

끊임없이 도전하는 개발자, 유재광입니다

CONTACT

marmong9770@gmail.com

010-9219-5630





도전하는 개발자 유재광입니다

안녕하세요, 끊임없이 도전하는 개발자 유재광입니다.
컴퓨터공학을 전공하며 데이터 분석과 백엔드 개발에 깊은 열정을 가지고 있습니다.
다양한 프로젝트 경험을 통해 문제 해결 능력과 팀워크를 배양했으며, 항상 새로운 기술을 배우고 성장하는 것을 목표로 하고 있습니다.

유재광 / JaeGwang Yu

1997.05.29 서울 특별시

Tel. 010-9219-5630

Email. marmong9770@gmail.com

서울특별시 송파구 오금로 44길 8

Git : <https://github.com/JAEKWANG97>

GRADUATION

2016 문정고등학교 졸업

2017 협성대학교 컴퓨터공학과 입학

2024 협성대학교 컴퓨터공학과 졸업

PROJECT

2023 대피소 할당 최적화 프로젝트

2023 데이콘 추석 맞이 추석 선물 수요량 예측 AI 해커톤 (Private 5등 달성)

2024 여행 정보 제공 및 계획 수립 웹 프로젝트

SKILL

Vue.js, JavaScript, Python, Java,
Spring Boot, MySQL, Elasticsearch, Git,
MyBatis, AWS (S3, EC2, RDS), Docker,
Swagger

유재광은 어떤 능력을 가지고 있을까?

다양한 백엔드와 프론트엔드 개발, 데이터 분석 및 처리, 클라우드 및 DevOps 기술을 보유한 개발자로, 다음과 같은 능력을 갖추고 있습니다

백엔드 개발 및 운영

- 백엔드 개발에 대한 이해도와 실제 서비스 개발 및 운영 경험 보유
- Spring Boot, Java를 활용한 프로젝트 수행 경험
- RESTful API 설계 및 구현 능력

데이터 분석 및 처리

- Python과 pandas를 사용한 데이터 수집, 전처리 및 분석 능력
- MySQL, Elasticsearch를 통한 데이터베이스 관리 경험

클라우드 및 DevOps 기술

- AWS(S3, EC2, RDS)를 이용한 클라우드 인프라 구축 및 운영 경험
- Docker를 활용한 컨테이너화 및 CI/CD 파이프라인 구축 능력
- Git을 통한 버전 관리 및 협업 능력

프론트엔드 개발

- Vue.js 및 React.js를 사용한 프론트엔드 개발 경험
- JavaScript를 이용한 동적 웹 애플리케이션 개발 능력
- 사용자 중심의 인터페이스 설계 및 구현 경험

PROJECT.1

여행 정보 제공 및 여행 계획 수립 프로젝트

PINTRAVEL

01

ABOUT PROJECT

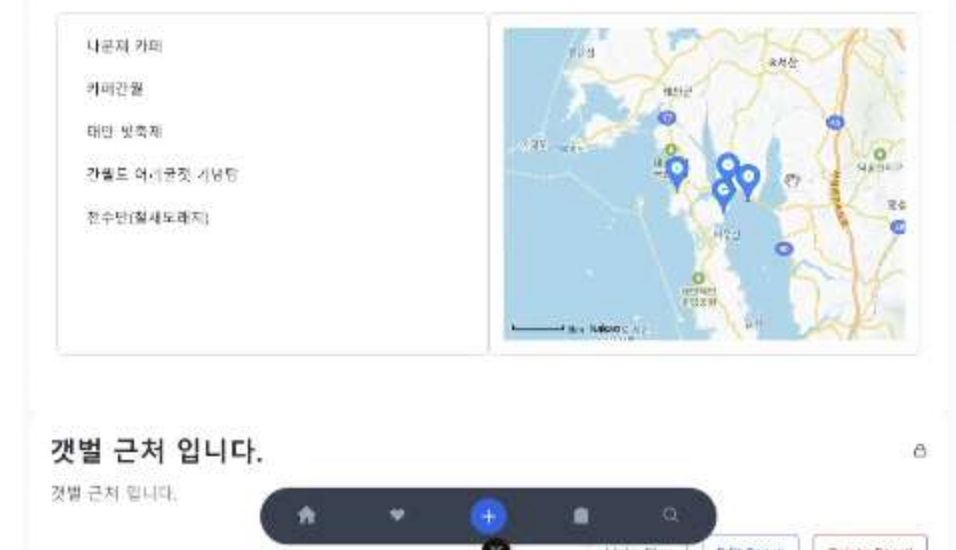
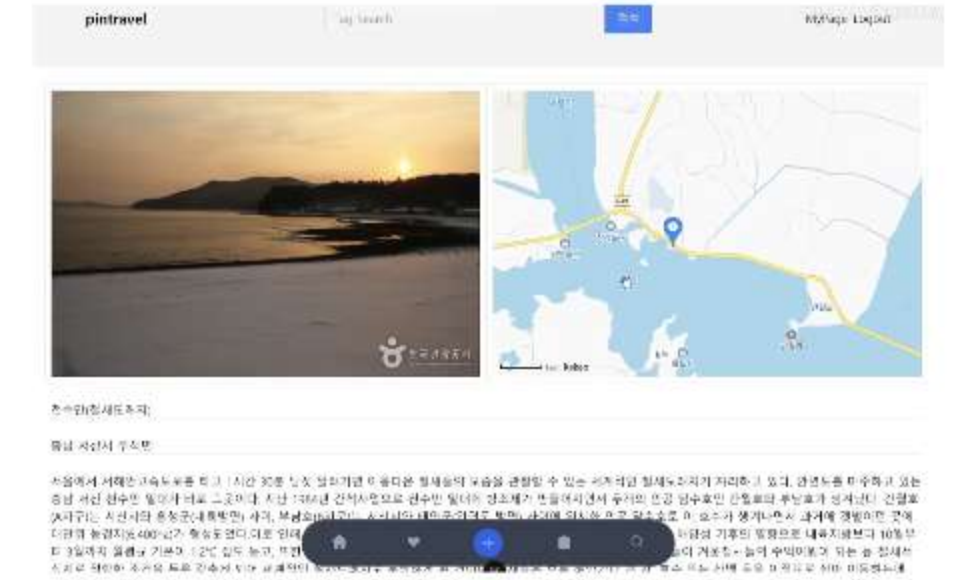
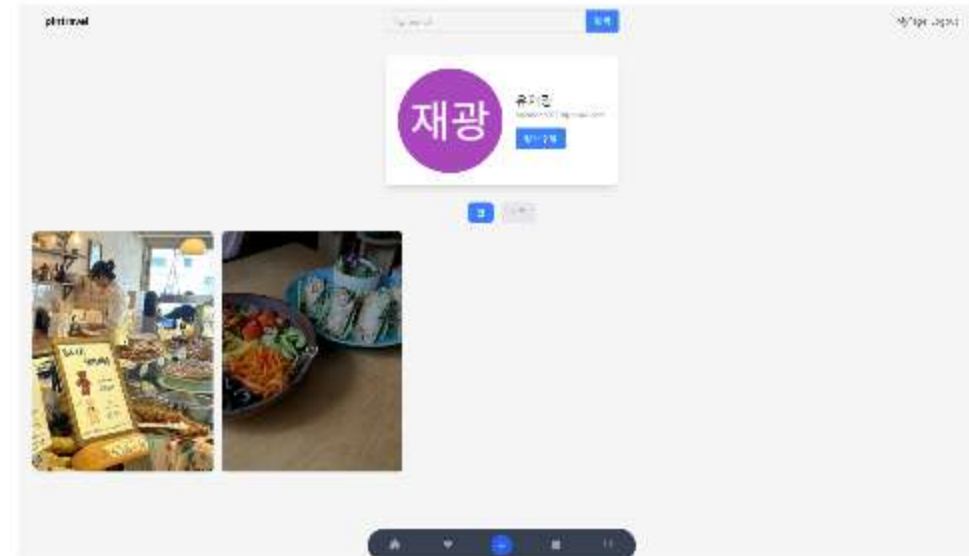
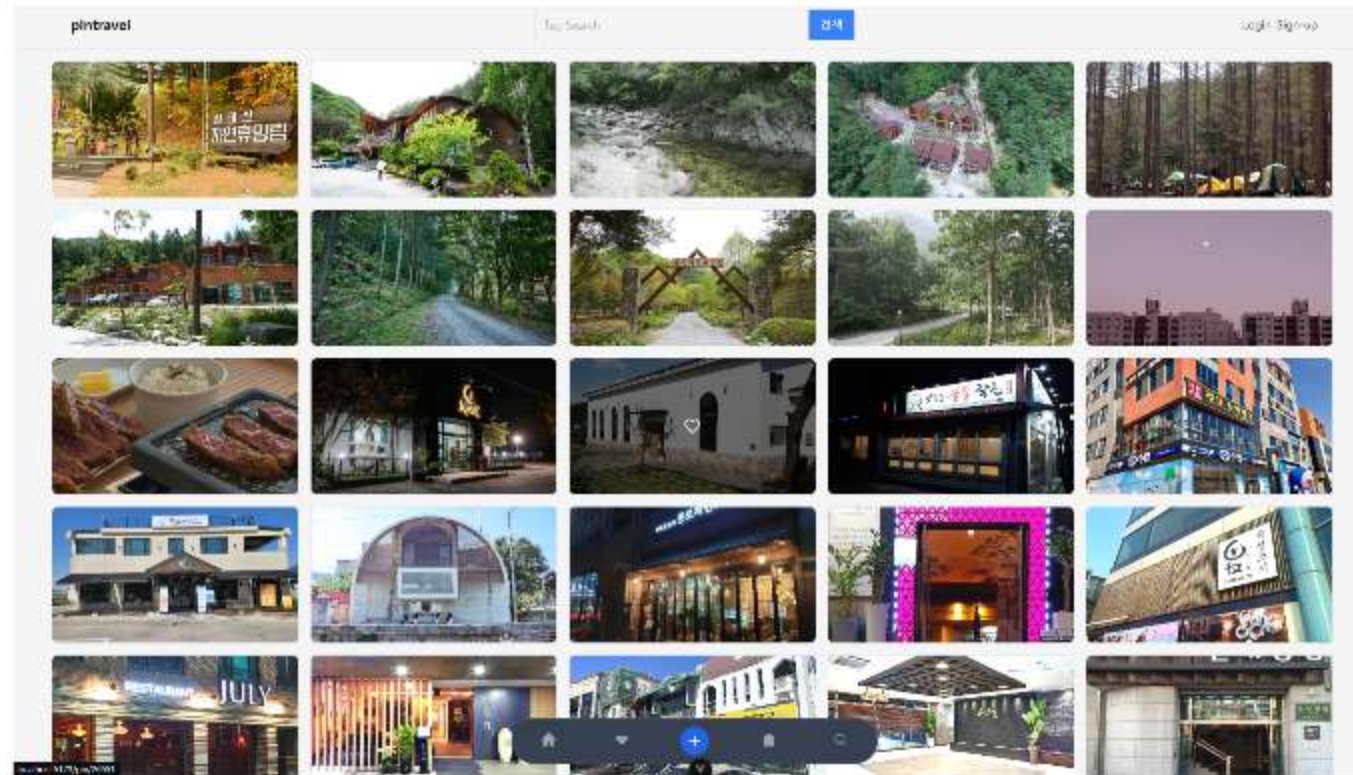
PinTravel은 여행자들에게 맞춤형 여행 정보를 제공하고,
효율적인 여행 계획 수립을 지원하는 웹 애플리케이션입니다.

이 프로젝트는 사용자가 여행지와 관련된
다양한 정보를 쉽게 찾기 위해 제작되었습니다..

PinTravel

여행을 '장바구니' 하다.

여행자들에게 유용한 정보와 맞춤형 여행 일정을 제공
사용자 친화적인 인터페이스를 통해 간편한 여행 계획 수립 지원



역할

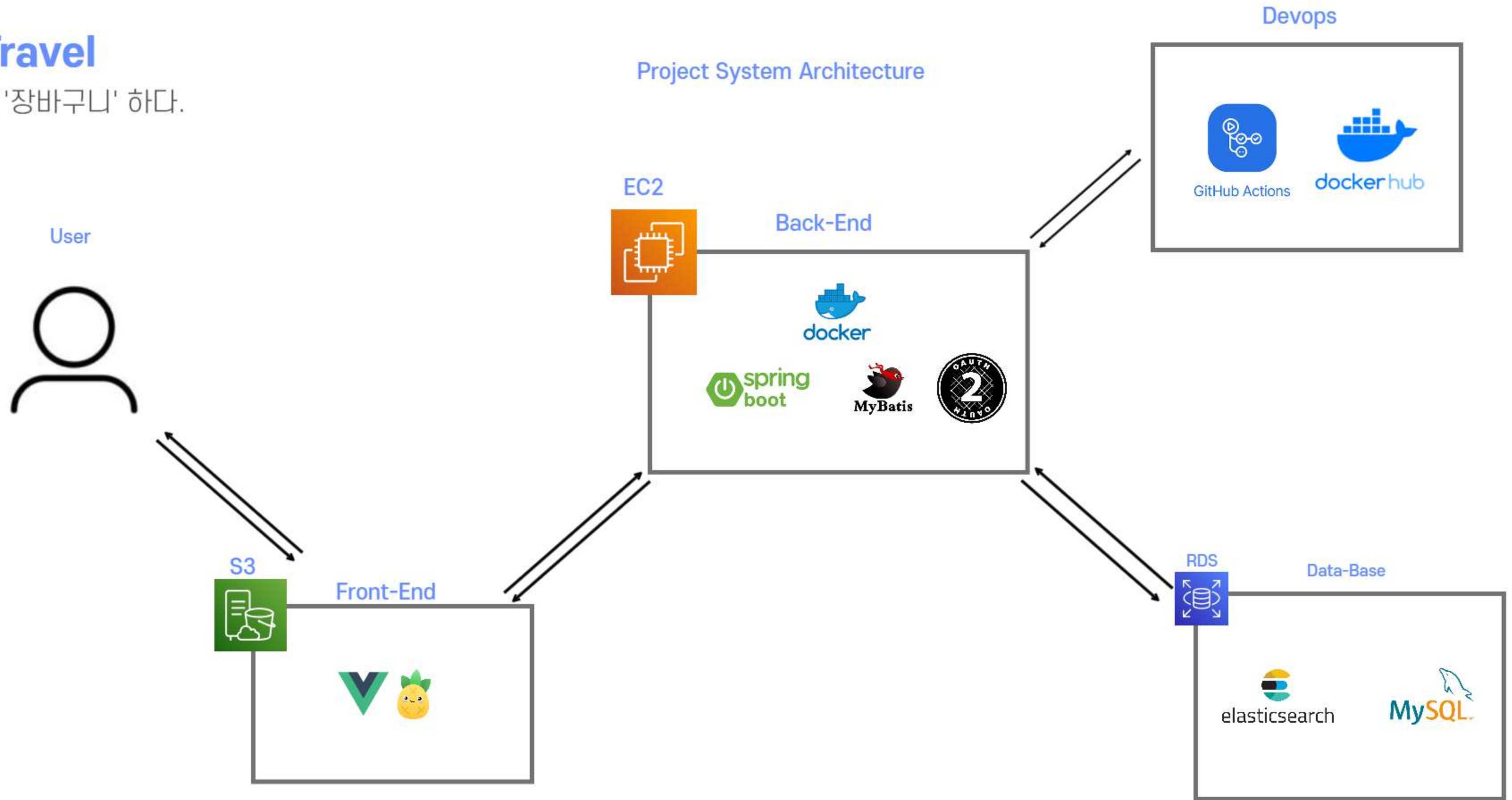
팀장 및 백엔드 개발자

기술 스택

Java, Spring Boot, MySQL, mybatis,
RESTful API, vue.js, aws, docker,
elasticSearch, python, sklearn

PinTravel

여행을 '장바구니' 하다.



소개

PinTravel은 Infinity Scroll 기반의 사진 및 태그를 활용하여 사용자의 여행 정보와 취향을 스크랩하고, 이를 통해 여행 계획 수립에 도움을 주는 웹 애플리케이션입니다.
이 프로젝트는 Pinterest를 벤치마킹하여 개발되었습니다.

기간 / 인원

24.05.11~24.05.26

프론트엔드 및 백엔드 2인 팀 프로젝트

핵심 기능

키워드 기반 검색

- 사용자가 관심 있는 키워드로 여행지를 검색할 수 있는 기능
- 태그 미리보기 제공: 사용자가 입력한 키워드와 관련된 태그를 미리 보여줌
- 검색 결과 제공: Elasticsearch를 활용하여 키워드 기반의 관련 여행지 및 명소를 빠르고 정확하게 검색하여 제공
- 기술 스택: Elasticsearch
- 위시리스트

사용자가 가고 싶은 여행지를 카테고리별로 저장해두는 기능

- 카테고리별 저장: 여행지를 사용자 정의 카테고리에 따라 분류하여 저장
- 군집화 및 경로 수립: 저장된 여행지들을 군집화(K-means 클러스터링)하여 최적의 여행 경로 제안
- 기술 스택: Vue.js, MySQL, AWS S3
- 구글 소셜 로그인 기능

구글 계정을 사용하여 간편하게 로그인할 수 있는 기능

- 구글 OAuth 2.0: 구글 소셜 로그인 API를 사용하여 사용자 인증
- 기술 스택: Spring Boot, OAuth 2.0
- 추천 게시물 기능

특정 게시물을 클릭했을 때, 그 게시물과 가까운 관광지 게시물을 추천하여 상세 뷰 하단에 노출하는 기능

- 관련 관광지 추천: 사용자가 선택한 게시물과 연관된 관광지를 추천
- 군집화된 카테고리 유도: 사용자가 특정한 게시물을 통해 다른 지역 및 카테고리로 자연스럽게 이동하도록 유도
- 의도: 사용자가 한 게시물에서 계속하여 다른 게시물로 이동하며 지역에 군집화된 정보를 탐색하도록 유도
- 기술 스택: Vue.js, MySQL

PinTravel

여행을 '장바구니' 하다.

지원자 개인 기여

역할 요약

- 스켈레톤 Front-End 구현
- 프로젝트 설계
- 쿼리 최적화
- 배포
- 리팩토링
- CI/CD 구축

멀티 태그 기반 검색 기능 구현:

- 사용자가 다중 태그를 선택하여 검색할 수 있는 기능 개발
- Elasticsearch를 활용하여 다양한 태그 조건을 만족하는 여행지 검색

인피니티 스크롤 구현:

- 사용자 경험 향상을 위해 페이지네이션 대신 인피니티 스크롤 기능 개발
- Vue.js와 API 연동을 통해 스크롤 시 자동으로 추가 데이터 로딩

거리 기반 추천 게시물 관련 쿼리 최적화:

- MySQL과 Elasticsearch를 활용하여 사용자 위치 기반으로 추천 게시물을 효율적으로 조회할 수 있도록 쿼리 최적화

Docker Hub를 통해 EC2로 백엔드 배포:

- CI/CD 파이프라인 설정 및 GitHub Actions를 사용해 Docker 이미지 빌드 및 배포 자동화
- Docker Hub에 이미지 푸시 후 EC2 인스턴스에서 이미지 가져와 컨테이너 실행

S3를 통해 프론트 엔드 배포:

- Vue.js로 개발한 프론트 엔드를 빌드하여 S3 버킷에 업로드

관광지 태깅을 위해 공공데이터 전처리:

- 공공데이터 API를 통해 수집한 데이터를 분석 및 전처리
- Hannanum 형태소 분석기를 사용하여 명사 추출 및 키워드 태깅

JWT 인증 인가 방식 AOP로 리팩터링:

- 기존의 인증 및 인가 로직을 AOP(Aspect-Oriented Programming)로 리팩터링
- 코드의 중복 제거 및 유지보수성 향상
- Spring Security와 JWT를 사용하여 안전하고 효율적인 인증 인가 시스템 구축

OAuth 로그인 구현:

- Google OAuth 2.0을 사용하여 소셜 로그인 기능 구현
- 사용자 인증 후 JWT 토큰 발급 및 사용자 세션 관리
- 프론트 엔드와 백엔드의 통합을 통해 원활한 로그인 경험 제공

PinTravel

여행을 '장바구니' 하다.

개발하면서 어려웠던 점 및 해결 방법

팀원과 notion을 활용하여 문제가 있을 때 마다 각자의 트러블 슈팅을 기록하였습니다.

index로 연립미상 초기화할 완료 후	해결	2024년 4월 25일 오후 3:43	저 제공 양	저 제공 양	2024년 5월 16일 오후 4:30
사유리터 의존성 추가시 로그인 해야함	해결	2024년 4월 25일 오후 5:54	저 제공 양	저 제공 양	2024년 5월 16일 오후 4:30
mapper가 안 잡히는 상황이었음	해결	2024년 4월 26일 오후 2:19	저 제공 양	저 제공 양	2024년 4월 26일 오후 5:30
Swagger에서 delete 가 안되는 점	해결	2024년 4월 26일 오후 3:16	저 제공 양	저 제공 양	2024년 4월 26일 오후 5:30
Controller 에서 리퀘스트바디가 null인 점	해결	2024년 4월 26일 오후 5:03	저 제공 양	저 제공 양	2024년 4월 26일 오후 5:30
Insert Query문 403 Error	해결	2024년 4월 26일 오후 5:05	서현빈	저 제공 양	2024년 5월 10일 오후 8:16
MapScanner가 Mapper를 못 찾는 상황	해결	2024년 5월 10일 오후 8:15	저 제공 양	저 제공 양	2024년 5월 11일 오전 11:16
flip-book 라이브러리의 오류	해결	2024년 5월 10일 오후 8:15	저 제공 양	저 제공 양	2024년 5월 11일 오전 11:15
프론트에서 받아오는 데이터가 undefined인 문제	해결	2024년 5월 10일 오후 8:16	저 제공 양	저 제공 양	2024년 5월 11일 오전 11:23
CORS 문제로 프론트 백엔드 연동 어려움	해결	2024년 5월 11일 오전 11:06	저 제공 양	저 제공 양	2024년 5월 11일 오전 11:26
페이징 처리 쿼리가 성능이 떨어짐	해결	2024년 5월 12일 오후 11:34	저 제공 양	저 제공 양	2024년 5월 22일 오전 9:57
게시글 하나 삭제 시 관련 전부 삭제하는가?	해결	2024년 5월 13일 오후 2:48	서현빈	서현빈	2024년 5월 14일 오후 1:46
무한스크롤 거리가준으로 무한으로 불러내는 방법?	해결	2024년 5월 13일 오후 5:27	서현빈	저 제공 양	2024년 5월 22일 오전 9:57
swagger를 통해 pin이 저장되지 않는 문제	해결	2024년 5월 13일 오후 5:54	서현빈	서현빈	2024년 5월 14일 오후 1:24
버밀번호 해시	해결	2024년 5월 14일 오후 1:24	서현빈	서현빈	2024년 5월 14일 오후 1:27
Maroon 디자인 적용 후 논리적 순서가 헷갈림에서 아래로 이동하는 점	미해결	2024년 5월 14일 오후 1:31	저 제공 양	저 제공 양	2024년 5월 16일 오후 5:18
이미지 모양이 느린 점	해결	2024년 5월 14일 오후 4:59	저 제공 양	저 제공 양	2024년 5월 15일 오후 12:13
뒤로가기가 안 만들어져서 사용자 경험이 좋지 않은 점	해결	2024년 5월 15일 오후 11:06	저 제공 양	저 제공 양	2024년 5월 16일 오후 1:22
검색 시 유사도를 어떻게 해야할지에 대한 고민	해결	2024년 5월 16일 오후 1:22	저 제공 양	저 제공 양	2024년 5월 17일 오후 5:35
JWT 로그인 구현	해결	2024년 5월 16일 오후 2:02	서현빈	서현빈	2024년 5월 16일 오후 3:23
검색어 요청시 한글 연고당 문제	해결	2024년 5월 16일 오후 3:55	저 제공 양	저 제공 양	2024년 5월 16일 오후 4:29
project 검색시 불필요한 요청 줄이기 debounce	해결	2024년 5월 16일 오후 4:27	저 제공 양	저 제공 양	2024년 6월 6일 오후 9:48
검색 상 구현시 해시태그 추가하기	해결	2024년 5월 16일 오후 4:46	저 제공 양	저 제공 양	2024년 5월 17일 오전 10:34
로그인 상태가 유지가 되지 않는 문제	해결	2024년 5월 16일 오후 5:13	서현빈	서현빈	2024년 5월 16일 오후 5:17
세고코임하얀 스토어에 저장된 userInfo가 중복되는 문제	해결	2024년 5월 17일 오전 9:11	서현빈	저 제공 양	2024년 5월 19일 오후 1:32
!!! User의 Board 보내일 !!!	해결	2024년 5월 17일 오전 10:10	서현빈	저 제공 양	2024년 6월 6일 오후 9:48
배열 정보를 서버에 전송시 get 대신 post를 해야함	해결	2024년 5월 17일 오전 10:33	저 제공 양	저 제공 양	2024년 5월 17일 오후 5:43
관련된 이미지를 계속 보여주는 법	해결	2024년 5월 17일 오후 5:34	저 제공 양	저 제공 양	2024년 5월 17일 오후 5:35
문제점 : DetailView를 로드할 때 3개의 쿼리가 나간다.		2024년 5월 17일 오후 5:35	저 제공 양	저 제공 양	2024년 5월 17일 오후 10:15
사진 업로드할 수 있는 점 개발중인데, 사용자가 주소 입력해야할까 아니면, kakao API를 활용해서 사용자가 입력할 수 있게 해야할까 아니면 사진의 메타데이터를 가져와서 있어야할까	해결	2024년 5월 19일 오후 3:37	저 제공 양	저 제공 양	2024년 6월 6일 오후 10:05
검색 시 쿼리문 작성에 대한 고민	해결	2024년 5월 20일 오전 11:00	저 제공 양	저 제공 양	2024년 6월 6일 오후 10:06
중서널 체이닝(Optional Chaining) 문법	해결	2024년 5월 20일 오후 12:04	서현빈	서현빈	2024년 5월 20일 오후 12:09
ElasticSearch & Spring data Elastic & Spring boot	해결	2024년 5월 20일 오후 2:28	저 제공 양	저 제공 양	2024년 6월 6일 오후 10:06
Pin 불러올 때 종여요여부를 어떻게 가져올???	해결	2024년 5월 20일 오후 5:30	서현빈	저 제공 양	2024년 6월 6일 오후 10:06
How to Add Board		2024년 5월 20일 오후 5:46	서현빈	서현빈	2024년 5월 20일 오후 5:49
소셜 로그인어 한번 등록한 이후로 다시 되는 여음	해결	2024년 5월 21일 오후 9:42	서현빈	저 제공 양	2024년 6월 6일 오후 10:06
v-Hot v-show	해결	2024년 5월 22일 오후 3:16	서현빈	저 제공 양	2024년 6월 6일 오후 10:06

PinTravel

여행을 '장바구니' 하다.

그 중 기억에 남는 이슈

문제 1: 데이터 전처리

어려움: 공공데이터 API를 통해 수집한 데이터가 비정형적이고 불완전하여 정형화하고 필요한 정보를 추출하는 데 어려움이 있었습니다.

해결 방법: Python과 Pandas를 활용하여 데이터를 정제하고 필요한 컬럼만 추출하여 새로운 데이터셋을 생성. Hannanum 형태소 분석기를 사용하여 명사를 추출하고 이를 키워드로 태깅하여 검색에 활용하였습니다.

문제 2: 인피니티 스크롤 구현

어려웠던 점:

사용자 경험을 향상시키기 위해 인피니티 스크롤을 구현하는 과정에서 스크롤 이벤트를 효율적으로 처리하는 것이 어려웠습니다.

해결 방법:

Vue.js와 Intersection Observer API를 활용하여 스크롤이 하단에 도달할 때 자동으로 추가 데이터를 로드하는 방식을 구현했습니다.

백엔드에서 페이지네이션을 지원하도록 설계하여, 필요한 데이터만 클라이언트로 전달하도록 최적화했습니다.

문제 3: 거리 기반 추천 시스템의 쿼리 최적화

어려웠던 점:

사용자 위치를 기반으로 가까운 여행지를 추천하는 기능을 구현하는 과정에서, 거리 계산 쿼리가 비효율적이어서 성능 문제가 발생했습니다.

해결 방법:

MySQL과 Elasticsearch를 함께 사용하여 거리 계산을 처리했습니다.

지오스페이셜 인덱스를 활용하여 효율적으로 위치 기반 데이터를 처리하고, 필요한 경우 Elasticsearch를 통해 검색 결과를 보완했습니다.

문제 4: CI/CD 파이프라인 설정

어려웠던 점:

GitHub Actions를 활용하여 CI/CD 파이프라인을 설정하는 과정에서, Docker 이미지를 빌드하고 EC2에 배포하는 자동화 과정이 복잡했습니다.

해결 방법:

GitHub Actions 워크플로우 파일을 작성하여 코드 푸시 시 자동으로 Docker 이미지를 빌드하고 Docker Hub에 푸시하도록 설정했습니다.

EC2 인스턴스에서 새로운 이미지를 가져와 컨테이너를 실행하는 스크립트를 작성하여, 자동 배포가 가능하도록 했습니다.

문제 5: OAuth 소셜 로그인 구현

어려웠던 점:

Google OAuth 2.0을 사용하여 소셜 로그인 기능을 구현하는 과정에서, JWT 토큰 관리와 사용자 세션 관리가 복잡했습니다.

해결 방법:

Spring Security와 JWT를 사용하여 안전하고 효율적인 인증 인가 시스템을 구축했습니다.

프론트엔드와 백엔드 간의 통합 테스트를 통해 원활한 로그인 경험을 제공했습니다.

PROJECT.3

테이콘 데이터 분석 경진대회

추석 맞이 추석 선물 수요량 예측 AI 해커톤

02

추석 맞이 추석 선물 수요량 예측 AI 해커톤

프로젝트 개요

목적: 추석 선물 세트의 수요량을 예측하여 최적의 재고 관리 및 마케팅 전략 수립

기간: 2023년 9월 27일 - 2023년 10월 3일

성과: 대회 5등 달성

사용한 기술 및 도구

프로그래밍 언어: Python

라이브러리: AutoGluon, H2O AutoML, LightGBM, XGBoost, CatBoost, RandomForest, pandas, scikit-learn

데이터 전처리: 범주형 변수 원-핫 인코딩, 데이터 스케일링

모델 평가 지표: RMSE (Root Mean Squared Error)

프로젝트 설명

이 프로젝트는 추석 선물 세트의 수요량을 예측하기 위해 다양한 머신러닝 모델을 실험하고 최적의 모델을 선정하는 것을 목표로 하였습니다. 데이터 전처리와 다양한 모델 실험을 통해 최종적으로 AutoGluon을 사용하여 대회에서 높은 성과를 거두었습니다.

DAICON커뮤니티대회학습랭킹더보기

구독 안내

RI


추석 맞이 추석 선물 수요량 예측 AI 해커톤

알고리즘 | 정형 | 회귀 | 추석 | RMSE

🏆 상금 : 인증서

📅 2023.09.27 ~ 2023.10.04 09:59 [+ Google Calendar](#)

👤 420명 📅 마감



🔗

참여중

대회안내

데이터

코드 공유

토크

리더보드

제출

☰ 개요

📋 규칙

🕒 일정

🏆 상금

📄 동의사항

[배경]

안녕하세요 데이터 여러분!
추석 맞이 추석 선물 수요량 예측 AI 해커톤에 오신 것을 환영합니다.
데이콘에서 추석을 맞아 추석 선물 수요량 예측 AI 해커톤을 개최했습니다.
다른 사람들과 인사이트를 겨루며 알고리즘 대회의 즐거움을 느껴 보세요.

[주제]

추석 선물 수요량 예측

[설명]

추석 맞이 기념 대회에 자유롭게 참여해보세요!

[주최 / 주관]

데이콘

추석 맞이 추석 선물 수요량 예측 AI 해커톤

모델 평가 및 선택

여러 모델을 실험한 후, AutoGluon의 성능이 가장 우수하여 최종 모델로 선택하였으며, 이를 통해 대회 5등을 달성하였습니다.

트러블 슈팅

문제 1: 데이터 전처리

문제: 원-핫 인코딩 후 데이터 스케일링에서 변환 오류 발생.

해결 방법: 원-핫 인코딩 후 스케일링 범위를 재설정하고, 수요 예측에서 음수가 나오는 문제를 해결하기 위해 음수 값을 0으로 초기화.

문제 2: 모델 선택

문제: 여러 모델 중 최적의 모델을 선택하는 과정에서 성능 변동이 심하여 안정적인 모델 선택이 어려웠음.

해결 방법: AutoML 도구를 활용하여 다양한 모델을 자동으로 실험하고 최적의 모델을 선택. 특히 AutoGluon을 사용하여 모델 성능을 비교하고, 가장 낮은 RMSE를 기록한 모델을 최종 선택.

추석 맞이 추석 선물 수요량 예측 AI 해커톤 성과 및 과정

추석 맞이 추석 선물 수요량 예측 AI 해커톤

알고리즘 | 정형 | 회귀 | 추석 | RMSE

④ 상금 : 인증서

인원 420명 마감










참여중

순위기준

● WINNER ● 1% ● 4% ● 10%

전체 랭킹 >

#	팀	팀 멤버	최종점수	재출수	등록일
5	RidFlow		120.40349	22	8달 전
1	O_O		116.39308	78	8달 전
2	basslibrary		119.57596	51	8달 전
3	Kororu		119.7716	25	8달 전
4	이세익인공지능		120.34849	27	8달 전
5	RidFlow		120.40349	22	8달 전
6	주머니뒤		120.59206	45	8달 전

923732	<div>submit.csv</div> <div>autogluon을 사용하고 데이터 전처리를 임의로 함</div> <div># 범주형 변수 원-핫 인코딩 train_x = pd.get_dummies(train_x, columns=['쇼핑몰 구분', '도시 유형', '지역 유형', '쇼핑몰 유형', '선물 유형']) test_x = pd.get_dummies(test_x, columns=['쇼핑몰 구분', '도시 유형', '지역 유형', '쇼핑몰 유형', '선물 유형']) # 데이터 스케일링 scaler = StandardScaler() train_x[['추석까지 남은 기간(주)', '가격(원)']] = scaler.fit_transform(train_x[['추석까지 남은 기간(주)', '가격(원)']]) test_x[['추석까지 남은 기간(주)', '가격(원)']] = scaler.fit_transform(test_x[['추석까지 남은 기간(주)', '가격(원)']]) edit</div> <div>2023-10-02 15:28:48</div> <div>108.0990854067</div> <div></div>
923014	<div>submit.csv</div> <div>이전과 동일, ID 컬럼 삭제 edit</div> <div>2023-10-01 16:17:46</div> <div>107.4046230384</div> <div></div>
923008	<div>submit.csv</div> <div>autogluon인데 데이터 전처리를 안함. 전적으로 모델에게 맡김. edit</div> <div>2023-10-01 16:13:18</div> <div>107.4046230384</div> <div></div>
921961	<div>submit.csv</div> <div>autogluon edit</div> <div>2023-09-29 19:32:25</div> <div>107.070641377</div> <div></div>
924281	<div>submit.csv</div> <div>edit</div> <div>2023-10-03 16:08:45</div> <div>106.8002641499</div> <div></div>
923054	<div>submit.csv</div> <div>time_limit = 1800 label = '수요량' metric = 'root_mean_squared_error' predictor = TabularPredictor(label, eval_metric=metric).fit(train_data, time_limit=time_limit, presets='best_quality') edit</div> <div>2023-10-01 17:28:04</div> <div>105.7104492421</div> <div></div>
923341	<div>submit.csv</div> <div>time_limit = 1800 label = '수요량' metric = 'root_mean_squared_error' predictor = TabularPredictor(label, eval_metric=metric).fit(train_data, time_limit=time_limit, presets='best_quality') submission['수요량'] = predictor.predictions.round() edit</div> <div>2023-10-02 00:01:06</div> <div>105.695550539</div> <div></div>

PROJECT 03

송파구의 민방위 대피소 대피경로 최적화

송파구의 민방위대피소의 현황 및 분석

목차

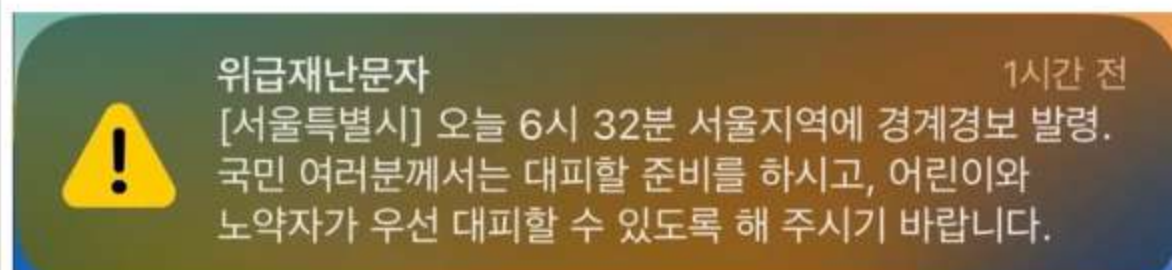
- 1 데이터 분석 주제 및 선정 이유
- 2 데이터 수집 방법
- 3 데이터 형태 및 전처리 방법
- 4 데이터 분석에 사용된 모형 및 방법론

- 5 분석 결과물
- 6 분석 결과물의 선정한 주제의
의사 결정에 적용한 결론

데이터 분석 주제 및 선정 이유

주제 선정 이유

서울특별시 경계경보 오발령 사건



2023년 5월 31일 오전 06시 32분 서울특별시가
2023년 북한 천리마-1 발사 사건에 대한 경계경보를
발령하고 위급재난문자를 발송하였으나, 이후 오발령
으로 밝혀진 사건



어디로 대피해야하는지
적혀져 있지
않은 재난 문자



무작정 대피해야 할 때
어디로 대피해야할까



수용인원을 넘기지는
않을까?

데이터 수집 방법

중요 데이터

서울특별시 송파구_
민방위대피시설

+

송파구 각 동별 인구

공공 데이터 포털

서울특별시 송파구_민방위대피시설

<https://www.data.go.kr/data/15062878/fileData.do>

전체 행 : 158

등록일2020-08-07

수정일2021-10-06

송파구청청

2023년 5월 송파구 인구 및 세대현황

<https://www.songpa.go.kr/www/index.do>

- 총 인구: 657,260명(남:316,082명,
여:341,178명)

- 총 세대:285,615세대

데이터 형태 및 전처리 방법

서울특별시 송파구_민방위대피시설 설 전처리 전

```
import pandas as pd
facility_df = pd.read_csv('서울특별시 송파구_민방위대피시설_20210914.csv', encoding='UTF-8')
print(facility_df.head())
```

	민방위대피시설명	민방위대피시설구분	소재지도로명주소	소재지지번주소	#
0	2호선 종합운동장역	2호선 종합운동장역	서울특별시 송파구 올림픽로 지하 23 (잠실동, 종합운동장역)		NaN
1	2호선 잠실새내역	2호선 잠실새내역	서울특별시 송파구 올림픽로 지하 140 (잠실동, 신천역)		NaN
2	2호선 잠실역	2호선 잠실역	서울특별시 송파구 올림픽로 지하 265 (잠실동, 잠실역2호선)		NaN
3	잠실성당 지하강의실	잠실성당 지하강의실	서울특별시 송파구 올림픽로12길 9 (잠실동, 잠실천주교회)		NaN
4	잠실종합상가	잠실종합상가	서울특별시 송파구 백제고분로 157 (잠실동, 반도가구전시장)		NaN

	위도	경도	민방위대피시설면적	대피가능인원수	개방여부	관리기관명	데이터기준일자
0	37.510968	127.073356	18388	22288	Y	서울특별시 송파구청	2021-09-14
1	37.511569	127.086373	8227	9972	Y	서울특별시 송파구청	2021-09-14
2	37.513320	127.100315	4752	5760	Y	서울특별시 송파구청	2021-09-14
3	37.510382	127.083241	874	1059	Y	서울특별시 송파구청	2021-09-14
4	37.505782	127.084317	1518	1840	Y	서울특별시 송파구청	2021-09-14

데이터 형태 및 전처리 방법

서울특별시 송파구_민방위대피시설 전처리 후

```
dtype: float64
민방위대피시설명      위도      경도  민방위대피시설면적  대피가능인원수
0      2호선 종합운동장역  37.510968  127.073356      18388      22288
1      2호선 잠실새내역  37.511569  127.086373       8227       9972
2      2호선 잠실역  37.513320  127.100315       4752       5760
3      잠실성당 지하강의실  37.510382  127.083241        874       1059
4      잠실종합상가  37.505782  127.084317       1518       1840
...
...
153     9호선 삼전역  37.504323  127.088084       8644      10477
154     석촌고분역  37.502464  127.096885       6841       8292
155     한성백제역  37.516340  127.116114       8969      10871
156     송파나루역  37.510350  127.112232       7833       9494
157  헬리오시티 지하주차장  37.497612  127.102533      507732     615432

[158 rows x 5 columns]
```


데이터 형태 및 전처리 방법

행정구역_읍면동(법정동)<국가공간
정보포털>

```
# GeoJSON 파일 읽기
gdf = gpd.read_file('HangJeongDong_ver20230401.geojson')

# 송파구에 해당하는 데이터만 필터링
gdf_songpa = gdf[gdf['sggnm'] == '송파구']

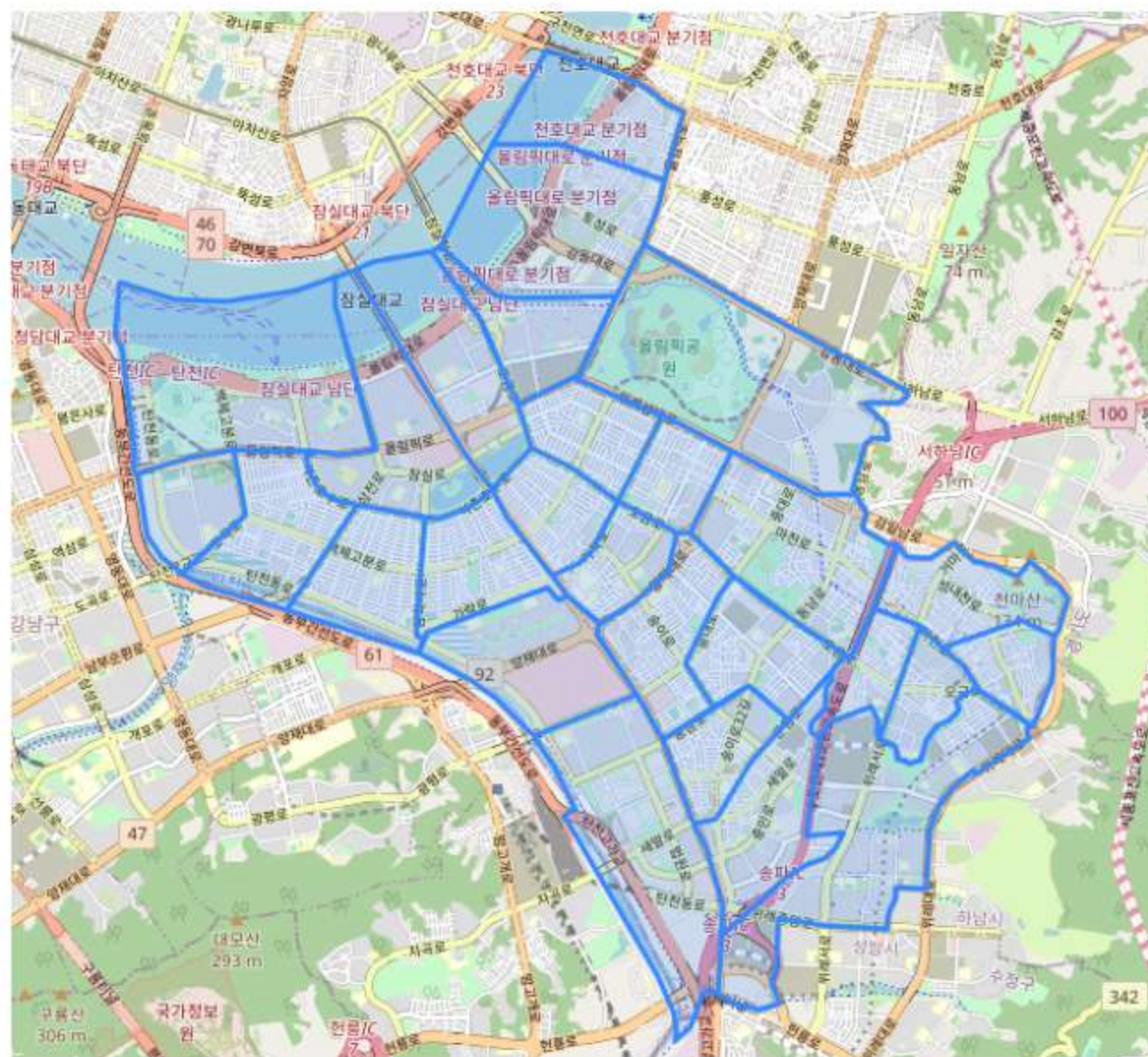
# 송파구 데이터를 지도에 추가
songpa_map = folium.Map(location=[37.5145, 127.105], zoom_start=12) # 송파구 지도 생성

folium.GeoJson(gdf_songpa).add_to(songpa_map) # 송파구 데이터를 지도에 추가

# 지도 출력
songpa_map.save('songpa_map.html') # HTML 파일로 저장
```

데이터 형태 및 전처리 방법

행정구역_읍면동(법정동)<국가공간
정보포털>



데이터 형태 및 전처리 방법

민방위대피시설 위치 추가

```
# folium 지도 생성

songpa_map = folium.Map(location=[37.514322572335935, 127.05748125608533], zoom_start=12)
# Iterate over the facility dataframe and add markers to the map
for index, row in facility_df.iterrows():
    facility_name = row['민방위대피시설명']
    latitude = row['위도']
    longitude = row['경도']

    # Add marker to the map
    folium.Marker([latitude, longitude], popup=facility_name).add_to(songpa_map)
```


데이터 형태 및 전처리 방법

민방위대피시설 위치 추가



데이터 형태 및 전처리 방법

송파구 각 지역 인구 수 만큼 좌표 생성

```
# 좌표 생성 후 csv 파일로 저장함
# 인구수만큼 포인트 생성
for row in gdf_songpa.iterrows():
    poly = row[1]['geometry']
    population = row[1]['인구수']
    for _ in range(int(population)): # 인구 수 만큼 포인트를 생성 하는데, 기호에 따라 100으로 나누거나 해도 괜찮을듯
        while True:
            point = Point(random.uniform(poly.bounds[0], poly.bounds[2]), random.uniform(poly.bounds[1], poly.bounds[3]))
            if poly.contains(point):
                points.append(point)
                break

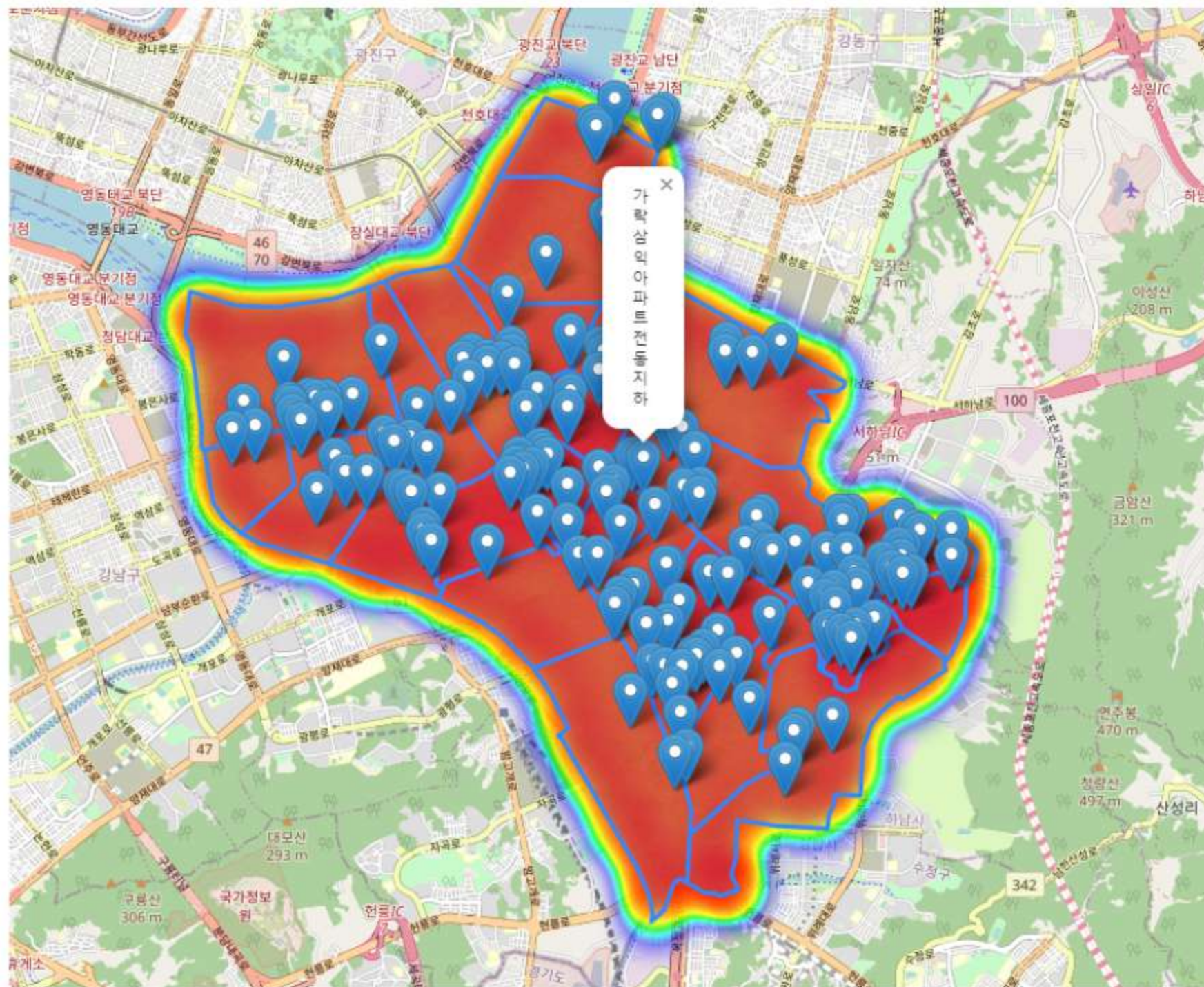
# 좌표들을 데이터프레임으로 저장
points_df = pd.DataFrame({'Latitude': [point.y for point in points], 'Longitude': [point.x for point in points]})

# csv파일로 저장
points_df.to_csv('points.csv', index=False)
```

송파구의 총 인구수인 657,260의 좌표가 생성

데이터 형태 및 전처리 방법

생성된 좌표를 통해 히트맵 추가



데이터 분석에 사용된 모형 및 방법론

분석에 사용 된 모형 및 방 법론

1. 데이터 준비

2. 클러스터링을 위한 데이터 준비

- K-means 클러스터링: 인원 데이터를 대피소 위치와의 거리에 따라 클러스터로 그룹화합니다.
- 거리 계산: 대피소와 인원 사이의 거리를 계산하여 가장 가까운 대피소를 할당하고, 가장 멀리 떨어진 인원을 재할당합니다.
- 전처리 된 대피소 데이터(updated_facility.csv)와 인원 데이터(points.csv)를 불러옵니다.
- 인원 데이터에서 'Latitude'와 'Longitude' 열을 추출하여 X 배열로 준비합니다.
- 대피소의 '위도'와 '경도'를 추출하여 init 배열로 준비합니다.

데이터 분석에 사용된 모형 및 방법론

3. 클러스터링 수행

- KMeans 클러스터링 알고리즘을 사용하여 X 데이터를 init으로 초기화하여 클러스터링을 수행합니다.
- 인원 데이터에 'cluster' 열을 추가하여 클러스터 할당 결과를 저장합니다.

4. 인원 할당 및 재할당

- 각 대피소에 할당된 인원 수를 계산합니다.
- 대피소의 수용인원을 초과하는 경우, 가장 멀리 있는 인원을 가장 가까운 대피소로 재할당합니다.

5. 시각화

- Matplotlib을 사용하여 클러스터링 결과를 산점도로 플롯합니다. 각 클러스터는 다른 색으로 표시됩니다.
- Folium 라이브러리를 사용하여 지도 위에 대피소와 인원 위치를 마커로 표시합니다. 대피소는 검은색으로 표시되고, 인원은 클러스터에 따라 다른 색으로 표시됩니다.

데이터 분석 에 사용된 모 형 및 방법론

K-means clustering algorithm을 사용
한 이유

가장 가까운 대피소로 이동하게 된다면?
특정 대피소는 수용인원을 초과하게 될것

```
# 대피가능 인원수를 초과하는 시설을 찾아냅니다.  
exceed_capacity_df = updated_facility[updated_facility['카운트'] > updated_facility['대피가능인원수']]  
# 카운트와 대피 가능 인원수의 비율을 계산합니다.  
exceed_capacity_df['비율'] = exceed_capacity_df['카운트'] / exceed_capacity_df['대피가능인원수']  
  
print(exceed_capacity_df)
```

66개의 대피소가 수용인원을 초과하게 되
므로 적절히 분배하게 할 필요가 있음

	민방위대피시설명	위도	경도	대피가능인원수	카운트	비율
3	잠실성당 지하강의실	37.510382	127.083241	1059	2173.0	2.051936
4	잠실종합상가	37.505782	127.084317	1840	3927.0	2.134239
6	잠실우성4차아파트 전동 지하	37.502691	127.082183	9600	10142.0	1.056458
8	문정1동주민센터	37.490073	127.124183	200	2901.0	14.505000
9	서울시농수산물공사 지하주차장	37.496467	127.113455	3375	9391.0	2.782519
...
146	송파파크데일2단지	37.496339	127.158164	4266	7870.0	1.844820
148	위례송파푸르지오 지하주차장	37.480209	127.139760	2635	3823.0	1.450854
151	서울동부지방법원	37.483493	127.119754	2001	9954.0	4.974513
152	위례동주민센터 지하주차장	37.481167	127.143936	477	8066.0	16.909853
154	석촌고분역	37.502464	127.096885	8292	9085.0	1.095634

[66 rows x 6 columns]

데이터 분석에 사용된 모형 및 방법론

1. 데이터 준비

```
# 먼저, 대피소와 인원 데이터를 로드합니다.  
shelters = pd.read_csv('updated_facility.csv')  
shelters['카운트'] = 0  
people = pd.read_csv('points.csv').sample(10000)  
  
# 클러스터링을 위한 데이터 준비  
X = people[['Latitude', 'Longitude']].values  
init = shelters[['위도', '경도']].values
```

데이터 분석에 사용된 모형 및 방법론

2~3. 클러스터링을 위한 데이터 준비 및 실행

```
# 클러스터링을 위한 데이터 준비
X = people[['Latitude', 'Longitude']].values
init = shelters[['위도', '경도']].values
```

```
# 클러스터링 수행
kmeans = KMeans(n_clusters=init.shape[0], init=init, n_init=1)
people['cluster'] = kmeans.fit_predict(X)
```

데이터 분석에 사용된 모형 및 방법론

4. 인원 할당 및 재할당

```
# 대피소가 수용인원을 초과한 경우, 가장 멀리 있는 인원을 가장 가까운 대피소로 재할당
# 대피소의 수용인원을 고려하여 인원을 재할당
for index, row in shelters.iterrows():
    count = counts.get(index, 0)
    while count > row['대피가능인원수']:
        # 대피소로부터 가장 멀리 떨어진 인원 찾기
        people_in_cluster = people[people['cluster'] == index].copy()
        people_in_cluster['distance'] = np.sqrt((people_in_cluster['Latitude'] - row['위도'])**2 + (people_in_cluster['Longitude'] - row['경도'])**2)
        farthest_person_index = people_in_cluster['distance'].idxmax()

        # 대피소 중에서 수용인원을 초과하지 않는 가장 가까운 대피소 찾기
        feasible_shelters = shelters[shelters['대피가능인원수'] >= count]
        feasible_shelters['distance'] = np.sqrt((feasible_shelters['위도'] - people_in_cluster.loc[farthest_person_index, 'Latitude'])**2 +
                                                (feasible_shelters['경도'] - people_in_cluster.loc[farthest_person_index, 'Longitude'])**2)
        nearest_shelter_index = feasible_shelters['distance'].idxmin()

        # 인원을 가장 가까운 대피소로 재할당
        people.loc[farthest_person_index, 'cluster'] = nearest_shelter_index

    # 인원 수 업데이트
    count -= 1
    counts[nearest_shelter_index] += 1
    counts[index] -= 1
```


데이터 분석에 사용된 모형 및 방법론

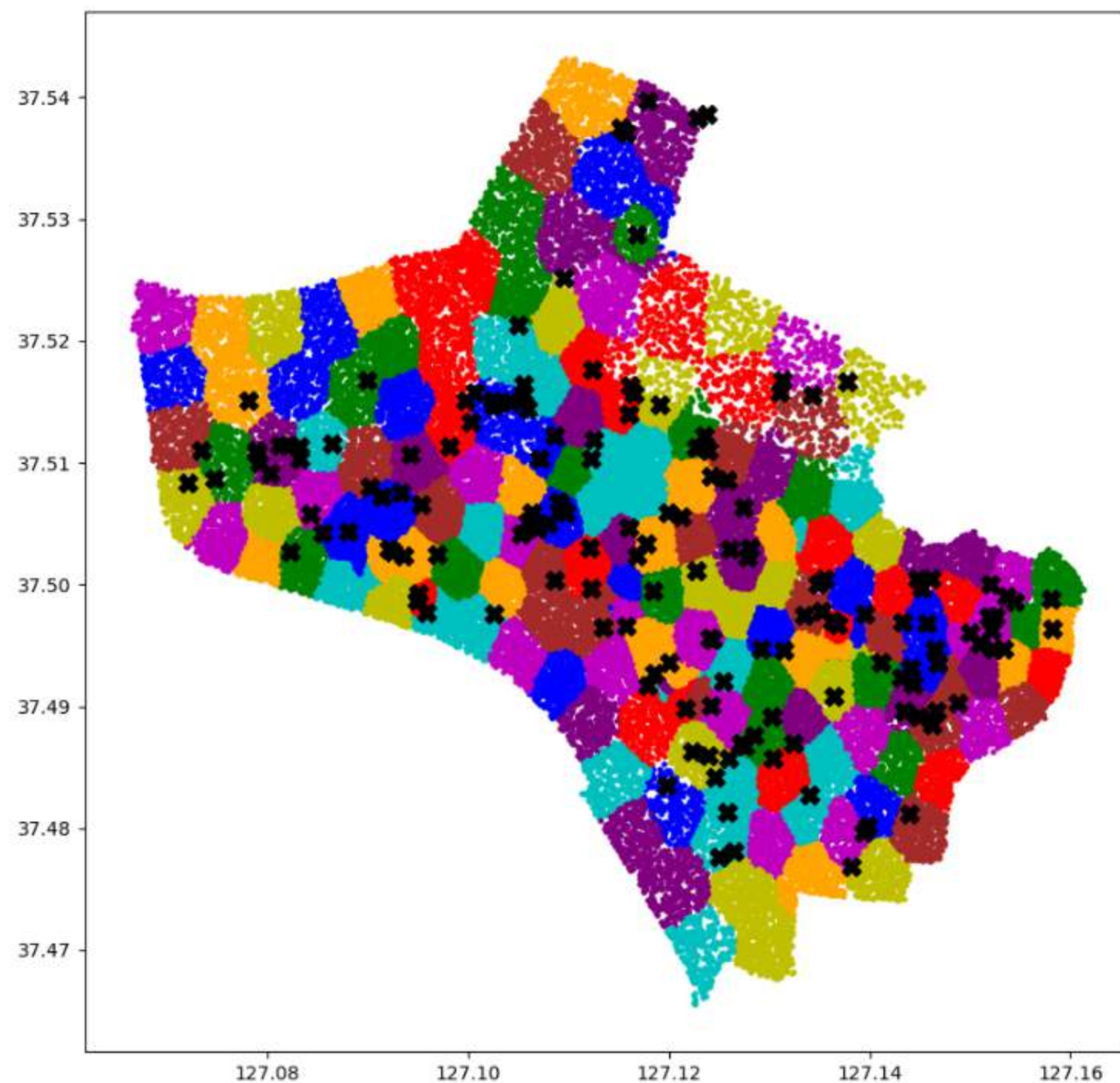
```
import matplotlib.pyplot as plt

# 각 클러스터에 대해 다른 색을 사용하여 사람들의 위치를 플롯합니다.
colors = ['b', 'g', 'r', 'c', 'm', 'y', 'orange', 'purple', 'brown']
fig, ax = plt.subplots(figsize=(10, 10))
for i in range(len(shelters)):
    cluster_points = people[people['cluster'] == i]
    ax.scatter(cluster_points['Longitude'], cluster_points['Latitude'], s=5, c=colors[i % len(colors)])

# 대피소의 위치를 플롯합니다.
ax.scatter(shelters['경도'], shelters['위도'], s=100, c='black', marker='x')

plt.show()
```

5. 시각화



분석 결과물

좌표

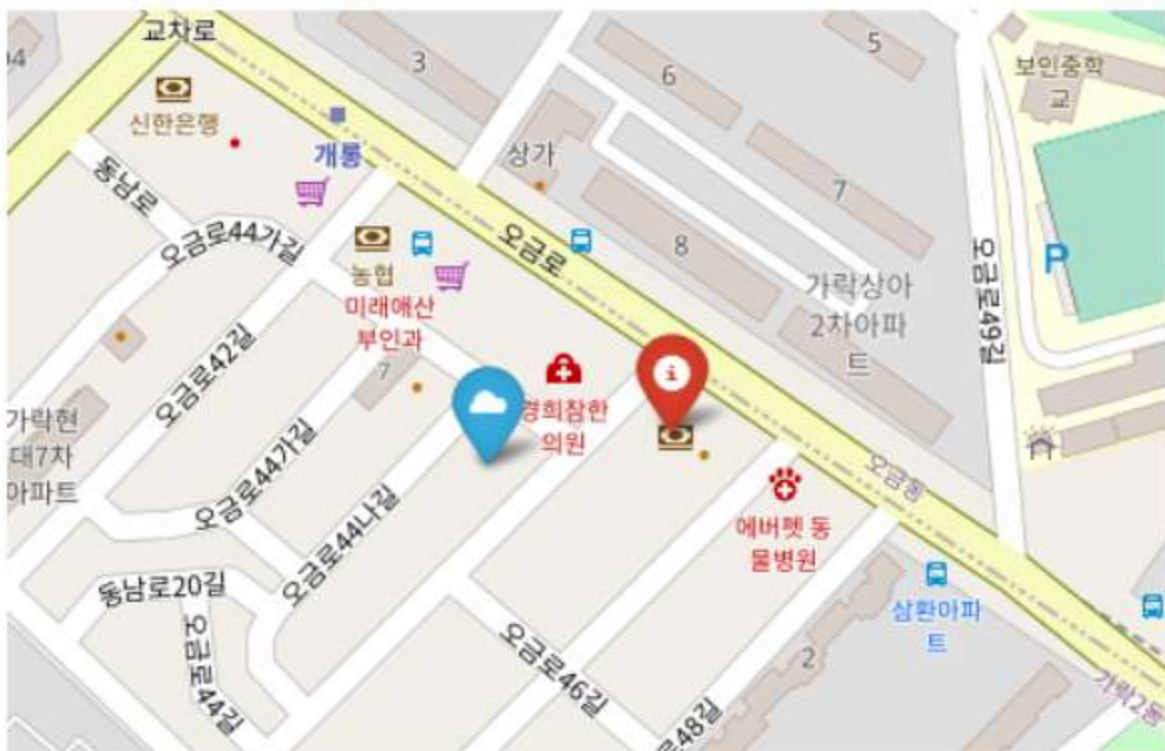
위도 : 37.49687939935719,
경도 : 127.13548474247413

대피 상황 때 우리집에서 가야할 대피소는?

```
my_location = pd.DataFrame({'Latitude': [37.49687939935719], 'Longitude': [127.13548474247413]})  
people = pd.concat([people, my_location], ignore_index=True)
```

```
my_cluster = people.loc[people['Latitude'] == 37.49687939935719]['cluster'].values[0]  
my_shelter = shelters.loc[my_cluster]  
print(my_shelter)  
print(f"가장 가까운 대피소는 위도 {my_shelter['위도']}, 경도 {my_shelter['경도']}에 위치하고 있습니다.")
```

```
[158 rows x 5 columns]  
민방위대피시설명      유니팜코리아 지하주차장  
위도                  37.49699  
경도                  127.136276  
민방위대피시설면적      862  
대피가능인원수          1044  
Name: 33, dtype: object  
가장 가까운 대피소는 위도 37.49699021, 경도 127.1362765에 위치하고 있습니다.
```



분석 결과물의 선정한 주제의 의사 결정에 적용한 결론

데이터 분석을 통해 재난 상황시 효율적인 대피소 위치 제공



이 분석 결과를 통해, 비상 상황 시 사람들이 가장 효율적으로 대피할 수 있는 경로를 분석했음. 이는 실제 비상 상황 발생 시, 즉각적인 대피 경로를 찾아 생명을 구하는 데 중요한 역할을 할 것으로 보임. 또한, 이 알고리즘이 실시간으로 적용되면, 대피소의 혼잡도를 고려하여 사람들을 동적으로 재분배하는 것이 가능해질 것임. 이는 비상 상황 시 더욱 효과적인 대피를 가능하게 함.

프로젝트의 문제점

- 직선 상의 거리를 구했기 때문에 교통 상황과 거리 및 시간을 고려해야함
- 계산 방식이 너무 비효율 적임. 추후 개선이 필요

- 해당 프로젝트는 송파구 내의 대피소만 고려
- 실제 상황에서는 인접한 타 지역의 대피소 위치까지 고려해야

거리 계산 방식

유동인구

- 지금 인구 데이터는 송파구 주민 데이터
- 실제 상황이라면 실시간 송파구 내 유동 인구까지 고려해야함

다른 지역의
대피소 반영

추가 대피소
위치 선정

- 대피 시 골든 타임이라는 개념이 있음(5분)
- 때문에 전 지역에서 5분내로 도착 가능한 지역에 대피소를 세워야 함
(추후 업데이트 예정)