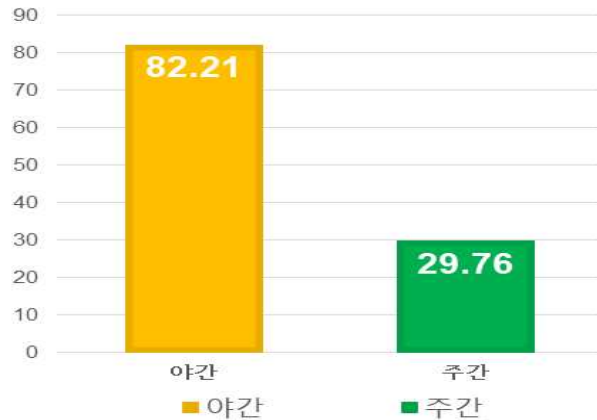


[그림 1-2] 서울 어린이 보행자사고



[그림 1-3] 노인 보행사상자 비율

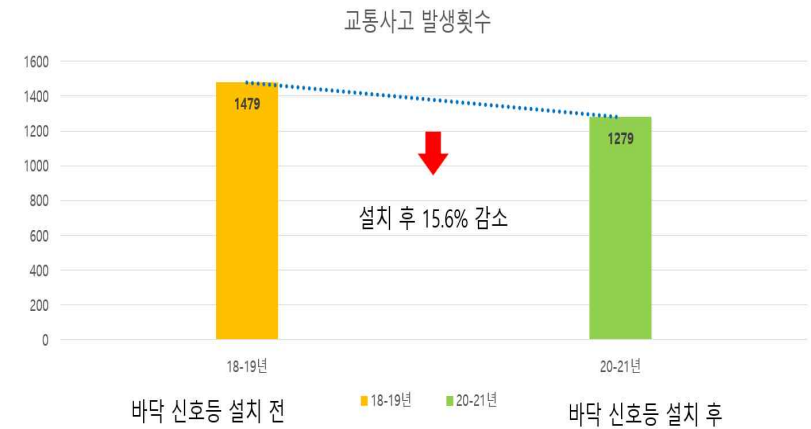
이에 국내·외에서 다차선도로와 교통량이 많은 도로, 취약계층 보호구역에서의 보행 안전을 위해 LED 바닥형 보행신호등을 설치하고 있다. 선행 연구 결과에 따르면 LED 바닥형 보행신호등은 주간 29.76%, 야간 82.21%의 교통사고 감소효과가 있는 것으로 확인되었다(신동협, 2021). 또한 횡단보도 대기선에서 추가적인 신호정보를 제공하여 횡단 중 교통사고 발생을 감소시킬 수 있다.



[그림 1-4] 바닥형 보행신호등 설치 시 교통사고 감소 효과



[그림 1-5] LED 바닥형 보행신호등



[그림 1-6] LED 바닥형 보행신호등 설치 전후 교통사고 발생 횟수

하지만 현재 국내는 2018년 대구광역시, 양주시, 남양주시 등 시범설치 구를 시작으로 점차 전국적으로 확대하고 있는 추세이나 예산편성 및 기타 문제로 인해 전 지역에 설치하지 못하고 소수의 지역에만 설치되고 있는 실정이다. 이에 설치되지 않은 지역에서 각 자치구마다 한정된 예산 내에서 LED 바닥형 보행신호등을 설치했을 때 최대

의 효과를 보일 수 있는 최적 입지에 관한 분석이 필요하다. 따라서 본 분석은 기존에 자치구마다 진행했던, 경험과 주민들의 민원 등을 기반으로 선정하는 방법에 더해 유동인구, 교통사고 수 등 관련 데이터를 활용하는 분류 모델을 개발하여 설치 최적 입지를 선정하는 것을 목표로 한다.

1.2 분석 목적 및 방향

□ 본 분석은 기존 LED 바닥 보행신호등이 설치된 장소를 관련 데이터를 활용하는 분류 모델을 개발하여 설치 최적 입지를 선정하여 스몸비족과 보행약자에게 안전한 보행환경 제공과 예산이 부족한 자치구에게 한정된 예산으로 최대의 효과를 제공함을 목적으로 한다.

본 주제와 관련된 LED 바닥형 보행신호등 위치 데이터, 교통사고 데이터, 서울특별시 횡단보도 위치 등 필요 데이터를 수집한다. 각 데이터를 법정동을 기준으로 통합하는 전처리 과정을 진행 후 다시 교통 데이터와 생활 데이터로 분류 및 취합한다.

LED 바닥형 보행신호등이 설치된 지역 중 상세 위치 데이터가 제공되는 서울특별시의 강남구, 광진구, 동대문구를 대상으로 하여 설치되기 전 18~19년도의 보행자 교통사고 건수와 설치 후 20~21년도 보행자 교통사고 건수를 비교하여 t-test를 수행한다. t-test 수행 전 shapiro test로 정규성을 검정하고, levene test로 등분산성을 검정한다. 검정 후 t-test 수행 결과 유의수준 5% 이내에서 3개의 구 모두 바닥형 보행신호등이 사고감소에 효과가 있다는 사실을 알 수 있었다.

<강남구 바닥형 보행신호등 효과 분석($\alpha = 0.05$)>

	정규성 검정	등분산성 검정	T-test
검정통계량	0.967(18-19) 0.945(20-21)	1.575	2.550
P-value	0.604(18-19) 0.216(20-21)	0.215	0.007

[그림 1-7] 강남구 바닥형 보행신호등 효과 분석

<광진구 바닥형 보행신호등 효과 분석($\alpha = 0.05$)>

	정규성	등분산성	T-test
검정통계량	0.176(18-19) 0.960(20-21)	0.002	3.853
P-value	0.395(18-19) 0.449(20-21)	0.964	0.000

[그림 1-8] 광진구 바닥형 보행신호등 효과 분석

<동대문구 바닥형 보행신호등 효과 분석($\alpha = 0.05$)>

	정규성	등분산성	T-test
검정통계량	0.978(18-19) 0.972(20-21)	1.757	6.340
P-value	0.861(18-19) 0.733(20-21)	0.191	0.000

[그림 1-9] 동대문구 바닥형 보행신호등 효과 분석

1.3 분석 결과 활용 방안

1.3.1 정책활용

□ 기존의 LED 바닥형 보행신호등의 입지 선정 정책은 주로 자치구의 경험을 기반으로 하거나 민원 다발지역, 사고 발생 건수, 어린이 보호구역의 유무, 차량 통행량을 기반으로 결정했다. 이에 본 분석에서 개발한 모델을 더하여 효과를 극대화할 수 있는 최적 입지 선정 의사결정에 도움을 줄 수 있으며 이는 곧 예산이 부족해 많이 설치할 수 없는 자치구라도, 최대의 효과를 볼 수 있도록 돕는다.

1.3.2 교통사고 방지

□ 기존에 LED 바닥형 보행신호등이 설치되지 않았던 지역에 데이터를 기반으로 최적의 위치를 도출한 지역에 설치하여 전방을 주시하지 않는 스몸비족에게 추가적인 신호정보를 제공함으로써 보행 편의와 횡단보도 사고 방지에 기여할 수 있다. 또한 어린이와 노인들을 비롯한 보행약자들도 안전하게 보행할 수 있는 환경을 조성할 수 있다.

1.3.3 상승효과

□ 현재 시중에 판매 중인 LED 바닥형 보행신호등에 추가적인 시설물 및 장치 등을 기획하여 제안하는 것으로 기존의 시제품보다 효과를 확대할 수 있다. 예를 들어 픽셀아트를 통해 바닥 신호등을 인식하게 하도록 유도하거나 바닥 신호등 근처에 형광 테이프를 설치하는 등의 방법으로 바닥형 신호등의 인식률을 높이고 LED 바닥형 보행신호등을 단독으로 사용하는 것보다 사고 위험을 감소시킬 수 있다.

2. 분석 데이터

2.1 분석 데이터 목록

2.1.1 원본 데이터 목록

☐ 원본 데이터 목록 표

[표 1-1] 원본 데이터 목록 표

데이터명	구분	출처
바닥형 보행신호등 위치 데이터	교통	공공데이터포털
보행자 교통사고 데이터	교통	TAAS(교통사고분석시스템)
서울특별시 횡단보도 위치 및 부착대 정보	교통	서울열린데이터광장
서울특별시 자동차 등록현황	교통	서울열린데이터광장
서울특별시 교차로 관련 정보	교통	서울열린데이터광장
지역별 교통안전지수	교통	TAAS(교통사고분석시스템)
서울특별시 행정동별 서울생활인구	생활	공공데이터포털
서울특별시 주민등록인구(동별) 통계	생활	서울열린데이터광장
서울특별시 평균연령(동별) 통계	생활	서울열린데이터광장
code_map	생활	KDX한국데이터거래소
행정구역코드	생활	통계청
행정구역 읍면동(법정동)	생활	국가공간정보포털

2.1.2 전처리 데이터 목록

☐ 전처리 데이터 목록 표

[표 1-2] 전처리 데이터 목록 표

데이터명	구분
법정동별_교통사고 횡수	교통 데이터
법정동별_횡단보도 수	교통 데이터
법정동_자동차 수	교통 데이터
법정동별_교차로 수	교통 데이터
법정동별_교통안전지수	교통 데이터
법정동별_주야별_생활인구	생활 데이터
법정동별_주민등록 수	생활 데이터
법정동별_평균연령	생활 데이터
법정동_면적	생활 데이터
서울특별시_행정구역_행정동_법정동 연계	생활 데이터

2.2 데이터 상세 설명

2.2.1 교통 데이터

□ 교통 원본 데이터

- 바닥형 보행신호등 위치 데이터 : 소재지주소 / 위도 / 경도
- 보행자 교통사고 데이터 : 사고일시 / 시군구 / 사고유형
- 서울특별시 횡단보도 위치 및 부착대 데이터 : X좌표 / Y좌표
- 서울특별시 자동차 등록현황 : 행정동 / 연료별
- 서울특별시 교차로 관련 정보 : 교차로명칭 / 교차로코드 / 지번
- 지역별 교통안전지수 : 시군구 / 안전지수

□ 교통 전처리 데이터

- 법정동별_교통사고 횡수 : 법정동별 주간사고 횡수/야간사고 횡수/야간사고 비율을 활용하여 야간사고 위험도를 산출한 데이터
- 법정동별_횡단보도 수 : 법정동별 횡단보도 수를 합계한 데이터
- 법정동_자동차 수 : 법정동별 자동차 수를 합계한 데이터
- 법정동별_교차로 수 : 법정동별 교차로 수를 합계한 데이터
- 법정동별_교통안전지수 : 법정동별 안전지수의 데이터

2.2.2 생활 데이터

□ 생활 원본 데이터

- 서울특별시 행정동별 서울생활인구 : 행정동코드 / 총생활인구수
- 서울특별시 주민등록인구(동별) 통계 : 동별 / 합계
- 서울특별시 평균연령(동별) 통계 : 동별 / 전체 평균연령
- code_map : 행정동코드 / 법정동코드
- 행정구역코드 : 행정기관코드 / 법정기관코드
- 행정구역 읍면동(법정동) : 법정동(읍면동단위) 경계도면 shp파일

□ 생활 전처리 데이터

- 법정동별_주야별_생활인구 : 법정동별 주·야 보정 생활 인구수를 나타낸 데이터
- 법정동별_주민등록 수 : 법정동별 주민등록 수를 합계한 데이터
- 법정동별_평균연령 : 법정동별 평균나이를 나타낸 데이터
- 법정동_면적 : 법정동별 기관 코드와 면적을 나타낸 데이터
- 서울시_행정구역_행정동_법정동 연계 : 서울시 구별 행정동, 법정동에 따른 구역분류, 기관 코드를 나타낸 데이터

2.3 데이터 정제 방안

2.3.1 교통 데이터 정제 방안

□ 서울특별시 차 대 사람 사고데이터 전처리

서울특별시 교통사고 데이터를 활용하여 동, 시간, 주·야구분 변수를 생성하고 야간 사고 비율을 계산하여 야간사고 위험도 변수 생성.

□ 동별 횡단보도 합계 전처리

QGIS를 이용하여 횡단보도 위치 데이터와 서울특별시 동별 위치 데이터를 레이어 후, 폴리곤인 횡단보도를 중심점 찾기 하여 동별 횡단보도 수를 카운트. 동별 횡단보도 합계를 추출한 뒤 행정구역 코드를 법정동 코드와 조인하여 법정동별 횡단보도 수 산출.

□ 자동차 보유 대수 전처리

서울특별시 자동차 등록현황 데이터를 활용하여 행정동별 합계 파악. code_map 데이터를 활용하여 법정동별 자동차 보유 대수 산출.

□ 교차로 데이터 전처리

서울특별시 교차로 관련 정보 데이터와 code_map 데이터를 활용하여 법정동별 교차로 수 산출.

□ 구별 교통안전지수 법정동별 전처리

지역별 교통안전지수 데이터와 code_map 데이터를 활용하여 법정동별 교통안전지수 산출.

2.3.2 생활 데이터 정제 방안

□ 서울특별시 생활인구 데이터 전처리

서울특별시 행정동별 서울생활인구 데이터 내 행정동 코드와 code_map 내 행정동 코드를 매치하여 법정동별 생활인구를 산출. 주간(7시~19시)/야간(20시~6시)을 구분하여 법정동별 주야별 생활인구 산출.

□ 주민등록인구 전처리

서울특별시 주민등록인구(동별) 통계 데이터의동이 행정동별로 나뉘어 있으므로 code_map 데이터를 이용하여 행정동을 법정동으로 변환하여 법정동별 주민등록인구 수를 산출.

□ 동별 평균연령 전처리

서울특별시 평균연령(동별) 통계 데이터와 code_map 데이터를 활용하여 법정동별 평균연령 산출.

□ 법정동 면적 전처리

법정동(읍면동단위) 경계도면을 나타낸 행정구역 읍면동(법정동) shp파일을 활용하여 법정동별 면적 산출.

□ 코드 데이터 전처리

code_map 데이터에서 서울특별시만 추출 후, 구 단위까지만 있는 결측치는 제거. 행정동 코드와 법정동 코드를 활용하여 행정동 당 법정동 개수 산출.

3. 분석 프로세스

3.1 분석 프로세스

3.1.1 데이터 수집

□ 바닥형 보행신호등 위치 데이터

바닥형 보행신호등의 현황을 파악하고 반영하기 위해 공공데이터포털에서 '바닥형 보행신호등 위치 데이터'를 수집했다. 기존에 바닥형 보행신호등이 설치된 횡단보도의 도로명주소, 위도, 경도 값을 포함하고 있다.

□ 보행자 교통사고 데이터

서울시의 법정동별 보행자의 교통사고 건수를 파악해, 바닥형 보행신호등이 필요한 타당성을 설명하는 데 사용할 보행자 교통사고 데이터를 수집했다. TAAS(교통사고분석시스템)에서 필요한 데이터만 얻기 위해 사고일시를 2018년부터 2021년까지 4개년으로, 사고유형을 차 대 사람으로 설정했다.

□ 서울특별시 횡단보도 위치 및 부착대 정보

최종적으로 바닥형 보행신호등을 설치할 횡단보도를 선정하기 위해, 횡단보도 현황을 파악할 수 있는 해당 shape 파일을 서울열린데이터광장에서 수집했다.

□ 서울특별시 자동차 등록현황

바닥형 보행신호등 설치 최적 입지를 선정하는 데 반영할 변수 중 하나로 사용하기 위해, 서울열린데이터광장에서 해당 데이터를 수집했다.

□ 서울특별시 교차로 관련 정보

바닥형 보행신호등을 설치할 최적 입지 선정의 변수 중 하나로, 해당 데이터를 서울열린데이터광장에서 수집했다.

□ 지역별 교통안전지수

바닥형 보행신호등을 설치할 최적의 입지를 선정하는 데 변수로 사용하기 위해, TAAS(교통사고분석시스템)에서 데이터를 수집했다.

□ 서울특별시 행정동별 서울생활인구

바닥형 보행신호등 설치 최적 입지 선정에 사용할 변수로 해당 데이터를 서울 열린데이터광장에서 수집했다. 장기체류외국인과 내국인에 대한 데이터를 대상으로 선정했다.

□ 서울특별시 주민등록인구(동별) 통계

바닥형 보행신호등 설치 최적 입지 선정에 사용하기 위한 변수로, 행정안전부에서 제공한 해당 데이터를 공공데이터포털에서 수집했다.

□ 서울특별시 평균연령(동별) 통계

서울 열린데이터광장에서 해당 데이터를 수집했으며, 바닥형 보행신호등 설치 최적 입지 선정을 위한 변수로 사용할 것이다.

□ code_map

데이터들의 주소 단위를 법정동으로 통일하기 위해 사용되는 또 다른 데이터로, KDX 한국데이터거래소에서 수집했다. 행정동과 법정동 코드 등의 정보를 포함하고 있다.

□ 행정구역코드

데이터들의 주소 단위를 법정동으로 통일하기 위해 통계청에서 해당 데이터를 수집했다. 행정구역 코드와 법정동 코드, 행정동과 법정동명을 포함하고 있어, 다른 단위의 주소들을 해당 데이터와 매치하면 법정동을 추출해 주소 단위를 통일할 수 있다.

□ 행정구역 읍면동(법정동) shp파일

법정동별 면적을 변수로 사용하기 위해 국토교통부의 국가공간정보포털 오픈마켓에서 해당 shape 파일을 수집했다.

3.1.2 데이터 전처리

□ 보행자 교통사고 데이터

원본 데이터인 보행자 교통사고 데이터를, 동별 주간/야간 교통사고 횟수, 야간사고 비율, 야간사고위험도 변수를 포함하는 데이터로 저장하기 위해 전처리 과정을 수행한다.

보행자 교통사고 데이터의 시군구 주소 변수에서 동만 추출해 변수로 저장한 후, 사고일시 변수가 20시부터 06시까지면 야간으로, 07시부터 19시까지면 주간으로 나타내는 주야 구분 변수를 생성한다. 동과 주야 구분 변수를 기준으로, 동별 주간/야간 교통사고 횟수 합계를 구한다.

해당 과정 후 데이터를 확인하면, NA 값이 존재한다는 것을 파악할 수 있다. NA 값으로 존재하는 해당 데이터는 주간 혹은 야간에 교통사고가 없었다는 것이므로, 각 NA 값을 0으로 대체한다. 이렇게 구해진 주간/야간 교통사고 횟수를 기반으로, 전체 사고 중 야간사고 비율을 계산한다.

야간사고 비율을 A, B, C그룹으로 나눠 범주형 변수인 야간사고 위험도 변수를 생성한다. 먼저, 주간사고 횟수가 너무 적어 야간사고 비율이 높게 책정되는 경우, 사고가 얼마 나지 않는 곳의 설치 필요도가 높게 나타나므로 주간사고가 10건 이상이 되어야 한다는 조건을 1차 조건으로 각각 A그룹과 B그룹에 걸어준다. 주간사고가 10건 이상이 되지 않으면 야간사고 비율이 높아도 C그룹으로 들어가게 된다. 1차 조건을 통과한 데이터 내에서, A그룹은 야간사고 비율이 37% 이상, B그룹은 25% 이상인 조건이고 나머지는 C그룹이다. 해당 기준 퍼센트는 각각 약 quantile 75%, quantile 25% 값이다. 이때, 해당 결과 A그룹에 38개, B그룹에 82개, C그룹에 299개의 동이 나뉜다.

최종적으로 법정동, 주간사고 횟수, 야간사고 횟수, 야간사고 비율, 야간사고위험도 변수가 포함된, 전처리가 완료된 데이터를 '법정동별_교통사고횟수' 파일로 저장한다.

□ code_map

code_map은 전국의 시도와 자치구, 군, 시, 행정구역명, 법정동명, 행정구역 코드, 법정동 코드 정보를 포함하고 있는 데이터이다. 필요 이상의 정보를 포함하고 있어, 분석에 알맞게 전처리 과정을 수행한다.

서울특별시의 정보만 필터링한 후 법정동까지 모든 정보가 서울특별시나 자치구로 같은 정보가 채워져 있는 행은 삭제하여 필요한 부분만 남긴다. 행정동 코드와 법정동 코드가 현행은 8자리인 것에 비해 해당 데이터는 10자리로 입력되어 있다. 현행 코드 뒤에 '00'이 붙어있어 그렇다는 것을 파악하고, 행정동 코드와 법정동 코드의 모든 값에 각각 100을 나눠주어 현행 코드에 맞게 변환한다. 그 후 나중에 행정동 데이터를 법정동으로 단위를 맞추는 데 사용하기 위해, 하나의 행정동에 포함되는 법정동의 개수를 나타내는 변수를 추가한다.

마지막으로 구, 행정동명, 법정동명, 행정동 코드, 법정동 코드, 법정동 개수 변수만 추출하여 해당 데이터를 'code_map' 파일로 저장한다.

□ 서울특별시 행정동별 서울생활인구

생활인구 데이터는 내국인과 장기체류외국인별, 1월부터 12월까지 월 단위로 각각 12개의 데이터로 존재한다. 총 24개의 데이터를 하나로 합치고 필요한 형태로 만들기 위해 전처리를 수행한다.

따로 존재하는 총 24개의 데이터를 불러온 후 하나의 데이터로 병합한다. 모두 변수명과 순서가 똑같이 저장되어 있으므로 행으로 붙인다. 생활인구 데이터는 기준일, 시간대, 행정동 코드, 총생활인구 수 변수와 성별 나이대별 생활인구 변수가 각각 저장되어 있다. 이중 필요한 변수인 기준일, 시간대, 행정동 코드, 총생활인구 수 변수만 추출한다. 이후에 주야별 생활인구 수를 구하기 위해 시간대 변수를 기준으로 20시부터 06시까지는 야간으로, 07시부터 19시까지는 주간으로 나타내는 주야 구분 변수를 생성한다.

최종적으로 기준일, 시간대, 행정동 코드, 총생활인구 수, 주야 구분 변수를 가진 데이터를 '생활인구(수정)' 파일로 저장한다.

□ 생활인구(수정) + code_map

1차 전처리를 완료한 code_map과 생활인구(수정) 데이터를 활용하여, 최종으로 사용할 법정동별 주야별 생활인구 데이터를 생성하는 전처리 과정을 진행한다.

code_map과 생활인구 데이터를 불러온 후 행정동 코드를 기준으로 생활인구 데이터에 code_map 데이터를 병합한다. 생활인구 데이터는 행정동 단위로, code_map 데이터는 법정동 단위로 저장되어 있어 행정동 코드를 기준으로 병합하면 생활인구 데이터의 하나의 값에 여러 개의 code_map 데이터값이 매칭된다. 생활인구 데이터의 하나의 행정동에 대한 값이 포함되는 모든 법정동에 대하여 해당 개수만큼의 행으로 중복되어 나타나는 것을 확인할 수 있다.

행정동 단위로 저장되어 있던 생활인구 데이터를 법정동 단위로 저장하기 위해 보정된 생활인구 수 변수를 추가한다. 행정동의 총생활인구 수 값을 포함한 법정동에 각각 나누어 저장하기 위해 보정된 생활인구 수는 행정동별 총생활인구 수에 행정동에 포함되는 법정동의 개수를 나눠 계산한다. 최종적으로 필요한 법정동별 주야별 평균 생활인구 수 데이터를 구하기 위해 법정동과 주야별로 보정된 생활인구 수를 더한 후 365로 나누어준다. 데이터가 1년 치 데이터이므로 평균을 구하기 위해 365로 나눈다.

그렇게 구해진 법정동, 주야 구분, 평균 보정된 생활인구 수 변수를 포함하는 데이터를 최종 '법정동별_주야별_생활인구' 파일로 저장하며, 해당 데이터의 전처리를 완료한다.

□ 서울시 자동차 등록현황 + code_map

행정동 주소, 자동차 연료별 등록 대수, 등록 대수 총합계 정보를 포함하고 있는 서울시 자동차 등록현황 데이터를 법정동별 자동차 등록 대수 데이터로 정리하기 위해 전처리를 수행한다.

서울시 자동차 등록현황 데이터를 불러온 후 필요한 변수인 행정동 주소와 총합계만 뽑아낸다. 뽑아낸 행정동 주소 변수에서 동만 추출해 따로 동 변수로 저장한다. 그 후 code_map 데이터를 불러와 행정동을 기준으로 자동차 등록현황 데이터에 code_map 데이터를 병합하여 법정동 값을 넣어준다.

자동차 등록현황 데이터도 행정동 단위로 저장되어 있으므로 법정동 단위로 나타내기 위해 보정된 값이 필요하다. 하나의 행정동에 속하는 여러 개의 법정동에 자동차 등록 대수를 나눠 저장하기 위해 총합계를 법정동 개수로 나눈 보정 합계를 구한다. 이를 통해 법정동별 자동차 등록 대수를 얻을 수 있다. 최종적으로 법정동과 보정 합계 값을 가지고 있는 데이터를 '법정동자동차수' 파일로 저장한다.

□ 서울특별시 횡단보도 위치 및 부착대 정보 + 행정구역코드 + code_map

해당 데이터는 횡단보도 위치의 경위도 값을 저장하고 있는 shape 파일이다. Q-Gis로 해당 데이터를 불러온 후 행정동 구역이 나타나는 shape 파일도 함께 불러온다. 두 레이어를 겹쳐서 행정동별 횡단보도 개수를 카운트한다.

해당 데이터의 횡단보도는 폴리곤으로 저장되어 있어 면적으로 나타난다. 이 경우 행정동의 경계에 걸치는 횡단보도는 중복되어 카운트된다. 이러한 오류를 방지하고자 횡단보도의 중심점을 찾아 폴리곤 값을 point 값으로 바꾼 후 각 행정동 면적 내의 횡단보도 개수를 카운트한다. 해당 카운트 값을 횡단보도 위치 정보 레이어의 테이블에 추가한 후, csv 파일로 저장한 후, 파이썬으로 필요한 변수인 행정구역 코드와 행정동명, 횡단보도 개수만 추출하여 다시 저장한다.

횡단보도 데이터를 법정동 단위로 변환하기 위해 code_map 데이터와 병합해야 하는데 횡단보도 데이터는 행정동 코드가 아닌 행정구역 코드만 나와 있다. 이에, 중간 연결 데이터로 행정구역 코드 데이터와 병합하여 행정동 코드 정보도 추가한다. 행정구역코드 데이터의 시도를 서울특별시로 필터링해 서울시의 정보만 추출한다. 그 후 행정구역 코드를 기준으로 횡단보도 데이터에 행정구역코드 데이터를 병합한다. 행정동 코드 정보 생긴 횡단보도 데이터에 행정동 코드를 기준으로 code_map 데이터를 병합한다. 행정동 단위의 값을 법정동 단위로 나누기 위해 보정된 횡단보도 개수가 필요하다. 보정된 횡단보도 개수는 행정동 단위로 측정된 횡단보도 개수에 하나의 행정동에 속하는 법정동의 개수를 나누어 구한다. 그 후 법정동을 기준으로 보정된 횡단보도 개수를 합하면 법정동별 횡단보도 개수를 알 수 있다. 최종적으로 법정동과 보정 횡단보도 개수 변수만 추출하여 '법정동별_횡단보도수' 파일로 저장한다.

□ 서울시 교차로 관련 정보 + code_map

서울시 교차로 관련 정보 데이터는 구 코드와 동 코드가 변수로 저장되어 있다. 해당 코드들은 code_map의 행정동 코드나 법정동 코드 어떤 것과도 매치할 수 없다. 그러므로 매칭될 수 있는 새로운 수정된 동 코드 변수를 만들어 준다. 수정된 동 코드는 원래의 구 코드 뒤에 동 코드를 이어 쓴 값으로 구한다. 병합할 code_map 데이터에도 수정된 동 코드 변수를 만들어 줘야 한다. 구 이름과 구 코드를 가지는 데이터프레임을 생성한 후 code_map 데이터에 구 이름을 기준으로 병합해 구 코드 변수를 추가한다. 기존의 법정동 코드를 1000으로 나눈 나머지에 100을 곱해 임의의 동 코드를 만든 후 구 코드 뒤에 해당 동 코드를 이어 쓰는 수정된 동 코드 변수를 생성한다. 동 코드를 기준으로 서울시 교차로 관련 정보 데이터에 code_map 데이터를 병합하면 법정동별 교차로 개수를 구할 수 있다. 그렇게 병합된 데이터의 법정동과 교차로 개수 변수만 추출하여 최종 '법정동별_교차로수' 파일로 저장한다.

□ 지역별 교통안전지수 + code_map

교통안전 지수는 동별로 측정되지 않고 구별로 측정되어 자치구 명과 교통안전 지수 변수를 포함하고 있다. 구별 단위의 데이터를 법정동 단위로 변환하기 위해 code_map 데이터를 함께 불러온다. 자치구 명을 기준으로 교통안전지수 데이터에 code_map 데이터를 병합한다. 해당 자치구에 속하는 모든 법정동에 해당 자치구의 교통안전 지수를 줄 것이므로 추가 변경사항 없이 법정동과 교통안전 지수 변수만 추출하여 '법정동별_교통안전지수' 파일로 저장한다.

□ 서울시 주민등록인구(동별) 통계 + code_map

서울시 주민등록인구 데이터는 행정동별로 한국인과 외국인 등록인구가 구해져 있다. 따로 구해져 있는 한국인과 외국인 등록 수를 더해 총주민등록인구를 구한다. 그 후에 행정동명을 기준으로 주민등록인구 데이터에 code_map 데이터를 병

합해 법정동 정보를 넣어준다. 행정동 단위로 측정된 수를 법정동으로 나누기 위해 보정된 주민등록인구 수가 필요하다. 총주민등록인구 수에 하나의 행정동에 속하는 법정동의 개수를 나눈 값으로 보정된 주민등록인구 수 변수를 추가한다. 최종적으로 필요한 법정동명과 보정된 주민등록 수 변수만 추출하여 '법정동별_주민등록수' 파일로 저장한다.

□ 서울시 평균연령(동별) 통계 + code_map

행정동별로 저장되어 있는 서울시 평균연령 데이터를 법정동 단위로 바꾸기 위해 code_map 데이터를 불러온다. 행정동명을 기준으로 평균연령 데이터에 code_map 데이터를 병합하여 법정동을 기준으로 평균연령 값의 평균을 구한다. 법정동명과 평균연령 변수만 추출하여 '법정동명_평균연령' 파일로 저장한다.

□ 행정구역 읍면동(법정동) shp 파일

법정동의 면적을 구하기 위해 법정동 shape 파일을 Q-Gis에서 레이어로 불러온다. 레이어의 속성 테이블에서 필드 계산기 기능을 사용해, 각 법정동의 면적 값을 가지는 열을 추가한다. 그 후 csv 파일로 내보내기 기능을 사용하여 해당 테이블을 csv 파일로 저장한다. 사용할 변수인 법정동 코드, 법정동명, 면적 변수만 추출하여 '법정동_면적' 파일로 저장한다.

□ 전처리 완료 데이터들 결합

각각 전처리가 완료된 데이터들을 법정동을 기준으로 모두 하나의 데이터로 병합한다. 이때 교통사고 데이터를 병합하면, 몇 개의 동에 대해 NA 값이 생기는 것을 확인할 수 있다. 이것은 해당 법정동에는 교통사고가 발생하지 않은 경우로 그런 경우 NA 값을 0으로 대체해준다. NA 처리까지 완료한 후 최종적으로 사용할 변수들만 추출해 최종 데이터로 저장하며 모든 전처리 과정을 완료한다.

3.1.3 EDA

□ 분류 클래스 산정

전처리 과정을 완료 후 얻은 데이터 중 기존에 바닥형 보행신호등이 깔린 동들을 추출해 저장한 후 해당 데이터의 교통사고 데이터를 활용하여 최적 입지 순위 그룹 변수를 구한다. 최적 입지 순위 그룹은 A, B, C 총 세 개의 그룹으로 나눈다. 먼저, A그룹의 조건은 바닥형 보행신호등 설치 전의 교통사고 건수가 전체 데이터의 중앙값보다 크며 해당 조건을 충족하는 데이터 중에서 quantile 80%의 값보다 크다는 것이다. B그룹은 두 가지 조건으로 나눌 수 있다. 한 조건은 바닥형 보행신호등 설치 전의 교통사고 건수가 전체의 중앙값보다 크며 해당 데이터의 quantile 30% 이상의 값일 때이다. 다른 조건은 설치 전의 교통사고 건수가 전체의 중앙값보다 작고 해당 조건을 충족하는 데이터 중 quantile 80% 이상의 값일 경우이다. C그룹은 설치 전의 교통사고 수가 중앙값을 넘지만 quantile 30% 값보다 작은 경우 설치 전의 교통사고 수가 중앙값을 넘지 못하고 quantile 80% 값보다 작은 경우이다. 이렇게 구해진 최적 입지 순위 그룹은 후에 모델링을 통해 분류할 세 개의 클래스가 된다.

□ SMOTE

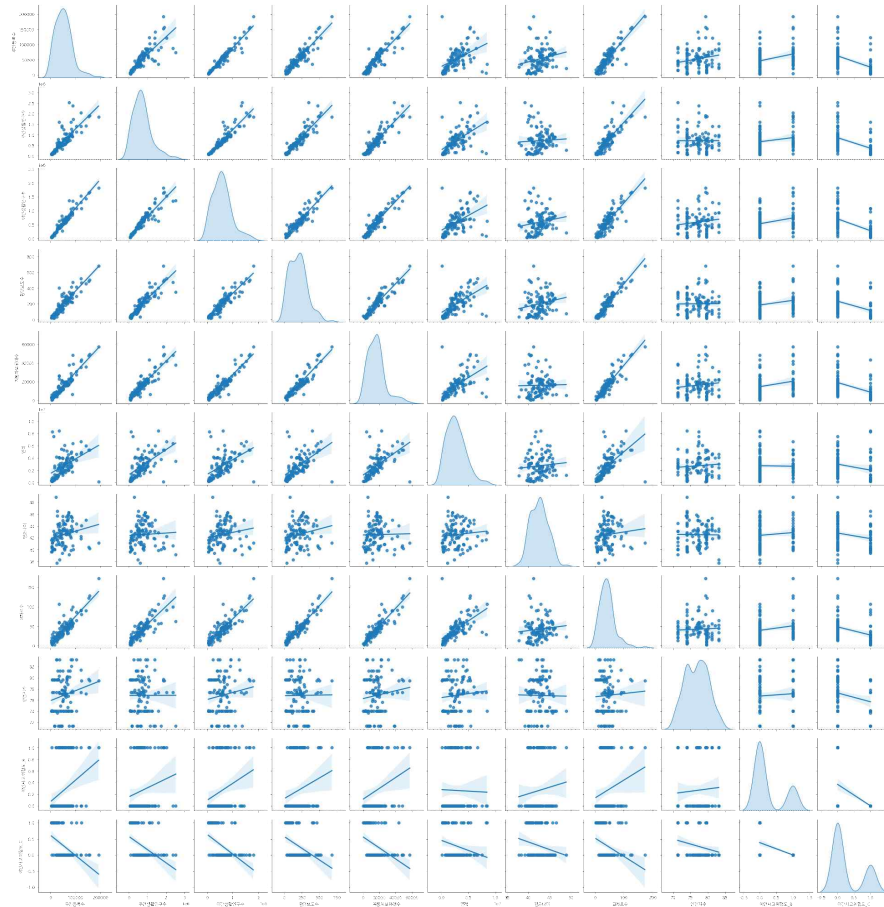
바닥형 보행신호등이 설치된 법정동에 대한 데이터는 총 70개의 동에 대한 정보만 있다. 데이터의 양이 부족하고 데이터 불균형 문제로 인해 신뢰할만한 모델이 생성되기 어려울 것이다. 따라서 데이터 불균형 문제를 해소함과 동시에 데이터의 양이 늘어나는 효과를 위해 오버샘플링을 진행하였다. 단순 무작위 추출을 통한 오버 샘플링의 경우 데이터를 단순히 복사해 과적합의 문제가 발생할 수 있다. 이를 방지하기 위해 알고리즘을 기반으로 추가 데이터를 생성하는 SMOTE 방식을 사용한다. SMOTE 과정을 통해 70개였던 데이터가 약 120개에 근접하게 오버 샘플링된 것을 확인할 수 있다. SMOTE를 통해 데이터 수 증가를 확인했으므로 각 A, B, C 클래스가 균일하게 잘 늘어났는지 확인하기 위해 시각화해본다. 해당 데이터는 약 10개의 변수를 저장하고 있어 시각화가 쉽지 않으므로 시각화는 PCA 방식과 t-SNE 방식 두 가지를 사용해 차원을 축소하여 진행한다.

□ 제곱근, 로그 변환

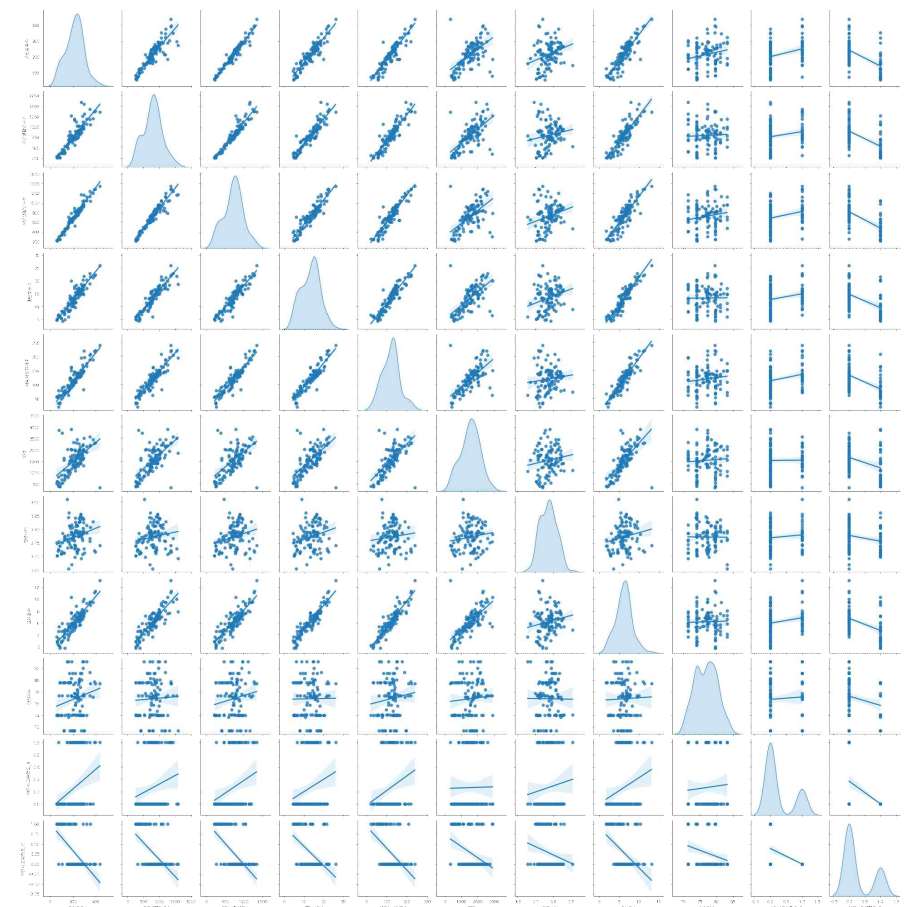
최종적으로 모델링에 사용할 데이터 중 X 변수로 사용될 변수 중 연속형 변수에 대해 왜도를 확인한다. 대부분 변수가 0에서 벗어나 있는 것을 파악할 수 있다. 왜도가 0에 가까울수록 좋은 데이터이므로 변수들의 변환 작업을 진행한다. 각각의 변수들에 대해 제곱근 변환과 로그 변환 중 어떤 방법이 적합할지 확인하기 위해 원래의 왜도 값, 제곱근 변환 후 왜도 값, 로그 변환 후 왜도 값을 출력해본다. 각각의 변수의 왜도 값을 확인해보고 가장 0에 가까운 값이 나오는 방법을 선택해 변환을 진행한다. 안전 지수 변수의 경우 원래 값의 왜도 값이 0에 가까운 좋은 정도이므로 따로 변환을 진행하지 않는다. 평균나이 변수의 경우 로그 변환을 한 후의 왜도 값이 가장 좋으므로 로그 변환을 진행한다. 나머지 주민등록 수, 주간 생활인구 수, 야간 생활인구 수, 횡단보도 수, 자동차 보유 대수, 면적, 교차로 수 변수들은 제곱근 변환 과정을 거친 왜도 값이 더 좋게 나타나므로 제곱근 변환해준다. 이러한 변환 과정을 통해 연속형 X 변수들의 왜도 값이 0에 가까운 적절한 수준이 되며 모델링의 성능을 높일 수 있다.

□ Robust Scaler (표준 정규화)

모델링에 사용하기 위해 수집 및 전처리하여 산출한 데이터는 각 변수 간의 단위 및 범위 차이가 크다. 어떤 변수는 최솟값이 0이고 최댓값이 약 170 정도인 것에 비해, 어떤 변수는 최솟값이 10,000이 넘고 최댓값은 3,000,000이 넘는 것도 있다. 이처럼 범위(단위)의 차이가 크므로 향후 모델 성능의 신뢰도와 안정성이 떨어질 것을 방지하기 위해 스케일링을 통해 변수들의 범위를 맞춰준다. 스케일링 기법은 Robust Scaler로 진행한다. Robust Scaler는 평균과 분산을 사용하는 기본 스케일링 기법과 달리 중앙값과 사분위수 범위를 사용하여 스케일링을 진행해 아웃라이어(이상치)의 영향을 최소화한다. 그러나 아웃라이어의 영향을 최소화한다고 해도 아웃라이어가 너무 많으면 스케일링 후에도 목표하는 분포에 근사하지 않을 수 있다는 문제가 존재한다. 즉 범위만 달라지는 것이 아닌 분포도 달라질 수 있다. 해당 문제를 인식하고 스케일링 전후 데이터의 분포도를 시각화해본다. 시각화 결과를 통해 스케일링을 진행한 후에도 분포는 일정하고 단위만 맞춰진 것을 확인할 수 있다.



[그림 3-1] 기존 데이터 분포



[그림 3-2] 로그, 제곱근 변환 후 분포 (스케일링 전 분포)

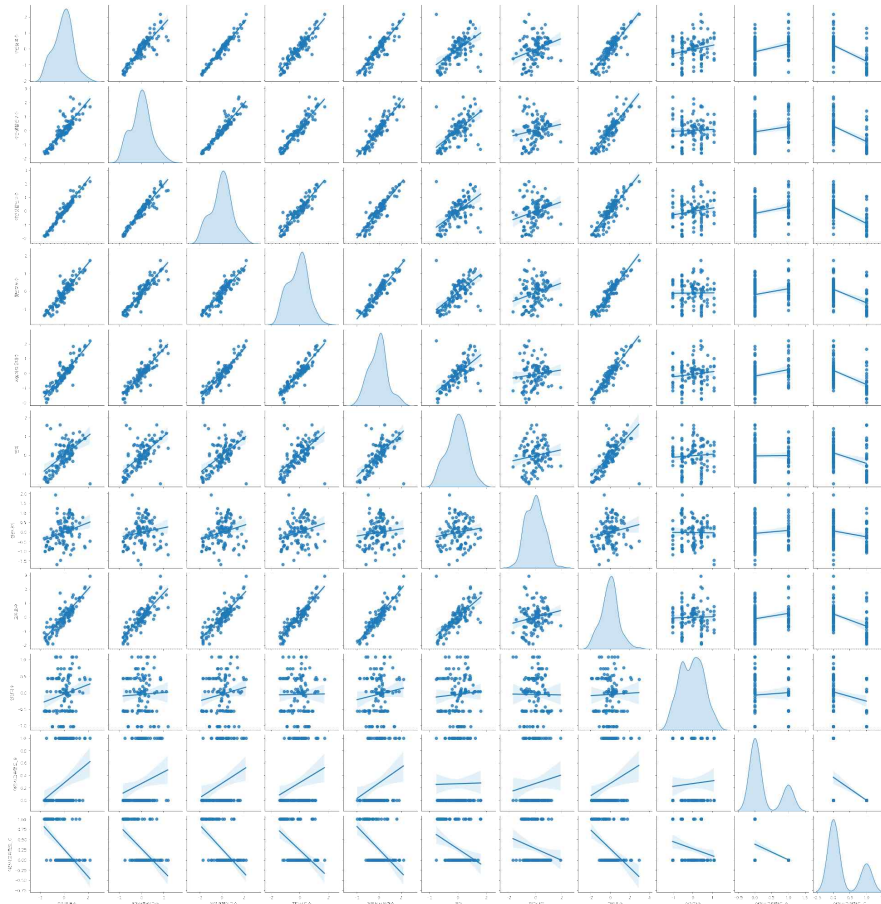
□ PCA(Principal Component Analysis : 주성분 분석)

먼저 PCA를 통한 시각화로 SMOTE 결과를 확인한다. PCA는 주성분 분석으로 고차원의 데이터 집합이 주어지면 원래의 고차원 데이터와 가장 비슷하면서 차원은 더 낮은 데이터를 찾아내는 방식이다. 즉, 변수의 특성을 살리면서 차원을 축소하는 방식이다.

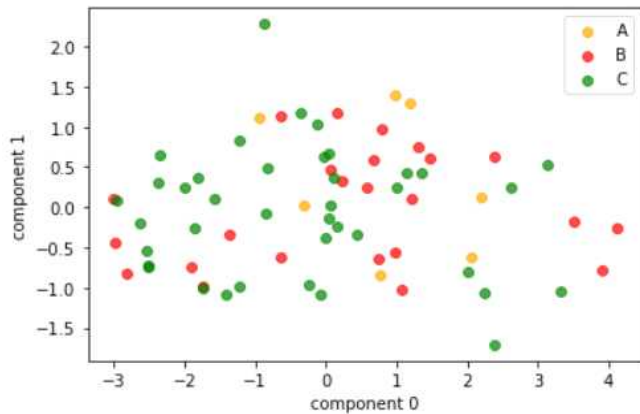
SMOTE를 진행하기 전의 원래의 데이터에 대해 진행한다. 차원을 2차원으로 축소를 PCA 모델을 만든 후 분류 클래스로 사용되는 최적 입지 순위 그룹 변수를 제외한 변수들을 학습시킨다. 그런 후 PCA 모델의 설명력을 확인해 보면 제1 주성분과 제2 주성분의 합이 약 76%인 것을 확인할 수 있다. 주성분들의 총설명력이 70%~80% 사이에 있으므로 해당 PCA 모델을 사용하여 2차원 축소를 진행한다. PCA 모델에 변수들을 적합 시켜 나온 제1 주성분과 제2 주성분 값을 데이터프레임으로 저장한 후 최적 입지 순위 그룹값을 추가해 저장해준다. 그런 후 최적 입지 순위 그룹값 별로 데이터를 나눈 후 시각화해 A, B, C 그룹의 2차원 평면상 분포를 확인한다.

SMOTE를 진행해 오버 샘플링된 데이터를 불러온다. 앞서 기존 데이터를 적합 시켜 만들어 둔 PCA 모델에 넣어 얻어지는 제1 주성분과 제2 주성분 값을 데이터프레임으로 저장한 후 전과 같이 최적 입지 그룹값을 추가해준다. 이전과 마찬가지로 해당 데이터를 최적 입지 그룹값 별로 나눈 후 시각화를 진행해 2차원 평면상에 A, B, C 그룹의 분포가 어떤지 확인한다.

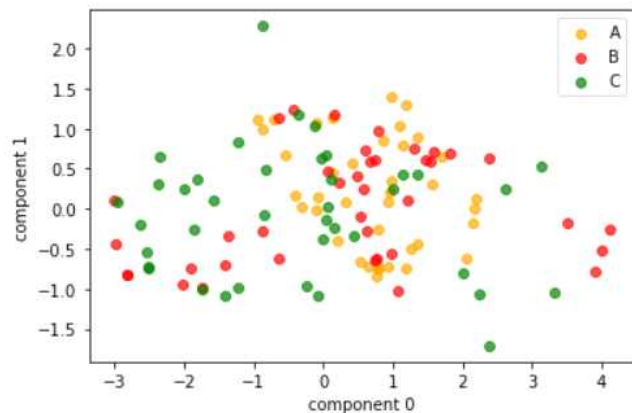
해당 시각화를 통해 확인한 결과, SMOTE를 통한 오버 샘플링으로 데이터가 과적합 없이 기존의 각 그룹에 대해 적절하게 잘 진행된 것을 확인할 수 있다.



[그림 3-3] 스케일링 후 분포



[그림 3-4] 기존 데이터 2차원 PCA



[그림 3-5] SMOTE 후 데이터 2차원 PCA

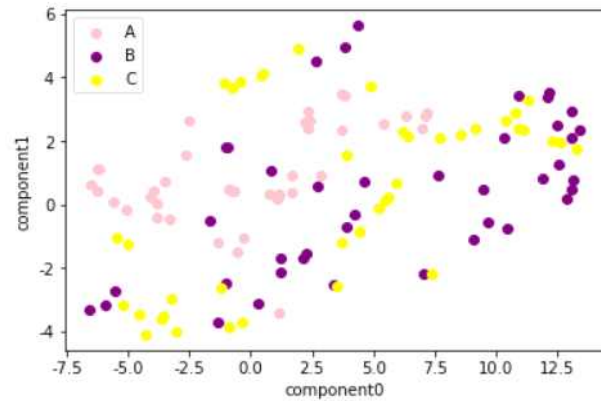
□ t-SNE (t-distributed stochastic neighbor embedding)

다음으로 t-SNE를 활용하여 시각화를 진행해, SMOTE 결과를 확인해 본다. t-SNE는 고차원의 복잡한 데이터를 차원을 축소해, 낮은 차원의 공간으로 나타내는 것이다. 고차원의 데이터를 시각화하기 위해 2차원 혹은 3차원으로 차원을 축소하는 데 사용한다. 변수들의 특성을 살려서 차원을 축소하는 PCA와 달리 t-SNE는 데이터 구조를 살려서 차원을 축소한다. 고차원의 데이터에서 비슷했던 데이터 구조는 낮은 차원에서 가깝게 대응하고 비슷하지 않았던 데이터 구조는 멀리 떨어져 대응한다. 즉, PCA의 변수들 특성을 살리는 것에 반해 변수 간의 거리를 살리면서 차원의 축소가 이루어진다.

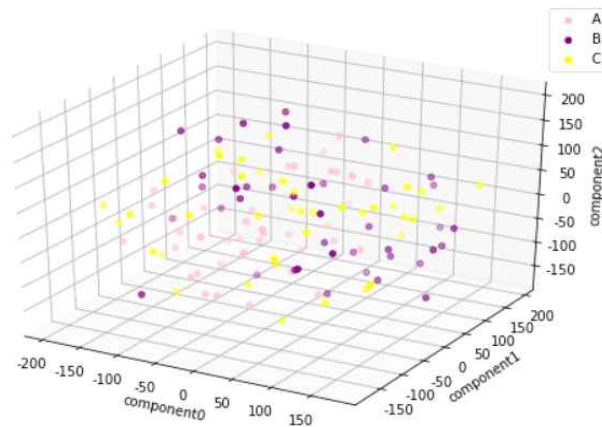
먼저 2차원으로 차원을 축소하는 t-SNE 모델을 생성한다. 그런 후 SMOTE를 통해 오버 샘플링이 완료된 데이터 중 최적 입지 순위 그룹을 제외한 변수들을 학습시키고 적합 시켜 나온 2차원 공간 값을 데이터프레임으로 저장한다. 해당 데이터에 원래의 최적 입지 순위 그룹값을 추가해준 후 해당 값을 기준으로 데이터를 나누는 후 시각화를 진행해 A, B, C그룹의 2차원 평면상에서의 분포를 확인한다.

3차원상에서의 분포도 시각화해 확인해 본다. 3차원으로 차원을 축소하는 t-SNE 모델을 생성한 수, 전과 같이 SMOTE를 완료한 데이터의 최적 입지 순위 그룹을 제외한 변수들을 학습시킨다. 이번에는 총 3차원의 공간 값이 나오는데 이것을 데이터프레임으로 저장하고 최적 입지 순위 그룹값을 추가해준다. 3차원상의 그림을 시각화하기 위해 3차원 그래프를 먼저 세팅해준다. 그런 후 앞에서 추가한 최적 입지 순위 그룹값을 기준으로 데이터를 나누고 시각화를 진행한다. 3차원상에 A, B, C 그룹의 분포가 나타나는 것을 확인한다.

2차원 평면과 3차원상의 분포에서 모두 A, B, C그룹에 대해 균일하게 오버 샘플링된 것을 확인할 수 있다.



[그림 3-6] SMOTE 후 데이터 2차원 t-SNE



[그림 3-7] SMOTE 후 데이터 3차원 t-SNE

3.1.4 모델링

□ 5가지 모델을 통한 최적 입지 동 산정

이번 분석에서 모델링의 목표는 정해진 클래스를 정확하게 분류하는 것이 아닌 정해진 클래스가 없는 데이터를 유사하게 분류하는 것이다. 기존에 라벨값이 부여된 데이터를 통해 라벨값이 없는 데이터들을 가장 유사한 그룹의 라벨값을 갖도록 모델링한다. 그러므로 train과 test 데이터로 나누지 않고 모든 데이터로 모델링을 진행해 최적 입지 순위 그룹을 분류하는 방향으로 진행한다.

해당 방향으로 진행할 경우 train과 test 데이터를 활용해 모델의 정확성을 검증하는 과정을 수행할 수 없다. 이를 보완하기 위해 여러 가지의 모델을 사용해 모든 모델이 설치 최적 입지 그룹이라고 판단한 것들을 채택하는 방법으로 진행한다. 모델링을 수행할 모델들은 'Random Forest', 'XGBoost Classifier', 'Gaussian Naive Bayes', '다중 Logistic Regression', 'SVC' 다섯 가지로 선정한다. 다양한 분류 모델을 사용하여 모델의 다양성을 높이고, 산출되는 결과의 타당성을 높인다.

3.2 분석 내용 및 방법

3.2.1 트리 기반 모델

트리 기반의 의사결정 모델은 Feature Space(특징 공간)를 여러 개의 영역으로 나누기 위해 동작하는 알고리즘이다. Feature Space를 회귀 또는 분류 기반으로 예측을 진행한다. 노드의 깊이가 너무 크지 않는 이상 해석도 쉬운 간단한 모델 중 하나이다. 변수들이 비선형적인 관계에서 잘 작동한다.

분류를 목적으로 사용하면 불순도를 이용해 분산을 감소시키며 노드를 분리한다. 이때 불순도를 감소시키는 정도에 따라 변수의 중요도를 책정하는데 그렇게 책정된 중요도가 가장 큰 변수들을 찾아내 알고리즘을 진행한다. 불순도를 기준으로 얻을 수 있는 변수의 중요도는 train과 test 데이터를 나누어 진행할 경우 train 데이터를 기반으로 얻어지는 중요도이므로 test 데이터에 대해서는 변수의 중요도가 어떻게 작용할지 보장할 수 없다는 문제가 존재한다. train 데이터에서는 중요하다고 여겨졌던 변수가 test 데이터에서는 그렇지 않을 수 있고 반대로 train 데이터에서는 중요하지 않았던 변수가 test 데이터에서는 중요할 수 있다.

트리 기반 모델의 이러한 특징이 이번 분석의 방향에 적합하다고 생각하여 해당 모델을 선정하게 되었다. train과 test 데이터로 나누지 않고 모델링을 진행하는 것에 좋은 결과를 가져올 것이라 기대한다.

□ Random Forest

트리 기반의 지도 학습 알고리즘이다. 정확성, 단순성, 유연성 덕에 가장 많이 사용되는 알고리즘 중 하나이다. 여러 질문을 거쳐 의사결정을 하는 의사결정트리를 여러 번 수행하는 의사결정트리 그룹이라 말할 수 있다. 기능을 무작위로 선택해 의사결정 트리의 숲(포레스트)을 만든 결과를 평균화한다.

주어진 데이터들의 최적 입지 순위 그룹을 A, B, C로 분류하기 위해 분류에 흔히 사용되고 높은 정확도의 결과를 얻을 수 있는 해당 모델을 선정했다.

GridSearch를 활용해 최적의 상태에서 학습해 결과를 산출할 수 있도록 한다. 모델링을 진행할 때는 여러 파라미터를 지정해야 하는데 GridSearch는 하이퍼

파라미터를 차례대로 입력해 학습하고 측정하면서 가장 좋은 파라미터를 알려준다. 우리의 모델에서는 n_estimators(결정트리의 개수)는 50개, max_depth(트리의 최대 깊이)는 11, min_samples_split(리프노드가 가질 최소의 샘플 데이터 수)은 2개가 가장 적절하다는 결과를 얻어 해당 파라미터로 모델링을 수행한다. 모델링을 진행한 후에는 Confusion table(혼동행렬)을 출력해 분류 예측값도 확인해 본다.

□ XGBoost Classifier

트리 기반의 알고리즘 앙상블 학습에서 주목받는 알고리즘 중 하나로 부스팅 기법의 한 종류다. 부스팅 알고리즘은 이전 모델의 오류를 차례대로 보완해나가는 방식으로 모델링을 진행한다. 예측 성능이 높다고 평가되는 GBM(Gradient Boosting Algorithm)에 기반하며 GBM의 단점을 보완했다. XGB classifier가 보완한 GBM의 대표적인 단점은 느린 수행시간과 과적합 등이다. 이에 따라 XGB는 빠른 수행시간으로 뛰어난 예측 성능을 가지며 과적합을 규제한다. 효과가 비교적 적은 분할은 가지치기해 불필요한 분할 수를 줄인다. 또한 모델 스스로 교차검증을 수행해 최적화된 반복 횟수로 진행한다. 반복 횟수를 지정하더라도 스스로 교차검증을 통해 최적화된 값이 산출됐다고 판단하면 지정된 반복 횟수에 미치지 못했어도 중간에 스스로 멈춘다.

XGB의 이러한 특징과 장점들이 좋은 분류 값을 구해줄 것이라 기대하여 해당 모델을 선정하였다.

XGB 역시 GridSearch를 통해 결정트리의 개수는 50개, 트리의 최대 깊이는 5, 그리고 subsample이라는 과적합을 규제하기 위한 조건인 각 트리마다의 데이터 샘플링 비율은 1이 가장 최적이라는 것을 산출해 모델에 적용한다.

3.2.2 확률 기반 모델

확률 기반 모델은 설명변수와 종속변수가 가지는 특정 확률에 근거한 관계를 사용한다. 베이지 정리에 기반을 두고 있는 모델인데 베이지 정리는 한 사건이 발생함으로써 인해 다른 사건의 확률이 어떻게 변하는지를 표현한 것이다. 그동안의 확률이 연역적 추론에 기반하는 것에 비해 베이지 정리는 귀납적(경험적) 추론에 기반한다. 이미 알고 있거나 직관적으로 추론이 가능한 데이터에서 샘플링 하는 연역적 방식과 달리 데이터로부터 특정 단서를 얻어 해당 단서로부터 확률을 추론하는 것이 귀납적, 즉 경험적 방식이다.

주어진 데이터로부터 스스로 단서를 얻어 확률을 추론해 결과를 산출하는 베이지 정리 기반의 확률 모델은 이번 분석에 적용하기에 적절한 모델링 방식이라 기대할 수 있다. 데이터의 X 변수들과 Y 변수인 최적 입지 순위 그룹 간의 어떠한 관계나 단서를 임의로 정하거나 발견할 수 없으므로 확률 기반 모델이 효과적이다.

□ Gaussian Naive Bayes

확률을 이용해 가장 합리적인 예측값을 계산하는 확률 기반 모델이다. 베이지 이론을 기반으로 하는데 각각의 변수가 주어질 때 해당 변수가 여러 분류 값 중 어디에 속할 확률이 가장 높은지를 계산해서 가장 적합한 분류 값을 산출한다.

주어지는 X 변수와 목표하는 Y 분류 값 간의 확률을 기반으로 분류하는 만큼 결과에 신뢰성을 높이기 위해 해당 모델을 선정했다.

해당 모델은 따로 지정해야 할 파라미터 값들이 존재하지 않아 간편하게 바로 진행할 수 있다.

3.2.3 선형 모델

선형 모델은 예측 변수인 X 변수와 반응 변수인 Y 변수 사이의 선형 관계를 다룬다. 가장 간단한 모델 중 하나이다. 학습 속도가 빠르고 예측이 빠르며 데이터의 양에 상관없이 학습이 잘 이루어진다는 장점이 있다.

□ Multinomial Logistic Regression

Multinomial Logistic Regression은 다중 로지스틱 회귀분석으로 로지스틱 회귀 분석은 Y 변수의 값이 1 또는 0을 가지는 이진 변수인 것에 비해 구하고자 하는 Y 변수의 범주가 3개 이상일 때 사용하는 로지스틱 회귀분석이다. 어떤 사건이 발생할 확률을 결정하는 데 사용되는 통계 모델로 정확한 예측값을 구하는 데 사용된다.

사용하는 모델의 다양성을 위해 선형 모델을 시도하기로 하며, 3개의 범주로 분류를 목표로 하므로 해당 모델을 선정했다.

GridSearch를 통해 C값은 0.5가 가장 최적의 값이라는 것을 산출해 모델에 적용한다. 이때 C값은 규제 강도를 조절하여 과적합을 규제하기 위한 파라미터 값의 역수이다. 역수이므로 C값을 감소시키면 규제 강도가 증가해 과대 적합을 방지하고, C값을 증가시키면 규제 강도가 감소해 과소 적합을 방지한다.

□ Support Vector Machine Classifier

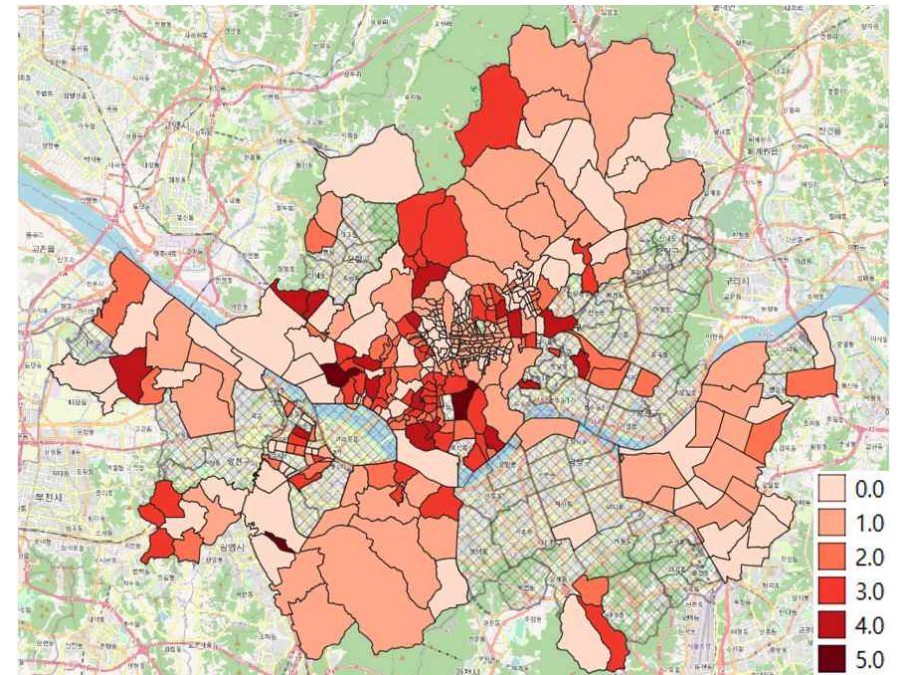
Support Vector Machine Classifier(SVC)는 값들을 분류하기 위한 기준선인 결정경계(Decision Boundary)를 정의하는 모델이다. 몇몇 관측치를 잘못 분류하더라도 나머지의 관측치를 더 잘 분류할 수 있어 과적합을 방지하는 분류기법이다. 몇몇 데이터만 기존의 분류 값이 존재하고 분류 값이 존재하지 않는 데이터들을 유사한 그룹으로 분류하기 위한 분석이므로 해당 모델이 적합하다고 판단해 선정했다. 분류되지 않은 데이터 값을 결정경계와 비교하여 분류하므로 변수들의 값이 비슷한 데이터들이 잘 분류될 것이라 기대한다.

GridSearch를 통해 가장 최적의 조건은 C값은 10, kernel은 rbf라는 것을 확인하고 모델에 적용한다. kernel은 선형 모델인 SVC에 비선형 특성을 추가해 모델을 더욱 강력하게 만들 수 있는 파라미터이다. 어떤 특성을 추가해야 할지 임의로 판단하기는 어려우나 GridSearch를 활용해 쉽게 구할 수 있다. 파라미터의 종류에는 linear, sigmoid, poly(다항), rbf(가우시안)가 있다.

4. 분석결과

4.1 모델링 결과

□ ‘Random Forest’, ‘XGBoost Classifier’, ‘Gaussian Naive Bayes’, ‘다중 Logistic Regression’, ‘SVC’ 총 다섯 개의 모델을 사용하여 다섯 개의 모델이 모두 A로 분류한 지역을 최종 LED 바닥형 보행신호등 설치 최적 입지로 선정하였다.



[그림 4-1] 모델링 결과 시각화

4.2 최적 입지 선정

	법정동	l_rf	l_xgb	l_gnb	l_lr	l_svc	p_rf	p_xgb	p_gnb	p_lr	p_svc	mean_A_prob
2	가리봉동	A	A	A	A	A	0.762240	0.880224	0.999690	0.757798	0.901829	0.860356
15	용산동2가	A	A	A	A	A	0.607642	0.902159	0.849667	0.405399	0.384048	0.629783
46	서교동	A	A	A	A	A	0.497895	0.616378	0.764760	0.485778	0.580569	0.589076

[그림 4-2] LED 바닥형 보행신호등 설치 최적 입지 선정 결과

□ 서울특별시 법정동을 기준으로 ‘구로구 가리봉동’, ‘마포구 서교동’, ‘용산구 용산동2가’ 총 세 개의 지역이 선정되었다.

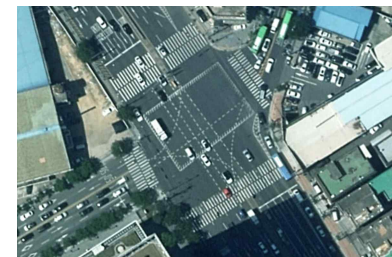
□ 다차선 도로, 취약계층 보호구역, 유동인구, 교통량과 같은 조건들을 고려하여 각 동마다 LED 바닥형 보행신호등 최적 입지에 해당하는 횡단보도들을 추가로 선정하였다.

□ 서울특별시 구로구 가리봉동

서울특별시 구로구 가리봉동은 산업단지와 주거단지 사이에 위치한 곳이다. 유동인구가 많은 가산디지털단지지역과 구로IC, 남부순환로가 존재해 교통량이 많은 ‘가산디지털단지지역구 교차로’와 인근에 아울렛, 시장, 주거밀집지역이 존재하는 ‘디지털단지오거리’를 LED 바닥형 보행신호등 설치 최적 입지로 선정하였다.



[그림 4-3] 가리봉동 내 LED 바닥형 보행신호등 설치 최적 횡단보도 위치



[그림 4-4] 가산디지털단지지역구 교차로

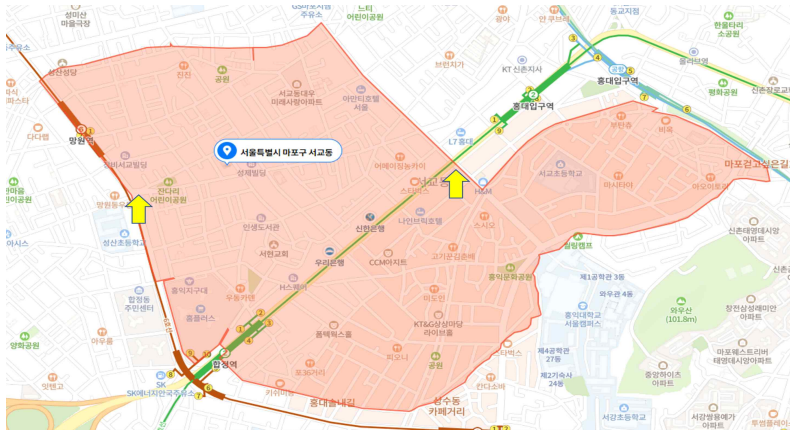


[그림 4-5] 디지털단지오거리

□ 서울특별시 마포구 서교동

서울특별시 마포구 서교동은 국내 최대의 대학가가 존재하는 곳으로, 유동인구, 교통량이 많은 곳이다.

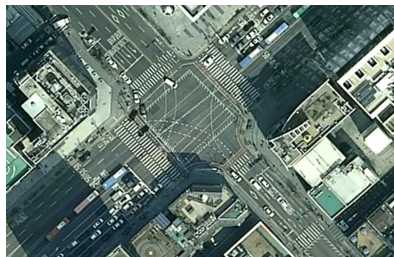
서울성산초등학교와 다차선 도로가 존재하는 '성산초교앞교차로'와 유동인구가 많은 2호선 홍대입구역과 다차선 도로가 존재하는 '홍대입구역사거리'를 LED 바닥형 보행신호등 설치 최적 입지로 선정하였다.



[그림 4-6] 서교동 내 LED 바닥형 보행신호등 설치 최적 횡단보도 위치



[그림 4-7] 성산초교앞교차로



[그림 4-8] 홍대입구역사거리

□ 서울특별시 용산구 용산동2가

서울특별시 용산구 용산동2가는 해방촌이라고도 칭하며 유동인구와 교통량이 모두 많은 지역이다.

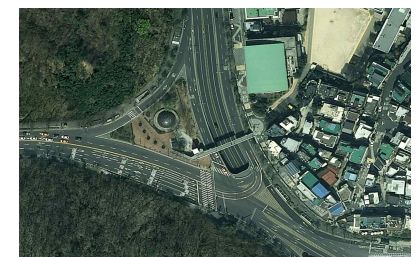
인근에 용산고등학교, 용산중학교, 서울삼광초등학교, 주거밀집지역이 위치한 '용산고교사거리'와 이태원지하차도와 녹사평역이 존재해 교통량과 유동인구가 많은 '녹사평역사거리'를 LED 바닥형 보행신호등 설치 최적 입지로 선정하였다.



[그림 4-9] 용산동2가 내 LED 바닥형 보행신호등 설치 최적 횡단보도 위치



[그림 4-10] 용산고교사거리



[그림 4-11] 녹사평역사거리

5. 활용 방안

5.1 문제점 개선 방안

□ 본 분석은 LED 바닥형 보행신호등이 설치되지 않은 곳 중에서 설치 최적 입지를 선정하여 보행자에게는 안전한 보행환경을, 자치구에게는 교통사고 발생의 감소와 예산의 절감을 제공하기 위해 수행하였다.

□ 분석 결과, 서울특별시 ‘구로구 가리봉동’, ‘마포구 서교동’, ‘용산구 용산동2가’를 LED 바닥형 보행신호등 최적 입지로 선정하였다. 위 지역은 다차선도로, 취약계층 보호구역, 많은 유동인구와 교통량을 보유하고 있는 지역으로 LED 바닥형 보행신호등 설치 시 보행사고 발생을 효과적으로 감소시킬 것으로 보인다.

□ 데이터 분석에 기반하여 바닥형 보행신호등의 최적 입지를 선정함으로써, 사고 감소 효과를 크게 볼 수 없는 지역에 설치하는 일을 예방할 수 있다. 따라서 설치 비용을 줄여 각 자치구별 교통안전시설 관련 예산을 절감할 수 있다.

5.2 업무 활용 방안

□ 정책결정 활용

LED 바닥형 보행신호등이 설치되지 않은 자치구에게 객관적 데이터에 기반한 설치 최적 지역을 선정함으로써 신뢰도 높은 정책결정을 할 수 있다.

□ 안전한 보행환경 제공

LED 바닥형 보행신호등의 설치로 모든 국민이 안심하며 이용할 수 있는 횡단보도 보행환경을 제공할 수 있으며, 이는 2022년 하반기부터 이슈가 되고 있는 도로교통 안전에 기여할 수 있다.

□ 예산절감의 효과

데이터에 기반한 LED 바닥형 보행신호등의 최적 입지를 제안함으로써 사고감소 효과를 크게 볼 수 없는 지역의 설치를 예방함으로써 예산절감의 효과를 기대할 수 있다.

□ 도시계획과 연계

교통사고, 생활인구, 주민등록수를 반영한 모델을 기반으로 현재 데이터 플랫폼 구축에 기반한 스마트국토를 목표로 하는 현재 서울특별시의 정책 방향에 기여할 수 있다.

□ 국민의식 제고

LED 바닥형 보행신호등의 설치로, 운전자 및 보행자에게 보행사고 주의지역임을 자각시키고, 보행사고에 대한 경각심을 갖게 함으로써 보행안전에 대한 국민의식을 향상시킬 수 있다.

□ 상승효과

시중에 판매중인 LED 바닥형 보행신호등에 추가적으로 LED, 반사테이프가 설치된 안전바, 고보조명, 픽셀 아트 등의 시설물 및 장치를 기획하였다. LED 바닥형 보행신호등과 함께 추가적인 신호를 제공함으로써 보행자 보호 인프라를 확충하여 보행사고 예방에 큰 효과를 기대할 수 있을 것이다.

6 참고자료

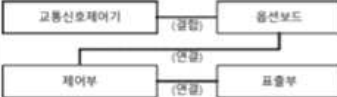

6.1 표준지침

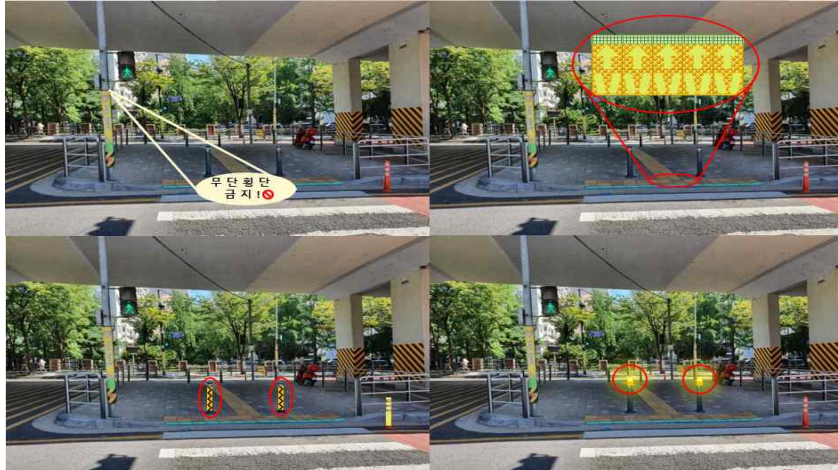
□ 표준지침

□ 주요 개정사항

- 경찰청 「교통신호제어기 표준규격서」 상 '디지털 교통신호제어기' 도입에 따른 디지털 교통신호제어기용 보조장치 개발을 위한 개정
- 교통신호 관련 보조장치의 원활한 설치를 장려 및 중복심의 (신호등 설치 시, 보조장치 설치 시)에 따른 개정
- 시험기준 상세화, 오타자 수정 등

<바닥형 보행신호등 보조장치 표준지침 신·구조문 대비표>

현 행	개 정
3. 용어 정의 (5) 휘도균일도(Luminous Intensity Uniformity) : 표출부의 밝기가 전 표면에 균일하게 구현되는 정도	3. 용어 정의 (5) 휘도균일도(Luminance Uniformity) : 표출부의 밝기가 전체 표면에 균일하게 구현되는 정도
5. 설치 기준·위치 및 배열 / 5.1 설치기준 바닥형 보행신호등 보조장치는 왕복4차로 이상인 도로중에서 보행자 통행이 빈번하고 보행자 횡단사고가 잦은 보행구역(어린이 노면차량안전)에 설치할 수 있다. 이의 지역 중 교통안전상 부득이하게 설치할 필요가 있을 경우는 위 기준을 고려하여 관련 지방경찰청(또는 경찰서) 교통안전시설심의위원회의 결정에 따라 설치할 수 있다.	5. 설치 기준·위치 및 배열 / 5.1 설치기준 바닥형 보행신호등 보조장치는 왕복4차로 이상인 도로중에서 보행자 통행이 빈번하여 보행자의 횡단사고가 잦은 지점 등 보행자의 안전을 위해 필요한 지점에 설치할 수 있다.
6. 구성 요소 / 6.3 옵션보드 교통신호제어기에 설치되어 해당 보행신호등의 신호상태 등을 제어부에 전달하는 장치  [그림 5] 교통신호제어기와 제품 구성 요소의 연결	6. 구성 요소 / 6.3 신호상태정보 취득부 6.3.1 아날로그 교통신호제어기 아날로그 교통신호제어기에 옵션보드를 설치하여 해당 보행신호등의 신호상태 등을 제어부에 전달  [그림 5] 교통신호제어기와 제품 구성 요소의 연결
7. 제품 요건 / 7.2 제어부 / 7.2.2 제어부의 재료 및 성능 (5) 제어부에 컨버터 등 표출부의 구동을 위한 전원장치가 사용될 경우 KS 인증을 득한 장치를 사용하여야 한다.	7. 제품 요건 / 7.2 제어부 / 7.2.2 제어부의 재료 및 성능 (5) 제어부에 컨버터 등 표출부의 구동을 위한 전원장치가 사용될 경우 KC 인증을 득한 장치를 사용하여야 한다.



[그림 5-1] 추가 시설물 기획

□ 시방서

현 형	개 정
7. 제품 요건 / 7.3 옵션보드 옵션보드의 구조 및 요구사항은 다음과 같다. (1) 표준 교통신호제어기와와의 호환되어야 한다. (2) 표준 교통신호제어기로부터 시보정보, 보행신호상태, 보행녹색점멸시간 정보를 수신하여야 한다. (3) 옵션보드의 구성, 구조 및 통신과 관련한 규약은 경찰청 교통신호제어기 표준규격서를 따른다. (4) 선택적으로 바딕형 보행신호등의 정상작동 여부를 표준 교통신호제어기의 주제어기(CPU)로 송신할 수 있어야 한다. (5) 이에 대한 적합성은 교통신호제어기 부품호환성 기능검사를 통하여 확인한다. (6) 옵션보드는 교통신호제어기 CPU로부터 50 ms이하 간격으로 정보를 제공받아 바딕형 보행신호등 제어부로 전달하여야 한다.	7. 제품 요건 / 7.3 신호상태정보 위독부 7.3.1 아날로그 교통신호제어기 아날로그 교통신호제어기와 옵션보드의 구조 및 요구사항은 다음과 같다. (1) 표준 교통신호제어기와의 호환되어야 한다. (2) 표준 교통신호제어기로부터 시보정보, 보행신호상태, 보행녹색점멸시간 정보를 수신하여야 한다. (3) 옵션보드의 구성, 구조 및 통신과 관련한 규약은 경찰청 교통신호제어기 표준규격서를 따른다. (4) 선택적으로 바딕형 보행신호등의 정상작동 여부를 표준 교통신호제어기의 주제어기(CPU)로 송신할 수 있어야 한다. (5) 이에 대한 적합성은 교통신호제어기 부품호환성 기능검사를 통하여 확인한다. (6) 옵션보드는 교통신호제어기 CPU로부터 50 ms이하 간격으로 정보를 제공받아 바딕형 보행신호등 제어부로 전달하여야 한다.
9. 시험기준 및 방법 / 9.2 시험방법 / IV. 부분 결함시험 (표준부 + 제어부) / 9.2.16 중심회도 및 조광제어 (1) 시험하고자 하는 바딕형 보행신호등을 정규상태로 세워놓고 본 지점 8방향에 표시된 각각의 지점(각도에 대해) 회도를 측정한다. 이 측정으로 알릴의 안정화 후에 25℃ 주변 온도에서 수행되어야 하며, 측정거리는 최소한 10m이상의 거리를 유지하여야 한다.	7.3.2 디지털 교통신호제어기 (1) 경찰청 「교통신호제어기 표준 규격」의 '6.2.3.10 신호구동부(SLC) → 신호정보 이용 외부장치에 따라 디지털 교통신호제어기에서 보행신호등의 신호상태 등을 제어부에서 직접 수신한다. (2) 경찰청 「교통신호제어기 표준 규격」의 '6.2.4 디지털 교통신호제어기의 옵션 기능 처리 기준에 따라 디지털 교통신호제어기에서 보행신호등의 신호상태 등을 제어부에서 직접 수신할 수도 있다.
9. 시험기준 및 방법 / 9.2 시험방법 / IV. 부분 결함시험 (표준부 + 제어부) / 9.2.16 중심회도 및 조광제어 (1) 시험하고자 하는 바딕형 보행신호등을 정규상태로 세워놓고 본 지점 8방향에 표시된 각각의 지점(각도에 대해) 회도를 측정한다. 이 측정으로 알릴의 안정화 후에 25℃ 주변 온도에서 수행되어야 하며, 측정거리는 최소한 10m이상의 거리를 유지하여야 한다.	9. 시험기준 및 방법 / 9.2 시험방법 / III. 부분 결함시험 (표준부 + 제어부) / 9.2.16 중심회도 및 조광제어 (1) 시험하고자 하는 바딕형 보행신호등을 정규상태로 세워놓고 중심축과 광위도계의 광축이 일치되도록 하고, 아래 그림과 같이 표준부의 중심면적에 대해 회도를 측정할 수 있도록 광위도계를 위치시킨다.
9. 시험기준 및 방법 / 9.2 시험방법 / IV. 부분 결함시험 (표준부 + 제어부) / 9.2.18 회도균일도 (2) 이러한 기하조건이 유지되도록, 바딕형 보행신호등 또는 광 위도계를 수직이동시키며 지점에 표시된 측정지점의 회도값을 측정한다.	9. 시험기준 및 방법 / 9.2 시험방법 / III. 부분 결함시험 (표준부 + 제어부) / 9.2.18 중심회도 및 조광제어 (2) 이러한 기하조건이 유지되도록, 바딕형 보행신호등 또는 광 위도계를 수직이동시키며 아래와 같은 측정지점의 회도값을 측정한다.
9. 시험기준 및 방법 / 9.2 시험방법 / IV. 부분 결함시험 (표준부 + 제어부) / 9.2.26 소비전력, 역률 및 종고조파함유율 (2) 단, 역률과 종고조파함유율의 측정을 위하여 현장에 설치되는 표준부의 수량에 해당하는 부하를 연결할 수 있다.	9. 시험기준 및 방법 / 9.2 시험방법 / III. 부분 결함시험 (표준부 + 제어부) / 9.2.26 소비전력, 역률 및 종고조파함유율 (2) 단, 소비전력, 역률과 종고조파함유율의 측정을 위하여 현장에 설치되는 표준부의 수량에 해당하는 부하를 연결할 수 있다.
9. 시험기준 및 방법 / 9.2 시험방법 / V. 중 결함시험 (표준부 + 제어부 + 옵션보드를 장착한 표준제어기)	9. 시험기준 및 방법 / 9.2 시험방법 / IV. 중 결함시험 (표준부 + 제어부 + 옵션보드를 장착한 표준제어기)
11. 보증 및 명판화 <표 10> 명판 내용	11. 보증 및 명판화 <표 10> 명판 내용

LED 바닥신호등 공사 시방서 (특별시방서)

3. 시공방법

3.1 전원 공급 방법 선택

(1) **신호제어기 사용시** : 신호제어기 위치 확인→전기선로 실측→신호제어기 신호전류 연결→컨트롤박스 부착위치 선정→도면대조

(2) 길이 측정

3.2 점자블록 재단 및 절단

- 교통약자 이동편의 증진법에 의한 안전유도블럭(점자블럭) 설치 법
: 도로에서 300mm 떨어진곳에 점자블록 설치(경계석 폭 200mm)



- 점자블록과 경계석 위치 사이에 바닥신호등 설치 구간이므로 현재 경계석과 점자블록과 붙여져 있는 경우 100mm 인도쪽으로 옮겨서 설치

(1) 먹줄로 정확한 길이 재단(100mm폭)

(2) 절단기를 사용하여 정확한 재단 : 절단시 먼지발생과 안전에 유의하여 작업하며, 절단기 날의 길이를 확인한다. (80mm)



3.3 터파기 및 바닥 고르기

(1) **기존 점자블록 및 기타 장애물 제거** : 기존 점자블록의 절단부분이 흐트러지지 않도록 작업한다.

(2) **바닥 고르기** : 모래 2 : 콘크리트 1의 비율로 혼합한 콘크리트 작업을 하고, 모래의 깊이가 65mm가 되도록 투입 후 수평이 되도록 다진다.



3.4 사진 촬영 및 현장 정리

매 공정 진행 전, 중, 후 사진을 촬영하여 보고 및 보관용 자료 수집 한다.

모든 공사 완료 후 기존 시설 위치 및 상태 원상 복구하며 주변을 정리 한다.

LED 바닥신호등 공사 시방서 (특별시방서)

3.5 LED 바닥신호등 설치

- (1) LED바닥신호등에 전선 연결 : 방수용 전용 커넥터를 활용하여 전원선 연결 후 수축관을 활용하여 마감 처리한다.



3.6 전기 배선공사 및 콘트롤 박스 설치

(1) 전기 배선공사 :

- 절단기로 35mm 절단한다.
- 해머드릴로 150mm 깊이가 되도록 팔 것.
- 후핵시를 배관을 묻는다.
- 노비로 전선을 넣어 안전블록에 전원을 연결한다.

(커넥터를 이용하여 각각의 스크류를 맞춰, 안쪽으로 밀면서 조임, 확실히 조인 후, 본드 수축튜브<한쪽에 미리 끼워져 있어야 함>를 덮고 열풍기로 마무리함)

(2) 콘트롤박스 설치

- 신호등 지주대에 콘트롤 박스를 고정.
- 전원선을 콘트롤박스에 연결.
- 출력 24V를 연결한다.(+,- 극성주의)
- 발전기를 가동하여 전원을 넣어 블록에 불이 켜지는지 확인한다.
- 접지저항을 테스터 확인.(접지봉은 60cm)



3.7. 주변 정리정돈

- * 주변 정리정돈과 함께 폐기물 처리



6.2 사고 현황

□ 교통사고 현황 및 추정

구분	전체 교통사고	보행자 교통사고	어린이 교통사고	고령자 교통사고	보행자 교통사고 건수 비율
2014년	12,693건	3,800건	557건	1,879건	29.9%
2015년	12,757건	3,683건	526건	1,879건	28.9%
2016년	12,192건	3,604건	473건	1,982건	29.6%
2017년	11,753건	3,391건	481건	2,038건	28.9%
2018년	11,937건	3,361건	411건	2,147건	28.2%
2019년	12,992건	3,631건	471건	2,325건	27.9%
연평균 증감률	0.5%	-0.9%	-5.5%	4.4%	-1.4%
2020년 추정	13,057건	3,598건	455건	2,427건	27.6%

자료 : TAAS 교통사고분석시스템, 각년도

6.3 바닥신호등 효과

□ 야간사고 효과

(2) 야간사고

분석그룹의 야간 사전 교통사고 건수는 36건, 사후 교통사고 건수는 14건, 비교그룹 사전 교통사고 건수는 87건, 사후 교통사고 건수는 73건 발생했다. 사고감소율은 82.21%로 분석되어 주간 보다는 야간에 더욱 효과적인 것으로 추정된다.

[표 3-7] 바닥형 보행신호등 야간 교통사고 분석결과

분석그룹의 사전 교통사고 건수	36	교통사고 변화건수	15.864
분석그룹의 사후 교통사고 건수	14	교통사고 효율성 척도	0.178
비교그룹의 사전 교통사고 건수	87	교통사고 효율성 척도 표준편차	0.088
비교그룹의 사후 교통사고 건수	73	신뢰수준 구간(95%)	0.01~0.35
사후시점의 해당도로에서 조사된 실제 사고건수	14	비교그룹방법 사고감소율	82.21%
사후시점의 해당도로에서 사업이 시행되지 않았을 경우 예측된 사고건수	29.864	단순사고건수 사고감소율	61.11%

□ 보행자사고 효과

(3) 단일로 보행자사고

단일로 보행자사고의 사전 교통사고 건수는 3건, 사후 교통사고 건수는 2건이며 비교그룹의 사전 교통사고 건수는 10건, 사후 교통사고 건수는 7건이다. 비교그룹방법으로 사고감소율 효과분석 결과 43.93% 사고 감소효과로 단일로 유형에서 가장 높은 것으로 추정된다.

[표 3-11] 바닥형 보행신호등 단일로 보행자 교통사고 분석결과

분석그룹의 사전 교통사고 건수	3	교통사고 변화건수	-0.091
분석그룹의 사후 교통사고 건수	2	교통사고 효율성 척도	0.561
비교그룹의 사전 교통사고 건수	10	교통사고 효율성 척도 표준편차	0.351
비교그룹의 사후 교통사고 건수	7	신뢰수준 구간(95%)	-0.13~1.25
사후시점의 해당도로에서 조사된 실제 사고건수	2	비교그룹방법 사고감소율	43.93%
사후시점의 해당도로에서 사업이 시행되지 않았을 경우 예측된 사고건수	1.909	단순사고건수 사고감소율	33.33%

□ 현장조사 결과

3.3.2 현장조사 결과

지점 기준으로 출발의 경우 아차산역은 전체 보행자 901명 중 이상보행자는 34명으로 약 3.8% 수준이며 천호역은 전체 963명 중 이상보행자는 124명으로 약 12.9%로 분석되었다. 도착분석의 경우 아차산역은 전체 보행자 1,245명 중 이상보행자는 19명으로 약 1.5% 수준이며 천호역은 전체 1,024명의 보행자 중 이상보행자는 29명으로 약 2.8% 수준으로 분석되었다.

[표 3-21] 바닥형 보행신호등 현장조사 결과

구분	지점명	정상보행		이상보행		계		위반율	
		주	야	주	야	주	야	주	야
출발	아차산역	605	262	22	12	627	274	3.5%	4.4%
	천호역	457	382	60	64	517	446	11.6%	14.3%
도착	아차산역	612	614	5	14	617	628	0.8%	2.2%
	천호역	647	348	19	10	666	358	2.9%	2.8%

신동협, 비교그룹방법을 이용한 바닥형 보행신호등 보조장치 설치 효과 분석,
2021.08

7. 부록

7.1 사례조사 정리



출처 : <https://blog.naver.com/kdj570/221704176381>

바닥형 보행신호등보행신호 음성안내장치 설치 확대 추진

박광하 기자 | 승인 2022.08.02 19:22 | 댓글 0



■경찰청, 보행관련 설비 규정 개정·배포

교통안전시설심의위 결정 없이
사고 잦은 지점이면 설치 허용

디지털 교통신호제어기 통해
신호상태 제어부 직수신 허용



'바닥형 보행신호등'은 보행신호등이 설치된 횡단보도에 선택적으로 설치되는 보행 보조장치로, 횡단보도 대기선 바닥에 보행신호를 점등해 보행자에게 추가적인 신호정보를 제공하는 장치다. 바닥형 보행신호등은 보행 편의 향상과 교통사고 방지 효과가 있어 최근 설치운영이 전국적으로 증가하고 있다.

'바닥형 보행신호등 보조장치 표준지침'의 주요 개정 내용을 살펴보면, 경찰청은 교통신호 관련 보조장치의 확대 설치를 촉진하고자 설치기준을 개정했다.

"눈에 확 띄네요" 바닥신호등 전국 도입 확대...서울 자치구 '천차만별'

기사입력 : 2021년09월04일 06:00 | 최종수정 : 2021년09월04일 06:00



가+ 가- 프린트

전국 바닥신호등 760개...6개월 사이 2배 이상 증충
서울시 자치구 가운데 마포구 '0'개...연내 설치 계획 無

[서울=뉴스핌] 최현민 기자 = # 지난 1일 오후 7시쯤 한강공원에 가기 위해 횡단보도 앞에서 신호를 기다리며 핸드폰을 보고 있던 김도원(34) 씨는 깜짝 놀랐다. 바닥에 있던 빨간색 LED(발광다이오드) 불빛이 횡단보도 신호에 맞춰 초록색으로 바뀌었기 때문이다. 김씨는 "(우리 동네에서는) 한번도 본적이 없는데 횡단보도 신호가 바닥에도 나오니 신기하다"면서 "가급 핸드폰을 보다 신호가 바뀐지도 모르고 뒤늦게 뛰어가곤 했는데, 여기서 그칠 일이 없겠다"며 발걸음을 옮겼다.

안전한 보행환경 조성을 위해 주로 초등학교나 거주자 밀집지역 등에 설치된 바닥형 보행신호등 보조장치(바닥신호등)가 전국적으로 늘어나고 있는 가운데 서울지역은 자치구별로 천차만별인 것으로 나타났다. 자치구별로 교통안전시설 설치나 유지관리에 필요한 예산이 한정 편성되는데 따른 결과다. 설치비용도 만만치 않아 자치구별 차이는 더 크게 벌어질 것이란 우려도 나온다.

40개 이상 바닥신호등이 설치된 자치구가 있는 반면 1개도 채 설치되지 않은 곳도 있었다. 바닥신호등이 없는 자치구는 강북구·마포구·성북구·용산구·종로구 등 5개구다. 도봉구와 강동구도 각각 1개, 2개에 불과했다.

이 외에 ▲광진구(35) ▲성동구(24) ▲금천구(22) ▲영등포구(18) ▲중랑구·양천구(17) ▲서대문구(13) ▲강서구(10) ▲구로구·서초구(8) ▲노원구·관악구(5) ▲동대문구·동작구(4) ▲은평구(3) 등이다.

바닥신호등이 도입된지 2년이 지난 만큼 자치구별로 올해는 바닥신호등 설치에 박차를 가한다는 계획이다. 강북구는 연내 29개 횡단보도에 바닥신호등을 설치할 계획이다. 도봉구와 용산구도 각각 5개, 3개를 설치할 계획이며, 종로구 역시 관내 초등학교 인근 2군데에 우선적으로 설치를 검토하고 있다.

다만 마포구는 올해도 바닥신호등 설치 계획은 없다고 밝혔다. 횡단보도에 바닥신호등을 설치하기 위해선 큰 비용이 발생하기 때문이다.

마포구 관계자는 "기본적인 교통안전시설이나 유지관리를 하는 예산 범위내에서 할 수가 없다"면서 "올해 바닥신호등을 설치하기 위해 검토된 사안은 없다"고 전했다.

7.3 분석 상세코드

□ 데이터 전처리

```
### 데이터 전처리
a1 = pd.read_csv("사고데이터 및 바닥신호등 폴더/설치법정동사고감소율그룹2.csv")
a2 = pd.read_csv("2019전처리완료데이터/2019최종데이터.csv")
df = pd.merge(a1.iloc[:,[1,5]], a2, how = "left", left_on = "동", right_on = "법정동")
df = df.drop(["동"],axis = 1) # 필요없는 변수 삭제
df = df.set_index("법정동") # 법정동을 인덱스로 설정
df = df.rename(columns = {"group":"최적입지순위그룹", "전체평균연령":"평균나이"}) # group 변수 이름 변경
df = df.iloc[:,[0,1,2,3,4,5,6,8,9,10,7]]
df
y1 = df["최적입지순위그룹"]
x1 = df.drop(["최적입지순위그룹"],axis = 1)

# 범주형변수 더미화
x1 = pd.get_dummies(x1, drop_first = True)
x1

# cv값 지정
from sklearn.model_selection import KFold
cv = KFold(n_splits = 4, shuffle = True, random_state = 42)

# smote 기법 사용(불균형 데이터 처리)
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=999)
x1, y1 = smote.fit_resample(x1, y1)

# 데이터 변환(왜도가 0에 가까운 방향으로)
x1.iloc[:,[0,1,2,3,4,5,7]] = np.sqrt(x1.iloc[:,[0,1,2,3,4,5,7]])
x1.iloc[:,6] = np.log1p(x1.iloc[:,6])

# 데이터의 범위가 유사해지도록 scaling 수행(이상치의 영향을 줄이기 위해 RobustScaler 사용)
```

```
from sklearn.preprocessing import RobustScaler
scaler = RobustScaler()
x1.iloc[:,9] = scaler.fit_transform(x1.iloc[:,9])
x1
```

□ RandomForest

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
rf1 = RandomForestClassifier(random_state = 42)
parameters = {"n_estimators": [50, 100, 200, 500, 1000],
              "max_depth": [3, 5, 7, 9, 11],
              'min_samples_split':[2, 3, 5, 7, 9, 11, 13]
              }
grid_search1 = GridSearchCV(estimator=rf1,
                           param_grid=parameters,
                           n_jobs=-1,
                           cv = cv
                           )
grid_search1.fit(x1, y1)
grid_search1.best_params_
# 'max_depth': 11, 'min_samples_split': 2, 'n_estimators': 50

from sklearn.ensemble import RandomForestClassifier
rf1 = RandomForestClassifier(max_depth = 11, min_samples_split = 2, n_estimators = 50, random_state = 42)
rf1.fit(x1, y1)
```

- * n_estimators : 결정트리의 개수를 지정
- * max_depth : 트리의 최대 깊이
- * min_samples_split : 리프노드가 되기 위해 필요한 최소한의 샘플 데이터 수

□ XGB Classifier

```

from sklearn.model_selection import GridSearchCV
from xgboost import XGBClassifier
xgb1 = XGBClassifier(random_state = 42)
parameters = {'max_depth': [3, 5, 7, 9, 11],
              'subsample': [0.2, 0.4, 0.6, 0.8, 1.0],
              "n_estimators": [50, 100, 200, 500, 1000]}
grid_search2 = GridSearchCV(estimator=xgb1,
                           param_grid=parameters,
                           n_jobs=-1,
                           cv = cv
                           )
grid_search2.fit(x1, y1)
grid_search2.best_params_

# 'max_depth': 5, 'n_estimators': 50, 'subsample': 1.0

from xgboost import XGBClassifier
xgb1 = XGBClassifier(max_depth = 5, subsample = 1, n_estimators = 50, random_state
                    = 42)
xgb1.fit(x1, y1)

```

- * n_estimators : 결정트리의 개수를 지정
- * max_depth : 트리의 최대 깊이
- * subsample : 각 트리마다 데이터 샘플링 비율, over-fitting 방지

□ Gaussian Naive Bayes

```

from sklearn.naive_bayes import GaussianNB
gnb1 = GaussianNB()
gnb1.fit(x1,y1)

```

□ Multinomial Logistic Regression

```

from sklearn.linear_model import LogisticRegression
lr1 = LogisticRegression(multi_class='multinomial', solver='lbfgs', random_state =
42)
parameters = {"C": [0.001, 0.01, 0.1, 0.5, 1, 3, 5, 10, 30, 50]}
grid_search3 = GridSearchCV(estimator=lr1,
                           param_grid=parameters,
                           n_jobs=-1,
                           cv = cv
                           )
grid_search3.fit(x1, y1)
grid_search3.best_params_
# 'C': 0.5

from sklearn.linear_model import LogisticRegression
lr1 = LogisticRegression(multi_class='multinomial', solver='lbfgs', C = 0.5, random_state = 42)
lr1.fit(x1,y1)

```

- * C : 규제 강도를 조절하는 alpha 값의 역수

□ Support Vector Machine Classifier

```

from sklearn.model_selection import GridSearchCV
import sklearn.svm as svm

svc1 = svm.SVC(random_state = 42)
parameters = {"C":[0.001, 0.01, 0.1, 0.5, 1, 3, 5, 10, 30, 50],
              "kernel":["linear", 'sigmoid', 'rbf', 'poly']}

grid_search4 = GridSearchCV(estimator=svc1,
                             param_grid=parameters,
                             n_jobs=-1,
                             cv = cv
                             )

grid_search4.fit(x1, y1)
grid_search4.best_params_

# 'C': 10, 'kernel': 'rbf'

import sklearn.svm as svm
svc1 = svm.SVC(C = 10, kernel = "rbf", probability=True, random_state = 42)
svc1.fit(x1, y1)

```

- * C : 규제 강도를 조절하는 alpha 값의 역수
- * kernel : 가우시안 rbf 커널

□ 최적입지 선정

```

# 모델 별 분류 라벨 불러오기

rf1 = pd.read_csv("모델링_분류라벨/RandomForest_label12.csv")
xgb1 = pd.read_csv("모델링_분류라벨/XGBClassifier_label12.csv")
gnb1 = pd.read_csv("모델링_분류라벨/gnb_label12.csv")
lr1 = pd.read_csv("모델링_분류라벨/LogisticRegression_label12.csv")
svc1 = pd.read_csv("모델링_분류라벨/SVC_label12.csv")

# 데이터 결합
df1 = pd.merge(rf1, xgb1, how="outer", on="법정동")
df1 = pd.merge(df1, gnb1, how="outer", on="법정동")
df1 = pd.merge(df1, lr1, how="outer", on="법정동")
df1 = pd.merge(df1, svc1, how="outer", on="법정동")

# 컬럼명 재정의
df1.columns = ["법정동", "l_rf", "p_rf", "l_xgb", "p_xgb", "l_gnb", "p_gnb",
               "l_lr", "p_lr", "l_svc", "p_svc"]

df1 = df1.iloc[:, [0, 1, 3, 5, 7, 9, 2, 4, 6, 8, 10]] # 데이터 순서 정렬

# 5개의 모델 모두에서 A가 나온 지역 필터
all_include1 = df1.dropna()

# 5개 모델의 A확률 평균을 변수로 생성
all_include1["mean_A_prob"] = all_include1.iloc[:, 6:].mean(axis=1)

# 최적입지 데이터
all_include1

```

□ t-sne 시각화

```
# 데이터 불러오기
a1 = pd.read_csv("/content/drive/MyDrive/4조 프로젝트/사고데이터 및 바닥신호등 폴더/
설치법정동사고감소율그룹2.csv")
a2 = pd.read_csv("/content/drive/MyDrive/4조 프로젝트/2019전처리완료데이터/2019최종테
이터.csv")

df = pd.merge(a1.iloc[:,[1,5]], a2, how = "left", left_on = "동", right_on = "법정
동")
df = df.drop(["동"],axis = 1)
df = df.set_index("법정동") # 법정동을 인덱스로 설정
df = df.rename(columns = {"group":"최적입지순위그룹"}) # group 변수 이름 변경
df

y = df["최적입지순위그룹"]
x = df.drop(["최적입지순위그룹"],axis = 1)

# 범주형변수 더미화
x = pd.get_dummies(x, drop_first = True)
x

from sklearn.manifold import TSNE
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import StandardScaler
# 데이터 변환

x.iloc[:,[0,1,2,3,4,5,7]] = np.sqrt(x.iloc[:,[0,1,2,3,4,5,7]])
x.iloc[:,6] = np.log1p(x.iloc[:,6])

# 데이터 스케일링

scaler = RobustScaler()
x.iloc[:,9] = scaler.fit_transform(x.iloc[:,9])

# 2차원으로 차원축소하는 모델 생성(SMOTE 이전)

model = TSNE(n_components=2,random_state = 999)
```

```
tsne_df = model.fit_transform(x)

# tsne 완료된 데이터 데이터프레임 변환
tsne_df = pd.DataFrame(tsne_df, columns = ["component0","component1"])
tsne_df

# y값 붙여주기
tsne_df["target"] = np.array(y)
tsne_df

# target 별 분리
tsne_df_0 = tsne_df[tsne_df['target'] == "A"]
tsne_df_1 = tsne_df[tsne_df['target'] == "B"]
tsne_df_2 = tsne_df[tsne_df['target'] == "C"]

# target 별 시각화(SMOTE 이전)
plt.scatter(tsne_df_0['component0'], tsne_df_0['component1'], color = 'pink', label = 'A')
plt.scatter(tsne_df_1['component0'], tsne_df_1['component1'], color = 'purple', label = 'B')
plt.scatter(tsne_df_2['component0'], tsne_df_2['component1'], color = 'yellow', label = 'C')
plt.xlabel('component0')
plt.ylabel('component1')
plt.legend()
plt.show()

### SMOTE 적용

from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=999)
x, y = smote.fit_resample(x, y)

# 2차원으로 차원축소하는 모델 생성(SMOTE 이후)
model = TSNE(n_components=2,random_state = 999)
tsne_df = model.fit_transform(x)

# tsne 완료된 데이터 데이터프레임 변환
tsne_df = pd.DataFrame(tsne_df, columns = ["component0","component1"])
tsne_df

# y값 붙여주기
```

```
tsne_df["target"] = np.array(y)
tsne_df

# target 별 분리
tsne_df_0 = tsne_df[tsne_df['target'] == "A"]
tsne_df_1 = tsne_df[tsne_df['target'] == "B"]
tsne_df_2 = tsne_df[tsne_df['target'] == "C"]

# target 별 시각화(SMOTE 이후)
plt.scatter(tsne_df_0['component0'], tsne_df_0['component1'], color = 'pink', label = 'A')
plt.scatter(tsne_df_1['component0'], tsne_df_1['component1'], color = 'purple', label = 'B')
plt.scatter(tsne_df_2['component0'], tsne_df_2['component1'], color = 'yellow', label = 'C')
plt.xlabel('component0')
plt.ylabel('component1')
plt.legend()
plt.show()
```