송파구의 민방위 대피소
대피경로 최적화

송파구의 민방위대피소의 현황 및 분석

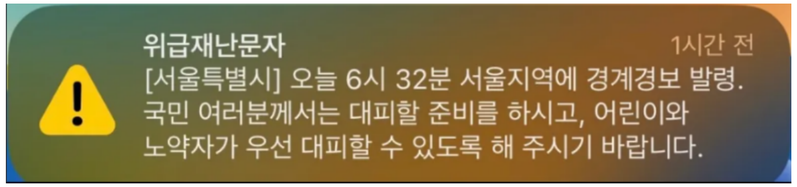
컴퓨터공학과 20170681 유재광

목차

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 오발령으로 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

등록일 2020-08-07

수정일 2021-10-06

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

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

	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)

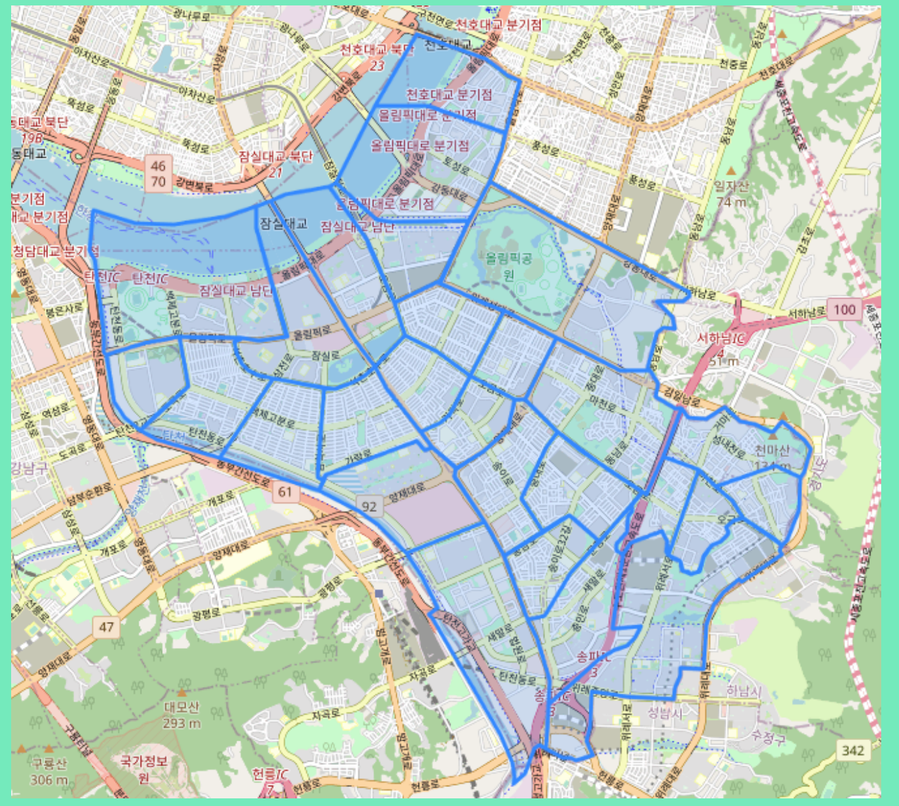
<국가공간정보포털>

```
# 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)

<국가공간정보포털>



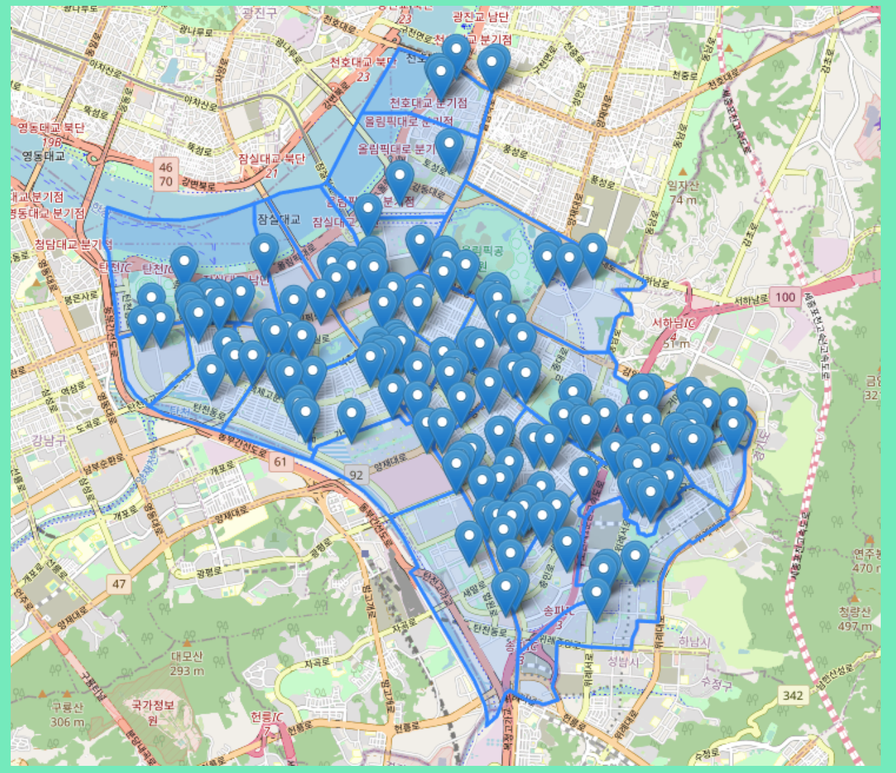
데이터 형태 및 전처리 방법

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

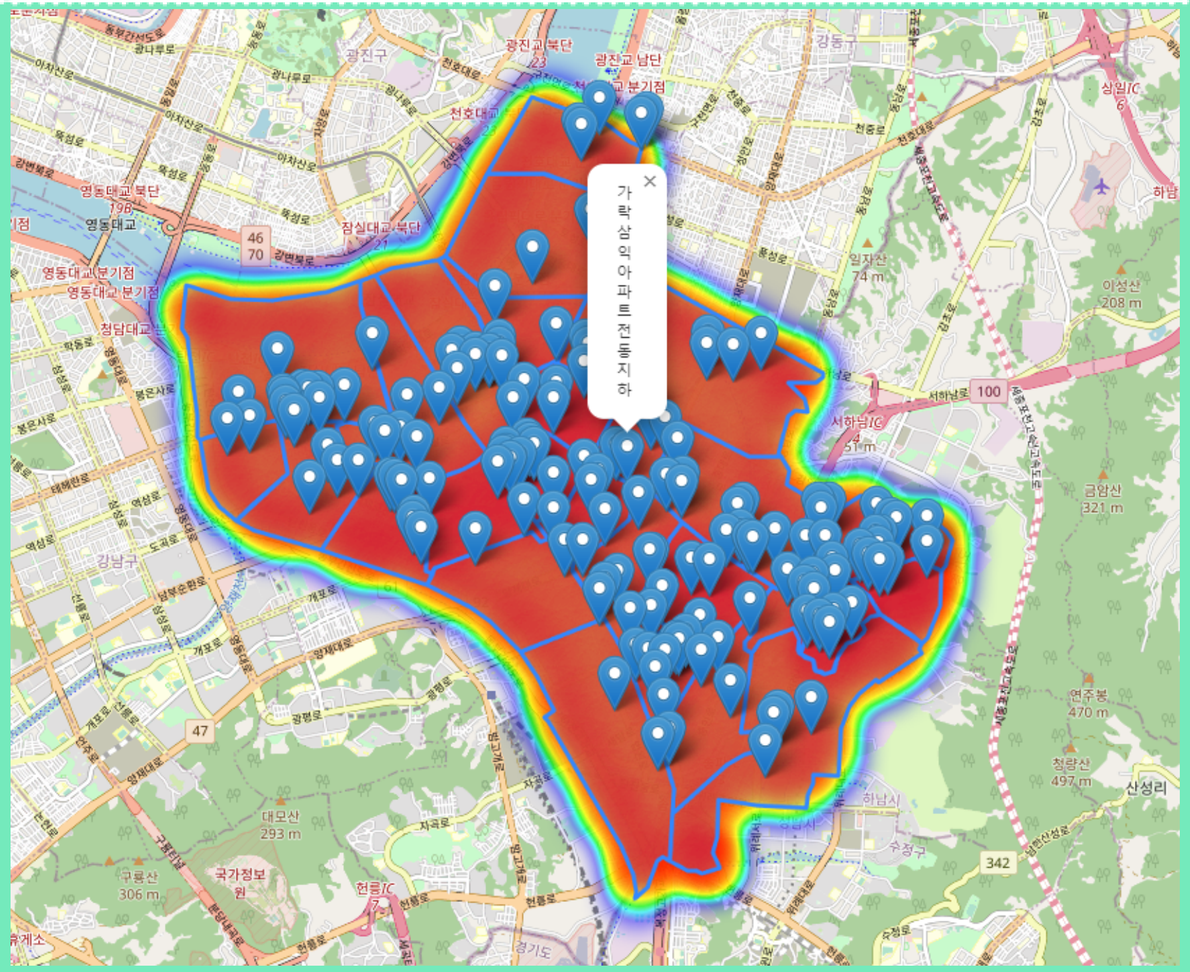
송파구 각 지역 인구 수 만큼 좌표 생성

```python
# 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 사용된 모델 및 방법론

- 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.

1. 데이터 준비

- 전처리 된 대피소 데이터(updated_facility.csv)와 인원 데이터(points.csv)를 불러옵니다.

2. 클러스터링을 위한 데이터 준비

- 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 분석에 사용된 모형 및 방법론

K-means clustering algorithm

을 사용한 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

되므로 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    서울시농수산식품공사 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

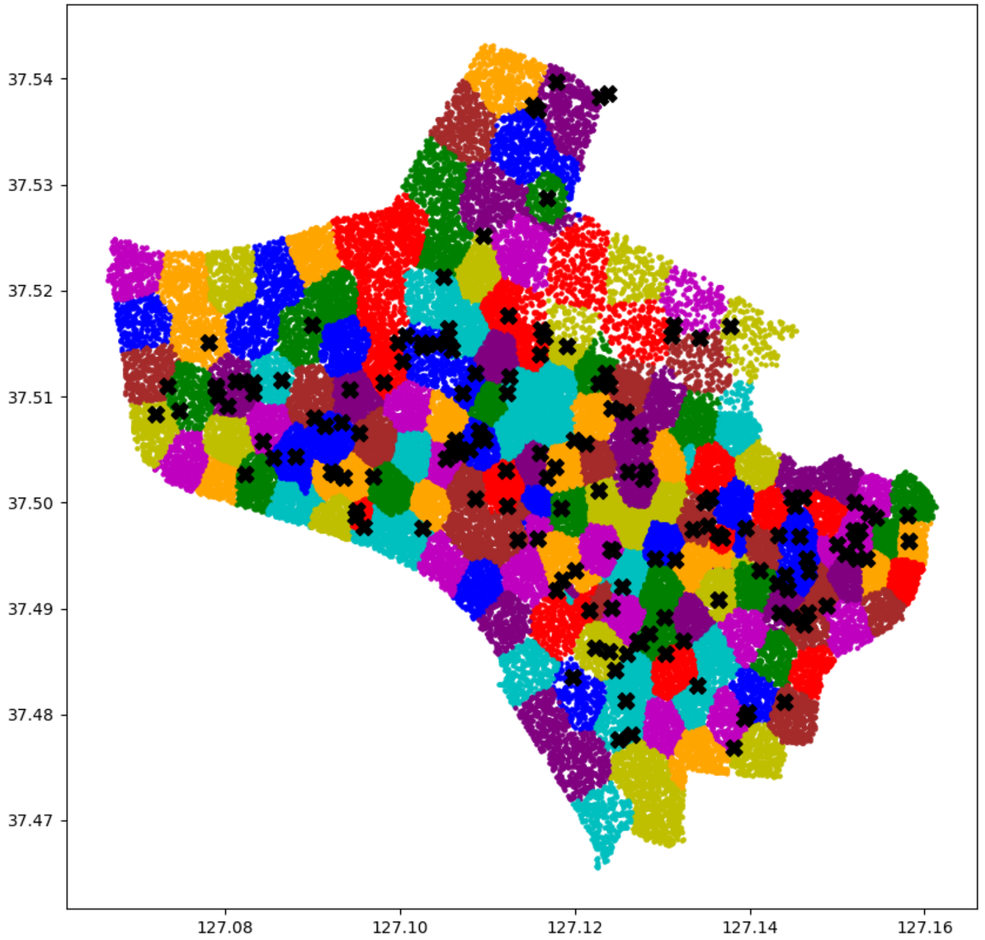
4. 인원 할당 및 재할당

```
# 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

5. 시각화

```python
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()
```



분석 결과물

대피 상황 때 우리집에서 가야할 대피소는?

좌표

위도 : 37.49687939935719,

경도 : 127.13548474247413


```
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

즉각적인 대피소 위치 제공

효과적인 대피 가능

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 것 으로 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

거리 계산 방식

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

유동인구

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

다른 지역의 대피소 반영

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

추가 대피소 위치 선정

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