

어디로 ‘밥퍼’를 옮겨야 할까

동대문구 주요현안 해결방안

BOB-UP

0. 개요

AI MAG

0. 개요

1

2

3

4

5

BOB
UP



1. 주제선정

2. 문제점 파악

3. 핵심측정지표 선정

4. 데이터 수집/처리

5. 머신러닝 알고리즘

6. 해결방안 제시

“머신러닝을 활용한
우리 동대문의 문제해결 방안을
제시해볼까”

2025 동대문구
공공데이터 활용
정책 아이디어

총 상금
280만 원

정책 아이디어
공모전



0. 개요

1

2

3

4

5

BOB
UP



1. 주제선정

2. 문제점 파악

3. 핵심측정지표 선정

4. 데이터 수집/처리

5. 머신러닝 알고리즘

6. 해결방안 제시



행복을 여는 동대문구

“비전2026”에서는
이런 문제를
제시했어요

4

밥퍼 민원을 해소하고 민생 지원을 강화하겠습니다

핵심 과제

9 밥퍼 민원 해소

- 사업내용
- 주변 환경 정비·순찰 및 인근 지역 금연거리 지정
 - 무료 급식 배달 서비스 시행
 - 주민 협의체 구성
 - 주변 학교 통학로 및 교차로 일대 안심 보안관 운영

추진계획

- 주변 순찰 및 무단투기 단속
- 밥퍼 무료 급식소 이용 노숙인을 위한 배달 서비스 실시
- 주민 협의체 정기 회의 개최
- 안심보안관 선발·배치, 안전한 통학로 조성





1. 주제선정

2. 문제점 파악

3. 핵심측정지표 선정

4. 데이터 수집/처리

5. 머신러닝 알고리즘

6. 해결방안 제시



[지혜를 찾아서] '밥퍼 35년' 다일공동체 최일도 목사



최일도 목사(가운데)와 박펴나눔운동본부 김미경 부본부장(왼쪽), 박희진 간사가 다음날 제공할 설영당을 준비하고 있다. 김경빈 기자

0. 개요

1

2

3

4

5

BOB
UP



1. 주제선정

2. 문제점 파악

3. 핵심측정지표 선정

4. 데이터 수집/처리

5. 머신러닝 알고리즘

6. 해결방안 제시

생활 안전 기반 강화

3-1

법폐 민원 해소 및 이용 서비스 개선 추진

- ▶ 사회복지과
- ▶ 동봉과
- ▶ 청소행정과
- ▶ 보건정책과

실태와 과제

- * '법폐' 이용자 및 주변 보행자의 쓰레기 무단투기 행위로 민원 발생
- * 깨끗한 환경 조성을 위해 무단투기 단속, 길거리 청소 등 조치 필요
- * 다수인의 무료 급식자 및 노숙인들이 모이는 법폐 인근 지역을 금연거리로 지정하여 간접흡연으로 인한 폐해를 예방하고 구민 건강 증진에 기여
- * 법폐나눔동봉부 무료 급식소 운영 현황 (2022.11. 기준)
 - 이용 현황 : 800여 명(주 6회), 11:00~12:30/급식 단식 : 3,500원(1인, 1식)
 - 일 2회 취약 지역 거리 노숙인 순찰 및 계도
- * 법폐 주변 학교 등 하굣길 및 관내 초등학교 통학로 일대에 노인 일자리 활동 인력을 동대문구 인심보안관으로 배치하여 안전한 등-하굣길 조성 및 노인 일자리 활성화 추진

추진 계획

- * 법폐 주변 순찰 및 무단투기 단속
 - 단속 인원 : 무단투기 단속반(임기제공무원 1명, 공공근로 2명)
 - 단속 주기 : 주 2~3회 법폐 배식 시간 짐작 단속
- * 환경공무원 배치 및 주기적 청소
 - 단속반 등 법폐 인근(황률로~답십리로) 금연거리 지정
 - 금연거리(구역) 지정을 위한 의견 수렴 및 지정 고시·광고
 - 금연거리(구역) 안내 표시판 및 노면 표지판 설치
 - 금연 계도 및 홍보
 - (상주) 거리 노숙인 무료 급식 배달
 - 무료 급식 이용 회망 여부 확인 및 컵라면 등 부식 배달 서비스 제공
 - 거리 노숙인 무료 급식소 이용 및 결식 우려자 현황 확인
 - 배달 서비스 이용자 시설 입소 및 지원 사업 참여 유도
 - 법폐 경로식당 이용 저소득 어르신 지원 유형 변경
 - 경로식당 이용 저소득 어르신 지원 주제 실태 파악
 - 급식 지원 사업 권역별 수행 기관 연계로 중심 도시락 배달
 - 주요 거점 지역을 중심으로 노숙인 순찰 강화 및 부식 제공 등 노숙인 보호 강화
 - 법폐 관련 주민 협의체 조성 계획 수립 및 주민 협의체 정기회의 개최
 - 법폐 주변 학교 등-하굣길 및 관내 초등학교 통학로 일대 인심보안관 활동 개시

연차별 추진 계획	주요 사업	2023	2024	2025	2026		
		법폐 주변 무단투기 단속반 등 배치(명)	4	4	4		
금연거리 지정 및 홍보	지정 및 고시·광고	계도 및 홍보	계도 및 홍보	계도 및 홍보	계도 및 홍보		
저소득 어르신 급식 지원 사업 도시락 전환	도시락 배달 추진	도시락 배달 추진	도시락 배달 추진	도시락 배달 추진	도시락 배달 추진		
법폐 관련 주민 협의체 구성	계획 수립 및 협의체 구성	협의체 운영	협의체 운영	협의체 운영	협의체 운영		
안심보안관 운영(명)	50	60	70	75			
연차별 투자 수요 (단위 : 백만원)	구분	계	2023	2024	2025		
		2026	3,019	712	740	769	798
금연거리 지정 및 홍보	계	2	2	-	-	-	-
저소득 어르신 급식 지원 사업 도시락 전환	641	116	146	175	204		
안심보안관 운영	2,376	594	594	594	594	594	

해결 방안 :
무료 급식소를 위한 위치선정?



0. 개요

1

2

3

4

5

BOB
UP



1. 주제선정

2. 문제점 파악

3. 핵심측정지표 선정

4. 데이터 수집/처리

5. 머신러닝 알고리즘

6. 해결방안 제시

침해성

“집주변이 너무 분비는 거 같아요” “아이들의 등하교길이 위험해요”

거주지, 교육시설로부터 멀리

1. 주제선정

2. 문제점 파악

3. 핵심측정지표 선정

4. 데이터 수집/처리

5. 머신러닝 알고리즘

6. 해결방안 제시

침해성

“집주변이 너무 분비는 거 같아요” “아이들의 등하교길이 위험해요”

거주지, 교육시설로부터 멀리

“거리가 너무 더러워져요”

위생성

흡연가능 구역 근처로

1. 주제선정

2. 문제점 파악

3. 핵심측정지표 선정

4. 데이터 수집/처리

5. 머신러닝 알고리즘

6. 해결방안 제시

침해성

“집주변이 너무 분비는 거 같아요” “아이들의 등하교길이 위험해요”

거주지, 교육시설로부터 멀리

“거리가 너무 더러워져요”

위생성

접근성

“노숙인, 저소득 노인분들이 많이 이용했으면 ...”

출연가능 구역 근처로

교통시설, 복지시설
근처로

1. 주제선정

2. 문제점 파악

3. 핵심측정지표 선정

4. 데이터 수집/처리

5. 머신러닝 알고리즘

6. 해결방안 제시

침해성

“집주변이 너무 분비는 거 같아요” “아이들의 등하교길이 위험해요”

거주지, 교육시설로부터 멀리

“거리가 너무 더러워져요”

위생성

접근성

“노숙인, 저소득 노인분들이 많이 이용했으면 ...”

교통시설, 복지시설
근처로

안전성

“근처에 경찰서가 있다면 ...”

방범시설이 많은 곳에

1. 주제선정

2. 문제점 파악

3. 핵심측정지표 선정

4. 데이터 수집/처리

5. 머신러닝 알고리즘

6. 해결방안 제시

침해성

“집주변이 너무 분비는 거 같아요” “아이들의 등하교길이 위험해요”

거주지, 교육시설로부터 멀리

“거리가 너무 더러워져요”

위생성

접근성

“노숙인, 저소득 노인분들이 많이 이용했으면 ...”

교통시설, 복지시설
근처로

안전성

“근처에 경찰서가 있다면 ...”

경쟁성

방범시설이 많은 곳에

“기존 사업이 커버하지 못하는 곳으로 가고 싶은데 ..”

급식지원사업 시설로부터 멀리



0. 개요

1

2

3

4

5

BOB
UP



1. 주제선정

2. 문제점 파악

3. 핵심측정지표 선정

4. 데이터 수집/처리

5. 머신러닝 알고리즘

6. 해결방안 제시

거주지

국토교통부 건축HUB 건축물대장 정보 서비스(API)

거주지 / 비거주지 좌표
데이터

DATA .GO .KR
공공데이터포털

0. 개요

1

2

3

4

5

BOB
UP



1. 주제선정

2. 문제점 파악

3. 핵심측정지표 선정

4. 데이터 수집/처리

5. 머신러닝 알고리즘

6. 해결방안 제시

거주지

국토교통부 건축HUB 건축물대장 정보 서비스(API)

거주지 / 비거주지 좌표 데이터

위생성

서울특별시 동대문구 흡연시설 현황
서울특별시 동대문구 가로휴지통
현황

DATA
. GO . KR
공공데이터포털

0. 개요

1

2

3

4

5

BOB
UP



1. 주제선정

2. 문제점 파악

3. 핵심측정지표 선정

4. 데이터 수집/처리

5. 머신러닝 알고리즘

6. 해결방안 제시

거주지

국토교통부 건축HUB 건축물대장 정보 서비스(API)

거주지 / 비거주지 좌표 데이터

위생성

서울특별시 동대문구 흡연시설 현황

서울특별시 동대문구 가로휴지통
현황

침해성

전국 초중고등학교 위치데이터

서울 지역 대학교 위치데이터

DATA
. GO . KR
공공데이터포털

0. 개요

1

2

3

4

5

BOB
UP



1. 주제선정

2. 문제점 파악

3. 핵심측정지표 선정

4. 데이터 수집/처리

5. 머신러닝 알고리즘

6. 해결방안 제시

거주지

국토교통부 건축HUB 건축물대장 정보 서비스(API)

거주지 / 비거주지 좌표 데이터

위생성

서울특별시 동대문구 흡연시설 현황

서울특별시 동대문구 가로휴지통

현황

침해성

전국 초중고등학교 위치데이터

서울 지역 대학교 위치데이터

안전성

경찰청 및 경찰관서 위치주소

현황

행정안전부 CCTV 정보

DATA
공공데이터포털
.GO.KR

0. 개요

1

2

3

4

5

BOB
UP



1. 주제선정

2. 문제점 파악

3. 핵심측정지표 선정

4. 데이터 수집/처리

5. 머신러닝 알고리즘

6. 해결방안 제시

거주지

국토교통부 건축HUB 건축물대장 정보 서비스(API)

거주지 / 비거주지 좌표 데이터

위생성

서울특별시 동대문구 흡연시설 현황

서울특별시 동대문구 가로휴지통 현황

침해성

전국 초중고등학교 위치데이터

서울 지역 대학교 위치데이터

안전성

경찰청 및 경찰관서 위치주소

현황

행정안전부 CCTV 정보

경쟁성

급식지원사업 위치데이터

DATA
공공데이터포털
.GO.KR

0. 개요

1

2

3

4

5

BOB
UP



1. 주제선정

2. 문제점 파악

3. 핵심측정지표 선정

4. 데이터 수집/처리

5. 머신러닝 알고리즘

6. 해결방안 제시

거주지

국토교통부 건축HUB 건축물대장 정보 서비스(API)

거주지 / 비거주지 좌표 데이터

위생성

서울특별시 동대문구 흡연시설 현황
서울특별시 동대문구 가로휴지통 현황

침해성

전국 초중고등학교 위치데이터
서울 지역 대학교 위치데이터

안전성

경찰청 및 경찰관서 위치주소
행정안전부 CCTV 정보

경쟁성

급식지원사업 위치데이터

노인인구

동대문구 주민등록인구통계
유동인구 데이터 (API)

DATA . GO . KR
공공데이터포털

0. 개요

1

2

3

4

5

BOB
UP



1. 주제선정

2. 문제점 파악

3. 핵심측정지표 선정

4. 데이터 수집/처리

5. 머신러닝 알고리즘

6. 해결방안 제시

거주지

국토교통부 건축HUB 건축물대장 정보 서비스(API)

거주지 / 비거주지 좌표 데이터

위생성

서울특별시 동대문구 흡연시설 현황

서울특별시 동대문구 가로휴지통

현황

침해성

전국 초중고등학교 위치데이터

서울 지역 대학교 위치데이터

안전성

경찰청 및 경찰관서 위치주소

현황

행정안전부 CCTV 정보

경쟁성

급식지원사업 위치데이터

노인인구

동대문구 주민등록인구통계

유동인구 데이터 (API)

접근성

전국 지하철 위치 데이터

전국 버스 위치 데이터

전국 복지시설 데이터

DATA . GO . KR
공공데이터포털

1. 주제선정

2. 문제점 파악

3. 핵심측정지표 선정

4. 데이터 수집/처리

5. 머신러닝 알고리즘

6. 해결방안 제시

거주지

국토교통부 건축 HUB 건축물대장 정보
거주지 / 비거주지 주민 데이터

비스 (API)

침해

접근

(Open)
공공데이터
활용!

위생성

흡연시설 현황

현황

전성

지역 위치주소
현황

전부 CCTV 정보

주민 인구통계

노인인구

DATA 공공데이터포털
.GO.KR

0. 개요

1

2

3

4

5

BOB
UP



1. 주제선정

2. 문제점 파악

3. 핵심측정지표 선정

4. 데이터 수집/처리

5. 머신러닝 알고리즘

6. 해결방안 제시

K-Means 사용
다기준 의사결정법



MCLP 기반
사용자 정의 알고리즘

양상을 기법을 활용하여
우수후보지 선정



1. 주제선정

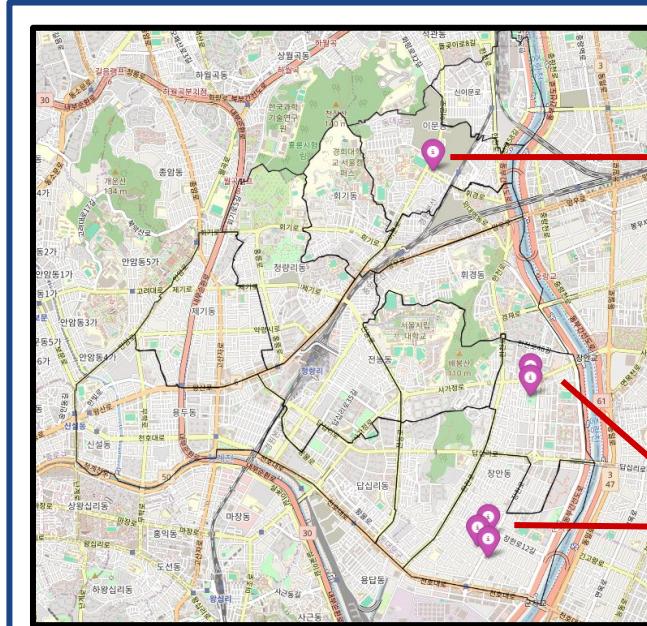
2. 문제점 파악

3. 핵심측정지표 선정

4. 데이터 수집/처리

5. 머신러닝 알고리즘

6. 해결방안 제시



이문동

장안동

0. 개요

1. EDA



1. Data 탐색

2. 위도, 경도

3. category

4. 이상치 제거

1. 토대대장 건축물
2. 동대문구 흡연시설 위치
3. 동대문구 가로 휴지통 위치
4. 전국 초중등 학교 주소
5. 고등 학교별 주소
6. 동대문구 경찰서 및 파출소 주소
7. 동대문구 cctv
8. 동대문구 동행제작소 주소
9. 동대문구 복지시설 주소
10. 동대문구 행정동별 노인인구 통계

1. 주소 → 위도, 경도
2. 데이터를 category
3. 이상치 확인
4. 하나의 DataFrame



1. Data 탐색

2. 위도, 경도

3. category

4. 이상치 제거

1. geopy

```
from geopy.geocoders import Nominatim
```

2. geokakao

```
import geokakao as gk
```

```
def lat_lon(address):
```

```
    '''
```

```
    address: 문자열 타입의 도로명 및 번지 주소
```

```
    '''
```

```
    lation = gk.convert_address_to_coordinates(address)
```

```
    if lation == None: # 만약 실제 주소가 아니라면
```

```
        return np.nan, np.nan # None값 반환하므로 nan 반환
```

```
    else:
```

```
        lat, lon = lation
```

```
        # 반환된 lat, lon은 문자열 타입이므로 실수형으로 캐스팅 후 반환
```

```
        return float(lat), float(lon)
```



1. Data 탐색

2. 위도, 경도

3. category

4. 이상치 제거

```
from geopy.geocoders import Nominatim  
  
geolocator = Nominatim(user_agent="my_geocoder_app")  
location = geolocator.geocode('서울특별시 동대문구 신설동 4-4')  
  
print(f"Address: {location.address}")  
print(f"Latitude: {location.latitude}")  
print(f"Longitude: {location.longitude}")
```

- 위도: 0.0045
- 경도: 0.0013
- 약 518m 차이

Address: 신설동, 용신동, 동대문구, 서울특별시, 02586, 대한민국
Latitude: 37.573478
Longitude: 127.0257068

```
location = gk.convert_address_to_coordinates('서울특별시 동대문구 신설동 4-4')  
print(f"Latitude: {location[0]}")  
print(f"Longitude: {location[1]}")
```

Latitude: 37.5780061765192
Longitude: 127.02708308127



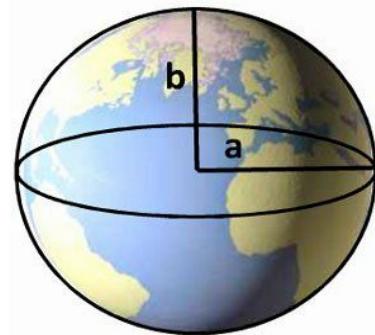
1. Data 탐색

2. 위도, 경도

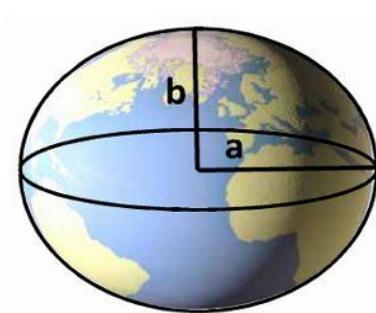
3. category

4. 이상치 제거

HAVERSINE



위도 (Lat) = b
경도 (Lon) = a



```
R = 6371000 # 지구 반지름
phi_1 = math.radians(lat1)
phi_2 = math.radians(lat2)

delta_phi = math.radians(lat2 - lat1)
delta_lambda = math.radians(lon2 - lon1)

a = math.sin(delta_phi / 2.0) ** 2 + math.cos(phi_1) * math.cos(phi_2) * math.sin(delta_lambda / 2.0) ** 2

c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))

meters = R * c
```



1. Data 탐색

2. 위도, 경도

3. category

4. 이상치 제거

토지대장 건축물 (27,268)

주거 건물

(21,530)

비주거 건물

(5,574)

변환 성공

(20,170)

변환 실패

(1,366)

변환 성공

(20,170)

변환
실패

(158)

약 94.4 % 변환 성공

변환 실패 주소

1. 서울특별시 동대문구 신설동 4번지

실제 올바른 주소 입력

1. 서울특별시 동대문구 신설동 4-4
 2. 서울특별시 동대문구 신설동 4-6
 3. 서울특별시 동대문구 신설동 4-8
 4. 서울특별시 동대문구 신설동 4-10
-

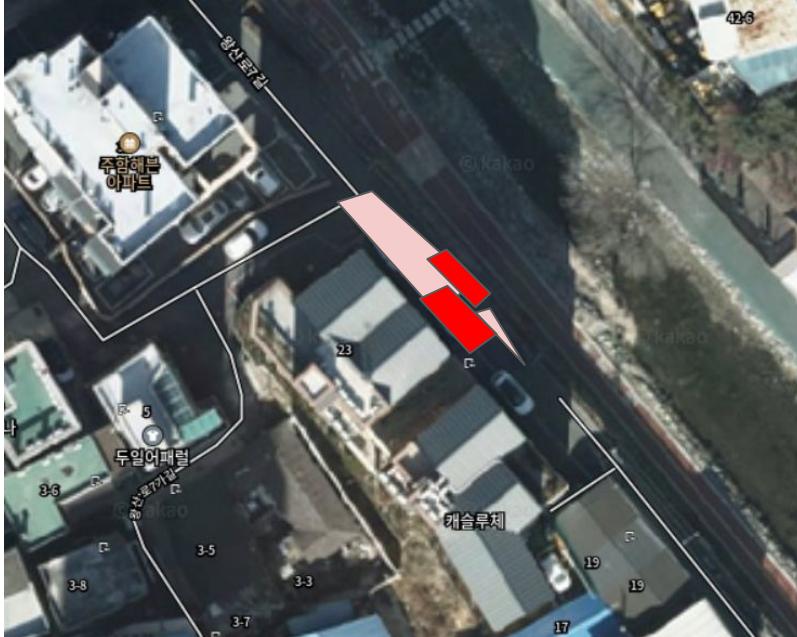


1. Data 탐색

2. 위도, 경도

3. category

4. 이상치 제거



0

1. EDA

2

3

4

5

BOB
UP

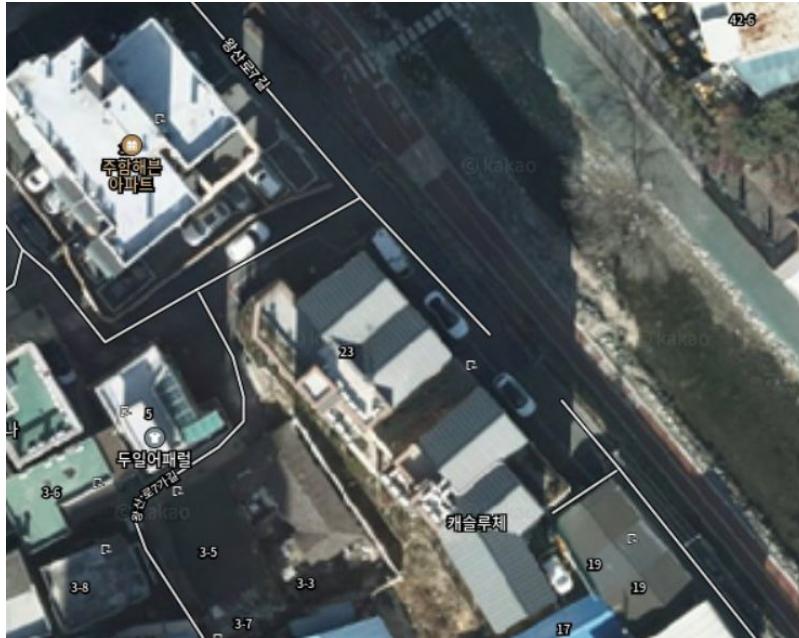


1. Data 탐색

2. 위도, 경도

3. category

4. 이상치 제거





1. Data 탐색

접근성

위생성

안전성

- 버스
- 지하철
- 복지시설
- 흡연시설
- 쓰래기통
- 경찰서
- CCTV

2. 위도, 경도

3. category

침해성

경쟁성

- 초중고
- 노유자 시설
- 주거시설
- 대학교
- 동행제작소

4. 이상치 제거



1. Data 탐색

2. 위도, 경도

3. category

4. 이상치 제거

1. 반환되지 않은 시설물은 제거

주소	위도	경도
서울특별시 동대문구 장안벚꽃로 309	37.58567641	127.0700416
서울특별시 동대문구 망우로6길 48	37.585736	127.0582142
서울특별시 동대문구 전농로16길 61		
서울특별시 동대문구 사가정로 59	37.57452207	127.0519943
서울특별시 동대문구 신이문로 16	37.60189649	127.064885

2. 목적과 다른 시설물 제거.

경희중고 앞
버스 정류소

교통시설

밥퍼 대체 위치
후보군

교육시설

경희중고



1. Data 탐색

2. 사용자 정의 함수

3. 시설물 데이터

4. 이상치

제거 항목 (100개)

- e편한세상정계센트럴포레 아파트
 - 경희중고
 - 답십리2동두산 아파트
 - 답십리2동한양 아파트 앞
 - 답십리초등 학교 앞 답십리촬영소 .
 - 우성그린 아파트
 - 대광고등 학교 앞
-



1. Data 탐색

2. 사용자 정의 함수

3. 시설물 데이터

4. 이상치

1

좌표계 변환 함수

transform_coordinates('주소') : 주소를 입력하면 위도 경도로 반환

```
from pyproj import Transformer

def transform_coordinates(lat, lon):
    # KATECH 좌표계 정의 (TM 투영, GRS80 타원체)
    katech_proj = "+proj=tmerc +lat_0=38 +lon_0=127 +k=0.9999 +x_0=200000
                    +y_0=600000 +ellps=GRS80 +units=m +no_defs"
    # WGS84 좌표계 정의
    wgs84_proj = "+proj=latlong +datum=WGS84 +ellps=WGS84"
    # WGS84 -> KATECH 변환기 생성
    transformer = Transformer.from_proj(katech_proj, wgs84_proj)
    # 변환 수행 (위도, 경도 -> x, y)
    x, y = transformer.transform(lon, lat)
    return x, y
```



1. Data 탐색

2. 사용자 정의 함수

3. 시설물 데이터

4. 이상치

2

국토교통부 건축 HUB로부터 (비)거주지 주소 추출

building_df(api, sigungu_code, beopjeong_dong_code) :

```
def building_df (api_key, sigungu_code, beopjeong_dong_code):
    df_top = None # 초기 데이터프레임 설정
    for code in beopjeong_dong_code # 동대문구의 법정동을 반복하는 코드
        base_url = # url에 시군구 코드와 법정동 코드 및 apy key 입력
        response = requests.get(base_url)
        total_count = (int(json.loads(response.text)["response"]["body"]["totalCount"])/100) + 2
        for page in range(1, total_count): # 1 페이지 부터 최대 페이지까지 반복
            url = # url에 시군구 코드와 법정동 코드 및 apy key 입력
            response = requests.get(url)
            data_dict = json.loads(response.text)
            if "item" not in data_dict["response"]["body"]["items"]:
                continue # 데이터가 없을 경우 건너뜀
            df_bottom = pd.DataFrame(data_dict["response"]["body"]["items"]["item"])
            if df_top is None:
                df_top = df_bottom # 첫 데이터프레임 할당
            else:
                df_top = pd.concat([df_top, df_bottom]) # 이후부터 밑으로 계속 연결
    return df_top.reset_index() # 인덱스 제거 후 반환
```





1. Data 탐색

2. 사용자 정의 함수

3. 시설물 데이터

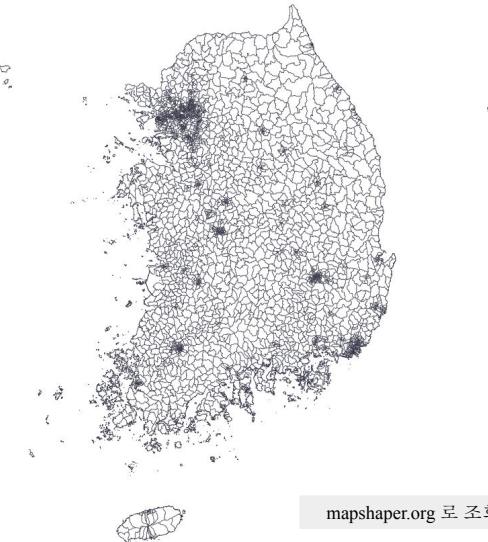
4. 이상치

1

대체 후보지 설정

대한민국 지도 전체 좌표

BND, ADM, DONG, PG ▾





1. Data 탐색

2. 사용자 정의 함수

3. 시설물 데이터

4. 이상치

1

대체 후보지 설정

동대문구 시작화

```

1 ### 1. 대한민국 전체 좌표(shape file) 불러오기
2 shp_path = "/content/drive/MyDrive/DDM/BND_ADMIN_DONG_PG.shp"
3 dong_gdf = gpd.read_file(shp_path)
4
5
6 ### 2. 동대문구 행정동 필터링
7 df_ddm = dong_gdf[dong_gdf['ADM_CD'].str.startswith("11060")]
8
9 ### 3. 공간투영법을 활용하여 변환
10 df_ddm = df_ddm.to_crs(epsg=5179)
11
12 ### 4. 각 동별 중심점 좌표 산출
13 df_ddm['centroid'] = df_ddm.geometry.centroid
14 df_ddm['lat'] = df_ddm['centroid'].y
15 df_ddm['lon'] = df_ddm['centroid'].x
16
17 ### 5. 각 동별 경계선 및 중심점 시작화
18 fig, ax = plt.subplots(figsize=(10, 10))
19 df_ddm.plot(ax=ax, color='none', edgecolor='black', linewidth=1)
20 df_ddm.set_geometry('centroid').plot(ax=ax, color='red', markersize=20)
21
22 ### 6. 동 이름 라벨링
23 if 'ADM_NM' in df_ddm.columns:
24     for idx, row in df_ddm.iterrows():
25         ax.annotate(text=row['ADM_NM'], xy=(row['lon'], row['lat']),
26                     xytext=(3, 3), textcoords="offset points", fontsize=8)
27
28 ax.set_title("동대문구 행정동 경계 및 중심점")
29 plt.show()

```



1. Data 탐색

2. 사용자 정의 함수

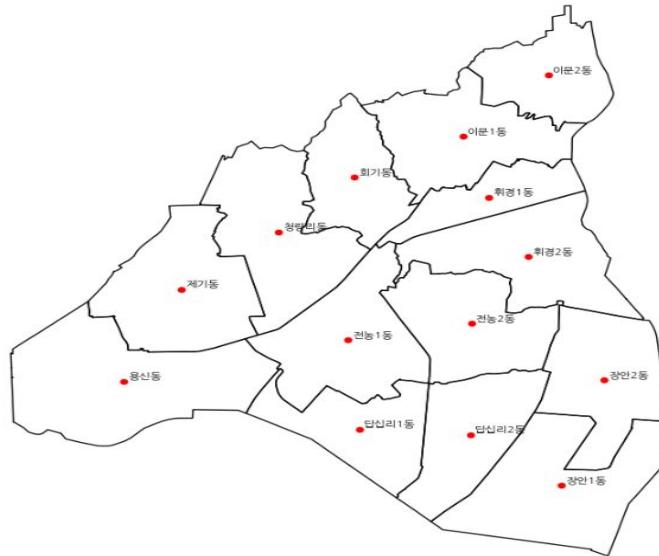
3. 시설물 데이터

4. 이상치

1

대체 후보지 설정

동대문구 시각화





1. Data 탐색

2. 사용자 정의 함수

3. 시설물 데이터

4. 이상치

1

대체 후보지 설정

동대문구 내 대체후보지 1차 선정지



1432개

```
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 1432 entries, 0 to 1431
Data columns (total 3 columns):
 #   Column   Non-Null Count Dtype  
 --- 
 0   geometry  1432 non-null  geometry
 1   lon        1432 non-null  float64 
 2   lat        1432 non-null  float64 
dtypes: float64(2), geometry(1)
memory usage: 33.7 KB
```

0

1. EDA

2

3

4

5

BOB
UP

1. Data 탐색

2. 사용자 정의 함수

3. 시설물 데이터

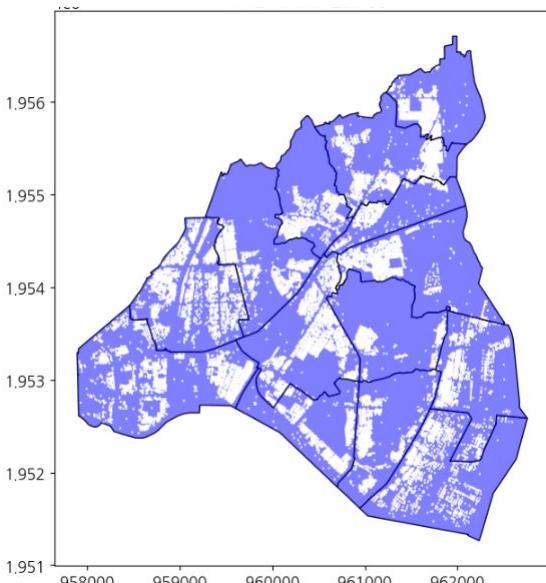
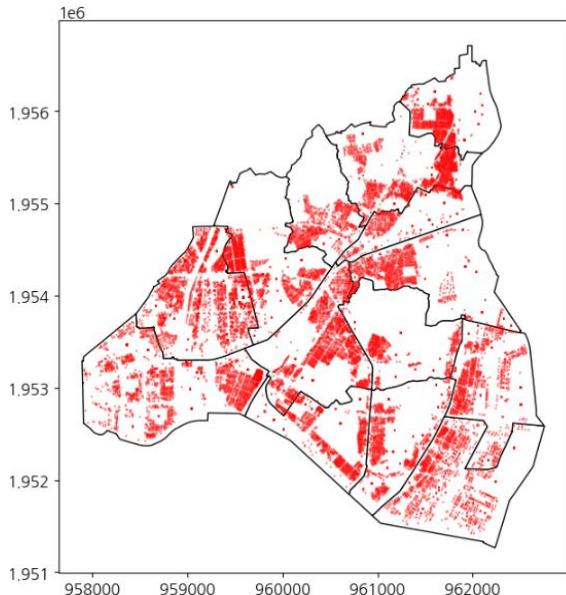
4. 이상치

2

대체 후보지 설정

동대문구 내 주거용 건물 위치

동대문구 내 비거주용 공간



0

1. EDA

2

3

4

5

BOB
UP

1. Data 탐색

2. 사용자 정의 함수

3. 시설물 데이터

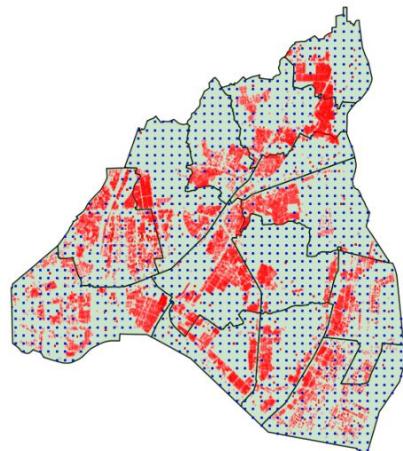
4. 이상치

2

대체 후보지 설정

동대문구 내 대체 후보지 2차 선정

동대문구 내 대체 후보지 2차 선정



```
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 1432 entries, 0 to 1431
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   geometry    1432 non-null   geometry
 1   lon         1432 non-null   float64 
 2   lat         1432 non-null   float64 
dtypes: float64(2), geometry(1)
memory usage: 33.7 KB
```

```
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 1067 entries, 0 to 1066
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   geometry    1067 non-null   geometry
 1   lon         1067 non-null   float64 
 2   lat         1067 non-null   float64 
dtypes: float64(2), geometry(1)
memory usage: 25.1 KB
```

0

1. EDA

2

3

4

5

BOB
UP

1. Data 탐색

2. 사용자 정의 함수

3. 시설물 데이터

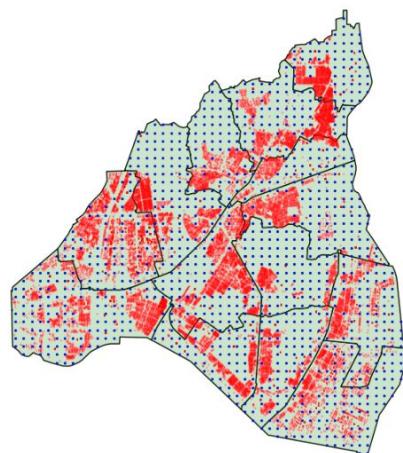
4. 이상치

2

대체 후보지 설정

동대문구 내 대체 후보지 2차 선정

동대문구 내 대체 후보지 2차 선정



```
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 1432 entries, 0 to 1
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   geometry    1432 non-null   geometry
 1   lon          1432 non-null   float64
 2   lat          1432 non-null   float64
dtypes: float64(2), geometry(1)
memory usage: 25.1 KB
```

1,067개

```
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 1067 entries, 0 to 1066
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   geometry    1067 non-null   geometry
 1   lon          1067 non-null   float64
 2   lat          1067 non-null   float64
dtypes: float64(2), geometry(1)
memory usage: 25.1 KB
```



0. 개요

1. EDA

2. K-MEANS

3. FULU



1. 소개

2. 변수 생성

3. class 생성

4. 모델 결과

1 다기준 의사결정법(MCDM)

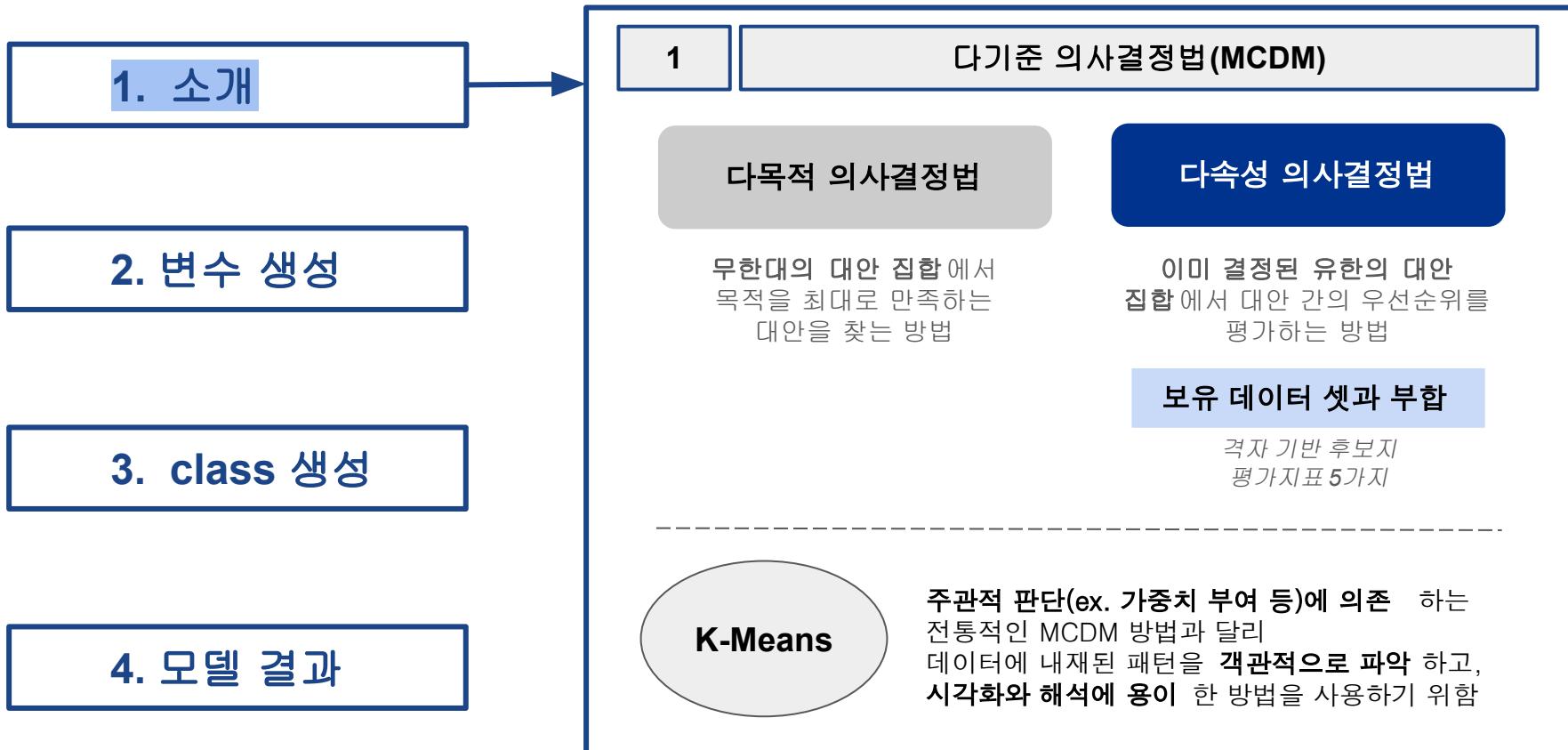
MCDM이란?

- 다수의 기준 또는 목적을 지닌 복잡한 의사결정을 최적화하는 방법론이다.
- 특히 입지선정과 같이 다수의 구성원에게 절대적인 영향을 미치는 정책적 의사결정에는 타당한 의사결정 방법론을 활용하는 것이 필수적이다.

한정숙 and 이다희. (2018). 다기준 정책 의사결정 방법론 연구. 질서경제저널, 21(4), 131-148.

다양한 평가 지표를 통해 후보지를 평가하고자 하는 본 프로젝트의 특성을 고려할 때 다기준 의사결정 접근방식을 활용하는 것이 적절







1. 소개

2. 변수 생성

3. class 생성

4. 모델 결과

1

비교 거리 계산 준비

Haversine 활용 사용자 정의 함수

두 지점의 위도와 경도를 입력 받아 지구 표면의 두 지점 간의 거리를 계산하는 공식

```
def haversine_distance(lat1, lon1, lat2, lon2):
    # 위도(lat1, lat2), 경도(lon1, lon2)를 받아 haversine로 거리(km) 계산.
    return haversine((lat1, lon1), (lat2, lon2), unit='km')
```

시설 분류 기준 결정

평가지표

- 접근성
- 경쟁성
- 침해성
- 안전성
- 위생성

시설분류(대)

- 교통시설
- 복지시설
- 급식지원사업
- 교육시설
- 방범시설
- 흡연시설

보다 세부적인 파악을 위해 시설분류(대)를 분류 기준으로 삼기로 함

0

1

2. K-Means

3

4

5

BOB
UP



1. 소개

2. 변수 생성

3. class 생성

4. 모델 결과

2

비교 거리 변수 생성

1. 기준 거리

밥퍼와 시설과의 최소 거리

2. 후보지별 거리

각 후보지별 시설과의 최소 거리

3. 비교 거리

후보지별 거리 / 기준 거리

geometry	lon	lat	교육시설 _min_dist	교통시설 _min_dist	방범시설 _min_dist	복지시설 _min_dist	흡연시설 _min_dist	급식지원 사업 _min_dist	교육시설 _relative_dist	교통시설 _relative_dist
POINT 127.02374 37.57194)	127.023743	37.571942	0.643940	0.261033	0.029088	0.544913	0.132557	1.628681	1.930693	1.648557
POINT 127.02374 37.57284)	127.023737	37.572844	0.570499	0.161708	0.032346	0.446927	0.080219	1.547370	1.710491	1.021269
POINT 127.02373 37.57375)	127.023731	37.573745	0.488670	0.065196	0.059240	0.350211	0.005056	1.468398	1.465147	0.411747
POINT 127.02373 37.57465)	127.023726	37.574646	0.393900	0.049409	0.029142	0.256204	0.022288	1.392165	1.181006	0.312046
POINT 127.02372 37.57555)	127.023720	37.575548	0.302666	0.032889	0.008825	0.169480	0.029241	1.319146	0.907462	0.207713

2

3



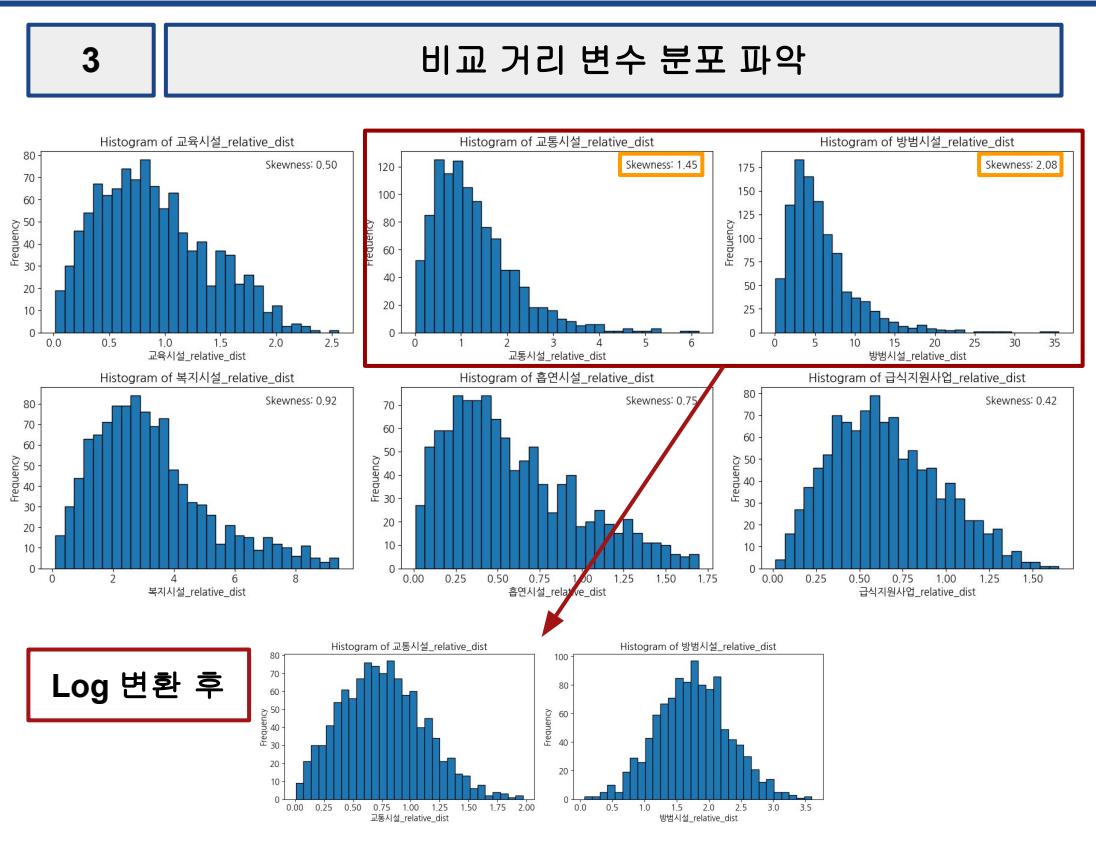
1. 소개

2. 변수 생성

3. class 생성

4. 모델 결과

Log 변환 후



K-Means와 같은 유clidean 거리를 사용하는 알고리즘은 데이터의 분포에 민감하기 때문에 로그변환 적용이 유리

0

1

2. K-Means

3

4

5

BOB
UP



1. 소개

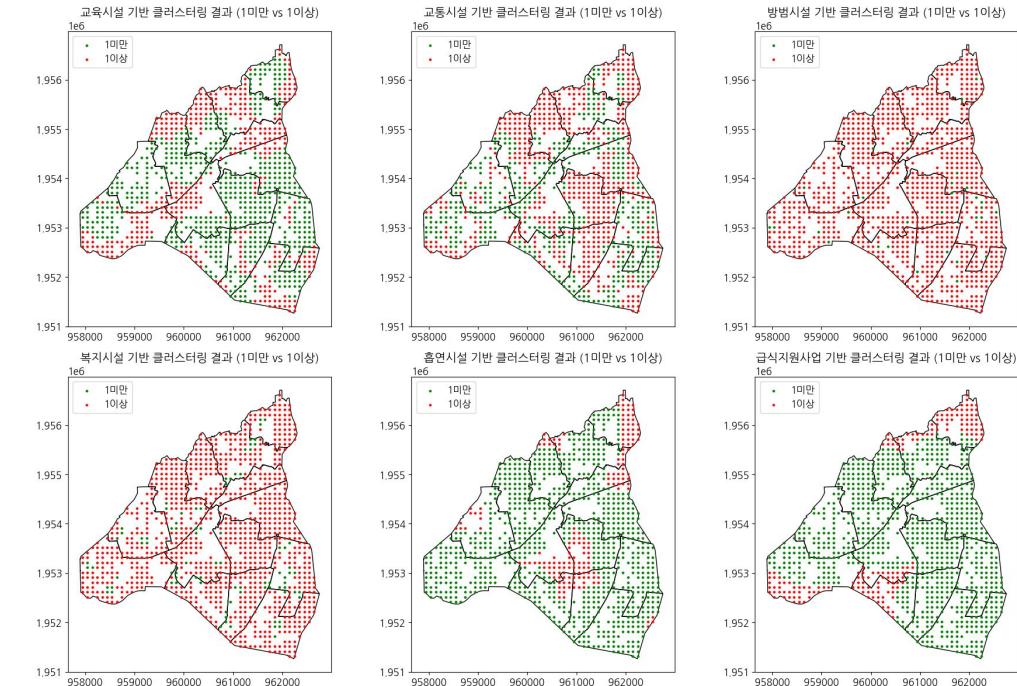
2. 변수 생성

3. class 생성

4. 모델 결과

1

1 초과, 1 미만으로 class 분류





1. 소개

2. 변수 생성

3. cluster 생성

4. 모델 결과

2

전처리

변수들의 방향 통일

가까울 수록 좋은 시설

1. 교통시설
2. 복지시설
3. 방범시설
4. 흡연 시설

멀 수록 좋은 시설

1. 급식지원사업
2. 교육시설

값이 작을 수록
좋은 방향으로
통일

* 식 : $1 / (\text{거리} + 1)$

```
def transform_for_preference(df, preference_dict):
    # 'large_better'인 경우 value -> 1 / (value + 1)로 변환 (값이 작을수록 좋게 만듦)
    # 'small_better'인 경우 그대로 유지
    df_copy = df.copy()

    for col in relative_dist_columns:
        facility = col.replace('_relative_dist', '') # 예: '교통시설'
        if facility not in preference_dict:
            # 혹시 딕셔너리에 없는 경우, small_better로 가정
            continue

        if preference_dict[facility] == 'large_better':
            # 멀수록 좋은 지표 => 1/(distance+1)으로 변환
            df_copy[col + '_transformed'] = 1 / (df_copy[col] + 1)
        else:
            # 가까울수록 좋은 지표 => 그대로 사용
            df_copy[col + '_transformed'] = df_copy[col]

    return df_copy
```



1. 소개

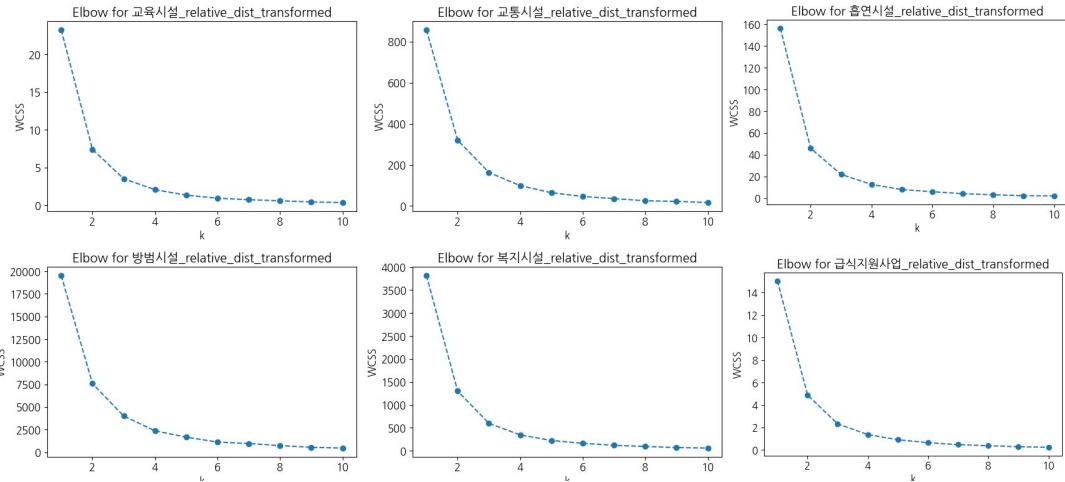
2. 변수 생성

3. class 생성

4. 모델 결과

3

Elbow method 시각화



K = 3 으로 결정

- 1) $k = 2$ 로 설정 시 단순한 분류로 세밀한 차이를 보기 어려움
- 2) 군집의 다양성을 위해서 $K = 3$ 으로 설정 후 5개 이상의 변수가 겹치는 지점을 추출

0

1

2. K-Means

3

4

5

BOB
UP



1. 소개

2. 변수 생성

3. cluster 생성

4. 모델 결과

4

클러스터링 코드

KMEANS 활용 함수 정의

```
def cluster_1d_kmeans(series, k=2, random_state=42):
    data = series.values.reshape(-1, 1)
    kmeans = KMeans(n_clusters=k, random_state=random_state)
    kmeans.fit(data)
    labels = kmeans.labels_
    centroids = kmeans.cluster_centers_.flatten()
    best_cluster = np.argmin(centroids) # 가장 작은 centroid => 우수
    best_bool = pd.Series(labels == best_cluster, index=series.dropna().index)
    return best_bool, centroids, labels
```

차원 변환

비교 거리 값이 가장
작은 데이터 포함

각 시설분류에 대한 우수 후보지 지정

```
chosen_k = 3

for col in transformed_cols:
    facility = col.replace('_relative_dist_transformed', '')

    best_bool, centroids, labels = cluster_1d_kmeans(candidate_gdf_4326_transformed_preference[col], k=chosen_k)
    best_col = facility + '_best'

    # 우수 여부 bool
    candidate_gdf_4326_transformed_preference[best_col] = False
    candidate_gdf_4326_transformed_preference.loc[best_bool.index, best_col] = best_bool
    candidate_gdf_4326_transformed_preference[best_col] = candidate_gdf_4326_transformed_preference[best_col].fillna(False)
```

0

1

2. K-Means

3

4

5

BOB
UP



1. 소개

2. 변수 생성

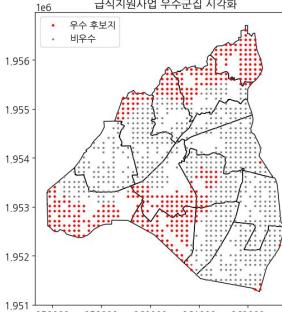
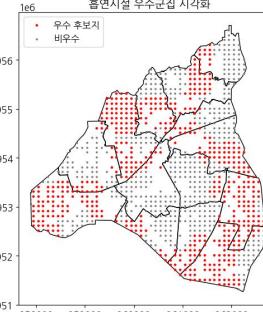
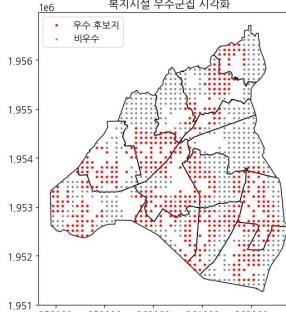
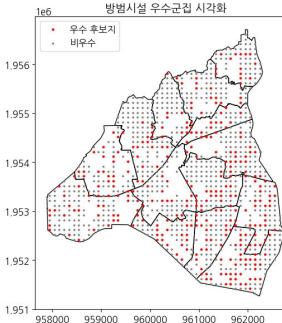
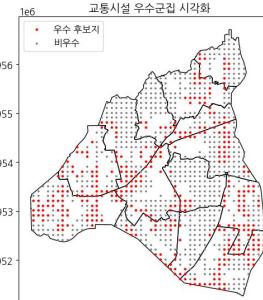
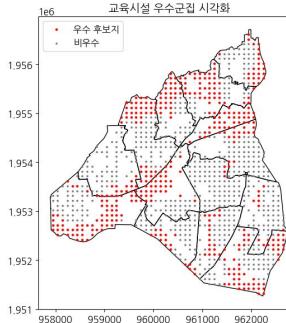
3. cluster 생성

4. 모델 결과

5

시설분류 별 시각화

K = 3



0

1

2. K-Means

3

4

5

BOB
UP



1. 소개

2. 변수 생성

3. cluster 생성

4. 우수후보지

1

우수후보지 선정

```
best_cols = [f + '_best' for f in facility_list]
candidate_gdf_5179_transformed_preference['best_count'] = candidate_gdf_5179_transformed_preference[best_cols].sum(axis=1)

thresholds = [1, 2, 3, 4, 5, 6] # 반복할 threshold 값들 (6개가 있으니까)
fig, axs = plt.subplots(2, 3, figsize=(15, 10))
axs = axs.flatten()

for i, th in enumerate(thresholds):
    ax = axs[i]
    # threshold 적용: best_count >= th → final_best
    candidate_gdf_5179_transformed_preference['final_best'] = candidate_gdf_5179_transformed_preference['best_count'] >= th

    # 행정동 경계 표시
    df_ddm.plot(ax=ax, color='none', edgecolor='black', linewidth=1)

    # 우수 후보지 (final_best == True)
    candidate_gdf_5179_transformed_preference[candidate_gdf_5179_transformed_preference['final_best'] == True].plot(
        ax=ax, color='blue', markersize=8, label='우수 후보지'
    )
    # 그 외 후보지
    candidate_gdf_5179_transformed_preference[candidate_gdf_5179_transformed_preference['final_best'] == False].plot(
        ax=ax, color='gray', markersize=3, label='비우수'
    )

    ax.set_title(f"Threshold = {th}")
    ax.legend()
```

시설분류 별 우수 후보지의 겹치는 개수를 구해서 시각화



1. 소개

2. 변수 생성

3. cluster 생성

4. 우수후보지

2

후보지 dataframe

후보지 dataframe

0

1

2. K-Means

3

4

5

BOB
UP



1. 소개

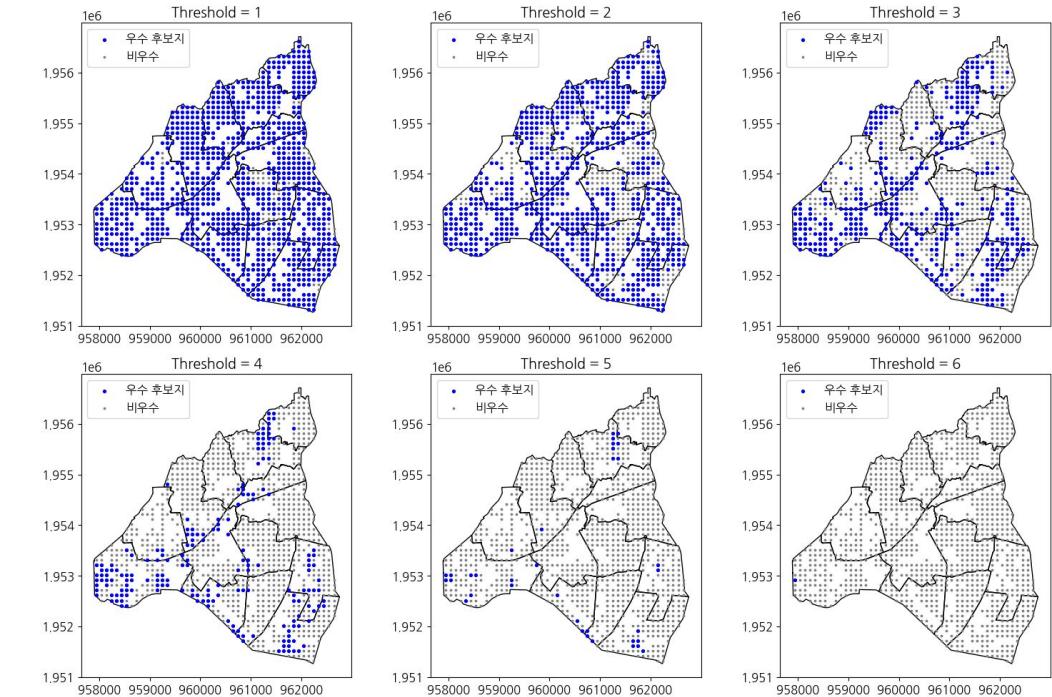
2. 변수 생성

3. cluster 생성

4. 우수후보지

3

겹치는 우수 후보지 시각화



0

1

2. K-Means

3

4

5

BOB
UP



1. 소개

2. 변수 생성

3. cluster 생성

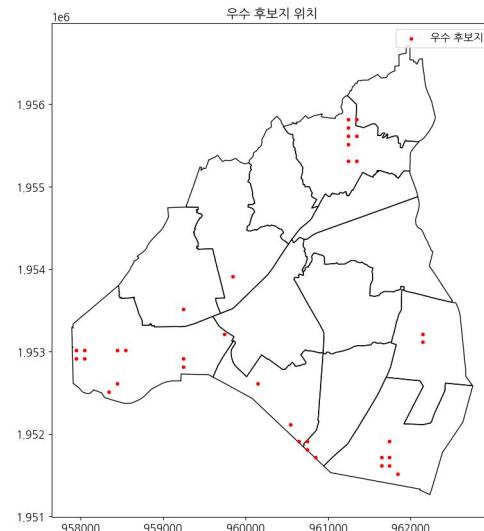
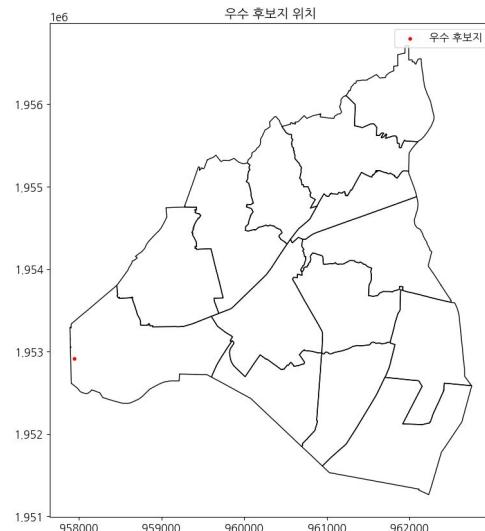
4. 우수후보지

4

겹치는 개수 결정

6개 겹치는 우수 후보지

5개 이상 겹치는 우수 후보지

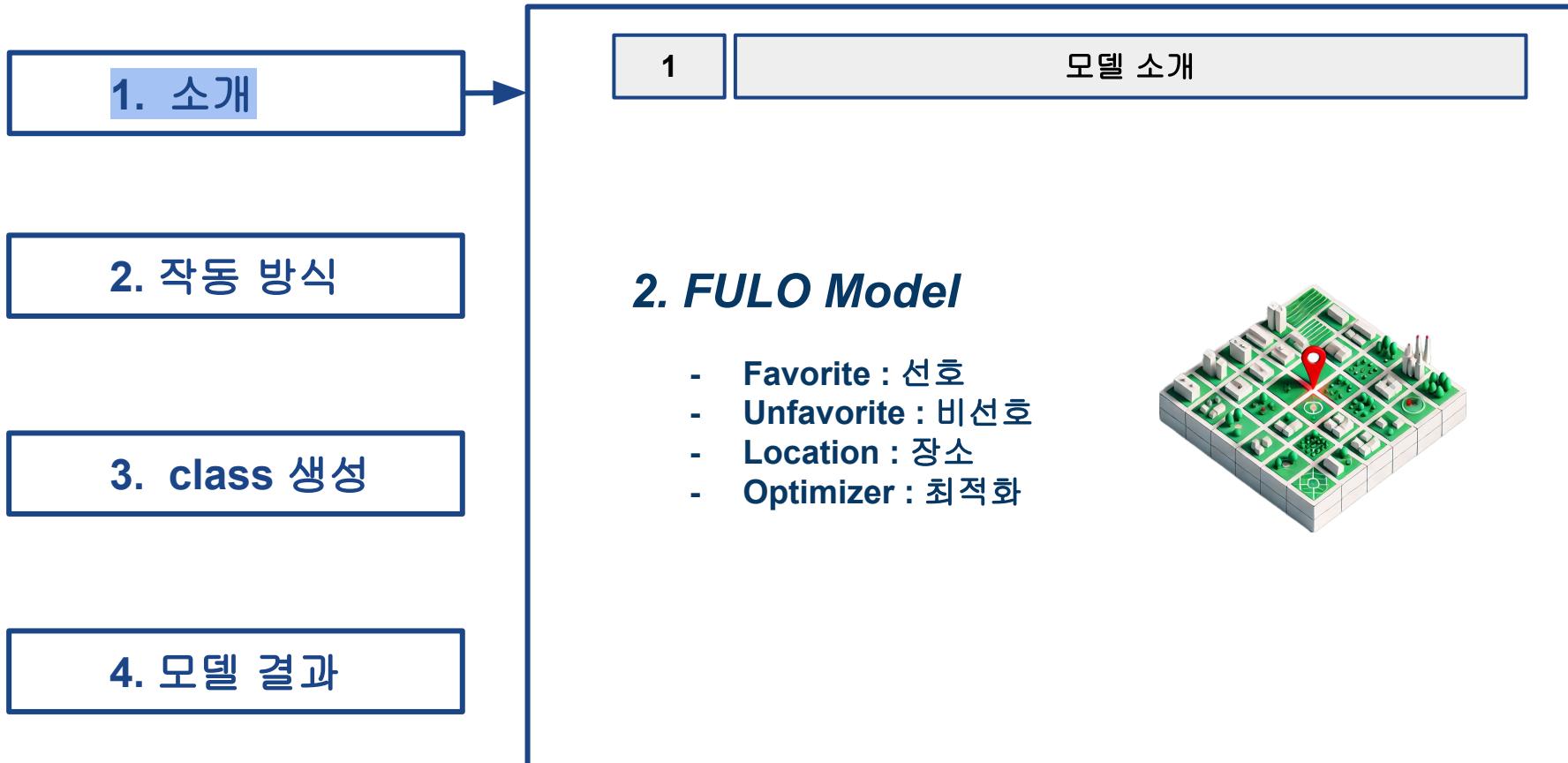


0. 개요

1. EDA

**2. K-MEANS
Extensions**

3. FULU





1. 소개

2. 작동 방식

3. class 생성

4. 모델 결과

1

모델 소개

MCLP(Maximal covering location problem)
: 시설물이 커버하는 수요량을 최대화 하는
위치 선정 알고리즘.

$$\begin{aligned}
 & \text{Maximize} \sum_{i=1} W_i y_i \\
 & \text{Subject to} \quad \sum_{j \in N_i} x_j \geq y_i \quad \forall i \\
 & \quad \sum_j x_j = p \\
 & \quad x_j = \{0,1\} \quad \forall i \\
 & \quad y_j = \{0,1\} \quad \forall i
 \end{aligned}$$

목적함수 AED가 설치되어 총족되는 수요의 합을 최대화

제약조건 1 : 집합 N_i 에 속한 후보지 중 적어도 한 곳에 시설물이 입지하면 i 는 커버

제약조건 2 p : 설치할 실외 AED의 수

제약조건 3 x : 해당 point에 AED가 설치되면 1, 안되면 0

제약조건 4 y : 해당 point가 적어도 하나의 AED에 의해 커버되면 1, 안되면 0

i : 동대문구 내 수요 지점 인덱스 j : 동대문구 내 후보지 인덱스 w : 해당 지점에서의 수요량

N_i : 수요 지점 i 로부터 AED 유효거리 안에 있는 후보지의 집합



1. 소개

2. 작동 방식

3. class 생성

4. 모델 결과

1

모델 소개

동대문구 법정동

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.61

37.60

37.59

37.58

37.57

37.56

37.



> KPMG

1. 소개

2. 작동 방식

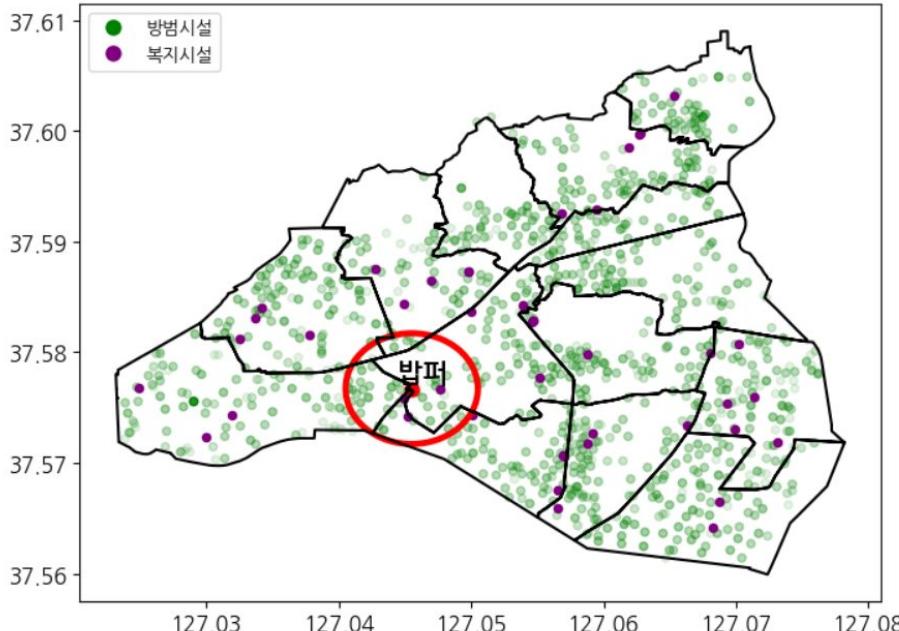
3. class 생성

4. 모델 결과

1

모델 소개

동대문구 법정동



0

1

2

3. FULO

4

5

BOB
UP

1. 소개

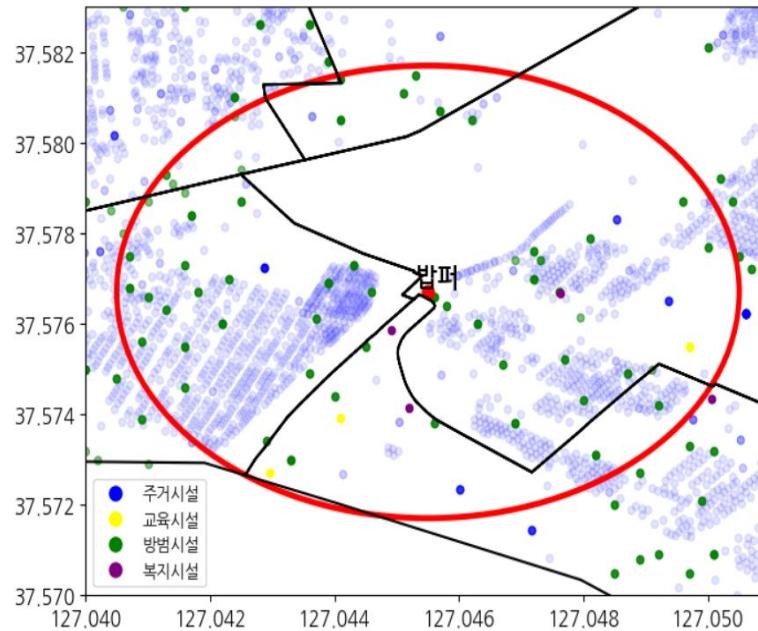
2. 작동 방식

3. class 생성

4. 모델 결과

2

작동방식





1. 소개

2. 작동 방식

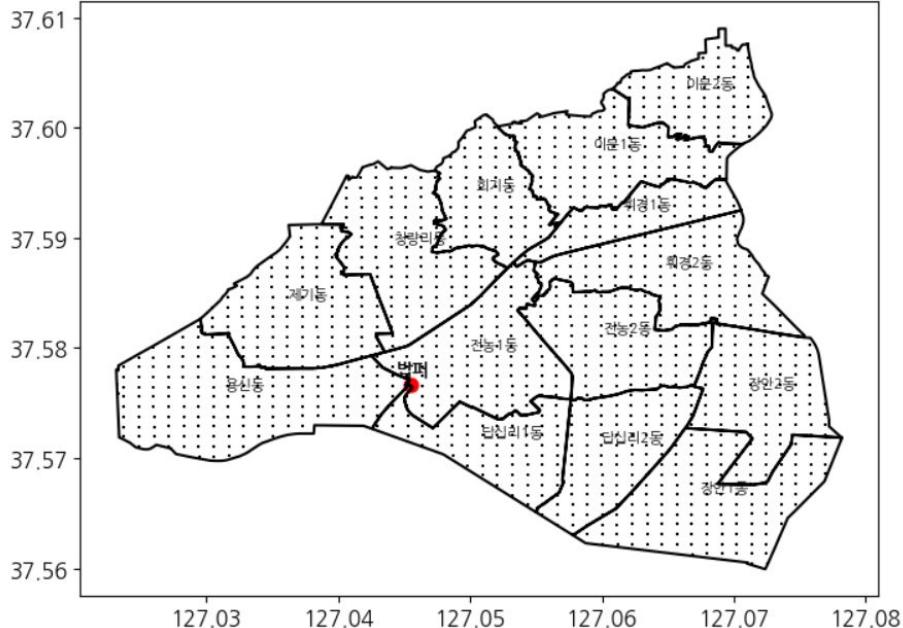
3. class 생성

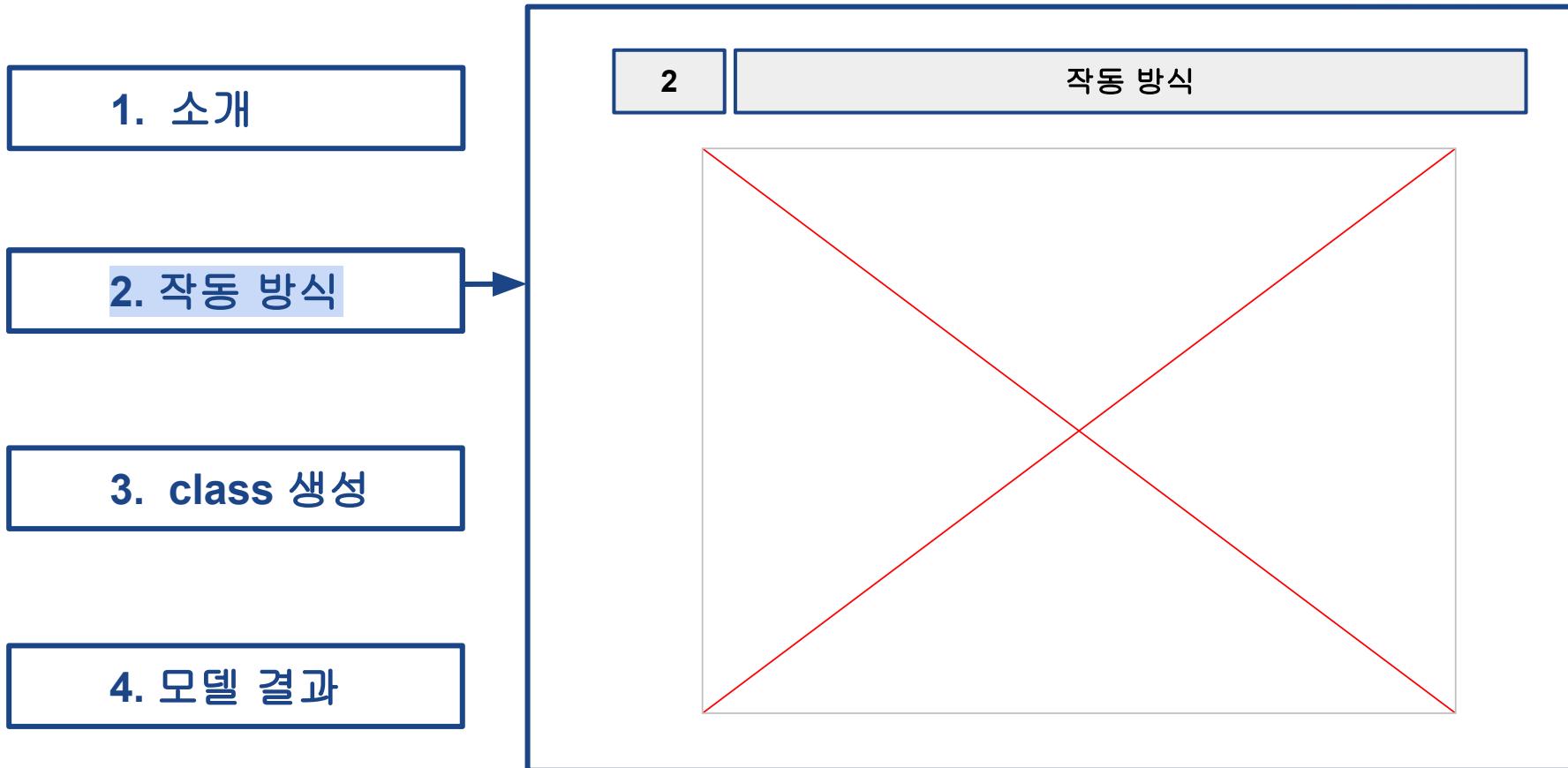
4. 모델 결과

2

작동 방식

동대문구 법정동







1. 소개

2. 작동 방식

3. class 생성

4. 모델 결과

2

모델 작동 방식

1. 밤퍼 근반 550m 이내 시설물 탐색
2. Grid cell마다 근방 550m 탐색
3. 밤퍼 보다 positive 시설물이 많고, negative 시설물 적은 Grid 반환





1. 소개

2. 작동 방식

3. class 생성

4. 모델 결과

3

모델 작성

```
class FULO:  
    def __init__(self, init_location, positive_data, negative_data, radius_km, area):  
        self.init_location = init_location # 초기 위치 (위치 데이터 예상)  
        self.positive_data = positive_data # 긍정적 데이터프레임 (많을수록 좋음)  
        self.negative_data = negative_data # 부정적 데이터프레임 (적을수록 좋음)  
        self.radius_km = radius_km # 반지름 (km 단위)  
        self.area = area # 영역  
  
    def df_copy(self, df):  
        return df.copy()  
  
    def positive_distance(self, standard_locate, data=None):  
        return # 특정 위치로부터 radius_km 해상범위 안의 건물 수 측정  
  
    def negative_distance(self, standard_locate, data=None):  
        return # 특정 위치로부터 radius_km 해상범위 안의 건물 수 측정  
  
    def grid_cell(self):  
        return # 영역 안에 grid를 생성  
  
    def evaluate_location(self, location):  
        return # 긍정 개수, 부정 개수 반환  
  
    def candidates_info_filtering(self, current_pos, current_neg, candidates_info):  
        return # 기준 건물보다 긍정개수가 많고, 부정개수가 적은 좌표 추출  
  
    def recommend_better_location(self):  
        return # 현재보다 더 좋은 좌표를 반환
```





1. 소개

2. 작동 방식

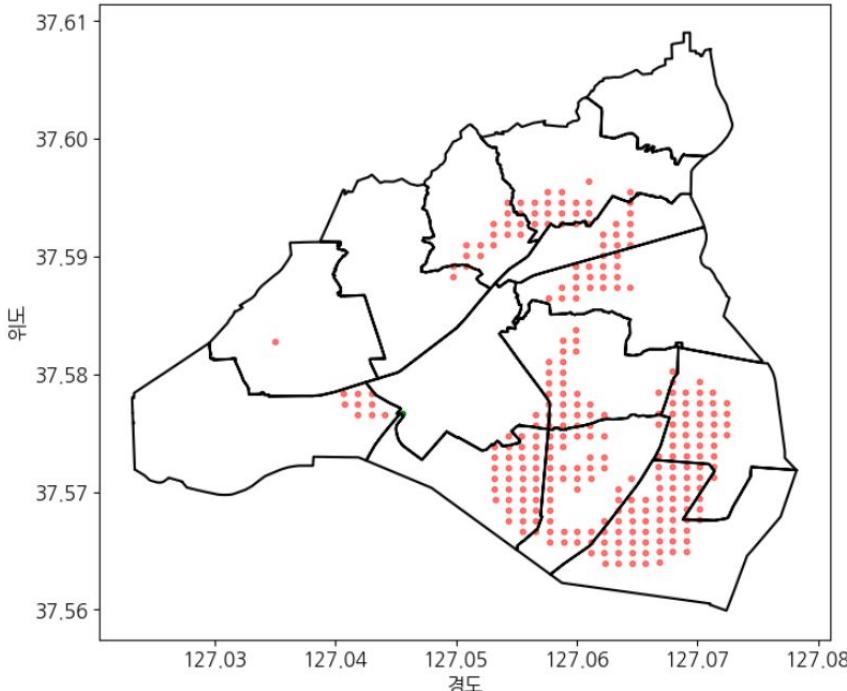
3. class 생성

4. 모델 결과

4

대체 후보지 설정

동대문구 법정동





1. 소개

2. 작동 방식

3. class 생성

4. 모델 결과

4

Model평가

1. 현재 위치보다 negative 요소가 적은 위치를 찾는 모델
2. 초기점으로 positive 요소가 많은 마을 이위로 빼고

$$1 - \left(\frac{\text{모델 추천 위치의 } 550m \text{ 반경이내 negative 개수}}{\text{밥퍼 위치 } 550m \text{ 반경이내 negative 개수}} \right)$$

rank 1: [37.56490575716029, 127.06908049830815],	: 46.424%
rank 2: [37.56397943392325, 127.06229199126444],	4.815%
rank 3: [37.57479118510581, 127.06109624561525],	3.266%
rank 4: [37.58742236150174, 127.06442007446036],	3.206%
rank 5: [37.57659804749267, 127.0622181384529],	.147%

최저 점수는 location:[37.58280339643234, 127.03499960706336], 향상 성 0.06%





1. 소개

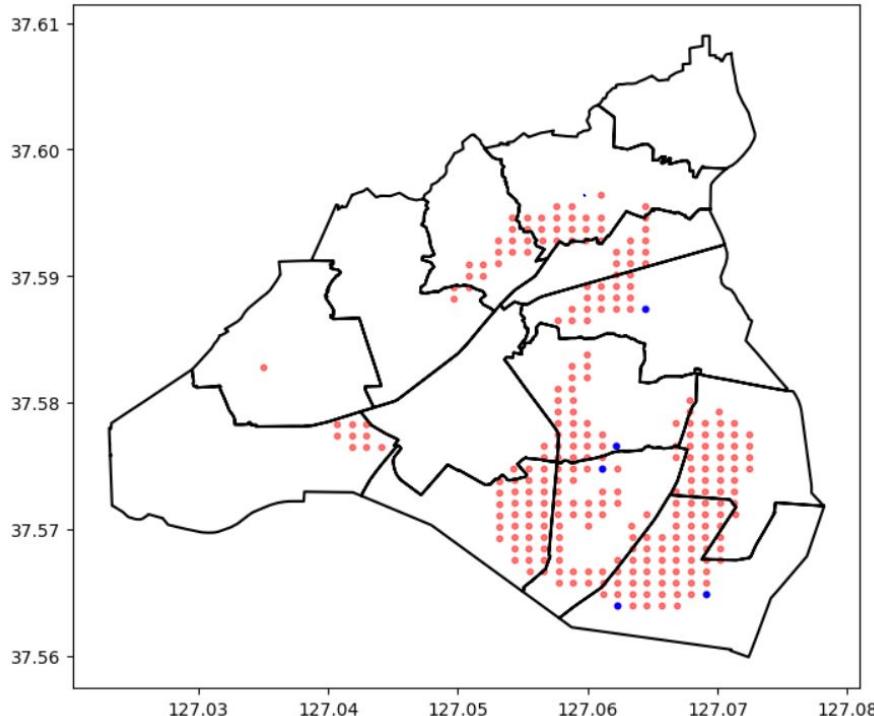
2. 작동 방식

3. class 생성

4. 모델 결과

4

대체 후보지 설정



0. 개요

1. EDA

2~3. ML

4. ENSEMBLE



1. 설명

1

K-Means + FULO

2. 시각화

Shapely

Geopandas

3. 최종 후보지

mapping()

: Geometry 객체를
GeoJSON 형식으로 반환하여
시각화할 수 있는 형태로
바꿈

공간 조인 (sjoin)

: 두 dataset의 객체들이
공간적으로 어떻게
연관되어 있는지에 따라
데이터 연결

4. 후보지 결정

행정동 경계 데이터
=> GeoJSON 형식으로 변환

**K-Means 최종 후보지 &
FULO 최종 후보지
=> intersect**

final_best	index_right	location	positive	negative	lat_right	lon_right
False	148	[37.59642303426622, 127.0609692171382]	225	1661	37.596423	127.060969





1. 설명

2. 시각화

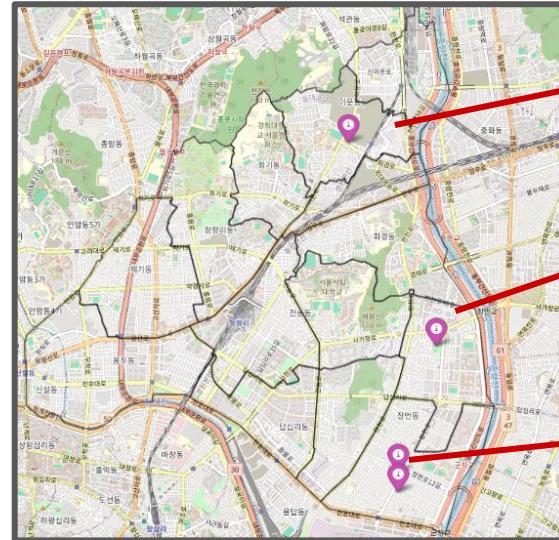
3. 최종 후보지

4. 후보지 결정

1

지도 시각화

Folium



이문동

재개발 지역

장안동

공원 지역

장안동

교회, 식당 번화가



3. 최종 후보지

(37.577, 127.071)



지리적 특징

- 장안근린공원
- 교회 인근
- 대로변과 인접
- 식당가
- 대로변 반대편에 주거지역 다수 존재
- 버스 정류장 인근

법정동

장안동

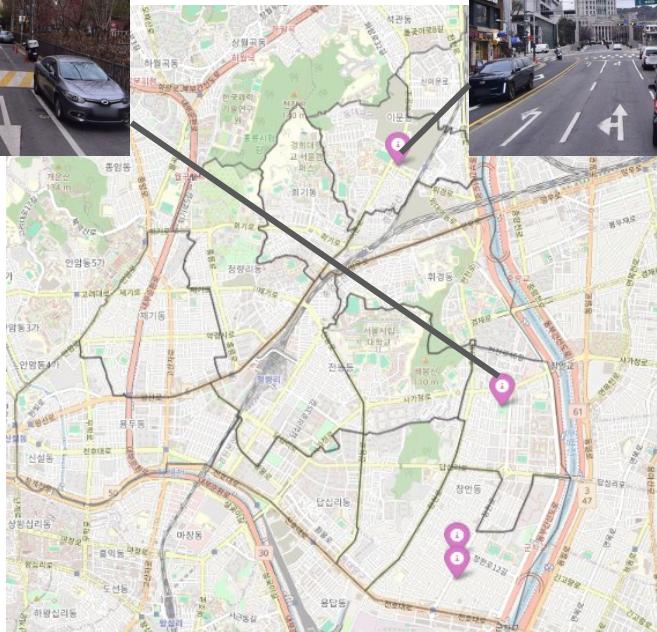
4. 앙상블

5

BOB
UP



(37.596, 127.061)



지리적 특징

- 외대 인근
- 발달된 지역
- 대로변 식당 상가
- 재개발 지역 인근

법정동

이문동



4. 앙상블

5

BOB
UP



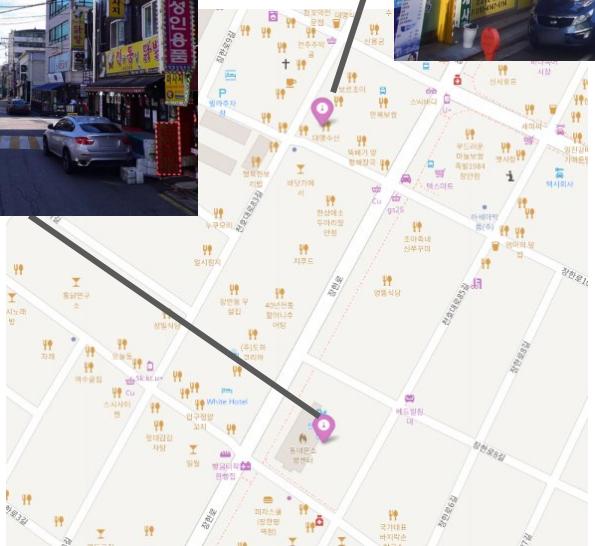
(37.563, 127.066)

지리적
특징

- 동대문소방센터
- 장한평 역 인근
- 식당가
- 어린이집, 초등학교, 중학교 도보 10분 이내

법정동

장안동



(37.565, 127.067)

지리적
특징

- 식당 상가
- 장한평 역 인근
- 식당가
- 어린이집, 초등학교, 중학교 도보 10분 이내

법정동

장안동



1. 설명

2. 시각화

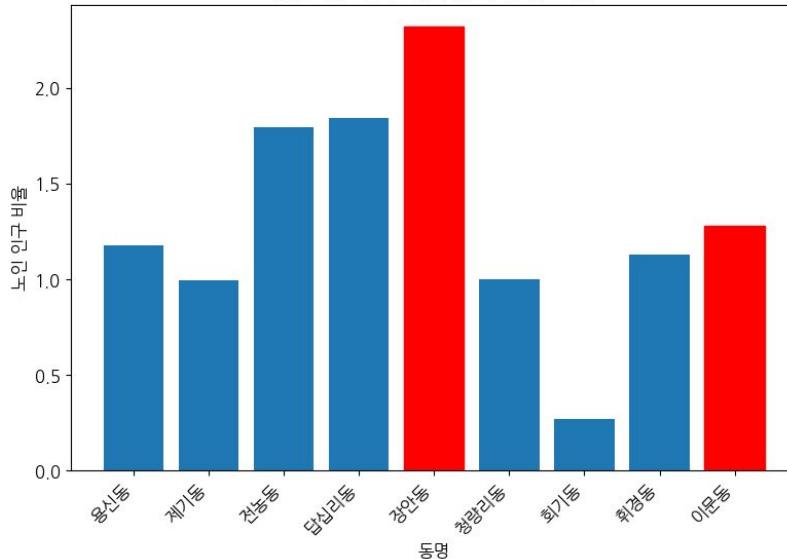
3. 최종 후보지

4. 후보지 결정

1

주민등록 노인 인구 비율

청량리동 대비 각 동의 노인 인구 비율



무료급식소 수요 : 장안동 > 이문동

산출 식 : 각 동 65세 이상 노인 인구 / 청량리동 65세 이상 노인 인구



4. 후보지 결정

최종 후보지

2

후보지 데이터 셋 비교

교육시설_best	교통시설_best	방범시설_best	복지시설_best	흡연시설_best	급식지원사업_best	positive	negative	lon_left	lat_left
True	True	True	True	True	False	225	1661	127.060969	37.596423
True	True	True	True	True	False	227	964	127.066821	37.563996
True	True	True	True	True	False	271	1397	127.066811	37.565799
True	True	True	True	True	False	256	1357	127.071273	37.577533

이문동 후보지 삭제
: negative 가장 많음, positive 가장 적음

