

# 장애인 콜택시 현황 분석

2024-1 데이터수집을 위한 프로그래밍



3조

i 중국언어문화학과 19 이정인  
경영학과 21 장수연  
회계학과 21 김나현  
인공지능학과 22 박요한



# 목차

- |    |              |    |       |
|----|--------------|----|-------|
| 1. | 문제 인식        | 7. | 솔루션   |
| 2. | 주제 선정        | 8. | 역할 분담 |
| 3. | 데이터 수집       |    |       |
| 4. | 데이터 전처리      |    |       |
| 5. | 데이터 탐색       |    |       |
| 6. | 데이터 분석 및 시각화 |    |       |



# \* 문제 인식

## 시각 장애인 유튜버 ‘원샷한솔’

- 구독자 86.3만명
- 시각 장애인의 일상 공유

## 장애인으로서의 고충 공유

- 교통, 사회적 인식, 일상 속 소소한 불편함 등 다양한 주제로 장애인의 삶을 보여주는 콘텐츠 제작

## 장애인 이동권(교통) 문제 인식

- 버스, 택시, 지하철 등 교통 부분에서 장애인들이 큰 불편함을 겪고 있음을 인식



# 주제 선정

## 1) 빅카인즈에서 Excel 다운로드

- keyword = ‘교통약자’, ‘장애인 교통수단’

## 2) 크롤링 후 txt 파일로 저장

- 수업시간 코드에서 일련번호 제외: str(cnt) 생략

## 3) 형태소 분석 후 워드클라우드 생성

- 신문사별 txt 파일을 keyword별로 합쳐서 진행

```
def text_to_wordcloud(file_name):  
    import konlpy  
    from konlpy.corpus import kolaw  
    from konlpy.tag import Komoran, Okt, Kkma, Hannanum  
    from matplotlib import font_manager, rc  
    from nltk import Text  
    from wordcloud import WordCloud  
  
    kolaw.fileids()  
  
    import os  
    current_dir = os.getcwd()  
    file_path = os.path.join(current_dir, file_name)  
    with open(file_path, 'r', encoding='utf-8') as file:  
        c = file.read()  
  
    # 형태소 분석기  
    hannanum = Hannanum()  
    okt = Okt()  
  
    ## 원도우즈 - 한글 폰트 설정 - 맑은  
    font_name = font_manager.FontProperties(fname="c:/  
Windows/Fonts/malgun.ttf").get_name()  
    rc('font', family=font_name)  
  
    # 단어 빈도  
    kolaw = Text(okt.nouns(c), name="kolaw")  
    path = 'c:/Windows/Fonts/malgun.ttf'  
  
    # 워드 클라우드  
    wc = WordCloud(width = 1000, height = 600,  
        background_color="white", font_path = path)  
    plt.imshow(wc.generate_from_frequencies(kolaw.vocab()))  
    plt.axis("off")  
    plt.show()
```

# 크롤링, txt파일 생성

```
chrome_options = webdriver.ChromeOptions()  
driver = webdriver.Chrome()  
driver.implicitly_wait(5)  
start_time = time.time()  
  
keyword = '교통약자'  
path = base_path + 'NewsResult_교통약자.xlsx'  
for press in ['조선일보', '중앙일보', '동아일보', '경향신문', '한겨레']:  
    run(path, keyword, press)  
driver.quit()
```

```
chrome_options = webdriver.ChromeOptions()  
driver = webdriver.Chrome()  
driver.implicitly_wait(5)  
start_time = time.time()  
  
keyword = '장애인 교통수단'  
path = base_path + 'NewsResult_장애인 교통수단.xlsx'  
for press in ['조선일보', '중앙일보', '동아일보', '경향신문', '한겨레']:  
    run(path, keyword, press)  
driver.quit()
```

## ‘고통약자’ 검색

# #교통 #택시 #장애인



# 키워드별 파일 합치기

selected\_files =

'data/조선일보\_교통약자.txt',  
'data/중앙일보\_교통약자.txt',  
'data/동아일보\_교통약자.txt',  
'data/경향신문\_교통약자.txt',  
'data/한겨레\_교통약자.txt'

```
output_file = 'combined_교통약자.txt'
```

combined\_content = ""

```
for file_path in selected_files:
```

```
with open(file_path, 'r', encoding='utf-8') as file:  
    combined_content += file.read() + "\n"
```

```
with open(output_file, 'w', encoding='utf-8') as output:  
    output.write(combined_content)
```

# 워드클라우드

```
text_to_wordcloud('combined._교통약자.txt')
```

# ‘장애인 교통수단’ 검색

# #택시 #운행 #콜



## # 키워드별 파일 합치기

selected\_files =

'data/조선일보\_장애인 교통수단.txt',  
'data/중앙일보\_장애인 교통수단.txt',  
'data/동아일보\_장애인 교통수단.txt',  
'data/경향신문\_장애인 교통수단.txt',  
'data/한겨레\_장애인 교통수단.txt'

```
output_file = 'combined_장애인 교통수단.txt'
```

combined\_content = ""

```
for file_path in selected_files:
```

```
with open(file_path, 'r', encoding='utf-8') as file:  
    combined_content += file.read() + "\n"
```

```
with open(output_file, 'w', encoding='utf-8') as output:  
    output.write(combined_content)
```

# 워드클라우드

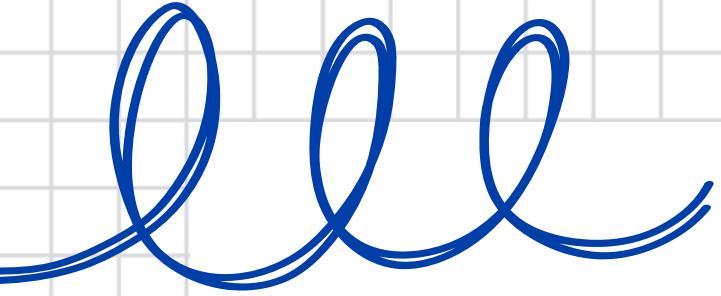
```
text_to_wordcloud('combined_장애인 교통수단.txt')
```

# “장애인 콜택시”

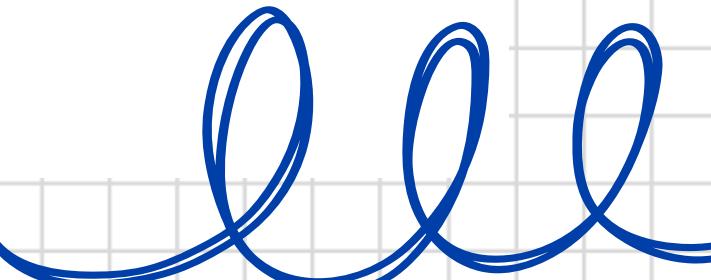
총 691대

전화/문자/인터넷/앱을 통해 차량 신청 → 자동배차

보행상 장애가 있는 장애정도가 심한(1~3급) 장애인 대상



# 데이터 수집



## 1. 장애인콜시스템

main data

Open API

장애인콜택시 이용정보

## 2. 서울시 장애인 현황(등급별/동별)

sub data

Excel file

서울시 등록장애인의 장애등급별(1급-6급) 통계

# 데이터 전처리

```
def FindDisabledCallTaxi(findDate):  
    service_key = '...'  
  
    url = f'http://openapi.seoul.go.kr:8088/  
          {service_key}/xml/disabledCalltaxi/1/100/' + findDate  
  
    req = requests.get(url)  
    soup = BeautifulSoup(req.text, 'html.parser')  
  
    NOs = soup.find_all('no')  
    CARTYPEs = soup.find_all('cartype')  
    RECEIPTIMEs = soup.find_all('receipttime')  
    SETTIMEs = soup.find_all('settime')  
    RIDETIMEs = soup.find_all('ridetime') ...
```

## 데이터 수집 함수

- OpenAPI 서비스 키 저장
- OpenAPI 요청 URL 저장
- requests.get(url)으로 URL로 HTTP GET 요청 전송
- BeautifulSoup으로 HTML 파싱
- soup.find\_all로 데이터 추출
- 각 태그 이름에 해당하는 요소들 리스트로 저장

# 데이터 전처리

```
for 컬럼명 in zip(컬럼명):  
    if CARTYPE.get_text() != "":  
  
        cartype = CARTYPE.get_text()  
        if cartype == '중형승합':  
            cartype = '중형 승합'  
        elif cartype == '중형승용':  
            cartype = '중형 승용'  
        elif cartype == '중영 승용':  
            cartype = '중형 승용'  
        elif cartype == '대형승용':  
            cartype = '대형 승용'  
  
        receiptime =  
        ConvertDate(RECEIPTIME.get_  
        text())  
        settime =  
        ConvertDate(SETTIME.get_text  
        ())  
        ridetime =  
        ConvertDate(RIDETIME.get_te  
        xt())  
  
time1 = settime - receiptime  
time2 = ridetime - settime  
time3 = ridetime -  
receiptime
```

## 데이터 정리 및 컬럼 생성

- 'CARTYPE' 값이 없는 경우 제외
- CARTYPE의 다양한 형태의 값을 통일
- ConvertDate 함수 사용하여 날짜 문자열 → datetime 객체
- 배차 대기 시간(time1) / 대기시간(time3) 계산 및 컬럼 추가
- 배차 대기 시간 = 배차 시간 - 접수 시간
- 대기 시간 = 승차 시간 - 접수 시간

# 데이터 전처리

```
def ConvertDate(in_date):
    list_date = re.split('[-:]',in_date)
    intyear = int(list_date[0])
    intmonth = int(list_date[1])
    intday = int(list_date[2])
    inthour = 0
    intmin = 0
    intsec = 0
    if len(list_date) == 7:
        inthour = int(list_date[4])
        if list_date[3] == '오전':
            if inthour == 12:
                inthour = 0
            else:
                if inthour != 12:
                    inthour = inthour + 12
        intmin = int(list_date[5])
        intsec = int(list_date[6])
```

## 문자열 → 날짜 데이터 변환 함수

- re.split을 통해 문자열을 하이픈(-), 콜론(:), 공백( )을 기준으로 나눈 후 list\_date에 저장
- 나누어진 list의 요소들을 정수로 변환하여 year, month, day, hour, min, sec을 각각 표현하는 변수 생성
- 오전과 오후를 구분하기 위해 24시간 형식으로 변경 후 hour에 할당

# 데이터 전처리

```
if ridetime>=settime and settime >= receiptime:  
    NO_list.append(NO.get_text())  
    CARTYPE_list.append(cartype)  
    RECEIPTIME_list.append(receiptime)  
    SETTIME_list.append(settime)  
    RIDETIME_list.append(ridetime)  
    STARTPOS1_list.append(STARTPOS1.get_text())  
    STARTPSO2_list.append(STARTPSO2.get_text())  
    ENDPOS1_list.append(ENDPOS1.get_text())  
    ENDPOS2_list.append(ENDPOS2.get_text())  
  
    TIME1_list.append(time1)  
    TIME2_list.append(time2)  
    TIME3_list.append(time3)
```

## 데이터 유효성 검사

- if문을 통해 시간 순서가 올바른지 (접수 시간이 배차 시간 보다 빠르고, 배차 시간이 승차 시간보다 빠른지) 확인
- 유효한 데이터만 list에 append
- 이후 각 list를 이용해 DataFrame 생성

# 데이터 전처리

```
sDate = datetime(2024, 5, 1, 0, 0, 0)
for i in range(0,10):
    dt = sDate + timedelta(days=i)
    strDate =
    dt.strftime('%04Y%02m%02d')
    df = findDisabledCallTaxi(strDate)

    if i == 0:
        DisabledCallTaxi = df
    else:
        DisabledCallTaxi =
pd.concat([DisabledCallTaxi, df])

DisabledCallTaxi['TIME1'] =
round((DisabledCallTaxi['TIME1'].dt.seconds / 60),1)
DisabledCallTaxi['TIME2'] =
round((DisabledCallTaxi['TIME2'].dt.seconds / 60),1)
DisabledCallTaxi['TIME3'] =
round((DisabledCallTaxi['TIME3'].dt.seconds / 60),1)
```

## 데이터 프레임 생성

- sDate에 데이터 수집 시작할 날짜 설정
- for문의 range로 며칠 간의 데이터 수집할지 설정
- findDisabledCallTaxi 함수로 데이터 수집, DataFrame 생성
- if문을 통해 첫번째 반복 이후로는 기존 DataFrame에 concat하여 병합
- round 함수 사용하여 분 단위로 전환

# 데이터 탐색

	No	Cartype	ReceiptTime	Settime	RideTime	StartPos1	StartPos2	EndPos1	EndPos2	Time1	Time2	Time3	
0	8366	중형 승합	2024-05-01 00:00:00	2024-05-01 00:05:47	2024-05-01 00:38:37	종로구	종로1.2.3.4가동	광진구	광장동	5.8	32.8	38.6	
1	8366	중형 승합	2024-05-01 00:00:00	2024-05-01 00:05:47	2024-05-01 00:38:37	종로구	종로1.2.3.4가동	광진구	광장동	5.8	32.8	38.6	
2	9373	중형 승합	2024-05-01 00:06:00	2024-05-01 00:10:48	2024-05-01 00:28:09	마포구	망원제1동	서대문구	남가좌제2동	4.8	17.4	22.2	
3	7843	중형 승합	2024-05-01 00:06:38	2024-05-01 00:49:05	2024-05-01 01:18:12	금천구	시흥제5동	금천구	시흥제2동	42.4	29.1	71.6	
4	5799	중형 승합	2024-05-01 00:30:11	2024-05-01 00:37:32	2024-05-01 01:08:30	서초구	반포4동	강남구	수서동	7.4	31.0	38.3	
...	...	...	...	...	...	...	...	...	...	...	...	...	
4758	8262	중형 승합	2024-05-10 23:39:00	2024-05-10 23:58:10	2024-05-11 00:34:03	관악구		조원동	구로구	개봉제1동	19.2	35.9	55.0
4759	4967	중형 승합	2024-05-10 23:40:00	2024-05-11 00:02:31	2024-05-11 00:27:56	관악구		대학동	관악구	신사동	22.5	25.4	47.9
4760	8094	중형 승합	2024-05-10 23:54:00	2024-05-11 00:16:06	2024-05-11 00:44:07	서초구	방배2동	성북구	장위제3동	22.1	28.0	50.1	
4761	9373	중형 승합	2024-05-10 23:59:00	2024-05-11 00:56:35	2024-05-11 01:29:30	용산구		한강로동	동대문구	신설동	57.6	32.9	90.5
4762	9145	중형 승합	2024-05-10 23:59:00	2024-05-11 00:43:49	2024-05-11 00:44:02	김포시	김포2동	구로구	오류제2동	44.8	0.2	45.0	

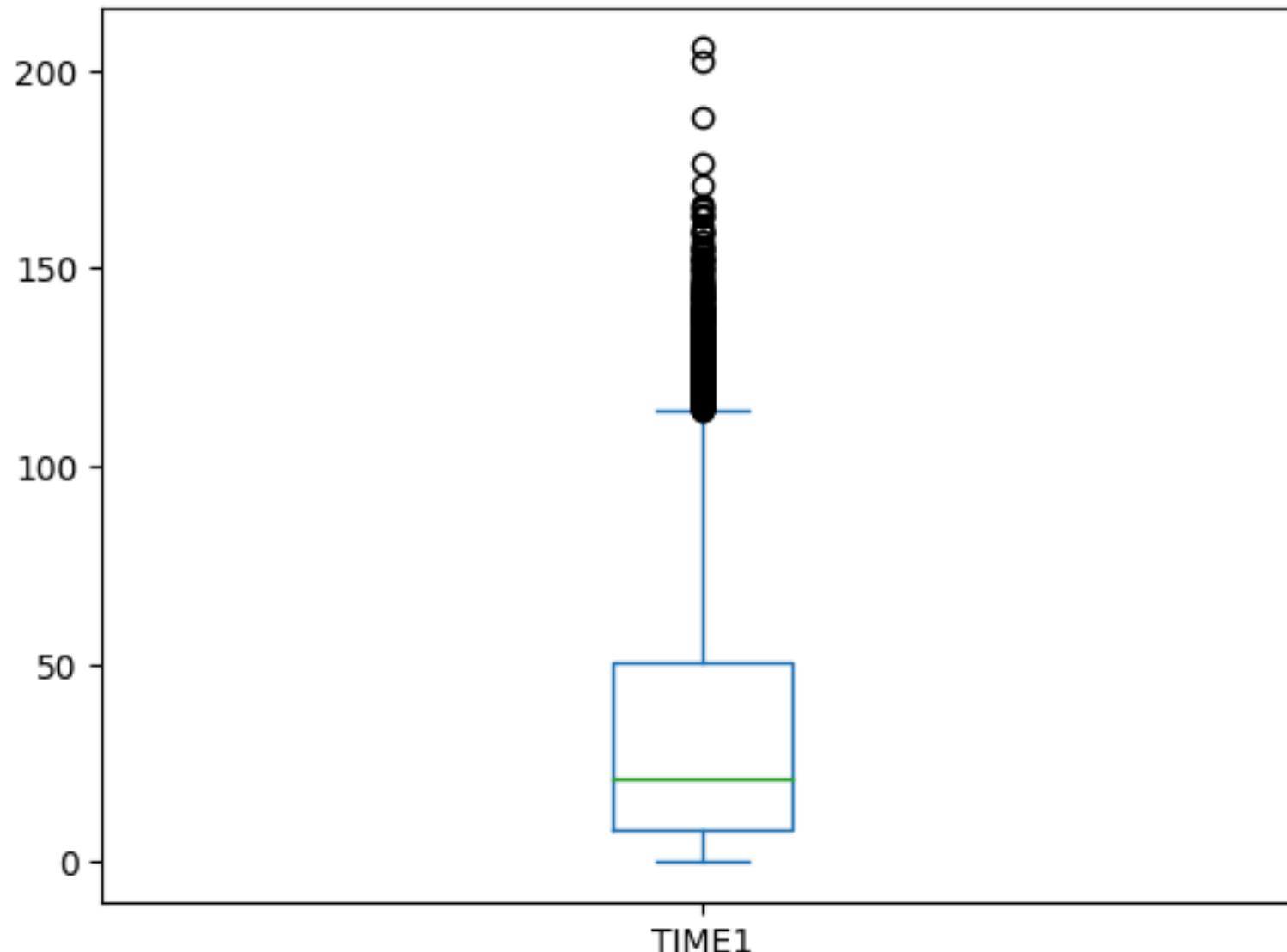
# 데이터 탐색

```
<class 'pandas.core.frame.DataFrame'>
Index: 35881 entries, 0 to 4762
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   NO          35881 non-null   object  
 1   CARTYPE     35881 non-null   object  
 2   RECEIPTIME  35881 non-null   datetime64[ns]
 3   SETTIME     35881 non-null   datetime64[ns]
 4   RIDETIME    35881 non-null   datetime64[ns]
 5   STARTPOS1   35881 non-null   object  
 6   STARTPS02   35881 non-null   object  
 7   ENDPOS1    35881 non-null   object  
 8   ENDPOS2    35881 non-null   object  
 9   TIME1       35881 non-null   float64 
 10  TIME2      35881 non-null   float64 
 11  TIME3      35881 non-null   float64 
dtypes: datetime64[ns](3), float64(3), object(6)
memory usage: 3.6+ MB
```

## 전처리 완료 후 데이터 프레임

- .info()
- 컬럼 : 12개
- 데이터 수 : 35881개
- 시간 정보 컬럼 형식 : datetime
- 대기 시간 계산된 컬럼 형식 : float
- 이외 컬럼 형식 : object

# 데이터 탐색



## TIME1 컬럼

- min/max 차이, 표준편차가 매우 큼
- Box plot을 통해 outlier가 굉장히 많음을 파악
- 평균값 사용이 적합하지 않음

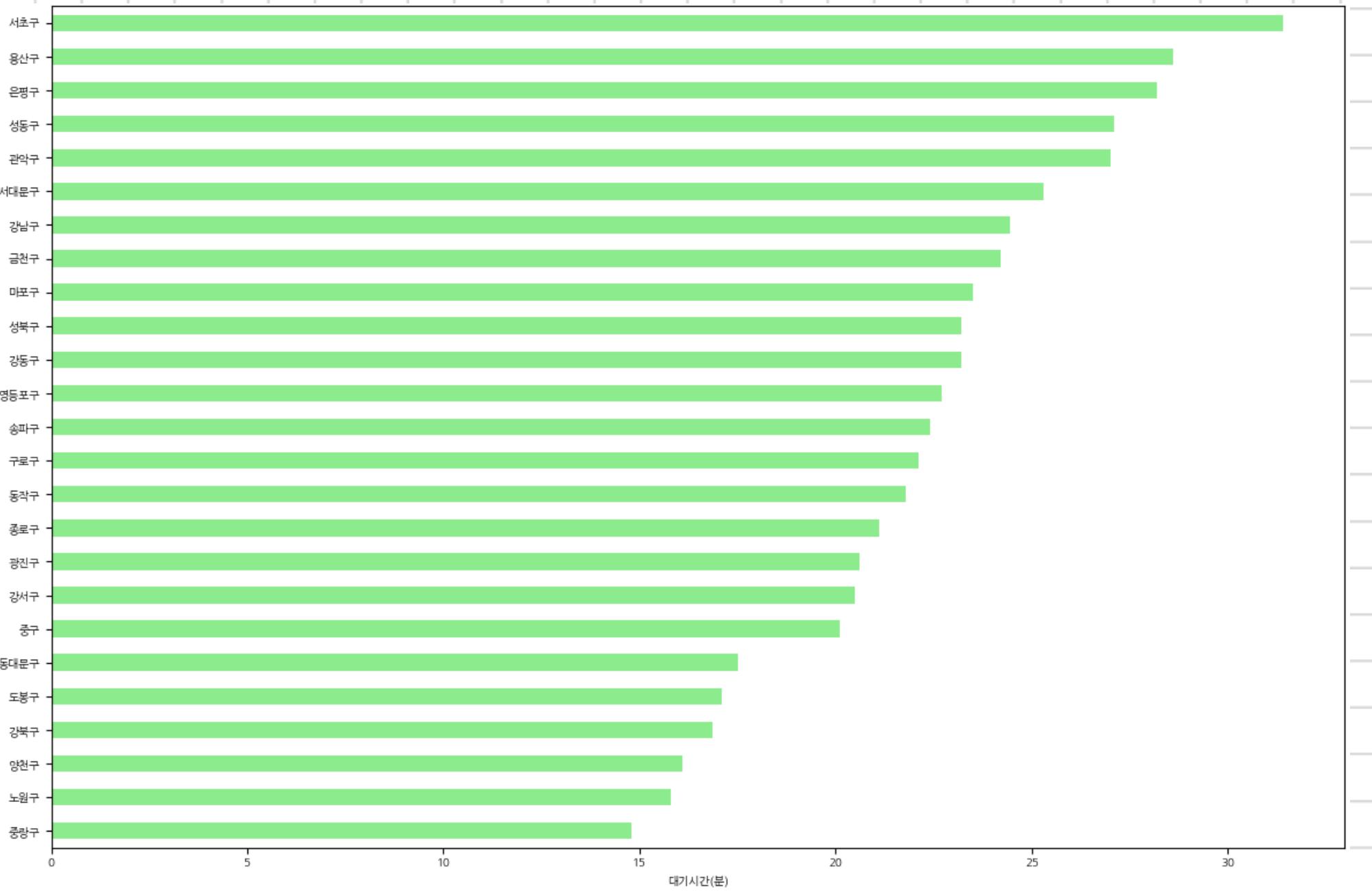
-> 차후 분석에서 주로 중앙값 사용

## TIME2, TIME3 컬럼

- 교통 상황, 배차 시 택시 위치 등의 영향을 크게 받음

-> 차후 분석에서 주로 TIME1 컬럼 사용

# 행정구역별 배차대기시간



#TIME1 중앙값 (서울만 포함)

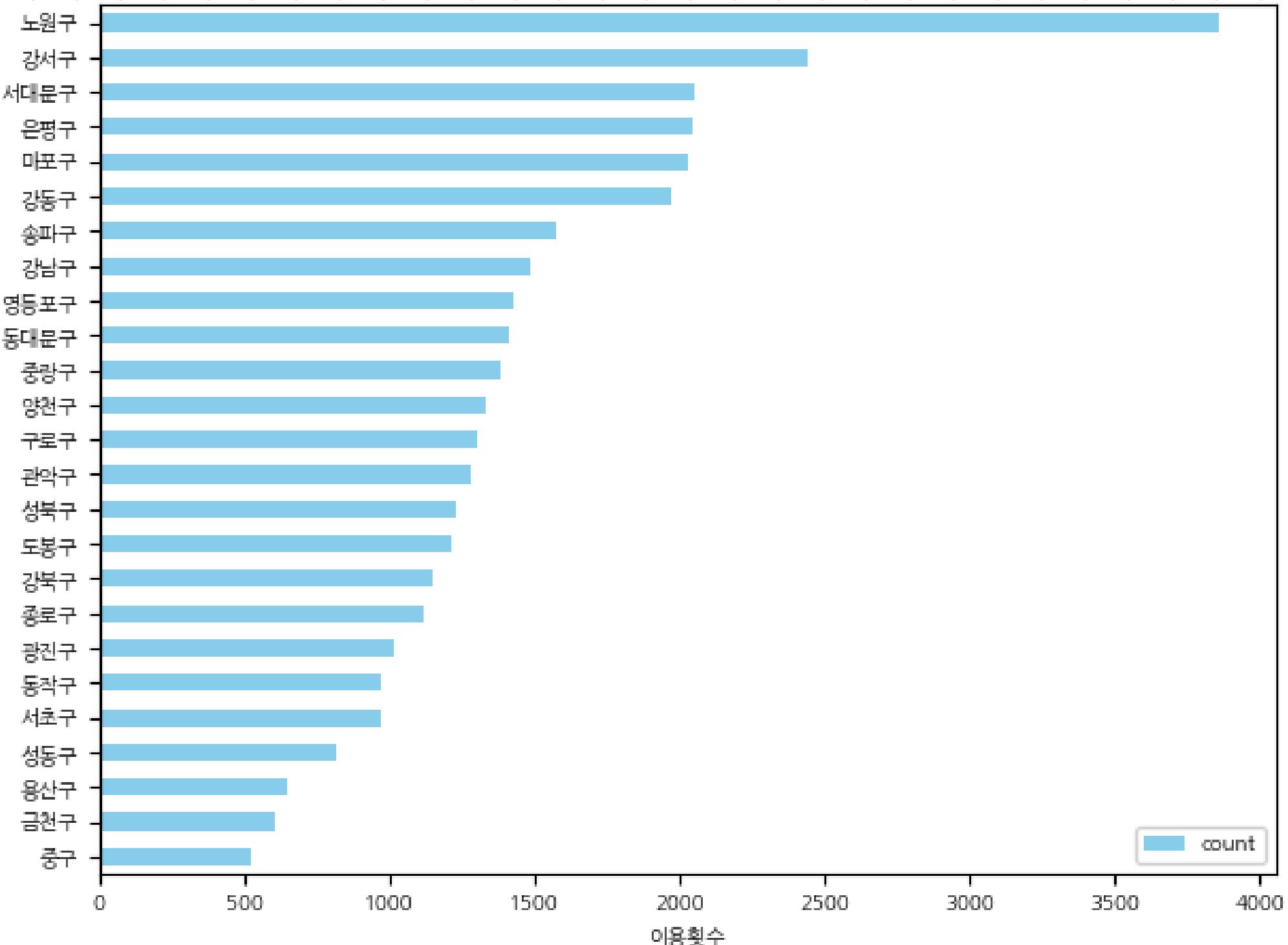
```
pos_median = DisabledCallTaxi.groupby('STARTPOS1')[  
    ['TIME1']].median().sort_values(ascending=False)  
pos_median_slec = pos_median.drop(['광명시', '하남시',  
    '김포시', '성남시분당구', '구리시', '양주시', '성남시수정구'])
```

## 분석

배차대기시간의 중앙값 Range :  
14.80(min) ~ 31.40(max)

max 값은 min 값의 약 2배 수준

# 행정구역별 이용횟수



#각 구 당 이용횟수를 나타낸 dataframe

topSTARTPOS1 =

DisabledCallTaxi['STARTPOS1'].value\_counts().to\_frame()

#서울 외엔 표본이 적어 서울시만 포함

topSTARTPOS1\_slec = topSTARTPOS1.iloc[:25]

## 분석

노원구의 이용횟수가 압도적으로 높음

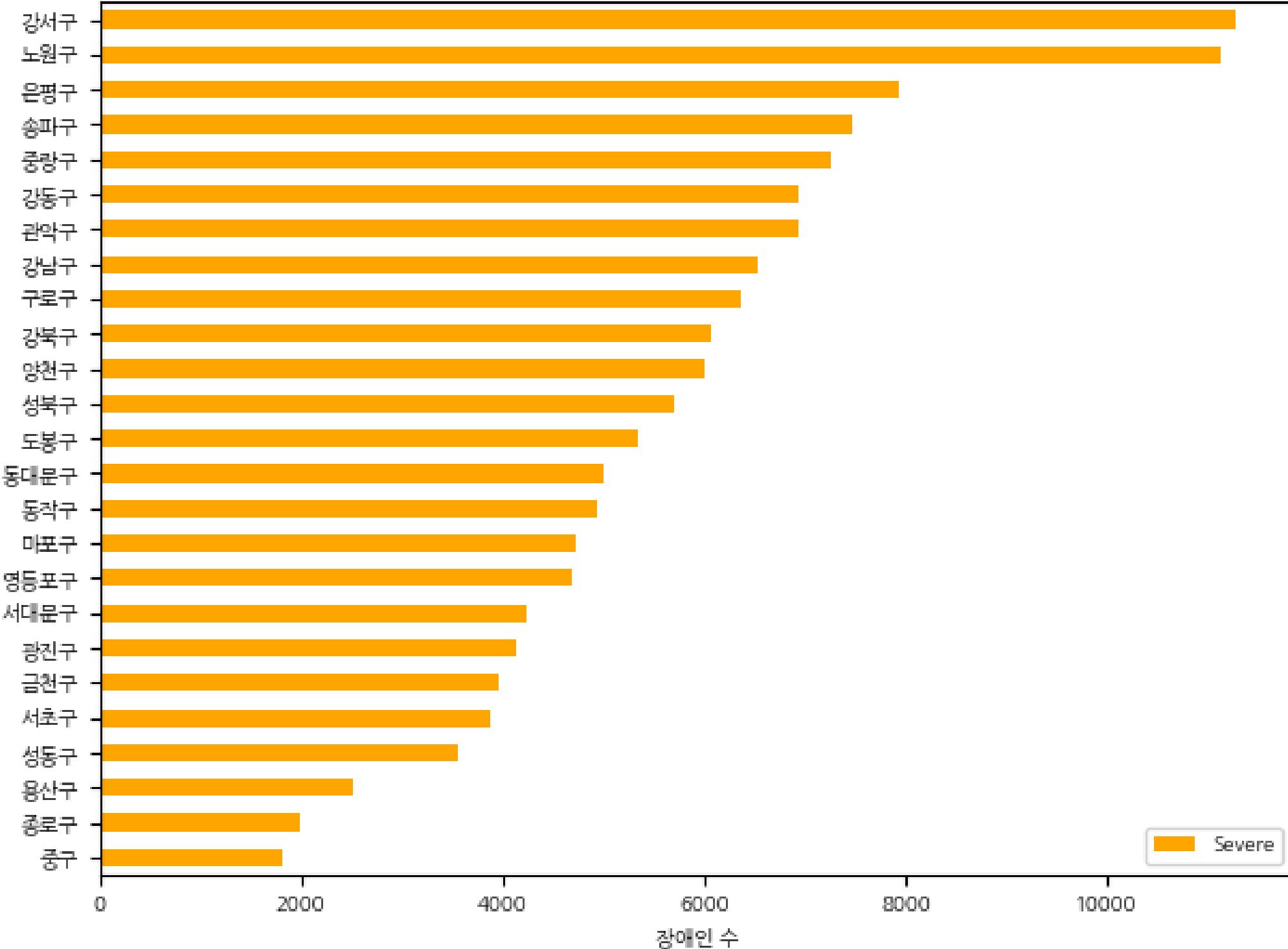
이용횟수의 Range :

524(min)~3859(max)

max 값은 min 값의 7배 수준

행정구역별 이용횟수의 차이가 큼

# 행정구역별 중증 장애인 수



```
data_df_cleaned = data_df_cleaned.dropna(subset=['Region2'])  
data_df_cleaned = data_df_cleaned[data_df_cleaned['Region2'] != '소계']  
  
data_df_sorted =  
    data_df_cleaned.sort_values(by='Severe',  
                                ascending=False)  
severe = data_df_sorted[['Region2',  
                           'Severe']].set_index('Region2')
```

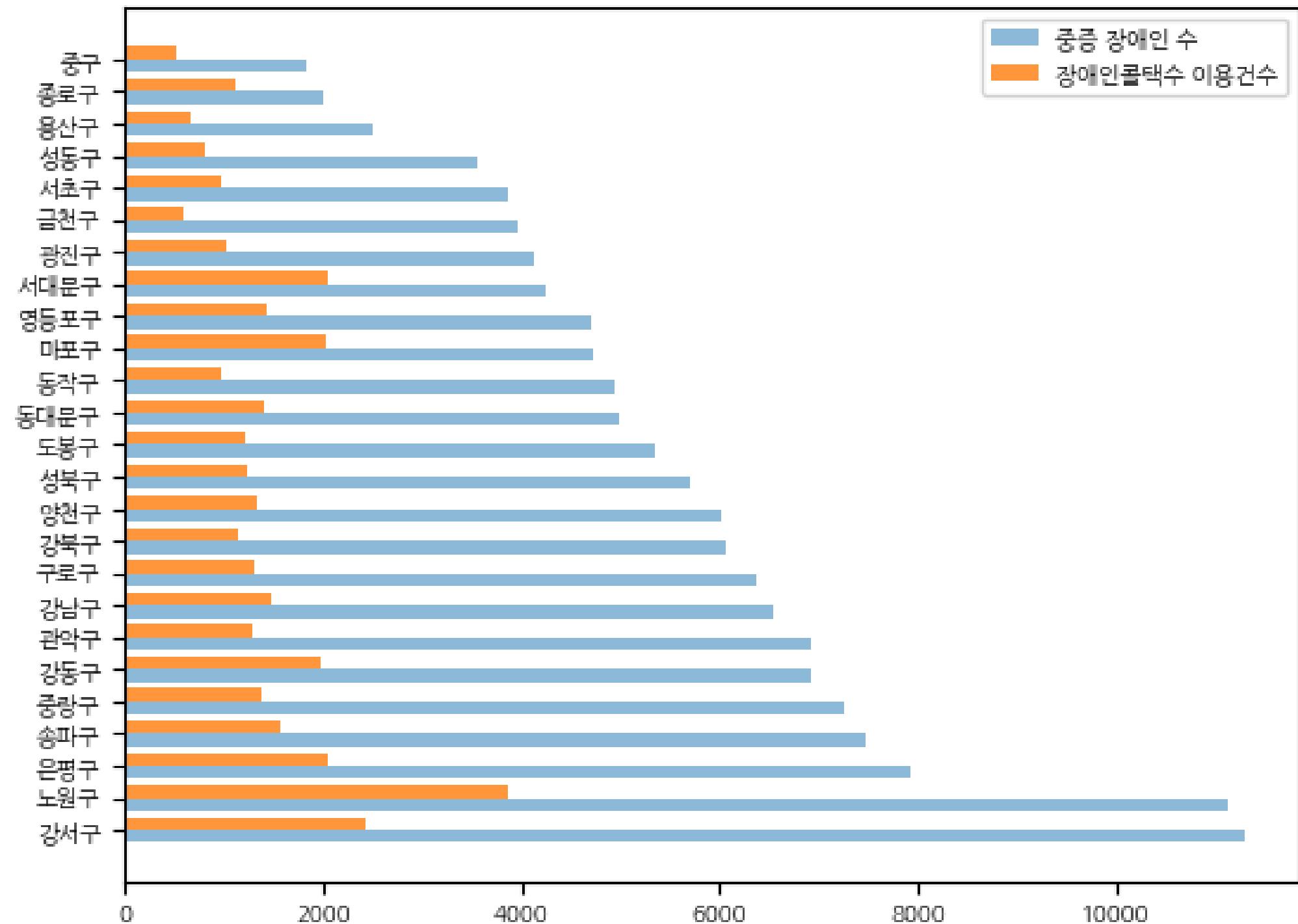
## 분석

중증 장애인 수의 Range :  
1827(min) ~ 11290(max)

max 값은 min 값의 약 6배 수준

행정구역별 중증 장애인 수 차이 매우 큼

# 이용횟수와 중증 장애인 수



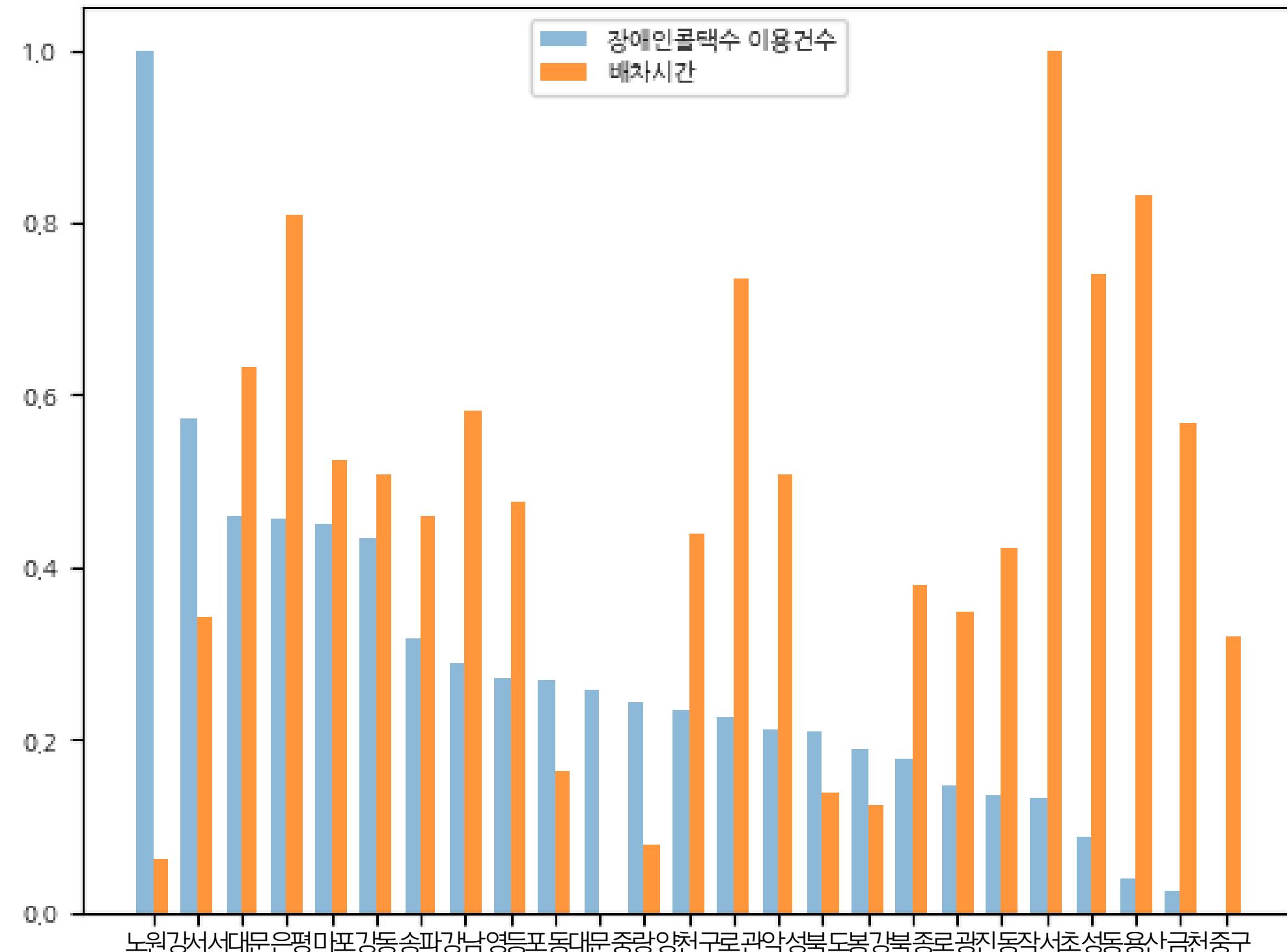
#행정구역을 기준으로 join

df\_join = severe.join(topSTARTPOS1\_slec)

## 분석

중증 장애인 수와 이용횟수의  
양의 상관관계(0.79) 파악 가능

# 이용횟수와 배차대기시간



값의 Scale 차이로 인해 그래프 그리기에 부적합  
-> **MinMaxScaler** 사용 후 그래프 생성

#행정구역을 기준으로 join

```
df_join2 =  
topSTARTPOS1_slec.join(pos_median_slec)
```

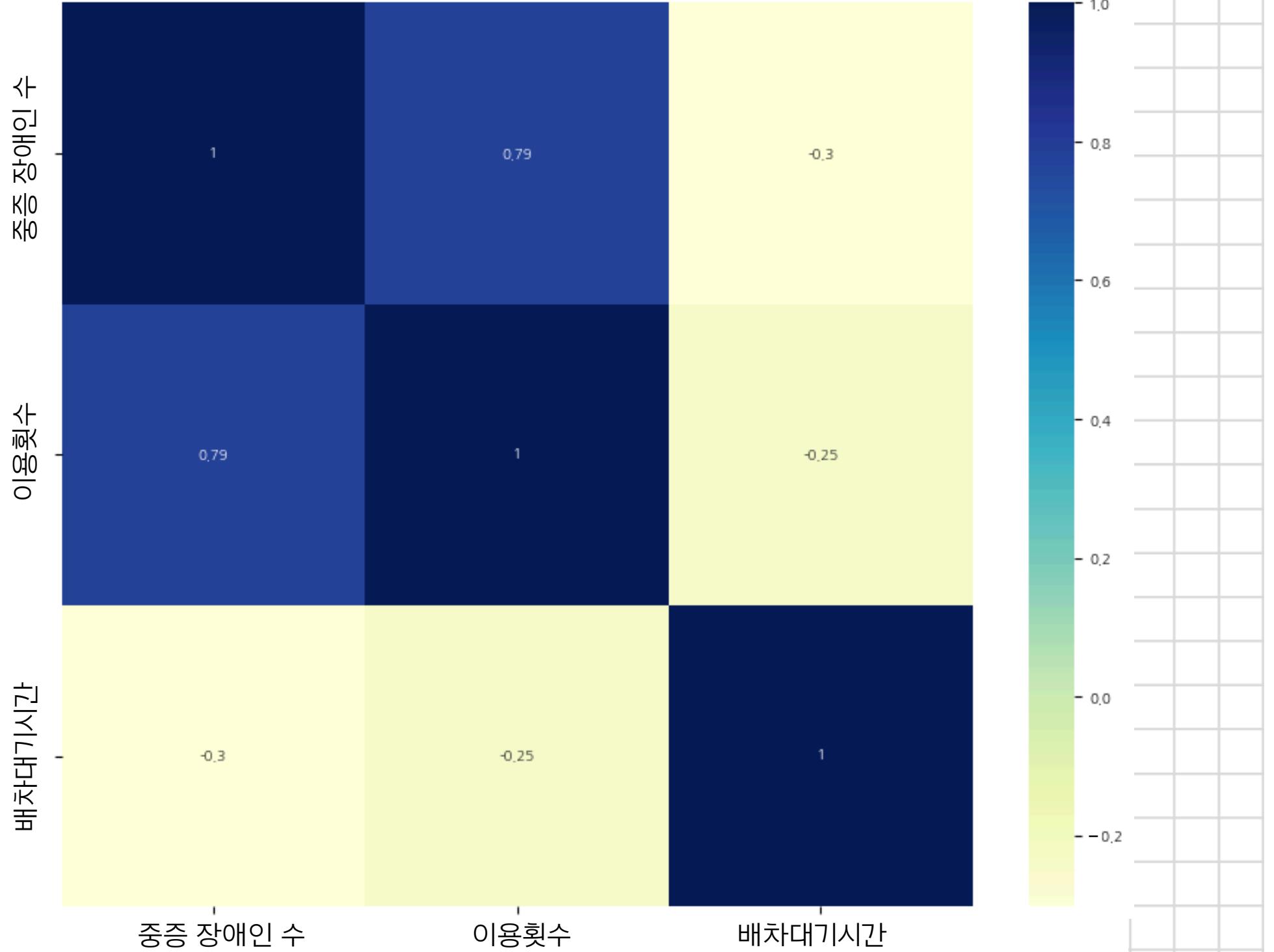
## 분석

수요가 많은 행정구역(이용횟수가 많은)에  
배차가 빨리될 것이라 예측

낮은 음의 상관관계(-0.25)

극단의 값을 볼 때는 상관관계가 있어 보이나,  
일반화하기 어려움

# Heat Map



#앞서 사용한 DataFrame merge

#행정구역별 중증 장애인 수, 이용횟수, 배차대기시간간  
df\_join3 = df\_join.merge(df\_join2, on='STARTPOS1')

## 분석

이용횟수와 중증 장애인 수의 **높은 양의 상관관계**

예상대로 중증 장애인 수가 많은 행정구역에서  
장애인 콜택시 이용 건수가 많음

이용횟수와 배차대기시간, 중증 장애인 수와 배차대기시간  
**낮은 음의 상관관계**

수요 많은 행정구역에  
배차가 빨리 될 거라 예상했지만  
수치적으로는 상관관계가 없음

# 솔루션

## 대기시간 격차 해소 필요



대기시간의 min / max 차이와 표준편차가  
매우 큰 수준

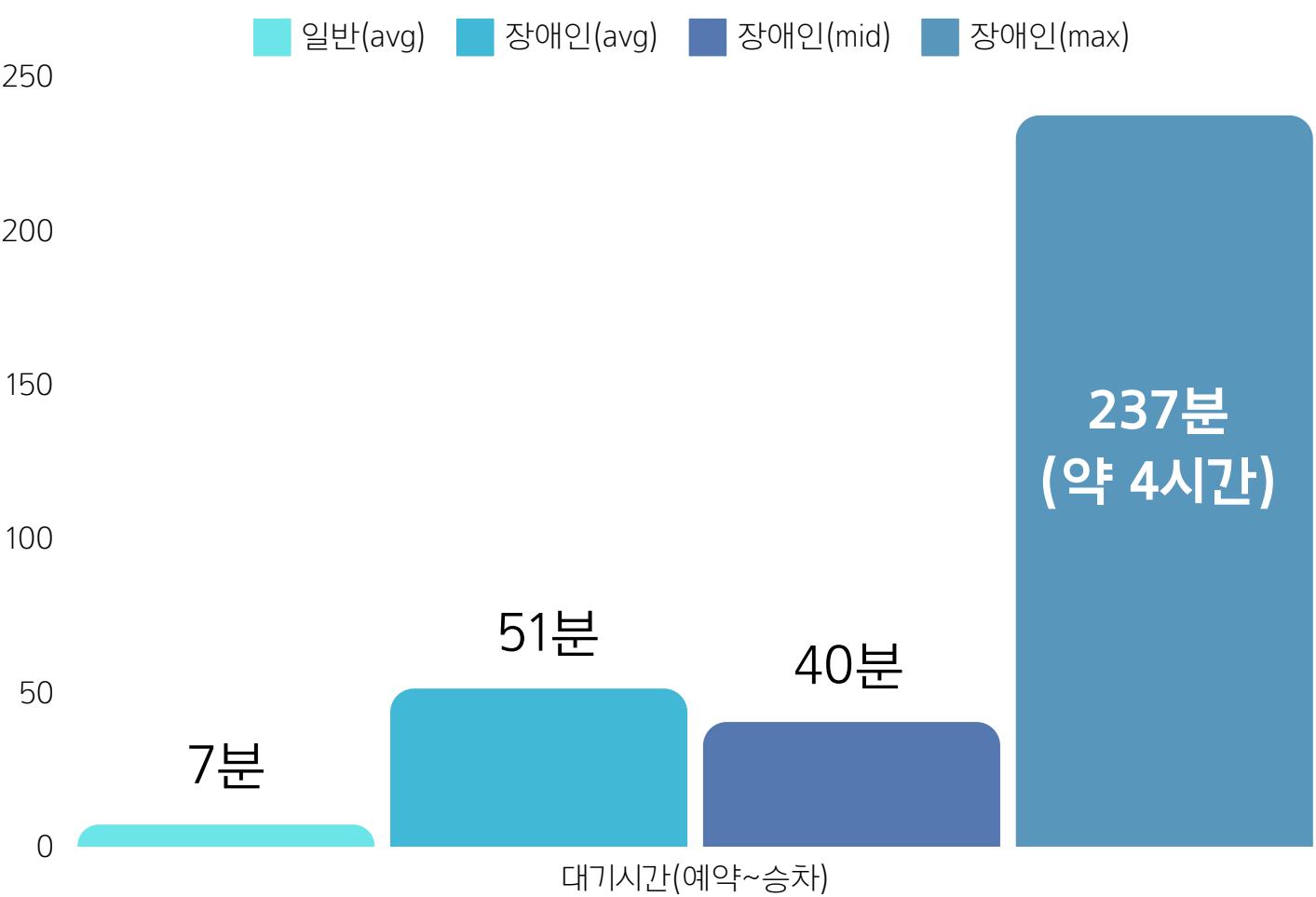
-> 고객이 대기시간을 예측하기 어려움

“5분이 될지, 2시간이 될지 모르는 대기시간에 대한  
불확실성이 가장 큰 불안 요소”

“내가 원하는 시간에 콜택시를 부르는 게 아니라,  
콜택시가 원하는 시간에 내가 맞춰야 한다.”

# 솔루션

## 일반택시 대기시간과 비교



## 대상자 수 대비 부족한 차량

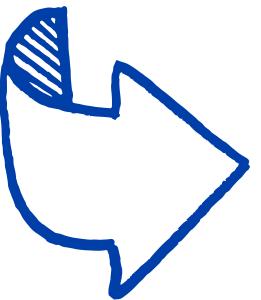
대상자 수 대비 장애인콜택시 차량 대수의 비율 0.5%



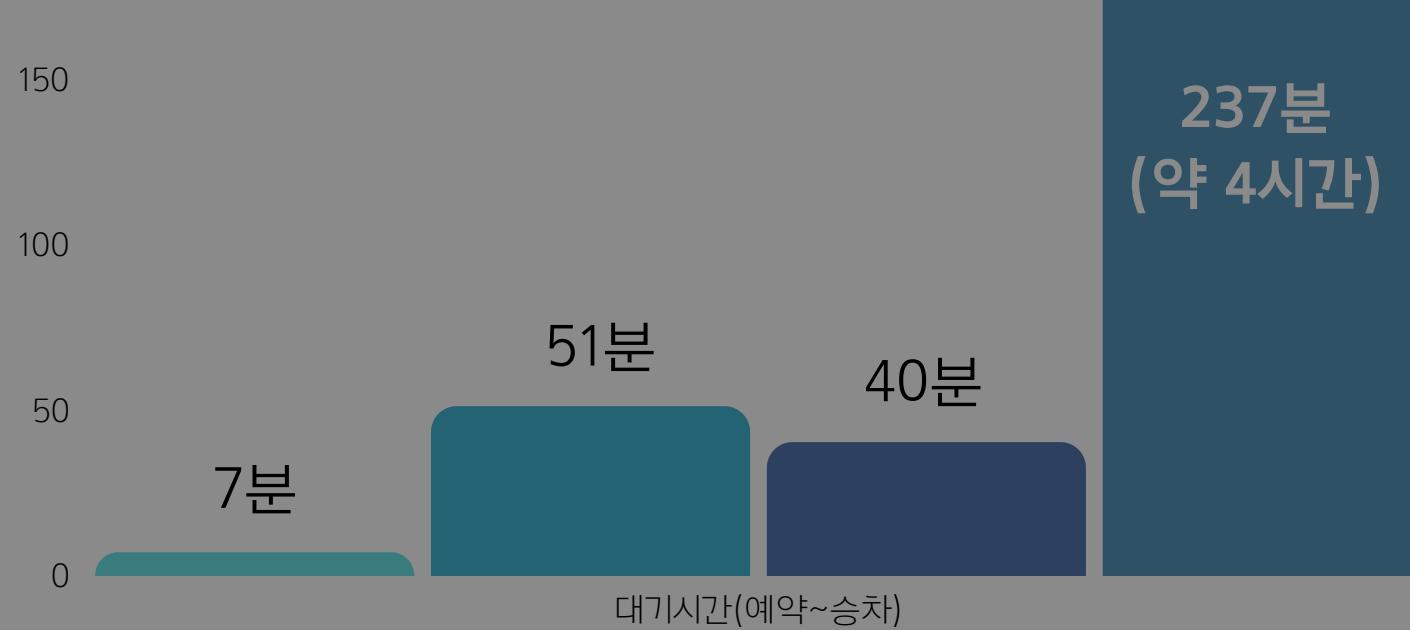
# 솔루션

인바택시 대기시간과 비교

대상자 스 대비 브조한 차량



## 택시 증차, 운전원 충원





# 역할 분담



이정인

중국언어문화학과 19  
조장, 수집, 전처리, 발표



장수연

경영학과 21  
기획, 분석, 솔루션, 시각화, 발표, PPT



김나현

회계학과 21  
크롤링, 텍스트마이닝, 솔루션, PPT



박요한

인공지능학과 22  
전처리, 정리, 분석, 시각화

〈참고 자료〉

2022년 택시서비스 시민만족도 조사, 서울연구원, 안기정 외 2인

윤유경, “장애인 콜택시 대기시간 32분? 연합뉴스 팩트체크가 지워버린 ‘현실’”, 미디어오늘,

<https://www.mediatoday.co.kr/news/articleView.html?idxno=303348>

원샷한솔 유튜브