데이터기반 행정으로 국민의 삶의질을 개선하라!

데이턴십 해커톤 제 4 회

서울시 장애인콜택시 대기시간 단축을 위한 머신러닝 기반 수요량 예측 모델 제안

(서울시 장애인 콜택시 수요량 예측모델 개발 및 배치선정 기준 제안)

분석 결과보고서

참여조 : 28 조

참여자 : 구병모(조장)

김찬우

박은희

이민화

장문용

정범수

씨에스리 컨소시엄

CSLEE kpc 한국생산성본부

목 차

1. 분석개요

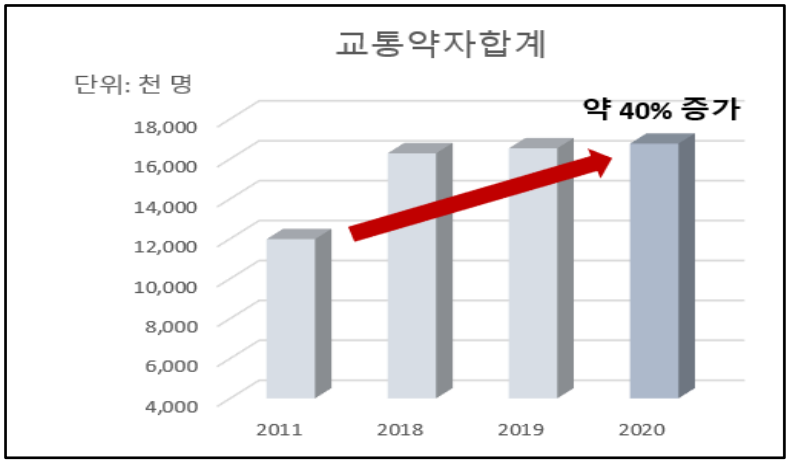
1.1 분석배경 및 개요

1.1.1 교통약자와 특별교통수단 현황

○ **교통약자**란 장애인, 고령자, 임산부, 영유아를 동반한 사람, 어린이 등 일상생활에서 이동에 불편을 느끼는 사람(교통약자법 제 2 조 제 1 호)으로 규정하고 있으며, 환자와 같은 일시적 이동 약자도 교통약자에 해당될 수 있다.

○ **특별교통수단**이란 이동에 심한 불편을 느끼는 교통약자의 이동을 지원하기 위해 휠체어 탑승설비 등을 장착한 차량을 말한다.

○ 각 지역은 교통약자의 이동 편의를 위하여 국토 해양부령이 정하는 일정 대수 이상의 특별교통수단을 운행해야 하며, 특별교통수단 및 이동지원센터의 운영 등에 관하여 필요한 사항은 당해 지방자치단체의 조례로 정하도록 하고 있다.[1]



[그림 1] 교통약자 합계

[1] 홍현근(2018),「교통약자 이동편의증진 정책현황」,『월간교통』, 한국교통연구원, 6-15.

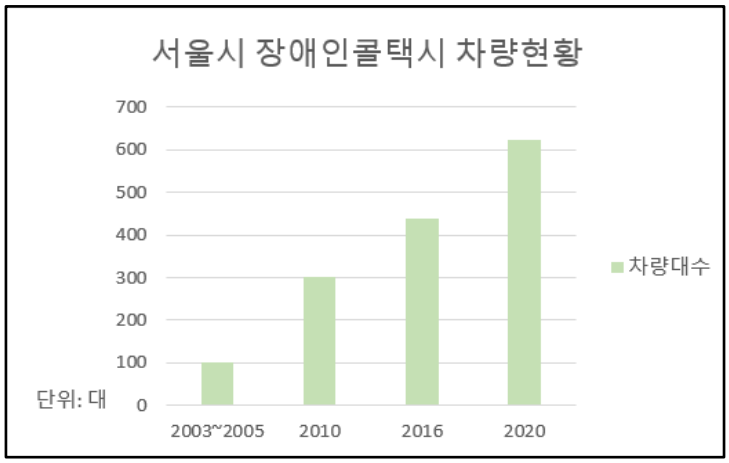
○ 2011 년부터 2020 년까지 교통약자 증가율은 약 40%(국토교통부 '2020 년도 교통약자 이동 편의 실태조사')로, 2020 년 기준 대략 16,763 천명 정도가 교통약자에 해당하는 것으로 집계되었다. 전체인구 대비 교통약자 비율은 지속적 증가 추세이고 이에 따른 교통약자를 위한 특별교통수단 또한 증가하고 있다.

○ 교통약자 유형 중 특히 장애인은 대중교통 이용에 물리적·신체적으로 가장 많은 불편을 겪는 유형으로, 일반 버스 및 지하철 등의 대중교통 이용에 있어서 제한을 많이 받고 있는 것으로 나타났다.[2] 장애로 인해 일상에서 이동에 제한받는 사회 구성원의 불편을 공감하고 해결하는 것은 국민 생활 개선에 필수적인 과제이다. 따라서 본 프로젝트 목적인 '교통약자 이동 편의성 증진'의 대상을 장애인으로 한정하고, 장애인콜택시 시스템 개선으로 그 목적을 달성하고자 하는 바이다.

[2] 예병정, "'장애인 이동' 저상버스 늘리는 서울시…문제는 더딘 속도", 파이낸셜뉴스, 2021.07.12. https://www.fnnews.com/news/202107121751221579

1.1.2 장애인 콜택시 운영 현황

1.1.2.1 서울시 장애인 콜택시 운영 현황



[그림 2] 서울시 장애인콜택시 차량현황

○ 서울시설공단에 따르면, 서울시 장애인콜택시는 장애인의 사회참여 확대를 위해 2003 년 운행을 개시하였다. 최초 100 대로 운행을 시작한 이래로 점차 증가하여 2020 년 622 대가 운행 중이다. 이용 시민이 전화/문자, 인터넷 또는 앱을 통해 차량을 신청하면 접수 순서에 따라 가장 근접한 차량을 연결해주고, 차량이 이용 시민 출발지로 이동하면 이용자가 탑승하여 목적지까지 운행되는 시스템이다.

○ 이용 대상은 보행상 장애가 있는 장애 정도가 심한(기존 1~3 급) 장애인으로, 복지 카드를 제출하여 가입한 후 이용할 수 있다. 이용 권역은 서울시 전역이며 진료 및 치료목적으로 이용할 때에는 인접 12 개 시, 항공권 소지 시에는 인천국제공항까지 예외적으로 운행하고 있다. 신청접수는 필요한 시간에 바로 접수하는 '바로콜'과 휠체어를 이용하는 장애 정도가 심한 지체, 뇌병변 장애인을 대상으로 24 시간 전부터 접수 가능한 '전일 접수', 정기적으로 동일 시각에 동일 출발지에서 동일

목적지로 이동하는 고객을 위한 '정기접수'로 나뉜다. 이용요금은 5km 까지 기본요금 1,500 원이며 초과 시 10km 까지 2,900 원이고 10km 초과 시에는 km 당 70 원씩 추가된다. 시간 및 지역 할증은 없다.

○ 배차 기준은 전일 접수자 우선 배차이고 바로콜 접수 시 접수순서 20 점 + 대기시간 40 점 + 거리 30 점 합산 후 차량배차가 이루어진다. 운전자 퇴근 시간대에는 근거리에 있는 고객의 목적지가 차고지 방향과 유사한 경우 우선 배차되고 고객 대기시간이 2 시간 이상으로 긴 경우 먼 거리에서도 당겨서 배차된다.

1.1.2.2 대전시 장애인 콜택시 성공 요인

○ 대전시는 2019 년 복권기금 위원회에서 실시한 이용자 만족도 조사에 이어 2020 년 장애인콜택시 만족도에서 전국 1 위를 차지하였다.

○ 서울보다 약 3 년 늦은 2006 년에 운행대수 5 대로 장애인콜택시를 운행 개시하였다(대전교통약자이동지원센터).

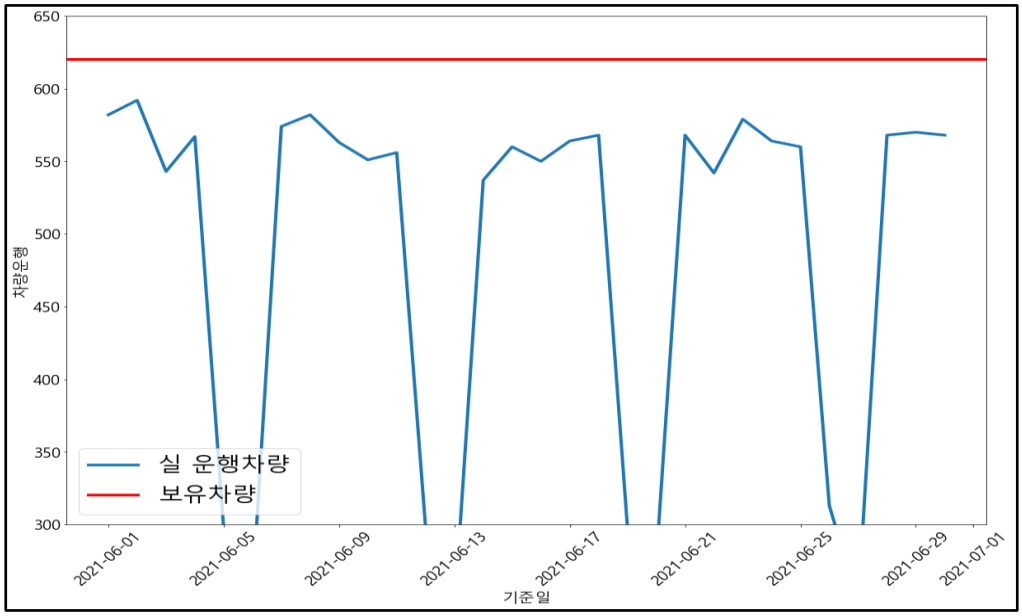
○ 현재는 특별교통수단(특장차) 86 대, 전용 임차 택시(일반차량) 90 대, 바우처택시(일반차량) 150 대로 총 326 대가 운영 중에 있다. 차량 이용 희망자는 출발지와 목적지를 실명으로 콜센터에 접수(전화, 인터넷, 모바일)하면 콜접수자에게 근접 차량이 배차되고 운전원이 이용자 확인 후 승차해서 목적지까지 이동지원 하는 방식으로 진행된다. 이때 비휠체어 회원은 이용 횟수 정책(일 6 회, 월 60 회)을 적용한다. 이용요금은 3km 까지 기본요금 1,000 원이고 초과 시 440m 당 100 원씩 추가된다.

○ 대전시 이동 약자 지원센터는 4 년 전부터 ‘예약제’를 없애고 서울의 바로콜과 유사한 ‘즉시 콜’ 방식으로만 운영하는 점을 이용자 만족도 향상의 주요 요인으로 설명한다. 서울시의 전일 접수자 우선 배치 제도를 없앤 것이다. 이 밖에도 2019 년부터 모든 차량 내부에 CCTV 를 설치한 것도 각종 민원 발생 감소의 요인으로 꼽힌다.[3]

1.1.2.3 서울시 장애인콜택시 문제점

대전광역시의 사례에서 알 수 있듯이 즉시 콜 방식이 이용자들의 만족도를 향상할 수 있는 요인이 될 수 있다. 즉시 콜 방식을 성공적으로 운영하기 위한 전제 조건으로 수요와 공급의 균형이 이루어져야 한다. 하지만 서울시 장애인콜택시의 바로콜 제도의 경우, 서울시에서 보유하고 있는 622 대(2021 년 기준)의 차량과 실제 운행되는 차량의 수에는 차이가 존재하여 수요량이 많은 시간대에는 빠른 배차가 현실적으로 어려운 실정이다.

[3] 최선중, “전국 장애인 콜택시 만족도 1 위 대전…비결은?”, KBS NEWS, 2021.02.25. https://news.kbs.co.kr/news/view.do?ncd=5126189

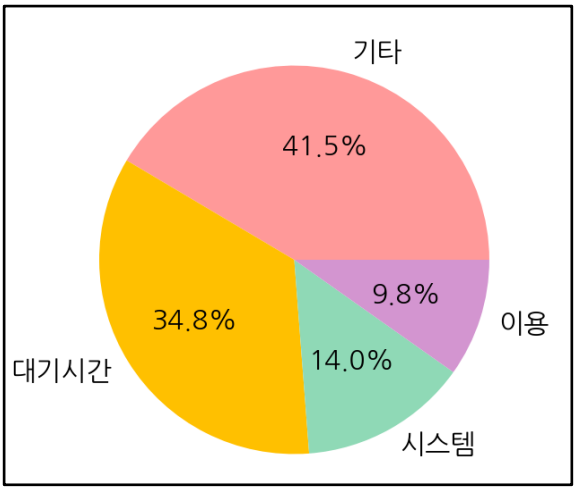


[그림 3] 보유차량과 실 운행차량

이러한 현상의 원인으로 주 52 시간 근무제 등의 이유로 이용자의 수요가 많은 시간대에 운전기사의 인력이 부족하다는 문제점을 전화 인터뷰를 통해 발견할 수 있었다.[4] 이는 결과적으로 이용고객의 대기시간 증가로 이어진다. 실제로 서울시설공단 시민의 소리 민원 데이터(2014 년~2021 년 7 월)를 분석한 결과, 민원 건수에서 대기시간에 대한 민원 건수는 34.8%로 기타 민원을 제외하면 가장 주요한 민원 유형으로 차지했다.[5]

[4] 2021.08.12 서울시설공단 장애인콜택시 운영처 전화인터뷰

[5] 임주찬, “대기만 2~3 시간 ‘장애인 콜택시’는 오늘도 예약 전쟁”, 오마이뉴스, 2019.08.22. http://www.ohmynews.com/NWS_Web/View/at_pg.aspx?CNTN_CD=A0002563980



[그림 4] 2014 년~2021 년 7 월 서울시설공단 시민의 소리 민원데이터

○ 서울시설공단에서 발표한 '장애인콜택시 종합현황철(2021 년 4 월)'에 기재된 '시간대별 대기시간 현황'에 따르면, 접수부터 도착까지 장애인콜택시 평균 대기시간은 26.5 분이고, 혼잡 시간대에는 1 시간 이상 소요된다는 점을 알 수 있다.[6]

6 이슬기, "기약없는 장애인콜택시, '불편은 장애인 몫?'", 에이블뉴스, 2021.06.17. http://www.ablenews.co.kr/News/NewsContent.aspx?CategoryCode=0014&NewsCode=001420210617092108288037

[그림 5] 서울시 민원데이터 워드클라우드

[그림 6] 대전시 민원데이터 워드클라우드

○ 서울시의 민원 데이터에서 가장 많이 등장한 키워드는 ‘기사’, ‘시간’, ‘배차’, ‘차량’, ‘전화’가, 대전시에서는 ‘기사’, ‘이용’, ‘접수’, ‘휠체어’, ‘교육’이 상대적으로 빈도가 높게 나타났다. 이것을 통해 서울시 장애인콜택시 게시판에서 대기시간과 관련된 ‘시간’, ‘배차’ 키워드가 더욱 빈번하게 언급되었다는 것을 알 수 있다.

1.2 분석목적 및 방향

○ 교통약자 이동 수요가 늘어나면서 특별교통수단의 보급도 매년 증가하고 있으나 장애인이 특별교통수단을 이용하기에는 실제 운영되는 차량 공급량의 부족 등으로 대기시간이 많이 소요되고, 서비스의 질적 수준도 미흡한 실정이다.

○ 앞서 살펴본 것처럼 서울시 장애인콜택시 관련 민원 중 가장 높은 비율을 차지하는 유형은 '대기시간이 길다'라는 것이다. 이상적으로 이용자의 대기시간 감소를 위해 가장 효과적인 방법은 콜택시 차량 대수를 증가시키는 것이다. 그러나 재원은 한정적이기 때문에 본 조는 수요량을 예측하여 공급량을 조절하고, 공급량의 한계에 의해 불균형이 발생하는 경우 시뮬레이션을 활용하여 대기시간 감소를 위한 증차 효과성이 최대가 되는 수준을 제시하여 서비스의 질적 수준을 향상하고자 한다.

1.3 분석 결과 활용 방안

○ (정책 활용) 교통약자인 장애인의 자유로운 이동성 보장으로 교통 복지서비스 향상

○ (생활 개선) 시행중인 장애인콜택시 시스템 대기시간 관련 불만 민원 개선으로 이용자 만족도 향상

○ (이용 효율화) 차량 및 차고지 최적 배치와 증차 적정 수준 제시를 통한 자원·예산 낭비 최소화

2. 분석 데이터

2.1 분석 데이터 목록

구분	분석 데이터	기간	제공기관
장애인 콜택시	장애인 콜시스템 현황	17~21.6	서울시설공단 장애인 콜택시 (https://www.sisul.or.kr/open_content/calltaxi/)
	장애인 콜택시 일별 이용 현황	17~21.6	서울시설공단 장애인 콜택시 (https://www.sisul.or.kr/open_content/calltaxi/)
참여성	서울시 동별 장애인 현황	17~21.6	서울 열린 데이터 광장 (https://data.seoul.go.kr/)
	서울시 생활인구 현황	19~21.6	서울 열린 데이터 광장 (https://data.seoul.go.kr/)
	서울특별시 사회복지시설	17~21.6	공공데이터포털 (https://www.data.go.kr/)
기상정보	서울특별시 기상관측자료	17~21.6	기상자료 개방포털 (https://data.kma.go.kr/cmmn/main.do)

교통정보	서울시 지점별 일자별 교통량	17~21.6	서울시 교통정보 시스템 (https://topis.seoul.go.kr/)
	서울시 주차장 정보 표준데이터	17~21.6	서울 열린 데이터 광장 (https://data.seoul.go.kr/)
민원조사	서울시설공단 시민의 소리 민원데이터	17~21.6	서울시설공단 장애인 콜택시 (https://www.sisul.or.kr/open_content/calltaxi/)
	대전교통약자 이동지원센터 자유게시판	17~21.6	대전교통약자 이동지원센터 (https://djcall.or.kr/)

2.2 데이터 상세 설명

구분	분석 데이터	데이터 형식	생성주기
공공데이터	서울시동별장애인현황	csv	연도별
	서울시주차장정보표준데이터	csv	연도별
	서울특별시 사회복지시설	csv	월별
	서울시 지점별 일자별 교통량	csv	월별

	서울특별시 기상관측자료	csv	실시간
	장애인 콜시스템	csv	일별
	장애인 콜택시 일별 이용 현황	csv	일별
	서울시설공단 시민의 소리 민원데이터	csv	실시간
	대전교통약자 이동지원센터 자유게시판	csv	실시간
	서울시 차량 이동률 top10 구	shp	연간
	동별 장애인 콜택시 이용 출발지 비중	shp	월간
	동별 장애인 콜택시 이용 목적지 비중	shp	월간
	서울시 내 공영주차장 버퍼	shp	비정기 (자료변경시)
	서울시 장애인 사회복지시설 현황	shp	월간

2.2.1 서울시 동 별 장애인 현황 [.CSV]

○ 서울 열린 데이터 광장에서 제공하는 자료로, 장애 유형 별로 서울시 구별·동별 장애인 남자 계, 여자 계, 총 계 내역이 포함되어 있다.

기간	자치구	동	합계	합계	합계	장애유형	장애유형	장애유형	장애유형	장애유형	장애유형
기간	자치구	동	합계	합계	합계	지체	지체	지체	뇌병변	뇌병변	뇌병변
기간	자치구	동	계	남자	여자	계	남자	여자	계	남자	여자
2020	합계	합계	394,190	228,386	165,804	172,606	97,303	75,303	40,905	24,244	16,661
2020	종로구	소계	6,015	3,488	2,527	2,586	1,388	1,198	582	371	211
2020	종로구	사직동	284	167	117	124	76	48	34	14	20
2020	종로구	삼청동	110	58	52	44	23	21	13	7	6
2020	종로구	부암동	317	181	136	125	64	61	27	16	11
2020	종로구	평창동	538	290	248	214	110	104	57	34	23
2020	종로구	무악동	328	181	147	116	56	60	30	21	9

2.2.2 서울시 지점별·일자별 교통량 데이터 [.CSV]

○ 서울 열린 데이터 광장에서 제공하는 자료로, 각 지점에서 측정한 교통 이동 방향과 교통량을 시간대별로 기록된 데이터이다.

일자	요일	지점명	지점번호	방향	구분	0시	1시	2시
20210101	금	성산로(금화터널	A-01	유입	봉원고가차도->ᄒ	297	274	150
20210102	토	성산로(금화터널	A-01	유입	봉원고가차도->ᄒ	189	135	103
20210103	일	성산로(금화터널	A-01	유입	봉원고가차도->ᄒ	258	146	107
20210104	월	성산로(금화터널	A-01	유입	봉원고가차도->ᄒ	209	98	116
20210105	화	성산로(금화터널	A-01	유입	봉원고가차도->ᄒ	296	209	150
20210106	수	성산로(금화터널	A-01	유입	봉원고가차도->ᄒ	344	213	160
20210107	목	성산로(금화터널	A-01	유입	봉원고가차도->ᄒ	182	106	120
20210108	금	성산로(금화터널	A-01	유입	봉원고가차도->ᄒ	240	174	126
20210109	토	성산로(금화터널	A-01	유입	봉원고가차도->ᄒ	329	231	166
20210110	일	성산로(금화터널	A-01	유입	봉원고가차도->ᄒ	250	166	105
20210111	월	성산로(금화터널	A-01	유입	봉원고가차도->ᄒ	211	147	100

2.2.3 서울특별시 사회복지시설 [.CSV]

○ 공공 데이터 포털에서 제공하는 자료로, 시군구별 장애인이 이용하는 유형의 복지시설의 정보가 담긴 데이터이다.

통계년월	통계시도명	통계시군구명	복지시설유형코드	복지시설수
201701	서울특별시	종로구	장애인거주시설	7
201701	서울특별시	종로구	장애인지역사회재활시	4
201701	서울특별시	중구	장애인거주시설	2
201701	서울특별시	중구	장애인지역사회재활시	8
201701	서울특별시	중구	장애인생산품판매시설	1
201701	서울특별시	용산구	장애인거주시설	2
201701	서울특별시	용산구	장애인지역사회재활시	9
201701	서울특별시	성동구	장애인거주시설	4
201701	서울특별시	성동구	장애인지역사회재활시	4

2.2.4 서울시 주차장 정보 표준데이터 [.CSV]

○ 서울 열린 데이터 광장에서 제공하는 자료로, 서울시에 존재하는 공영주차장 유형과 가격에 대한 정보가 담긴 데이터이다.

PARKING_	PARKING_	OPERATIO	WEEKDAY_	WEEKDAY_	WEEKEND_	WEEKEND_	HOLIDAY_	HOLIDAY_	SATURDA	HOLIDAY_	FULLTIME_	RATES
초안산근린	노외 주차	시간제 주	900	1900	900	1900	900	1900	무료	무료	0	0
마들스타디	노외 주차	시간제 주	0	2400	0	2400	0	2400	무료	무료	100000	150
마장동공영	노외 주차	시간제 주	0	0	0	0	0	0	유료	유료		
영등포여고	노외 주차	시간제 주	0	2400	0	2400	0	2400	무료	무료	65000	50
당산근린공	노외 주차	시간제 주	0	2400	0	2400	0	2400	무료	무료	100000	150
대림운동장	노외 주차	시간제 주	0	2400	0	2400	0	2400	무료	무료	65000	50
휘경마을	노외 주차	시간제 +	900	1900	900	1900	900	1900	무료	무료		1000
중산공영주	노외 주차	시간제 +	900	1900	900	1900	900	1900	무료	무료		1000
방학동 도	노외 주차	시간제 +	900	2200	900	2200	900	2200	무료	무료	0	100
은평평화공	노외 주차	시간제 주	0	2400	0	2400	0	2400	무료	무료	120000	150
압구정 42	노외 주차	시간제 주	900	2100	900	2100	900	2100	무료	무료	200000	400

2.2.5 서울시 기상관측자료 [.CSV]

○ 기상자료개방포털에서 제공하는 자료로, 각 구별 기상 관측자료가 담긴 데이터이다.

일시	지점명	기온(℃)	풍속(m/s)	강수량(mm)
2017-01-01 0:00	강남구	1.7	1.3	0
2017-01-01 0:00	강동구	-0.2	2.1	0
2017-01-01 0:00	강북구	1.7	0.6	0
2017-01-01 0:00	강서구	3.2	1.8	0
2017-01-01 0:00	관악구	0.3	1.6	0
2017-01-01 0:00	광진구	1.3	0.3	0
2017-01-01 0:00	구로구	0.2	0.6	0
2017-01-01 0:00	금천구	2.3	0.6	0
2017-01-01 0:00	노원구	-2.6	0.6	0
2017-01-01 0:00	도봉구	0.1	0.5	0
2017-01-01 0:00	동대문구	1.7	1	0
2017-01-01 0:00	동작구	2	0.4	0
2017-01-01 0:00	마포구	2.6	2.4	0

2.2.6 장애인 콜시스템 [.CSV]

○ 서울시설공단 복지경제 내 장애인 콜택시에서 제공하는 자료로, 구·시간 별 장애인 콜택시의 이용횟수와 대기시간 등 자료가 담긴 데이터이다.

time	startpos1	복지시설수	num	wait
2017-01-01 0:00	강남구	40	1	39.62
2017-01-01 0:00	강동구	31	0	0
2017-01-01 0:00	강북구	22	0	0
2017-01-01 0:00	강서구	47	0	0
2017-01-01 0:00	관악구	26	0	0
2017-01-01 0:00	광진구	13	0	0
2017-01-01 0:00	구로구	28	0	0
2017-01-01 0:00	금천구	12	0	0
2017-01-01 0:00	노원구	46	0	0
2017-01-01 0:00	도봉구	24	0	0
2017-01-01 0:00	동대문구	5	0	0
2017-01-01 0:00	동작구	17	0	0

2.2.7 장애인 콜택시 일별 이용 현황 [.CSV]

○ 서울시설공단 복지경제 내 장애인 콜택시에서 제공하는 자료로, 장애인콜택시의 전반적인 운영 정보가 담긴 데이터이다.

기준일	차량운행	희망건	탑승건	평균대기시간	평균요금	평균승차거리
2015-01-01	213	1023	924	23.2	2427	10764
2015-01-02	420	3158	2839	17.2	2216	8611
2015-01-03	209	1648	1514	26.2	2377	10198
2015-01-04	196	1646	1526	24.5	2431	10955
2015-01-05	421	4250	3730	26.2	2214	8663
2015-01-06	417	3991	3633	23.6	2211	8545
2015-01-07	410	4085	3676	24.7	2230	8646
2015-01-08	419	4030	3728	21.2	2231	8683
2015-01-09	424	4167	3813	21.8	2215	8506
2015-01-10	215	1916	1645	41.2	2447	11123

2.2.8 서울시설공단 시민의 소리 민원데이터 [.CSV]

○ 서울시설공단 장애인 콜택시 시민의 소리에 올라온 민원글로, 서울시에서 운행하는 장애인 콜택시에 대한 의견이 담긴 데이터이다.

제목	내용
77라3586 인정호	인정호운전자님을 칭찬 합니다 항상 반갑게 맞아주셔서 기분이 좋고 이동중에도 웃는 모습을 보면서 저도 좋
72우1553최장철	2021년7월7일오전에 경희의료원에서양주하늘소풍수목장왕복콜불어서 71버0823차를기다리면서 우연히최
전일 접수 예약지	■ 예전과 같이 전일 접수 예약제(7시, 8시, 10시)에 맞는 차량 운행을 부탁드립니다. 전일 접수 예약제는 매일
78무4545김영자	7월22일 오후 3시경서울재활병원에서방학동까지 오는 차안에 복지카드를 두고 내렸는데두고 내린줄도 저
서울32자1075박	2021년6월16일하고6월17일2일동안거여동에서 신내동갈때배차됐는데7시차인박순배기사님께서배차됐는데
77라 7655 기사님	아이 치료를 위해 요샌 일주일에 두번씩 콜을 이용하고 있습니다~~ 너무너무 더워진 날씨에 도착해 내렸다 탓
75마5794설평권	오늘오후11시41분경에 면목동중랑구민회관쪽에서 면목동명인한의원에갈때 오래만에배차됐는데설평권기사
7월 19일 월요일	7월 19일 월요일 첫 콜을 타고 난 후 다시 한 번 더 장애인 콜택시 운영 관계자 분들과 기사님들에게 감사함을
누구의기준인가?	매번 장애인 콜택시 볼때마다 속이 쓰립니다.통화해서 물어보면 서울시에서 정해는 기준이라나?하루에
면목차고지휴무?	2021년6월29일에저녁에 거여동에서면목동올때콜불었는데 몇분기다렸는지까먹었지만 고맙게면목차고지휴

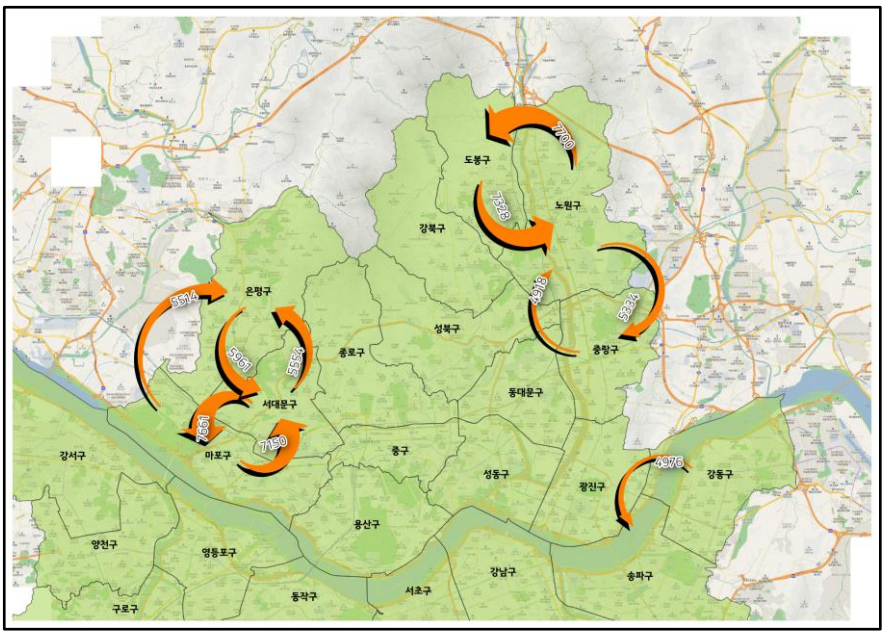
2.2.9 대전교통약자 이동지원센터 자유게시판 [.CSV]

○ 대전교통약자 이동지원센터 자유게시판에 올라온 민원글로, 대전시에서 운행하는 장애인 콜택시에 대한 의견이 담긴 데이터이다.

제목	내용
자유게시판 안내	안녕하세요.대전교통약자이동지원센터입니다. 현재 자유게시판은 센터 차량을 이용하시는 회원님들의 의견
장애인콜택시	안녕하세요봉산휴먼시아2단지아파트 에서 중구 대흥동까지 이용요금 얼마정도 나오나요?
기사분의 마인드	오늘 오후 2시 55분에 70부 9918 차를 이용해서 집에서 둔산동까지 갔습니다.차에 타자 2점식 벨트를 꺼내서
e대전장애인콜택	아래 요금표는 경기도 성남시와 하남시 경기도 일산등 경기도지역 요금 표입니다. - 누가보아도 괜찮은 요금
기사분들 마스크	안녕하세요.2021년 6월 30일 전용임차택시 이용 중 발생한 일입니다.오전 7:00 (61바 3519) 에 이용한 전용임
안녕하세요? 사랑	요즘 장애인 콜택시(장콜) 을 타면 사무실에서 열심히 교육을 했다고 느껴 집니다.기사분들이 의식을 가지고
생명과 관련된 문	안녕하십니까? 선진 교통 문화를 위해 애쓰시는 관계자 분들의 노고에 감사합니다. 그동안 느껴왔던 문제이
장애인콜택시	장애인콜택시 접수하면 금방 오나요?
안녕하세요, 선진	어제 가까운 거리를 왕복했습니다. 기사분들이 가장 기본적인 안전띠와 안전 운전에 심각한 인식 부족에 글을
근거리 배차 원칙	안녕하세요 평소 장콜을 감사하며 이용하는 회원입니다.근데 오늘은 너무 ㅎ황당한 일이 있어서 이렇게 글을

2.2.10 서울시 차량 이동률 top10 구 [.SHP]

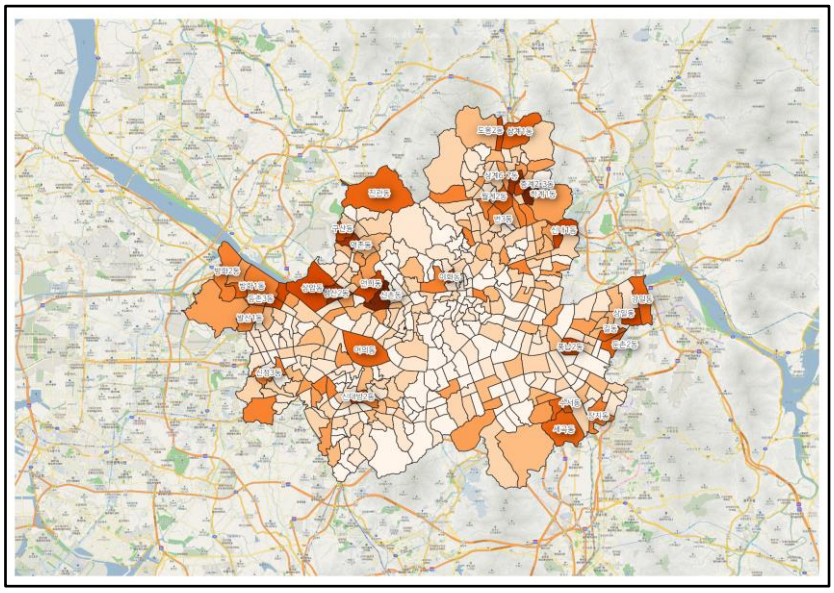
○ 서울시설공단 장애인 콜택시에서 제공하는 자료로, 구에서 다른 구로 차량 이동률 정보를 나타내는 데이터이다.



[그림 7] 서울시 차량 이동

2.2.11 동별 장애인 콜택시 이용 출발지 비중 [.SHP]

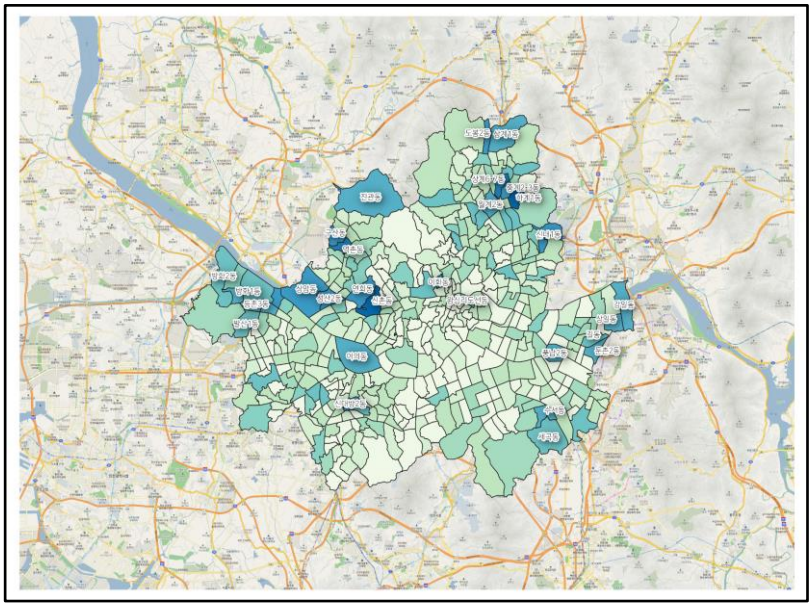
○ 서울시설공단 장애인 콜택시에서 제공하는 자료로, 콜택시 이용 출발지 비중이 담긴 데이터이다.



[그림 8] 서울시 동별 출발지 비중

2.2.12 동별 장애인 콜택시 이용 목적지 비중 [.SHP]

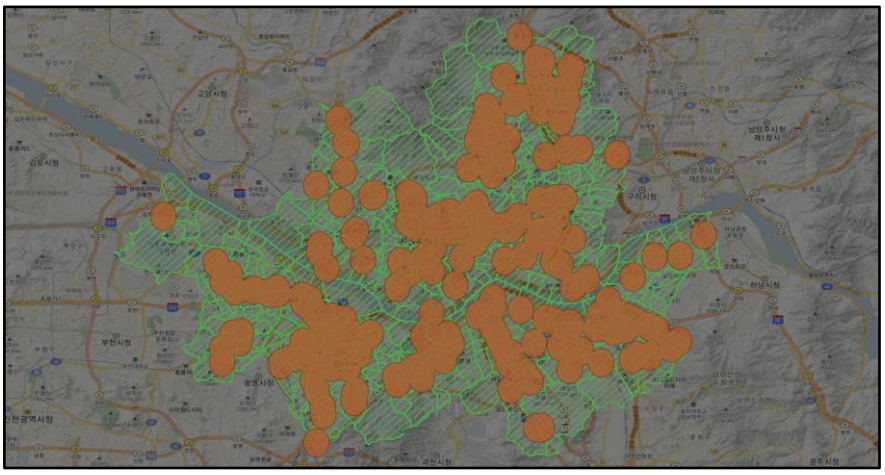
○ 서울시설공단 장애인 콜택시에서 제공하는 자료로, 콜택시 이용 목적지 비중이 담긴 데이터이다.



[그림 9] 서울시 동별 목적지 비중

2.2.13 서울시 내 공영주차장 버퍼 [.SHP]

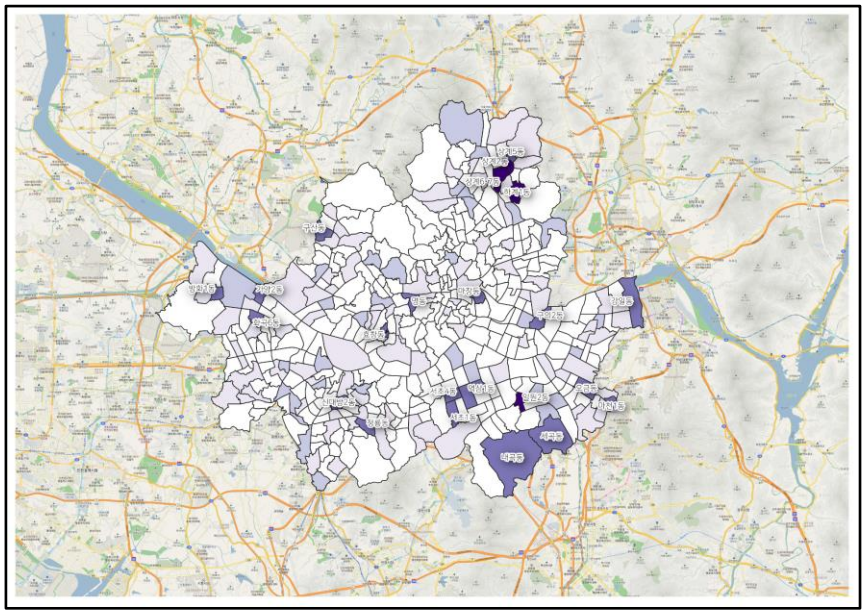
○ 서울 열린 데이터 광장에서 제공하는 자료로, 공영주차장의 위치를 버퍼로 나타내는 데이터이다.



[그림 10] 서울시 공영주차장 버퍼

2.2.14 서울시 장애인 사회복지시설 현황 [.SHP]

○ 공공데이터포털에서 제공하는 자료로, 구에 있는 사회복지시설 현황이 담긴 데이터이다.



[그림 11] 서울시 장애인 복지시설 현황

2.3 데이터 정제 방안

2.3.1 결측값 대체

2.3.1.1 기상관측자료

○ 시간 단위로 정렬된 기상관측자료의 결측값을 이전 시간의 기상 값으로 대체

2.3.1.2 서울시 지점별 일자별 교통량

○ 각 구의 여러 지점에서 산출된 교통량은 각 구의 최대 교통량을 가진 지점값으로 선정하고 일정 날짜의 시간대 교통량 결측값은 해당 날짜의 평균으로 대체

2.3.2 이상치 확인

○ 대기시간이 2 시간 이상인 데이터들은 2 시간으로 일괄 적용

○ 서울시설 공단이 보유한 622 대 이상인 데이터들을 622 대로 일괄적용

○ 나머지 컬럼의 Q1, Q3 에서 3 * IQR 이상 떨어진 값을 이상치로 하여 이상치를 확인한 결과 이상치가 발견되지 않음

2.3.3 라벨 인코딩

○ 서울시 각 구 데이터에 대해 라벨 인코딩 적용

2.3.4 표준 정규화

○ 다른 데이터 값에 비해 수치가 큰 서울시 지점별 일자별 교통량 데이터와 서울시 동별 장애인 현황 값은 정규화 값으로 대체

2.3.5 불용어 제거

○ 민원 데이터에서 명사를 제외한 불용어들을 정제한 후 워드클라우드 생성

3. 분석 프로세스

3.1 데이터 준비

○ 장애인 콜택시의 대기시간 지연 문제를 민원 문제와 차량 배치 문제로 나누어 확인

○ 차량 배치 문제는 대기시간에 영향을 미칠 수 있는 요인에 대한 데이터를 수집 및 정제하여 사용 가능한 데이터 셋 준비

○ 민원 문제는 서울시설공단 시민의 소리 민원 데이터와 대전 교통약자 이동지원센터 자유게시판 크롤링을 통해 데이터 수집

3.2 분석 및 모델링

[그림 12] 분석프로세스

○ EDA 를 통해 각 요인의 결측치와 이상치를 제거하고 콜택시 이용량과의 상관분석을 실시한 후 관련성이 높은 요인들을 통해 우선 배치를 위한 점수 산정

3.3 결과 도출 및 시각화

○ 구별 시간별 장애인 콜택시 수요량 예측 모델을 개발해 필요지역에 우선 배치

3.3.1 분석 내용 및 방법

3.3.1.1 요인도출

3.3.1.1.1 요인 근거

○ 장애인 콜택시에 영향을 미치는 요인을 서울시설공단을 기준으로 외부와 내부로 나누어 판단 가능

○ 날씨, 시간대(외부): 서울 택시 분석 리포트에 따라 택시의 공급은 한정돼 있으나 수요는 시간대, 날씨 등의 영향으로 변동이 심함

○ 복지시설 현황(외부): 구 내 복지시설이 많다는 것은 구 내 장애인 비중이 높다는 것을 의미. 따라서 장애인 콜택시 수요량에 영향을 미칠 것으로 예상

○ 대기시간(내부): 장애인 콜택시 관련 민원 데이터를 분석한 결과 대기 시간 관련 민원이 가장 많음 (차후 수요량에 영향을 미칠 것으로 예상)

3.3.1.1.2 요인 인식

○ 요인들을 속성에 따라 분류

○ 인구정보: 구별 장애인수 데이터

○ 교통정보: 교통량 데이터, 주차장 데이터, 대기시간

○ 기상정보: 계절, 온도, 강수량 데이터

○ 공휴일: 주말, 공휴일 데이터

3.3.1.1.3 가설 설정

○ 교통량은 장애인 콜택시 대기시간에 영향을 미친다.

○ 기상상황이 장애인 콜택시 수요량에 영향을 미친다.

○ 구내 장애인 복지시설 수가 콜택시 수요량에 영향을 미친다.

○ 구별 유동인구수가 콜택시 수요량에 영향을 미친다.

○ 구별 장애인 인구수가 콜택시 수요량에 영향을 미친다.

○ 공휴일, 주말 여부에 따라 콜택시 수요량에 차이가 있다.

3.3.1.2 상관분석

3.3.1.2.1 요인 변수들에 대해 장애인 콜택시 수요량과의 상관 분석

○ 콜택시 수요량과의 관련성을 파악하기위해 상관 분석을 실시

○ 상관 분석을 활용해 수요량과 관련성이 적은 변수 제거

○ 상관 분석 결과를 히트맵으로 시각화

3.3.1.2.2 요인 상관 분석 결과에 따른 다중공선성 문제 해결

○ 변수간의 분산팽창지수(VIF)를 구하여 모델의 다중공선성 파악

○ VIF 가 10 보다 큰 변수들이 발견되지 않음

3.3.1.3 구별 시간별 콜택시 수요량 예측 모델 개발

○ 통합된 데이터들을 바탕으로 머신러닝 진행

- LightGBM, Catboost, XGB regressor 세가지 모델들을 앙상블

○ 과적합 방지를 위해 5-fold cross validation 진행

○ 모델 최적화를 위해 random search 를 진행하여 최적의 하이퍼파라미터 도출

3.3.1.4 구별 시간별 콜택시 대기시간 예측 모델 개발

○ 이용자가 많아 실질적으로 대기시간이 많이 발생하는 시간대인 오전 7 시에서 오후 6 시까지 데이터로 한정

○ 통합된 데이터들을 바탕으로 머신러닝 진행

- LightGBM, Catboost, XGB regressor 세가지 모델들을 앙상블

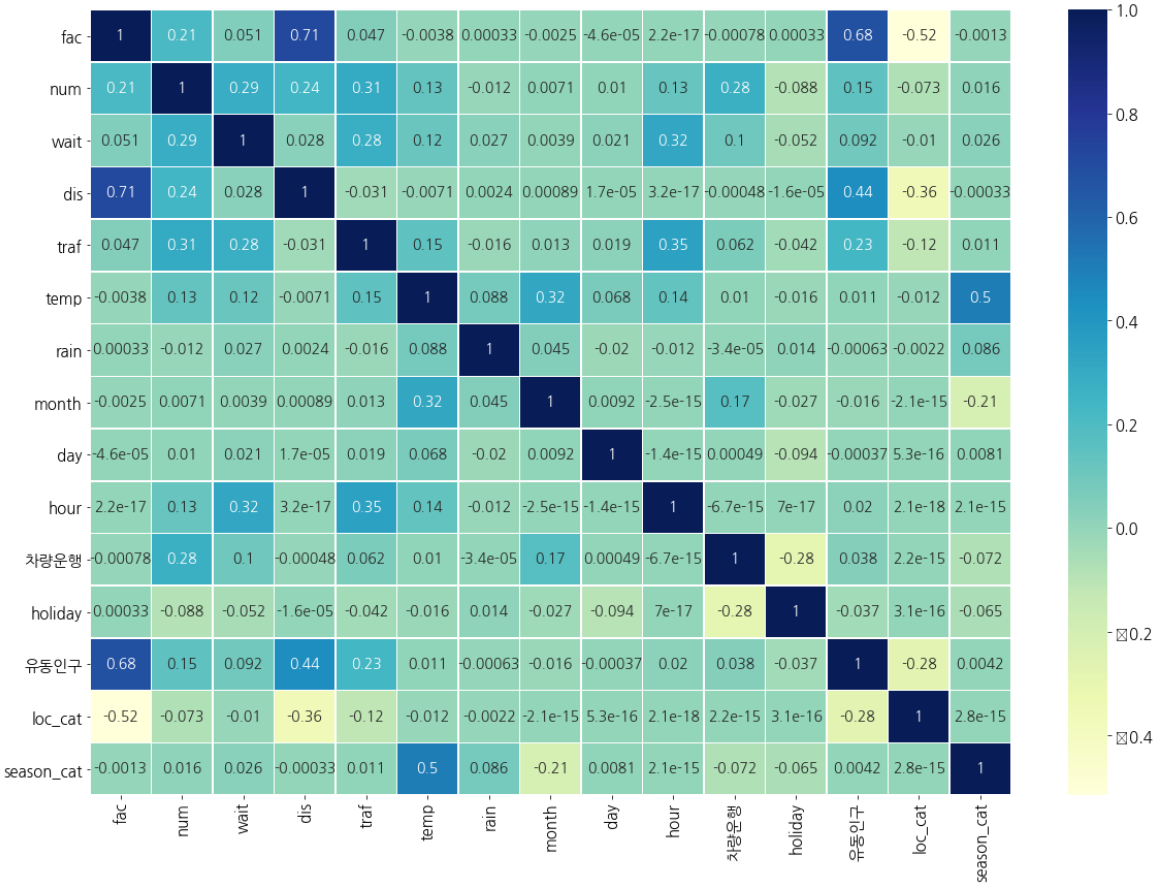
○ 과적합 방지를 위해 5-fold cross validation 진행

○모델 최적화를 위해 random search 를 진행하여 최적의 하이퍼파라미터 도출

4. 분석결과

4.1 EDA (탐색적 데이터 분석) 및 현황분석

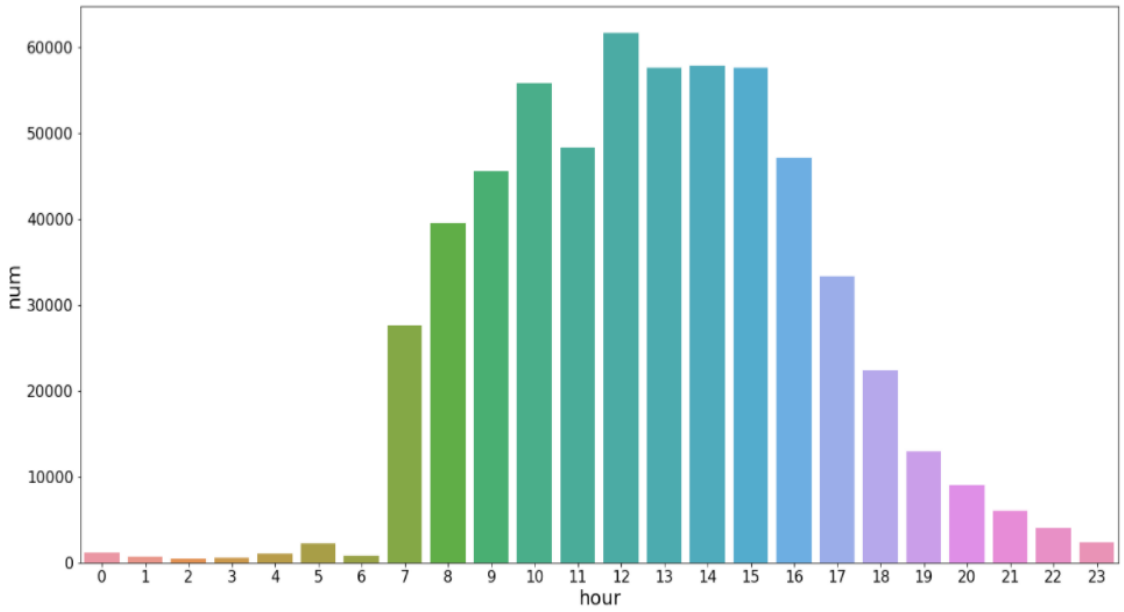
4.1.1 EDA



[그림 13] 수요량 예측 모델 상관관계 히트맵

○ 히트맵을 통해 상관관계를 확인한 결과 종속변수인 num(수요량)에 독립변수인 fac(복지시설수), wait(대기시간), dis(장애인수), traf(교통량)가 주요한 인자임을 확인할 수 있다.

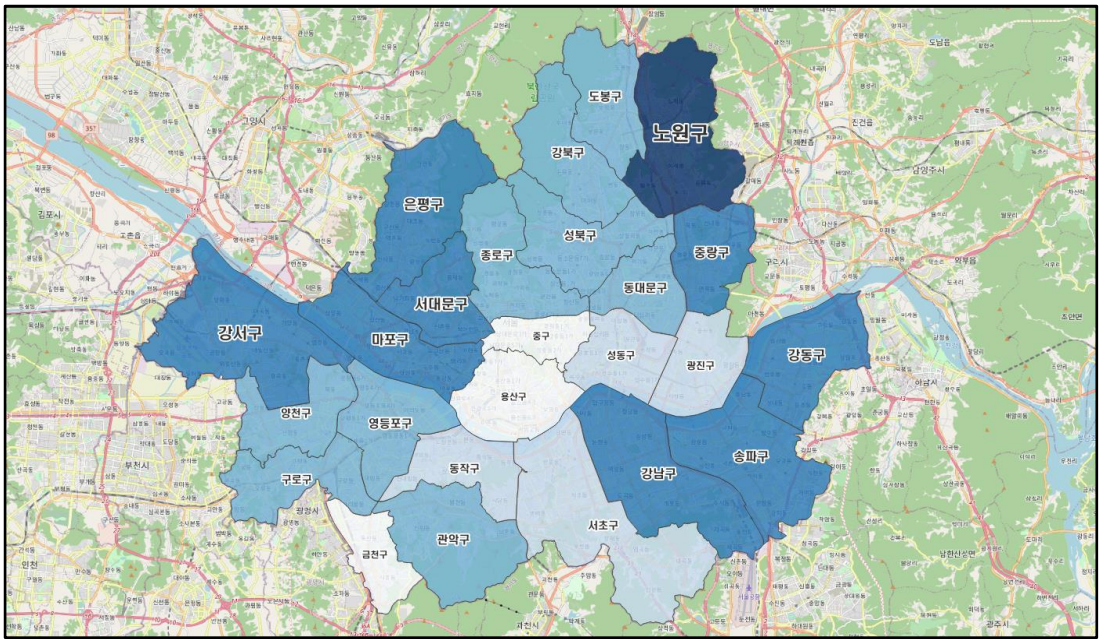
4.1.2 현황분석



[그림 14] 2021 년 시간대별 콜택시 이용량

○ 그래프를 봤을 때 7 시에서 18 시 사이의 이용량이 가장 많으며, 특히 점심시간대 콜택시 이용량이 가장 높은 것을 알 수 있다.

○ 7 시에서 18 시 이외의 시간대에서는 상대적으로 이용량이 적다는 것을 확인할 수 있다.

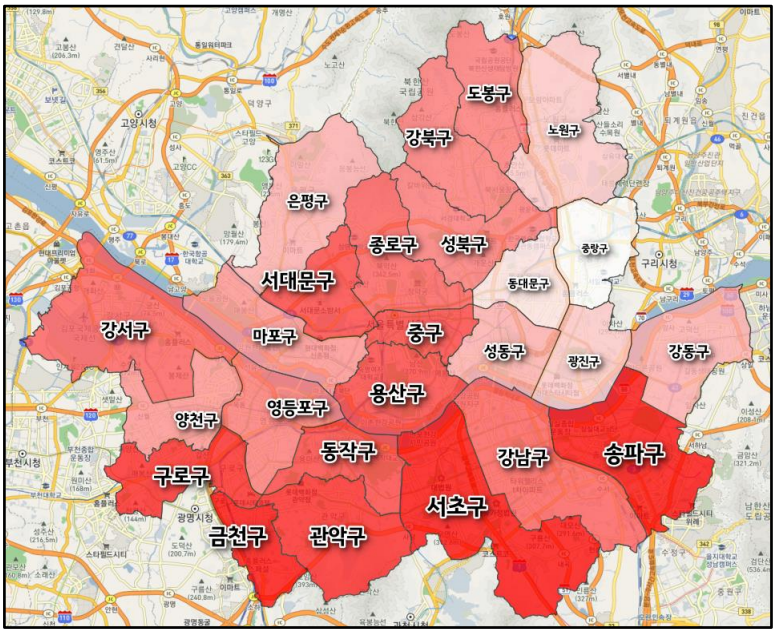


[그림 15] 2021 년 구별 장애인 콜택시 이용량 현황

○ 2021 년 구별 장애인 콜택시 이용량을 누적해본 결과, 노원구가 162,467 건으로 가장 많은 이용량을 기록하였다.

○ 노원구를 선두로 강서구, 은평구, 서대문구가 그 뒤를 이었다.

○ 용산구, 중구, 금천구가 가장 적은 장애인 콜택시 이용량을 기록하였다.



[그림 16] 2021 년 구별 장애인 콜택시 대기시간 현황

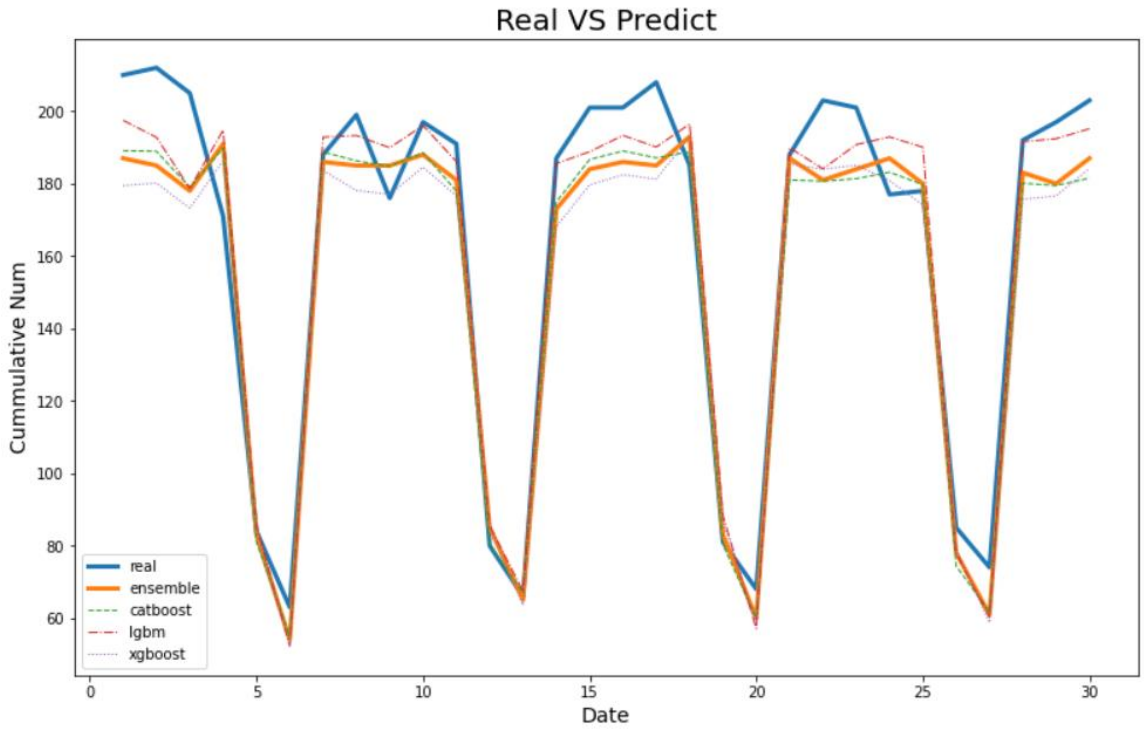
○ 2021 년 구별 장애인 콜택시 대기시간을 비교하기 위해, $\frac{\Sigma 이용량 \times 대기시간}{하루\ 전체\ 이용량}$수식을 통해 구별 평균 대기시간을 산출하였다.

○ 서초구가 평균 대기시간 약 42 분으로 가장 긴 대기시간이 발생했고 송파구, 금천구, 관악구가 그 뒤를 이었다.

○ 노원구, 광진구, 동대문구, 중랑구는 평균 대기시간 약 35 분 미만으로 가장 짧은 대기시간을 기록하였다.

4.2 머신러닝 기반 수요량 및 대기시간 예측 모델 개발

4.2.1 수요량 예측 모델



[그림 17] 2021 년 6 월 강남구 실제 장애인 콜택시 이용량과 모델 예측 수요량 값

○ 머신러닝 기반 수요량 예측 모델을 개발하기 위해 2020 년 1 월부터 2021 년 5 월까지의 데이터를 train 데이터셋, 2021 년 6 월 한 달 간의 데이터를 test 데이터셋으로 분리하여 학습을 진행하였다.

○ 복지시설 수, 대기시간, 구별 장애인 수, 구별 교통량, 기온, 시간대, 운행된 차량 수, 휴일 여부, 유동인구 등이 학습에 활용되었다.

○ 학습을 위해 선정된 모델은 CatBoost, XGBoost, LightGBM 세 가지 모델이며, 각각 학습을 진행한 후 앙상블하여 최종 예측값을 산출하였다.

○ 앙상블을 진행하기 전, Random Search 를 통해 세 개의 모델의 최적 하이퍼파라미터를 5-fold 방식을 사용하여 산출하였다.

○ 앙상블 결과, 평가지표 MAE(평균절대오차) 1.484 / RMSE (평균 2 제곱근오차) 2.705 / R-square(결정계수) 0.905 의 성능을 보이며 단일 모델 대비 더욱 개선된 수치를 산출할 수 있었다.

4.2.2 대기시간 예측 모델

○ 머신러닝 기반 대기시간 예측 모델을 개발하기 위해 2020 년 1 월부터 2021 년 5 월까지의 데이터를 train 데이터셋, 2021 년 6 월 한 달 간의 데이터를 test 데이터셋으로 분리하여 학습을 진행하였다.

○ 이용량이 없는 시간대는 대기시간 또한 존재하지 않기 때문에 장애인 콜택시의 주 이용시간인 07~18 시의 데이터를 선정하였다.

○ 복지시설 수, 콜택시 이용량, 구별 장애인 수, 구별 교통량, 기온, 시간대, 운행된 차량 수, 휴일 여부, 유동인구 등이 학습에 활용되었다.

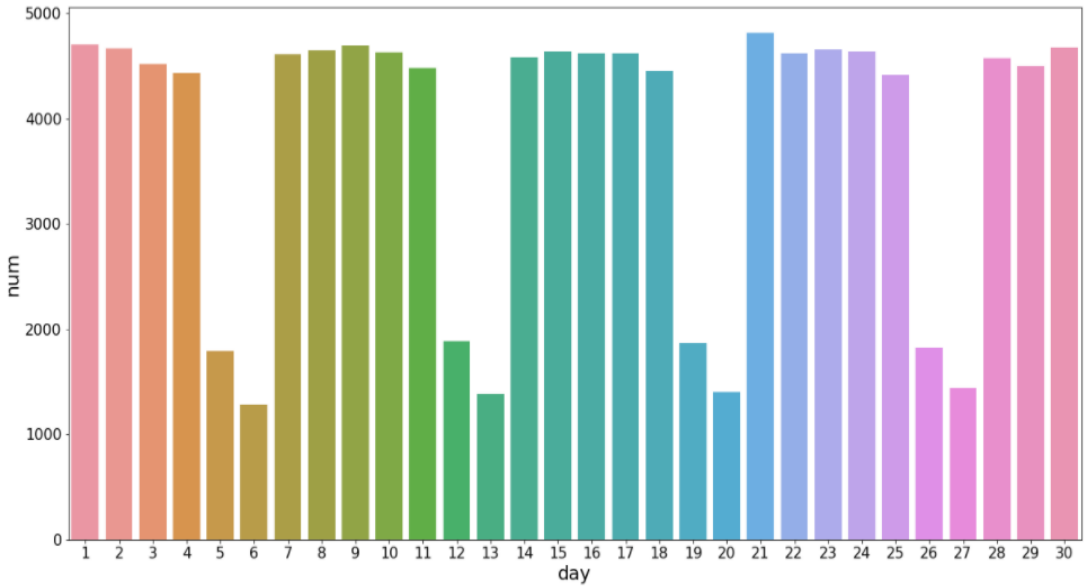
○ Random Search 를 통해 CatBoost, XGBoost, LightGBM 세가지 모델의 최적 하이퍼파라미터를 5-fold 방식을 사용하여 산출하였다.

○ 세가지 모델 앙상블 결과, CatBoost 단일 모델보다 좋지 않은 성능 수치를 보여주어 CatBoost 단일 모델로 선정하였다.

○ CatBoost 모델은 MAE(평균절대오차) 9.418 / RMSE(평균제곱근오차) 14.510 / R-square(결정계수) 0.536 의 성능을 보였다.

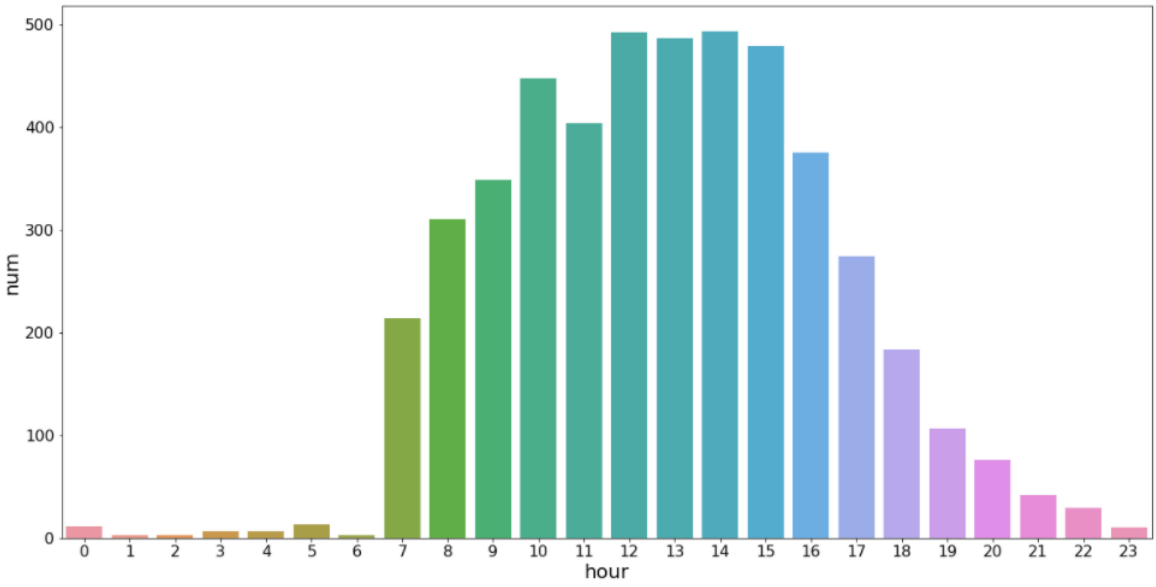
○ 해당 모델을 추후 당일 운행된 차량 수의 수치를 조정하는 증차 시뮬레이션에 활용한다.

4.3 모델 기반 차량배치 우선순위 지역 시각화



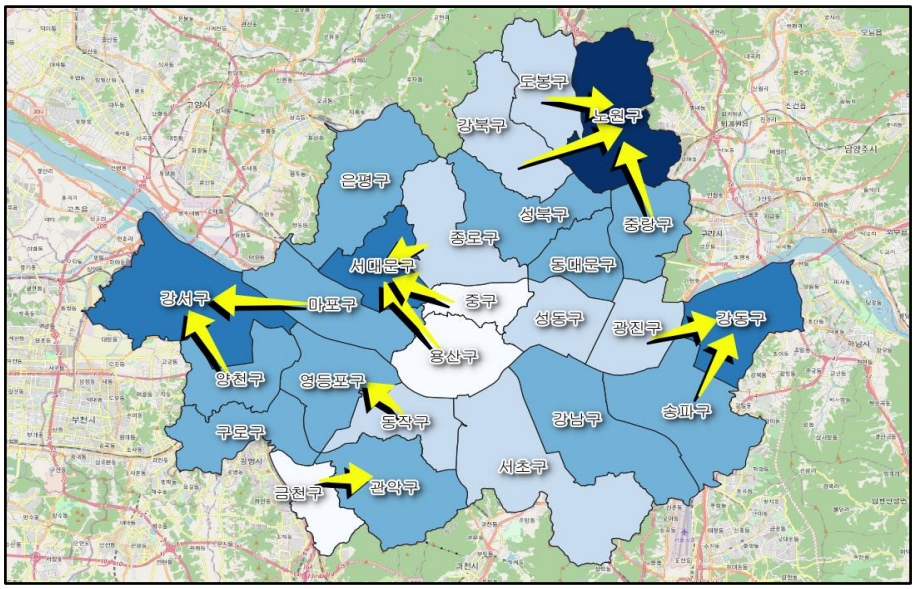
[그림 18] 2021 년 6 월 일별 콜택시 이용량

○ 6 월 중 콜택시 이용량이 가장 많은 날은 21 일이다.



[그림 19] 2021 년 6 월 21 일 시간대별 콜택시 이용량

○ 6 월 21 일 장애인콜택시 이용량이 가장 많은 시간은 12 시이다.



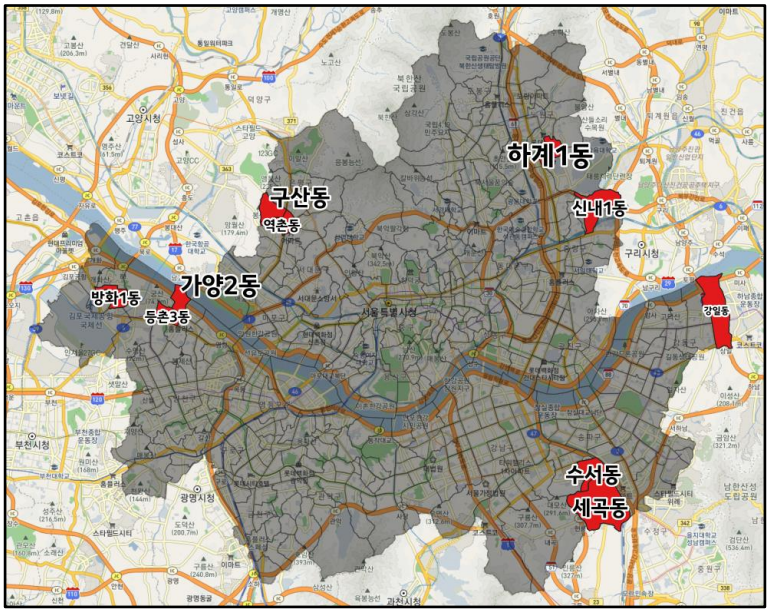
[그림 20] 2021 년 6 월 21 일 오후 12 시의 구별 수요량 예측

○ 앞선 그래프를 토대로 이용량이 가장 많은 시점(2021 년 6 월 21 일)의 콜택시 우선순위 분석 결과 노원구, 강서구, 서대문구, 강동구 순이었다.

○ 해당 시각화 자료를 통해 구별 수요와 공급의 균형을 맞추어 대기시간을 감소시킬 수 있을 것이다.

○ 예를 들어 수요량이 많은 시간대에는 상대적으로 수요량이 적은 주변 구에서 콜택시를 이동시키는 방법을 고안할 수 있을 것이다.

4.4 차고지 최적 입지 선정



[그림 21] 차고지 최적 입지

4.4.1 순위 산정 방법

○ 순위를 산정하기 위한 요인은 '2021.01.01 ~ 2021.06.30' 기간동안 측정된 데이터로 한정하였다.

○ 동별로 콜택시 이용량, 장애인수, 복지시설 수 각각의 순위를 산정한후 합산하였다.

○ 합산한 순위를 오름차순으로 정리하여 최종 순위를 산출하였다.

4.4.2 순위 산정 결과

동	복지시설 수	이용량	장애인수	복지시설 _rank	장애인수 _rank	이용량 _rank	최종rank	rank
하계1동	6	17923	2126	1	9	1	11	1
가양2동	4	6195	2458	5	3	12	20	2
구산동	4	5377	2248	5	5	18	28	3
수서동	2	5046	2150	24	8	22	54	4
세곡동	3	4960	1967	12	18	24	54	4
신내1동	2	6340	1923	24	22	10	56	6
방화1동	3	4915	1819	12	24	25	61	7
역촌동	1	9902	2181	56	7	5	68	8
등촌3동	1	6308	3827	56	1	11	68	8
강일동	3	5042	1700	12	34	23	69	10

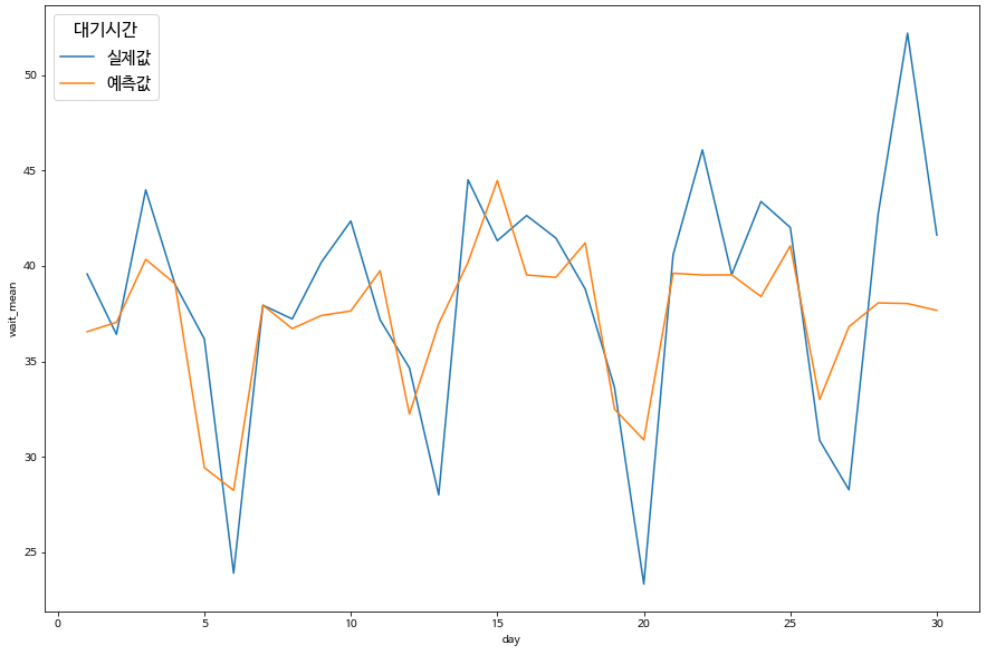
[그림 22] 차고지 우선 순위 산정

○ 앞서 지정한 순위 산정 방법을 통해 상위 10 개의 지역을 선정하였다.

○ 차고지 추가 설치 위한 최적 입지 1 위로 선정된 '노원구 하계 1 동'의 경우, 누적 콜택시 이용량 17,923 건, 거주 장애인수 2,125 명, 보유 복지시설 수 6 개를 기록하였다.

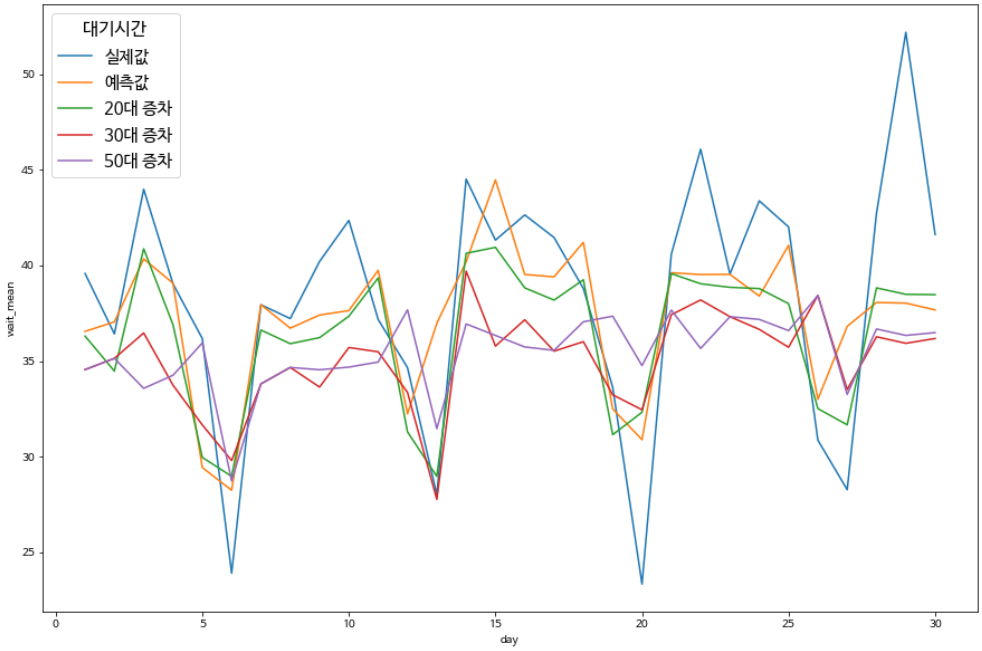
○ 동일 순위의 경우, 이용량, 장애인수, 복지시설 수 순위에 따라 순위를 선정하였다.

4.5 콜택시 차량 증차 시 대기시간 변화 모델 시뮬레이션



[그림 23] 대기시간의 실제값과 예측값 비교

○ 대기시간 예측모델로 산출된 예측값과 실제값 비교결과, 실제 대기시간과 예측값의 차이가 존재하나 그 차이가 크지 않음을 확인할 수 있었다.



[그림 24] 증차 시뮬레이션별 대기시간 비교

○ 대기시간 예측모델의 변수 중 하나인 '일별 콜택시 운행 대수'의 수치를 '20 대 증차', '30 대 증차', '50 대 증차'로 각각 조정하여 증차 시뮬레이션을 통해 대기시간 변화를 표현하였다.

○ 30 대를 증차한 경우 대기시간 감소폭이 가장 컸고, 50 대 증차한 경우는 30 대 증차한 경우와 수치 차이가 크지 않았다. 따라서 30 대를 증차하는 것이 비용효과성이 가장 높은 것으로 판단할 수 있다.

5. 활용방안

5.1 문제점 개선 방안

○ (생활 개선) 장애인 이동권 확대로 국민 편의 증진

○ (이용 효율화) 차량 및 차고지 최적 배치와 적정 수준의 증차를 통한 자원·예산 낭비 최소화

[그림 25] 차고지 최적 배치 1 순위 지역 예시

○ (정책 개선) 시행중인 장애인 콜택시 시스템 불만 민원 개선으로 이용자 만족도 향상

* 30 대의 차량 증차를 통해 실제 대기시간의 약 20% 감소 가능

○ (대책 제안) 기존 교통약자의 배려가 미흡한 대중교통의 보완

5.2 업무 활용 방안

○ (정책 활용) 교통약자인 장애인의 자유로운 이동성 보장으로 교통 복지서비스 향상

○ (타 자치구 정책 활용) 수요예측 모델 타 자치구 활용으로 전반적인 장애인콜택시 시스템 질적 상향 평준화

○ (자원 활용) 이용되지 않고 있는 공영 차고지 사용으로 인한 자원 활용

○ (일자리 창출) 차량 증차에 따른 자연스러운 일자리 참여 기회 제공

6. 참고자료

- 국토해양부,「교통약자 이동편의 실태조사」, 2008.05.
- 이병화, 양희택(2017),「서울과 경기도의 장애인 콜택시 이용현황 빅데이터 분석연구」. Asia-pacific Journal of Multimedia Services Convergent
- 이범규(2019),「장애인 특별교통수단 운행 및 서비스 개선방안 연구」, 대전세종연구원.
- 교통안전공단,「교통약자 DRT 서비스 표준모델 기획연구」, 2017.12.
- 시원선, 양희택, 이선화(2020).「장애인콜택시 전국 통합체계 마련을 위한 연구」, 한국장애인개발원.
- 김영석, 박준환, 김대명,「장애인의 지역 간 이동 편의 증진을 위한 교통 서비스 실태 및 개선방안」, 국회입법조사처, 2019.12.24.
- 박진혁, 장영재(2015),「서울시 장애인 콜택시 고객 대기시간 감소를 위한 자동배차 알고리즘 및 최적 차량 공급 제안」,『대한산업공학외 준계공등학술대회 논문집」, 대한산업공학회.
- 박상우, 김용미(2015),「일본 휠체어 이용자의 대중교통 시설 현황 및 특별교통수단 이용현황 자료수집을 위한 출장 결과보고」, 국외출장보고서.
- 홍현근(2018),「교통약자 이동편의증진 정책현황」,『월간교통』, 한국교통연구원, 6-15.
- 이슬기, “기약없는 장애인콜택시, ‘불편은 장애인 몫?”, 에이블뉴스, 2021.06.17. http://www.ablenews.co.kr/News/NewsContent.aspx?CategoryCode=0014&NewsCode=001420210617092108288037
- 임주찬, “대기만 2~3 시간 ‘장애인 콜택시’는 오늘도 예약 전쟁”, 오마이뉴스, 2019.08.22.

http://www.ohmynews.com/NWS_Web/View/at_pg.aspx?CNTN_CD=A0002563980

- 조봉현, “말많고 탈많은 장애인 콜택시, 이렇게 개선하자.”, 소셜포커스, 2019.12.26. https://www.socialfocus.co.kr/news/articleView.html?idxno=6176
- 박성용, “반복되는 장애인콜택시 내 성범죄, ‘성범죄 예방교육’ 제도개선 나서”, 월페어뉴스, 2021.08.05. https://www.welfarenews.net/news/articleView.html?idxno=75143
- 최선중, “전국 장애인 콜택시 만족도 1 위 대전…비결은?”, KBS NEWS, 2021.02.25. https://news.kbs.co.kr/news/view.do?ncd=5126189
- 예병정, “‘장애인 이동’ 저상버스 늘리는 서울시…문제는 더딘 속도”, 파이낸셜뉴스, 2021.07.12. https://www.fnnews.com/news/202107121751221579

7. 부록

7.1 분석 코드

7.1.1 크롤링 및 워드클라우드 코드

```
### 서울시설공단 홈페이지에서 민원 제목과 내용 크롤링하기
import requests
from bs4 import BeautifulSoup
import lxml.html

li = []
ti = []

for i in range(1,43):
    url =
'https://www.sisul.or.kr/open_content/calltaxi/qna/qnaMsgList.do?pgno={}'.format(i)
    response = requests.get(url)
    soup = BeautifulSoup(response.text,'html.parser')

    for a in soup.select('#detail_con > div.generalboard > table > tbody > tr'):
        title = a.select_one('td.left.title > a')
        conurl = title.get('href')
        li.append(conurl)
        ti.append(title.text)
```

```
content = []


for i in li:
    newurl = 'https://www.sisul.or.kr/'+ i
    response = requests.get(newurl)
    soup = BeautifulSoup(response.text,'html.parser')
    a = soup.select('tbody > tr')[2].select_one('td').text.strip()
    content.append(a.replace('\r',''))


data = pd.DataFrame({'제목' : ti , '내용' : content})


data.to_csv('민원데이터.csv',index = False,encoding = 'euc-kr')
```

```
# 명사 csv 파일 list 로 변환
import pandas as pd

data = pd.read_csv('명사.csv',encoding ='utf-8')
data = pd.DataFrame(data['명사'].value_counts()).reset_index()
data = data.groupby('구분')['개수'].sum().reset_index()

f = open('stopwords_korean.txt','rt',encoding ='utf-8')
lines = f.readlines()
stop_words = []

for line in lines:
    line = line.replace('\n','')
    stop_words.append(line)
f.close()

text = " ".join(text)
```

```
from PIL import Image
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.font_manager as f
```

```
from wordcloud import WordCloud


# mask 를 택시모양으로 지정
mask = np.array(Image.open('taxi.png'))


text = str(text)


# 워드클라우드 생성하기
wordcloud = WordCloud(font_path =
'C:/Users/user/Desktop/NanumFontSetup_TTF_GOTHIC/NanumGothic.ttf',backgroun
d_color = 'white',colormap ='cool',max_font_size = 100,mask = mask).generate(text)


# 워드클라우드 그리기
plt.figure(figsize = (20,10))
plt.imshow(wordcloud)
plt.savefig('wordcloud.png')
```

7.1.2 수요량 예측 모델링 코드

```
# 코랩 한글폰트 설치


!sudo apt-get install -y fonts-nanum
!sudo fc-cache -fv
```

```
!rm ~/.cache/matplotlib -rf

import matplotlib.pyplot as plt

plt.rc('font', family='NanumBarunGothic')
```

```
# 수요량 예측

import pandas as pd

import numpy as np


## 데이터 불러오기

data=pd.read_csv("/content/drive/Shareddrives/공빅

프로젝트/결과데이터/수요량용_통합데이터.csv", encoding = "EUC-KR")


## train, test 분리

### 2021 년 6 월 데이터를 test 로 선정, 2020 년 1 월 ~ 2021 년 5 월 train 으로 선정

test = data[(data["year"] == 2021) & (data["month"] == 6)]

test = test.reset_index(drop = True)

data = data[~((data["year"] == 2021) & (data["month"] == 6))]

train = data[data["year"] >= 2020]

train.drop(['year'], axis = 1,inplace =True)

test.drop(['year'], axis = 1,inplace =True)

X_train = train.drop('num',axis =1 ,inplace =False)

y_train = train['num']

X_test = test.drop('num',axis = 1,inplace =False)

y_test = test['num']
```

```
# 수요량 예측 모델링 - Catboost
!pip install catboost
from catboost import CatBoostRegressor
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error


cat = CatBoostRegressor()
cat.fit(X_train, y_train, verbose = 100)
pred = cat.predict(X_test)


### 평가지표 산출
print('MAE:',np.round(mean_absolute_error(y_test,pred),3),'RMSE:
',np.round(np.sqrt(mean_squared_error(y_test,pred)),3),'R_2 :' ,
np.round(r2_score(y_test,pred),3))


## 하이퍼파라미터 랜덤서치
param_distributions = {
    'learning_rate' : [0.01,0.1,0.05,0.03],
    'iterations' : [100,200,500,1000,1200],
    'subsample' : [0.8, 0.9, 1]
```

```
}


randomized_search = RandomizedSearchCV(cat,
param_distributions=param_distributions, n_iter=50, return_train_score=True,
scoring = 'neg_root_mean_squared_error')
randomized_search.fit(X_train, y_train)


df =
pd.DataFrame(randomized_search.cv_results_).sort_values(by=['mean_test_score',
'mean_train_score'], ascending=False)
print(df[['params', 'mean_train_score', 'mean_test_score']])


## 최적 하이퍼파라미터로 모델생성
cat1 = CatBoostRegressor(learning_rate = 0.1, iterations = 1000, subsample = 1)
cat1.fit(X_train, y_train, verbose = 100)
pred_cat = cat1.predict(X_test)


# 평가지표 산출
print('MAE :' ,np.round(mean_absolute_error(y_test,pred_cat),3),'RMSE :
',np.round(np.sqrt(mean_squared_error(y_test,pred_cat)),3),'R_2 :' ,
np.round(r2_score(y_test,pred_cat),3))
```

```
##수요량 예측 모델링 - LGBM
from lightgbm import LGBMRegressor
import random

lgb_reg = LGBMRegressor()
lgb_reg.fit(X_train,y_train)
pred = lgb_reg.predict(X_test)

# 평가지표 산출
print('MAE :' ,np.round(mean_absolute_error(y_test,pred),3),'RMSE :
',np.round(np.sqrt(mean_squared_error(y_test,pred)),3),'R_2 :' ,
np.round(r2_score(y_test,pred),3))

###하이퍼파라미터 랜덤서치
n_estimators = [int(x) for x in range(100,400,50)]
max_depth = [int(x) for x in range(2,15)]
min_samples_split = [2,5,10]
min_samples_leaf = [1,2,4]
learning_rate = [0.01,0.05,0.1]
subsample = [0.8,0.9,1]

random_grid = {'n_estimators' : n_estimators,
               'max_depth' : max_depth,
               'min_samples_leaf' : min_samples_leaf,
```

```
            'min_samples_split' : min_samples_split,
            'learning_rate' : learning_rate,
            'subsample' : subsample}
lgb = LGBMRegressor()

rnd_search = RandomizedSearchCV(lgb,param_distributions=random_grid,n_iter=
200,cv = 5,scoring = 'neg_mean_absolute_error',verbose = 100 , n_jobs
=4,random_state =42)
rnd_search.fit(X_train,y_train)

###최적 하이퍼파라미터로 모델생성
lgb1 = LGBMRegressor(max_depth=8, min_samples_leaf=2, min_samples_split=10,
n_estimators=250, subsample=0.8)
lgb1.fit(X_train,y_train)
pred_lgb = lgb1.predict(X_test)
# 평가지표 산출
print('MAE :' ,np.round(mean_absolute_error(y_test,pred_lgb),3),'RMSE :
',np.round(np.sqrt(mean_squared_error(y_test,pred_lgb)),3),'R_2 :' ,
np.round(r2_score(y_test,pred_lgb),3))
```

```
##수요량 예측 모델링 - xgboost
import xgboost
xgb = xgboost.XGBRegressor()
xgb.fit(X_train, y_train)
pred = xgb.predict(X_test)

# 평가지표 산출
print('MAE :' ,np.round(mean_absolute_error(y_test,pred),3),'RMSE :
',np.round(np.sqrt(mean_squared_error(y_test,pred)),3),'R_2 :' ,
np.round(r2_score(y_test,pred),3))

###하이퍼파라미터 랜덤서치
n_estimators = [80, 100, 200]
max_depth = [6,8,10]
learning_rate = [0.3, 0.1, 0.01]

random_grid = {'n_estimators' : n_estimators,
            'max_depth' : max_depth,
            'learning_rate' : learning_rate,
}

xgb = XGBRegressor()
rnd_search = RandomizedSearchCV(xgb,param_distributions=random_grid,n_iter= 15,cv =
5,scoring = 'neg_mean_absolute_error',verbose = 5 , n_jobs =4,random_state =42)
```

```
rnd_search.fit(X_train,y_train)


###최적 하이퍼파라미터로 모델생성
xgb1 = xgboost.XGBRegressor(n_estimators=80, learning_rate=0.3, max_depth=8)
xgb1.fit(X_train, y_train)
pred_xgb = xgb1.predict(X_test)
# 평가지표 산출
print('MAE :' ,np.round(mean_absolute_error(y_test,pred_xgb),3),'RMSE :
',np.round(np.sqrt(mean_squared_error(y_test,pred_xgb)),3),'R_2 :' ,
np.round(r2_score(y_test,pred_xgb),3))
```

7.1.3 앙상블 코드

```
##Catboost, LGBM, xgboost 모델 앙상블


#음수값은 0 으로 대체
pred_cat[pred_cat<0] = 0
pred_lgb[pred_lgb<0] = 0
pred_xgb[pred_xgb<0] = 0


#3 가지 예측값 평균
pred_mean = np.mean([pred_cat, pred_lgb, pred_xgb], axis = 0)
pred_mean2 = np.round(pred_mean,0)
```

```
print('MAE :' ,np.round(mean_absolute_error(y_test,pred_mean2),3),'RMSE :
',np.round(np.sqrt(mean_squared_error(y_test,pred_mean2)),3),'R_2 :' ,
np.round(r2_score(y_test,pred_mean2),3))


##모델별 예측값
pred_mean2 = pd.DataFrame(pred_mean2, columns = ["pred"])
pred_mean2


pred_cat = pd.DataFrame(pred_cat, columns = ["pred_cat"])
pred_lgb = pd.DataFrame(pred_lgb, columns = ["pred_lgb"])
pred_xgb = pd.DataFrame(pred_xgb, columns = ["pred_xgb"])


demand_final = pd.concat([X_test, y_test, pred_mean2], axis =1)
demand_final = pd.concat([demand_final, pred_cat, pred_lgb, pred_xgb], axis=1)
demand_final
```

7.1.4 수요량 시각화 코드

```
## 날짜별로 데이터 합침
demand_final_0 = demand_final[demand_final['loc_cat'] == 0]
group1 = demand_final_0['num'].groupby(demand_final_0['day'])
group2 = demand_final_0['pred'].groupby(demand_final_0['day'])
group3 = demand_final_0['pred_cat'].groupby(demand_final_0['day'])
group4 = demand_final_0['pred_lgb'].groupby(demand_final_0['day'])
group5 = demand_final_0['pred_xgb'].groupby(demand_final_0['day'])
```

```
g_real = group1.sum()

g_pred = group2.sum()

g_cat = group3.sum()

g_lgb = group4.sum()

g_xgb = group5.sum()


plt.figure(figsize=(13, 8))


plt.plot(g_real.index, g_real, linestyle='-', linewidth=3, label='real') # solid

plt.plot(g_pred.index, g_pred, linestyle='-', linewidth=3, label='ensemble') # dotted

plt.plot(g_cat.index, g_cat, linestyle='--', linewidth=1, label='catboost') # 'dashed'

plt.plot(g_lgb.index, g_lgb, linestyle='-.', linewidth=1, label='lgbm') # dashdotted

plt.plot(g_xgb.index, g_xgb, linestyle=':', linewidth=1, label='xgboost') # dashdotted


plt.title('Real VS Predict', fontsize=20)

plt.ylabel('Cummulative Num', fontsize=14)

plt.xlabel('Date', fontsize=14)

plt.legend()

plt.show()
```

7.1.5 대기시간 예측 모델링 코드

```
##데이터 불러오기
data2 = pd.read_csv("/content/drive/Shareddrives/공빅
프로젝트/산출물/결과데이터/대기시간용_통합데이터.csv", encoding = "EUC-KR")


##train, test 분리
test = data2[(data2["year"] == 2021) & (data2["month"] == 6)]
test = test.reset_index(drop = True)
data2 = data2[~((data2["year"] == 2021) & (data2["month"] == 6))]
train = data2[data2["year"] >= 2020]


X_train = train.drop(['wait', 'year'],axis =1 ,inplace =False)
y_train = train['wait']
X_test = test.drop(['wait', 'year'],axis = 1,inplace =False)
y_test = test['wait']
```

```
##대기시간 예측 모델링 - Catboost
cat = CatBoostRegressor()


cat.fit(X_train, y_train, verbose = 100)


pred = cat.predict(X_test)
```

```
# 평가지표 산출
print('MAE :' ,mean_absolute_error(y_test,pred),'RMSE :
',np.sqrt(mean_squared_error(y_test,pred)),'R_2 :' , r2_score(y_test,pred))


###하이퍼파라미터 랜덤서치
param_distributions = {
    'learning_rate' : [0.01,0.1,0.05,1,0.005],
    'iterations' : [100,200,500,1000,1200],
    'subsample' : [0.8, 0.9, 1]
}

randomized_search = RandomizedSearchCV(cat,
param_distributions=param_distributions, n_iter=50, return_train_score=True,
scoring = 'neg_mean_absolute_error')
randomized_search.fit(X_train, y_train)

###최적 하이퍼파라미터로 모델생성
cat = CatBoostRegressor(iterations=1000, learning_rate= 0.1, subsample = 1)


cat.fit(X_train, y_train, verbose = 100)


pred_cat2 = cat.predict(X_test)
```

```
# 평가지표 산출
print('MAE :' ,mean_absolute_error(y_test,pred_cat2),'RMSE :
',np.sqrt(mean_squared_error(y_test,pred_cat2)),'R_2 :' , r2_score(y_test,pred_cat2))
```

```
##대기시간 예측 모델링 - LGBM
lgb_reg = LGBMRegressor()

lgb_reg.fit(X_train,y_train)

pred = lgb_reg.predict(X_test)
# 평가지표 산출
print('MAE :' ,np.round(mean_absolute_error(y_test,pred),3),'RMSE :
',np.round(np.sqrt(mean_squared_error(y_test,pred)),3),'R_2 :' ,
np.round(r2_score(y_test,pred),3))

###하이퍼파라미터 랜덤서치
n_estimators = [int(x) for x in range(100,400,50)]
max_depth = [int(x) for x in range(2,15)]
min_samples_split = [2,5,10]
min_samples_leaf = [1,2,4]
learning_rate = [0.01,0.05,0.1]
subsample = [0.8,0.9,1]
```

```
random_grid = {'n_estimators' : n_estimators,
               'max_depth' : max_depth,
               'min_samples_leaf' : min_samples_leaf,
               'min_samples_split' : min_samples_split,
               'learning_rate' : learning_rate,
               'subsample' : subsample}
lgb = LGBMRegressor()


rnd_search = RandomizedSearchCV(lgb,param_distributions=random_grid,n_iter=
200,cv = 5,scoring = 'neg_mean_absolute_error',verbose = 100 , n_jobs
=4,random_state =42)
rnd_search.fit(X_train,y_train)


###최적 하이퍼파라미터로 모델생성
lgb_reg = LGBMRegressor(subsample =  0.8, n_estimators =  200, min_samples_split =
5, min_samples_leaf = 2, max_depth = 13, learning_rate =  0.05)


lgb_reg.fit(X_train,y_train)


pred_lgb2 = lgb_reg.predict(X_test)
# 평가지표 산출
print('MAE :' ,np.round(mean_absolute_error(y_test,pred_lgb2),3),'RMSE :
',np.round(np.sqrt(mean_squared_error(y_test,pred_lgb2)),3),'R_2 :' ,
np.round(r2_score(y_test,pred_lgb2),3))
```

```
##대기시간 예측 모델링 - xgboost
xgb = xgboost.XGBRegressor()
xgb.fit(X_train, y_train)
pred = xgb.predict(X_test)

# 평가지표 산출
print('MAE :' ,np.round(mean_absolute_error(y_test,pred),3),'RMSE :
',np.round(np.sqrt(mean_squared_error(y_test,pred)),3),'R_2 :' ,
np.round(r2_score(y_test,pred),3))

###하이퍼파라미터 랜덤서치
n_estimators = [80, 100, 200]
max_depth = [5, 7, 9]
learning_rate = [0.3, 0.1, 0.05]
subsample = [0.8,0.9,1]
n_jobs = [8,10,12]

random_grid = {'n_estimators' : n_estimators,
               'max_depth' : max_depth,
               'learning_rate' : learning_rate,
               'subsample' : subsample,
               'n_jobs' : n_jobs}
xgb = XGBRegressor()
```

```
rnd_search = RandomizedSearchCV(xgb,param_distributions=random_grid,n_iter=
200,cv = 5,scoring = 'neg_mean_absolute_error',verbose = 100 , n_jobs
=4,random_state =42)
rnd_search.fit(X_train,y_train)


###최적 하이퍼파라미터로 모델생성
xgb = xgboost.XGBRegressor(n_estimators=80, max_depth=9, learning_rate=0.1,
subsample=1, n_jobs=8)
xgb.fit(X_train, y_train)
pred_xgb2 = xgb.predict(X_test)


# 평가지표 산출
print('MAE :' ,np.round(mean_absolute_error(y_test,pred_xgb2),3),'RMSE :
',np.round(np.sqrt(mean_squared_error(y_test,pred_xgb2)),3),'R_2 :' ,
np.round(r2_score(y_test,pred_xgb2),3))
```

```
##Catboost, LGBM, xgboost 모델 앙상블
#음수값은 0 으로 대체
pred_cat2[pred_cat2<0] = 0
pred_lgb2[pred_lgb2<0] = 0
pred_xgb2[pred_xgb2<0] = 0


#3 가지 예측값 평균
pred_mean3 = np.mean([pred_cat2, pred_lgb2, pred_xgb2], axis = 0)
```

```
pred_mean4 = np.round(pred_mean3,1)


# 평가지표 산출
print('MAE :' ,np.round(mean_absolute_error(y_test,pred_mean3),3),'RMSE :
',np.round(np.sqrt(mean_squared_error(y_test,pred_mean3)),3),'R_2 :' ,
np.round(r2_score(y_test,pred_mean3),3))
# 평가지표 산출
print('MAE :' ,np.round(mean_absolute_error(y_test,pred_mean4),3),'RMSE :
',np.round(np.sqrt(mean_squared_error(y_test,pred_mean4)),3),'R_2 :' ,
np.round(r2_score(y_test,pred_mean4),3))
```

7.1.6 증차시 대기시간 시뮬레이션 코드

```
# 증차시 대기시간 시뮬레이션
y_test1 = pd.DataFrame(y_test)


pred1 = cat.predict(X_test)
pred1 = pd.DataFrame(pred1, columns = ["pred1"]) # 기존 예측값


#20 대 증차
X_test2 = X_test.copy()
X_test2["차량운행"] = X_test2["차량운행"] + 20


cat_sim = cat.predict(X_test2)
cat_sim  = pd.DataFrame(cat_sim , columns = ["pred2"])
```

```
#30 대 증차

X_test3 = X_test.copy()

X_test3["차량운행"] = X_test2["차량운행"] + 30


cat_sim2 = cat.predict(X_test3)

cat_sim2  = pd.DataFrame(cat_sim2 , columns = ["pred3"])


#50 대 증차

X_test4 = X_test.copy()

X_test4["차량운행"] = X_test2["차량운행"] + 50


cat_sim3 = cat.predict(X_test4)

cat_sim3 = pd.DataFrame(cat_sim3, columns = ["pred4"])
```

```
#데이터 합치기

day = pd.concat([X_test, y_test1, pred1, cat_sim, cat_sim2, cat_sim3], axis = 1)

day


###일별 평균 대기시간 계산

day["num_wait"] = day["num"] * day["wait"]

day["pred_wait"] = day["num"] * day["pred1"]

day["pred_wait2"] = day["num"] * day["pred2"]

day["pred_wait3"] = day["num"] * day["pred3"]

day["pred_wait4"] = day["num"] * day["pred4"]
```

```
day1 = day.groupby(["month", "day"])["num","num_wait",
"pred_wait","pred_wait2","pred_wait3","pred_wait4"].sum()
day1


day1["wait_mean"] = day1["num_wait"] / day1["num"]
day1["pred_wait_mean"] = day1["pred_wait"] / day1["num"]
day1["pred_wait2_mean"] = day1["pred_wait2"] / day1["num"]
day1["pred_wait3_mean"] = day1["pred_wait3"] / day1["num"]
day1["pred_wait4_mean"] = day1["pred_wait4"] / day1["num"]


day1 = day1.reset_index()
```

7.1.7 증차시 대기시간 시뮬레이션 시각화 코드

```
plt.figure(figsize = (15,10))
sns.lineplot(x = day1["day"], y=day1["wait_mean"])
sns.lineplot(x = day1["day"], y=day1["pred_wait_mean"])
plt.legend(labels=["실제값","예측값"], title = "대기시간")


plt.legend(labels=["실제값","예측값"], fontsize = 15, title = "대기시간",title_fontsize=16, loc
= "upper left")


plt.savefig('/content/drive/Shareddrives/공빅 프로젝트/산출물/결과데이터/sim1.png')


plt.figure(figsize = (15,10))
```

```
sns.lineplot(x = day1["day"], y=day1["wait_mean"])
sns.lineplot(x = day1["day"], y=day1["pred_wait_mean"])
sns.lineplot(x = day1["day"], y=day1["pred_wait2_mean"])

plt.legend(labels=["실제값","예측값", "20 대 증차"], fontsize = 15, title =
"대기시간",title_fontsize=16, loc = "upper left")
plt.savefig('/content/drive/Shareddrives/공빅 프로젝트/산출물/결과데이터/sim2.png')

plt.figure(figsize = (15,10))
sns.lineplot(x = day1["day"], y=day1["wait_mean"])
sns.lineplot(x = day1["day"], y=day1["pred_wait_mean"])
sns.lineplot(x = day1["day"], y=day1["pred_wait2_mean"])
sns.lineplot(x = day1["day"], y=day1["pred_wait3_mean"])

plt.legend(labels=["실제값","예측값", "20 대 증차", "30 대 증차"], fontsize = 15, title =
"대기시간",title_fontsize=16, loc = "upper left")
plt.savefig('/content/drive/Shareddrives/공빅 프로젝트/산출물/결과데이터/sim3.png')

plt.figure(figsize = (15,10))
sns.lineplot(x = day1["day"], y=day1["wait_mean"])
sns.lineplot(x = day1["day"], y=day1["pred_wait_mean"])
sns.lineplot(x = day1["day"], y=day1["pred_wait2_mean"])
sns.lineplot(x = day1["day"], y=day1["pred_wait3_mean"])
sns.lineplot(x = day1["day"], y=day1["pred_wait4_mean"])
```

```
plt.legend(labels=["실제값","예측값", "20 대 증차", "30 대 증차", "50 대증차"], fontsize = 15,
title = "대기시간",title_fontsize=16, loc = "upper left")
plt.savefig('/content/drive/Shareddrives/공빅 프로젝트/산출물/결과데이터/sim4.png')
```