빅데이터활용센터

데이터 분석 및 컨설팅

지원사례집 2024

BIG Data

IG Data

대구광역시 DAEGU METROPOLITAN CITY
대구디지털혁신진흥원 Daegu Digital Innovation Promotion Agency
대구빅데이터활용센터 DAEGU BIG DATA

빅데이터활용센터

데이터 분석 및 컨설팅

지원사례집 2024

BIG Data

IG Data

대구광역시 DAEGU METROPOLITAN CITY
대구디지털혁신진흥원 Daegu Digital Innovation Promotion Agency
대구빅데이터활용센터 DAEGU BIG DATA

BigData

Contents

빅데이터

활용센터

Case

지원사례

2
0
2
4

빅데이터활용센터

데이터 분석 및 컨설팅
지원사례집 2024

분석지원

Big Data

접수번호 2024-1

동티모르 교육 관리 정보 시스템(EMIS) 현황 분석

분석 요청자	소속	기관	성명	박OO
	휴대전화	-	전자우편	-
분석유형	■ 데이터 전처리	■ 데이터 시각화	□ 머신러닝	□ 기타
데이터 분석가	김건욱 센터장, 장원준 전임			
분석 기간	2024.02 ~ 2024.02(1개월)			

가.

분석 주제

분석 주제(요청내용)

- 동티모르 국가와 연계하여 이러닝 국제 컨설팅 사업 보고서를 작성 중에 있으며, 동티모르 국가의 교육 관리 정보 시스템(EMIS)에 적재할 데이터를 활용해 현황을 파악하고자 함

분석 필요성 및 전략

- 국내에서는 교육행정의 정보화와 교육 시스템의 디지털화를 지원하기 위해 교육정보 통계시스템으로 정보공시 현황을 제공하고 수요자 중심의 정보공시 서비스를 운영함
- 해당 서비스에서는 초·중등학교 정보공시제를 기반으로 하여 교육부에서 정한 공시 기준에 따라 매년 1회 이상 학교알리미에 공시하고 있으며, 학생·교원현황·시설·학교폭력발생현황·위생·교육여건·재정상황·급식상황·학업성취 등과 같은 학교의 주요 정보들을 쉽게 확인할 수 있음
- 이에 본 서비스와 유사한 수준의 데이터 시각화 현황판을 제작하고자 함

<그림 1> 학교알리미 공식 홈페이지

나.

데이터 분석 및 결과

활용 데이터

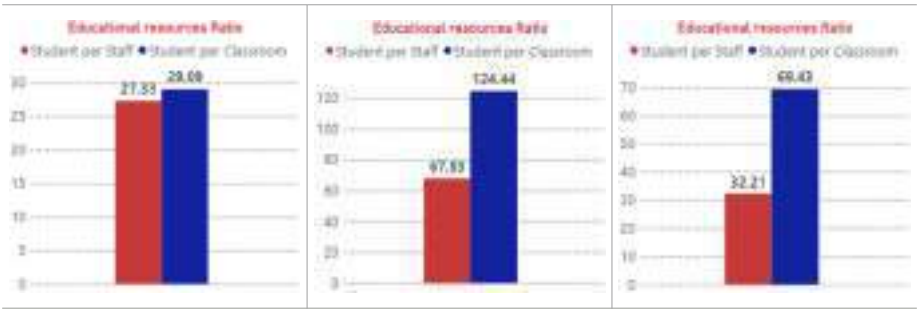
- 본 분석에서는 동티모르 교육정보 공시 시스템을 구축하기 위해 총 1,453개 학교를 대상으로 조사한 “Annual School Questionnaire” 설문조사 응답자료를 사용함
- 해당 자료는 학교 설립 및 위치, 종교 등에 대한 기본 정보와 재원 현황 및 학생/교사 현황에 대한 자료를 다루고 있음

구분	세부 내용	비고
동티모르 EMIS 정보	School name, Level, Type, Grade, Students, Classrooms, Staff, Meja, Kadeira, Sala Biblioteka, Toilet, Water source, Eletricity Source etc.	-

<표> 동티모르 EMIS 데이터

데이터 분석 결과

- 동티모르는 총 학생수 366,171명, 교사수 14,029명으로 교사 1인당 담당 학생수가 약 26.1명 정도이며, Basic School 27.33명, Secondary 67.83명, Technical 32.21명으로 Secondary School의 교원 부족이 가장 높은 것으로 확인됨
- 또한 총 학급수는 13,480개로 한 학급당 담당 학생수가 약 27.16명 정도이며, Basic School 29.09명, Secondary 124.44명, Technical 69.43명으로 교사 1인당 담당 학생수와 동일하게 Secondary, Technical, Basic School 순으로 나타나며, Secondary School의 학급 병목이 클 것으로 보임

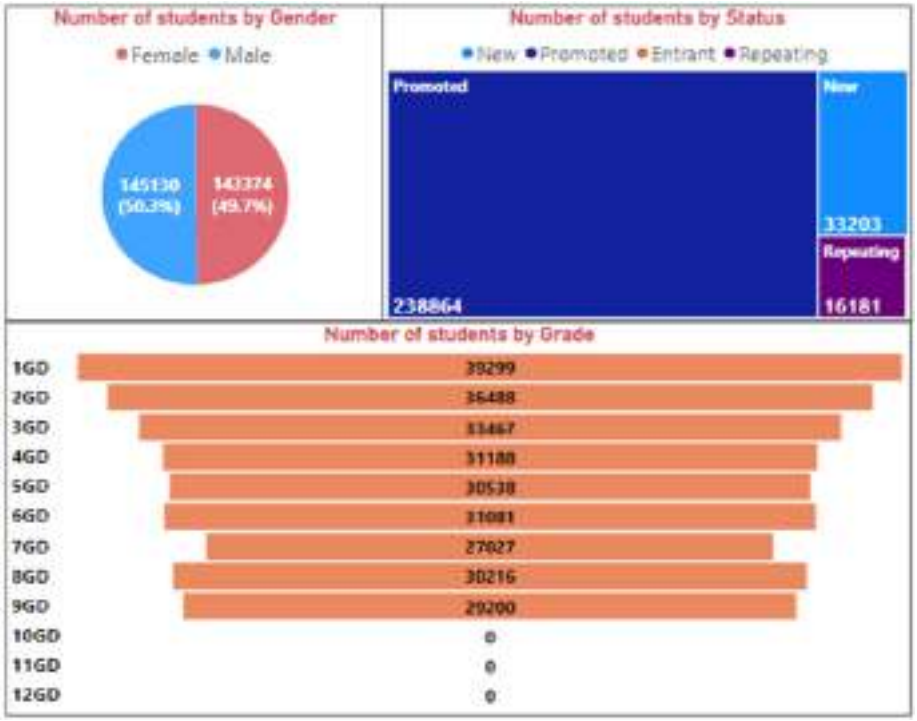


Basic School	Secondary School	Technical School

<그림 2> 교사 및 학급 현황

- 성, 등급별, 상태별 학생 현황을 확인해본 결과 Basic School의 경우 남학생 145,130명(50.3%), 여학생 143,374명(49.7%)로 성비가 균등하며 승급생이 238,864명으로 가장 많고 다음으로 신입생 33,203명 복학생 16,181명으로 나타남

- 학년의 경우 1학년부터 9학년까지 구성되며 1학년이 39,299명으로 가장 많고 7학년이 27,027명으로 가장 적은 것으로 나타남



<그림 3> Basic School 학생 현황

- 학교 도서관 현황에 대해 학교 유형, 종교, 등급별로 확인한 결과 상, 중 등급에서 모든 학교에서 Private이 보유한 평균 도서관 수가 가장 높았으나 하 등급에서는 Catholic이 가장 높은 것으로 확인됨

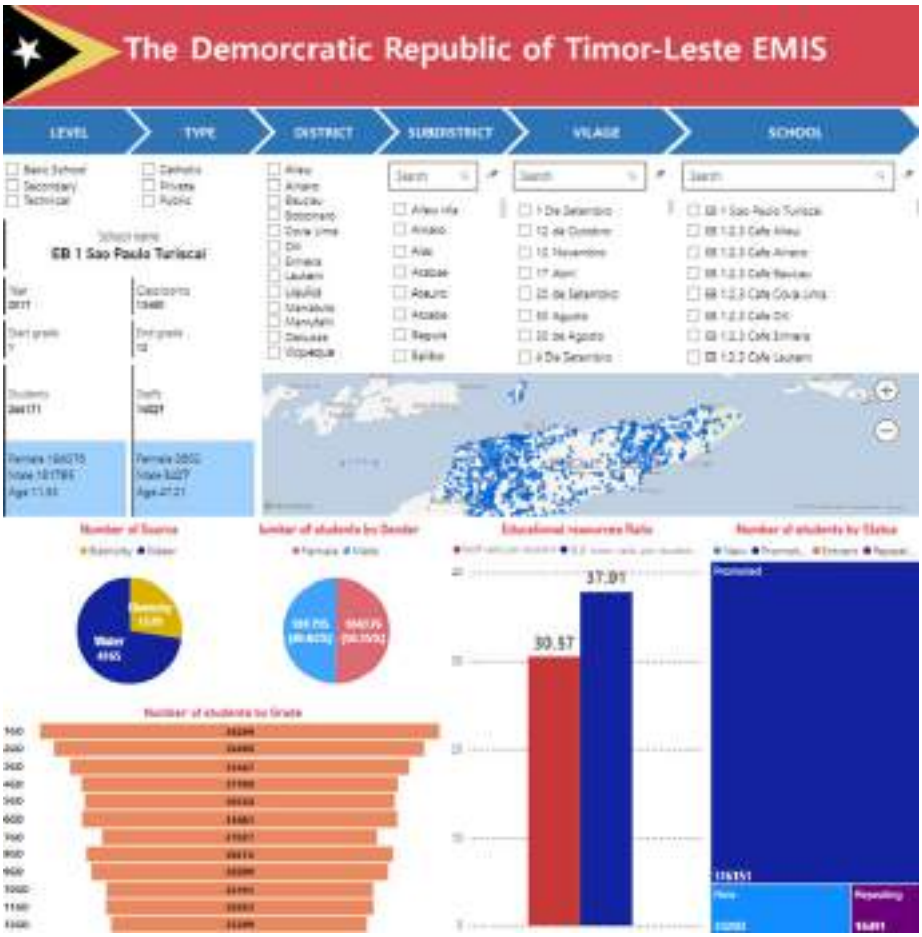
<그림 4> 도서관 재원 현황

다.

분석 활용 전략

동티모르 EMIS PowerBI 구축

- 해당 분석에 대해 자유롭게 EMIS 공시자료 정보를 파악할 수 있도록 PowerBI 시각화 툴을 활용하였으며, 국내 학교알리미와 유사한 수준으로 학교 수준, 종교, 도시를 선택함에 따라 학급과 학생 현황을 파악할 수 있음



<그림 5> 학교 현황 대쉬보드

Appendix

```
import pandas as pd
df = pd.read_excel('EMIS_class.xlsx')

class_melt_cols = [col for col in df.columns if col.startswith('values')]
df_class_melted = pd.melt(df, id_vars=['school_name', 'level_name', 'type_name', 'district_
name', 'Students', 'Classrooms','class'],
                value_vars=class_melt_cols,
                var_name='var_name', value_name='values')

import pandas as pd
df = pd.read_excel('EMIS_std_total_furniture.xlsx')

class_melt_cols = [col for col in df.columns if col.startswith('std_total_furniture_')]
df_std_total_furniture_melted = pd.melt(df, id_vars=['school_name', 'level_name', 'type_
name', 'district_name', 'Students', 'Classrooms','std_total_furniture'],
                value_vars=class_melt_cols,
                var_name='var_name', value_name='values')
df_std_total_furniture_melted
df_std_total_furniture_melted.to_excel("EMIS_std_total_furniture2.xlsx", index=False)

df = pd.read_excel('EMIS_prof_total_furniture.xlsx')
melt_cols = [col for col in df.columns if col.startswith('prof_total_furniture_')]
df_melted = pd.melt(df, id_vars=['school_name', 'level_name', 'type_name', 'district_name',
'Students', 'Classrooms','prof_total_furniture'],
                value_vars=melt_cols,
                var_name='var_name', value_name='values')

def categorize_var(x):
  if x == "prof_total_furniture_values_per_Classroom":
    return "Classroom"
  elif x == "prof_total_furniture_values_per_Student":
    return "Student"
  else:
    return "Value"

def categorize_furniture(x):
  if x == "Aat":
    return "Lower"
  elif x == "Diak":
    return "Upper"
  else:
    return "Middle"
```

```
def categorize_url1(x):
  if x == "Aat":
    return "https://cdn-icons-png.flaticon.com/512/3389/3389176.png"
  elif x == "Diak":
    return "https://cdn-icons-png.flaticon.com/512/3179/3179672.png"
  else:
    return "https://cdn-icons-png.flaticon.com/512/3179/3179682.png"

def categorize_url2(x):
  if x == "Student":
    return "https://cdn-icons-png.flaticon.com/512/5310/5310717.png"
  elif x == "Classroom":
    return "https://cdn-icons-png.flaticon.com/512/2422/2422454.png"
  else:
    return "https://cdn-icons-png.flaticon.com/512/684/684831.png"

df_melted['class2'] = df_melted['prof_total_furniture'].apply(categorize_furniture)
df_melted['URL'] = df_melted['prof_total_furniture'].apply(categorize_url1)
df_melted['var_name2'] = df_melted['var_name'].apply(categorize_var)
df_melted['URL2'] = df_melted['var_name2'].apply(categorize_url2)
```

접수번호 2024-2

정보공개 청구서 및 사전정보 주요 키워드 시각화

분석 요청자	소속	지자체	성명	구OO
	휴대전화	-	전자우편	-
분석유형	■ 데이터 전처리	■ 데이터 시각화	▢ 머신러닝	▢ 기타
데이터 분석가	김건욱 센터장, 장원준 전임			
분석 기간	2024.03 ~ 2024.03 (1개월)			

가.

분석 주제

분석 주제(요청내용)

- 정보공개제도란 국가기관·지방자치단체 등 공공기관이 업무 수행 중 생산·접수하여 보유· 관리하는 정보를 국민에게 공개함으로써, 국민의 알권리를 보장하고 더 많은 정보를 바탕으로 국정운영에 대한 참여를 유도하기 위한 제도임
- 정보 공개 포털(www.open.go.kr)을 통해 시민들은 공공기관이 직무상 작성 또는 취득하여 관리하고 있는 문서(전자문서 포함) · 도면 · 사진 · 필름 · 테이프 · 슬라이드 및 기타 이에 준하는 매체 등에 기록된 사항을 청구할 수 있음
- 각 지자체 및 공공기관에서는 국민들이 정보공개를 청구하기 전에 국민이 필요로 하는 정보를 선제적·능동적 공개하는 사전정보 공표 제도를 함께 운영하며 비공개 대상 정보 외에 국민이 알아야 할 필요가 있는 모든 정보, 즉 사전정보대상을 매년 선별함
- 이에 향후 사전정보 목록을 정보공개 청구현황을 기반으로 업데이트하기 위해 각각의 주요 출현하는 키워드를 비교하고자 함

분석 필요성 및 전략

- 요청받은 청구서 중 지속해서 중복하여 요청하는 청구서가 있어 실질적으로 수요가 높은 청구 유형이 무엇인 지 파악할 필요가 있음
- 본 분석에서는 청구서 현황 내 악성 형태의 중복되는 청구서를 제거하고 텍스트 처리 방식을 활용해 청구서 현황 및 사전정보 목록의 주요 명사를 추출하여 비교하고자 함

나.

데이터 분석 및 결과

활용 데이터

- 본 분석에서 활용된 데이터는 아래와 같이 서구청에서 2021년부터 2개년간 요청받은 청구서와 2015년부터 약 9개년간 개방한 사전정보 목록 두 가지 주요 데이터 구성됨
  1) 대구 서구 청구서 현황: 대구 서구청 정보담당관으로부터 개인정보가 제거한 상태로 제공받았으며, 접수번호, 접수일, 청구제목, 청구내용 등이 포함된 총 3,811건의 데이터
  2) 대구 서구 사전정보 목록: 사전정보번호, 분류체계, 사전정보 제목, 시기, 주기 등이 포함된 총 491건의 데이터

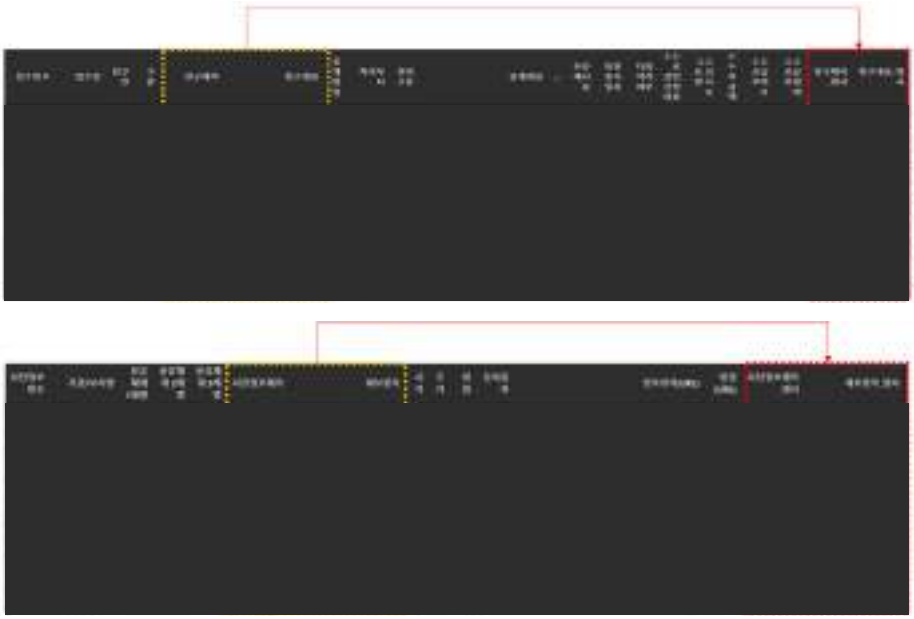
구분	데이터명	항목	건수	기간
1	대구 서구 청구서 현황	접수번호, 접수일, 구분, 청구제목, 청구내용, 공개방법, 처리부서, 결정구분, 공개내용, 비공개(부분공개)내용 및 사유, 비공개사유, 결정통지일, 공개일, 공개형태, 교부방법 등	'21.01.01 ~'22.12.31	3,811
2	대구 서구 사전정보 목록	사전정보번호, 기관/부서명, 분류체계 1레벨, 분류체계 2레벨, 분류체계 3레벨, 사전정보제목,세부항목, 시기, 주기, 방법, 등록일자 등	'15.03.13 ~'23.06.22	491

<표> 활용 데이터

데이터 분석(전처리)

– 데이터 가공

- 첫 번째 데이터 파일에 대해서는 '청구제목'과 '청구내용' 열을 결합한 후 해당 열에서 명사를 추출해 이후 청구서에 어떤 주제가 많이 나타나는지 파악하고자 함. 예를 들어, "의료비 청구"라는 청구서의 경우 "의료비"와 "청구"가 명사로 추출될 것임
- 동일하게 두 번째 데이터 파일에 대해서는 '사전정보제목'과 '세부항목' 열을 결합한 후 해당 열에서 명사를 추출하여 사전정보 목록에 어떤 종류의 정보가 포함되어 있는지를 파악함
- 파이썬에서 제공하는 Okt 라이브러리를 기반으로 명사 추출 작업을 통해 텍스트 데이터를 단어 단위로 분해하고, 이를 통해 주요 주제 및 항목을 파악함
- 추출된 명사를 새로운 열에 저장하여 기존 데이터프레임에 추가함



<그림 1> 명사 추출 결과

다.

분석 활용 전략

결론

- 각 데이터 별로 추출된 주요 출현 명사를 시각화하기 위해 PowerBI 시각화 툴을 활용해 워드클라우드를 제작함
- 청구서 현황의 경우 지도, 점검, 시설, 등록, 위반, 관리 등 지역 내 정비 사업 및 위생, 시설 점검 관련 요청이 주로 나타나는 것으로 확인됨
- 사전정보 목록의 경우 주택, 정비, 건축, 관내, 재정, 재개발 등 주택 재개발 관련 정보를 주로 공개해 온 것으로 확인됨
- 따라서 향후 위생 점검 및 시설 관리 관련 정보를 사전정보 목록으로 개방할 필요가 있을 것으로 판단됨

청구서 현황

사전정보 목록

<그림 2> 데이터별 워드클라우드

Appendix

```
import pandas as pd
from konlpy.tag import Okt


df= pd.read_excel('대구서구 청구서전체현황(2021-01-01_2022-12-31).xlsx')


okt = Okt()


# Define a function to extract nouns from a single piece of text
def get_nouns(text):
    return okt.nouns(text)


# Apply the function to the specified column and create a new column with the results
df['청구제목_명사'] = df['청구제목'].apply(get_nouns)
df['청구내용_명사'] = df['청구내용'].apply(get_nouns)

df.to_csv('청구서 현황 전처리(서구).csv', encoding='utf-8-sig', index=False)

df = pd.read_excel('대구서구 사전정보 목록(시스템 다운).xlsx')

from konlpy.tag import Okt

okt = Okt()

def get_nouns(text):
    return okt.nouns(text)

df['사전정보제목_명사'] = df['사전정보제목'].apply(get_nouns)
df['세부항목_명사'] = df['세부항목'].apply(get_nouns)

df.to_csv('사전정보공표 현황 전처리(서구).csv', encoding='cp949', index=False)
```

접수번호 2024-3

전기차 충전소 현황판 제작

분석 요청자	소속	기관	성명	이OO
	휴대전화	-	전자우편	-
분석유형	■ 데이터 전처리	■ 데이터 시각화	▢ 머신러닝	▢ 기타
데이터 분석가	김건욱 센터장, 장원준 전임			
분석 기간	2024.03 ~ 2024.03 (1개월)			

가.

분석 주제

분석 주제(요청내용)

- 대구시에서는 전기자동차 사용확대를 지원하고자 전기차 충전소를 운영하고 있으며 대구공공시설관리공단에서 직영으로 운영하고 있는 충전기와 위탁하여 운영하고 있는 민간 충전기로 구분됨
- 2024년 현재 기준 총 302개소가 대구시 전역에 운영되고 있으며, 수성구가 68개소로 가장 많고 달서구가 56개소로 2위, 북구가 54개소로 3위이며 중구가 17개로 가장 적은 것으로 나타남
- 향후 증가하는 전기차 수요를 대비해 지역별, 충전소별 충전 수요를 정량적으로 파악하기 위해 현황판 제작을 요청함

분석 필요성 및 전략

- 전기차 충전소를 충전시간, 충전금액, 충전량 3가지 측면에서 구분하여 파악해 지역별, 충전소 별 수요를 파악할 필요가 있음
- 이를 시기별, 충전소 유형(급속/완속) 별로 조건을 부여하여 파악할 수 있게 제공하기 위해 BI 툴을 활용한 현황판을 제작하고자 함

나.

데이터 분석 및 결과

활용 데이터

- 본 분석에서 활용된 데이터는 아래와 같이 세 가지 주요 유형으로 구성
  1) 공단직영 충천소: 충전소명, 주소, 급속/완속, 충전시작일시, 충전종료일시, 사용전력(kWh), 사용요금(원) 항목이 존재하며 2020년부터 2022년까지 총 3개년간 축적된 데이터로 이루어짐
  2) 위탁직영 충천소: 공단직영과 유사하게 충전소명, 주소, 급속/완속, 충전시작일시, 충전종료일시, 사용전력(kWh), 사용요금(원) 항목이 존재하며 2020년부터 2022년까지 약 3개년간 축적된 데이터로 이루어짐
  3) 충천소 현황: 충전소 명과 주소로 이루어짐

구분	데이터명	항목	건수	기간
1	공단직영	충전소명, 주소, 충전소ID, 급속/완속, 충전시작일시, 충전종료일시, 사용전력(kWh), 사용요금(원)	334,506	'20.01.01 ~'22.12.31
2	위탁운영	충전소명, 주소, 충전소ID, 급속/완속, 충전시작일시, 충전종료일시, 사용전력(kWh), 사용요금(원)	71,618	'20.01.01 ~'22.12.31
3	충전소 현황	충전소 명, 주소	302	'24 기준

<표> 활용 데이터

데이터 분석(전처리)

- 데이터 분석 과정은 아래와 같음

1) 공단직영 및 위탁운영 데이터를 표준화하여 동일한 데이터 프레임으로 결합

2) 날짜와 시간의 결합: '충전시작일자'와 '충전시작시간', '충전종료일자'와 '충전종료시간'을 결합하여 새로운 datetime 열을 생성. 이 과정에서 오류가 발생하면 해당 행에는 NaT(Not-a-Time) 값 할당

3) 데이터 병합: 공단 직영과 위탁 운영 충전소 데이터프레임을 병합하고, 누락된 데이터가 있는 행 제거

4) 위경도 정보 추가: 충전소 현황에서 충전소 주소를 기반으로 위,경도를 추출할 수 있는 Geocoding 실시, 메인 데이터프레임과 병합

5) 충전량(kWh) 정제: '충전량(kWh)' 열에서 'Wh' 문자열을 제거하고, 비어 있는 값은 0으로 대체한 후, 해당 열을 실수형으로 변환

6) 충전 시간 변환: '충전시간' 열의 값을 분 단위로 변환하는 함수를 적용합니다. 이 함수는 시간 형식의 문자열을 받아 총 분으로 변환

7) 충전 금액 정제: '충전금액(원)' 열에서 쉼표를 제거하고, 정수형으로 변환

8) 최종적으로 충전소의 위치 정보(위도, 경도), 충전량, 충전 시간, 충전 금액 등 충전소 관련 중요 정보를 포함(상세 코드 첨부)

	충전소	주소	충전기	급속/완속	충전시작일	충전종료일	충전시간	충전량(kWh)	충전금액(원)	충전소 명	위도	경도
0	DTC섬유박물관	팔공로 227(지하1층 T6)	1	급속	2020-01-01 10:01	2020-01-01 10:28	26	13.43	2334	DTC섬유박물관	35.957646	128.662033
1	DTC섬유박물관	팔공로 227(지하1층 T6)	1	급속	2020-01-01 11:28	2020-01-01 11:31	3	1.41	245	DTC섬유박물관	35.957646	128.662033
2	DTC섬유박물관	팔공로 227(지하1층 T6)	1	급속	2020-01-01 11:46	2020-01-01 12:01	15	7.87	1367	DTC섬유박물관	35.957646	128.662033
3	DTC섬유박물관	팔공로 227(지하1층 T6)	1	급속	2020-01-01 12:10	2020-01-01 12:47	37	16.62	2888	DTC섬유박물관	35.957646	128.662033
4	DTC섬유박물관	팔공로 227(지하1층 T6)	1	급속	2020-01-01 13:04	2020-01-01 13:26	22	10.99	1910	DTC섬유박물관	35.957646	128.662033

<그림 1> 최종 데이터 프레임

다.

분석 활용 전략

현황판 제작 지원

- PowerBI를 활용해 현황판을 제작하였으며 상세 기능 내용은 아래와 같음
- 전반적으로 대시보드는 EV 충전소 사용 및 배포를 시각적으로 표현하며, 다년간의 추세, 사용 패턴 및 EV 인프라 성장을 분석
- 대시보드는 대화형이므로 사용자가 특정 데이터 포인트와 추세를 더 깊이 파고들 수 있음
- 상단의 그래프는 일일 충전시간, 충전금액, 충전량의 트렌드를 나타내며 빨간색 마커로 표시된 곳은 특정 이상치 및 변동성이 큰 시점을 나타냄. 추세선은 데이터의 일반적인 증가 또는 계절성을 나타냄.
- 하단의 지리적 지도에는 EV 충전소의 위치를 나타내며 하단 우측의 기간과 충전소 종류, 충전소 명을 조절하여 현황판을 확인할 수 있음

<그림 2> 대구 공공시설관리공단 전기차 충전소 현황판

Appendix

```
import glob

# Pattern to match files related to corporation and consignment
corporation_files_pattern = '*공단직영.csv'
consignment_files_pattern = '*위탁운영.csv'
```

```
# Using glob to find files that match the patterns
corporation_files = glob.glob(corporation_files_pattern)
consignment_files = glob.glob(consignment_files_pattern)


# Reading and concatenating the corporation files
df_corporation = pd.concat([pd.read_csv(file, encoding='cp949') for file in corporation_
files])


# Reading and concatenating the consignment files
df_consignment = pd.concat([pd.read_csv(file, encoding='cp949') for file in consignment_
files])


# Function to combine date and time into a datetime, handling errors gracefully
def combine_date_time(row, date_col, time_col):
    date_time_str = f"{row[date_col]} {row[time_col]}"
    try:
        return pd.to_datetime(date_time_str)
    except ValueError:
        return pd.NaT  # Return Not-a-Time for rows with parsing errors


# Apply the function to create new datetime columns
df_consignment['충전시작일'] = df_consignment.apply(combine_date_time, date_col='충전
시작일자', time_col='충전시작시간', axis=1)
df_consignment['충전종료일'] = df_consignment.apply(combine_date_time, date_col='충전
종료일자', time_col='충전종료시간', axis=1)


df_consignment = df_consignment[['충전소명', '주소', '충전기ID', '급속/완속', '사용전력(kWh)', '
사용요금(원)', '충전시작일', '충전종료일']]
df_consignment = df_consignment.rename(columns={'충전소명':'충전소','충전기ID':'충전기','
사용전력(kWh)':'충전량(kWh)','사용요금(원)':'충전금액(원)'})
df_electro = pd.concat([df_corporation, df_consignment]).dropna()


xy = pd.read_csv('충전소 현황 위경도좌표.csv', encoding='cp949')
df_electro_xy = pd.merge(df_electro, xy[['충전소 명','위도','경도']], left_on='충전소', right_on='
충전소 명', how = 'left')
```

```
df_electro_xy = df_electro_xy[~df_electro_xy['위도'].isna()]

df_electro2 = df_electro_xy.drop_duplicates().reset_index().drop(columns='index').
dropna()

# Function to clean and convert Charging amount (kWh)
def clean_kwh_amount(df):
    df['충전량(kWh)'] = df['충전량(kWh)'].str.replace('Wh', '').replace('', np.nan).astype(float).
astype(float).fillna(0.0)
    return df


# Function to convert Charging time to minutes
def convert_charging_time(df):
    def time_to_minutes(time_str):
        parts = time_str.split(':')
        # Depending on the number of parts, assign hours, minutes, and seconds
        if len(parts) == 3:
            hours, minutes, seconds = map(int, parts)
        elif len(parts) == 2:
            hours, minutes = map(int, parts)
            seconds = 0
        elif len(parts) == 1:
            hours = 0
            minutes = int(parts[0])
            seconds = 0
        else:
            raise ValueError(f"Unexpected time format: {time_str}")

        return hours * 60 + minutes + seconds / 60.0

    df['충전시간'] = df['충전시간'].apply(time_to_minutes).astype(int)
    return df


# Function to clean and convert Charging amount (KRW)
def clean_krw_amount(df):
    df['충전금액(원)'] = df['충전금액(원)'].str.replace(',', '').astype(int)
    return df


# Applying the functions to the DataFrame
df_electro2 = clean_kwh_amount(df_electro2)
df_electro2 = convert_charging_time(df_electro2)
df_electro2 = clean_krw_amount(df_electro2)
```

접수번호 2024-4

고속도로 터널 내 사고 데이터와 기상정보 데이터 시공간 결합

분석 요청자	소속	학생	성명	김OO
	휴대전화	-	전자우편	-
분석유형	■ 데이터 전처리	▢ 데이터 시각화	▢ 머신러닝	▢ 기타
데이터 분석가	김건욱 센터장			
분석 기간	2024.03 ~ 2024.03(1개월)			

가.

분석 주제

분석 주제(요청내용)

- 고속도로 터널 내에서 발생하는 사고는 종종 대규모의 인명 피해와 재산 손실을 초래, 이러한 사고들은 터널 내 특수한 환경 조건과 결합하여 더 큰 위험을 야기할 수 있음
- 기상 조건은 사고 발생률에 큰 영향을 미칠 수 있는 중요한 외부 요인으로, 눈, 비, 안개와 같은 조건은 운전자의 시야를 제한하고 도로의 상태를 악화시켜 사고위험 증가
- 이러한 연관성을 깊이 있게 분석하는 것이 필요하며, 효과적인 사고 예방 전략을 마련하고, 고속도로 터널의 안전성을 향상시킬 수 있는 방안을 도출하고자 함
- 이를 위해 기상청 종관기상관측자료(AOS), 기상청 지역별 상세관측자료(AWS)와 한국도로공사 교통사고 빅데이터를 시공간 결합하여 분석을 수행하기 위한 데이터 마트 구축을 요청

분석 필요성 및 전략

- 고속도로 터널 사고는 단순한 피해 규모를 넘어서 장시간의 도로 폐쇄와 같은 광범위한 사회적, 경제적 파급 효과를 발생
- 특히, 기상 조건이 사고 발생에 직접적으로 영향을 미칠 수 있는 경우, 이를 사전에 파악하고 대응할 수 있는 시스템이 필요
- 기상 조건에 따른 사고 발생 패턴을 이해함으로써, 위험한 기상 조건을 사전에 예측하고, 적절한 경고 시스템을 구축하여 운전자에게 정보를 제공
- 본 분석에서는 고속도로 터널 내에서 발생한 사고 데이터와 기상청에서 제공하는 기상청 종관기상관측자료(AOS) 및 기상청 지역별 상세관측자료(AWS) 결합하여 사용
- 데이터 분석 과정에서는 사고 데이터에 기온, 강수량, 습도 등의 기상 조건을 결합(left join)하며, 시·군 단위로 1시간 단위의 데이터를 반영
- 특정 지역의 기상 관측 자료가 2개 이상인 경우 평균값을 적용하여 분석의 정확성을 높이는 방법을 선택
- 따라서 사고 발생 시점과 위치에서의 기상 조건을 상세히 분석할 수 있는 데이터 마트를 구축하여 추후 분석시 사고 발생 위험을 높이는 기상 조건 식별

나.

데이터 분석 및 결과

활용 데이터

- 본 분석에서 활용된 데이터는 아래와 같이 두 가지 주요 유형으로 구성

  1) 고속도로 터널 사고 데이터: 사고 발생 일시, 위치(시도) 및 기타 관련 정보를 포함

  2) 기상청 종관기상관측자료(ASOS) 및 지역별 상세관측자료(AWS): 기온, 강수량, 습도 등 다양한 기상 조건의 정보를 포함합니다. 전국단위의 데이터로 상호간 결측값을 보완하는 역할

- 활용 데이터의 규모는 2,500만건이 넘는 방대한 자료로 Python 프로그래밍 언어를 활용하여 데이터 전처리 수행

구분	데이터 유형	데이터 내용	시간적 범위	데이터 규모
1	고속도로 터널 사고 데이터	사고 일시, 위치, 사고 관련 상세 정보 등	2013 ~2022	3,615
2	기상청 종관기상관측자료(ASOS)	기상 관측 일시, 위치, 기온, 강수량, 습도 등	2013 ~2022	7,536,284
3	기상청 지역별 상세관측자료(AWS)	기상 관측 일시, 위치, 기온, 강수량, 습도 등	2013 ~2022	25,719,072

<표> 활용 데이터

데이터 분석(전처리)

- 데이터 분석 과정은 아래와 같음

  1) 데이터 수집: 고속도로 터널 사고 데이터와 기상 데이터(ASOS, AWS)를 의뢰인으로 부터 수집, 이상치 및 결측치 등 데이터 품질 검사

<표> 데이터 품질 검사(구문정확성, 의미정확성 등)

2) 데이터 가공: 수집된 데이터를 시공간 결합에 적합한 형태로 가공, 전처리 과정에는 데이터의 인코딩 설정, 결측치 처리, 필요한 정보의 추출 및 변환 등

<표> 시공간 매트릭스로 데이터 변환

3) 데이터 결합: 사고 데이터와 기상 데이터를 '지점(공간)'과 '일시(일자별 시간)'를 기준으로 결합(left join), 이 과정에서 특정 지역의 기상 관측 자료가 2개 이상인 경우, 해당 지점의 기상 조건 데이터를 평균 값을 계산

<표> 결합 키 생성, 이종 데이터 결합

다.

분석 활용 전략

결론

- 결합 데이터는 고속도로 터널을 포함한 도로 안전 관련 정책 및 프로그램의 설계에 중요한 영향을 미칠 수 있는 연구의 기초자료로 활용
- 기상 조건을 고려한 사고 예방 및 대응 전략은 잠재적으로 사고 발생률을 줄이고, 인명 피해 및 경제적 손실을 최소화하는 데 기여
- 전국 단위의 기상정보와 교통사고를 결합한 데이터 마트 구축으로 빅데이터 활용센터내 이용자들의 기상정보 데이터 가공시 편의제공 가능

<표> 최종 산출물(결합 데이터)

Appendix

```
import pandas as pd
import os

# 데이터 로드
# 기상 데이터: ASOS 2013년 데이터를 예시로 사용
df = pd.read_csv('./ASOS_2013_2022/ASOS_2013.csv', encoding='cp949')
# 초기 데이터 확인
print(df.head())

# 디렉토리 설정 및 파일 이름 생성
directory = "./ASOS_2013_2022"
```

```
file_names = [f"ASOS_{year}.csv" for year in range(2013, 2023)]

# 모든 CSV 파일을 하나의 DataFrame으로 결합
df_list = []
for file_name in file_names:
    file_path = os.path.join(directory, file_name)
    temp_df = pd.read_csv(file_path, encoding='cp949')
    df_list.append(temp_df)
combined_df = pd.concat(df_list, ignore_index=True)

# 기상 데이터에서 필요한 정보만 추출 및 가공
# 예시로, '기온(°C)'의 평균을 사용하여 피벗 테이블 생성
temp_pivot = combined_df.pivot_table(index='지점', columns='일시', values='기온(°C)', aggfunc='mean')

# Pivot된 DataFrame을 다시 long format으로 변환하여 분석에 용이하게 만듦
melted_df = temp_pivot.reset_index().melt(id_vars=['지점'], var_name='일시', value_name='기온')

# 사고 데이터 로딩 및 초기 가공
ta_df = pd.read_excel("./고속도로 터널_사고시점, 지역-1.xlsx")
# 사고 발생 일시를 '일시' 컬럼에 맞춰 형식 변환
ta_df['일시'] = ta_df['사고일자'].astype(str) + " " + ta_df['사고시간'].astype(str).str[:2] + ":00"
# 열 이름 변경 및 결측치 처리
ta_df.rename(columns={"기상관측지점2": "지점"}, inplace=True)
ta_df.dropna(subset=['지점'], inplace=True)
ta_df['지점'] = ta_df['지점'].astype(int)
# '지점'과 '일시'를 조합하여 결합을 위한 키 생성
ta_df['key'] = ta_df['지점'].astype(str) + " " + ta_df['일시']

# 기온 데이터와 사고 데이터 결합
# 'key'를 기준으로 내부 조인을 사용하여 필요한 정보만을 결합
joined_df = pd.merge(ta_df, melted_df, on="key", how='left')

# 최종 데이터 프레임에서 필요하지 않은 중복 열 제거
final_columns_to_save = joined_df.drop(columns=['지점_x', '지점_y', '일시_x', '일시_y'])
# 최종 데이터를 CSV 파일로 저장
final_columns_to_save.to_csv('final_joined_df.csv', index=False, encoding='cp949')
```

```
# 기상 데이터에서 필요한 정보만 추출 및 가공
# 예시로, '기온(°C)'의 평균을 사용하여 피벗 테이블 생성
temp_pivot = combined_df.pivot_table(index='지점', columns='일시', values='기온(°C)', aggfunc='mean')

# Pivot된 DataFrame을 다시 long format으로 변환하여 분석에 용이하게 만듦
melted_df = temp_pivot.reset_index().melt(id_vars=['지점'], var_name='일시', value_name='기온')

# 사고 데이터 로딩 및 초기 가공
ta_df = pd.read_excel("./고속도로 터널_사고시점, 지역-1.xlsx")
# 사고 발생 일시를 '일시' 컬럼에 맞춰 형식 변환
ta_df['일시'] = ta_df['사고일자'].astype(str) + " " + ta_df['사고시간'].astype(str).str[:2] + ":00"
# 열 이름 변경 및 결측치 처리
ta_df.rename(columns={"기상관측지점2": "지점"}, inplace=True)
ta_df.dropna(subset=['지점'], inplace=True)
ta_df['지점'] = ta_df['지점'].astype(int)
# '지점'과 '일시'를 조합하여 결합을 위한 키 생성
ta_df['key'] = ta_df['지점'].astype(str) + " " + ta_df['일시']

# 기온 데이터와 사고 데이터 결합
# 'key'를 기준으로 내부 조인을 사용하여 필요한 정보만을 결합
joined_df = pd.merge(ta_df, melted_df, on="key", how='left')

# 최종 데이터 프레임에서 필요하지 않은 중복 열 제거
final_columns_to_save = joined_df.drop(columns=['지점_x', '지점_y', '일시_x', '일시_y'])
# 최종 데이터를 CSV 파일로 저장
final_columns_to_save.to_csv('final_joined_df.csv', index=False, encoding='cp949')
```

접수번호 2024-5

교통약자 통행행태의 이질성에 대한 데이터 분석

분석 요청자	소속	기타(교직원)	성명	남OO
	휴대전화	-	전자우편	-
분석유형	■ 데이터 전처리	■ 데이터 시각화	▢ 머신러닝	▢ 기타
데이터 분석가	김건욱 센터장			
분석 기간	2024.06 ~ 2024.07(2개월)			

가.

분석 주제

분석 주제(요청내용)

- 기존 연구에서는 국내 교통약자 통행특성을 설문조사, 특별교통수단 승하차 이력, 교통카드 등의 자료로 시·공간적으로 분석하고 정책적 시사점을 제안한 사례가 많음. 그러나 장애 중증도와 유형별로 통행특성을 비교 분석한 연구는 부족한 실정임
- 국외 연구에서는 장애 중증도와 유형에 따라 통행행태를 분석한 사례가 존재하나, 국내와 국외의 사회·경제적·지리적 여건이 상이하기 때문에 국내 환경에 맞는 연구가 필요함. 또한, 국외 연구들은 대범주 및 특정 장애 중증도에 한정되어 연구가 수행된 한계가 있음
- 이에 따라, 교통약자들이 주요 교통수단으로 활용하는 특별교통수단 승하차 이력 자료를 활용하여 장애 중증도와 유형별로 세분화하여 그룹 간 동질성과 이질성을 가지는 통행특성을 분석하고, 이를 바탕으로 그룹화하여 연구하고자 함
- 해당 데이터 분석은 센터를 방문한 일반시민(교직원)이 요청한 것으로, 개인 연구에 활용될 예정이며, 향후 지자체 교통약자 정책수립에 기초자료로 활용될 가능성이 있음

선행연구 조사

- 교통카드 데이터 활용 연구: Lee et al. (2017)은 서울시 스마트카드 자료를 활용해 대중교통 의존도가 높은 교통약자(노인, 청소년)와 저소득층의 대중교통 이동성 수준과 취약지역을 분석함. 연구 결과, 서울시 15개의 대중교통 우선 개선지역을 도출함. Kwon et al. (2021)은 서울시 교통카드 데이터를 이용해 교통약자의 대중교통 환승통행을 추출하고, 일반인과 교통약자의 환승통행 시간차가 큰 지점을 취약지점으로 정의했으며, 일부 현장조사를 통해 분석 결과를 보정함. 대다수 환승 취약지점의 이동 경로에 교통약자를 위한 이동편의시설이 부족한 것으로 나타났으며, 이는 교통약자 환승통행의 초기 연구로 의의가 있음
- 기타 연구 사례: 특별교통수단 승하차와 대구푸드 식당 정보를 결합한 교통약자 맞춤형 식당 추천(Lee et al., 2021), 특별교통수단 장기대기수요 네트워크 분석(Park et al., 2021), 교통약자 보행자 교통사고 군집분석(Hwang et al., 2022) 등 다양한 연구가 수행되어 정책적 시사점과 아이디어를 제안함
- 국외 연구 사례: 국외 교통약자 연구는 고령자와 장애인을 대상으로 통행 빈도, 수단 선택, 통행 시간, 거리와 장애, 기술 요인을 고려하여 장애 중증도와 일반인을 비교 분석한 사례가 존재함. 최근 리뷰 논문에서는 115개의 연구를 비교 분석하여 장애의 유형을 이동성(Mobility), 인지(Cognitive), 감각(Sensory)으로 구조화해 시사점을 제시함(Park et al., 2022). 연구 결과, 장애인은 일반인에 비해 비업무 통행이 10~30% 낮고, 대중교통 및 택시를 선호하며, 단거리 통행이 주로 이루어진다는 결과를 도출함

- 장애 중증도 및 유형 관련 연구: Schmöcker et al. (2005)은 보행 장애, 청각, 시각, 인지 장애가 비업무 여행 빈도에 부정적인 영향을 미친다고 밝힘. Neven et al. (2018)은 보행 장애가 4.2통행/일에서 1.8통행/일로 감소시키는 것으로 분석함

나. 데이터 분석 및 결과

활용 데이터

- 본 연구에서는 국내 교통약자들의 유형별 통행행태를 분석하기 위해 대구시설공단의 특별교통수단 승하차 이력 자료를 활용함. 자료는 공공데이터 포털에서 수집되었으며, 공간적 범위는 대구광역시, 시간적 범위는 2017년 1월 1일부터 2021년 1월 31일까지 약 4년간의 승하차 자료 4,001,294건을 포함함
- 대구시의 특별교통수단은 "나드리콜"이라는 명칭으로 서비스되고 있음. 이 서비스는 2009년 2월 특장차량 30대를 시작으로 도입되었으며, 현재는 개인택시와 연계하여 총 443대(특장차량 163대, 개인택시 280대)의 차량을 운영 중임. 여기서 특장차량은 휠체어 시설이 부착된 차량을 의미함
- 나드리콜의 이용 대상과 이용 요금은 다음과 같음

구분	세부 내용	비고
주요내용	-보행 장애로 인한 중증 장애인 -공공교통 이용이 어렵다는 진단서를 제출한 3급 이상의 장애인, 국가유공자, 65세 이상 노인	- 기본 요금: 3km 이내 1,000원 - 추가 요금: 3km에서 10km까지 km당 300원, 10km 초과 시 km당 100원 - 요금 한도: 도심 3,300원, 시외 6,600원

데이터 분석

1) 방법론

- 본 연구에서는 제공된 데이터의 통행특성을 세부적으로 분석하기 위해 기존 변수를 가공하여 파생변수를 생성함. 출발지와 도착지의 위도와 경도 좌표를 기반으로 거리를 계산하기 위해 하버사인 공식(Haversine Formula)을 적용하여 이동거리를 추정함
- 하버사인 공식은 유클리드 거리계산 방법에서 지구가 구형임을 감안한 이동거리 계산방법으로, 두 좌표 값을 기반으로 거리를 계산하는 알고리즘임. 또한 도착시간과 출발시간 차이를 계산하여 이동시간을 산출함
- 추가적으로, 특정 조건별 통행특성을 분석하기 위해 출발시간을 기준으로 연, 월, 요일, 일, 시간 등의 시간 속성 변수를 추출하여 파생변수로 생성하고 데이터셋을 구축함

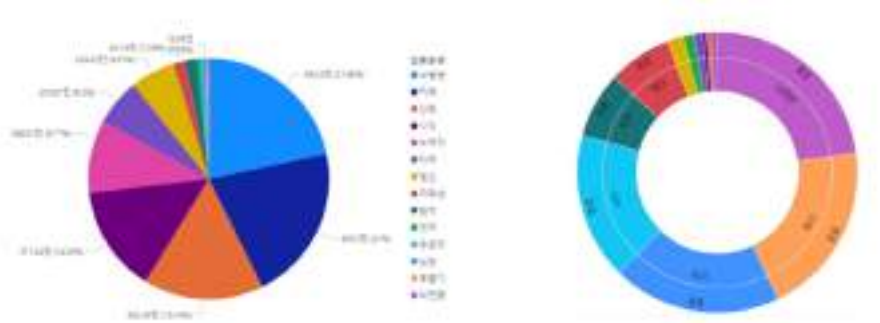
- 데이터의 결측치는 존재하지 않았으며, 이상치는 대구광역시 행정구역을 출발지 또는 도착지로 하지 않은 데이터가 상당수 존재하여 이를 제거함. 이 외에도 이동시간, 이동거리, 배차시간 데이터의 경우 이상치가 존재하여 IQR 기법과 Box-plot 등 시각적 방법을 통해 제거하여 데이터의 잡음을 최소화함
- 특별교통수단 차량을 이용하는 장애유형은 총 17종으로 구성되어 있으며, 전체 데이터의 99%를 설명하는 장애유형 14종을 본 분석에 활용함. 표본 수가 적은 장루·요루, 간, 안면 질환의 경우 데이터를 제외하여 최종적으로 3,979,357건의 데이터를 분석에 활용함

Data preprocessing	Analysis of trip characteristics by type	Cluster analysis
Remove outliers Remove missing values Derived variable creation (trip distance, time) Exploratory analysis Data: Special transportation boarding and alighting data	– Analysis of trip characteristics by disability type – Analysis of trip characteristics by time and day of the week – Analysis of trip characteristics by purpose of trip and disability Descriptive statistical analysis Visualization	– Cluster analysis considering time and spatial trip characteristics – Estimate the optimal number of clusters – Dimension reduction, model comparison Hierarchical cluster analysis K-Means Algorithm

<표> 데이터 분석 방법론

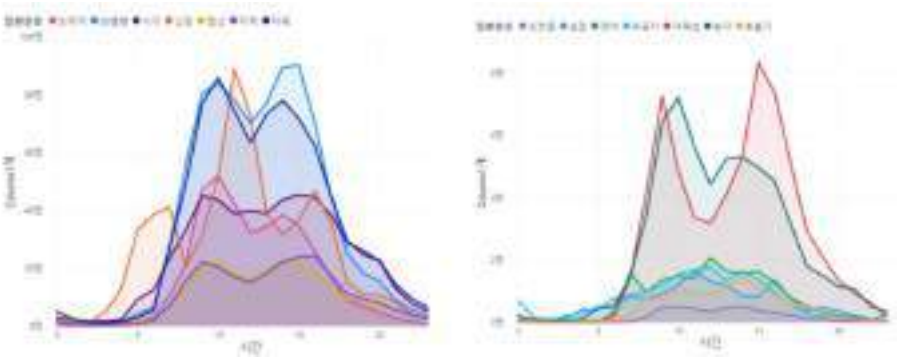
2) 유형별 통행특성 분석

- 장애유형별 통행특성을 살펴보면 뇌병변(22%), 지체(21%), 신장(16%), 시각(14%), 노약자(10%), 지적(6%), 정신(6%) 순으로 나타남. 이는 특별교통수단을 이용하는 주요 이용계층임을 알 수 있음
- 장애 정도는 보건복지부에서 구분하는 기준에 따라 중증과 경증으로 구분했으며, 대부분의 장애유형이 중증에 속하고 일부 이용자만 경증장애에 포함되는 것으로 나타남



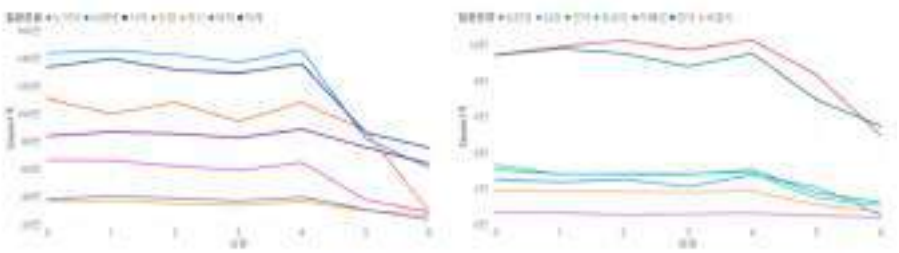
<그림 1> 장애 유형별 교통 분포 (왼쪽), 장애 정도별 교통 특성 분포 (오른쪽)

- 시간대별 통행특성을 살펴보면, 대다수 장애유형에서 오전과 오후 첨두시간에 통행이 집중되어 있음을 알 수 있음. 특히 신장 장애의 경우 오전 12시를 기점으로 통행이 급격히 증가하는데, 이는 대다수 신장장애 유형의 교통약자들이 병원 방문을 목적으로 이용하는 것으로 추정됨
- 이 외에도 뇌전증, 심장, 언어, 유공자, 호흡기 장애의 경우 통행분포가 특정 시간에 집중되지 않으며, 첨도(kurtosis)가 낮아 극단적인 편차나 이상치가 적음을 알 수 있음



<그림 2> 장애 유형별 시간대별 교통 분포

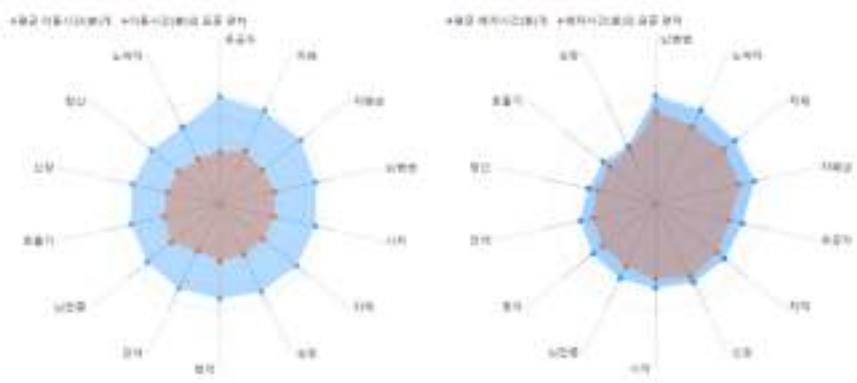
- 요일별 통행특성을 살펴보면, 월요일부터 금요일까지 유사한 패턴을 보이며 토요일과 일요일에 통행이 급격히 감소하는 것으로 나타남
- 장애유형별로 살펴보면 주말 통행에 일부 차이가 존재함. 지적과 정신 장애는 평일과 주말의 차이가 상대적으로 낮고, 신장 장애는 일요일에 급격히 감소하는 것으로 분석됨



<그림> 장애 유형별 요일별 교통 분포

- 장애유형별 통행행태를 세부적으로 살펴보기 위해 출발지에서 도착지까지 이동한 시간과 특별교통수단을 호출하여 배차되기까지의 시간인 배차시간을 분석함
- 평균과 표준편차를 분석지표로 사용하여 장애유형별 통행행태를 시각화한 결과, 평균 이동시간은 유공자, 지체, 자폐성, 뇌병변 순으로 높게 나타남. 특히 지체 장애의 경우 표준편차도 높아 이동시간의 분포가 산재되어 있음을 알 수 있음

– 배차시간의 경우 뇌병변, 노약자, 지체, 자폐성, 유공자 순으로 나타났으며, 평균과 표준편차가 유사한 형태로 시각화됨. 이는 뇌병변, 노약자, 지체 장애의 경우 차량을 배차받기까지 상대적으로 시간이 많이 소요됨을 의미하며, 이는 이용시설물의 시공간적인 특성에 기인한 것으로 판단됨



<그림 3> 평균 이동시간의 평균 및 표준편차(왼쪽), 평균 배차시의 평균 및 표준편차(오른쪽)

다.

분석 활용 전략

결론

– 본 연구는 대구시설공단의 특별교통수단 승하차 이력 데이터를 통해 교통약자들의 통행특성을 세부적으로 분석함. 주요 결론은 다음과 같음
– 주요 이용 계층: 뇌병변(22%), 지체(21%), 신장(16%), 시각(14%) 순으로 이용이 많음
– 장애 정도: 대부분의 장애유형이 중증에 속함
– 시간대별 특성: 오전과 오후 첨두시간에 통행이 집중되며, 신장 장애는 병원 방문으로 인해 오전 12시에 통행이 급증함
– 요일별 특성: 월요일부터 금요일까지 유사한 통행 패턴을 보이며 주말에는 통행이 감소함
– 이동시간과 배차시간: 이동시간은 유공자, 지체, 자폐성, 뇌병변 순으로 길며, 배차시간은 뇌병변, 노약자, 지체 순으로 길게 나타남

정책적 시사점

- 운영 효율성 제고: 중증 장애인과 고령자 대상의 운영 효율성 향상 예상됨
- 시간대별 수요 대응: 첨두시간대 통행수요에 맞춘 차량 배치 가능함
- 주말 통행 지원: 주말에도 안정적인 서비스 제공 가능함
- 시설물 개선: 배차시간이 긴 장애유형에 대한 접근성 개선 가능함
- 본 연구는 교통약자 지원 정책 수립에 기초자료를 제공함으로써, 보다 포괄적이고 효율적인 교통약자 지원 정책 수립에 기여할 수 있음

Appendix

```
df = pd.read_csv('나드리콜_운영데이터.csv', index_col=0)

#초단위 분으로 변경
df['배차시간(분)'] = df['배차시간(초)']/60
df['이동시간(분)'] = df['이동시간(초)']/60

df.drop(['이동시간(초)','배차시간(초)','이동속도(km/h)'], axis=1, inplace=True)

#결측치 확인, 데이터 결측치 없음
print('데이터 결측치', df.isna().sum())

#아웃라이어 시각화 확인
import matplotlib.pyplot as plt
plt.rcParams['font.family'] = 'Malgun Gothic'
plt.subplots(figsize=(18,6))
plt.title("Outliers visualization")
df.boxplot();

#승차지점 위경도 시각화
pickup_longitude = list(df['출발지 경도'])
pickup_latitude = list(df['출발지 위도'])
plt.subplots(figsize=(10,6))
plt.plot(pickup_longitude, pickup_latitude, '.', alpha = 1, markersize = 10)
plt.xlabel('pickup_longitude')
plt.ylabel('pickup_latitude')
plt.show()
```

```
#위경도 이상치 제거
df = df[(df['출발지 경도'] > 128.25)]
df = df[(df['출발지 위도'] < 36.25)]

#승차지점 위경도 시각화
pickup_longitude = list(df['출발지 경도'])
pickup_latitude = list(df['출발지 위도'])
plt.subplots(figsize=(10,6))
plt.plot(pickup_longitude, pickup_latitude, '.', alpha = 1, markersize = 10)
plt.xlabel('pickup_longitude')
plt.ylabel('pickup_latitude')
plt.show()

#하차지점 위경도 시각화
dropoff_longitude = list(df['목적지 경도'])
dropoff_latitude = list(df['목적지 위도'])
plt.subplots(figsize=(10,6))
plt.plot(dropoff_longitude, dropoff_latitude, '.', alpha = 1, markersize = 10)
plt.xlabel('dropoff_longitude')
plt.ylabel('dropoff_latitude')
plt.show()

#위경도 이상치 제거
df = df[(df['목적지 경도'] > 128)]
df = df[(df['목적지 경도'] < 130)]
df = df[(df['목적지 위도'] > 35)]
df = df[(df['목적지 위도'] < 37)]

#하차지점 위경도 시각화
dropoff_longitude = list(df['목적지 경도'])
dropoff_latitude = list(df['목적지 위도'])
plt.subplots(figsize=(10,6))
plt.plot(dropoff_longitude, dropoff_latitude, '.', alpha = 1, markersize = 10)
plt.xlabel('dropoff_longitude')
plt.ylabel('dropoff_latitude')
plt.show()

#위경도 이상치 제거
df = df[(df['목적지 경도'] < 128.9)]
df = df[(df['목적지 위도'] < 36.05)]
```

```
#하차지점 위경도 시각화
dropoff_longitude = list(df['목적지 경도'])
dropoff_latitude = list(df['목적지 위도'])
plt.subplots(figsize=(10,6))
plt.plot(dropoff_longitude, dropoff_latitude, ',', alpha = 1, markersize = 10)
plt.xlabel('dropoff_longitude')
plt.ylabel('dropoff_latitude')
plt.show()

#이동시간 분포 시각화
plt.subplots(figsize=(20,6))
plt.rcParams['font.family'] = 'Malgun Gothic'
plt.hist(df['이동시간(분)'].values, bins=100)
plt.xlabel('이동시간(분)')
plt.ylabel('number of records')
plt.show()

#이동시간 80 이하는 제거
df = df[(df['이동시간(분)'] < 80)]

#이동시간 분포 시각화
plt.subplots(figsize=(20,6))
plt.rcParams['font.family'] = 'Malgun Gothic'
plt.hist(df['이동시간(분)'].values, bins=100)
plt.xlabel('이동시간(분)')
plt.ylabel('number of records')
plt.show()
#배차시간 분포 시각화
plt.subplots(figsize=(20,6))
plt.rcParams['font.family'] = 'Malgun Gothic'
plt.hist(df['배차시간(분)'].values, bins=100)
plt.xlabel('배차시간(분)')
plt.ylabel('number of records')
plt.show()

#배차시간 140 이하는 제거
df = df[(df['배차시간(분)'] < 140)]

#배차시간 분포 시각화
plt.subplots(figsize=(20,6))
plt.rcParams['font.family'] = 'Malgun Gothic'
plt.hist(df['배차시간(분)'].values, bins=100)
plt.xlabel('배차시간(분)')
plt.ylabel('number of records')
plt.show()
```

```
#이동거리 시각화
plt.subplots(figsize=(20,6))
plt.rcParams['font.family'] = 'Malgun Gothic'
plt.hist(df['이동거리(km)'].values, bins=100)
plt.xlabel('이동거리(km)')
plt.ylabel('number of records')
plt.show()

#이동거리 30 이하는 제거
df = df[(df['이동거리(km)'] < 30)]

#이동거리 시각화
plt.subplots(figsize=(20,6))
plt.rcParams['font.family'] = 'Malgun Gothic'
plt.hist(df['이동거리(km)'].values, bins=100)
plt.xlabel('이동거리(km)')
plt.ylabel('number of records')
plt.show()

#아웃라이어 시각화 확인
import matplotlib.pyplot as plt
plt.rcParams['font.family'] = 'Malgun Gothic'
plt.subplots(figsize=(18,6))
plt.title("Outliers visualization")
df.boxplot();

#주요질환 분류
df2 = df[(df['질환분류']=='뇌병변') | (df['질환분류']=='지체')
| (df['질환분류']=='신장') | (df['질환분류']=='시각')
| (df['질환분류']=='노약자') | (df['질환분류']=='지적')
| (df['질환분류']=='정신') | (df['질환분류']=='자폐성')
| (df['질환분류']=='청각') | (df['질환분류']=='언어')
| (df['질환분류']=='유공자') | (df['질환분류']=='심장')
| (df['질환분류']=='호흡기') | (df['질환분류']=='뇌전증')]

df2['고객 승차시간'] = pd.to_datetime(df2['고객 승차시간'])

#시간타임 추출(re)-승차시간 기준
df2['년'] = df2['고객 승차시간'].dt.year
df2['월'] = df2['고객 승차시간'].dt.month
df2['요일'] = df2['고객 승차시간'].dt.weekday
df2['시간'] = df2['고객 승차시간'].dt.hour

df2.drop(['접수시간','고객 승차시간','고객 하차시간','배차구분'], axis=1, inplace=True)
```

```
#탐색적 분석
import seaborn as sns
a,b = plt.subplots(1,1, figsize=(10,5))
sns.boxplot(df2['질환분류'], df2['이동시간(분)'])

df3 = df2[['질환분류','출발지 행정동','목적지 행정동','년','월','일','요일','시간','배차시간(분)','이동시간(분)','목적지분류','배차 차량종류']]

from sklearn.preprocessing import LabelEncoder
label = ['질환분류','출발지 행정동', '목적지 행정동', '목적지분류', '배차 차량종류']
df3[label] = df3[label].apply(LabelEncoder().fit_transform)

#카테고리 데이터 타입변경
dtype=['질환분류','출발지 행정동','목적지 행정동', '목적지분류','배차 차량종류']
for i in df3[dtype]:
    df3[i] = df3[i].astype('category')
df3.dtypes

from sklearn.preprocessing import MinMaxScaler
minmax = ['배차시간(분)','이동시간(분)']
scaler = MinMaxScaler()
df3[minmax] = scaler.fit_transform(df3[minmax])
df3.head()

#필요없는 열 제외, 독립변수와 종속변수 분리

df3_y = df3['질환분류']
drop = ['질환분류']
df3_x = df3.drop(drop, axis=1)

#2차원으로 축소하여 시각화하여 이질성과 동질성을 판단

from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(df3)
df3_pca = pca.transform(df3)

# 시각적으로 차이가 없어보임
plt.scatter(df3_pca[:,0], df3_pca[:,1], c=df3_y, linewidths=1, edgecolors='green')
plt.show()

#군집분석으로 유사한 특성을 가지는 데이터를 그룹핑하기!

### 여러개의 클러스터링 개수를 List로 입력 받아 각각의 실루엣 계수를 면적으로 시각화한 함수 작성
```

```
def visualize_silhouette(cluster_lists, X_features):

    from sklearn.datasets import make_blobs
    from sklearn.cluster import KMeans
    from sklearn.metrics import silhouette_samples, silhouette_score

    import matplotlib.pyplot as plt
    import matplotlib.cm as cm
    import math

    # 입력값으로 클러스터링 개수들을 리스트로 받아서, 각 갯수별로 클러스터링을 적용하고 실루엣 개수를 구함
    n_cols = len(cluster_lists)

    # plt.subplots()으로 리스트에 기재된 클러스터링 만큼의 sub figures를 가지는 axs 생성
    fig, axs = plt.subplots(figsize=(4*n_cols, 4), nrows=1, ncols=n_cols)

    # 리스트에 기재된 클러스터링 개수들을 차례로 iteration 수행하면서 실루엣 개수 시각화
    for ind, n_cluster in enumerate(cluster_lists):

        # KMeans 클러스터링 수행하고, 실루엣 스코어와 개별 데이터의 실루엣 값 계산.
        clusterer = KMeans(n_clusters = n_cluster, max_iter=500, random_state=0)
        cluster_labels = clusterer.fit_predict(X_features)

        sil_avg = silhouette_score(X_features, cluster_labels)
        sil_values = silhouette_samples(X_features, cluster_labels)

        y_lower = 10
        axs[ind].set_title('Number of Cluster : '+ str(n_cluster)+'\n' \
                          'Silhouette Score :' + str(round(sil_avg,3)) )
        axs[ind].set_xlabel("The silhouette coefficient values")
        axs[ind].set_ylabel("Cluster label")
        axs[ind].set_xlim([-0.1, 1])
        axs[ind].set_ylim([0, len(X_features) + (n_cluster + 1) * 10])
        axs[ind].set_yticks([])  # Clear the yaxis labels / ticks
        axs[ind].set_xticks([0, 0.2, 0.4, 0.6, 0.8, 1])

        # 클러스터링 개수별로 fill_betweenx( )형태의 막대 그래프 표현.
        for i in range(n_cluster):
            ith_cluster_sil_values = sil_values[cluster_labels==i]
            ith_cluster_sil_values.sort()

            size_cluster_i = ith_cluster_sil_values.shape[0]
            y_upper = y_lower + size_cluster_i
```

```
    color = cm.nipy_spectral(float(i) / n_cluster)
        axs[ind].fill_betweenx(np.arange(y_lower, y_upper), 0, ith_cluster_sil_values, \
                        facecolor=color, edgecolor=color, alpha=0.7)
        axs[ind].text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))
        y_lower = y_upper + 10

    axs[ind].axvline(x=sil_avg, color="red", linestyle="--")

### 여러개의 클러스터링 개수를 List로 입력 받아 각각의 클러스터링 결과를 시각화 함수
def visualize_kmeans_plot_multi(cluster_lists, X_features):

    from sklearn.cluster import KMeans
    from sklearn.decomposition import PCA
    import pandas as pd
    import numpy as np

    # plt.subplots()으로 리스트에 기재된 클러스터링 만큼의 sub figures를 가지는 axs 생성
    n_cols = len(cluster_lists)
    fig, axs = plt.subplots(figsize=(4*n_cols, 4), nrows=1, ncols=n_cols)

    # 입력 데이터의 FEATURE가 여러개일 경우 2차원 데이터 시각화가 어려우므로 PCA 변환하여 2
차원 시각화
    pca = PCA(n_components=2)
    pca_transformed = pca.fit_transform(X_features)
    dataframe = pd.DataFrame(pca_transformed, columns=['PCA1','PCA2'])

     # 리스트에 기재된 클러스터링 개수들을 차례로 iteration 수행하면서 KMeans 클러스터링 수행
하고 시각화
    for ind, n_cluster in enumerate(cluster_lists):

        # KMeans 클러스터링으로 클러스터링 결과를 dataframe에 저장.
        clusterer = KMeans(n_clusters = n_cluster, max_iter=500, random_state=0)
        cluster_labels = clusterer.fit_predict(pca_transformed)
        dataframe['cluster']=cluster_labels

        unique_labels = np.unique(clusterer.labels_)
        markers=['o', 's', '^', 'x', '*']

        # 클러스터링 결과값 별로 scatter plot 으로 시각화
        for label in unique_labels:
            label_df = dataframe[dataframe['cluster']==label]
            if label == -1:
                cluster_legend = 'Noise'
```

```
    else :
            cluster_legend = 'Cluster '+str(label)
        axs[ind].scatter(x=label_df['PCA1'], y=label_df['PCA2'], s=70,\
                edgecolor='k', marker=markers[label], label=cluster_legend)

    axs[ind].set_title('Number of Cluster : '+ str(n_cluster))
    axs[ind].legend(loc='upper left')

  plt.show()

# 전처리 및 머신러닝 모형학습은 Python, 시각적 분석은 PowerBI를 활용하여 시각화 수행
```

접수번호 2024-6

공공기관 헬프데스크 운영 효율성 개선을 위한 요청사항 시계열 분석

분석 요청자	소속	기관	성명	전○○
	휴대전화	-	전자우편	-
분석유형	■ 데이터 전처리	■ 데이터 시각화	▢ 머신러닝	▢ 기타
데이터 분석가	김건욱 센터장, 김현정 사무원			
분석 기간	2024.07 ~ 2024.07(1개월)			

가.

분석 주제

분석 주제(요청내용)

- 헬프데스크를 통해 직원들의 다양한 요구사항을 처리하고 있으며 시스템 및 시설의 문제를 해결하고 지원하고 있음. 헬프데스크는 직원들의 업무 효율성을 높이고 문제 해결을 신속하게 지원하는데 중요한 역할을 함
- 그러나 빈번하고 복잡한 요청 사항으로 인해 헬프데스크의 운영이 과부하되는 경우가 많음. 이러한 현상은 업무의 지연이 발생할 가능성이 있음
- 이에 따라, 헬프데스크에 접수된 요청 사항을 체계적으로 분석하는 것이 필요하며, 효과적인 대응 전략을 마련하고 업무 효율성을 제고하고자 함
- 헬프데스크에 접수된 요청 사항을 분석하기 위해 부서별, 직급별, 시기별, 개인별로요청 사항을 세분화하고, 각 부서와 직급에서 발생하는 주요 요구 사항과 특정 시기에 집중되는 요청의 경향을 파악하기 위해 시계열 분석 기법을 이용한 데이터 분석을 요청함

분석 필요성 및 전략

- 분석 요청자에게 제공 받은 데이터는 약 3년 동안 접수된 헬프데스크 요청사항 의 수와 특성을 일별로 기록한 데이터임
- 시계열 데이터 분석 기법을 사용하여 시간의 흐름에 따라 기록된 데이터를 분석하고 주요 패턴, 트렌드 및 변동성을 파악하는 것이 필요함
- 이를 통해 시간에 따른 변화 양상을 이해할 수 있으며, 효과적인 대응 방안을 마련하는 데 도움이 될 것임
- 각 분류 레벨에서 주요 패턴을 파악하여 효율적인 자원 배분과 문제 해결을 위한 기초 자료로 활용할 수 있음
- 본 분석에서는 특정 카테고리(LEVEL, 직책, 신청자 id)의 레이블 수가 많은 경우, 시각화 결과의 직관적인 이해를 돕고 가독성을 높이기 위해 상위 n개의 레이블만 출력하는 방법을 선택함

나.

데이터 분석 및 결과

활용 데이터

1) 개요

- 본 분석에서 활용된 데이터는 아래와 같이 구성됨
- 신청자, 직급, 분류(대, 중, 소), 신청 시기, 신청 내용 등이 담겨있으며, 신청 내용의 경우 정형화 되어있지 않은 텍스트 데이터로 구성됨

데이터명	내용	시간 범위	데이터 규모
헬프데스크 지원요청 내역	신청자, 직급, 분류(대, 중, 소), 신청 시기, 신청 내용 등	2021년 8월 ~2024년 6월 (약 3년)	약 28,900건

<표> 활용 데이터

2) 가명 처리

- 데이터 분석에 앞서 데이터 제공자의 요청으로 대구가명정보활용지원센터에서 데이터 가명처리를 진행함
- 사원번호와 사원이름 정보를 가진'REQSTR_ID'와 'REQSTR_NM'컬럼에 대하여 가운데 모두'*'로 마스킹 처리함

컬럼명	개인정보형태	가명 처리 전	가명 처리 후
'REQSTR_ID'	사원번호	'12345'	'1***5'
'REQSTR_NM'	사원이름	'홍길동'	'홍*동'

<표> 가명 처리 과정(예시)

데이터 분석(전처리)

- '호칭'과'PROCESS_CN'컬럼에 결측치가 존재하여 아래와 같이 처리함
- '호칭': 사원 데이터가 있는 경우 결합 가능하므로 공백으로 대체
- PROCESS_CN': 담당자 답변내용이므로 본 분석에는 필요하지 않으므로 공백으로 대체

```
# 데이터 로드
df = pd.read_csv('helpdesk_info_main_가명(utf8).csv')

# 결측치 확인 결과 '조정', 'PROCESS_CN' 칼럼에 결측치가 많이 존재함.
df.isna().sum()
```

✓ 0.2s

```
REQSTR_ID          0
REQSTR_NM          0
REQST_DT           0
LEVEL1             0
LEVEL2             0
LEVEL3             0
TITLE              0
조정               167
PROCESS_CN        1648
REQST_CN             9
dtype: int64
```

```
# 분석에 미치는 영향을 최소화하기 위해 공란 nan 값을 '' 로 대체
df = df.fillna('')

# 결측값이 잘 처리되었는지 확인
pd.DataFrame(df.isna().sum()).T  # 각 칼럼의 결측값 개수를 세고, 이를 데이터프레임으로 변환하여 출력
```

✓ 0.0s

	REQST_DT	LEVEL1	LEVEL2	LEVEL3	TITLE	REQSTR_ID	REQSTR_NM	조정	PROCESS_CN	REQST_CN
0	0	0	0	0	0	0	0	0	0	0

<표> 결측치 확인 및 처리

– 'REQST_DT'컬럼을 datetime형식으로 변환하여 시계열 기반의 분석이 가능하도록 처리

```
# 'REQST_DT' 칼럼을 datetime 형식으로 변환
df['REQST_DT'] = pd.to_datetime(df['REQST_DT'])
df['REQST_DT']
```

✓ 0.0s

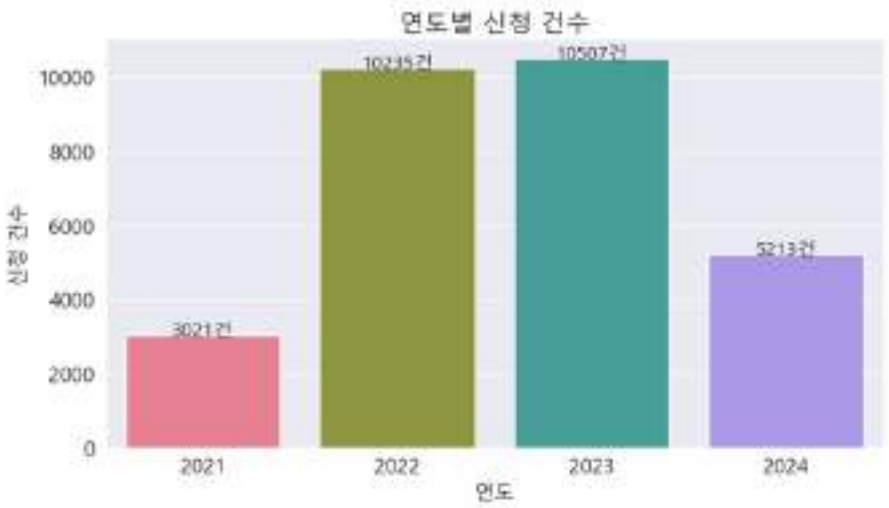
```
0       2022-01-07 12:00:00
1       2022-10-06 12:00:00
2       2023-10-06 12:00:00
3       2022-10-05 12:00:00
4       2024-03-13 12:00:00
                ...
28971   2022-10-05 12:00:00
28972   2023-10-11 12:00:00
28973   2022-08-22 12:00:00
28974   2024-01-26 12:00:00
28975   2022-03-11 12:00:00
Name: REQST_DT, Length: 28976, dtype: datetime64[ns]
```

<표> datetime형식으로 변환

데이터 분석(시각화)

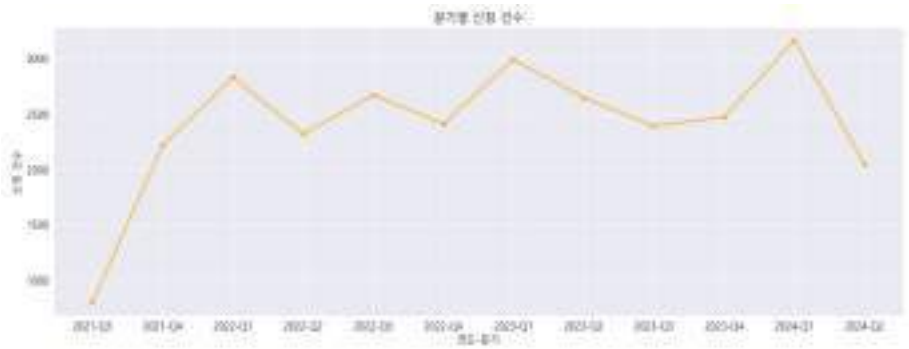
1) 시기별 현황

- 데이터가 2021년 8월부터 2024년 6월까지 집계되어 있어, 정확한 12개월의 월별 추이를 파악하기 어려움. 따라서 연도별, 분기별, 요일별로 신청 건수를 집계하여 시각화를 진행함
- 연도별 신청 건수 분포를 살펴보면 2021년(3,021건), 2022년(10,235건), 2023년(10507건), 2024년(5,213건)으로 집계됨
- 2021년 데이터는 8월부터 집계되었으므로 연간 총 건수가 적을 수 있음
- 2024년은 아직 진행 중인 연도로, 데이터가 완전하지 않음에도 불구하고 2021년 하반기 신청 건수를 이미 초과함. 이는 신청 건수가 지속적으로 증가하고 있음을 나타냄



<그림 1> 연도별 신청 건수 분포

- 분기별 신청 건수는 아래 그림과 같이 전반적으로 1분기(Q1)에 신청 건수가 높게 나타난 후 점차 감소하는 패턴을 보임. 특히 2022년과 2024년의 1분기에는 신청 건수가 두드러지게 높은 모습을 보였으며, 이는 연초에 요청이 집중되는 경향이 있는 것으로 추정됨



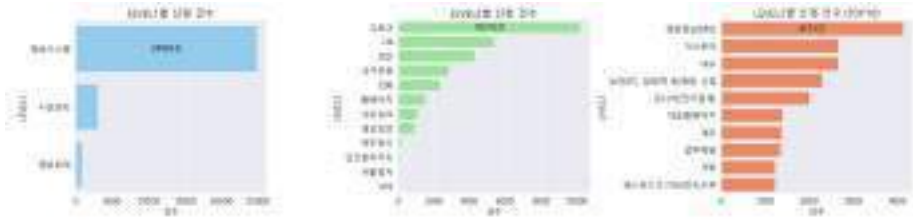
<그림 2> 분기별 신청 건수 분포

– 요일별 신청 건수는 아래 그림과 같이 월요일부터 금요일까지 상대적으로 높은 신청 건수가 유지되다가 토요일과 일요일에 신청 건수가 급격히 줄어드는 패턴이 모든 연도에서 일관되게 나타남. 이는 대다수의 신청자들이 헬프데스크 시스템을 평일에 이용하는 것을 의미함

<그림 3> 요일별 신청 건수 분포

2) LEVEL별 현황

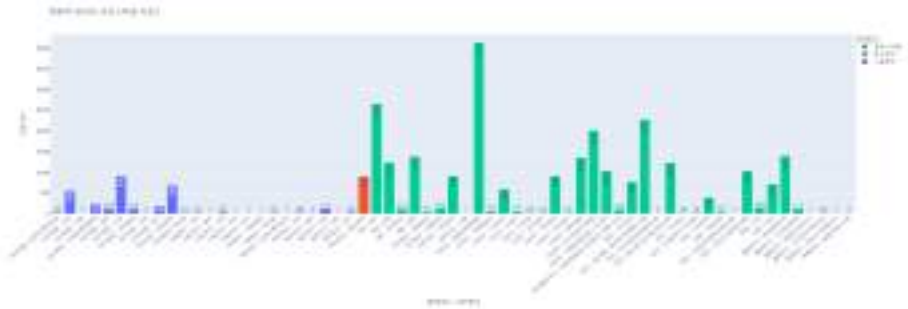
– 헬프데스크에서 처리하는 요구사항은 계층적 분류 체계를 통해 LEVEL1(대분류), LEVEL2(중분류), LEVEL3(소분류)로 구분되어 체계적으로 분류됨
– 각 레벨별로 신청 건수를 집계하고 많은 신청 건수를 기록한 카테고리를 아래 그림과 같이 시각화함
– 각 레벨별 최대 신청 건수를 기록한 카테고리는 LEVEL1은 '정보시스템'(24,986건), LEVEL2는 '다모아'(10,216건), LEVEL3는 '경영정보(MIS)'(4,131건)임



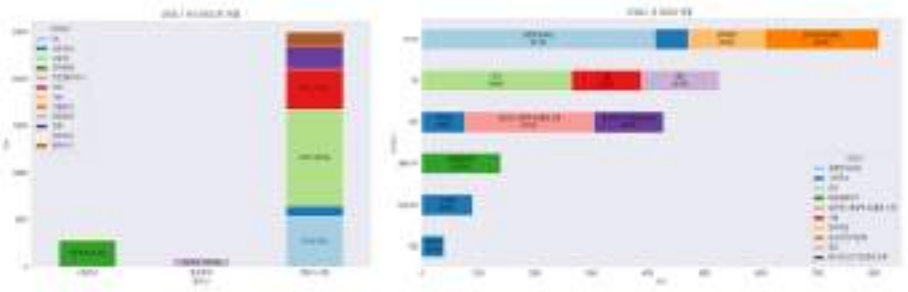
<그림 4> LEVEL별 신청 건수 분포

– 각 레벨의 계층적 구조를 한눈에 파악하기 위해 각 레벨별 신청 건수를 집계하여 아래 그림과 같이 계층적 막대 차트로 시각화함
– 또한, 하위 레벨별 비율 분포를 한눈에 파악하기 위해 데이터의 패턴, 상관관계, 유사성 등을 빠르게 탐색하고 이해할 수 있도록 도와주는 Stacked Bar Plot을 활용하여 시각화함
– 대부분의 하위 레벨(LEVEL2, LEVEL3)의 카테고리가 LEVEL1의 '정보시스템' 카테고리에 속함

- 다모아-경영정보(MIS)', 'OA-서울', '보안-개인 PC' 순으로 신청 건수가 많았음
- 특히 ,'영상회의(LEVEL1)' 카테고리에서는 '영상회의(LEVEL2)' 관련 요청만이 존재하며 다른 중분류는 존재하지 않음. 영상회의 카테고리가 단일한 중분류로만 구성되어 있음을 알 수 있음

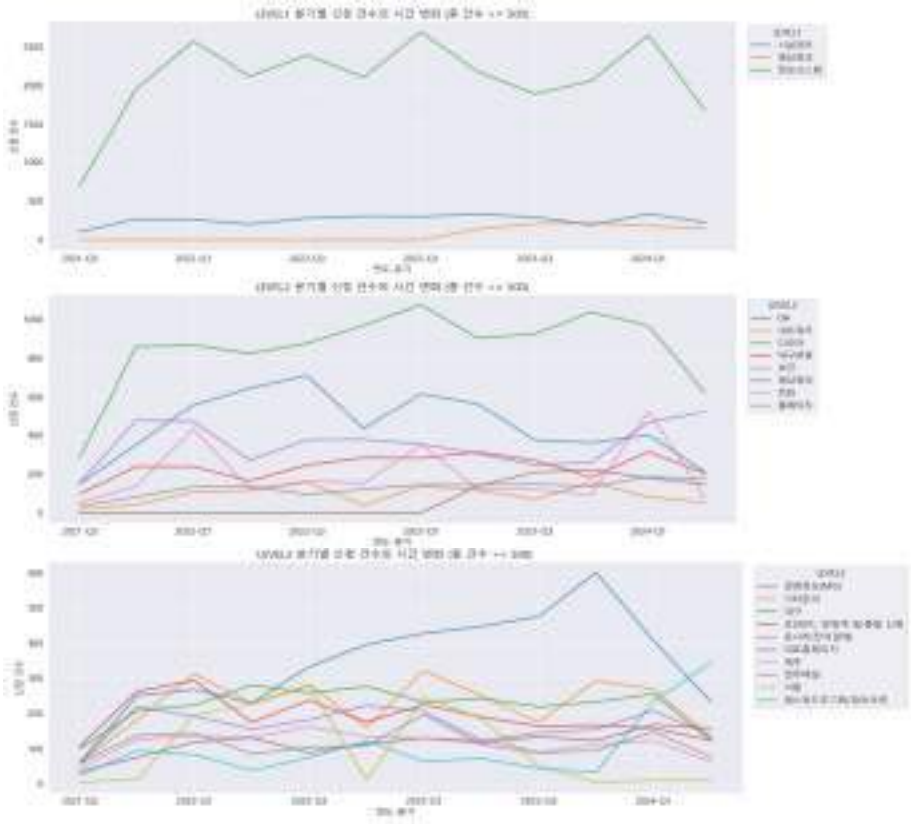


<그림 5> LEVEL별 계층적 막대 차트



<그림 6> 하위 LEVEL별 비율 분포

- 각 레벨별 시계열 변화를 분석하기 위해 레벨별로 분기별 신청 건수의 변화를 아래 그림과 같이 시계열 그래프로 나타냄
- LEVEL1의 경우, '정보시스템' 카테고리가 분석기간 동안 일관되게 높은 신청 건수를 기록하였으며, 특히 1분기마다 피크를 보임. 이는 매년 1분기에 정보시스템 관련 문제가 발생하는 것으로 추정할 수 있음
- LEVEL2의 경우, '다모아' 카테고리가 지속적으로 높은 건수를 기록하였으며, 2023년 1분기에 피크를 보임. 또한 'OA' 카테고리는 2022년 1분기에 큰 증가를 보였으나 이후 감소하는 경향을 보임. 이는 특정 시기에 OA 관련 문제가 집중되었음을 의미함
- LEVEL3의 경우, '경영정보(MIS)' 카테고리가 2022년 1분기와 2023년 4분기에 피크를 보이며, 다모아 내에서 경영정보 관련 요청이 빈번히 발생함을 나타냄



<그림 7> LEVEL별 신청 건수 시계열 분포

3) 직책별 현황

- 각 레벨의 계층적 구조를 한눈에 파악하기 위해 Stacked Bar Plot을 활용하여 시각화함
- 시각화 결과의 직관적인 이해를 돕고 가독성을 높이기 위하여 상위 직책 10개만 시각화함
- 직책별 신청 건수는 선임, 수석, 책임, 주임, 연구원 순으로 높게 나타남
- 또한 LEVEL1의 경우, 대부분의 직책에서 '정보시스템' 관련 요청이 집중되는 경향을 보이며, 이는 '정보시스템'이 모든 직책에서 필수적이고 중요한 역할임과 동시에 해당 직책들이 '정보시스템'의 원활한 운영을 위해 많은 지원이 필요함을 의미함
- LEVEL2의 경우, '다모아'와 'OA' 관련 요청이 빈번하게 발생하고 있으며, '보안'과 '네트워크' 관련 요청도 선임 및 수석 직책에서 높은 비율을 차지하고 있어, 해당 시스템들에 대한 지원 강화가 필요함을 의미함

- LEVEL3의 경우, 대부분의 직책에서 경영정보시스템(MIS) 관련 요청이 빈번하게 발생하고 있으며, 이들 직책에서 경영정보 시스템의 중요성을 나타냄
- '대구', '서울' 관련 요청도 선임, 수석, 책임 직책에서 높은 비율을 차지하고 있어, 해당 지역의 시스템과 관련된 지원 강화가 필요함을 의미함

다.

분석 활용 전략

결론

- 본 연구는 헬프데스크에 접수된 요구사항을 체계적으로 분석하기 위해 시계열 분석 기법을 적용하였음. 주요 결론은 다음과 같음
- 주요 요청 카테고리 : 정보시스템(24,986건), 다모아(10,216건), 경영정보(MIS)(4,131건) 등이 주요 요청 카테고리로 나타남. 이는 헬프데스크가 주로 정보시스템 관련 문제를 처리하고 있음을 보여줌
- 특정 시기 집중 요청 : 분기별 분석 결과, 특히 1분기에 요청 건수가 집중되는 경향이 확인됨. 이는 연초에 시스템 사용이 증가하면서 발생하는 문제들이 많은 것으로 추정됨
- 직책별 요청 패턴 : 선임, 수석, 직원 등 직책별로 정보시스템 관련 요청이 많았으며, 각 직책의 요청 패턴을 통해 직무별 필요 지원 사항을 파악할 수 있었음

정책적 시사점

- 운영 효율성 제고 : 정보시스템 관련 요청이 다수를 차지하는 만큼 해당 분야에 대한 인력과 자원을 집중 배치하여 운영 효율성을 높일 필요가 있음
- 시간대별 대응 전략 : 1분기에 요청이 집중되는 경향을 고려하여, 연초에 대비한 자원 배분과 지원 대기 시간 단축 방안을 마련해야 함
- 데이터 기반 정책 수립 : 본 분석에서 도출된 데이터를 기반으로 헬프데스크 운영 및 지원 정책을 수립함으로써 민원 처리의 효율성을 제고하고, 사용자 만족도를 높일 수 있을 것으로 기대됨

Appendix

```
# 라이브러리 불러오기
import pandas as pd
import numpy as np

# 데이터 시각화 관련 라이브러리
import seaborn as sns
import matplotlib.pyplot as pl
import plotly.graph_objects as go
import plotly.express as px
import matplotlib as mpl
import matplotlib.font_manager as fm

# 데이터 로딩 및 전처리
df = pd.read_csv('helpdesk_info_main_가명(utf8).csv')

# 결측치 확인 결과 '호칭', 'PROCESS_CN' 컬럼에 결측치가 많이 존재함.
df.isna().sum()

# 분석에 미치는 영향을 최소화하기 위해 모든 nan 값을 ''로 대체
df = df.fillna('')

# 각 컬럼의 결측값 개수를 세고, 이를 데이터프레임으로 변환하여 출력
pd.DataFrame(df.isna().sum()).T

# 'REQST_DT' 컬럼을 datetime 형식으로 변환
df['REQST_DT'] = pd.to_datetime(df['REQST_DT'])


# 연도별 신청 건수 시각화
# 'REQST_DT' 열에서 연도 추출하여 '연도' 열 생성
df['연도'] = df['REQST_DT'].dt.year

# 연도별 민원 건수 시각화를 위한 Figure 설정 (크기 지정)
plt.figure(figsize=(8, 5))

# sns.countplot을 사용하여 연도별 민원 건수 시각화
sns.countplot(x='연도', data=df, palette='husl')

# 각 데이터 포인트에 라벨 추가
# 연도별 민원 건수를 계산하고 정렬
yearly_counts = df['연도'].value_counts().sort_index()
```

```
# 각 연도에 해당하는 건수 값을 그래프에 텍스트로 표시
for i in range(len(yearly_counts)):
        plt.text(i, yearly_counts.values[i] + 3, f'{yearly_counts.values[i]}건', ha='center',
fontsize=12)


# 그래프의 제목 및 축 라벨 설정
plt.title('연도별 신청 건수')
plt.xlabel('연도')
plt.ylabel('신청 건수')

# 그래프 레이아웃 조정 및 출력
plt.tight_layout()
plt.show()

# 분기별 신청 건수 시각화
# 'REQST_DT' 열에서 '연도-분기' 형식으로 변환하여 '연도-분기' 열 생성
df['연도-분기'] = df['REQST_DT'].dt.to_period('Q').astype(str).str.replace('Q', '-Q')

# 연도별 민원 건수를 계산하고 정렬
quarterly_counts = df['연도-분기'].value_counts().sort_index()

# 시각화 설정: 그림 크기 지정
plt.figure(figsize=(20, 14))

# 시각화1 : 연도-분기별 신청 건수 라인 그래프
plt.subplot(2, 1, 1)  # 2×1 레이아웃에서 첫 번째 서브플롯

# 연도-분기별 신청 건수를 라인 그래프로 시각화
plt.plot(quarterly_counts.index.astype(str), quarterly_counts, marker='o', color="orange",
alpha=0.6, linewidth=3)

# 그래프의 제목 및 축 라벨 설정
plt.title('분기별 신청 건수')
plt.xlabel('연도-분기')
plt.ylabel('신청 건수')

# 요일별 신청 건수 시각화
# 요일을 한글로 표시하기 위해 요일 이름 리스트 생성
weekdays = ['월요일', '화요일', '수요일', '목요일', '금요일', '토요일', '일요일']

# 'REQST_DT' 열에서 요일을 한글로 추출하여 '요일' 열 생성
df['요일'] = df['REQST_DT'].dt.day_name(locale='ko_KR')
```

```
# 요일별 전체 신청 건수 계산
total_weekly_counts = df['요일'].value_counts().reindex(weekdays, fill_value=0)

# 시각화 설정: 그림 크기 지정
plt.figure(figsize=(12, 6))

# sns.countplot을 사용하여 요일별 신청 건수를 연도별로 시각화
sns.countplot(data=df, x='요일', hue='연도', order=weekdays, palette='viridis')



# 그래프 제목 및 축 라벨 설정
plt.title('연도별 요일별 신청 건수')
plt.xlabel('요일')
plt.ylabel('신청 건수')
plt.legend(title='연도')

# 그래프 레이아웃 조정 및 출력
plt.tight_layout()
plt.show()

# LEVEL별 신청 건수 시각화
# 'LEVEL1'별 신청 건수를 계산
level1 = df['LEVEL1'].value_counts()

# 'LEVEL2'별 신청 건수를 계산
level2 = df['LEVEL2'].value_counts()

# 'LEVEL3'별 신청 건수를 계산
level3 = df['LEVEL3'].value_counts()

# 'LEVEL3'의 레이블 수가 많으므로 상위 10개만 선택
level3_top10 = level3.head(10)

# 시각화 설정: 1×3 레이아웃의 서브플롯 생성
fig, axs = plt.subplots(1, 3, figsize=(20, 5))

# LEVEL1별 신청 건수 시각화
sns.barplot(x=level1.values, y=level1.index, color='lightskyblue', ax=axs[0])
axs[0].set_title('LEVEL1별 신청 건수')
axs[0].set_xlabel('건수')

# LEVEL1의 최대값 텍스트 표시
max_index = level1.idxmax()  # 최대값의 인덱스 찾기
```

```
max_value = level1.max()  # 최대값 찾기
axs[0].text(max_value / 2, level1.index.get_loc(max_index), f'{max_value}건',
        va='center', ha='center', color='black')  # 텍스트 표시

# LEVEL2별 신청 건수 시각화
sns.barplot(x=level2.values, y=level2.index, color='lightgreen', ax=axs[1])
axs[1].set_title('LEVEL2별 신청 건수')
axs[1].set_xlabel('건수')

# LEVEL2의 최대값 텍스트 표시
max_index = level2.idxmax()
max_value = level2.max()
axs[1].text(max_value / 2, level2.index.get_loc(max_index), f'{max_value}건',
        va='center', ha='center', color='black')


# LEVEL3별 신청 건수 (상위 10개) 시각화
sns.barplot(x=level3_top10.values, y=level3_top10.index, color='coral', ax=axs[2])
axs[2].set_title('LEVEL3별 신청 건수 (TOP10)'
axs[2].set_xlabel('건수')

# LEVEL3 (TOP10)의 최대값 텍스트 표시
max_index = level3_top10.idxmax()
max_value = level3_top10.max()
axs[2].text(max_value / 2, level3_top10.index.get_loc(max_index), f'{max_value}건',
        va='center', ha='center', color='black')

# 레이아웃 조정 및 출력
plt.tight_layout()
plt.show()


# 계층적 막대 차트 시각화
# LEVEL1, LEVEL2, LEVEL3별 신청 건수 집계
df_grouped = df.groupby(['LEVEL1', 'LEVEL2', 'LEVEL3']).size().reset_index(name='Count')

fig = go.Figure()

# 색상 팔레트 설정
colors = px.colors.qualitative.Plotly

for i, level1 in enumerate(df_grouped['LEVEL1'].unique()):
    df_level1 = df_grouped[df_grouped['LEVEL1'] == level1]
    fig.add_trace(go.Bar(
        x=df_level1['LEVEL2'] + ' - ' + df_level1['LEVEL3'],
```

```
        y=df_level1['Count'],
        name=level1,
        marker_color=colors[i % len(colors)],
        text=df_level1['Count'],
        textposition='auto'
    ))

fig.update_layout(
    barmode='stack',
    title='계층적 데이터 구조 (막대 차트)',
    xaxis_title='LEVEL2 - LEVEL3',
    yaxis_title='신청 건수',
    font=dict(size=12),
    width=2000,
    height=800,
    xaxis=dict(tickangle=-45),
    legend=dict(title='LEVEL1', x=1, y=1)

fig.show()


# LEVEL별 신청건수 시계열 분포 시각화
# LEVEL1 신청 건수 집계
level1_quarterly_counts = df.groupby(['연도-분기', 'LEVEL1']).size().unstack().fillna(0)

# LEVEL2 신청 건수 집계
level2_quarterly_counts = df.groupby(['연도-분기', 'LEVEL2']).size().unstack().fillna(0)

# LEVEL3 신청 건수 집계
level3_quarterly_counts = df.groupby(['연도-분기', 'LEVEL3']).size().unstack().fillna(0)

# LEVEL3 카테고리 선정 (상위 10개 카테고리 선택)
top_n = 10
top_level3_categories = level3_quarterly_counts.sum().nlargest(top_n).index
top_level3_quarterly_counts = level3_quarterly_counts[top_level3_categories]

# 시각화 결과의 가독성을 위해 일정 기간 동안의 총 신청 건수가 500 이상인 경우만 선택
threshold_value = 500
level1_quarterly_counts_filtered = level1_quarterly_counts.loc[:, level1_quarterly_counts.
sum() >= threshold_value]
level2_quarterly_counts_filtered = level2_quarterly_counts.loc[:, level2_quarterly_counts.
sum() >= threshold_value]
top_level3_quarterly_counts_filtered = top_level3_quarterly_counts.loc[:, top_level3_
quarterly_counts.sum() >= threshold_value]
```

```
# 시각화
fig, ax = plt.subplots(3, 1, figsize=(18, 18))

# LEVEL1 시각화
level1_quarterly_counts_filtered.plot(ax=ax[0], alpha=0.6, linewidth=3)
ax[0].set_title('LEVEL1 분기별 신청 건수의 시간 변화 (총 건수 >= 500)')
ax[0].set_ylabel('신청 건수')
ax[0].set_xlabel('연도-분기')
ax[0].legend(title='LEVEL1', bbox_to_anchor=(1.01, 1), loc='upper left')

# LEVEL2 시각화
level2_quarterly_counts_filtered.plot(ax=ax[1], alpha=0.6, linewidth=3)
ax[1].set_title('LEVEL2 분기별 신청 건수의 시간 변화 (총 건수 >= 500)')
ax[1].set_xlabel('연도-분기')
ax[1].set_ylabel('신청 건수')
ax[1].legend(title='LEVEL2', bbox_to_anchor=(1.01, 1), loc='upper left')

# 주요 LEVEL3 시각화
top_level3_quarterly_counts_filtered.plot(ax=ax[2], alpha=0.6, linewidth=3)
ax[2].set_title('LEVEL3 분기별 신청 건수의 시간 변화 (총 건수 >= 500)')
ax[2].set_xlabel('연도-분기')
ax[2].set_ylabel('신청 건수')
ax[2].legend(title='LEVEL3', bbox_to_anchor=(1.01, 1), loc='upper left')

plt.tight_layout()
plt.show()

# 직책별 LEVEL1별 신청 건수 시각화
# 상위 10개 직책 확인
top_n = 10
top_positions = df['호칭'].value_counts().nlargest(top_n).index

# 상위 10개의 직책만 포함하도록 데이터프레임 필터링
top_positions_df = df[df['호칭'].isin(top_positions)]

# 직책별, Level1별 신청 건수 집계
positions_level1_counts = top_positions_df.pivot_table(index='호칭', columns='LEVEL1',
values='count', aggfunc='size', fill_value=0)

# 직책별 총 신청 건수를 기준으로 내림차순 정렬
positions_level1_counts['counts'] = positions_level1_counts.sum(axis=1)
positions_level1_counts = positions_level1_counts.sort_values(by='counts',
ascending=True).drop(columns='counts')

# 색상 팔레트 설정
palette = sns.color_palette("husl", len(positions_level1_counts.columns))
```

```
# 누적 막대 차트 생성
ax = positions_level1_counts.plot(kind='barh', stacked=True, figsize=(14, 8), color=palette)

# 각 막대에 텍스트 및 비율 표시
threshold = 1000
for i, (level1, row) in enumerate(positions_level1_counts.iterrows()):
    total = row.sum()
    for j, (level2, value) in enumerate(row.items()):
        if value > threshold:  # 값이 threshold 값보다 큰 경우에만 텍스트를 표시
            percentage = value / total * 100
            x_pos = value / 2 + row.iloc[:j].sum()
            y_pos = i
            ax.text(x_pos, y_pos,f'{percentage:.1f}%', va='center', ha='center', color='black')

# 전체 길이가 threshold 이하인 막대 투명 처리
for i, total in enumerate(positions_level1_counts.sum(axis=1)):
    if total <= threshold:
        for bars in ax.containers:
            bars[i].set_alpha(0.3)

plt.title('직책별 LEVEL1 신청 건수')
plt.xlabel('신청 건수')
plt.ylabel('신청자 직책')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
# 직책별 LEVEL2 신청건수 시각화
# 상위 n개 직책 확인
top_n = 10
top_positions = df['호칭'].value_counts().nlargest(top_n).index

# 상위 n개의 직책만 포함하도록 데이터프레임 필터링
top_positions_df = df[df['호칭'].isin(top_positions)]

# 직책별, Level2별 신청 건수 집계
positions_level2_counts = top_positions_df.pivot_table(index='호칭', columns='LEVEL2',
values='count', aggfunc='size', fill_value=0)

# 직책별 총 신청 건수를 기준으로 내림차순 정렬
positions_level2_counts['counts'] = positions_level2_counts.sum(axis=1)
positions_level2_counts = positions_level2_counts.sort_values(by='counts',
ascending=True).drop(columns='counts')

# 색상 팔레트 설정
palette = sns.color_palette("husl", len(positions_level2_counts.columns))
```

```
# 누적 막대 차트 생성
ax = positions_level2_counts.plot(kind='barh', stacked=True, figsize=(14, 8), color=palette)

# 각 막대에 텍스트 및 비율 표시
threshold = 600
for i, (level2, row) in enumerate(positions_level2_counts.iterrows()):
    total = row.sum()
    for j, (level2, value) in enumerate(row.items()):
        if value > threshold:  # 값이 threshold 값보다 큰 경우에만 텍스트를 표시
            percentage = value / total * 100
            x_pos = value / 2 + row.iloc[:j].sum()
            y_pos = i
                    ax.text(x_pos,  y_pos,f'{level2}  ({percentage:.1f}%)',  va='center',  ha='center',
color='black', fontsize=11)


# 전체 길이가 threshold 이하인 막대 투명 처리
threshold = 800
for i, total in enumerate(positions_level2_counts.sum(axis=1)):
    if total <= threshold:
        for bars in ax.containers:
            bars[i].set_alpha(0.2)

# 그래프 제목과 축 라벨 설정
plt.title('직책별 LEVEL2 신청 건수')
plt.xlabel('신청 건수')
plt.ylabel('신청자 직책')
plt.tight_layout()
plt.show()

# 직책별 LEVEL3별 신청 건수 시각화
# 상위 n개 직책 확인
top_n1 = 10
top_positions = df['호칭'].value_counts().nlargest(top_n1).index

# 상위 n개의 직책만 포함하도록 데이터프레임 필터링
top_positions_df = df[df['호칭'].isin(top_positions)]

# 상위 n개의 LEVEL3 레이블 선택
top_n2 = 10
top_level3 = df['LEVEL3'].value_counts().nlargest(top_n2).index

# 직책별, Level3별 신청 건수 집계 (상위 10개 Level3만 포함)
positions_level3_counts  =  top_positions_df[top_positions_df['LEVEL3'].isin(top_level3)].
pivot_table(index='호칭', columns='LEVEL3', values='count', aggfunc='size', fill_value=0)
```

```
# 직책별 총 신청 건수를 기준으로 내림차순 정렬
positions_level3_counts['counts'] = positions_level3_counts.sum(axis=1)
positions_level3_counts = positions_level3_counts.sort_values(by='counts',
ascending=True).drop(columns='counts')

# 색상 팔레트 설정
palette = sns.color_palette("husl", len(positions_level3_counts.columns))

# 누적 막대 차트 생성
ax = positions_level3_counts.plot(kind='barh', stacked=True, figsize=(20, 8), color=palette)

# 각 막대에 텍스트 및 비율 표시
threshold = 500
for i, (level3, row) in enumerate(positions_level3_counts.iterrows()):
    total = row.sum()
    for j, (level3, value) in enumerate(row.items()):
        if value > threshold:  # 값이 threshold 값보다 큰 경우에만 텍스트를 표시
            percentage = value / total * 100
            x_pos = value / 2 + row.iloc[:j].sum()
            y_pos = i
                    ax.text(x_pos,  y_pos,f'{level3}  ({percentage:.1f}%)',  va='center',  ha='center',
color='black', fontsize=10)

# 전체 길이가 threshold 이하인 막대 투명 처리
threshold = 840
for i, total in enumerate(positions_level2_counts.sum(axis=1)):
    if total <= threshold:
        for bars in ax.containers:
            bars[i].set_alpha(0.2)

# 그래프 제목과 축 라벨 설정
plt.title('직책별 LEVEL3 신청 건수 (상위 10개 LEVEL3)')
plt.xlabel('신청 건수')
plt.ylabel('신청자 직책')
plt.tight_layout()
plt.show()
```

 # 전처리 및 시각적 분석에 Python을 활용함.

접수번호 2024-7

공공기관 사내 헬프데스크 요구사항의 **효율적 관리를 위한 토픽 모델링 기반** 심층 분석

분석 요청자	소속	기관	성명	전OO
	휴대전화	-	전자우편	-
분석유형	▢ 데이터 전처리	▢ 데이터 시각화	■ 머신러닝	▢ 기타
데이터 분석가	김건욱 센터장, 윤주혁 사무원			
분석 기간	2024.07 ~ 2024.07(1개월)			

가.

분석 주제

분석 주제(요청내용)

- 헬프데스크에 접수되는 요구사항은 업무처리와 관련된 직원들의 불편 및 개선 사항을 담고 있으며, 이러한 데이터는 업무 효율성 및 만족도 향상을 위한 중요한 원천자료임
- 따라서 요구사항 텍스트 집합의 주제 및 특성을 깊이 있게 분석할 필요가 있으며, 이를 위해 토픽 모델링 기법을 활용하고자 함
- 기존에는 오픈소스를 통해 구현하기 편리하고 정량적 평가법을 적용할 수 있는 문서 내 단어의 등장빈도를 중심으로 분석하는 잠재디리클레할당(Latent Dirichlet Allocation 이하 LDA)기법을 자주 사용하였으나, 최근에는 BERT 기반의 문장 단위 토픽 모델링 기법도 등장함
- 본 분석에서는 헬프데스크에 접수되는 요구사항을 문장단위 및 단어 단위로 종합 분석이 가능한 CombinedTM 기법을 사용하여 최근 3년간의 헬프데스크 요청 데이터를 텍스트 분석함
- 이를 통해 헬프데스크의 운영 효율성 및 효과적인 업무처리 개선 전략 마련을 위한 기초자료로 활용될 수 있도록 함

분석 필요성 및 전략

- 업무 처리 과정에서 발생하는 직원들의 불편함과 개선 요구가 반영된 요구사항을 체계적으로 분석하는 것은 업무 효율성을 높이고 전체적인 조직의 생산성과 긍정적인 업무 환경 조성에 도움이 될 수 있음
- 기존의 LDA 기법과 발전한 BERT 기반의 문장 단위 토픽 모델링을 결합한 CombinedTM을 도입함으로써 더 정교하고 의미 있는 분석 결과를 얻고자 함
- 토픽 모델링 기법 적용: CombinedTM 기법을 사용하여 요구사항 텍스트 집합의 주제 및 키워드를 발굴함. 이를 통해 요구사항 속 숨겨진 패턴과 특성을 파악
- 정량적 수치 평가지표인 Coherence NPMI, UMass를 활용하여, 최적의 토픽 수 선정에 필요한 기준지표로 설정
- 도출된 결과를 효과적으로 해석하기 위해 워드클라우드 및 LDA결과 시각화에 특화된 PyLDAvis 라이브러리를 활용

나.

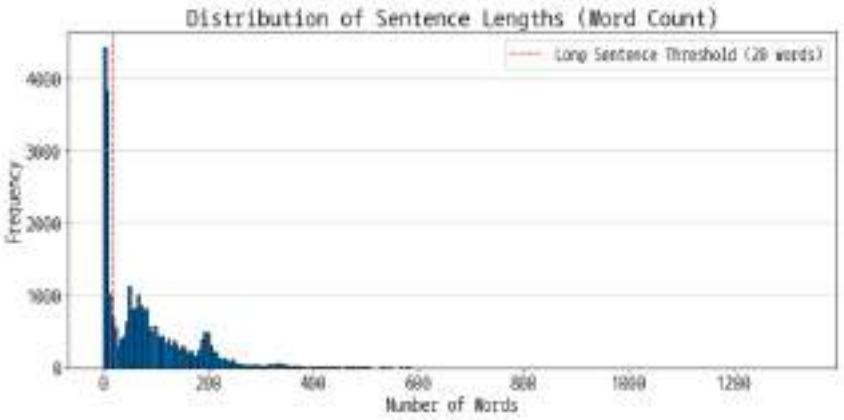
데이터 분석 및 결과

활용 데이터

- 본 분석에서 활용된 데이터는 아래와 같이 구성됨
- 신청자, 직급, 분류(대, 중, 소), 신청 시기, 신청 내용 등이 담겨있으며, 신청 내용의 경우 정형화 되어있지 않은 텍스트 데이터로 이루어짐
- 신청 내용 속 문장길이(Mecab 형태소 분석기를 활용)를 확인한 결과, 긴 문장과 짧은 문장이 섞여 있는 분포가 나타남

연번	데이터명	내용	시간 범위	데이터 규모
1	헬프데스크 지원요청 내역	신청자, 직급, 분류(대, 중, 소), 신청 시기, 신청 내용 등	2021년 8월 ~2024년 6월 (약 3년)	약 28,900건

<표> 활용 데이터



<표> 문장 길이 분포

데이터 가명처리 적용

- 데이터 분석에 앞서 데이터 제공자의 요청으로 대구가명정보활용지원센터에서 데이터 가명처리를 진행
- 분석 대상 데이터를 가명처리 솔루션을 통해 확인한 결과 식별 정보(신청자명, 사원번호)가 존재하여 가명처리(양 끝 문자를 제외한 모든 문자 '*'로 변경)를 적용

데이터 분석(전처리)

  - 결측치 확인 후 컬럼 특성 별 적절한 처리과정 적용

- '호칭' 컬럼에 결측치 존재 : 사원 데이터가 있는 경우 결합 가능하므로 공백으로 대체
- 'PROCESS_CN' 컬럼에 결측치 존재 : 담당자 답변내용이므로 본 분석에는 필요하지 않으므로 공백으로 대체
  - 데이터 내 'TITLE' 및 'REQST_CN' 컬럼은 작성자가 직접 요구사항을 작성한 데이터로 구성된 열이므로, 토픽 분석을 위해 이를 결합. 새로운 컬럼인 'Combined_Text' 컬럼을 생성함
  - 토픽모델링을 진행하기 전 불용어 사전 정의(제공받은 소스코드에 불용어 추가)
  - 온전한 문장을 대상으로 분석을 진행하는 BERTopic 구조는 불용어 처리를 가급적 줄이는 게 좋으나, 분석 데이터가 적은 적어 일관적인 토픽 도출 및 해석에 어려움을 주는 것을 방지하기 위해 아래와 같이 불용어 사전을 정의함

<그림 1> 불용어 사전 목록

데이터 분석(머신러닝)

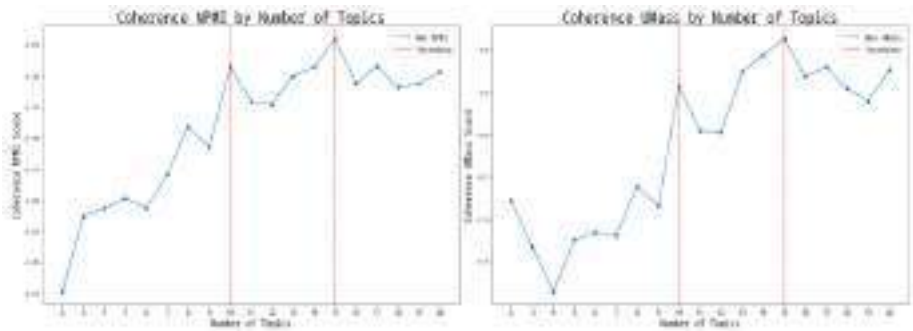
- Tokenizer 및 Vectorizer 설정 및 BoW 생성
  - 토큰화 이전에 기본적인 전처리 함수(한글 제외 문자 제거 등) 적용
  - 한국어 형태소 분석기 중 성능이 좋은 ko-Mecab을 토큰 추출에 사용
  - 문서의 특성을 더 잘 반영하여 중요 단어에 높은 가중치를 부여하고, 자주 등장하지만 의미가 적은 단어의 영향을 줄여 분석의 정확성을 높이기 위해 TF-IDF Vectorizer를 사용
  - BERT로 생성된 임베딩과 결합할 BoW 생성

<그림 2> 전처리 규칙 설정, Tokenizer 및 Vectorizer 정의 및 설정

- Pretrained BERT 모델을 활용한 임베딩 생성
  - 생성된 BoW와 결합하여 분석할 문장 임베딩을 사전 학습된 BERT 모델로 생성
  - SentenceTransformer 라이브러리에 있는 사전 학습 모델 중 카카오 브레인이 공개한 KorNLI, KorSTS 두 한국어 데이터셋을 모두 활용하여 학습된 모델 중 성능이 뛰어난 'ko-sroberta-multitask' 모델을 사용하여 임베딩을 생성함
  - BoW와 생성된 임베딩을 종합하여 CombinedTM에 필요한 학습 데이터를 생성함

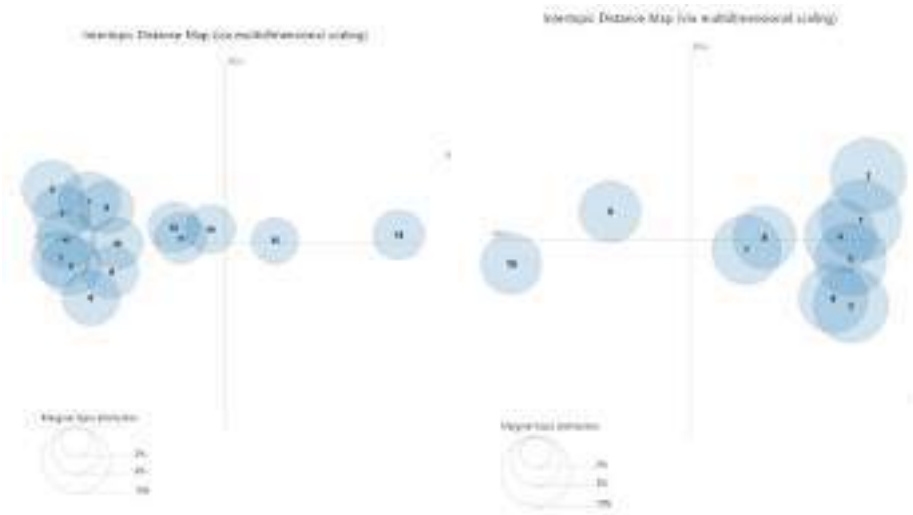
<그림 3> 사전학습된 BERT 모델 로드 및 학습용 데이터 생성

- 모델 학습 및 Coherence NPMI, UMass를 통한 최적 토픽수 탐색
  - 생성한 학습 데이터셋을 통해 토픽 모델링 진행
  - 두 지표 모두 값이 클수록(양의 정수 방향) 더 좋은 토픽 일관성을 나타냄. 이는 해당 토픽의 단어들이 서로 더 잘 연관되어 있으며 해당 토픽의 단어들이 같은 문서에서 자주 함께 나타난다는 것을 의미함



<표> Coherence NPMI 및 UMass 지표

  - 지표상으로는 15개의 토픽이 제일 적절하나, 실제 토픽들의 분포 및 등장 단어 빈도 및 결과를 확인해본 결과, 의미 중복 및 불필요한 토픽 분할이 있어 토픽수를 줄이면서도 종합 지표가 두 번째로 높은 10개의 토픽을 선정함



<표> 15개 토픽 분포(좌), 10개 토픽 분포(우)

분석 결과

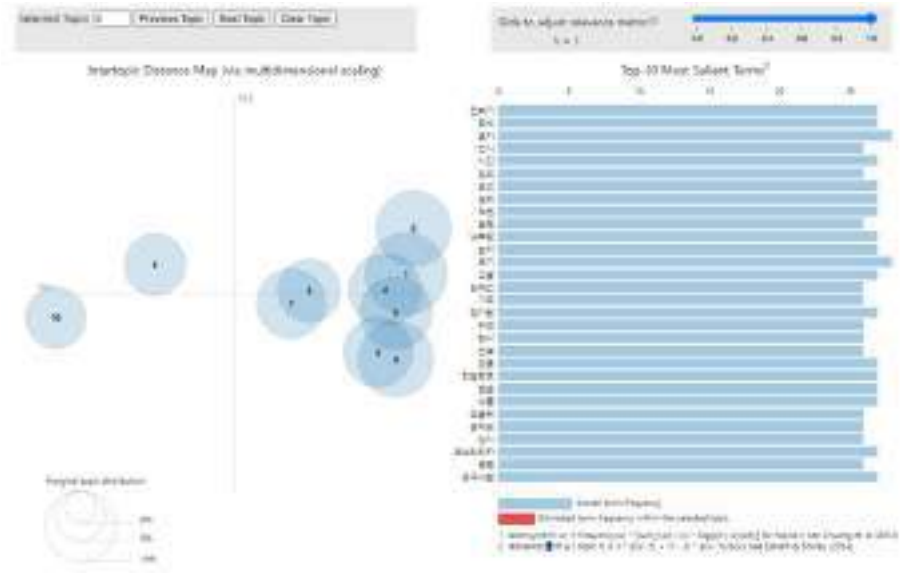
- 토픽모델링 결과
  - 토픽 모델링 결과 아래와 같이 토픽별 상위 10개 키워드가 도출됨

	0	1	2	3	4	5	6	7	8	9
Topic0	삭제	메일링	리스트	계정	메일	모바일	성일	목적	명시	주소
Topic1	전화기	회수	인사	설치	발령	명령	설정	퇴직	입사	개편
Topic2	초기	비밀	패스워드	해제	사물함	잠김	시설	개인	화장실	잠금
Topic3	출장	결재	지정	오류	복명	교육	결의	카드	대결	발생
Topic4	연동	홈페이지	담당자	게시	대표	폐기	사업	디지털	정부	데이터
Topic5	휴가	시간	수정	근태	중계	근무	실제	출퇴근	워크	스마트
Topic6	권고	덕원	임의	고용	내부망	환자	언론	수출	정가람	김재건
Topic7	차량	등록	전화	출입	휴대	권한	방화벽	시스템	개방	서버
Topic8	제주	점검	복합	교육장	권고	연결	지하	아프리카	통부	기정
Topic9	영상	예외	사이트	신청자	인가	접속	본원	대구	장소	증빙

<그림 4> 토픽별 상위 키워드 10개

- 워드 클라우드 시각화

• 토픽모델링 결과 시각화



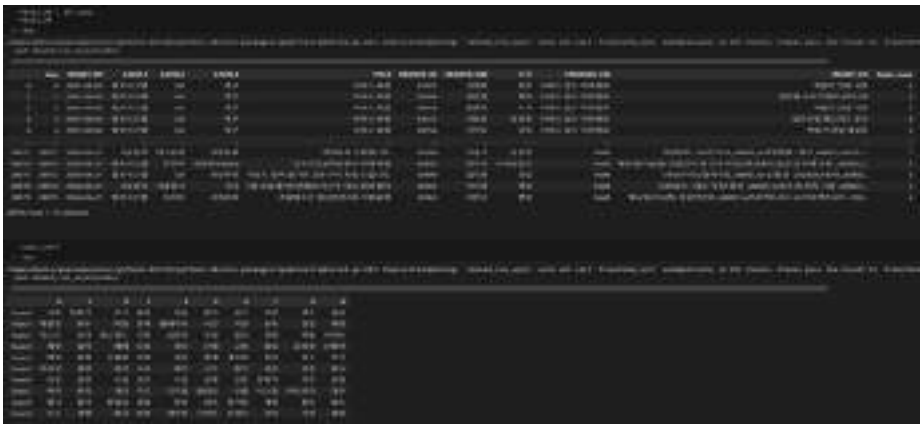
<표> 토픽모델링 결과 분포 시각화

다.

분석 활용 전략

결론

- 업무처리와 관련된 직원들의 불편 및 개선 사항을 분석하여 업무 효율성 및 만족도 향상 대책 마련에 필요한 기초자료로 활용
- 토픽에 포함된 핵심 키워드를 종합 분석하고 전문가 자문 등을 통해 일관적으로 발생되는 문제를 파악 가능
- 생성된 새로운 설명자를 통해 기존에는 파악할 수 없었던 새로운 문제를 발굴하고 해결하여 업무 효율성 제고 기대



<그림 5> 최종 산출물

Appendix

```
# 데이터 로드 및 전처리 관련
import os  # 운영체제와 상호작용하기 위한 모듈
import openpyxl  # 엑셀 파일을 읽고 쓰기 위한 모듈
import pandas as pd  # 데이터 조작 및 분석을 위한 라이브러리
import numpy as np  # 수치 계산을 위한 라이브러리
import re, warnings, random  # 정규 표현식, 경고 메시지 제어, 난수 생성을 위한 모듈
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer  # 텍스트 데이터를 벡터화하기 위한 도구
from konlpy.tag import Mecab  # 한국어 형태소 분석기

# 진행상황 시각화
from tqdm import tqdm  # 반복문 실행 상태를 시각적으로 보여주는 라이브러리

# 머신러닝 관련
import torch  # 딥러닝 및 텐서 연산을 위한 프레임워크
import gc  # 가비지 컬렉션(메모리 관리) 모듈

# 데이터 시각화 관련
import matplotlib.pyplot as plt  # 데이터 시각화를 위한 라이브러리
plt.rc('font', family='NanumGothicCoding')  # Matplotlib에서 한글 폰트 설정
from wordcloud import WordCloud  # 워드 클라우드 생성을 위한 라이브러리
import pyLDAvis as vis  # 토픽 모델링 결과를 시각화하는 도구

import warnings
# 경고창 무시
warnings.filterwarnings("ignore", category=DeprecationWarning)  # 특정 경고 메시지 무시
import matplotlib as mpl
mpl.rcParams['axes.unicode_minus'] = False  # Matplotlib에서 음수 기호 깨짐 방지

# 데이터 형태 확인
df = pd.read_excel('data/helpdesk_info_main.xlsx').reset_index(names='key')
df.head(2)

pd.DataFrame(df.isna().sum()).T

# 모든 nan 값을 ''로 대체
df = df.fillna('')  # 데이터프레임의 모든 결측값을 빈 문자열로 대체

# 제목과 내용을 결합(제목의 경우, 직접 타이핑하기 때문에 토픽모델링에 포함시켜야함)하여 새로운 문장 생성
df['Combined_Text'] = df['TITLE'] + ' ' + df['REQST_CN']  # 'TITLE'과 'REQST_CN' 컬럼을 결합하여 'Combined_Text'라는 새로운 컬럼 생성
```

```
# 결측값이 잘 처리되었는지 확인
pd.DataFrame(df.isna().sum()).T  # 각 컬럼의 결측값 개수를 세고, 이를 데이터프레임으로 변환하여 출력
temp = df.copy()

# 토크나이저를 사용하여 단어 개수 계산
temp['word_count'] = temp['REQST_CN'].apply(lambda x: len(Mecab().morphs(x)))

# 긴 문장의 기준: 20단어 이상  ( 영어 기준, 러프한 확인용)
long_sentences = temp[temp['word_count'] > 20]

# 문장 길이 분포 시각화
plt.figure(figsize=(14, 6))
plt.hist(temp['word_count'], bins=250, alpha=0.99, color='#007ACC', edgecolor='black')
plt.title('Distribution of Sentence Lengths (Word Count)', fontsize=24)
plt.xlabel('Number of Words', fontsize=18)
plt.ylabel('Frequency', fontsize=18)
plt.axvline(x=20, color='red', linestyle='--', label='Long Sentence Threshold (20 words)')
plt.legend(fontsize=16)
plt.grid(axis='y', alpha=0.75)
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
plt.show()

# 필요한 라이브러리 설치
# %pip install contextualized-topic-models==2.5.0  # CTM 라이브러리 설치
# %pip install pyldavis  # pyLDAvis 라이브러리 설치
# Mecab 설치 방법은 https://github.com/hephaex/mecab-ko?tab=readme-ov-file 참조

# CTM 관련 import
from sentence_transformers import SentenceTransformer  # 문장 임베딩을 위한 라이브러리
from contextualized_topic_models.models.ctm import CombinedTM  # CTM 모델 클래스
from contextualized_topic_models.utils.data_preparation import TopicModelDataPreparation, bert_embeddings_from_list  # 데이터 준비 유틸리티
from contextualized_topic_models.evaluation.measures import CoherenceNPMI, CoherenceUMASS  # 토픽 일관성 측정
from contextualized_topic_models.utils.preprocessing import WhiteSpacePreprocessing  # 전처리 유틸리티

# 재현성을 위해 랜덤 시드값 전부 고정
random.seed(2024)  # 파이썬 기본 랜덤 시드 고정
np.random.seed(2024)  # numpy 랜덤 시드 고정
torch.manual_seed(2024)  # PyTorch 랜덤 시드 고정
```

```
# Topic 모델링을 위한 문서 집합 생성
documents = df['Combined_Text'].to_list()  # 'Combined_Text' 열의 모든 값을 리스트로 변환하여 documents 변수에 저장

# 원본 문장 구조를 최대한 보존하면서 최소한의 전처리만 진행된 문서 집합
preprocessed_documents = []

# 불용어 리스트 정의
stopwords = ['사용', '청사', '정보', '명', '해당', '역', '상세', '지원', '요청', '팀', '작업', '확인', '제목', '서비스', '신청', '통합', '작성', '신청서','시기', '첨부', '스캔',
             '내용', '요망', '선', '업무', '사항', '번호', '관리', '추가', '건', '처리', '층', '및', '재', '증', '사용자', '기간', '회의','서울', '기타', '예정', '기록', '문의',
             '등', '성명', '부서', '종류', '생', '호', '앞', '실', '수리', '일자', '예시', '위', '경우', '포함', '다모아', '모아','파일', '비', '사유', '성함', '관련', '번',
             '편', '본부', '기능', '메뉴', '문서', '층수', '위치', '안녕', '습니다', '세요', '감사', '입니다','비슷', '으로부터', '그러나', '합니다', '에서', '드립니다', '부탁',
            '으로', '는데', '가능', '드리', '해서', '예제', '바랍니다','누구', '만약', '위한', '깅에' ,'따른','아울러','군데'
             ]

# stopwords = []

def clean_text(text):
    # 특수 문자를 공백으로 대체하고 개행 문자를 제거
    text = re.sub(r"[^가-힣\s]", " ", text.replace('\n', ' '))
    # 연속된 공백을 단일 공백으로 대체
    text = re.sub(r'\s+', ' ', text)
    # 한 글자 단어 제거
    text = ' '.join([word for word in text.split() if len(word) > 1])
    return text

# 원본 문서 전처리 (특수문자 제거 및 한글만 남기기)
preprocessed_documents = [clean_text(line) for line in tqdm(documents) if line and not line.replace(' ', '').isdecimal()]

class CustomTokenizer:
    def __init__(self, tagger, stopwords):
        self.tagger = tagger  # 형태소 분석기 초기화
        self.stopwords = stopwords  # 불용어 리스트 초기화

    def clean_text(self, text):
        # 특수 문자 제거
        text = re.sub(r"[^가-힣\s]", "", text)
        # 한글 외 문자 제거
        text = re.sub(r"[^ㄱ-ㅎㅏ-ㅣ가-힣 ]", "", text)
        return text
```

```
    def __call__(self, sent):
        # 텍스트 정리
        sent = self.clean_text(sent)
        # 형태소 분석을 통해 단어 토큰화
        word_tokens = self.tagger.morphs(sent)
        # 불용어 제거 및 길이가 1 이하인 단어 제거
        result = [word for word in word_tokens if word not in self.stopwords and len(word) > 1]
        return result

# 커스텀 토크나이저 초기화
custom_tokenizer = CustomTokenizer(Mecab(), stopwords)

# TfidfVectorizer 초기화 및 BoW 임베딩 생성
vectorizer = TfidfVectorizer(tokenizer=custom_tokenizer, max_features=3000)  # TF-IDF 기반 벡터화

train_bow_embeddings = vectorizer.fit_transform(preprocessed_documents)  # 전처리된 문서를 벡터화하여 BoW 임베딩 생성

# 결과 출력
print(train_bow_embeddings.shape)  # 생성된 BoW 임베딩의 형태 출력

vocab = vectorizer.get_feature_names_out()  # 벡터화 과정에서 추출된 단어 목록
id2token = {k: v for k, v in zip(range(0, len(vocab)), vocab)}  # 단어 인덱스와 단어를 매핑하는 딕셔너리 생성

print(len(vocab))  # 고유 단어의 개수 출력

# GPU 사용 여부 설정
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")  # CUDA 사용 가능 시 GPU, 아니면 CPU 설정

# BERT 임베딩 생성 함수 수정
def bert_embeddings_from_list_gpu(documents, model_name):
    model = SentenceTransformer(model_name)  # BERT 모델 초기화
    model = model.to(device)  # 모델을 GPU로 이동
    embeddings = model.encode(documents, batch_size=32, convert_to_tensor=True, show_progress_bar=True, device=device)  # 문서 리스트를 BERT 임베딩으로 변환
    return embeddings.cpu().numpy()  # 결과를 다시 CPU로 이동하여 numpy 배열로 변환

# GPU를 사용하여 임베딩 생성
# train_contextualized_embeddings = bert_embeddings_from_list_gpu(preprocessed_documents, "sentence-transformers/xlm-r-100langs-bert-base-nli-stsb-mean-tokens")  # BERT 임베딩 생성
```

```
train_contextualized_embeddings = bert_embeddings_from_list_gpu(preprocessed_documents, "jhgan/ko-sroberta-multitask") # 한국어 BERT 임베딩 생성 [https://github.com/jhgan00/ko-sentence-transformers?tab=readme-ov-file]

# 가비지 컬렉션 실행
gc.collect() # 가비지 컬렉션을 통해 사용하지 않는 메모리 정리
# 캐시된 메모리 비우기
torch.cuda.empty_cache() # GPU 캐시된 메모리 비우기

# SBERT 데이터 준비
qt = TopicModelDataPreparation() # TopicModelDataPreparation 클래스 초기화

training_dataset = qt.load(train_contextualized_embeddings, train_bow_embeddings, id2token) # 훈련 데이터셋 생성

# 여러 토픽 수에 따른 평가 결과를 저장할 리스트
topic_nums = range(2, 21)
npmi_scores = []
umass_scores = []

# 문서를 단어 리스트 형식으로 변환
texts = [doc.split() for doc in preprocessed_documents]

for num_topics in topic_nums:
  # CTM (Contextualized Topic Model) 모델을 생성 및 학습
  # bow_size: BoW 벡터의 크기, contextual_size: BERT 임베딩 차원, n_components: 토픽 수, num_epochs: 에폭 수
  ctm = CombinedTM(bow_size=len(vocab), contextual_size=768, n_components=num_topics, num_epochs=10)
  ctm.fit(training_dataset) # 모델 학습

  # 학습된 모델을 저장할 디렉토리 설정
  model_dir = f'models/ctm_{num_topics}_topics'
  if not os.path.exists(model_dir):
    os.makedirs(model_dir) # 디렉토리가 없으면 생성
  ctm.save(model_dir) # 모델 저장

  # 모델에서 토픽 리스트 추출
  topics = ctm.get_topic_lists()

  # CoherenceNPMI 계산
  npmi = CoherenceNPMI(texts=texts, topics=topics) # NPMI 계산을 위한 객체 생성
  npmi_score = npmi.score() # NPMI 점수 계산
  npmi_scores.append(npmi_score) # NPMI 점수를 리스트에 추가
```

```
    # CoherenceUMass 계산
    umass = CoherenceUMASS(texts=texts, topics=topics)  # UMass 계산을 위한 객체 생성
    umass_score = umass.score()  # UMass 점수 계산
    umass_scores.append(umass_score)  # UMass 점수를 리스트에 추가

    # 결과 출력
     print(f"Number of Topics: {num_topics}, Coherence NPMI: {npmi_score}, Coherence UMass: {umass_score}")

    # 가비지 컬렉션 실행
    gc.collect()  # 메모리 정리
    # 캐시된 메모리 비우기
    torch.cuda.empty_cache()  # GPU 캐시된 메모리 비우기

# Identify the highest scores
max_npmi_score = max(npmi_scores)
max_umass_score = max(umass_scores)
max_npmi_index = npmi_scores.index(max_npmi_score)
max_umass_index = umass_scores.index(max_umass_score)

# 토픽이 10개인 경우의 인덱스 식별
topic_10_index = topic_nums.index(10)

plt.figure(figsize=(24, 8))

# Coherence NPMI 플롯
plt.subplot(1, 2, 1)
plt.plot(topic_nums, npmi_scores, marker='o')
plt.title('Coherence NPMI by Number of Topics', fontsize=30)
plt.xlabel('Number of Topics', fontsize=19)
plt.ylabel('Coherence NPMI Score', fontsize=19)
plt.xticks(range(min(topic_nums), max(topic_nums) + 1, 1),fontsize=15)  # X축 눈금 레이블 크기 설정
plt.axvline(x=topic_nums[max_npmi_index], color='r', linestyle='--', linewidth=1.5, label='Max NPMI')  # 최대 NPMI에서 빨간색 점선
plt.axvline(x=10, color='r', linestyle='--', linewidth=1.5, label='Secondary')  # 토픽이 10개인 경우 파란색 점선
plt.scatter(topic_nums[max_npmi_index], max_npmi_score, color='r', zorder=5)  # 최대 NPMI에서 빨간색 점
plt.scatter(10, npmi_scores[topic_10_index], color='r', zorder=5)  # 토픽이 10개인 경우 파란색 점
plt.legend(fontsize=15)
```

```
# Coherence UMass 플롯
plt.subplot(1, 2, 2)
plt.plot(topic_nums, umass_scores, marker='o')
plt.title('Coherence UMass by Number of Topics', fontsize=30)
plt.xlabel('Number of Topics', fontsize=19)
plt.ylabel('Coherence UMass Score', fontsize=19)
plt.xticks(range(min(topic_nums), max(topic_nums) + 1, 1),fontsize=15)  # X축 눈금 레이블 크기 설정
plt.axvline(x=topic_nums[max_umass_index], color='r', linestyle='--', linewidth=1.5, label='Max UMass')  # 최대 UMass에서 빨간색 점선
plt.axvline(x=10, color='r', linestyle='--', linewidth=1.5, label='Secondary')  # 토픽이 10개인 경우 파란색 점선
plt.scatter(topic_nums[max_umass_index], max_umass_score, color='r', zorder=5)  # 최대 UMass에서 빨간색 점
plt.scatter(10, umass_scores[topic_10_index], color='r', zorder=5)  # 토픽이 10개인 경우 파란색 점
plt.legend(fontsize=15)

plt.tight_layout()
plt.show()



# CombinedTM 객체 생성
ctm = CombinedTM(bow_size=len(vocab), contextual_size=768, n_components=num_topics, num_epochs=20)
# 모델 디렉토리 경로와 에포크 설정
model_dir = 'models/ctm_10_topics/contextualized_topic_model_nc_10_tpm_0.0_tpv_0.9_hs_prodLDA_ac_(100, 100)_do_softplus_lr_0.2_mo_0.002_rp_0.99'
epoch = 9
# 모델 로드
ctm.load(model_dir=model_dir, epoch=epoch)

topic_distributions = ctm.get_doc_topic_distribution(training_dataset)  # 각 문서에 대한 토픽 분포 계산

# 각 문서의 토픽 할당
assigned_topics = np.argmax(topic_distributions, axis=1)  # 각 문서에 대해 가장 높은 확률을 가진 토픽 할당

df['Topic_num'] = assigned_topics  # 데이터프레임에 할당된 토픽 번호 추가
df.to_csv(KCTM_result_appended_data.csv',index=False)  # 결과를 CSV 파일로 저장
# df = df.drop(columns=['Combined_Text'],axis=1)
df  # 데이터프레임 출력
```

```
# pd.DataFrame(ctm.get_topic_lists(),columns=['Topic0','Topic1','Topic2','Topic3','Topic4','Topic5','Topic6','Topic7','Topic8','Topic9']).to_csv('topic_list.csv',index=False)
topic_word = pd.DataFrame(ctm.get_topic_lists(),columns=['Topic0','Topic1','Topic2','Topic3','Topic4','Topic5','Topic6','Topic7','Topic8','Topic9']).T
topic_word

lda_vis_data = ctm.get_ldavis_data_format(vocab, training_dataset, n_samples=10)

ctm_pd = vis.prepare(**lda_vis_data)
vis.display(ctm_pd)

def get_wordclouds(ctm, num_topics, n_words=20, background_color="black", width=600, height=300,
                   font_path='/usr/share/fonts/truetype/nanum/NanumBarunGothic.ttf', n_rows=5, n_cols=4,dpi=300):
    """
    서브플롯을 사용하여 여러 토픽에 대한 워드클라우드를 그립니다.

    :param ctm: 토픽 모델 객체
    :param num_topics: 표시할 토픽 수
    :param n_words: 각 워드 클라우드에 표시할 단어 수
    :param background_color: 워드 클라우드의 배경 색상
    :param width: 각 워드 클라우드 이미지의 너비
    :param height: 각 워드 클라우드 이미지의 높이
    :param font_path: 워드 클라우드에 사용할 폰트의 경로
    :param n_rows: 서브플롯의 행 수
    :param n_cols: 서브플롯의 열 수
    """
    # 서브플롯 생성
    fig, axes = plt.subplots(n_rows, n_cols, figsize=(n_cols * 6, n_rows * 4), dpi=dpi)
    axes = axes.flatten()

    for topic_id in range(num_topics):
        # 각 토픽의 단어 분포 가져오기
        word_score_list = ctm.get_word_distribution_by_topic_id(topic_id)[:n_words]
        word_score_dict = {tup[0]: tup[1] for tup in word_score_list}
        # 워드 클라우드 생성
     wordcloud = WordCloud(width=width, height=height, background_color=background_color, font_path=font_path,
                           colormap='viridis', # 색상맵을 viridis로 설정
                           font_step=1, max_font_size=100, relative_scaling=0.5,
                           prefer_horizontal=0.9
                          ).generate_from_frequencies(word_score_dict)
        # 서브플롯에 워드 클라우드 이미지 표시
        axes[topic_id].imshow(wordcloud, interpolation='bilinear')
```

```
        axes[topic_id].axis("off")
        axes[topic_id].set_title("Topic " + str(topic_id), fontsize=18)

    # 남은 서브플롯 숨기기
    for ax in axes[num_topics:]:
        ax.axis("off")

    plt.tight_layout()
    plt.show()

# 사용 예제
get_wordclouds(ctm, num_topics=10, n_words=20, font_path='/usr/share/fonts/truetype/
nanum/NanumSquareEB.ttf', n_rows=4, n_cols=3,dpi=600)

# 최종 결과 확인
final_df = df.copy()
final_df

topic_word
```

접수번호 2024-8

대구로 리뷰 데이터를 활용한 샘플 데이터 생성

분석 요청자	소속	시민	성명	김OO
	휴대전화	-	전자우편	-
분석유형	■ 데이터 전처리	□ 데이터 시각화	□ 머신러닝	□ 기타
데이터 분석가	김건욱 센터장, 김현정 사무원			
분석 기간	2024.08 ~ 2024.08(1개월)			

가.

분석 주제

분석 주제(요청내용)

- 대구빅데이터활용센터는 데이터 분석 인프라를 개방하여 공공 및 민간 데이터를 다양한 연구와 분석에 활용할 수 있도록 지원하고 있으며, 대구로 배달앱 거래이력 데이터를 포함한 다양한 데이터를 보유
- 그러나 보안상의 이유로 원본 데이터의 외부 반출이 불가능한 상황에서, 일반시민(경진대회 참여자)은 원본 데이터의 구조적 패턴과 통계적 특성을 반영한 샘플 데이터를 활용하여 감성 분석을 진행할 수 있도록 지원을 요청
- 감성 분석은 리뷰 데이터에 포함된 소비자들의 긍정적 또는 부정적 반응을 파악하여, 배달앱 서비스의 사용자 경험을 향상시키는 데 필요한 인사이트를 도출하는 중요한 분석 기법임
- 해당 샘플 데이터는 빅데이터 경진대회 분석에 활용될 예정이며, 향후 유사 요청이 있을 시 활용될 수 있을 것임

분석 필요성 및 전략

- 샘플 데이터는 원본 데이터의 구조적 패턴과 통계적 특성을 반영하여 생성되며, 이를 통해 보안 요구사항을 준수하면서도 안전하게 활용될 수 있도록 설계됨. 따라서 원본 데이터와 유사한 분석 환경을 제공할 수 있음
- 샘플 데이터 생성 과정에서는 복잡한 기계 학습 모델을 사용하는 방법 대신, 무작위 샘플링과 데이터 변형 기법을 활용하는 방법을 선택
- 가맹점명은 다양한 접두사와 접미사를 조합하여 무작위로 생성되고, 리뷰 내용은 기존 데이터의 문장 구조를 기반으로 변형. 작성일과 별점은 원본 데이터의 분포를 유지하면서 무작위로 생성
- 이를 통해 실제 데이터에 대한 직접적인 접근 없이도 변환 데이터를 활용하여 안전하게 초기 분석과 모델링 작업을 수행할 수 있음
- 궁극적으로, 대구 빅데이터활용센터의 데이터 분석 인프라를 활성화하고, 향후 유사한 연구 및 분석 요청에 대한 대응력을 강화하는 데 중요한 도구로 작용

나.

데이터 분석 및 결과

활용 데이터

- 본 분석에서 활용된 데이터는 아래와 같이 구성됨
- 가맹점명, 리뷰 내용, 작성일, 별점 정보를 포함

데이터명	내용	시간 범위	데이터 규모 (레코드 수)
대구로 배달앱 거래 이력 데이터 (리뷰)	가맹점명, 리뷰내용, 작성일, 별점 정보	2022년 1월 ~2022년 12월	627,817

<표> 활용 데이터

데이터 분석

- 데이터 분석 과정은 아래와 같음

  1) 데이터 수집: 대구로 배달앱 거래이력 데이터 중 리뷰데이터를 수집, 결측치 등 데이터 품질 검사

<표> 데이터 품질 검사

2) 데이터 가공: 수집된 데이터를 샘플 데이터 생성에 적합한 형태로 가공, 전처리 과정에서는 결측치 처리, 필요한 정보의 추출 및 변환 등

<표> 필요한 정보 추출 및 변환

3) 데이터 생성: 샘플 데이터를 생성하기 위해 필요한 함수를 정의, 실행

- 가맹점명: 다양한 접두사, 아이템, 접미사를 조합하는 방식 사용. 각각의 요소는 사전에 정의된 리스트에서 무작위로 선택되어 새로운 가맹점명 생성
- 리뷰내용: 원본 리뷰 텍스트를 10대 남녀가 자주 사용하는 말투와 대구 사투리로 변환.사전 정의된 변환 규칙을 적용
- 작성일: 시간적 분포의 유지를 위해 데이터셋의 시작일과 종료일 범위 내에서 무작위로 날짜와 시간을 선택하는 방법을 사용

<표> 컬럼별 함수 정의

```
# 그룹정보와 변환된 리뷰를 포함한 DataFrame 생성
df['가맹점명'] = [generate_random_store_name() for _ in range(len(df))]
df['리뷰내용'] = [to_teen_daegu_dialect(random.choice(review_texts)) for _ in range(len(df))]
df['작성일'] = [random_date(start_date, end_date) for _ in range(len(df))]
df['별점'] = [random.randint(1, 5) for _ in range(len(df))]

# 합성 데이터를 엑셀 파일로 저장
output_file_path_translated = 'D:/Synthetic data/fake_synthetic_data.xlsx'
df.to_excel(output_file_path_translated, index=False)
```

<표> 샘플 데이터 생성

다.

분석 활용 전략

결론

- 생성된 샘플 데이터는 대구로 배달앱 리뷰 데이터를 기반으로 하여, 외부 연구자들이 안전하게 초기 분석 및 모델 개발을 수행할 수 있는 기초 자료로 활용될 수 있음
- 또한, 원본 데이터의 보안성을 유지하면서도, 실제 데이터와 유사한 패턴과 구조를 통해 연구자들이 개발한 모델을 적용할 때 유의미한 결과를 기대할 수 있는 중요한 도구로 작용
- 향후 유사한 데이터 요청 및 분석 작업에서 활용될 수 있으며, 센터 내 이용자들에게 데이터 가공 및 분석의 편의를 제공할 수 있는 기반을 마련함

	가맹점명	리뷰내용	작성일	별점
0	정갈한스테이크바	땅치는 무족권 세트 3번 이지	2022-12-17 11:33:21	1
1	최고피자식당	맛있고 배달도 빠르네요~ 자주 주문할께요!\n리뷰서비스 주신 콘도 잘 묵었심더~\n...	2022-05-30 08:40:29	3
2	맛있는떡볶이코너	짱짱! 맛있었어용~ 고마워용~\n쟬 좋아하는 떡볶이임ㅋㅋ	2022-03-05 18:36:41	5
3	싱싱한스테이크카페	맛있습니다.\n리뷰 이벤트 참여 안 해서 사진을 안 찍었는데\n그럼에도 꼭 리뷰 달...	2022-05-31 07:09:19	2
4	신선한라면바	맛있어요! 덕분에 배 터지게 먹었어요~여러가지 서비스로 푸짐해서 좋아요	2022-11-02 03:38:04	3
...	...	...	...	...
595594	행복한김밥집	수제버거 짱,, 정말 모든재료들이 다 맛있었는데 \n딱 하나만 아쉬워용 빵이 퍽퍽빵...	2022-10-25 01:28:20	4
595595	진짜치킨바	종종 포장으로 먹던 집인데 대구로에도 입점했네요~ 역시 맛있어요! 잘 먹었어요!!\n	2022-08-21 02:43:05	1
595596	황금찜닭가게	음식이 정말 존맛탱~ 잘 먹었습니다!\n떡볶이 제 스타일!!!ㅇ 맛있었어용!	2022-06-26 11:16:04	3
595597	바른김밥마을	배달 요청에 문앞에 두고 벨 눌러 달라고 했는데 문앞에 두고 그냥 가셔서 30분은 ...	2022-09-01 14:20:45	3
595598	바른찜닭횟집	너무 맛있게 먹느라 한참 먹고나서 사진촬영!! 생각났어요 ㅠ 오늘은 사진 없이 리뷰...	2022-11-17 12:40:11	5

<표> 최종 산출물(변환 데이터)

Appendix

```
import pandas as pd
import openpyxl
import random
from datetime import datetime, timedelta

# 데이터 로드
file_path = 'C:/Users/ddibo/Synthetic_data/리뷰.xlsx'
df = pd.read_excel(file_path)
#초기 데이터 확인
print(df.head())

# 결측치 확인
df.isna().sum()

# 리뷰 내용이 없는 행 제거
df = df.dropna(subset=['리뷰내용'])

# 가맹점명, 리뷰내용, 별점의 고유값 추출
store_names = df['가맹점명'].unique()
review_texts = df['리뷰내용'].unique()
star_ratings = df['별점'].unique()

# 데이터셋의 작성일 범위 정의
start_date = df['작성일'].min()
end_date = df['작성일'].max()

# 가맹점명 생성 함수
# 새로운 가맹점명을 무작위로 생성하기 위한 함수를 정의
def generate_random_store_name():
    # 가맹점 이름을 구성하는 접두사, 주요 아이템, 접미사 리스트를 정의
    prefixes = ["바른", "최고", "행복한", "맛있는", "싱싱한", "정성", "황금", "푸른", "신선한", "정갈한",
"별미", "진짜"]
    items = ["치킨", "피자", "국밥", "초밥", "족발", "떡볶이", "돈까스", "찌개", "스테이크", "샐러드", "햄
버거", "라면", "김밥", "찜닭", "파스타"]
     suffixes = ["집", "가게", "식당", "주점", "전문점", "카페", "횟집", "마을", "코너", "뷔페", "바", "포
차", "그릴"]

    # 각각의 리스트에서 무작위로 하나씩 선택해 결합하여 가맹점명 생성
    return random.choice(prefixes) + random.choice(items) + random.choice(suffixes)
```

```
# 리뷰 변환 함수
def to_teen_daegu_dialect(review):
    # 원본 리뷰 내용을 변환할 대체 표현을 사전에 정의
  transformations = {
    '맛있네요': '존맛탱',
    '재주문의사★★★★★입니다~': '진짜 또 시킬거임!',
    '최고예요!': '짱짱!',
    '맛있는 음식 감사합니다.': '맛있었어용~ 고마워용~',
    '치킨먹고싶음 항상 여기서 시켜먹어요~!!': '치킨 땡기면 무조건 여기임!',
    '잘 먹었습니다.': '잘 묵었심더~',
    '음식이 정말 맛있네요~ 잘 먹었습니다!': '음식 핵맛! 잘 묵었심더!',
    '배달 빠르고 맛도 좋습니다~ 잘 먹었습니다.': '배달 개빠르고 맛도 존맛탱! 잘 묵었심더~',
    '덕분에 든든한 식사 했습니다~': '덕분에 배 터지게 먹었어요~',
    '너무 잘 먹었습니다~': '너무 잘 묵었심더~',
    '배달 빠르고 좋아요': '배달 겁나 빠르고 좋아요~',
    '사장님 친절해요': '사장님 개친절해요~'
  }
  # 사전에 정의한 변환 규칙을 원본 리뷰 내용에 적용
  for original, transformed in transformations.items():
    review = review.replace(original, transformed)
  return review

# 무작위 날짜 생성 함수
# 주어진 날짜 범위 내에서 무작위로 날짜를 생성하기 위한 함수를 정의
def random_date(start, end):
  return start + timedelta(seconds=random.randint(0, int((end - start).total_seconds())))

# 샘플 데이터 생성
# 기존 데이터를 기반으로 가맹점명, 리뷰 내용, 작성일, 별점을 무작위로 생성하여 새로운 데이터를 만듦
df['가맹점명'] = [generate_random_store_name() for _ in range(len(df))]
df['리뷰내용'] = [to_teen_daegu_dialect(random.choice(review_texts)) for _ in range(len(df))]
df['작성일'] = [random_date(start_date, end_date) for _ in range(len(df))]
df['별점'] = [random.randint(1, 5) for _ in range(len(df))]

# 생성된 샘플 데이터를 엑셀 파일로 저장
output_file_path_translated = 'D:/Synthetic data/synthetic_data.xlsx'
df.to_excel(output_file_path_translated, index=False)
```

접수번호 2024-9

생활인구 빅데이터마트 구축을 위한 데이터전처리

분석 요청자	소속	기업	성명	-
	휴대전화	-	전자우편	-
분석유형	■ 데이터 전처리	▢ 데이터 시각화	▢ 머신러닝	▢ 기타
데이터 분석가	김건욱 센터장, 윤주혁 사무원			
분석 기간	2024.08 ~ 2024.08(1개월)			

가.

분석 주제

분석 주제(요청내용)

- 현재 대구시는 2040년 도시기본계획 수립을 진행중이며, 이를 위한 기반자료로 계획인구 계획인구 도시기본계획 수립 시, 해당 도시나 지역의 장래 인구를 예측하여 설정한 목표 인구 수
- 를 산정할 필요가 있음
- 기 작성된 2030 도시기본계획에서 예측한 인구수가 현재 측정된 실재 수치에 비해 상당히 감소된 수치임이 확인되며 더욱 감소할 전망을 보이고 있음
- 따라서, 국토교통부는 도시기본계획 수립 시, 계획인구 예측치와 실재 수치간의 편차를 줄이기 위해, 통계청의 추정치('2024년 계획인구' : 206만)를 기본으로 하되, 대구시의 다양한 인구 데이터를 반영하도록 지침을 내린 상황임
- 다양한 인구데이터를 반영하기 위해 빅데이터활용센터가 보유한 SKT 통신사 생활인구 데이터 중 PCELL 단위 데이터를 활용하여 2040년 계획인구 수 추계에 필요한 기초자료를 만들고자 함
- 본 분석에서는 약 1억 9천만건에 달하는 3개년치의 대규모 생활인구 데이터를 전처리 및 분석할 시, 서버 자원을 최대한 효율적으로 활용하여 빠르게 분석할 수 있는 분산 클러스터 기반의 생활인구 빅데이터 마트 구축을 하고자 함
- 이를 통해 향후 빅데이터 분석시 전처리 및 통계분석에 소요되는 시간을 단축하여 이용자 편의성을 높이고자 함

분석 필요성 및 전략

- 빅데이터 분석 및 전처리 단계에서 대부분 Python을 활용하고 있음
- 그러나, Python의 메모리 관리와 쓰레드 안전성을 보장하기 위해 존재하는 GIL(Global Interpreter Lock)은 분석에 사용되는 서버 환경 즉, 멀티코어 CPU를 효율적으로 활용하는 데 제약을 줌
- 이 때문에 빅데이터 분석에 필수적인 병렬 분산처리를 위한 CPU 바운드 작업에서는 멀티프로세싱 기법을 적용하는 것이 필수적임
- 기존의 멀티프로세싱 방법론은 프로세스 간 데이터를 주고받기 위해서는 큐, 파이프 등의 IPC 메커니즘을 사용해야 하는데, 이는 구현이 복잡함과 동시에 프로세스 간의 상태 공유가 어렵고, 디버깅도 어려워지는 문제가 있음
- 특히 프로세스 간의 동기화 문제를 해결하는 것이 까다로워 일반적으로 활용하기엔 진입장벽이 높음
- 따라서, 대규모 데이터의 병렬 분산처리에 특화된 Python의 Dask 라이브러리를 활용하여 생활인구 빅데이터 마트 구축을 하고자 함

1) 계획인구 : 도시기본계획 수립 시, 해당 도시나 지역의 장래 인구를 예측하여 설정한 목표 인구 수

나.

데이터 분석 및 결과

활용 데이터

- 본 분석에서 활용된 데이터는 아래와 같이 구성됨
- 시간대, 행정동코드, X좌표, Y좌표, 주거인구, 직장인구, 방문인구, 소지역 코드 등이 담겨있으며, 좌표계는 5179 기반 좌표계로 이루어져 있음
- 데이터수집의 셀 단위를 확인할 수 있도록, 소지역 코드가 존재함
- 데이터 무결성 검증을 한 결과, 2021년 9월의 데이터에서 달성군 및 달서구 데이터가 누락된 것이 발견됨
- 또한, '소지역코드' 열에서 일부 데이터의 형식오류(데이터 인코딩 에러)가 발생하고 있어, 추후 원천데이터의 재검수를 통해 대체 혹은 삭제가 필요함

연번	데이터명	내용	시간 범위	데이터 규모
1	통신사 생활인구 데이터	시간대, 행정동코드, X좌표, Y좌표, 주거인구, 직장인구, 방문인구, 소지역 코드	2021년 1월 ~2024년 5월 (약 3년)	약 1억9천만 건

<표> 활용 데이터

<표> 일부 데이터 형식오류

데이터 전처리용 로컬 클러스터 구축

– Dask 라이브러리를 활용하여, CPU의 물리 코어 및 하이퍼쓰레딩이 지원되는 CPU의 경우, 쓰레드의 개수에 맞춰 데이터 병렬 분산처리를 위한 로컬 클러스터를 구축

<표> 로컬 클러스터 구축 현황

데이터 전처리

• 데이터 결측치 확인 후 데이터 전처리 작업 진행
  – 사전 확인된 데이터 형식오류 및 데이터 누락 외에는 결측치 발견되지 않음
  – 주거인구, 직장인구, 방문인구 컬럼별로 집계를 진행
  – 시간대별(0시~23시) 데이터를 모두 합한 'H_T_sum','W_T_sum','V_T_sum' 컬러 생성
  – 메모리 활용 최적화를 위해, 분석에 사용하지 않는 컬럼은 drop진행
  – 시계열 데이터열 str 형식에서 datetime 형식으로 형 변환
  – 추후 기간 별 분석이 용이하도록, 주·월·분기·년도별로 통계처리된 데이터 생성
  – 기간, 좌표를 중심으로 groupby 기접 적용 후 sum 연산 적용

• 작업 프로세스 확인
  – 내부적으로 데이터를 청크 단위로 나누어 처리함으로써 메모리 효율성을 극대화
  – 따라서, 메모리에 적재할 수 없는 대규모 데이터셋 처리에 특화되어 있음
  – 데이터 구조는 Pandas와 유사하지만, 계산 그래프를 기반으로 동적 태스크 스케줄링을 통해 계산 작업을 자동으로 병렬화하여 시스템 효율적으로 관리하여 전처리 진행

분석 결과

- 시간 범위별 데이터 생성
  - 추후 시간대 별 분석 및 다른 분석에도 활용할 수 있도록 시간대별 데이터 생성함

다. 결론

유용성 및 활용 전략

- 모듈화된 코드에 기반한 높은 재생산성
  - 모듈화 된 코드를 활용하여, 추후 목적에 맞춰 다른 통계분석이 진행할 수 있도록 코드 재생산성을 높임
  - Pandas 및 numpy 기반의 연산 및 통계처리는 모두 적용이 가능하도록 구현하였기 때문에, 추후 유사한 데이터 분석시에 활용 가능함
- 자원 활용성 극대화
  - 현재 빅데이터 활용센터가 보유하고 있는 서버 자원을 더욱 효율적으로 활용하여 대규모 데이터 분석에 수행되는 시간을 절감할 수 있음
- 높은 확장성을 활용한 데이터 전처리 API 구축
  - Dask는 로컬 환경에서의 병렬 처리뿐만 아니라, 대규모 클러스터 환경에서도 확장 가능하므로, Docker 혹은 Kubernetes와 같은 환경으로 전처리용 서버 배포시에도 Dask 클러스터를 쉽게 구성할 수 있음
  - 이를 활용하여, 추후 로컬 기반의 데이터 전처리 과정을 넘어선 API 형식의 데이터 전처리 파이프라인을 구축하여 효율적인 자원 관리 및 분석 서비스 제공이 가능함

Appendix

```
import numpy as np
import pandas as pd

import dask.array as da
import dask.bag as db

from dask.distributed import LocalCluster
import dask.dataframe as dd
import glob

import numpy as np
import dask.array as da

cluster = LocalCluster(n_workers=12, threads_per_worker=2)
client = cluster.get_client()

client

# 파일 경로 패턴 정의
file_pattern = '2021/01.서비스인구/*pcell_time_pop*.csv'
# glob을 사용하여 파일 패턴에 맞는 모든 파일 리스트 가져오기
file_list = glob.glob(file_pattern)
# 파일 경로 패턴 정의
file_pattern = '2022/01.서비스인구/*pcell_time_pop*.csv'
file_list.extend(glob.glob(file_pattern))
file_pattern = '2023/01.서비스인구/*/*pcell_time_pop*.csv'
file_list.extend(glob.glob(file_pattern))
file_pattern = '2024/01.서비스인구/*/*pcell_time_pop*.csv'
file_list.extend(glob.glob(file_pattern))
file_list.remove('2021/01.서비스인구\\daegu_service_pcell_time_pop_202109.csv')

len(file_list)

# Load data
df = dd.read_csv(file_list, delimiter='|', encoding='utf-8',blocksize='50MB')
df

# 전체 행의 수를 구합니다.
total_rows = df.isna.sum().compute()
```

```
# 결과 출력
print("전체 행의 수:", total_rows.compute())

#결측치 확인
df.isna().sum().compute()

# Generate columns lists
h_columns = [f'H_T_{str(i).zfill(2)}' for i in range(24)]
w_columns = [f'W_T_{str(i).zfill(2)}' for i in range(24)]
v_columns = [f'V_T_{str(i).zfill(2)}' for i in range(24)]

# Sum columns for each group
df['H_T_sum'] = df[h_columns].sum(axis=1)
df['W_T_sum'] = df[w_columns].sum(axis=1)
df['V_T_sum'] = df[v_columns].sum(axis=1)

# Drop original columns
df = df.drop(columns=h_columns + w_columns + v_columns)

# Convert STD_YMD to datetime
df['STD_YMD'] = dd.to_datetime(df['STD_YMD'], format='%Y%m%d')

# Function to group by period and save
def group_by_period_and_save(df, period, filename_prefix):
    period_df = df.copy()
    if period == 'week':
        period_df['period'] = period_df['STD_YMD'].dt.to_period('W').dt.start_time
    elif period == 'month':
        period_df['period'] = period_df['STD_YMD'].dt.to_period('M').dt.start_time
    elif period == 'quarter':
        period_df['period'] = period_df['STD_YMD'].dt.to_period('Q').dt.start_time
    elif period == 'year':
        period_df['period'] = period_df['STD_YMD'].dt.to_period('Y').dt.start_time
    else:
        raise ValueError("Invalid period. Choose from 'week', 'month', 'quarter', 'year'.")

    period_df['year'] = period_df['STD_YMD'].dt.year

    # Group by the period and other columns, then sum
    grouped_df = period_df.groupby(['period', 'HCODE','X_COORD', 'Y_COORD']).agg({
        'H_T_sum': 'sum',
        'W_T_sum': 'sum',
        'V_T_sum': 'sum'
    }).reset_index()
```

```
    return grouped_df

grouped_df = group_by_period_and_save(df, 'week', 'weekly_grouped')

# Example usage
group_by_period_and_save(df, 'week', 'weekly_grouped')
group_by_period_and_save(df, 'month', 'monthly_grouped')
group_by_period_and_save(df, 'quarter', 'quarterly_grouped')
group_by_period_and_save(df, 'year', 'yearly_grouped')

# 파일 무결성 검증
df = dd.read_csv('2021/01.서비스인구/daegu_service_pcell_time_pop_202109.csv',
delimiter='|', encoding='utf-8',blocksize='25MB')
df

df = dd.read_csv('2021/01.서비스인구/daegu_service_pcell_time_pop_202109.csv',
delimiter='|', encoding='utf-8',blocksize='25MB')
for i in range(52):
    try:
        df.partitions[i].compute()
    except:
        print(i)

file_path = '2021/01.서비스인구/daegu_service_pcell_time_pop_202110.csv'

# 775MB 이후 위치 계산 (바이트 단위)
start_position = 1200 * 1024 * 1024  # 775MB

# 파일의 특정 위치 이후부터 읽기
with open(file_path, 'rb') as file:
    file.seek(start_position)  # 파일 포인터를 775MB 위치로 이동
    raw_data = file.read()  # 그 위치부터 끝까지 읽기

# raw 데이터를 문자열로 변환
raw_data_str = raw_data.decode('utf-8')

print(raw_data_str[:])

df['HCODE'] = df['HCODE'] // 100000
df['HCODE'].unique().compute()

# 조건 필터링 및 계산
filtered_df = df[(df['HCODE'] // 100000 == 27710.0) | (df['HCODE'] // 100000 == 27290.0)]
result = filtered_df.compute()
```

접수번호 2024-10

계획인구 산정을 위한 대구시 생활권별 인구 통계

분석 요청자	소속	기업	성명	김OO
	휴대전화	-	전자우편	-
분석유형	■ 데이터 전처리	▢ 데이터 시각화	▢ 머신러닝	▢ 기타
데이터 분석가	김건욱 센터장, 김현정 사무원			
분석 기간	2024.08 ~ 2024.08(1개월)			

가.

분석 주제

분석 주제(요청내용)

- 도시기본계획 수립 과정에서 인구배분계획은 도시의 장기적인 발전을 위한 핵심 요소 중 하나임
- 도시기본계획은 각 생활권이나 구역별로 계획인구(상주인구, 주간인구 등)를 어떻게 분배할 것인지에 대한 전략을 수립하는 것을 의미하며, 이를 위해 통계청의 장래인구 추계와 지역 특성을 고려하여 인구를 추정하고 배분함
- 계획인구 산정을 위해 각 생활권의 특성과 인구 변동성을 체계적으로 파악하는 것이 필요하며, 도시 계획 및 정책 수립 과정에서 실질적인 데이터 기반을 제공하고, 장기적인 도시 발전 전략 수립에 기여할 수 있는 방안을 도출하고자 함
- 본 분석의 목적은 각 생활권별 인구 특성과 변동성을 체계적으로 파악하여 계획 인구 산정을 위한 기초 통계자료를 제공하는 데 있음. 이를 위해 SKT 통신사 생활인구 데이터를 활용하여 각 생활권별 전체 인구, 거주 인구, 방문 인구, 직장 인구의 월평균 증가율을 산출함

분석 필요성 및 전략

- 대구시의 계획인구 산정은 도시 발전과 인프라 구축에 있어 중요한 요소이며, 정확한 인구 추정은 사회적·경제적 파급 효과를 미리 예측하는 데 필수적
- 특히, 통신사 데이터를 활용한 생활인구 분석은 실시간으로 변동하는 인구의 흐름과 생활권별 특성을 파악할 수 있는 도구로, 기존의 통계청 장래인구 추정 방식보다 더 세밀하고 정확한 분석이 가능
- 계획 인구 산정에 근거가 되는 자료를 제시함으로써 도시 계획의 정확성을 높이고, 효율적인 자원 배분 및 인프라 구축 전략을 수립하는데 기여
- 본 분석에서는 SKT 생활인구 데이터 및 분석 의뢰인이 제공한 생활권 공간 벡터 데이터를 결합하여 사용
- 데이터 규모가 방대한 관계로 DASK 기반의 병렬 분산처리 분석 프로세스 환경에서 분석을 진행

나.

데이터 분석 및 결과

활용 데이터

– 본 분석에서 활용된 데이터는 아래와 같이 두 가지 주요 유형으로 구성

1) 시간대별 서비스 인구 데이터 : SKT 통신횟수 자료를 바탕으로 시간대별 거주 인구, 직장 인구, 방문 인구 정보를 포함하며, 2021년 1월부터 2024년 5월까지의 대구광역시 인구 데이터를 다룸

2) 생활권 공간벡터 데이터 : 대구광역시의 각 생활권의 위치 정보를 포함한 벡터 데이터로, 각 생활권을 구분하고 공간적으로 결합하는 데 활용됨

연번	데이터명	내용	시간 범위	데이터 규모
1	시간대별 서비스 인구 데이터	기준년월일, 행정동코드, X_좌표, Y_좌표, 시간대별 거주/직장/방문 인구	2021년 1월 ~2024년 5월 (약 3년)	190,282,340
2	생활권 공간벡터 데이터	생활권명, 위치 정보	-	12

<표> 활용 데이터

데이터 분석(전처리)

– 데이터 전처리 과정은 아래와 같음

1) 데이터 수집: 생활인구 데이터 중 시간대별 서비스 인구 데이터만 수집

```
# 파일 경로 패턴 정의
file_pattern = 'Data/2021/01.서비스인구/*pcell_time_pop*.csv'

# glob 모듈을 사용하여 파일 패턴과 일치하는 모든 파일 경로를 가져오기
file_list = glob.glob(file_pattern)

# 파일 경로 패턴 정의
file_pattern = 'Data/2022/01.서비스인구/*pcell_time_pop*.csv'
file_list.extend(glob.glob(file_pattern))
file_pattern = 'Data/2023/01.서비스인구/*/*pcell_time_pop*.csv'
file_list.extend(glob.glob(file_pattern))
file_pattern = 'Data/2024/01.서비스인구/*/*pcell_time_pop*.csv'
file_list.extend(glob.glob(file_pattern))
```

```
# 데이터를 로드, blocksize - Dask가 파일을 나누어 읽는 단위 설정
df = dd.read_csv(file_list, delimiter='|', encoding='utf-8',blocksize='500MB')
df = df.drop(axis=1,columns=['HCODE','BLOCK_CD'])
df
```

<표> 데이터 로드

2) 데이터 가공: 각 시간대별 서비스 인구 데이터의 통계량(중위값, 사분위 수 등)을 분석 한 후, 이상치로 판단되는 값은 극단값의 영향을 최소화하고 데이터의 분포를 유지하기 위해 중위값으로 대체함. 이후 이상치 처리된 데이터를 시공간 결합에 적합한 형태로 가공 후, 3차원 좌표를 2차원으로 변환

```
# 통계량 계산
stats = df[all_columns].describe().compute()

# 중위값으로 이상치 대체하는 함수 정의
def replace_outliers_with_median(df, columns, stats, multiplier=1.5):
    def replace_partition(part, stats, columns):
        for column in columns:
            # IQR(사분위 범위)를 이용하여 이상치의 하한선과 상한선을 계산
            Q1 = stats.loc['25%', column]
            Q3 = stats.loc['75%', column]
            IQR = Q3 - Q1
            lower_bound = Q1 - multiplier * IQR
            upper_bound = Q3 + multiplier * IQR
            median = stats.loc['50%', column]

            # 이상치 조건을 만족하는 경우 중위값을 대체
            condition = (part[column] < lower_bound) | (part[column] > upper_bound)
            part[column] = np.where(condition, median, part[column])

        return part

    # Dask DataFrame에서 각 파티션에 대해 이상치 대체 함수 적용
    df = df.map_partitions(replace_partition, stats=stats, columns=columns)

    return df


df = replace_outliers_with_median(df, all_columns, stats)
```

<표> 통계량 계산 및 이상치 처리

```
# 좌표 데이터를 사용하여 공간 데이터프레임으로 변환
gdf = gpd.read_file('[illegible].shp')
gdf = gdf.reset_index(names='idx')

# 3차원 좌표 데이터를 2차원으로 변환
def convert_polygon_z_to_2d(polygon_z):
    # 3차원 Polygon의 (x, y) 좌표만을 추출하여 2D Polygon 생성
    polygon_2d = Polygon([(x, y) for x, y, z in polygon_z.exterior.coords])
    return polygon_2d

gdf['geometry'] = gdf['geometry'].apply(convert_polygon_z_to_2d)
gdf = gdf.to_crs(epsg=5179)
gdf
```

<표> 3차원 좌표 데이터를 2차원 공간 데이터로 변환

3) 데이터 결합: 시간대별 서비스 인구 데이터를 생활권 공간 벡터 데이터와 결합, 각 좌표(X_COORD, Y_COORD)를 활용하여 좌표별로 생활권 (idx)를 할당한 후, 이를 주기별로 그룹화하여 인구 통계 분석을 위한 기본 데이터 생성

```
# 좌표를 사용하여 공간 데이터와 생활권 인덱스 할당
def assign_polygon_idx(df, gdf, x_col='X_COORD', y_col='Y_COORD'):

    # 좌표를 Point 객체로 변환하여 GeoSeries 생성
    points = gpd.GeoSeries([Point(xy) for xy in zip(df[x_col], df[y_col])])

    # 각 포인트가 포함된 폴리곤의 idx 값을 찾기
    def get_polygon_idx(point):
        for idx, polygon in gdf.iterrows():
            if polygon['geometry'].contains(point):
                return polygon['idx']
        return None  # 어떤 폴리곤에도 포함되지 않는 경우 None 반환

    df['idx'] = points.apply(get_polygon_idx)
    return df
```

```
# Dask DataFrame에서 각 좌표에 해당하는 폴리곤 인덱스를 할당
comp = df.map_partitions(assign_polygon_idx, gdf)
result_df = comp.compute()

# 주기 및 인덱스 별로로 그룹화하고 합계
grouped_df = result_df.groupby(['period', 'idx']).agg({
    'H_T_sum': 'sum',
    'W_T_sum': 'sum',
    'V_T_sum': 'sum'
}).reset_index()
```

<표> 결합 키 생성, 통계처리를 위한 데이터 도출

데이터 분석(통계 분석)

- 통계 분석 과정은 아래와 같음

  1) 데이터 수집: 전처리 결과 도출된 여러 연도의 데이터를 하나의 데이터 프레임으로 결합

  2) 데이터 결합: 결합키(idx)를 기준으로 데이터를 그룹화하여 거주 인구(H_T_sum), 직장 인구(W_T_sum), 방문 인구(V_T_sum)의 합계를 계산한 전체 인구(T_T_sum) 컬럼 생성

  3) 월 평균 증가율 계산: 각 생활권별로 월별 증가율의 평균을 구해 월평균 증가율을 산출. 이때, 전년 동월 대비 증가율을 사용하여 계절적 요인과 연간 주기의 영향을 배제함

다.

분석 활용 전략

결론

1) 결과

- 본 분석을 통해 도출된 대구시 각 생활권의 월평균 증가율은 거주 인구, 방문 인구, 직장 인구로 구분되어 산출됨. 이 데이터는 도시 계획 및 정책 수립에 중요한 근거자료로 활용될 수 있음
- 특히, 통신사 데이터를 활용하여 실시간 인구 변화와 이동 패턴을 포착함으로써, 기존 방식보다 더 정교하고 현실성 있는 계획인구 산정이 가능
- 또한, 각 생활권의 특성에 맞춘 맞춤형 계획이 가능하며 이를 통해 대구시의 도시 개발과 자원 배분 전략의 실효성을 높일 수 있음

2) 결과 해석 시 유의사항

- 2023년 9월의 달성군과 달서구 데이터 누락으로 인해, 해당 지역이 포함된 생활권의 인구 증가율이 왜곡되었을 가능성이 있음. 특히, 달성군과 달서구가 모두 포함된 생활권에서는 인구 증가율이 실제보다 과대 또는 과소평가되었을 가능성이 크므로, 이러한 생활권의 분석 결과는 신중하게 해석해야 함
- 또한, 군위군이 2023년 7월 1일 자로 대구광역시에 편입되었으며, 데이터에는 2023년 8월부터 반영되었음. 군위군이 포함된 생활권 경계가 다수 존재하여, 이번 분석에서는 군위군을 제외하지 않고 분석을 진행하였음. 따라서 군위군이 포함된 생활권의 분석 결과는 편입에 따른 인구 변동성을 충분히 고려하여 해석해야 함
- 결과적으로, 달성군, 달서구, 그리고 군위군과 관련된 생활권의 분석에서는 데이터 신뢰성을 확보하기 위해 추가적인 데이터 수집 및 보완 작업이 필요함. 이를 통해 계획 인구 산정의 정확성을 더욱 높일 수 있을 것으로 판단됨

구분	생활권명	시군구명
1	뉴k2생활권	군위군, 동구, 북구, 수성구
2	도심생활권	남구, 달서구, 달성군, 동구, 북구, 서구 ,수성구, 중구
3	동군위생활권	군위군, 동구
4	동대구생활권	동구, 북구, 수성구, 중구
5	서대구생활권	남구, 달서구, 북구, 서구
6	성서생활권	달서구, 달성군, 북구, 서구
7	수성생활권	남구, 달서구, 달성군, 동구, 수성구, 중구
8	신공항생활권	군위군
9	안심생활권	동구, 수성구
10	월배화원생활권	남구, 달서구, 달성군
11	칠곡생활권	달성군, 북구, 서구
12	현풍생활권	달성군

<표> 생활권별 시군구 구성

정책적 시사점

- 인구 증가율이 높은 생활권에서는 인프라 확충, 주거 공간 개발, 교통망 확장 등의 전략이 필요함. 특히, 거주 인구의 급증이 예상되는 지역은 이에 따른 공공 서비스 및 인프라의 사전 준비가 요구됨
- 인구 감소 또는 변동성이 큰 생활권에서는 인구 유출을 방지하기 위한 정책적 지원, 예를 들어 일자리 창출 및 청년층 유입을 유도할 수 있는 방안이 필요함. 또한, 이러한 지역의 경우 장기적인 인구 예측 모델을 활용한 정책적 대응이 필수적임

Appendix

1. 데이터 전처리

```
import numpy as np
import pandas as pd
import dask.array as da
import dask.bag as db
from dask.distributed import LocalCluster
import dask.dataframe as dd
import glob

# Dask를 활용한 데이터 불러오기 및 클러스터 설정
# Dask의 LocalCluster를 사용하여 클러스터 생성
cluster = LocalCluster(n_workers=14, threads_per_worker=2)

# 생성된 클러스터에 연결된 클라이언트 가져오기
client = cluster.get_client()

# 파일 경로 패턴 정의
file_pattern = 'Data/2021/01.서비스인구/*pcell_time_pop*.csv'

# glob 모듈을 사용하여 파일 패턴에 맞는 모든 파일 리스트 가져오기
file_list = glob.glob(file_pattern)

# 파일 경로 패턴 정의
file_pattern = 'Data/2022/01.서비스인구/*pcell_time_pop*.csv'
file_list.extend(glob.glob(file_pattern))
file_pattern = 'Data/2023/01.서비스인구/*/*pcell_time_pop*.csv'
file_list.extend(glob.glob(file_pattern))
file_pattern = 'Data/2024/01.서비스인구/*/*pcell_time_pop*.csv'
file_list.extend(glob.glob(file_pattern))

# 데이터 로드 'blocksize' = Dask가 파일을 나누어 읽는 단위 설정
df = dd.read_csv(file_list, delimiter='|', encoding='utf-8',blocksize='500MB')
df = df.drop(axis=1,columns=['HCODE','BLOCK_CD'])


#중위값으로 이상치 대체
# 컬럼 리스트 생성
h_columns = [f'H_T_{str(i).zfill(2)}' for i in range(24)]
w_columns = [f'W_T_{str(i).zfill(2)}' for i in range(24)]
v_columns = [f'V_T_{str(i).zfill(2)}' for i in range(24)]
```

```
# 모든 컬럼 리스트를 하나로 결합
all_columns = h_columns + w_columns + v_columns

# 통계량 계산
stats = df[all_columns].describe().compute()
stats.to_csv('D:/Project/stats.csv')
# 저장한 통계량을 pandas로 다시 로드
stats = pd.read_csv('D:/Project/stats.csv').set_index('Unnamed: 0')
stats

# 중위값으로 이상치 대체하는 함수 정의
def replace_outliers_with_median(df, columns, stats, multiplier=1.5):
    def replace_partition(part, stats, columns):
        for column in columns:
            # IQR(사분위 범위)를 이용하여 이상치의 하한선과 상한선을 계산
            Q1 = stats.loc['25%', column]
            Q3 = stats.loc['75%', column]
            IQR = Q3 - Q1
            lower_bound = Q1 - multiplier * IQR
            upper_bound = Q3 + multiplier * IQR
            median = stats.loc['50%', column]

            # 이상치 조건을 만족하는 경우 중위값을 대체
            condition = (part[column] < lower_bound) | (part[column] > upper_bound)
            part[column] = np.where(condition, median, part[column])

        return part

    # Dask DataFrame에서 각 파티션에 대해 이상치 대체 함수 적용
    df = df.map_partitions(replace_partition, stats=stats, columns=columns)

    return df

df = replace_outliers_with_median(df, all_columns, stats)


# 주기별로 파일 그룹화 및 저장
# 각 컬럼에 대해 집계
df['H_T_sum'] = df[h_columns].sum(axis=1)
df['W_T_sum'] = df[w_columns].sum(axis=1)
df['V_T_sum'] = df[v_columns].sum(axis=1)

# 원본 컬럼 제거
df = df.drop(columns=h_columns + w_columns + v_columns)
```

```
# STD_YMD 컬럼 datetime 형식으로 변환
df['STD_YMD'] = dd.to_datetime(df['STD_YMD'], format='%Y%m%d')

# 주기별로 그룹화하여 결과를 저장하는 함수 정의
def group_by_period_and_save(df, period, filename_prefix):
    period_df = df.copy()
    if period == 'week':
        period_df['period'] = period_df['STD_YMD'].dt.to_period('W').dt.start_time
    elif period == 'month':
        period_df['period'] = period_df['STD_YMD'].dt.to_period('M').dt.start_time
    elif period == 'quarter':
        period_df['period'] = period_df['STD_YMD'].dt.to_period('Q').dt.start_time
    elif period == 'year':
        period_df['period'] = period_df['STD_YMD'].dt.to_period('Y').dt.start_time
    else:
        raise ValueError("Invalid period. Choose from 'week', 'month', 'quarter', 'year'.")

    period_df['year'] = period_df['STD_YMD'].dt.year

    # 주기 및 기타 컬럼별로 그룹화하고 집계
    grouped_df = period_df.groupby(['period', 'X_COORD', 'Y_COORD']).agg({
        'H_T_sum': 'sum',
        'W_T_sum': 'sum',
        'V_T_sum': 'sum'
    }).reset_index()


    # 추가 처리를 위해 pandas 데이터 프레임으로 변환
    grouped_df = grouped_df.compute()

    # 주기별로 결과를 파일로 저장
    if period in ['week', 'month']:
        # 연도별로 나누어 저장
        years = grouped_df['period'].dt.year.unique()
        for year in years:
            year_df = grouped_df[grouped_df['period'].dt.year == year]
            year_df.to_csv(f'{filename_prefix}_{year}.csv', encoding='utf-8', index=False)
            print(f"결과가 '{filename_prefix}_{year}.csv' 파일에 저장되었습니다.")
    else:
        # 전체 데이터프레임을 하나의 파일로 저장
        grouped_df.to_csv(f'{filename_prefix}.csv', encoding='utf-8', index=False)
        print(f"결과가 '{filename_prefix}.csv' 파일에 저장되었습니다.")
```

2. 공간 데이터 결합

```
# 공간 데이터 결합을 위한 라이브러리 불러오기
import geopandas as gpd
from shapely.geometry import Point
from shapely.geometry import Polygon

# Dask의 LocalCluster를 사용하여 클러스터 생성
cluster = LocalCluster(n_workers=14, threads_per_worker=2)

# 생성된 클러스터에 연결된 클라이언트 가져오기
client = cluster.get_client()

# 파일 경로 패턴 정의
file_pattern = 'result_data/monthly_grouped(coord)_*.csv'

# glob 모듈을 사용하여 파일 패턴에 맞는 모든 파일 리스트 가져오기
file_list = glob.glob(file_pattern)

# 데이터 로드 'blocksize' = Dask가 파일을 나누어 읽는 단위 설정
df = dd.read_csv(file_list[3], encoding='utf-8',blocksize='3MB')

# 생활권 경계 공간 데이터를 읽어와 GeoDataFrame(gdf)으로 저장
gdf = gpd.read_file('유성\대구광역시생활권shp.shp')
# 인덱스를 재설정하여 'idx'라는 이름으로 인덱스 컬럼을 추가
gdf = gdf.reset_index(names='idx')
# POLYGON의 3차원 좌표를 2차원 좌표로 변환하는 함수 정의
def convert_polygon_z_to_2d(polygon_z):
    # 3차원 POLYGON의 (x, y) 좌표만을 추출하여 2D POLYGON 생성
    polygon_2d = Polygon([(x, y) for x, y, z in polygon_z.exterior.coords])
    return polygon_2d
# GeoDataFrame(gdf)의 'geometry' 컬럼에 있는 3차원 POLYGON 데이터를 2차원 POLYGON
으로 변환
gdf['geometry'] = gdf['geometry'].apply(convert_polygon_z_to_2d)
# 좌표계를 5179좌표계로 설정
gdf = gdf.to_crs(epsg=5179)

# 공간 데이터 결합 파일 저장
gdf.to_csv('gdf.csv',index=False)

# DataFrame 내 좌표들이 포함된 폴리곤의 idx 값을 새로운 열로 추가하는 함수
def assign_polygon_idx(df, gdf, x_col='X_COORD', y_col='Y_COORD'):
    # 좌표를 Point 객체로 변환하여 GeoSeries 생성
    points = gpd.GeoSeries([Point(xy) for xy in zip(df[x_col], df[y_col])])
```

```
    # 각 포인트가 포함된 폴리곤의 idx 값을 찾기
    def get_polygon_idx(point):
        for idx, polygon in gdf.iterrows():
            if polygon['geometry'].contains(point):
                return polygon['idx']
        return None  # 어떤 폴리곤에도 포함되지 않는 경우 None 반환

    df['idx'] = points.apply(get_polygon_idx)
    return df

# 각 데이터 파티션(df)마다 gdf의 폴리곤(생활권 경계)에 해당하는 인덱스를 할당하는 함수(assign_
polygon_idx)를 적용
comp = df.map_partitions(assign_polygon_idx, gdf)
# 모든 파티션에서의 연산을 병렬로 처리한 결과(comp)를 한 번에 모아 하나의 최종 DataFrame으
로 변환
result_df = comp.compute()



# 주기 및 인덱스 컬럼별로 그룹화하고 집계
grouped_df = result_df.groupby(['period', 'idx']).agg({
    'H_T_sum': 'sum',
    'W_T_sum': 'sum',
    'V_T_sum': 'sum'
}).reset_index()


3. 생활권별 인구 통계 분석
# CSV 파일 경로 리스트
file_paths = [
    'D:/Project/2021_idx_생활권.csv',
    'D:/Project/2022_idx_생활권.csv',
    'D:/Project/2023_idx_생활권.csv',
    'D:/Project/2024_idx_생활권.csv'
]

# CSV 파일을 하나의 DataFrame으로 읽기
df_list = [pd.read_csv(file) for file in file_paths]
df = pd.concat(df_list, ignore_index=True)

# 'period' 컬럼을 datetime형식으로 변환하고, 연도, 월 컬럼 추가
df['period'] = pd.to_datetime(df['period'], format='%Y-%m-%d')
df['year'] = df['period'].dt.year
df['month'] = df['period'].dt.month
df = df.iloc[:,1:]
```

```
# 공간 데이터 결합 파일 불러오기
gdf = pd.read_csv('D:/Project/gdf.csv')

# 인덱스, 연도, 월 기준으로 groupby
temp = df.groupby(['idx','year','month']).sum().reset_index()

# 필요한 컬럼만 남김
gdf = gdf[['idx','Layer']]

# idx 기준으로 temp와 gdf 데이터 병합
df = pd.merge(gdf, temp, on='idx',how='left')
df = df.drop('idx', axis=1)

# 전체인구집계 컬럼 생성 (거주+직장+방문 인구 합계)
df.loc[:, 'T_T_sum'] = df.loc[:, 'H_T_sum':'V_T_sum'].sum(axis=1)

# 연도, 월, layer 기준으로 그룹화하고 합계 계산
grouped_df = df.groupby(['year', 'month', 'Layer']).agg({
    'H_T_sum': 'sum',
    'W_T_sum': 'sum',
    'V_T_sum': 'sum',
    'T_T_sum': 'sum',
}).reset_index()

# pct_change 전에 데이터 정렬 (옵션)
grouped_df = grouped_df.sort_values(by=['year', 'month', 'Layer'])
# 증가율 컬럼 생성
grouped_df['거주인구증가율(%)'] = grouped_df.groupby(['Layer','month'])['H_T_sum'].pct_
change() * 100
grouped_df['직장인구증가율(%)'] = grouped_df.groupby(['Layer','month'])['W_T_sum'].pct_
change() * 100
grouped_df['방문인구증가율(%)'] = grouped_df.groupby(['Layer','month'])['V_T_sum'].pct_
change() * 100
grouped_df['전체인구증가율(%)'] = grouped_df.groupby(['Layer','month'])['T_T_sum'].pct_
change() * 100

# 컬럼 이름 변경
grouped_df = grouped_df.rename(columns={
    'year':'연도',
    'month':'월',
    'Layer':'생활권명',
    'H_T_sum':'거주인구집계',
    'W_T_sum':'직장인구집계',
    'V_T_sum':'방문인구집계',
    'T_T_sum': '전체인구집계'
```

```
})

# 소수점 처리
for col in grouped_df.columns:
    if col != '생활권명':
        if '증가율' in col:
            grouped_df[col] = grouped_df[col].apply(lambda x: round(x, 1) if pd.notnull(x) else x)
        else:
            grouped_df[col] = grouped_df[col].apply(lambda x: int(x) if pd.notnull(x) and
isinstance(x, (int, float)) else x)

# 피봇테이블 생성
fin = pd.pivot_table(grouped_df,
                     index=['생활권명', '월'],
                     columns='연도',
                     values=['거주인구집계', '직장인구집계', '방문인구집계', '전체인구집계', '거주인구증가율
(%)', '직장인구증가율(%)', '방문인구증가율(%)', '전체인구증가율(%)'])

# 컬럼 레벨 변경 및 정렬
fin = fin.swaplevel(axis=1).sort_index(axis=1, level=0)

# 원하는 순서로 컬럼 재정렬
desired_order = ['거주인구집계', '직장인구집계', '방문인구집계', '전체인구집계', '거주인구증가율
(%)', '직장인구증가율(%)', '방문인구증가율(%)', '전체인구증가율(%)']

# 각 연도별로 컬럼 순서 재정렬
ordered_columns = []
for year in sorted(fin.columns.levels[0]):
    for col in desired_order:
        ordered_columns.append((year, col))

fin = fin.reindex(columns=pd.MultiIndex.from_tuples(ordered_columns))

# 컬럼 이름 복원
fin.columns.names = ['연도', '지표']

# 집계 컬럼들에 3자리마다 콤마 추가 및 NaN 값을 '-'로 대체
def format_values(x, col_name):
    if pd.isna(x):
        return '-'
    elif '증가율' in col_name[1]:
        return f"{x:.1f}"  # 증가율 컬럼은 소수점 유지
    else:
        return f"{int(x):,}"  # 나머지 컬럼은 소수점 제거 및 콤마 추가
```

```
fin = fin.apply(lambda col: col.map(lambda x: format_values(x, col.name)))

# 월평균 증가율을 계산하기 위해 '생활권', '시군구명', '연도'별로 월별 증가율 평균 계산
monthly_avg_growth = grouped_df.groupby(['생활권명','연도', '월']).agg({
   '거주인구증가율(%)': 'mean',
   '직장인구증가율(%)': 'mean',
   '방문인구증가율(%)': 'mean',
   '전체인구증가율(%)': 'mean',
}).reset_index()

# 피봇테이블 생성하여 '생활권', '시군구명' 및 연도별로 월평균 증가율을 정리
monthly_avg_pivot = pd.pivot_table(monthly_avg_growth,
                        index='생활권명',
                        columns='연도',
                          values=['거주인구증가율(%)', '직장인구증가율(%)', '방문인구증가율(%)', '전
체인구증가율(%)'],
                        aggfunc='mean')

# 컬럼 레벨 정리 및 이름 지정
monthly_avg_pivot.columns.names = ['지표', '연도']

# 월평균 증가율 테이블 값 형식 지정
monthly_avg_pivot = monthly_avg_pivot.apply(lambda col: col.map(lambda x: f"{x:.1f}" if
pd.notnull(x) else '-'))

monthly_avg_pivot

# 결과 피봇 테이블 저장
file_path = 'D:/Project/result_data/유성이앤씨_생활권.xlsx'
monthly_avg_pivot.to_excel(file_path, index=True)
```

접수번호 2024-11

건강보험공단 건강검진정보 데이터 전처리

분석 요청자	소속	기관	성명	오○○
	휴대전화	-	전자우편	-
분석유형	■ 데이터 전처리	▢ 데이터 시각화	▢ 머신러닝	▢ 기타
데이터 분석가	김건욱 센터장, 윤주혁 사무원			
분석 기간	2024.10 ~ 2024.10(1개월)			

가.

분석 주제

분석 주제(요청내용)

- 흡연은 다양한 건강 문제의 주요 원인으로 알려져 있으며, 폐암, 심혈관 질환, 만성 폐쇄성 폐질환(COPD) 등 여러 질환의 발생 위험을 높이고 전반적인 건강 상태와 삶의 질에 부정적인 영향을 미침
- 국민건강보험공단은 공공데이터 포털(www.data.go.kr)을 통해 국민 건강검진 정보를 제공하고 있으며, 이 데이터는 해당 연도에 건강검진을 받은 국민건강보험 가입자 100만 명의 기본 정보(시도 코드, 성별, 연령대 등)와 건강검진 내역(신장, 체중, 혈압, 혈당, 총콜레스테롤, 혈색소 등)으로 구성되어 있음
- 데이터는 직장 가입자와 20세 이상의 피부양자, 세대주인 지역 가입자, 20세 이상의 지역 가입자 중 일반 건강검진 이력이 있는 국민을 대상으로 연도별로 100만 명을 무작위로 선정하여 공개된 자료임
- 국민건강보험공단에서 매년 공개하는 이 건강검진 데이터를 활용하여 흡연 여부와 다양한 건강 지표 간의 통계적 상관관계를 시각화하기 위해 기초적인 데이터 전처리 작업이 필요함
- 전처리된 데이터셋을 활용하여 다양한 시각화 기법을 적용함으로써, 흡연과 건강 지표 간의 관계를 보다 명확히 분석하고자 함

분석 필요성 및 전략

- 건강보험공단에서 제공하는 5개년치(500만 건)의 원시 데이터를 시각화 용도로 활용하기 위해서는 데이터 전처리가 필수적임
- 원시 데이터에는 결측치, 이상치, 중복 데이터 등이 포함될 수 있으며, 이러한 요소들은 데이터 분석 및 시각화 결과의 신뢰성을 저해할 수 있음. 따라서 데이터의 품질을 높이기 위한 전처리 과정이 요구됨
- 전처리 과정에서는 먼저 결측치와 이상치를 식별하여 처리하는 작업이 필요함. 특히, 결측치가 많은 변수는 분석에 포함할지 신중히 결정해야 하며, 필요한 경우 중위수 대체, 머신러닝 기법 등의 데이터 보간 작업을 적용할 수 있음
- 또한, 변수 간 상관관계를 고려하여 파생 변수를 생성함으로써 분석의 깊이를 더할 수 있음. 이러한 전처리 과정을 거쳐 흡연이 건강에 미치는 영향을 시각화하는 데 필요한 기초 자료를 마련하고자 함

나.

데이터 분석 및 결과

활용 데이터

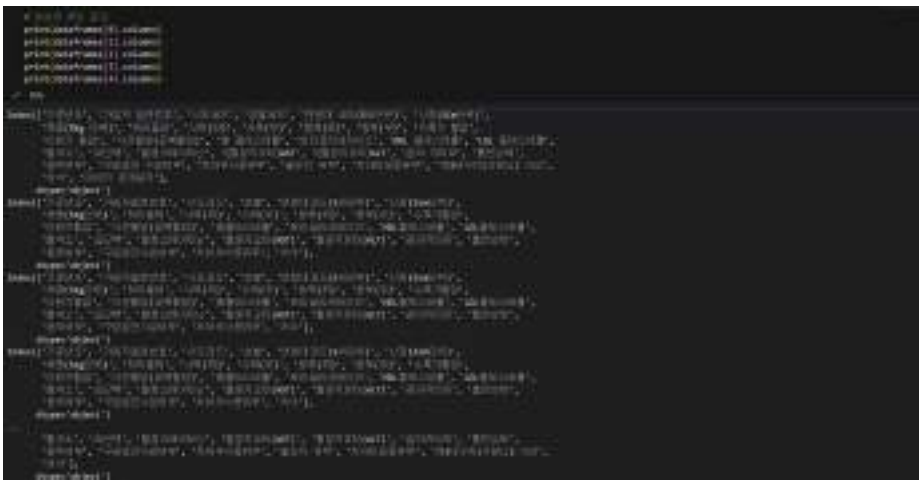
- 본 분석에 활용된 데이터는 국민건강보험공단이 제공하는 국민건강검진 자료로, 매년 건강검진을 받은 국민건강보험 가입자 중 100만 명을 무작위로 추출하여 구성된 표본임
- 이 데이터에는 기본 인적 사항인 연령대, 성별, 거주지 시도코드와 더불어 신체 측정값(신장, 체중, 허리둘레), 병리 검사 결과(혈압, 혈당, 콜레스테롤, 혈색소 등), 문진 정보(흡연 여부, 음주 여부 등)가 포함되어 있음
- 데이터는 개인정보 보호를 위해 비식별화 및 범주화 과정을 거쳤으며, 연령대는 5세 단위로 그룹화되고, 지역 정보는 시도 단위로 표시됨. 또한, 민감 질환 정보는 대분류 코드화하여 보호함

연번	데이터명	내용	시간 범위	데이터 규모	출처
1	건강검진정보	기준년도, 식별코드, 신장, 체중, 허리둘레, 시력, 체중, 수축기 혈압 등	2019 ~ 2023년	500만건	국민건강보험공단

<표> 활용 데이터

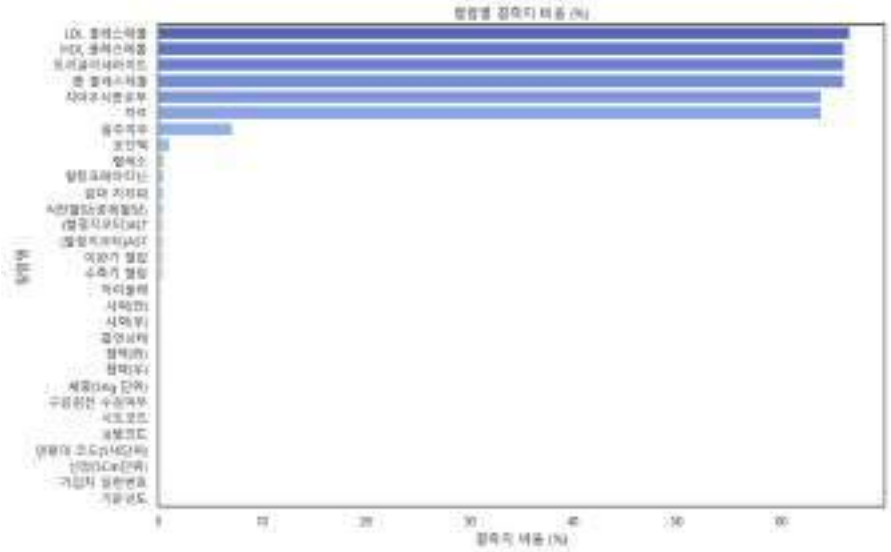
데이터 전처리

- 연도별 데이터 컬럼 통합
  - - 연도별 데이터를 파악해본 결과 컬럼 명 및 컬럼 개수가 일치하지 않은 것을 발견함. 컬럼명 통일 및 유효하지 않은 컬럼은 drop을 통해 제거함



<표> 19~23년 건강검진 데이터 컬럼

• 컬럼별 결측치 비율



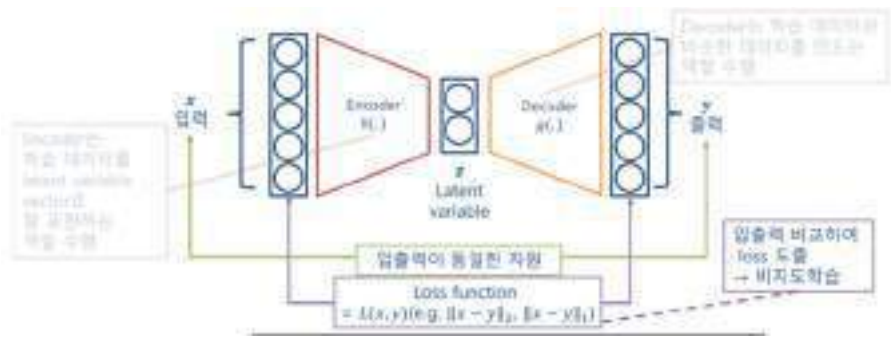
<표> 컬럼별 결측치 비율

– **분석에 불필요한 변수 제거**: 치석 및 치아 우식증은 데이터 분포와 흡연과의 영향성을 고려했을 때 시각화 목적에 부합하지 않는 변수로 판단하여 제거함

– **VAE(Variational Autoencoder)를 활용한 변수 특성 및 분포 학습**: 주요 수치형 변수인 허리둘레, 수축기 혈압, 이완기 혈압, 식전혈당(공복혈당), 총 콜레스테롤, 트리글리세라이드, HDL 콜레스테롤, LDL 콜레스테롤, 혈색소, 요단백, 혈청크레아티닌, 혈청 지오티(AST), 혈청 지피티(ALT), 감마 지티피는 분석에 필수적임. 이에 따라, 이상치가 없는 데이터를 사용해 VAE를 학습시켜 이러한 변수의 특성과 분포를 반영함

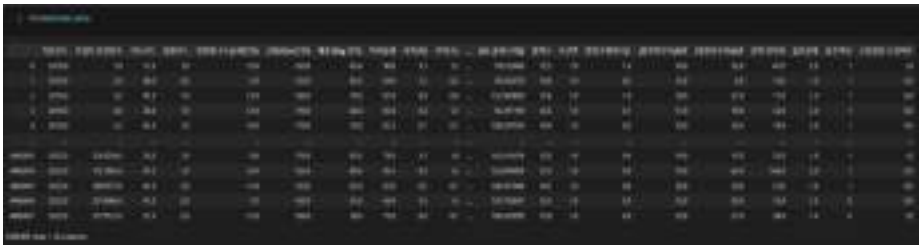
– **범주형 결측치 대체**: 범주형 변수인 흡연 상태와 음주 여부도 VAE 학습 데이터에 포함하여 데이터 특성 및 분포를 학습함. 이를 통해 결측치를 제거하는 대신, 변수 분포를 최대한 유지하면서 결측치 보간 작업을 수행함

• VAE를 활용한 결측치 대체작업



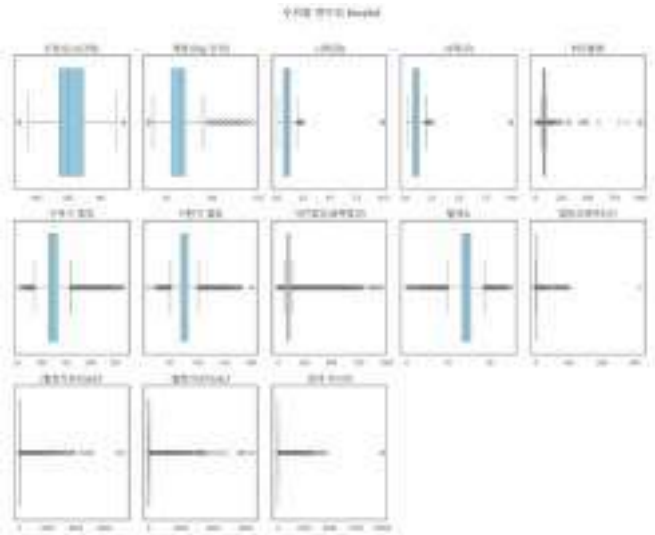
<표> VAE(Variational Auto Encoder) 구조

- VAE는 머신러닝의 비지도 학습 방법 중 하나로, 주어진 데이터의 잠재 표현을 학습하여 결측치를 보완하는 데 효과적으로 활용할 수 있음
- VAE 모델을 활용해 결측치가 없는 원본 데이터셋의 특성과 분포를 학습하도록 함
- 이후, 결측값이 포함된 데이터를 VAE에 입력하여 잠재 공간에서 복원된 값을 결측치 대체에 사용함. 이를 통해 원본 데이터의 분포를 최대한 유지하면서 결측치를 보완함



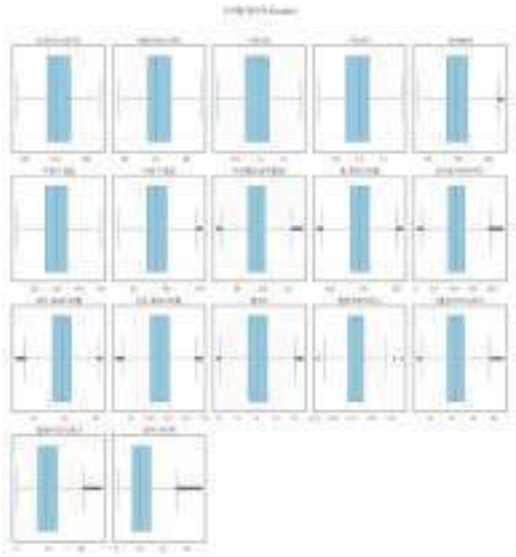
<표> 결측치가 일부 포함된 데이터 보간 결과

• 이상치 파악 및 처리
  - 결측치 보완 작업 후 Boxplot을 통해 대부분의 변수에서 이상치가 다수 존재함을 확인함
  - 특히, 혈압과 콜레스테롤 관련 변수, 혈청지오티(AST), 혈청지피티(ALT), 감마 지티피에서 이상치가 많이 나타남
  - 혈압(수축기, 이완기 혈압): 정상 범위를 벗어난 높은 값이 많으며, 고혈압 환자가 포함된 분포로 추정됨
  - 또한, 식전혈당(공복혈당)과 간 기능 지표((혈청지오티)AST, (혈청지오티)ALT, 감마 지티피)에서 높은 값의 이상치가 다수 발견되었으며, 혈색소와 혈청크레아티닌 등에서도 상당한 이상치가 나타남



<표> Boxplot을 통한 수치형 데이터의 이상치 파악

- 현재 데이터로는 시각화에 어려움이 있어 이상치 제거 작업을 수행하고자 함
- 이에 따라 IQR 기반의 이상치 처리 기법을 적용하였으며, 처리 후 결과 데이터 분포는 아래와 같음



<표> 전처리 완료 데이터

• 최종 데이터 세트 생성

- 데이터 전처리 및 이상치 처리 과정을 통해 최종 산출된 데이터는 총 3,069,946건 임

	기준년도	가입자 일련번호	시도코드	성별코드	연령대 코드(5세단위)	신장(5Cm단위)	체중(5Kg 단위)	허리둘레	시력(좌)	시력(우)	...	LDL 콜레스테롤	혈색소	요단백	혈청크레아티닌	(혈청지오티)AST	(혈청지오티)ALT	감마 지티피	흡연상태	음주여부	구강검진 수검여부
0	2019.0	7.0	41.0	1.0	6.0	175.0	70.0	81.3	0.9	1.0	...	93.000000	13.1	1.0	0.9	14.0	11.0	30.0	2.0	1.0	1.0
1	2019.0	27.0	46.0	1.0	10.0	170.0	75.0	83.0	0.5	0.7	...	123.000000	14.4	1.0	0.8	26.0	41.0	32.0	2.0	1.0	1.0
2	2019.0	42.0	47.0	1.0	5.0	175.0	90.0	93.0	1.2	1.2	...	117.000000	15.1	1.0	0.7	29.0	37.0	26.0	1.0	1.0	1.0
3	2019.0	64.0	41.0	2.0	10.0	155.0	55.0	74.9	1.2	1.2	...	97.000000	12.9	1.0	0.9	17.0	9.0	8.0	1.0	1.0	1.0
4	2019.0	101.0	11.0	1.0	13.0	165.0	60.0	80.8	0.9	0.8	...	103.000000	15.0	1.0	0.9	10.0	7.0	20.0	2.0	1.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3069941	2023.0	4088314.0	28.0	2.0	13.0	145.0	50.0	79.0	0.9	1.0	...	113.000000	13.7	1.0	0.8	30.0	23.0	24.0	1.0	0.0	0.0
3069942	2023.0	3833709.0	41.0	1.0	6.0	180.0	75.0	77.0	0.8	0.8	...	138.947026	16.9	1.0	1.1	30.0	20.0	16.0	1.0	1.0	1.0
3069943	2023.0	3265054.0	26.0	1.0	8.0	170.0	65.0	78.0	1.2	1.0	...	145.416474	13.6	1.0	0.9	19.0	13.0	22.0	1.0	1.0	1.0
3069944	2023.0	2618086.0	41.0	2.0	7.0	160.0	55.0	69.0	1.0	1.2	...	120.780607	13.5	1.0	0.9	22.0	20.0	16.0	1.0	0.0	0.0
3069945	2023.0	1279122.0	31.0	2.0	12.0	160.0	50.0	75.0	0.8	0.7	...	106.505899	13.0	1.0	0.8	29.0	21.0	34.0	1.0	0.0	1.0

3069946 rows × 28 columns

<표> 전처리 완료 데이터

다.

분석 활용 전략

결론

- 데이터 특성에 따른 결측치 보완 및 이상치 제거 전략
  - 본 데이터는 고혈압, 당뇨, 간 질환, 신장 질환 등 다양한 질환을 가진 사람들의 정보를 포함한 건강 검진 데이터로, 다양한 건강 상태를 반영하고 있음
  - 결측치가 일부 포함된 데이터가 다량 존재하여, 결측치가 없는 원본 데이터의 통계적 분포를 머신러닝 기법을 통해 학습하여 결측치를 보완함
  - 데이터 보완 작업 이후에도 이상치가 다수 존재하여, IQR 기반의 이상치 제거 작업을 수행함으로써 총 3,069,946건의 데이터 세트를 생성함
- 향후 분석 활용 방안
  - 변수 간 상관관계 분석을 통해 흡연, 음주, 연령대 등 주요 변수들이 건강 지표에 미치는 영향을 파악할 수 있음
  - 특히, 흡연 상태와 혈압, 간 기능 수치 등과의 관계를 시각화하여 흡연이 건강에 미치는 영향을 확인할 수 있을 것으로 기대됨

Appendix

```
import os
import re
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Seaborn 스타일 설정
sns.set_theme(style="whitegrid", palette="pastel")

plt.rc("font", family = "NanumGothic")
sns.set_theme(font="NanumGothic",
rc={"axes.unicode_minus":False}, style='white')

# 현재 경로에서 모든 CSV 파일을 찾아서 DataFrame으로 읽은 후, 하나의 DataFrame으로 연결
csv_files = [file for file in os.listdir() if file.endswith('.CSV')]
dataframes = [pd.read_csv(file,encoding='cp949') for file in csv_files]
# 컬럼명 통일 필요
print(dataframes[0].columns)
print(dataframes[1].columns)
print(dataframes[2].columns)
print(dataframes[3].columns)
print(dataframes[4].columns)
dataframes[0] = dataframes[0].drop(columns=['결손치 유무','치아마모증유무','제3대구치(사랑니) 이상','데이터 공개일자'])
dataframes[0].columns = ['기준년도', '가입자 일련번호', '시도코드', '성별코드', '연령대 코드(5세 단위)', '신장(5Cm단위)',
    '체중(5Kg 단위)', '허리둘레', '시력(좌)', '시력(우)', '청력(좌)', '청력(우)', '수축기 혈압',
    '이완기 혈압', '식전혈당(공복혈당)', '총 콜레스테롤', '트리글리세라이드', 'HDL 콜레스테롤', 'LDL 콜레스테롤',
     '혈색소', '요단백', '혈청크레아티닌', '(혈청지오티)AST', '(혈청지오티)ALT', '감마 지티피', '흡연 상태',
    '음주여부', '구강검진 수검여부', '치아우식증유무', '치석']

dataframes[1].columns = ['기준년도', '가입자 일련번호', '시도코드', '성별코드', '연령대 코드(5세단위)', '신장(5Cm단위)',
    '체중(5Kg 단위)', '허리둘레', '시력(좌)', '시력(우)', '청력(좌)', '청력(우)', '수축기 혈압',
    '이완기 혈압', '식전혈당(공복혈당)', '총 콜레스테롤', '트리글리세라이드', 'HDL 콜레스테롤', 'LDL 콜레스테롤',
     '혈색소', '요단백', '혈청크레아티닌', '(혈청지오티)AST', '(혈청지오티)ALT', '감마 지티피', '흡연 상태',
    '음주여부', '구강검진 수검여부', '치아우식증유무', '치석']
```

```
dataframes[2].columns = ['기준년도', '가입자 일련번호', '시도코드', '성별코드', '연령대 코드(5세 단위)', '신장(5Cm단위)',
    '체중(5Kg 단위)', '허리둘레', '시력(좌)', '시력(우)', '청력(좌)', '청력(우)', '수축기 혈압',
    '이완기 혈압', '식전혈당(공복혈당)', '총 콜레스테롤', '트리글리세라이드', 'HDL 콜레스테롤', 'LDL 콜레스테롤',
     '혈색소', '요단백', '혈청크레아티닌', '(혈청지오티)AST', '(혈청지오티)ALT', '감마 지티피', '흡연 상태',
    '음주여부', '구강검진 수검여부', '치아우식증유무', '치석']
dataframes[3].columns = ['기준년도', '가입자 일련번호', '시도코드', '성별코드', '연령대 코드(5세 단위)', '신장(5Cm단위)',
    '체중(5Kg 단위)', '허리둘레', '시력(좌)', '시력(우)', '청력(좌)', '청력(우)', '수축기 혈압',
    '이완기 혈압', '식전혈당(공복혈당)', '총 콜레스테롤', '트리글리세라이드', 'HDL 콜레스테롤', 'LDL 콜레스테롤',
     '혈색소', '요단백', '혈청크레아티닌', '(혈청지오티)AST', '(혈청지오티)ALT', '감마 지티피', '흡연 상태',
    '음주여부', '구강검진 수검여부', '치아우식증유무', '치석']
dataframes[4] = dataframes[4].drop(columns=['결손치 유무','치아마모증유무','제3대구치(사랑니) 이상'])
dataframes[4].columns = ['기준년도', '가입자 일련번호', '시도코드', '성별코드', '연령대 코드(5세 단위)', '신장(5Cm단위)',
    '체중(5Kg 단위)', '허리둘레', '시력(좌)', '시력(우)', '청력(좌)', '청력(우)', '수축기 혈압',
    '이완기 혈압', '식전혈당(공복혈당)', '총 콜레스테롤', '트리글리세라이드', 'HDL 콜레스테롤', 'LDL 콜레스테롤',
     '혈색소', '요단백', '혈청크레아티닌', '(혈청지오티)AST', '(혈청지오티)ALT', '감마 지티피', '흡연 상태',
    '음주여부', '구강검진 수검여부', '치아우식증유무', '치석']

# dataframes 리스트에 있는 모든 데이터프레임을 하나로 결합
merged_df = pd.concat(dataframes, axis=0, ignore_index=True)
merged_df

# 2. 각 열의 결측치 비율 파악 및 시각화
plt.figure(figsize=(12, 8))
missing_ratio = merged_df.isnull().mean().sort_values(ascending=False) * 100
sns.barplot(y=missing_ratio.index, x=missing_ratio, palette="coolwarm")
plt.title('컬럼별 결측치 비율 (%)')
plt.xlabel('결측치 비율 (%)')
plt.ylabel('컬럼명')
plt.show()
import os
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, Dataset
import numpy as np
```

```
import pandas as pd
from tqdm import tqdm

# CUDA 디버깅을 위한 환경 변수 설정
os.environ["CUDA_LAUNCH_BLOCKING"] = "1"

# GPU 설정
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# VAE 모델 정의
class VAE(nn.Module):
    def __init__(self, input_dim, hidden_dim, latent_dim):
        super(VAE, self).__init__()
        self.fc1 = nn.Linear(input_dim, hidden_dim)
        self.fc21 = nn.Linear(hidden_dim, latent_dim)  # 잠재 공간의 평균
        self.fc22 = nn.Linear(hidden_dim, latent_dim)  # 잠재 공간의 표준편차
        self.fc3 = nn.Linear(latent_dim, hidden_dim)
        self.fc4 = nn.Linear(hidden_dim, input_dim)

    def encode(self, x):
        h1 = torch.relu(self.fc1(x))
        return self.fc21(h1), self.fc22(h1)

    def reparameterize(self, mu, logvar):
        std = torch.exp(0.5 * logvar)
        eps = torch.randn_like(std)
        return mu + eps * std

    def decode(self, z):
        h3 = torch.relu(self.fc3(z))
        return self.fc4(h3)

    def forward(self, x):
        mu, logvar = self.encode(x)
        z = self.reparameterize(mu, logvar)
        return self.decode(z), mu, logvar

# MSE 손실 함수 정의 (NaN 방지용)
def vae_loss_mse(recon_x, x, mu, logvar):
    MSE = nn.functional.mse_loss(recon_x, x, reduction='sum')
    # 안정성을 위해 logvar.clamp(min=-20) 추가
   KLD = -0.5 * torch.sum(1 + logvar.clamp(min=-20) - mu.pow(2) - logvar.clamp(min=-20).
exp())
    return MSE + KLD
```

```
# 결측값이 없는 데이터 학습 함수
def train_vae(model, dataloader, optimizer, epochs=20):
    model.train()
    for epoch in range(epochs):
        total_loss = 0
        total_correct = 0
        total_samples = 0
        with tqdm(dataloader, unit="batch") as tepoch:
            tepoch.set_description(f"Epoch {epoch + 1}/{epochs}")
            for batch in tepoch:
                batch = batch.to(device)
                optimizer.zero_grad()
                recon_batch, mu, logvar = model(batch)
                loss = vae_loss_mse(recon_batch, batch, mu, logvar)

                # NaN 방지를 위해 체크
                if torch.isnan(loss):
                    # print("Warning: Loss is NaN, skipping this batch")
                    continue

                # 역전파 및 최적화
                loss.backward()
                optimizer.step()

                # 배치의 로스 및 정확도
                total_loss += loss.item()
                total_samples += batch.size(0)

                # 정확도 계산 (근사적 비교, 설정에 맞춰 수정 가능)
                correct = (recon_batch.round() == batch).sum().item()
                total_correct += correct

                # 진행 상황 업데이트
                tepoch.set_postfix(loss=loss.item(), accuracy=100 * total_correct / total_samples)

            avg_loss = total_loss / total_samples
            accuracy = 100 * total_correct / total_samples
            print(f"Epoch {epoch + 1}, Average Loss: {avg_loss:.4f}, Accuracy: {accuracy:.2f}%")

# CustomDataset 생성
class CustomDataset(Dataset):
    def __init__(self, data):
        if isinstance(data, pd.DataFrame):
            self.data = data.values
        else:
            self.data = data
```

```
        # 데이터 내 결측값이 있을 경우 0으로 채움 (필요시 다른 방법으로 처리 가능)
        self.data = np.nan_to_num(self.data)

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        return torch.tensor(self.data[idx], dtype=torch.float32)

# 결측값 보간 함수
def impute_missing_values(model, data_with_nan):
    model.eval()
    imputed_data = data_with_nan.copy()

    # 결측값 위치 확인
    missing_mask = np.isnan(imputed_data)
    imputed_data[missing_mask] = 0

    # 데이터 예측
    imputed_data = torch.tensor(imputed_data, dtype=torch.float32).to(device)
    with torch.no_grad():
        imputed_data, _, _ = model(imputed_data)

    # 복원된 값으로 결측값 채우기
    imputed_data = imputed_data.cpu().numpy()
    data_with_nan[missing_mask] = imputed_data[missing_mask]

    return data_with_nan

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
# 데이터 전처리
def preprocess_data(scaler,data, flag):
    if flag == 0:
        # 결측값을 0으로 대체
        data = np.nan_to_num(data, nan=0.0, posinf=0.0, neginf=0.0)
        # 스케일링
        data = scaler.fit_transform(data)
    else:
        data = scaler.transform(data)
    return data

# 실제 데이터 전처리 적용
data=preprocess_data(scaler,merged_df.dropna(axis=0).reset_index(drop=True),flag=0)
```

```
# 데이터셋에 전처리된 데이터 적용
dataset = CustomDataset(data)
dataloader = DataLoader(dataset, batch_size=32, shuffle=True)

# 결측값 있는 데이터에도 동일한 전처리 적용
missing_data = preprocess_data(scaler, merged_df[merged_df.isna().any(axis=1)].reset_
index(drop=True),flag=1)

# 모델 초기화 및 학습
input_dim = data.shape[1]
hidden_dim = 20
latent_dim = 5
vae = VAE(input_dim, hidden_dim, latent_dim).to(device)  # 모델을 GPU로 이동
optimizer = optim.Adam(vae.parameters(), lr=1e-3)

# 모델 학습
train_vae(vae, dataloader, optimizer, epochs=20)

# 결측값 보간
imputed_data = impute_missing_values(vae, missing_data)

print("Imputed Data with Missing Values:")
print(imputed_data)

interpolated_data = pd.DataFrame(scaler.inverse_transform(imputed_
data),columns=merged_df.columns)

# 1. 불필요한 연관 없는 데이터 제거 ⇒ 치아 우식증, 치석
columns_to_drop = ['치아우식증유무', '치석']

interpolated_data.drop(columns=columns_to_drop, inplace=True)

print("제거된 컬럼:", columns_to_drop)

# 3. 범주형 변수 값 이진화 ⇒
# 음주여부 : 0.5를 임계값으로 설정  ~ 0.5 ⇒ 0 | 0.5 ~ 1.0 ⇒ 1.0
# 흡연상태 1 ~ 1.5 ⇒ 1 |  1.6 ~ 2 ⇒ 2  | 2 ~ 2.5 ⇒ 2 | 2.5 ~ 3.0 ⇒ 3.0
# 범주형 변수 목록
categorical_fill_mode = ['흡연상태', '음주여부']

# 각 변수의 이진화 조건에 따라 값 변환
for col in categorical_fill_mode:
    if col in interpolated_data.columns:  # 변수 존재 여부 확인
        # 이진화 조건 적용
        if col == '음주여부':
            # 음주여부: 0.5 이하 ⇒ 0, 0.5 초과 ⇒ 1
```

```
            interpolated_data[col] = interpolated_data[col].apply(lambda x: 0 if x <= 0.5 else 1)
        elif col == '흡연상태':
            # 흡연상태: 주어진 범위에 따라 값 이진화
            def binarize_smoking_status(x):
                if 1 <= x <= 1.5:
                    return 1
                elif 1.6 <= x <= 2.5:
                    return 2
                elif 2.5 < x <= 3.0:
                    return 3
                else:
                    return x  # 범위를 벗어나는 값은 그대로 유지
            interpolated_data[col] = interpolated_data[col].apply(binarize_smoking_status)

# 4. 소수의 결측치는 행 단위로 제거
interpolated_data.dropna(inplace=True)

# 결과 확인
print("결측치 처리 후 데이터의 크기:", interpolated_data.shape)
print("결측치 처리 후 남은 결측치 개수:\n", interpolated_data.isnull().sum())

# 결측치가 없던 데이터와 concat
final_data = pd.concat([merged_df.dropna(axis=0).reset_index(drop=True).iloc[:,:-
2],interpolated_data]).reset_index(drop=True)
final_data

# 시각화할 수치형 변수 리스트
numeric_cols = [
    '신장(5Cm단위)', '체중(5Kg 단위)', '시력(좌)', '시력(우)', '허리둘레', '수축기 혈압', '이완기 혈압', '식
전혈당(공복혈당)', '총 콜레스테롤', '트리글리세라이드', 'HDL 콜레스테롤', 'LDL 콜레스테롤',
    '혈색소', '혈청크레아티닌', '(혈청지오티)AST', '(혈청지오티)ALT', '감마 지티피'
]

# 그래프 격자 설정
n_cols = 5  # 한 행에 표시할 그래프 수
n_rows = (len(numeric_cols) + n_cols - 1) // n_cols  # 총 행 수 계산

# Figure 생성
fig, axes = plt.subplots(n_rows, n_cols, figsize=(15, n_rows * 4))
fig.suptitle("수치형 변수의 Boxplot", fontsize=16, y=1.02)


# 각 변수에 대해 Boxplot 생성
for i, col in enumerate(numeric_cols):
    row = i // n_cols
```

```
    col_pos = i % n_cols
    sns.boxplot(data=final_data, x=col, ax=axes[row, col_pos], color="skyblue")
    axes[row, col_pos].set_title(f'{col}', fontsize=14)
    axes[row, col_pos].set_xlabel('')
    axes[row, col_pos].tick_params(axis='y', labelsize=10)

# 빈 플롯 제거 (총 변수 수가 격자보다 적은 경우)
for j in range(i + 1, n_rows * n_cols):
    fig.delaxes(axes[j // n_cols, j % n_cols])

plt.tight_layout()
plt.show()

def remove_outliers_iqr(df):
    """
    데이터프레임의 각 열에 대해 IQR을 사용하여 이상치를 제거합니다.
    """
    df_cleaned = df.copy()
    for column in df.select_dtypes(include=[float, int]).columns:
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
      df_cleaned = df_cleaned[(df_cleaned[column] >= lower_bound) & (df_cleaned[column]
<= upper_bound)]
    return df_cleaned

df_cleaned = remove_outliers_iqr(final_data)
df_cleaned = df_cleaned.reset_index(drop=True)
df_cleaned

# 시각화할 수치형 변수 리스트
numeric_cols = [
    '신장(5Cm단위)',  '체중(5Kg 단위)', '시력(좌)', '시력(우)', '허리둘레', '수축기 혈압', '이완기 혈압', '식
전혈당(공복혈당)', '총 콜레스테롤', '트리글리세라이드', 'HDL 콜레스테롤', 'LDL 콜레스테롤',
    '혈색소', '혈청크레아티닌', '(혈청지오티)AST', '(혈청지오티)ALT', '감마 지티피'
]

# 그래프 격자 설정
n_cols = 5  # 한 행에 표시할 그래프 수
n_rows = (len(numeric_cols) + n_cols - 1) // n_cols  # 총 행 수 계산

# Figure 생성
fig, axes = plt.subplots(n_rows, n_cols, figsize=(15, n_rows * 4))
fig.suptitle("수치형 변수의 Boxplot", fontsize=16, y=1.02)
```

```
# 각 변수에 대해 Boxplot 생성
for i, col in enumerate(numeric_cols):
    row = i // n_cols
    col_pos = i % n_cols
    sns.boxplot(data=df_cleaned, x=col, ax=axes[row, col_pos], color="skyblue")
    axes[row, col_pos].set_title(f'{col}', fontsize=14)
    axes[row, col_pos].set_xlabel('')
    axes[row, col_pos].tick_params(axis='y', labelsize=10)

# 빈 플롯 제거 (총 변수 수가 격자보다 적은 경우)
for j in range(i + 1, n_rows * n_cols):
    fig.delaxes(axes[j // n_cols, j % n_cols])

plt.tight_layout()
plt.show()
```

접수번호 2024-12

대구 주요 공원 인근 지역의 생활인구와 상권 영향 분석

분석 요청자	소속	기업	성명	서OO
	휴대전화	-	전자우편	-
분석유형	■ 데이터 전처리	■ 데이터 시각화	▢ 머신러닝	▢ 기타
데이터 분석가	김건욱 센터장, 윤주혁 사무원			
분석 기간	2024.10 ~ 2024.10(1개월)			

가.

분석 주제

분석 주제(요청내용)

- 도시기본계획 수립에서 인구 배분 계획은 도시의 장기적 발전을 위한 필수 요소임. 이를 통해 도시 성장과 발전 방향을 설정하는 기초가 마련됨
- 인구 배분 계획은 통계청의 장래인구 추계를 바탕으로 인구 증가와 감소 추세를 예측하고, 각 지역의 특성과 발전 가능성을 반영해 각 구역에 인구를 효율적으로 배분하는 데 활용됨
- 따라서 계획 인구 산정에 필요한 근거 자료를 명확히 제시하여, 도시 계획의 신뢰성을 높이고 자원 배분 및 인프라 구축의 효율성을 극대화하고자 함
- 이번 분석은 대구 주요 기반시설과 생활권별 특성, 그리고 각 지역의 인구 변동성을 체계적으로 파악하는 것을 목표로 함
- 특히, 대구시 내 주요 공원 인근 지역의 생활인구와 업종별 카드 매출 정보를 전처리 및 분석하여 공원이 미치는 영향을 파악하고자 함. 공원은 시민들에게 중요한 여가 및 휴식 공간일 뿐 아니라, 인근 상권과 생활인구에도 큰 영향을 미치는 핵심 기반시설임
- 공원 주변의 생활인구 현황을 분석함으로써, 해당 기반시설이 인근 상권과 경제 활동에 미치는 영향을 구체적으로 살펴보고자 함
- 생활인구 데이터와 상권 정보를 바탕으로 대형 공원 인근 지역의 기술 통계 분석을 수행하여, 공원과 같은 주요 기반시설이 인근 지역의 인구 분포와 경제 활동에 미치는 구체적 영향을 파악하고자 함
- 이를 통해 대구시의 인프라 개발과 자원 배분 정책 수립에 실질적 기여를 하는 자료를 도출하고, 공원 인근 지역의 경제 활성화와 인프라 투자 효율성 제고에 중요한 자료로 활용될 것임

분석 필요성

- **(공원의 경제적 영향 분석 필요성)** 공원은 도시 내에서 여가와 휴식을 제공하는 공간일 뿐만 아니라 주변 지역 상권과 생활인구에 영향을 미칠 수 있는 중요한 기반시설임. 이에 따라 공원 인근의 생활인구 및 상권 변동성을 파악함으로써 공원이 주변 경제에 미치는 영향을 분석하는 것은 중요함
- **(인구 변화에 따른 인프라 요구 예측)** 대구 주요 공원 인근 지역의 인구 분포와 경제 활동에 대한 구체적인 이해를 통해 도시계획 및 인프라 구축에 대한 실질적인 요구를 사전에 파악할 수 있음. 이를 바탕으로 인구 증가 또는 감소에 따른 인프라 수요를 예측하고, 생활권별로 최적의 자원 배분 전략을 수립할 수 있음
- **(데이터 기반 정책 수립의 필요성)** 공원 인근 생활인구와 상권에 대한 데이터를 바탕으로 공공 정책을 수립하는 것은 경험적 접근보다 더 높은 신뢰성과 정확성을 제공함

분석 전략

- **(시간대별 생활인구 분석)** 생활인구 데이터를 시간대별로 분류하여 8시간 단위로 구분된 생활인구 흐름을 파악함. 이를 통해 공원 인근 지역의 시간대별 인구 변동성을 파악하고, 주요 시간대에 따른 생활인구 밀집도를 분석하여 인구 집중 패턴을 도출할 수 있음
- **(업종별 카드 매출 데이터 분석)** 공원 주변 상권의 업종별 카드 매출 데이터를 분석하여 상권 특성을 심층적으로 파악함. 이 과정에서 생활인구와 상권 간의 상관관계를 도출하며, 공원 방문객과 상권 활성화 사이의 관계를 구체적으로 분석할 수 있음
- **(향후 분석 및 추후 고도화를 위한 기반 마련)** 본 분석을 바탕으로 추가적인 데이터 수집 및 모델링을 통해 향후 공원 인근 지역의 경제적 및 인구적 변동성을 더욱 고도화하여 예측할 수 있음

나.
데이터 분석 및 결과

활용 데이터

- 본 분석을 수행하기 위한 데이터로 생활인구, 신용카드 매출 데이터, 공원 공간정보 데이터를 활용함
- 분석 대상 기간은 2022년~2023년 이며, 공간적 범위는 대구광역시 전체를 대상으로 함

연번	데이터명	내용	시간 범위	데이터 규모	출처
1	생활인구 데이터	STD_YMD , TIME, SEX_AGE, HCODE, H_POP, W_POP, V_POP	2022~2023	69,127,095건	대구 빅데이터 활용센터
2	신용카드 매출 데이터	소비관광지역명칭, 가맹점행정동, 월별, 일별, 개인법인구분, 대분류, 중분류, 소분류, 카드이용금액, 카드이용건수	2022~2023	983,089 건	대구 빅데이터 활용센터
3	공원 데이터 shp	공원명, polygon	-	8건	자체 수집

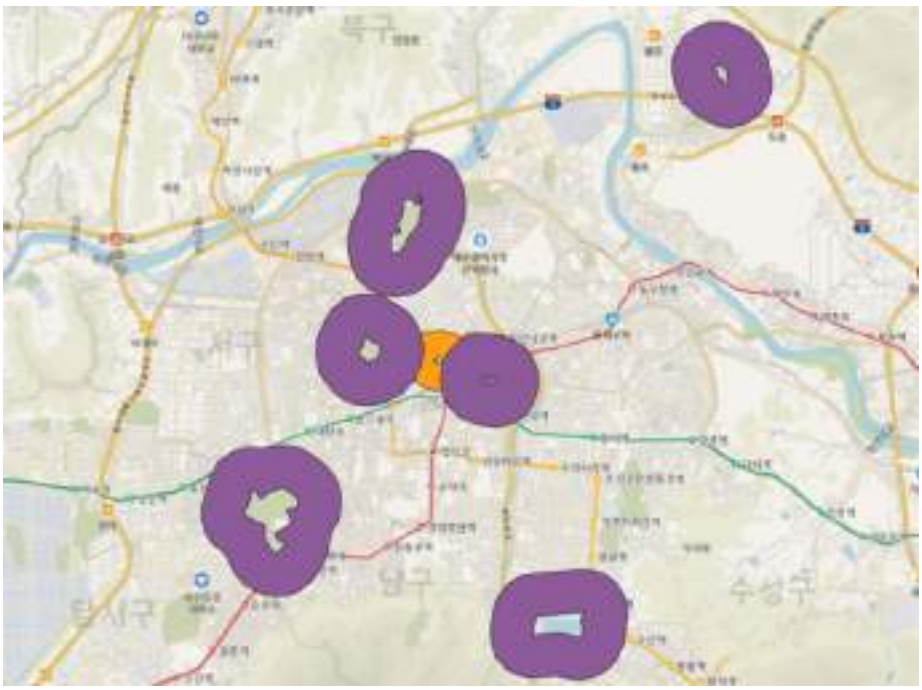
<표> 활용 데이터

공원별 분류 및 면적표

연번	분류	공원 명	결정면적 ()	버퍼 영역(m)	기타
1	근린공원(중구)	달성공원	129,700	1,000	
2	근린공원(달서구)	두류공원	1,878,400	1,000	
3	근린공원(북구)	침산공원	223,600	1,000	
4	유원지	수성유원지	1,212,680	1,000	
5	기타	비슬산자연공원(휴양림)	12,322,715	1,000	
6	수변공원	봉무공원	1,878,400	1,000	
7	근린공원(중구)	국채보상운동기념공원	43,715	1,000	
8	근린공원(중구)	경상감영공원	16,500	500	

<표> 분석 대상 공원

- 분석 목적인 공원 인근 지역의 생활인구 및 신용카드 매출 데이터 분석을 위해, 공원의 인근 지역의 범위 지정을 위한 버퍼 생성 작업을 수행함
- 도시공원 및 녹지 등에 관한 법률[1] 을 참조하여 공원 별 버퍼 영역을 3만 제곱미터 이상인 경우, 1000m 이하인 경우 500m로 나누어, 도보 접근 가능 범위를 지정함
- 나누어진 영역대로 버퍼를 생성한 결과는 아래와 같음



<표> 공원 면적에 따른 버퍼 생성결과

1) 「도시공원 및 녹지 등에 관한 법률」 제15조 제2항; 「도시공원 및 녹지 등에 관한 법률 시행규칙」, 국토교통부, 국토교통부령 제1315호 (2024. 3. 14. 시행).

데이터 전처리 및 공간 결합

- **생활인구 데이터 - 전처리**
  - 생활인구 데이터는 2022년 1월부터 2023년 12월까지의 집계된 데이터를 활용함
  - 데이터의 총 행 수는 69,127,095개이며, 데이터의 형태는 아래와 같음
  - 각 변수들의 정보를 확인한 결과, STD_YMD 컬럼은 {연도}{월}{일}로 구성되어 있으며, TIME은 0~23까지의 값, SEX_AGE는 {성별}_{연령대}로 구성되어 있어, 0~9세, 10~14,15~19,20~24,25~29,... 70대 이상 으로 이루어져 있음
  - HCODE는 행정동 코드로, 2022년에는 139개, 2023년에는 147개로 이루어져 있음. 이는 2023년 군위가 대구광역시 내로 편입되며 반영된 변화임
  - 본 분석에서는 공원이 모두 군위를 제외한 지역에 위치해 있어 군위 지역 코드 변화는 큰 문제가 되지 않음
  - 데이터의 이상치 및 결측치를 확인한 결과, 이상치는 중위값으로 대체하고 결측치는 제거하는 방식으로 데이터 전처리를 수행함

Unnamed: 0	H_POP	W_POP	V_POP
count	69099053	69127024	69127044
mean	470	66	121
std	421	140	132
min	0	0	0
25%	222	8	48
50%	375	26	91
75%	630	78	163
max	4719	5493	10739

```
1 df.isna().sum().compute()
✓ 6.0s
STD_YMD        0
TIME           0
SEX_AGE        0
HCODE          0
H_POP      28042
W_POP         71
V_POP         51
dtype: int64
```

<표> 데이터 통계량 확인(좌) / 데이터 결측치 확인(우)

- 본 분석의 목적에 맞춰 데이터를 가공하기 위해, 연도별, HCODE, 성별, 시간대(0~8시 : 오전, 8~16 : 오후, 16~24 : 저녁) 으로 나눠 판다스의 Groupby 작업을 수행함

<표> 데이터 그룹화 코드

- 그룹화 작업 후, H_POP(거주인구), W_POP(직장인구), V_POP(방문인구)의 합계 및 평균을 연산

- **생활인구 데이터 – 공간결합**
  - 생활인구와 신용카드 데이터는 공간적 범위가 행정동 단위로 구성되어 있음. 이를 분석에 적용하기 위해 대구광역시의 행정동 단위 shp 파일을 사용함
  - 이 데이터는 시군구 및 행정동 열을 포함하며, HCODE 열과 공원 반경 정보를 담은 폴리곤의 결합을 위해 geometry 열을 포함함

- 생활인구 데이터와 대구 공간 정보, 공원 공간 정보를 결합하는 순서는 다음과 같음
  i) 생활인구 데이터와 대구 공간 정보를 HCODE 기준으로 결합함
  ii) 공원 공간 정보를 geopandas의 sjoin(predicate='intersects') 조건으로 결합함
- geopandas의 sjoin에서 사용한 intersects는 두 폴리곤이 일부라도 겹칠 경우 결합하는 조건임
- 결합 완료 후, 생활인구의 합계와 평균으로 구분한 최종 데이터 생성

- **신용카드 매출 데이터 - 전처리**
  - 신용카드 매출 데이터는 2022년 1월부터 2023년 12월까지 집계된 신한카드 가맹점 매출 데이터를 활용함
  - 변수 확인 결과, 이상치는 가맹점 행정동에서 총 5,790건으로 전체 데이터에 미치는 영향이 적어 삭제함
  - 가맹점 분류는 소분류를 기준으로 분석하여 공원 인근의 매출액이 높은 업종을 파악하고자 함

<표> 결측치 (좌)

소분류 업종 종류 (우)

- 가맹점 행정동의 유니크 값을 확인한 결과, 총 53개로 대구시 행정동 139개(2022년) 및 147개(2023년)에 비해 상당히 낮은 수치임. 이는 데이터가 대구시 전역이 아닌 일부 지역의 가맹점 정보만 포함하고 있음을 의미함

```
array(['가창면', '고산2동', '공산동', '관문동', '구지면', '남산1동', '내당1동', '내당2.3동',
       '다사읍', '대명9동', '대봉1동', '대신동', '도원동', '도평동', '두류1.2동', '두산동',
       '만촌2동', '만촌3동', '범어3동', '범어4동', '봉덕2동', '봉덕3동', '불로.봉무동', '비산2.3동',
       '산격1동', '산격2동', '산격3동', '산격4동', '삼덕동', '상동', '상인2동', '상인3동',
       '성내1동', '성내2동', '성내3동', '성당동', '신암1동', '신천4동', '안심2동', '옥포읍',
       '용산1동', '유가읍', '지산2동', '진천동', '칠성동', '태전1동', '파동', '해안동', '현풍읍',
       '화원읍', '황금1동', '황금2동', '효목1동'], dtype=object)
```

<표> 가맹점 행정동 유니크 값

- '일별' 컬럼을 확인한 결과, 데이터가 일자별로 되어 있어 이를 연도 및 월별로 구분하는 신규 컬럼을 생성함
- 이후 가맹점 행정동, 연도, 월, 소분류를 기준으로 그룹화하여 카드 이용 금액과 이용 건수의 합계를 구함

다.

분석 활용 전략

결론 및 활용 전략

- **데이터 전처리와 분석 결과 요약**
  - 생활인구와 카드 매출 데이터는 2022년 1월부터 2023년 12월까지의 데이터를 활용하여 분석에 적합한 형식으로 전처리함
  - 시간대별, 성별, 연령대별 생활인구 데이터를 HCODE와 결합하고, 공원 반경에 따른 공간 결합을 통해 인근 지역의 생활인구 및 상권 매출 특성을 도출함
  - 또한, 생활인구 데이터는 시간대별로 8시간 단위로 나누어 공원 인근의 생활인구 밀집 패턴을 분석하였으며, 카드 매출 데이터는 공원 인근 상권에서의 소비 특성과 공원 방문의 상관성을 파악할 수 있도록 업종별로 정리함
- **공원 인근 생활인구, 경제 및 상권 특성 파악 가능**
  - 본 분석을 통해 공원 인근의 생활인구와 경제적 특성, 그리고 상권 매출의 특성을 파악할 수 있음.
  - 이는 공원 주변 상권 활성화와 자원 배분에 필요한 실질적인 데이터로 활용할 수 있으며, 특정 연령층과 업종 간의 관계를 파악해 다양한 기초 전략 수립에 기여할 수 있음

한계점 및 추후 보완 계획

- **데이터 기간 및 시계열 분석의 제한성**
  - 본 분석에 사용된 데이터는 2년치로, 공원 인근 상권과 생활인구 변화의 장기적인 추세를 파악하기에는 한계가 있음
  - 향후에는 5년 이상의 장기 시계열 데이터를 추가 확보하여, 공원 인근 지역의 인구 및 매출 변화를 보다 장기적으로 분석하고 예측할 수 있도록 보완할 필요가 있음
- **공간 범위의 정밀도 부족**
  - 선행 연구와 관계 법령을 참고하여 공원 반경 1000m 및 500m 범위를 공원 인근 지역으로 정의하였으나, 생활인구와 카드 매출 데이터가 행정동 단위로 제공되어 이를 정확히 반영하지 못한 한계가 있음
  - 향후 보다 세밀한 공간 데이터를 확보하거나, 행정동 경계를 고려한 보정 방법을 통해 공원 반경 내 생활인구와 상권 특성을 정밀하게 반영할 수 있도록 보완할 필요가 있음

Appendix

```
[생활인구 데이터]
import numpy as np
import pandas as pd

import dask.array as da
import dask.bag as db

from dask.distributed import LocalCluster
import dask.dataframe as dd
import glob
import os

# Dask의 LocalCluster를 사용하여 클러스터 생성
cluster = LocalCluster(n_workers=12, threads_per_worker=2)

# 생성된 클러스터에 연결된 클라이언트 가져오기
client = cluster.get_client()

# 파일 경로 패턴 정의
file_pattern = '../2022/01.서비스인구/daegu_service_sex_age_pop_*.csv'

# glob 모듈을 사용하여 파일 패턴에 맞는 모든 파일 리스트 가져오기
file_list = glob.glob(file_pattern)

# 파일 경로 패턴 정의
file_pattern = '../2023/01.서비스인구/*/daegu_service_sex_age_pop_*.csv'
file_list.extend(glob.glob(file_pattern))

# 데이터 로드 'blocksize' = Dask가 파일을 나누어 읽는 단위 설정
df = dd.read_csv(file_list, delimiter='|', encoding='utf-8',blocksize='150MB', assume_
missing=True)
# df = df.drop(axis=1,columns=['HCODE'])
df.head()

# 통계량 계산 (실행 시 메모리 문제 발생할 수 있음)
stats = df[['H_POP', 'W_POP', 'V_POP']].describe().compute()
stats.to_csv('E:/skt_proj/stats.csv')

# 저장한 통계량을 pandas로 다시 로드
stats = pd.read_csv('E:/skt_proj/stats.csv').set_index('Unnamed: 0')
stats.astype(int)
```

```
# 중위값으로 이상치 대체하는 함수
def replace_outliers_with_median(df, columns, stats, multiplier=1.5):
    def replace_partition(part, stats, columns):
        for column in columns:
            # IQR(사분위 범위)를 이용하여 이상치의 하한선과 상한선을 계산
            Q1 = stats.loc['25%', column]
            Q3 = stats.loc['75%', column]
            IQR = Q3 - Q1
            lower_bound = Q1 - multiplier * IQR
            upper_bound = Q3 + multiplier * IQR
            median = stats.loc['50%', column]

            # 이상치 조건을 만족하는 경우 중위값을 대체
            condition = (part[column] < lower_bound) | (part[column] > upper_bound)
            part[column] = np.where(condition, median, part[column])

        return part

    # Dask DataFrame에서 각 파티션에 대해 이상치 대체 함수 적용
    df = df.map_partitions(replace_partition, stats=stats, columns=columns)

    return df


df = replace_outliers_with_median(df, ['H_POP', 'W_POP', 'V_POP'], stats)

# 연도 그룹화 기준 컬럼 생성
df['YEAR_GROUP'] = df['STD_YMD'].astype(str).str[:4].astype(int).apply(lambda x: '22년' if
x == 2022 else '23년')

# 시간대 그룹화 기준 컬럼 생성
def time_of_day(hour):
    if 0 <= hour < 8:
        return '오전'
    elif 8 <= hour < 16:
        return '오후'
    else:
        return '저녁'

df['TIME_GROUP'] = df['TIME'].apply(time_of_day)

# 성별 그룹화 기준 컬럼 생성
df['SEX_GROUP'] = df['SEX_AGE'].str[0]
```

```
# 그룹화 코드
grouped = df.groupby(['YEAR_GROUP', 'HCODE', 'SEX_GROUP', 'TIME_GROUP'])
# grouped = df.groupby(['YEAR_GROUP', 'HCODE', 'SEX_GROUP'])

result = grouped.agg({
                'H_POP': ['sum', 'mean'],
                'W_POP': ['sum', 'mean'],
                'V_POP': ['sum', 'mean']
            }).reset_index().compute()

# 컬럼명을 편리하게 변경
result.columns = ['_'.join(col).strip() if col[1] else col[0] for col in result.columns.values]
result.rename(columns={
                'H_POP_sum': 'H_POP_TOTAL',
                'W_POP_sum': 'W_POP_TOTAL',
                'V_POP_sum': 'V_POP_TOTAL',
                'H_POP_mean': 'H_POP_MEAN',
                'W_POP_mean': 'W_POP_MEAN',
                'V_POP_mean': 'V_POP_MEAN'
                }, inplace=True)
result
[카드 매출 데이터]
import glob
import pandas as pd
import os

# 파일 경로 설정
base_path = r"H:\00. DIP\01. 오션라이트 분석\02. 신한카드_데이터_전처리\01. data\2023"

# 경로에서 파일 패턴에 맞는 파일 찾기
file_pattern = os.path.join(base_path, "*", "1.대구시_내국인_업종별_*.txt")
file_list = glob.glob(file_pattern)

# 파일 목록 확인
print(f"불러온 파일 목록: {file_list}")

# 파일 읽기 및 병합
dataframes = []
for file in file_list:
    df = pd.read_csv(file, sep="|", encoding="utf-8")  # 파일 형식에 따라 sep, encoding 수정
    dataframes.append(df)

# 모든 파일 병합
df = pd.concat(dataframes, ignore_index=True)
# 필요한 경우 파일 저장
df.to_csv("../01. data/2023/2023_업종별.csv", index=False, encoding="utf-8")
```

```
# 파일 경로 설정
base_path = r"H:\00. DIP\01. 오션라이트 분석\02. 신한카드_데이터_전처리\01. data\2022"

# 경로에서 파일 패턴에 맞는 파일 찾기
file_pattern = os.path.join(base_path, "*", "1.대구시_내국인_업종별_*.*")
file_list = glob.glob(file_pattern)

# 파일 목록 확인
print(f"불러온 파일 목록: {file_list}")

# 파일 읽기 및 병합
dataframes = []
for file in file_list:
    if 'csv' == file.split('.')[-1]:
        df = pd.read_csv(file, sep=",", encoding="cp949")  # 파일 형식에 따라 sep, encoding 수정
        dataframes.append(df)
    else:
        df = pd.read_csv(file, sep="|", encoding="utf-8")  # 파일 형식에 따라 sep, encoding 수정
        dataframes.append(df)
# 모든 파일 병합
df = pd.concat(dataframes, ignore_index=True)
df.to_csv("../01. data/2022/2022_업종별.csv", index=False, encoding="utf-8")

card_2022 = pd.read_csv('../01. data/2022/2022_업종별.csv')
card_2023 = pd.read_csv('../01. data/2023/2023_업종별.csv')
card_data  =  pd.concat([card_2022,card_2023]).drop(columns=['월별','개인법인구분','대분
류','중분류']).reset_index(drop=True)
card_data

# 연도 그룹화 기준 컬럼 생성
card_data['YEAR'] = card_data['일별'].astype(str).str[:4].astype(int).apply(lambda x: 2022
if x == 2022 else 2023)
card_data['MONTH'] = card_data['일별'].astype(str).str[4:6].astype(int)
card_data = card_data.drop(columns=['일별'])


card_data_sum = card_data.groupby(by=['가맹점행정동','YEAR', 'MONTH','소분류']).agg({
    '카드이용금액':'sum',
    '카드이용건수':'sum'
})
card_data_sum = card_data_sum.reset_index()
card_data_sum

# 파일 불러오기
daegu_block = gpd.read_file('../01. data/행정동_shp/DMM_BLOCK.shp')
```

```
# 에러 방지를 위한 CRS 설정
daegu_block = daegu_block.to_crs(epsg=3857)

# CRS 확인
print(f"daegu_block CRS: {daegu_block.crs}")

# 필요없는 열 삭제 ⇒ 격자로 공간범위를 지정함에 따른 기존 공간 분류기준(시군구) 삭제필요
daegu_block = daegu_block.drop(columns=['STD_YYYY','BLOCK_CD', 'SIDO_CD','SGNG_CD','LDONG_CD','SIDO_NM', 'index'])
daegu_block = daegu_block.rename(columns={'ADONG_CD':'HCODE'})
daegu_block['HCODE'] = daegu_block['HCODE'].astype('int64')
# Dissolve polygons based on the 'HCODE' column to combine polygons with the same HCODE
daegu_block = daegu_block.dissolve(by="HCODE").reset_index()
daegu_block

# 예시: 조건을 충족하는 행을 선택하는 '가맹점행정동_매칭' 열을 생성
daegu_block['가맹점행정동_매칭'] = daegu_block.apply(
    lambda x: x['ADONG_NM'] if x['ADONG_NM'] in card_data_sum['가맹점행정동'].values else (
     x['LDONG_NM'] if x['LDONG_NM'] in card_data_sum['가맹점행정동'].values else np.nan
   ), axis=1
)
# # '가맹점행정동_매칭' 값이 있는 행만 남김
daegu_block_filtered = daegu_block.dropna(subset=['가맹점행정동_매칭'])
daegu_block_filtered


# '가맹점행정동'과 '가맹점행정동_매칭'을 키로 병합
merged_df = pd.merge(
   card_data_sum,
   daegu_block_filtered,
   left_on='가맹점행정동',
   right_on='가맹점행정동_매칭',
   how='left'
)

merged_df = merged_df[['HCODE','geometry','SGNG_NM','ADONG_NM','LDONG_NM','YEAR','MONTH','소분류','카드이용금액','카드이용건수']]
merged_df
```

[공간 결합]

```
import numpy as np
import pandas as pd
import glob
import geopandas as gpd
# import dask_geopandas as dgpd
from shapely.geometry import Point
from shapely.geometry import Polygon

# 파일 불러오기
daegu_block = gpd.read_file('02. 전국_읍면동_shp파일/DMM_BLOCK.shp')

# 에러 방지를 위한 CRS 설정
daegu_block = daegu_block.to_crs(epsg=3857)

# CRS 확인
print(f"daegu_block CRS: {daegu_block.crs}")

# 필요없는 열 삭제 ⇒ 격자로 공간범위를 지정함에 따른 기존 공간 분류기준(시군구) 삭제필요
daegu_block = daegu_block.drop(columns=['STD_YYYY','BLOCK_CD', 'SIDO_CD','SGNG_
CD','LDONG_CD','SIDO_NM', 'index'])
daegu_block = daegu_block.rename(columns={'ADONG_CD':'HCODE'})
daegu_block['HCODE'] = daegu_block['HCODE'].astype('int64')
# Dissolve polygons based on the 'HCODE' column to combine polygons with the same
HCODE
daegu_block = daegu_block.dissolve(by="HCODE").reset_index()
daegu_block

df_inflow_sex_time_group = pd.read_csv('01. 대구 생활인구(2022~2023)/행정동_성별_시간
대별_생활인구_최종결과.csv')
df_inflow_sex_time_group = df_inflow_sex_time_group.rename(columns={'YEAR_
GROUP':'STD_YYYY'})
df_inflow_sex_time_group['STD_YYYY'] = df_inflow_sex_time_group['STD_YYYY'].str.
replace('22년', '2022').replace('23년', '2023')
df_inflow_sex_time_group['HCODE'] = df_inflow_sex_time_group['HCODE'].astype('int64')
df_inflow_sex_time_group

add_inflow_data_to_shp = pd.merge(left=daegu_block, right=df_inflow_sex_time_group,
on='HCODE',how='right').dropna()
add_inflow_data_to_shp

# 파일 불러오기
park_30k = gpd.read_file('03. 공원생성버퍼/면적_3만이상_공원_1km버퍼.shp')
```

```
# 에러 방지를 위한 CRS 설정
park_30k = park_30k.to_crs(epsg=3857)

# CRS 확인
print(f"park_30k CRS: {park_30k.crs}")

# 필요없는 열 삭제 ⇒ 격자로 공간범위를 지정함에 따른 기존 공간 분류기준(시군구) 삭제필요
park_30k = park_30k.drop(columns=['id','layer', 'path'])
print(park_30k)

# 파일 불러오기
park_under_30k = gpd.read_file('03. 공원생성버퍼/면적_3만이하_공원_500m버퍼.shp')

# 에러 방지를 위한 CRS 설정
park_under_30k = park_under_30k.to_crs(epsg=3857)

# CRS 확인
print(f"park_under_30k CRS: {park_under_30k.crs}")

# 필요없는 열 삭제 ⇒ 격자로 공간범위를 지정함에 따른 기존 공간 분류기준(시군구) 삭제필요
park_under_30k = park_under_30k.drop(columns=['id'])
print(park_under_30k)

park_geo = pd.concat([park_30k,park_under_30k])
park_geo

# 데이터 결합 : 폴리곤 간 결합이므로, 그리드와 일부라도 겹치는 영역이 있다면, 그 값을 더하도록 함
join_result = gpd.sjoin(add_inflow_data_to_shp, park_geo, how='inner',
predicate='intersects')
join_result
# 피벗 테이블 생성
pivot_table_sum = join_result.pivot_table(
    columns=['STD_YYYY','SEX_GROUP'],
    values=['H_POP_TOTAL','W_POP_TOTAL','V_POP_TOTAL'],
    index=['공원명', 'TIME_GROUP'],

    aggfunc='sum'
)
pivot_table_sum.to_csv('Total_Population_by_Park_Buffer.csv',index=False)
pivot_table_sum

# 피벗 테이블 생성
pivot_table_mean = join_result.pivot_table(
    columns=['STD_YYYY','SEX_GROUP'],
    values=['H_POP_MEAN','W_POP_MEAN','V_POP_MEAN'],
    index=['공원명', 'TIME_GROUP'],
```

```
    aggfunc='sum'
)
pivot_table_mean.to_csv('Average_Population_by_Park_Buffer.csv',index=False)
pivot_table_mean

card_geo = pd.read_csv('04. 신한카드 결합/card_processed_geo_merged.csv',encoding='cp949')
from shapely import wkt
card_geo['geometry'] = card_geo['geometry'].apply(wkt.loads)
card_geo = gpd.GeoDataFrame(card_geo, geometry='geometry')
# 좌표계 (CRS) 설정
card_geo.set_crs(epsg=3857, inplace=True)
card_geo

# 데이터 결합 : 폴리곤 간 결합이므로, 그리드와 일부라도 겹치는 영역이 있다면, 그 값을 더하도록 함
card_park_join_result = gpd.sjoin(card_geo, park_geo, how='inner', predicate='intersects')
card_park_join_result

# 소분류로 먼저 그룹화 후, 각 공원과 연도별로 카드 이용 금액 상위 3개 소분류를 필터링
top_3_by_subcategory = (
    card_park_join_result
    .groupby(['공원명', 'YEAR', '소분류'])
    .sum(numeric_only=True)[['카드이용금액','카드이용건수']]
    .reset_index()
    .sort_values(['공원명', 'YEAR', '카드이용금액'], ascending=[True, True, False])
    .groupby(['공원명', 'YEAR'])
    .head(3)
    .reset_index(drop=True)
)

# 각 공원과 연도별로 순위 컬럼 추가
top_3_by_subcategory['순위'] = (
    top_3_by_subcategory.groupby(['공원명', 'YEAR'])
    .cumcount() + 1
)

# 피벗 테이블 생성
pivot_table_sum = top_3_by_subcategory.pivot_table(
    columns=['YEAR'],
    values=['소분류','카드이용금액','카드이용건수'],
    index=['공원명', '순위'],
    aggfunc='sum'
)
pivot_table_sum
```

/2024/

빅데이터활용센터

데이터 분석 및 컨설팅
지원사례집 2024

컨설팅지원

Big Data

01
컨설팅지원

데이터분야
공공데이터

- **수 행 일 자** : 2024.03.29.
- **소　　속** : 지자체
- **질 의 자** : 배○○
- **데이터분야** : 관광데이터
- **지　　역** : 대구
- **응 답 자 명** : 박연진 전임

요청사항

- 수성구 축제(수성못 축제, 빛축제, 들안길 축제 등) 방문한 사람들의 출신, 이동경로 등을 파악하고 싶음
- 수성구 관내 사람들 통행 행적 어디 언제 많았는지 파악하고자함
- 수성구에 관광온 사람들의 관광코스 파악하고자함
- 즉, 수성구 관광 관련하여 인사이트 도출을 위해 이동통신 data를 받아 보았는데 어떻게 어떠한 방법으로 분석하면 좋을지 상담요청

지원내용

- 대구 빅데이터 활용센터 역할 및 주요 사업 안내, 분석실 제공데이터 및 분석 보조 사례 소개
- 대구시 관광데이터랩 소개, 생활인구 데이터와 관광인구 데이터의 차이점에 대해 설명 : 생활인구 데이터 안에는 어느 지역에서 오는지는 알 수 없음
- 특정 시간대 및 특정 장소에 대한 결과만 원한다면 활용센터 내 전문인력을 활용하여 분석 지원 가능하다고 안내
- 대구 관광데이터 랩 이용 추천 및 과거 활용센터에서 진행하였던 분석결과물 제공
- 더 전문적이고 집중적인 분석을 원할 시 빅데이터 분석과 경진대회에서 수행하고 있는 분석과제 신청을 통해 더 상세하고 구체적인 분석을 함께 진행할 수 있다고 안내함
- 또는, 분석 기술을 보유하고 있는 공급업체와 매칭 추천

02
컨설팅지원

데이터분야
공공데이터

- **수 행 일 자** : 2024.05.08.
- **소　　속** : 지자체
- **질 의 자** : 신○○
- **데이터분야** : 택시교통불편 신고 데이터
- **지　　역** : 대구
- **응 답 자 명** : 윤병진 전임

요청사항

- 대구 택시 교통 불편신고 데이터를 활용하여 서비스 개선을 위한 분석을 진행하고자 함
- 외부기업에 데이터를 맡기기 전에 자체적으로 개인정보 처리를 실시하였으나 일부 컬럼에 대해 마스킹 등의 익명처리 기술에 대한 전문성이 부족하여, 데이터의 전반적인 개인정보 보호 수준이 미흡한 상황임
- 이에 따라 데이터에 포함된 개인정보가 외부에 노출될 위험이 없는지 검토하고, 필요한 경우 적절한 개인정보 처리를 지원해 줄 것을 요청하였음

지원내용

- 자체 가명처리한 데이터를 검토한 결과, 민원 세부 내용 컬럼에 개인정보가 그대로 노출된 것을 확인하였습니다. 이에 따라 해당 컬럼에 대해 추가적인 가명처리를 실시하여 개인정보가 노출되지 않도록 조치하였음
- 또한 차량 번호가 개인을 특정할 수 있는 식별자로 취급될 우려가 있어, 총 30,636개의 레이블 중 11,638개의 유니크 값에 대해 인덱싱 처리를 통해 매핑처리를 실시하였음
- 차량 번호의 원본 데이터를 식별할 수 있도록 하기 위해 추가 정보를 함께 제공하여 전달하였으며, 이를 통해 필요한 경우 데이터의 원본을 추적할 수 있는 기능을 유지하면서도 개인정보 보호를 강화할 수 있도록 안내함
- 추가로 1차적으로 익명처리를 진행하였지만 텍스트로 구성되어있는 컬럼에 대해서는 센터 자체적으로 NER(Named Entity Recognition) 기법을 활용한 비정형 텍스트 데이터 식별 및 가명·익명 처리 기술을 개발 중에 있으며 향후 해당 데이터에 적용하여 더욱 안전한 데이터 활용을 지원할 수 있을것으로 안내하였음

03
컨설팅지원

데이터분야
공공데이터

- **수 행 일 자** : 2024.05.11.
- **소　　속** : 기업
- **질 의 자** : 김○○
- **데이터분야** : 교통
- **지　　역** : 대구
- **응 답 자 명** : 윤병진 전임

요청사항

- 2024년 대구 공공데이터 창업 경진대회에 데이터 제공을 위해 데이터 전처리를 진행하고 있으나, 개인정보식별 위험에 대한 우려로 개인식별위험 컨설팅 및 익명처리를 요청하였음
- 데이터는 고객, 주문, 결제, 리뷰, 가맹정, 배달팁 관련 총 6개의 파일을 제공하기 위하여 검토를 요청하였으며 대부분의 데이터는 개인정보 문제를 일으킬 가능성이 낮았지만, [고객 정보 데이터] 에서는 개인 식별 우려가 있는 부분이 있어 가명 및 익명 처리를 진행하여야 한다고 안내하였음

지원내용

- 개인정보 현황 이슈와 가이드라인을 소개하였음
- 1차원적으로 데이터에 개인정보가 있다고 문제되는 것과 함께 추가 정보가 있으면 개인식별의 우려가 있는 사례들에 대해 안내함
- A사의 데이터의 경우, [구군명, 동명, 성별, 생년월일, 아동급식 대상 여부]와 같은 컬럼들은 각각 따로 봤을 때는 개인정보 식별의 문제가 없어 보일 수 있으나, 생년월일과 같은 구체적인 정보는 다른 정보(해당 데이터셋의 다른 컬럼 포함)와 결합될 경우 특정 개인을 식별할 수 있는 가능성이 크다고 판단됨
- 따라서, 경진대회 특성상 불특정 다수가 데이터를 접할 가능성이 높기 때문에, 생년월일 정보를 연도와 월로 분할하고 나이 컬럼을 파생 변수로 생성하고, 기존의 생년월일 컬럼은 삭제처리 하였음
- 가명 처리 전과 후의 데이터를 함께 제공하여 내용을 안내하고, 변경된 데이터 형태에 맞게 수정한 테이블명세서를 제공하였음

04
컨설팅지원

데이터분야
공공데이터

- ☑ **수 행 일 자** : 2024.05.13.
- ☑ **소　　속** : 지자체
- ☑ **질 의 자** : 최○○
- ☑ **데이터분야** : 인구현황데이터
- ☑ **지　　역** : 대구
- ☑ **응 답 자 명** : 장원준 전임

요청사항

- 대구광역시 생활권 계획 및 수립을 위해 생활공간의 여건 변화와 주요 기반 시설별 이용 특징을 파악하기 위해 이용 현황을 종합적으로 분석한 보고서를 요청함
  ***요청한 주요 기반 시설** : 공원(근린공원, 어린이 공원, 소공원), 유원지, 하천(신천), 대표 문화시설(도서관, 박물관, 미술관)
- 분석 결과를 생활권 및 기반 시설 계획 수립을 위한 근거자료로 활용하고 여러 도시계획 적합성 판단에 사용할 것임
- 보유한 부동산종합공부시스템(KRAS) 데이터에는 건설 예정 시설이 포함되어 있지 않음. 따라서 지번별 토지이용계획 및 도시계획 정보를 확인할 수 있는 '토지이음' 웹사이트를 활용하여 분석할 것을 요청

지원내용

- 본 센터가 보유한 통신사 생활인구 데이터, 카드 데이터, 통신사 관광지 방문 인구 데이터를 활용하여 시간대별 주요 기반 시설 이용 현황을 열지도로 구현
- 열지도 시각화는 100m 격자 별로 방문한 인구를 집계한 것이기 때문에 요청한 주요 기반 시설 방문 목적성 여부 확인에는 한계가 있음. 따라서 전반적인 이용 현황 파악에 활용할 것을 권장함
- 군위군의 경우 데이터를 별도로 보유하고 있지 않아 본 분석에서 군위군은 제외함

05
컨설팅지원

데이터분야
공공데이터

- ☑ **수 행 일 자** : 2024.05.13.
- ☑ **소　　속** : 지자체
- ☑ **질 의 자** : 신○○
- ☑ **데이터분야** : 택시교통불편 신고 데이터
- ☑ **지　　역** : 대구
- ☑ **응 답 자 명** : 장원준 전임

요청사항

- 택시 민원이 증가하는 추세이며 기존에 민원 발생 시 조합이나 법인 회사가 택시 기사에게 보수 교육을 듣도록 조치했으나 오랫동안 업데이트되지 않은 교육 동영상으로 개선에 도움이 되지 않았음
- 택시 서비스 개선을 위해 두드리소 교통 불편 신고 관련 전산 데이터(120 콜센터 및 부서 신고전화 내용)와 새올상담민원 데이터 분석을 요청함
- 지역별 시기별 계절별 택시 민원 발생 현황을 키워드 중심으로 분석하여 일정한 유형이 확인될 경우 관련 내용을 법인 및 개인택시조합에 전파하여 민원에 대해 사전 대비하는 것이 최종 목표임

06
컨설팅지원

데이터분야
공공데이터

☑ **수 행 일 자** : 2024.05.14.
☑ **소　　속** : 지자체
☑ **질 의 자** : 권○○
☑ **데이터분야** : 교통사고 및 고령자 인구현황 데이터
☑ **지　　역** : 대구
☑ **응 답 자 명** : 김건욱 센터장

요청사항

- 고령자(65세 이상) 주요 밀집 지역 및 활동 시간에 대한 실효성 있는 분석을 통해 교통사고 발생 예상지역에 대한 선제적이고 다양한 대응 방안의 초석을 마련하고자함
- 고령자 생활인구 데이터, 구군별 도로명 주소, 교통시설물(횡단보도, 신호등 등) 설치 현황, 일자별 날씨, 고령자 복지시설물 등 다양한 데이터를 활용한 분석 및 시각화를 요청함
- 연령대별 분석이 아닌 고령자의 디테일한 나이별 분석이 가능한지 문의함

지원내용

- 나이별 분석의 경우 센터가 보유한 생활인구 데이터는 연령대별로 구분되어 있어 디테일한 연령별 분석을 원할 시, 보유한 주민등록인구 데이터에 대한 확보 여부 판단 후 분석을 진행하기로 협의
- 추가적으로 TAAS 교통사고 사망자 데이터와 생활인구 데이터를 활용하여 교통사고 현황 파악이 가능하며 횡단보도 데이터의 경우 아직 확보된 것이 없으므로 관련 자료를 찾아서 추후 안내하기로 함
- 데이터가 확보된 상태에서 여러 데이터를 결합하여 고령자 주요 밀집 지역 및 주요 활동 시간에 대한 시각화를 통해 현황을 파악할 수 있을 것으로 판단됨

07
컨설팅지원

데이터분야
공공데이터

☑ **수 행 일 자** : 2024.05.14.
☑ **소　　속** : 지자체
☑ **질 의 자** : 조○○
☑ **데이터분야** : SNS 데이터
☑ **지　　역** : 대구
☑ **응 답 자 명** : 김건욱 센터장

요청사항

- 대구시는 현재 3가지 언어권의 SNS 계정 (영어, 일어, 중국어)을 운영하고 있는데 관련 지표를 활용하여 이용자에게 인기 있는 콘텐츠 유형을 파악하고 싶음
- 현재 확인할 수 있는 데이터는 다음과 같음
  - 페이스북, 인스타그램 업로드 콘텐츠 본문
  - 게시물별 댓글과 DM(Direct Message)
  - 게시물별 조회수, 댓글, 공유, 좋아요 통계
  - 채널별 최다 접속 국가(도시) 통계
  - 채널별 접속 연령대, 성별 통계
- 분석 결과를 토대로 해외 SNS 콘텐츠 제작 시 중요한 근거 자료로 활용하여 이용자들에게 더욱 맞춤화된 콘텐츠를 제공하고자 함

08
컨설팅지원

데이터분야
공공데이터

- **수 행 일 자** : 2024.05.29.
- **소　　속** : 학생
- **질 의 자** : 박○○ 외 3인
- **데이터분야** : -
- **지　　역** : 대구
- **응 답 자 명** : 김건욱 센터장

지원내용

- 빅데이터 활용센터에서 수행한 SNS 분석 및 컨설팅 사례 소개
- SNS 관리자 계정 접속이 가능한지 문의 후 필요시 제공 받기로 협의
- 댓글 분석의 경우 현재 SNS 채널 댓글 수가 적어서 통상적인 분석에 그칠 가능성이 있음. SNS 관리자 페이지와 해외 소셜미디어 채널 운영 계획서나 월간보고서 등 다양한 자료를 참고하여 현황 파악을 위한 분석을 지원하겠음

요청사항

- 현재 학부 전공 수업에서 데이터 시각화 관련 과제를 진행하고 있고, 수업의 일환으로 실제 데이터 분석 환경에 대한 이해를 높이기 위해 빅데이터에 관심이 많은 미디어학과 학부생 및 유학생 대상으로 대구 빅데이터 활용센터 견학 및 소개 요청
- 빅데이터 활용센터에서 보유 중인 데이터 활용 가능 여부와 분석실 사용법에 대해 질의.

지원내용

- 센터에서 수행한 빅데이터 분석 및 컨설팅 사례를 소개해 주었으며 분석 시 사용한 툴과 데이터에 관련된 질의응답 진행
- 민간 데이터의 경우 외부에서 쉽게 구하지 못하는 한계점이 있기 때문에 학생들이 접할 수 있는 기회가 적음. 이에 센터에 방문 시 분석이 가능하다고 답변함
- 개인정보가 포함된 데이터의 경우 센터 내에 있는 가명 정보 활용지원센터를 방문하여 개인정보 일부를 삭제 또는 대체할 수 있음
- 또한, 빅데이터 분석에 관심 있는 학생들을 위한 경진대회 및 공모전이 진행될 예정임. 학생들의 참여를 적극적으로 독려하고 있으며 이를 통해 실제 데이터 분석을 경험할 수 있음

09
컨설팅지원

데이터분야
공공데이터

- **수 행 일 자** : 2024.06.12.
- **소　　속** : 시민
- **질 의 자** : 윤○○
- **데이터분야** : 교육
- **지　　역** : 대구
- **응 답 자 명** : 박연진 전임

요청사항

- 4학년 학생으로서 취업 준비 중임
- 최근 취업시장에서는 데이터 분석 관련하여 전문성을 요구하고 있음
- 학교에서 배우지 못한 데이터 분석 방법, 데이터 분석 공부 방법, 빅데이터 분석 트렌드 등을 배우고 싶음

지원내용

- 대구시 빅데이터 활용센터 운영 현황에 대한 설명: 빅데이터 활용센터는 다양한 데이터를 수집, 분석, 활용하여 정책 수립 및 지역 발전에 기여. 그동안 센터에서는 시민과 공무원을 대상으로 여러 차례 교육 진행. 교육 프로그램은 빅데이터의 기본 개념부터 고급 분석 기법까지 다양한 수준으로 구성. 앞으로도 지속적으로 교육 확대 예정
- 센터 내에서 자체적으로 제작한 교육 영상 소개 및 추천: 빅데이터 분석의 기초부터 심화 과정까지 다양한 내용을 포함. 교육생들이 언제 어디서나 쉽게 접근할 수 있도록 온라인 플랫폼을 통해 영상 제공
- 사례집을 주며 센터에서 진행했던 여러 분석 내용 소개
- 센터에서 진행 예정인 교육 내용에 대한 설명

교육명	교육일시(예정)
노코드 데이터 분석	2024.07.01. 13:30 ~
데이터 드리븐 디자인씽킹	2024.07.29. 13:30 ~
AI리터러시	2024.06813. 13:30 ~
챗GPT (오프라인)	2024.09.24. 13:30 ~
챗GPT (ZOOM)	2024.10.01. 13:30 ~

10
컨설팅지원

데이터분야
공공데이터

- **수 행 일 자** : 2024.06.20.
- **소　　속** : 시민
- **질 의 자** : 박○○
- **데이터분야** : 반도체
- **지　　역** : 대구
- **응 답 자 명** : 김현정 사무원

요청사항

- 신소재공학과 졸업생이며 반도체 생산/품질관리 직무를 목표로 하는 취업 준비생임
- 최근 많은 기업들이 자격요건 및 직무 요구사항으로 반도체 관련 전공지식뿐만 아니라 빅데이터 분석 역량을 요구하는 곳이 많음. 대학 전공수업의 일환으로 SW 기초 강의를 수강했으나 데이터 분석에 대한 이해도가 낮아 데이터 분석 관련 교육을 수강하여 직무 경험을 보완하고 역량을 키우고 싶음
- 추가적으로 센터에서 반도체 관련 데이터를 활용한 분석이 가능한지 문의

지원내용

- 대구 빅데이터 활용센터 소개, 분석실 보유 데이터 및 분석 사례 소개
- 교육에 대한 답변으로 현재 센터에서 비전공자들을 위한 데이터 리터러시 교육을 진행하고 있음. 데이터 분석에 필요한 기초 통계, 머신러닝/딥러닝 실습 등의 내용을 포괄적으로 다룰 예정이므로 이를 통해 데이터 분석 역량을 강화시킬 수 있음. 이 외에도 최신 데이터 분석 트렌드와 기술을 반영한 다양한 교육 프로그램을 지속적으로 진행할 예정이므로 수강하길 권유함
- 데이터에 대한 답변으로 현재 센터가 보유한 반도체 관련 데이터는 '반도체 설비 "클린 케이블" 양/불 이미지 데이터'가 있음. 이를 활용하여 클린 케이블 불량 탐지 예측 모델 개발 등 반도체 생산 및 품질관리에 필요한 다방면의 분석이 가능할 것으로 보임

11
컨설팅지원

데이터분야
공공데이터

☑ **수 행 일 자** : 2024.6.24.
☑ **소　　속** : 기업
☑ **질 의 자** : 김○○
☑ **데이터분야** : 유통
☑ **지　　역** : 대구
☑ **응 답 자 명** : 김현정 사무원

요청사항

- 제6회 대구 빅데이터 분석 경진대회 참여. 주제 선정 후 분석 시 활용 가능한 데이터 확인을 위해 방문
- 실제 분석 가능성 평가를 위해 활용 센터가 보유한 데이터에 대해 제공되는 샘플 데이터가 있다면 공식 홈페이지를 통해 공개 요청
- 분석 주제와 관련된 분석 작업의 일환으로 '대구로 배달앱 거래 이력 데이터'와 '대구로 배달앱 주문 데이터' 병합을 위해 두 데이터 간 '고객 코드' 컬럼 일치 여부 문의

지원내용

- 분석실에서 제공하는 데이터와 경진대회에서만 활용 가능한 데이터에 대해 설명함. 추가적으로 기존의 공공데이터/빅데이터 경진대회 수상작 및 분석 사례 소개
- 샘플 데이터 공개 요청에 대한 답변은 데이터를 제공한 기업들의 요청으로 샘플 데이터의 공개는 불가함. 대신, 빅데이터 활용센터 홈페이지에 게시되어 있는 **'대구 빅데이터 활용센터 활용 가능 데이터 설명서'**를 통해 데이터 구조 및 형식을 사전에 파악할 수 있으며 실제 RAW 데이터는 분석실을 방문하여 확인 및 분석이 가능함
- 대구로 데이터의 경우, 배달앱 거래이력 데이터의 '고객코드' 컬럼과 주문 데이터의 'cust_id' 컬럼이 일치함. 공통된 컬럼을 기준으로 두 데이터 프레임을 결합할 수 있으며, 이를 통해 가맹점별 거래 패턴, 주문 빈도, 매출액 분석 등 다양한 심층 분석이 가능하며, 보다 정교한 비즈니스 인사이트를 도출할 수 있음

12
컨설팅지원

데이터분야
공공데이터

- **수 행 일 자** : 2024.07.04.
- **소　　속** : 기관
- **질 의 자** : 전○○
- **데이터분야** : 사내데이터
- **지　　역** : 대구
- **응 답 자 명** : 박연진 전임

요청사항

- 사내 서비스 요청 사항 데이터 분석 통해 직원들의 빈번한 요구사항 발굴 등을 통하여 맞춤형 시스템 개선과 직원들 인식개선 및 교육 등으로 업무 효율성 제고
- 관련 하여 샘플 데이터를 바탕으로 분석기법 및 수행 방향, 정책 및 업무활용 방안에 대해 컨설팅 지원 요청

지원내용

- 텍스트 데이터를 활용한 토픽모델링은 대량의 문서에서 주요 키워드를 뽑아내고, 문서의 주제나 흐름을 파악하는 데 유용한 방법
- 이 방법론은 특정 주제나 분야에 대한 연구 동향을 분석하는 데 특히 효과적
- 예를 들어, 미래자동차와 관련된 연구 논문이나 기사들을 대상으로 토픽모델링을 적용하면, 해당 분야의 주요 이슈나 연구 방향을 파악할 수 있음
- 이에 유사한 분석결과가 센터 홈페이지에 등록되어 있음을 안내드렸음
- '토픽모델링을 활용한 국내외 미래자동차 연구동향 비교분석'이라는 연구에서는 국내외에서 진행된 미래자동차 관련 연구들을 대상으로 토픽모델링 기법을 적용하여, 각국의 연구 동향과 주요 키워드를 비교하였음
- 이 분석을 통해 각국의 연구 집중 영역, 주된 관심사, 그리고 연구의 발전 방향 등을 한눈에 파악 할 수 있음을 안내
- 따라서, 텍스트 데이터를 활용한 토픽모델링 방법론은 특정 분야의 연구 동향을 파악하고, 향후 연구 방향을 설정하는 데 있어 매우 유용한 도구임을 알려드림. 이와 관련된 구체적인 사례와 분석 결과는 센터 홈페이지를 통해 확인할 수 있음을 안내드림

13
컨설팅지원

데이터분야
공공데이터

☑ **수 행 일 자** : 2024. 7. 5.
☑ **소　　속** : 지자체
☑ **질 의 자** : 김○○
☑ **데이터분야** : 감염병 매개체
☑ **지　　역** : 대구
☑ **응 답 자 명** : 윤병진 전임

요청사항

- 매개체감염병 취약환경 분석을 통해 감염 방역망을 구축하는 과제를 진행 중이며, 이를 위해 각 구와 군의 방역 민원 접수 대장 및 유선 민원 내용을 데이터로 사용하고자 함
- 중구, 남구, 서구, 북구, 동구, 수성구, 달서구 등 7개 구와 달성군 1개 군에 대한 데이터를 별도의 형태로 수집하여 각 지자체마다 수집된 데이터에는 개인정보가 포함되어 있어 이를 분석 목적에 맞게 가명 및 익명 처리하는 것이 필요로 한 상황임
- 따라서 각 구와 군의 방역 민원 데이터을 검토하고 개인정보 식별 위험 평가를 실시하여 데이터의 가명 및 익명처리를 요청하였음.

지원내용

- 데이터를 전반적으로 검토하여 서구, 남구, 달서구에서 개인정보 이슈가 발생할 가능성이 있는 컬럼을 확인하였으며, 아래와 같은 작업을 수행하여 데이터를 안전하게 재가공 처리를 하였음
  - 서구 방역 민원 접수 대장: 성명, 연락처, 처리자 컬럼 삭제
  - 남구 방역 소독 민원 처리 대장: 민원인 성명, 연락처, 처리자 컬럼 삭제
  - 달서구 유선 민원 처리부: 민원인 연락처 컬럼 삭제
- 추가적으로, 개인정보 문제는 아니지만 데이터에 불결지나 민원 다발지의 정확한 주소가 기재되어 있는 것을 발견하였으며, 이러한 정보가 보안 없이 공개될 경우, 부동산 가치 하락, 사업 및 상권에 대한 악영향, 사회적 낙인과 같은 부작용이 발생할 수 있음을 안내하였음
- 따라서, 해당 과업에 대한 보안서약서를 작성하고 이를 준수하면서 분석을 진행할 것을 가이드라인으로 제시하였음

14
컨설팅지원

데이터분야
공공데이터

- **수 행 일 자** : 2024. 7. 9.
- **소　　속** : 학생
- **질 의 자** : 신○○, 김○○
- **데이터분야** : 복지
- **지　　역** : 대구
- **응 답 자 명** : 김현정 사무원

요청사항

- 제 6회 대구 빅데이터 분석 경진대회 데이터 시각화 분야 참여 고등학생임
- 빅데이터 활용센터가 보유한 '정신건강복지센터' 데이터를 활용하여 청소년 스트레스 지수가 높은 지역의 정신건강복지센터 위치를 시각화하는 것이 목적
- 주로 사용하는 분석 언어는 파이썬이며 '정신건강복지센터' 데이터를 확인한 결과 QGIS 프로그램에서 사용되는 geopackage(GPKG) 파일 형식임
- QGIS 및 geopackage 파일을 접해본 적이 없으며 이를 처리하는 방법에 대한 지원을 요청

지원내용

- 파이썬의 geopandas 라이브러리를 사용하여 geopackage 파일 형식을 파이썬으로 불러오는 방법을 안내. 데이터 셋을 전처리하는 방법과 각 레이어를 다루는 방법 설명
- 추가로 센터에서 진행했던 공간분석 및 지도 시각화 사례를 소개하고 학생들이 직접 다양한 시각화 기법을 적용해 볼 수 있도록 geopandas 및 folium을 활용한 시각화 예시 설명
- 센터 보유 데이터와 (예: 통신사 유동인구 데이터, 학교 정보 공시 데이터) 외부 공공 데이터(예: 나이스 교육정보 개방 포털의 학교 기본 정보 데이터) 등 추가 데이터의 통합 및 분석을 통해 보다 풍부한 인사이트를 도출할 수 있음을 안내
- 예시로 통신사 유동인구 데이터를 활용하여 특정 시간대나 지역의 유동인구 변화를 알 수 있으며 이를 활용한 다방면의 분석이 가능함을 안내

15
컨설팅지원

데이터분야
공공데이터

- **수 행 일 자** : 2024.7.11
- **소　　속** : 기관
- **질 의 자** : 박○○
- **데이터분야** : 교통
- **지　　역** : 대구
- **응 답 자 명** : 윤병진 전임

요청사항

- 현재 대구 빅데이터활용센터 분석실 내에서는 매년 나드리콜 운행 데이터를 받아 폐쇄망 내 분석실에 제공하였음
- 24년도 분석실 이용자에게 최신화된 나드리콜 데이터를 제공하고자 하나, 개인정보에 대한 중요도가 높아짐에 따라 제공하고 있던 데이터와 새로 제공할 데이터에 개인정보 이슈가 발생할수 있는지 검토하고 분석실 이용자들이 사용하기에 안전하도록 가명 및 익명처리를 요청하였음
- 따라서 기존에 가지고 있던 나드리콜 데이터의 재검토와 업데이트 되는 데이터의 스키마를 통일하고, 빅데이터활용센터 분석실에 데이터를 보관하려함

지원내용

- 나드리콜 운행 데이터는 고객번호, 차량번호, 배차일시, 호출장소, 목적장소 등의 정보를 포함하고 있으며, 이 중 '고객번호'와 '차량번호'는 고유식별자로 다른 변수들과 결합 시 개인식별 가능성이 존재하는 것으로 판단됨
- 특히 '배차일시', '호출장소', '목적장소' 등의 데이터는 조합될 경우 특정 개인을 식별할 수 있는 위험이 있으며, 이러한 위험을 줄이기 위한 관리적 조치가 필요함을 안내함
- '배차일시', '호출장소', '목적장소' 등의 시공간 데이터는 조합 시 특정 이용자를 식별할 우려가 있으나 해당 컬럼은 시공간적 분석을 위해 필수적인 데이터로 판단되었으며, 관리적·물리적 보안 조치를 통해 식별 가능성을 최소화하는 방향으로 데이터를 제공하는 것을 가이드라인을 제시함
- 나드리콜 데이터는 주로 공간 분석과 관련된 정보로, 재식별 시 직접적인 피해는 예상되지 않지만, 데이터를 안전하게 관리하기 위해 정기적인 모니터링과 물리적·관리적 보안을 강화하는 방식으로 개인정보 보호를 권장하였음

16
컨설팅지원

데이터분야
공공데이터

- **수 행 일 자** : 2024. 7. 15.
- **소 속** : 기관
- **질 의 자** : 전○○, 송○○, 이○○
- **데이터분야** : 사내데이터
- **지 역** : 대구
- **응 답 자 명** : 박연진 전임

요청사항

- 사내 서비스 요청 사항 데이터 분석 통해 직원들의 빈번한 요구사항 발굴 등을 통하여 맞춤형 시스템 개선과 직원들 인식개선 및 교육 등으로 업무 효율성 제고
- 관련 하여 샘플 데이터를 바탕으로 분석기법 및 수행 방향, 정책 및 업무활용 방안에 대해 컨설팅 지원을 받았으나, 구체적인 상담을 받고 싶어 재방문
- 분석한 방향이 목적 달성을 위한 올바른 분석 방법인지 문의

지원내용

- 기술적인 전반적인 분석방법은 맞다고 설명드림
- LDA 사용하면 되지만 최신 LDA 사용 필요
- 시계열 데이터가 있으니 시계열 분석도 추가하면 좋을 듯
- 토픽수를 5개로 정해 놓지말고 보통 토픽의 수를 정하는 방법론으로는 코히어런스(Cohereence)와 퍼플렉시티(Perplexity)가 있음을 설명. 이는 데이터마다 최적의 토픽 수를 정하는 데 사용. 해당 방법을 사용해서 토픽 수 정해서 분석 추천. 토픽 분석을 하여 결과를 2~3차원으로 시각화하여 점으로 표시함. 특정 영역에 어느 정도의 데이터가 있는지 직관적으로 파악할 수 있음. 예를 들어, 접수가 많은 영역에 특정 비율의 데이터가 있고, 다른 영역에도 비슷한 비율로 데이터가 분포되어 있다고 시각화할 수 있음. 이 시각화된 데이터를 통해 해당 영역에서 주로 발생하는 Q&A 주제를 파악할 수 있고, 이를 기반으로 특정 처리를 하면 문제 발생 비율이 얼마나 감소할지 수치로 예측 가능함 등을 설명 해드림

17
컨설팅지원

데이터분야
공공데이터

- **수 행 일 자** : 24. 7. 22.
- **소　　속** : 기업
- **질 의 자** : 김○○
- **데이터분야** : 생활인구데이터
- **지　　역** : 대구
- **응 답 자 명** : 김현정 사무원

요청사항

- 현재 2040년 도시기본계획 수립 중에 있으며, 이를 위해 계획인구를 산정할 필요가 있음. (*계획인구 : 도시기본계획 수립 시, 해당 도시나 지역의 장래 인구를 예측하여 설정한 목표 인구 수)
- 그러나, 기존 2030년 도시기본계획에서 예측한 인구 수가 현재에 못 미치고 있으며, 계속 감소할 전망임
- 이에 따라, 국토교통부는 도시기본계획 수립 시, 과도하게 부풀린 계획인구 산정을 방지하기 위해, 통계청의 추정치('2024년 계획인구' : 206만)를 기본으로 하고 대구시의 다양한 인구 데이터를 반영하도록 지침을 내림
- 빅데이터활용센터가 보유한 SKT 통신사 생활인구 데이터를 활용하여 2040년 계획인구 수를 추계할 방안 컨설팅 요청

지원내용

- 현재 센터가 보유한 통신사 생활인구 데이터는 '기준날짜연월일' 형태로 약 3년 간의 데이터임
- 이 데이터를 통해 시계열 패턴과 추세를 파악하여 이후를 예측할 수는 있으나, 3년 내외의 데이터를 기반으로 2040년까지의 계획인구를 추정하는 데는 논리적 한계가 있을 수 있음
- 따라서, 거주/방문/직장 인구로 구분된 통신사 생활인구 데이터를 기반으로 대구시 구 단위로 분석하고, 각 구분별 증가율을 비교한 결과를 제공하겠음. 이 자료를 계획 인구 산정 시 하나의 참고 자료로 활용하길 권장함
- 분석 계획은 아래와 같음
  - 1. 구 단위 분석 진행 : 각 구별 전체/거주/방문/직장 4개의 구분별로 월 평균 증가율을 산출하여 각 구별 인구 증가 추세를 파악
  - 2. 생활권 경계로 확대할 가능성 검토 : 구 단위 분석 결과를 확인한 후 증가율 수치의 타당성을 평가하여 생활권 경계로 확대여부 결정

18
컨설팅지원

데이터분야
공공데이터

- **수 행 일 자** : 2024. 7.30.
- **소 속** : 기관
- **질 의 자** : 박○○, 오○○
- **데이터분야** : 국가지식정보
- **지 역** : 대구
- **응 답 자 명** : 박연진 전임

요청사항

1. 자연어 검색어 필터링 방안
   - 검색 로그 데이터에 키워드 검색어와 자연어 스타일 검색어*가 혼재되어 있음. 특히 자연어 스타일 검색어는 사용자가 ChatGPT 프롬프트 입력 시처럼 문장형식으로 입력하는 검색어를 의미. 이로 인해 검색 로그 분석 시 두가지 유형의 검색어를 효과적으로 분류하는 것이 중요한 과제로 부각 되고 있음.
   - 자연어 스타일 검색어와 키워드 검색어를 명확하게 분류하기 위한 방안 필요. 구체적으로, 자연어 검색어를 식별하기 위한 텍스트 분석 기법이나 머신러닝 모델의 적용 가능성, 그리고 이러한 검색어를 효과적으로 분류할 수 있는 알고리즘 개발 방안을 검토요청. 필터링의 정확성과 효율성을 높이기 위해 필요한 데이터 전처리 및 특성 추출 방법에 대해서도 논의해 주시면 좋겠음.
     * ChatGPT 프롬프트 입력과 유사하게 문장형식의 검색어

2. 데이터 분석 계획 검토
   - 디지털집현전 사용자 데이터 분석 계획을 수립하였으나, 이 계획의 타당성을 검토하고 논리적 오류 및 부적절한 데이터 분석 방법에 대한 개선안을 도출하고자 함. 현재의 분석 계획은 사용자 행동 패턴을 이해하고 인사이트를 도출하기 위한 다양한 데이터 분석 기법을 포함하고 있으나, 실제로 데이터의 질과 분석 방법이 적절히 결합되어 있는지에 대한 검토가 필요
   - 디지털집현전 사용자 데이터 분석 계획의 전반적인 타당성 검토 요청. 특히, 사용되고 있는 분석 방법론의 적절성, 데이터 샘플링 및 분석 과정에서의 논리적 오류, 데이터의 정확성 및 신뢰성 확보 방안 등에 대해 깊이 있는 검토를 요청. 또한, 분석 계획의 개선점을 제시하고, 향후 데이터 분석의 정확성과 유용성을 높이기 위한 구체적인 방법론과 조치를 제안 요청

3. 기타사항
   - 오픈한 지가 오래되지 않아서 데이터가 많이 쌓여 있지 않음
   - 데이터들은 2억 4천만 건 있는데 사용자 데이터가 많지가 않음

19
컨설팅지원

데이터분야
공공데이터

- **수 행 일 자** : 2024. 8. 1.
- **소　　속** : 지자체
- **질 의 자** : 배○○
- **데이터분야** : 생활인구데이터
- **지　　역** : 대구
- **응 답 자 명** : 김현정 사무원

지원내용

- 아래와 같이 답변을 드림
- 토픽 모델링을 돌려가지고 군집화시켜서 유형별로 세그멘테이션에서 볼 수 있으면 나중에 대범주를 만들 때 우선순위를 정할 때도 도움이 되지 않을까 함
- 성별, 연령별, 또는 특정 키워드 등 에서 진입 장벽이 있는 것을 파악하여 진입 장벽을 낮추면 좋겠음. 또한, 그런 데이터를 기반으로 중점적으로 튜토리얼을 만드는 것이 좋을 것 같음
- 리텐션하는 유저와 이탈하는 유저를 분리해서 분석하는 것이 의미가 있을 수 있을 듯. 재방문율이 낮다고 하셨는데, 그 재방문율이 어느 정도인지 구체적인 수치가 필요. 일반적으로 고객 이탈 분석에서는 이탈을 정의하는 기준을 설정. 이 기준은 분석 목적에 따라 달라질 수 있음. 만약 자체적으로 이탈을 정의 하고, 리텐션하는 유저와 이탈하는 유저를 비교해 보면, 아마 키워드나 패턴 이 다를 수 있을 것이며, 이를 통해 논리를 도출할 수 있을 것
- 키워드 질의어랑 문장형식 질의어 구분은 품사 태깅에다가 룰기반 또는 특수문자 또는 직접 라벨링 천 개 정도로 해서 AI 학습하는 방법이 있을 듯

요청사항

- 논공읍 지역 발전 방안 구상 용역 수행 중, 주민들의 요구사항과 의견을 반영하기 위해 설문조사와 병행하여 빅데이터 기반의 소셜 미디어 분석을 통한 자료의 신뢰성 강화 필요
- '논공읍'이라는 키워드를 중심으로 소셜 미디어 상의 언급 빈도 분석을 통해 지역 관련 주요 관심사를 시각화(워드 클라우드)하고, 이를 통해 지역 발전 계획 수립 시 참고자료로 활용하고자 함
- 소셜 미디어(인스타그램 등) 및 웹 상의 공개된 데이터를 크롤링하여, 논공읍과 관련된 키워드 언급 빈도를 분석하고, 이를 바탕으로 시각화(워드 클라우드)한 결과를 제공해 줄 것을 요청함

지원내용

- 크롤링은 주로 인스타그램, 블로그, 트위터 등의 소셜 네트워크 플랫폼과 블로그 기반 데이터를 대상으로 하며, 최근 일정 기간 동안의 데이터를 분석 대상으로 함
- 그러나, 크롤링 결과는 데이터 수집 대상 플랫폼의 특성, 수집 기간, 데이터 필터링 방식 등에 따라 상이한 결과를 도출할 수 있음. 또한, 각 플랫폼의 API 접근 제한, 데이터 사용 정책 등을 준수해야 하며, 이에 따른 기술적 한계가 존재할 수 있음
- 따라서, 최근에는 크롤링 전문 업체가 많아져, 내부에서 직접 크롤링을 수행하기보다는 '썸 트렌드'(소셜 네트워크 및 블로그 기반), '빅카인즈'(한국언론진흥재단) 등과 같은 외부 전문 서비스를 활용할 것을 권장
- 현재 센터에서는 자체적인 크롤링 및 데이터 분석 지원을 제한적으로 제공하고 있으며, 전문 업체의 서비스 활용을 통해 보다 효율적이고 정교한 분석 결과를 얻는 것이 바람직함

20
컨설팅지원

데이터분야
공공데이터

☑ **수 행 일 자** : 2024.08.16.
☑ **소　　속** : 기관
☑ **질 의 자** : 오○○
☑ **데이터분야** : 자연어처리
☑ **지　　역** : 대구
☑ **응 답 자 명** : 윤주혁 사무원

요청사항

- 코드 리뷰 관련 온디바이스 로컬 환경에서의 SLLM 언어모델 생성방안 컨설팅 요청
- 현재 코드 리뷰 및 코드 생성 전문 오픈소스 언어모델을 목적에 맞게 파인튜닝한 후 온디바이스 모델로 탑재가 가능하도록 경량화 작업을 하고자 함
- 현재 직면한 문제점은, 모델 경량화를 위한 양자화에 따른 성능 감소 수치가 예측되지 않는다는 점임
- 또한, 양자화로 인해 성능이 감소되었을 경우 이를 보완할 수 있는 방안이 필요함

지원내용

- 양자화는 모델의 메모리와 계산 복잡도를 줄여 온디바이스 환경에서의 실행 가능성을 높이기 위해 중요한 작업임. 하지만, 양자화 과정에서 32비트 부동소수점 연산을 8비트 정수 연산으로 변환하는 등의 과정에서 모델의 정밀도와 성능이 저하되는 문제가 있음

- 양자화를 적용하기 전 사전 시뮬레이션을 통해 양자화가 모델 성능에 미치는 영향을 예측할 수 있음. 이를 통해 양자화의 정도를 조정하거나(Trade-Off 관계), 특정 레이어에만 양자화를 적용하는 등의 방법론이 존재함
- 모델 성능 평가를 위한 기준 지표(BLEU Score, ROUGE 등)를 설정하고, 양자화 후 지표 기반 평가 또한 가능하여 성능 감소 수치에 대한 정량적인 수치를 생성 가능함
- 또한, 모델 전체에 동일한 양자화를 적용하는 대신, 모델 성능과 관련된 레이어에는 높은 비트를, 덜 중요한 레이어에는 낮은 비트를 적용하는 혼합 정밀도 양자화를 기법을 적용한다면 성능 저하를 최소화할 수 있음. 그러나, LLM 모델 특성상 풀 오픈소스 모델이더라도, 레이어 구조가 상당히 복잡하기 때문에 레이어 간 중요도에 따른 판별기준을 수립하는데 오랜 시간이 걸릴 수 있음
- 레이어 간 중요도에 따른 양자화 처리방식 이외에도, 양자화 후 재학습하는 후처리 기법(Fine-Tuning)을 통해 양자화로 인한 성능 저하를 일정부분 복구할 수 있음. 특히, 질의자가 요청한 고정된 작업(코드 생성 및 코드리뷰 등)에서 일관된 성능을 유지하는 데 효과적인걸로 알려져 있음
- 제안한 방법 이외에도, 오픈소스 LLM 모델들을 SLLM으로 변환하여 온디바이스에서 구동이 가능하도록 시도한 사례는 상당히 많음. 그러나 LLM 모델의 구조 또한 중요하지만, 학습에 사용할 데이터의 품질 및 용량 또한 모델의 성능에 큰 영향을 미치므로 이 점을 고려하여 모델 개발의 방향성을 잡아갈 수 있도록 안내함

21
컨설팅지원

데이터분야
공공데이터

- **수 행 일 자** : 2024.08.20.
- **소　　속** : 학생
- **질 의 자** : 최○○
- **데이터분야** : 관광
- **지　　역** : 대구
- **응 답 자 명** : 윤주혁 사무원

요청사항

- 텍스트 데이터 분석을 위한 한국어 전처리 및 토큰화 라이브러리 설치 요청
- 자연어처리 후 워드클라우드를 통한 시각화 기법을 적용하고자 함
- 자연어처리 및 워드클라우드에 대한 전반적인 분석 및 시각화 프로세스 및 적용 과정에 대한 기본적인 안내 요청

지원내용

- 활용센터 이용자들에게 제공하는 분석 가상환경은 기본적으로 윈도우 환경으로 제공됨. 간단한 텍스트 분석시에 활용되는 라이브러리들은 윈도우 환경에서는 추가적인 환경설정(jdk 설치 및 JAVA_HOME 패스 설정 등)이 필수적임. 따라서, 윈도우 환경에서 텍스트 분석시 환경 설정방법을 안내함
- 워드클라우드 기법은 주로 텍스트 데이터 내에서 주요 주제를 직관적으로 파악하는데 유용한 기법임. 주로, 분석 대상 데이터 선정 ⇒ 전처리(텍스트 클렌징, 명사or형태소 추출 등) ⇒ 토큰화 ⇒ 시각화의 단계로 이루어짐
- 워드클라우드 라이브러리 내부에는 다양한 시각화 도구가 있으므로(클라우드 형태 지정, 커스텀 배경 이미지 기반 클라우드 생성 등), 이러한 기능들을 소개 및 분석하고자 하는 목적에 맞춰 이를 적절히 사용할 수 있도록 안내함

22
컨설팅지원

데이터분야
공공데이터

- **수 행 일 자** : 2024.08.23.
- **소　　속** : 시민
- **질 의 자** : 김○○ 외 5인
- **데이터분야** : 교육
- **지　　역** : 대구
- **응 답 자 명** : 김현정 사무원

요청사항

- 삼성 청년 SW 아카데미(SSAFY) 교육생으로서, 4차 산업혁명 관련(빅데이터) 특화 프로젝트를 준비 중임
- 프로젝트 주제를 정하기에 앞서, 관련 데이터 활용에 대한 실질적인 이해를 높이고자 대구 빅데이터활용센터 견학을 신청함
- 또한 센터의 이용 방법과 실제 빅데이터 분석 사례를 통해 기업에서의 데이터 활용 방식과 비즈니스 인사이트 도출 과정을 탐구하고, 이를 통해 프로젝트의 완성도를 높이고자함

지원내용

- 대구 빅데이터 활용센터에 대한 전반적인 소개와 함께, 교육생들이 준비 중인 빅데이터 프로젝트에 적용할 수 있는 센터의 다양한 데이터 및 분석 도구에 대한 구체적인 정보를 제공함
- 센터에서 수행한 다양한 분석 사례, 예를 들어, 3D 데이터 딥러닝 학습, 나드리콜 대기시간 예측 등의 프로젝트 사례를 소개하여, 실제로 기업에게 데이터 분석을 어떻게 지원하고 있는지를 설명함
- 실제 분석실 방문 시, 가상환경에 접속하여 데이터를 분석하는 과정을 시연함으로써, 교육생들이 센터에서 제공하는 데이터를 활용하는 방법을 직접 체험할 수 있도록 지원함. 이를 통해 프로젝트에 적용할 수 있는 다양한 방안을 모색할 수 있도록 도움
- 아울러 지역에서 진행되는 실제 데이터 분석 과정과 그 결과가 비즈니스 인사이트로 어떻게 연결되는지를 상세히 설명하여, 교육생들이 향후 프로젝트 방향성을 구체화할 수 있도록 지원

23
컨설팅지원

데이터분야
공공데이터

- **수 행 일 자** : 2024.08.23.
- **소　　속** : 기업
- **질 의 자** : 조○○
- **데이터분야** : 유통
- **지　　역** : 대구
- **응 답 자 명** : 윤병진 전임

요청사항

- 유통기업인 A사는 개인 맞춤형 마케팅 시스템과 고객 이탈 관리 프로그램 구축에 어려움을 겪고 있으며, 특히 데이터 전문 인력의 부재로 인해 개인별 맞춤 서비스를 제공하는 데 한계가 있다고 판단하여 데이터 기반 문제 해결을 위한 컨설팅을 요청하였음

*지역 식자재 전문 마트로 전국 17개 매장, 8개의 물류센터 보유

- 현재 기업은 POS 시스템을 통해 고객 데이터를 수집하고 있으나, 동일 고객의 다른 시기 결제 여부를 파악하기 어려운 상황이어서, 이를 해결할 구체적인 방안에 대한 자문을 요청하였음
- 국내 주요 유통기업들은 멤버십과 포인트 적립 시스템을 통해 고객 맞춤형 마케팅을 성공적으로 운영 중이며, 역시 이와 같은 성공 사례를 참고하여 개선 방안을 모색하고 있으나, 구체적인 프로세스를 수립하기 위해 이번 컨설팅을 진행하고자 함

지원내용

- A사의 강화를 위해 POS 데이터를 카드사 데이터와 결합하여 고객을 식별할 수 있는 방안을 검토하였으며, 이를 위해 데이터 처리 및 매핑 기술을 안내하여 동일 고객의 구매 이력을 효과적으로 추적할 수 있도록 지원하였음
- 주요 사례로 스타벅스와 이마트의 고객 맞춤형 마케팅 전략을 설명하였으며, 이들 기업은 앱과 멤버십을 통해 수집된 고객 데이터를 분석하여 맞춤형 혜택을 제공하고, 구매 전환율을 높이는 데 성공한 바 있음. 유사한 시스템을 구축하여 매출 증대와 고객 충성도 향상을 목표로 할 것을 권장하였음
- 삼성카드는 고객 데이터를 활용하여 고객 이해도와 비즈니스 성과를 극대화할 수 있으며, 이를 통해 고객 맞춤형 마케팅을 강화하고 지역 데이터 산업의 활성화에도 기여할 수 있을 것으로 기대됨
- 이후, 대구 가명정보활용지원센터에서 가명 결합 절차 및 적정성 평가에 대한 지원을 제공할 예정임

24
컨설팅지원

데이터분야
공공데이터

- **수 행 일 자** : 2024.09.20
- **소　　속** : 기업
- **질 의 자** : 김○○
- **데이터분야** : 자동차 부품
- **지　　역** : 대구
- **응 답 자 명** : 김현정 사무원

요청사항

- 현재 국내외 주요 자동차에 장착되는 자동차용 LED 및 컨트롤 모듈, 자동차용 무선충전 모듈, 기어변속 모듈, 식물공장용 LED, 스마트 보청기용 PCB 등을 생산 및 납품 중임
- 납품 업체의 특성상 제품 불량률 감소 및 납기일 준수가 중요한 과제이며, 이를 위해 부분적으로 품질 관리 시스템(MES)을 도입하여 공정별로 실시간 모니터링을 진행하고 있으나, 공정별 기계 제조사가 달라 통합 관리 시스템의 개발이 필요함
- 각 공정에서 발생하는 데이터는 외부 업체를 통해 관리하고 있어, 데이터 활용에 어려움이 있으며 1차 벤더사들의 요청 데이터 형식이 상이하여 데이터 표준화 및 적재 방식을 결정하는 데 문제가 있음
- 이에 따라 공정별 통합관리 시스템 도입과 데이터 적재 방식 및 표준 수립에 대한 자문 요청

지원내용

- SMD 생산 라인 점검 결과, 공정 프로세스의 최종 단계인 검출기에서 에러 발생 빈도가 높아, 이 문제를 해결하기 위해 추후 기계 교체에 대비한 데이터 백업 및 이력 관리 시스템 구축의 필요성을 설명함
- 각 공정별 데이터는 적재되고 있으나 데이터 현황 파악이 되고 있지 않음. 추후, 실제 DB 확인 후 논의하기로 함
- 데이터 표준화 및 통합 관리를 위한 전략으로, 데이터 관리에 대한 명확한 규정 및 원칙 수립이 필요함. 이를 위해서는 산업 표준과 정부 규정에 맞는 데이터를 수집하고, 1차 벤더사, IT 공급업체 등 다양한 이해관계자들의 협력이 필요함. 따라서 정부 지원 프로그램을 적극적으로 활용할 것을 권장함

25
컨설팅지원

데이터분야
공공데이터

- **수 행 일 자** : 2024.09.24
- **소　　속** : 학생
- **질 의 자** : 정○○
- **데이터분야** : 신한카드데이터
- **지　　역** : 대구
- **응 답 자 명** : 김현정 사무원

요청사항

- 소프트웨어 경진대회에 참여할 예정인 학생임
- 이를 위해 대구빅데이터활용센터의 신한카드 데이터를 활용한 프로젝트를 구상 중이며 프로젝트의 대략적인 설계 과정은 아래와 같음
  - 1. 리뷰 데이터를 기반으로 BERT 모델을 활용한 군집화를 통해 주요 키워드 도출
  - 2. 카드 데이터에 대해 머신러닝 기반 클러스터링을 적용하여 소비 패턴 분석
  - 3. 도출된 결과를 바탕으로 관광지 주요 키워드 및 소비 패턴을 직관적으로 시각화하는 웹페이지 개발 예정
- 프로세스 진행 중, 카드 데이터 분석에 적합한 머신러닝 모델 선택에 대해 고민하고 있으며, 이에 대한 전문적인 컨설팅을 요청함

지원내용

- 머신러닝 모델 선택과 관련하여, XGBoost와 LightGBM 모델의 장단점에 대해 설명하였으며, 앙상블 기법이 데이터 분석 성능을 높이는 데 유리한 점을 강조함
- 초기 단계에서는 랜덤포레스트(Random Forest)와 같은 간단한 모델을 사용해 빠르게 결과를 확인하고, 이후 프로젝트 진척에 따라 모델을 고도화할 것을 권장함. 이를 통해 다양한 모델을 실험하고 성능을 비교하는 것이 필요함
- 추후 모델 고도화 시에는 하이퍼파라미터 튜닝과 교차 검증(Cross Validation)을 통해 최적화된 모델을 도출하는 방법을 함께 논의할 것을 제안함

26
컨설팅지원

데이터분야
공공데이터

- **수 행 일 자** : 2024.09.25.
- **소　　속** : 학생
- **질 의 자** : 이○○
- **데이터분야** : 리뷰데이터
- **지　　역** : 대구
- **응 답 자 명** : 윤주혁 사무원

요청사항

- SNS 채널들을 통해 수집되어 빅데이터 활용센터에서 보유하고 있는 리뷰 데이터를 유사한 데이터끼리 효과적으로 군집화 하여, 주요 패턴 및 트렌드를 도출하고자 함
- 도출된 패턴 및 트렌드를 리뷰데이터의 위치 정보(장소명, 지명, 주소 등)을 기준으로 카드 데이터와 추후 결합하여 인사이트를 도출 및 API 개발을 하는데 참고하고자

지원내용

- 기획 초기에는 텍스트 분석에 사용할 모델을 BERT모델로 활용하고자 하였음. 이는 BERT의 뛰어난 문맥 이해 능력과 사전 학습된 언어 모델의 강점을 살려 신뢰성 높은 분석 결과를 만들어내고자 했기 때문임
- 따라서, 한국어 데이터셋으로 파인튜닝된 BERT 모델을 활용하여 리뷰 텍스트를 임베딩 벡터로 변환한 후 클러스터링 알고리즘을 적용하는 것을 추천하였음
- 그러나, 추가적인 컨설팅을 통해 질의자가 이용하고자 하는 BERT 모델이 주어진 분석 환경 및 기간에 비해서는 높은 계산 복잡도로 인해 효율성이 낮아 학습 및 처리 시간이 길어질수도 있다는 문제를 인식함
- 이용할 수 있는 컴퓨팅 자원이 한정적이고, 특수 목적(임베딩 추출 및 학습용)을 위해서는 더 가벼우면서도 기대만큼의 군집화 정확도를 이끌어낼 수 있는 모델을 선택하는 것이 필요했음
- 따라서 대안으로 사용할 수 있는 모델을 검토 및 선택한 결과 ELECTRA 모델을 사용하는 것을 제안함
- 이 모델은 BERT와 동일하게 생성기(generator)와 검출기(discriminator)로 이루어진 Transformer 기반의 모델이지만, 전체 단어를 대상으로 학습하여 더 빠르고 효율적인 학습 가능하다는 장점이 있음.
- 따라서 BERT 대비 학습 단계에서 더 높은 효율성과 성능을 기대할 수 있으며 상대적으로 적은 계산 자원으로 동일한 또는 더 나은 결과를 도출할 수 있다는 장점이 있음

27
컨설팅지원

데이터분야
공공데이터

- **수 행 일 자** : 2024.10.21.
- **소　　속** : 기업
- **질 의 자** : 김○○
- **데이터분야** : 자연어 처리
- **지　　역** : 대구
- **응 답 자 명** : 윤주혁 사무원

요청사항

- 당사는 디자인 및 웹/모바일 페이지 개발을 전문으로 하는 기업임. LLM(대규모 언어 모델)을 활용한 신규 서비스를 기획하고자 함
- 은행 업무나 관공서 홈페이지를 방문할 때 원하는 업무를 쉽게 찾지 못하는 경우가 많음. 블로그나 다른 사용자의 글을 참고해도 UI나 명칭이 다르게 표시되어 혼란을 겪거나, 제공된 매뉴얼이 최신화되지 않아 필요한 정보를 찾는 데 과도한 시간이 소요되기도 함
- 이에 따라, 사용자가 특정 상황을 설명하면 AI가 즉시 관련 페이지로 이동할 수 있는 새로운 솔루션을 LLM 모델을 활용하여 개발하고자 함
- 현재 오픈소스 LLM 모델을 활용한 초기 프로토타입 모델 개발을 계획 중이며, 구체적으로 어떤 데이터를 사용할지 고민이 필요한 상황임

지원내용

- **데이터 활용 및 전처리**
  - 기본적으로 AI hub의 공개 데이터인 “한국어 대화, 상담 음성, 고객 응대 음성”의 텍스트 또는 음성을 제외한 전사된 텍스트 파일을 사용하는 것이 개발 목적과 목표에 가장 부합함을 안내함
  - 은행, 관공서, 신청서 작성 등 정보가 필요한 업무 처리 공간의 고객 소리함 또는 고객센터 자료 등을 통해 고객이 어떤 질문을 했고, 그에 대한 답변이 무엇이었는지에 대한 데이터를 수집해야 함. 이는 1차적으로 대화의 맥락 이해와 2차적으로 대화 상태에 따른 도메인 기반 대화 주제 분류로 이루어져야 함
  - 따라서 AI hub에 공개된 대화 관련 데이터를 활용해 1차적으로 대화의 맥락 및 주제 파악 기능을 강화하고, 2차 목표에 맞춰 도메인 최적화된 데이터셋을 수집 및 전처리하는 것을 제안함
- **LLM 모델 활용**
  - Llama 3.1은 세계 최대 규모의 오픈소스 모델로, 다국어 지원과 강력한 추론 능력을 바탕으로 하고 있음
  - 오픈소스이므로 개발에 필요한 정보와 레퍼런스를 얻기가 용이하며, 유지보수와 고도화에도 적합함
  - 또한, 임베딩 추출, 멀티 모달, 온 디바이스 등 유연하고 다양한 기능을 포함하고 있어, 기획하는 서비스를 구축하고 고도화하는 데 매우 적합하므로 이를 제안함

28
컨설팅지원

데이터분야
공공데이터

☑ **수 행 일 자** : 2024.10.21.
☑ **소　　속** : 기업
☑ **질 의 자** : 임○○
☑ **데이터분야** : 경매데이터, 군집 분석
☑ **지　　역** : 대구
☑ **응 답 자 명** : 윤주혁 사무원

요청사항

- 고객 행동 데이터를 바탕으로 고객 맞춤형 경매 물건 추천 시스템을 구축해 보려고 함
- 고객의 관심 물건과 검색 이력, 매물 속성 정보를 분석하여 개인화된 물건 추천을 제공하는 것이 목표임
- 개인정보를 활용하지 않고도 고객 행동 데이터를 기반으로 서비스 개선 및 추천 알고리즘을 도입할 수 있는 가능성이 있는지 문의함
- 고객의 검색 이력과 관심 매물 정보에 기반하여 맞춤형 물건 추천 시스템을 개발함으로써 매각 성공률과 고객 만족도를 제고하는 것을 목표로 함

지원내용

- **데이터 결합 방법론 안내 및 지원**
  - 두 개의 정형 데이터를 특정 키 값을 기준으로 결합할 때 사용할 수 있는 다양한 방법론을 안내함. 주로 사용하는 방법인 inner join, left join, outer join에 대해서 설명함
  - Inner Join: 두 데이터셋에서 키 값이 일치하는 행만 결합하여 결과에 포함함.
  - Left Join: 기준 데이터셋(왼쪽)의 모든 데이터를 유지하며, 오른쪽 데이터셋에서 키 값이 일치하는 데이터를 결합함
  - Outer Join: 두 데이터셋의 모든 데이터를 포함하며, 일치하지 않는 경우에는 해당 위치에 빈 값으로 처리함
  - 데이터를 검토한 결과, 유저의 열람 기록 데이터를 기준으로 left join을 수행하는 것이 적합하다는 의견을 제시함. 유저의 열람 기록을 모두 포함하면서, 결합되는 다른 데이터셋에서 추가 정보를 불러올 때 필요한 데이터만 포함되게 하려고 함
- **머신러닝 기반 군집 분석 알고리즘 종류 및 특징별 추천 안내**
  - 군집 분석은 비지도 학습으로 데이터를 유사한 속성끼리 그룹화하는 방식임.
  - 각 군집 분석 알고리즘은 데이터의 특성에 따라 적합한 유형이 나뉘며, 주요 방식은 다음과 같이 나뉨을 안내하고, 각 모델을 적용하기 적합한 사례를 제시함
  - Partitioning-Based (분할 기반): K-means, K-medoids와 같은 알고리즘이 대표적임. 데이터의 중심을 기준으로 미리 정한 클러스터 개수로 데이터를 분할하는 방식이므로, 원형 혹은 구형으로 분포를 보이는 데이터 혹은 균등

한 분포를 보이는 데이터에 적합함

- Density-Based (밀도 기반): DBSCAN, OPTICS 알고리즘이 있음. 밀집된 영역을 클러스터로 간주하고, 밀도가 낮은 영역을 노이즈로 처리함. 비선형 구조(복잡한 구조)나 노이즈가 포함된 데이터에 유리함
- Model-Based (모델 기반): Gaussian Mixture Models(GMM) 등 특정 통계적 분포를 가정하여 클러스터를 형성함. 데이터의 통계적 특성이 명확할 때 적합함

29
컨설팅지원

데이터분야
공공데이터

- **수 행 일 자** : 24.10.21
- **소 속** : 기업
- **질 의 자** : 허○○
- **데이터분야** : 지능형 영상 감시 시스템
- **지 역** : 대구
- **응 답 자 명** : 김현정 사무원

요청사항

- 현재 당사가 보유한 지능형 영상감시시스템에 대해 연구개발을 지속 중이며, 한국인터넷진흥원에서 시행하는 지능형 CCTV 성능 인증을 취득하여 사업화하고 있음
- 화재, 쓰러짐, 군집, 익수 감지 등의 신규 분야를 개발하여 사업 영역을 확장하고자 하나, AI Hub 등을 통한 데이터 확보에는 한계가 있으며 자체 데이터 제작도 어려운 상황임
- 또한, 온디바이스 환경에서 파이썬 기반 모델의 성능 저하 문제로 인해 C++ 등 low-level 언어로의 모델 전환을 고려 중임
- 이에 따라, 데이터 확보 방안과 모델 개선 방향에 대한 조언을 요청함

지원내용

- 데이터 확보 방안으로는 기존 보유 데이터와 결합하거나 데이터 증강 기법을 활용하는 방안을 권장함. 예를 들어, 자체 수집한 영상 데이터를 증강 기법을 통해 다양한 상황에 맞게 변형하여 학습 데이터로 활용할 수 있음
- 또한, KISA, 경찰청 등 지능형 CCTV 관련 인증을 받은 기관과의 협력 또는 특정 데이터 수집 프로젝트 제안을 통해 목표에 맞는 데이터를 확보하는 것도 한 가지 방법임
- 모델 개선 방안으로는 온디바이스 환경에서의 성능을 극대화하기 위해 파이썬 기반 모델을 C++ 또는 기타 고성능 low-level 언어로 변환하는 것이 효과적일 수 있으나, 변환 과정에서 구현과 유지보수 측면의 추가 비용과 시간이 발생할 수 있음
- 따라서 각 CCTV 카메라가 영상 데이터를 송출하고 중앙 제어 시스템에서 연산을 집중적으로 수행하는 방식을 채택하여 디바이스 리소스를 절감하는 방안을 권장함

30
컨설팅지원

데이터분야
공공데이터

- **수 행 일 자** : 24.10.31
- **소 속** : 기업
- **질 의 자** : 황○○
- **데이터분야** : 3D 이미지 데이터
- **지 역** : 대구
- **응 답 자 명** : 김현정 사무원

요청사항

- A는 VR과 EEG를 결합한 사회정서지원 에듀테크와 메타버스 기반 3D 디지털 콘텐츠 등 다양한 서비스를 제공하는 스타트업으로, 아동·청소년 상담 데이터를 활용해 맞춤형 서비스와 제품 고도화를 위한 방안을 찾고자 함
- 현재 보유한 데이터는 CSV 파일 형식으로 저장되어 있으며, 텍스트 형태의 상담 데이터와 각도별 값이 포함된 3D 이미지 데이터 등 여러 유형이 있음 앞으로 데이터베이스 구축 시 이 데이터를 정형화해 저장하는 방안과, 데이터 분석을 통해 맞춤형 서비스를 개발하는 데 필요한 조언을 요청함

지원내용

- 서비스 개발 측면에서는 아동·청소년 정서 데이터를 분석하여 메타버스 기반 3D 콘텐츠와 결합하는 방법을 제안함. 가상 환경에서 정서적 반응을 실시간으로 시각화하고 피드백을 제공하는 인터랙티브 기능을 구현하면, 사용자 몰입도와 교육 효과를 높일 수 있을 것으로 기대됨. 이를 통해 맞춤형 정서 지원 콘텐츠를 제작하여 제품의 경쟁력을 강화할 수 있음
- 데이터 저장 측면에서는, 데이터베이스 구축 시 연령, 성별 등의 메타데이터와 3D 이미지 각도별 값을 구분해 정형화된 형태로 저장하는 방안을 권장함. 이를 통해 데이터 접근성과 분석 일관성을 높이고, 보다 세밀한 분석이 가능해짐
- 각도별 이미지 데이터를 개별 필드로 나눠 저장할 경우 특정 각도에서의 정서 반응 패턴을 더욱 구체적으로 파악할 수 있음. 이를 통해 연령대나 성별에 따른 차별화된 분석이 가능해져, 맞춤형 에듀테크 서비스 개발에 필요한 통찰을 제공할 수 있을 것으로 기대됨


대구빅데이터활용센터
DAEGU BIG DATA