Final Project Individual Report

Project Title : 카카오 API 기반 안전지수 측정 사이트

추상화

서울특별시에서 발생한 5대 범죄 및 음주 운전에 대한 분석을 통해 각 범죄 유형과 서울시 자치구의 범죄율을 머신러닝을 활용하여 조사했습니다. 이를 토대로 서울의 지역적 특성과 범죄 발생 간의 상관관계를 확인하고, 범죄 예방을 위해 시행된 제도 및 법령의 효과를 검증하고 예측하여 자체 안전 지표를 수립하는 것이 목표입니다.

한 달간의 작업 기간 동안 공공 데이터 포털, 서울 열린 데이터 광장, 스마트 치안 빅데이터 플랫폼, TAAS 교통사고 분석 시스템 등의 여러 사이트를 참고하여 데이터 수집을 진행했습니다. 각 범죄에 대한 상관관계 비교 분석을 통해 머신 러닝 및 장고 서비스 구현을 위한 데이터를 생성하고 전처리를 완료했습니다.

임의의 안전지수를 도출하기 위해 CCTV, 유흥시설, 편의점, 범죄율, 생활인구, 인구밀도 등 다양한 데이터들을 각 범죄와의 상관관계를 비교하였습니다. 양의 상관관계가 확인된 경우 안전지수를 낮추고, 음의 상관관계가 나타난 경우 안전지수를 높이는 계산식을 적용하여 서비스 구현을 목표로 하고 있습니다.

목차

1. 프로젝트 개요

지난해 발표된 경찰청 체감안전도 조사는 2022년 국민 체감안전도 조사 결과'에 따르면 작년 국민의 범죄 체감안전도는 전년보다 0.4점 내린 83점이었습니다. 이 중 여성 상대 범죄는 81.4점, 강도·살인은 84.6점으로 각각 0.3점, 0.6점 내리는 등 전 분야에서 체감안전도가 낮아졌습니다. 이어 지난해 8월을 기점으로 이상동기범죄를 시작으로 범행 예고·협박글이 줄을 이었고 전문가들은 시민들이 재난 수준의 불안을 느낄 것이라고 진단했습니다. 이러한 결과를 보고 우리 사회 치안에 대한 의문을 가지기 시작했고, 가장 인구 밀도가 높은 서울 특정지어 범죄 데이터를 분석하기로 결정했습니다.

2. 데이터 수집 및 전처리

```
crime_df = pd.read_csv("5대_범죄_발생현황_검거율.csv", encoding='cp949')
cctv_df = pd.read_csv("서울시 자치구 연도별 방범용 CCTV 운영 현황_230630기준.csv", encoding='cp949')
xy_df = pd.read_csv("서울시_자치구_중심점_2017.csv", encoding='cp949')
living_df = pd.read_csv("자치구단위 서울생활인구 일별 집계표.csv", encoding='cp949')
people_df = pd.read_csv("인구밀도_전처리.csv", encoding='cp949')
conv_df = pd.read_csv("편의점_구별_개수.csv", encoding='cp949')
drunk_df = pd.read_csv("서울시_유흥시설_개수.csv", encoding='cp949')
```

```
# 한글 출력의 글씨가 깨지지 않게 만들어 주는 폰트 설정
plt.rc("font", family = "Malgun Gothic")
sns.set(font="Malgun Gothic",
rc={"axes.unicode_minus":False}, style='white')
```

2-1. 전처리 과정

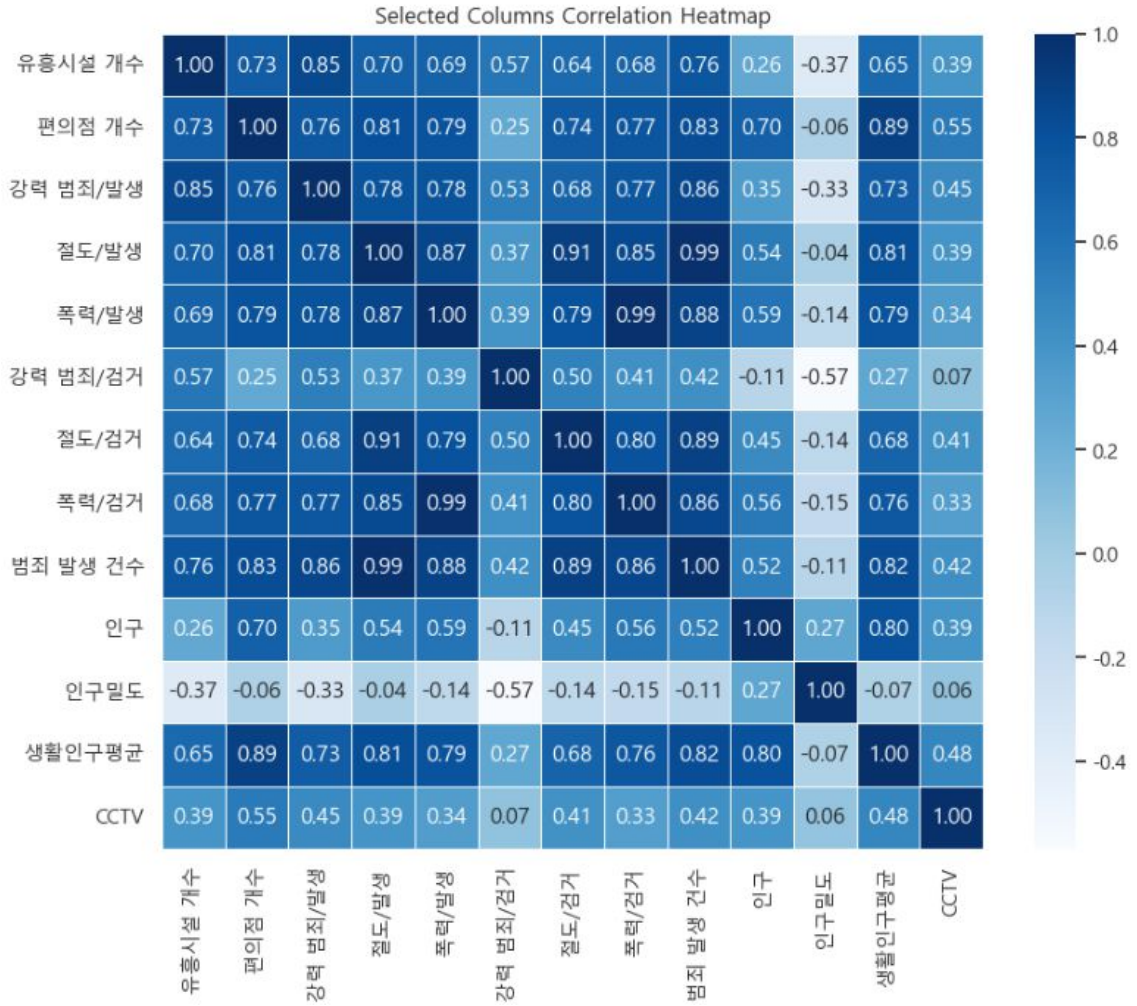
다음과 같은 CSV 파일을 여러개 준비하고, 하나의 데이터프레임으로 합치기 위한 작업을 시작한다. 범죄 데이터는 숫자가 굉장히 중요하기 때문에, 처음에 타입을 확인해서 숫자 형으로 잘 지정이 되어 있는지 우선 확인을 진행하였고, 데이터프레임이 모두 숫자 형으로 잘 지정되어 있어서 초기 전처리 및 데이터 병합을 위한 준비는 되어 있는 것 같다. 나중에 병합하게 되면, 컬럼 이름도 똑같게 맞춰야 하기 때문에, 이 부분은 우선 결측값 유무를 확인한 후, 결측값에 대해서는 범죄 건수가 0일 경우가 있기 때문에, 따로 결측값은 처리하지 않았다.

추후 지역 자치구별 비교를 위한 데이터를 좌표 데이터를 준비하고, 병합할 컬럼의 이름을 동일 하게 맞춰주는 작업을 진행하고, 데이터프레임을 합친다. 범죄 검거 및 범죄 발생과의 CCTV에 상관관계 비교를 위하여, CCTV 운영 현황을 추가시켰다. 생활 인구에 따른 범죄 지표 확인을 위해서 생활 인구 데이터,인구밀도 데이터를 추가하고, 기존 데이터프레임과 기준을 맞추기 위해서 일별 집계 데이터를 월별 평균으로 맞춰주고 필요 없는 컬럼은 최대한 삭제하고 기존 데이터프레임과 합친다. 일별 데이터프레임을 연도별 평균값으로 맞춰서, 전처리를 진행하고, 기존의 데이터프레임과 병합해서, 연도별로 생활 인구 평균을 만들어서 분석하던 데이터프레임과 병합을 진행한다.

모든 범죄 발생 건수를, .sum() 함수를 이용해서 더해주고 범죄 발생 건수 컬럼을 만든다.
그 후 생활 인구로 나누어서 생활 인구에 따른 범죄율 컬럼을 새로 생성한다. 살인과 강도 강간·추행 은 강력 범죄로 분류해서 세 가지를 합쳐서 따로 분류할 생각이다. 서로 건수가 미미하기도 하고, 강력 범죄로 분류해서 따로 발생률을 체크해서 그거에 따른 발생률을 체크해 보도록 하자. 마지막으로 편의점과 유흥시설 개수를 추가시켜서 각 시설간의 상관관계 비교를 준비하도록 하자.

3. 분석 내용

상관관계를 비교하기 이전에 필요한 데이터만 뽑아서 상관관계 비교를 진행할 것이다. 우리가 상관관계를 비교하기 위한 데이터들은 이정도 데이터만 넣어서 비교를 해보도록 하자.

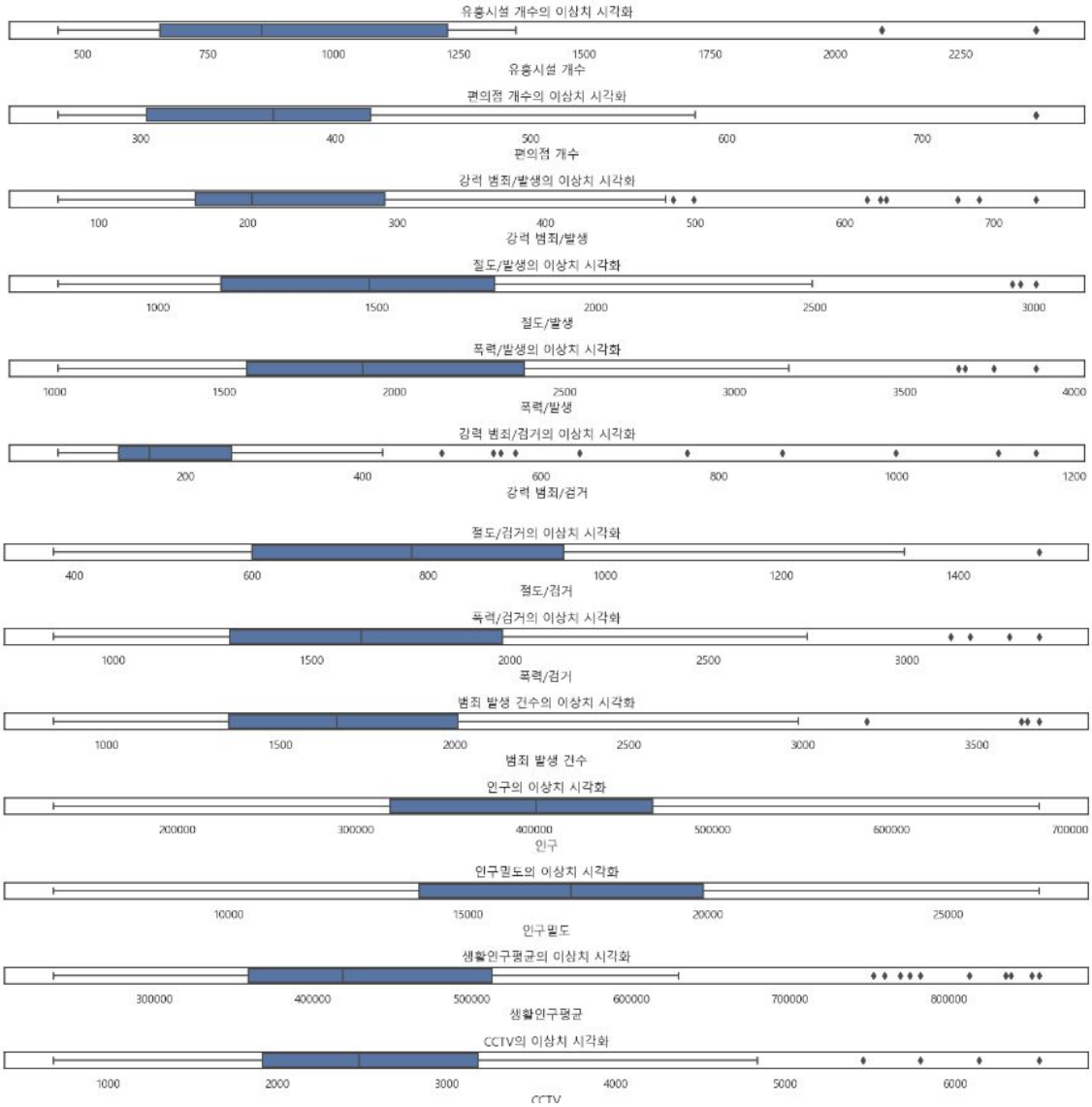


유흥시설/편의점 : 유흥시설이 많다면 범죄/발생이 늘어나게 된다. 그리고 전체적인 범죄 발생 건수도 증가한다. 마찬가지로 편의점 개수도 범죄/발생이 늘어나게 된다.

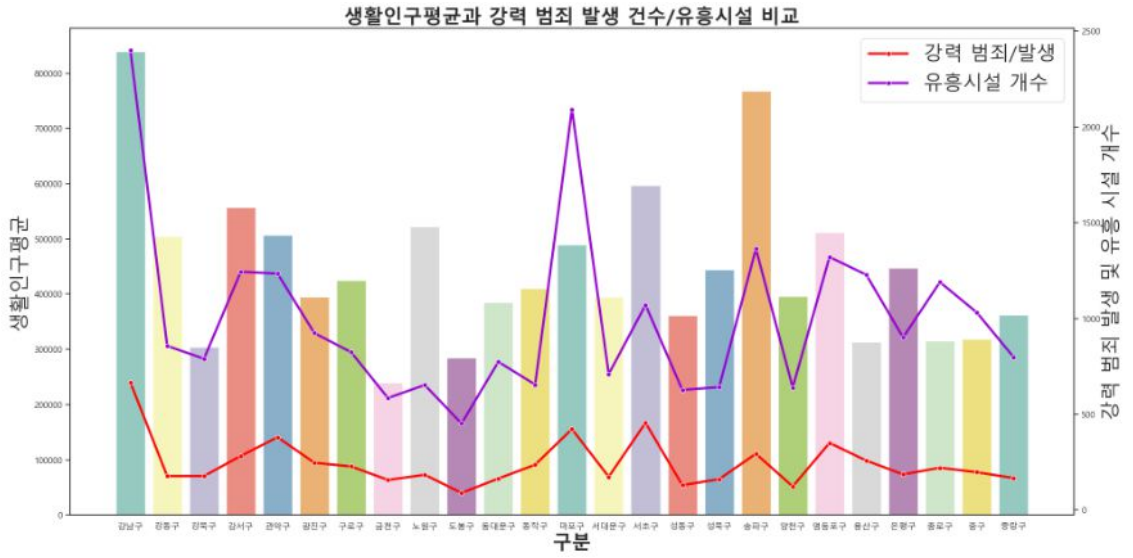
CCTV : CCTV가 많다고 해서 범죄/검거가 늘어나는 것이 아니다. CCTV와 검거는 관련이 없다 볼 수 있다. 그렇지만, 치안에 효과는 있을 것이니, 검거와의 상관관계가 없는게 아쉽다.

인구/인구밀도 : 인구가 늘어날수록 전체적인 범죄 건수는 늘어난다. 인구밀도가 늘어나면, 강력 범죄/발생의 건수가 소폭 줄어드는 것을 확인할 수 있다.

생활 인구 평균 : 범죄/발생 건수가 확실하게 늘어나게 된다.

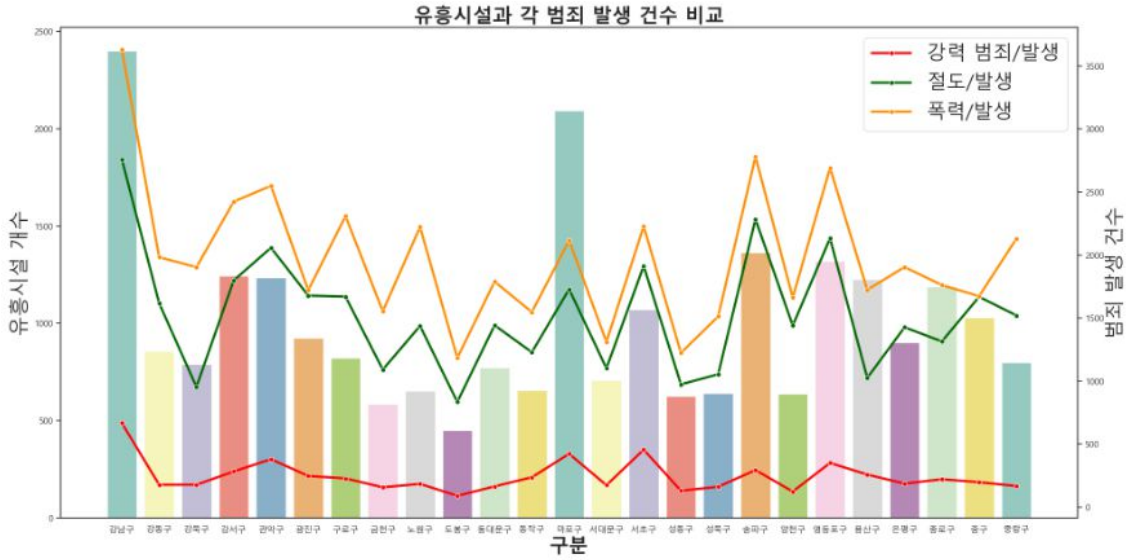


사실상 지역구마다 시설이나, 범죄 건수의 차이가 심하게 날 수도 있기에 발생하는 이상치 라고 생각할 수 있다. 이상치에 대해서 치환은 진행하지 않고, 그대로 진행하도록 하는 편이 좋을 것 이라 생각한다.



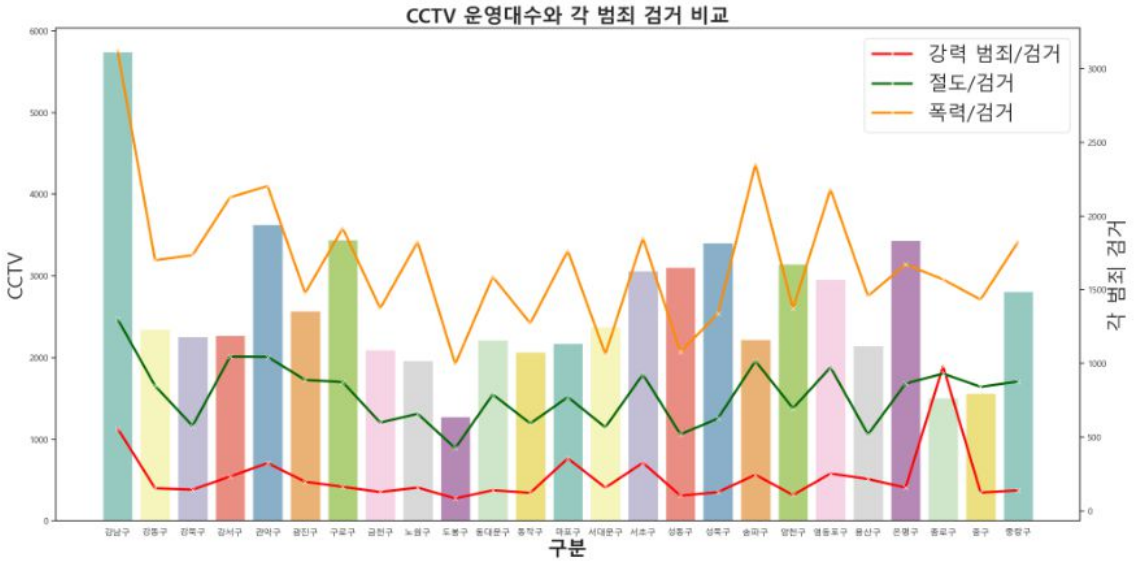
생활 인구 평균과 강력 범죄 발생 건수/유흥시설 비교

생활 인구 평균이 늘어남에 따라서 강력 범죄/발생 건수가 어느정도 미미한 관계는 있어보이나, 큰 상관관계 라기 보단 어느정도 관계가 있다고 볼 수 있을 것 같다. 하지만 유흥시설이 늘어남에 따라서 강력 범죄는 확실하게 증가하는 폭을 보이고 있다.



유흥시설과 각 범죄 발생 건수 비교

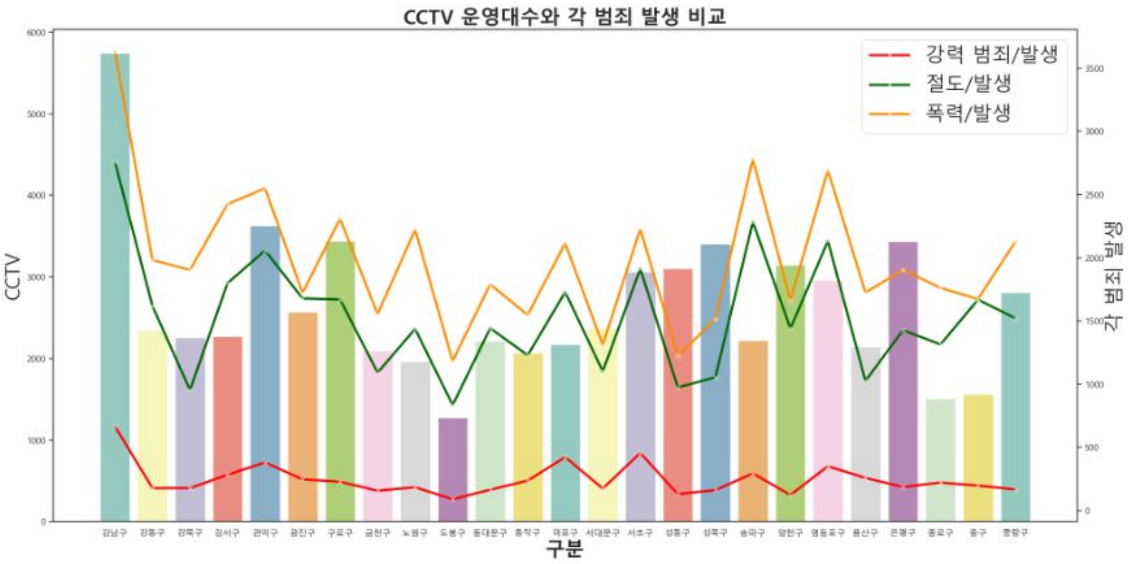
범죄 발생 건수가 증가함에 따라서 주변의 유흥시설을 비교해 봤을 때, 강력 범죄 이외에도 유흥시설은 범죄 발생에 꽤 큰 영향을 끼치고 있다고 볼 수 있다. 유흥시설과 범죄의 발생은 큰 상관관계를 보이고 있다.



CCTV와 각각의 범죄 검거 비교

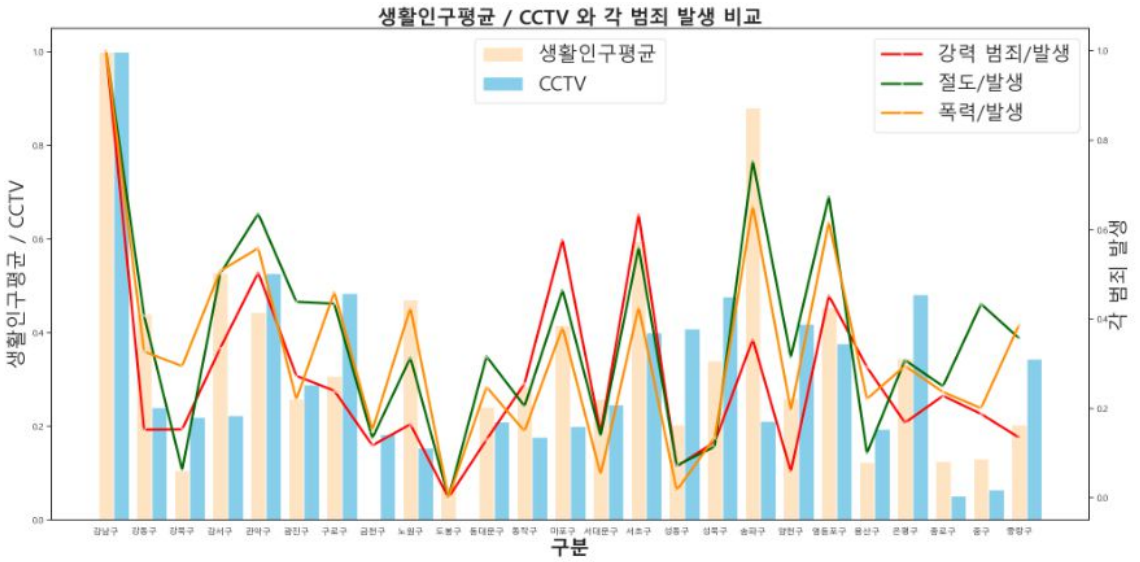
강남구의 경우 범죄가 많이 발생하는 곳인데, 그만큼 검거가 높은 곳을 보면 강남의 CCTV는 역할을 잘하고 있다 본다. 다른 자치구 들의 경우 CCTV에 비해서 검거가 굉장히 낮은편인 곳도 존재하기 때문에, 추가적인 CCTV가 필요할 수도 있다.

다만 종로구의 경우 어떤 이유에서인지 CCTV 운영 대수에 비해서 검거가 굉장히 높은 편에 속하는데, 현재 분석할 수 없는 다른 모종의 이유가 있을 거라 추측된다. 경찰의 인력이 많다거나, 종로가 서울의 중앙에 있다보니, 교통의 요충지라서 그럴 수도 있다.



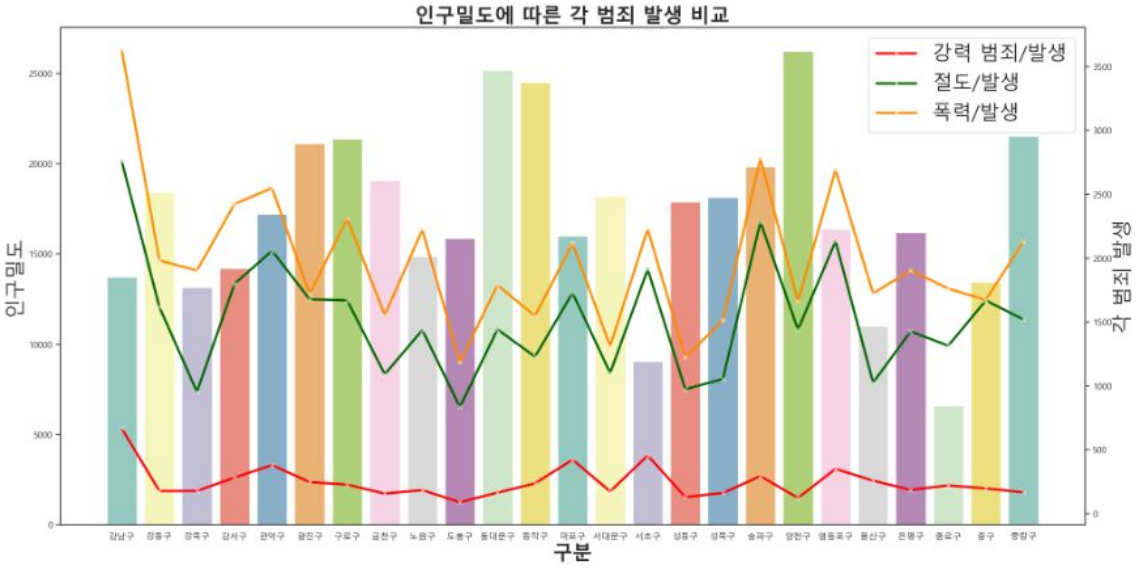
CCTV 운영 대수와 각 범죄 발생 비교

CCTV는 검거에는 어느정도 효과가 있었으나, 범죄 예방에는 효과적이지 못했다. CCTV가 많은 지역임에도 범죄 발생이 많이 일어나거나, CCTV가 너무 적다고 범죄의 발생이 기하급수적으로 늘어나는 것도 아니다.



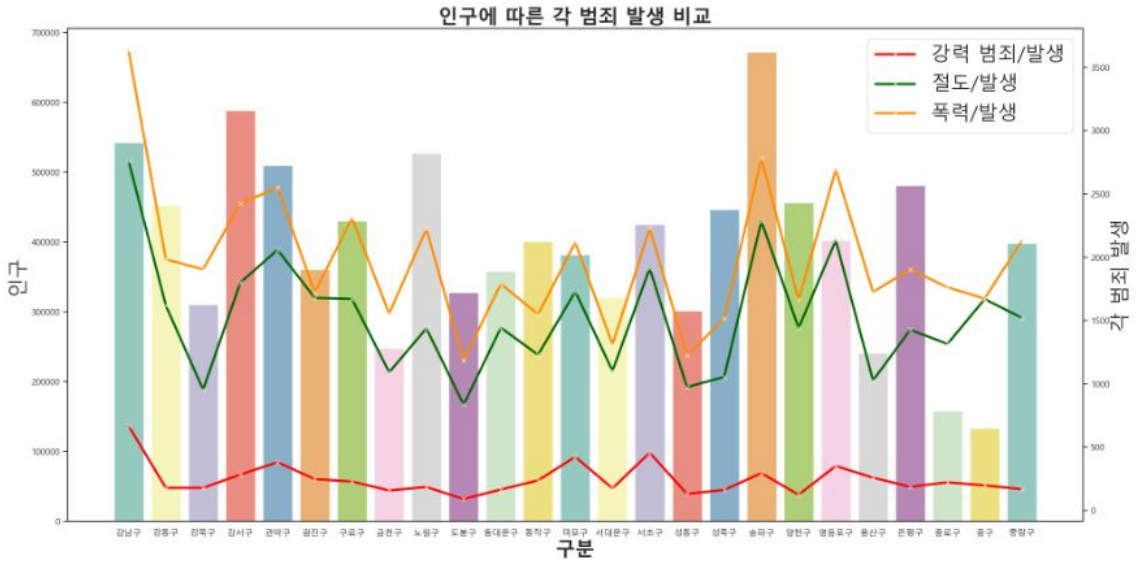
생활 인구 평균 및 CCTV와 각 범죄 발생 비교

생활 인구 평균이 늘어나는 만큼 CCTV의 대수도 그에 맞춰서 늘어나게 된다. 하지만, 생활 인구 평균에 비해서 CCTV가 적은 지역도 분명히 존재한다. 그 지역의 경우에는 범죄율이 다소 늘어나는 것을 확인하게 된다. 하지만, 생활 인구 평균에 비해서 CCTV 대수도 적고, 범죄율이 높은 지역도 분명하게 보이고 있다.



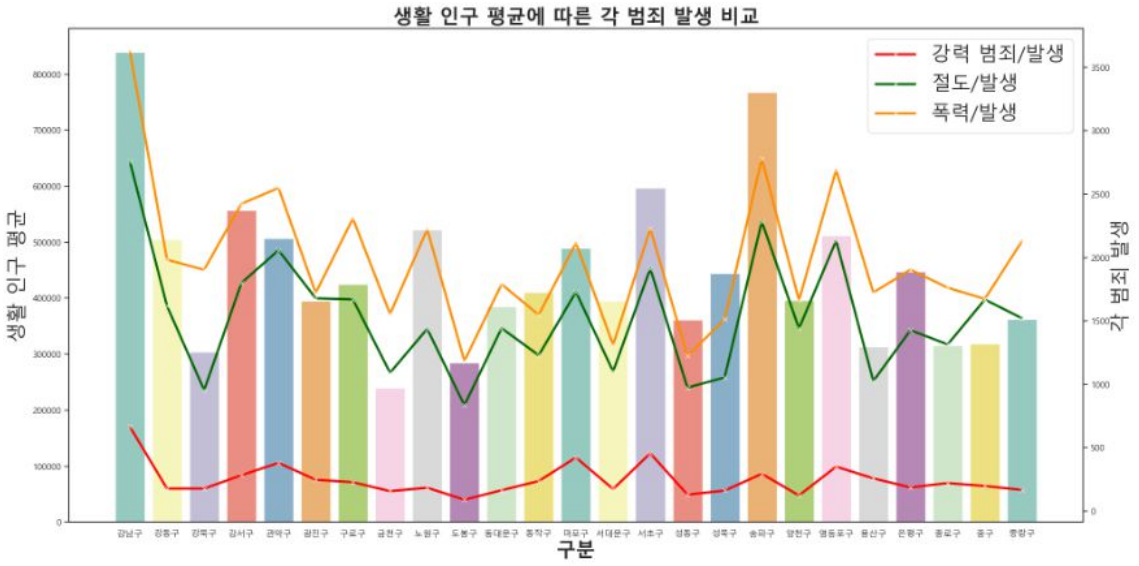
인구밀도에 따른 각 범죄 발생 비교

인구밀도가 적든 크든 각 범죄 발생과는 관계가 없다고 봐야 한다. 그래프상으로 보이는 수치에서도 인구밀도가 늘어남에 따라서 범죄가 늘어나는 것도 아니고, 오히려 인구밀도가 적은 곳에서 범죄가 많이 일어나기도 한다. 인구밀도와 범죄와의 상관관계는 없다.



인구에 따른 각 범죄 발생 비교

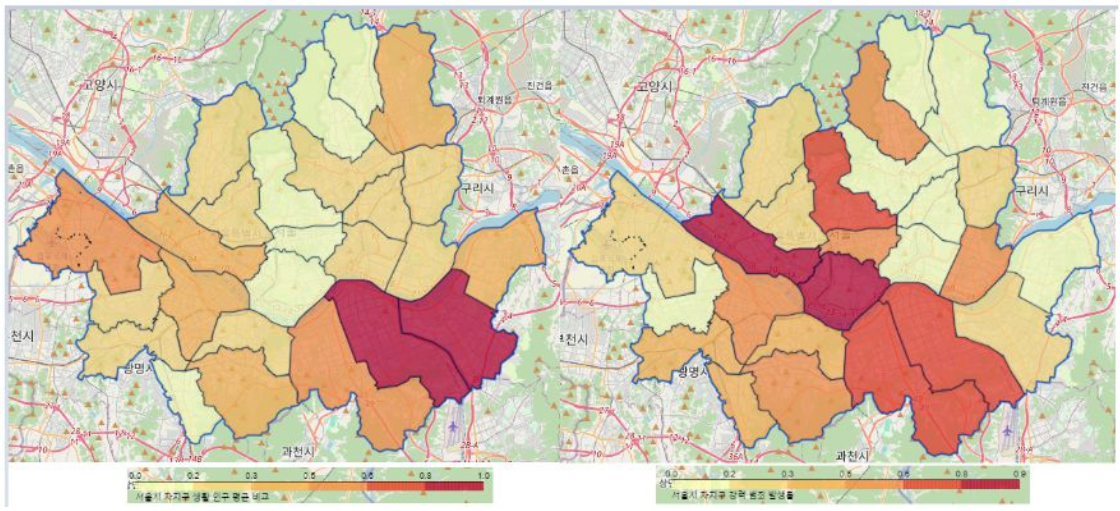
인구와 각 범죄 발생 비교를 했을 때, 인구가 높은 지역의 범죄 발생은 늘어나고 있음이 보여진다. 다만, 종로구와 중구의 경우 인구가 굉장히 적음에도 불구하고 범죄가 많이 발생 하고 있다. 송파구와 노원구의 경우 인구에 비해서 범죄가 적은 편에 속하는데, 이외 지역에는 어느 정도 상관관계가 있다.



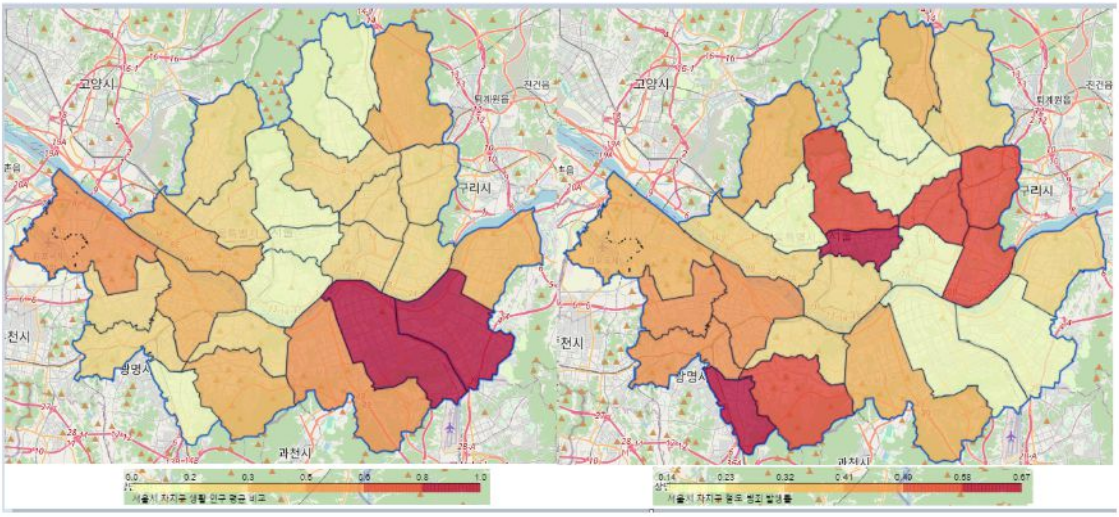
생활 인구 평균에 따른 각 범죄 발생 비교

생활 인구 평균을 가지고 각 범죄 발생 건수를 비교한 결과, 생활 인구 평균이 늘어나게 되면, 범죄 건수는 그만큼 늘어나고 있고, 어느정도 상관관계를 보이고 있다는 것을 알 수 있다.

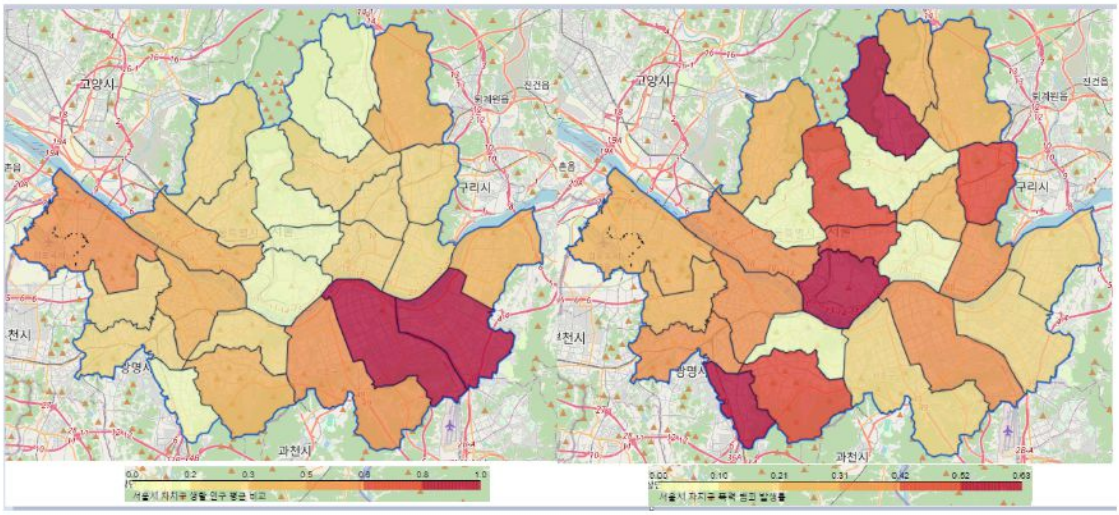
폴리움 활용 간단 시각화
서울의 지역구별로 경계선을 나누고, 그 경계선을 기준으로 서울의 지역구별로 생활 인구 평균과 각각의 범죄 발생률을 비교해 보는 지도를 그리도록 해보았다.



자치구별 생활 인구 평균에 따른 강력 범죄 발생률 비교
어느 정도 생활 인구 평균에 따라서 강력 범죄에 대한 발생률은 비슷한 수치가 나오고 있다. 다만 특정 지역의 경우 강력 범죄 발생률이 생활 인구 평균에 비해서 높은 수치를 나타내는 지역도 존재한다. 보통 생활 인구 평균이 적을 경우 강력 범죄 발생률은 낮아지고 있다.



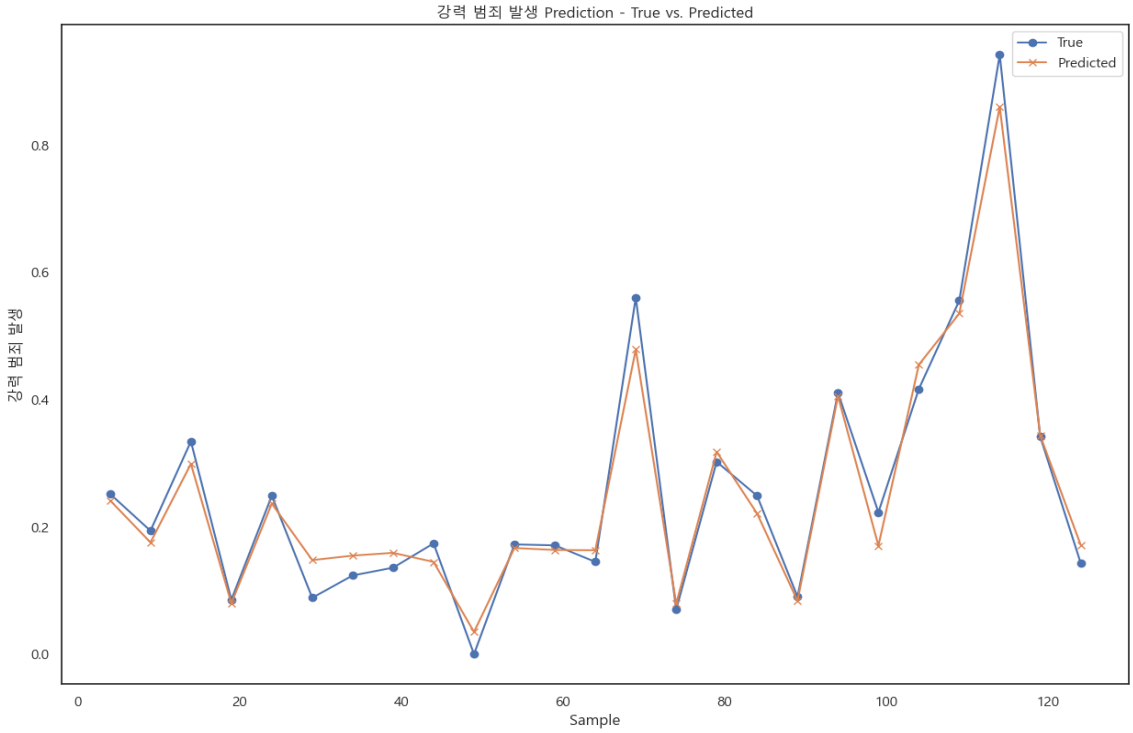
자치구별 생활 인구 평균에 따른 절도 범죄 발생률 비교
강남구 쪽의 경우에는 절도 범죄률이 강력범죄 비율에 비해서 현저하게 적다. 절도 범죄는 생활 인구 평균 과는 큰 상관관계는 없고, 특정 지역의 경우 다발적으로 발생하고 있는 것으로 보인다.

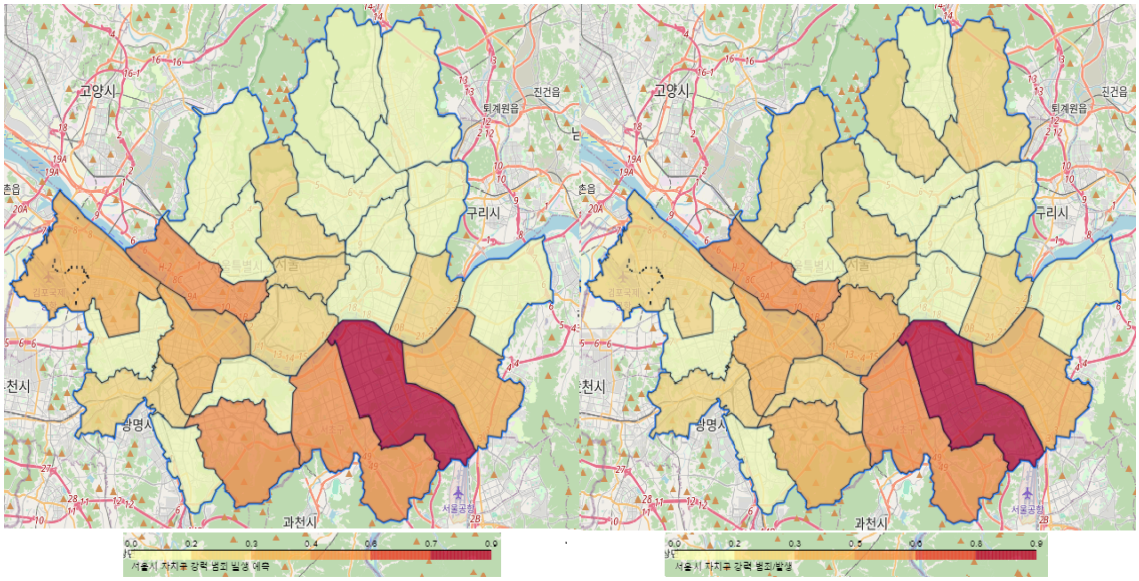


자치구별 생활 인구 평균에 따른 폭력 범죄 발생률 비교
폭력 범죄와 절도 범죄의 그래프는 비슷하게 나오는 것으로 보아, 폭력과 절도 범죄가 다수 발생하는 지역은 강력 범죄 발생률이 낮은 지역은 절도와 폭력 범죄의 비율이 전체적으로 높게 나오고 있다.

4. 머신러닝 예측 모델

각 범죄 발생 예측 모델 생성
우선 데이터 스케일링 작업을 통해서 스케일링을 진행한 후에 각 범죄의 발생률 예측 모델을 만들어 보자. 우선적으로 강력 범죄 발생에 대한 예측 모델을 생성해 볼건데, 랜덤포레스트와 그리드 서치를 활용해서 예측 모델을 만들어보자.

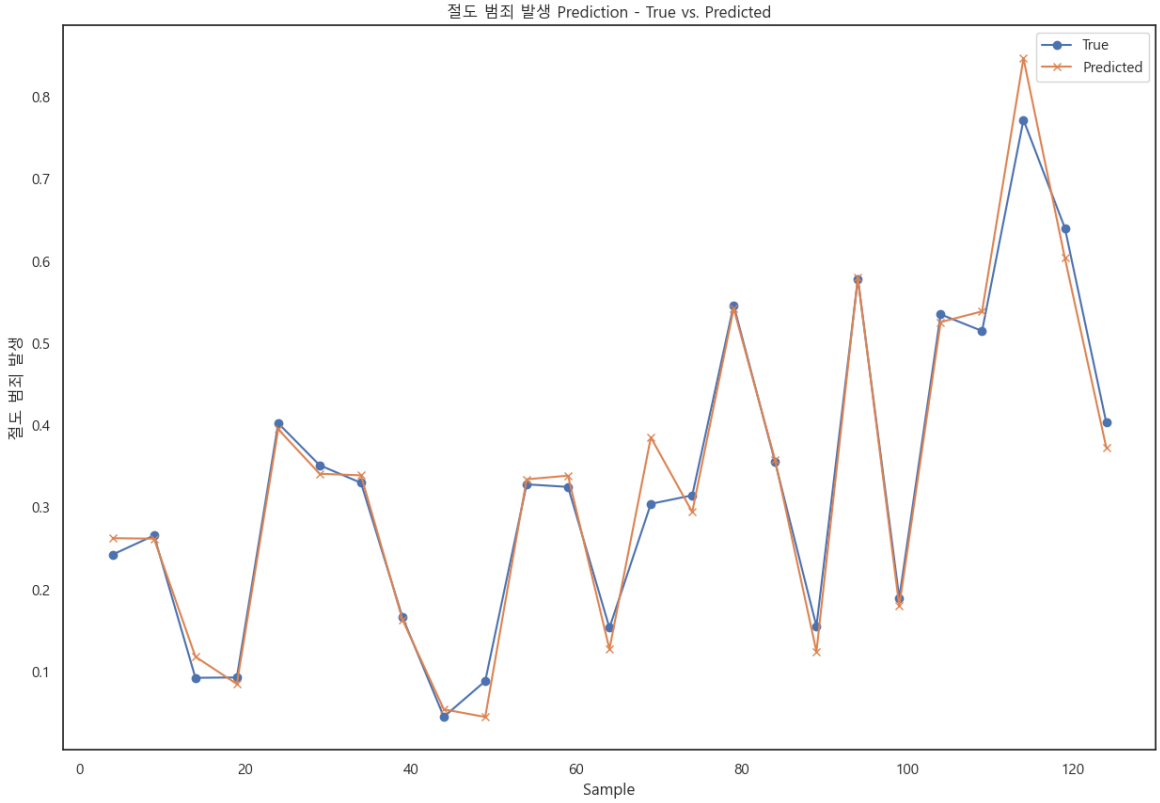


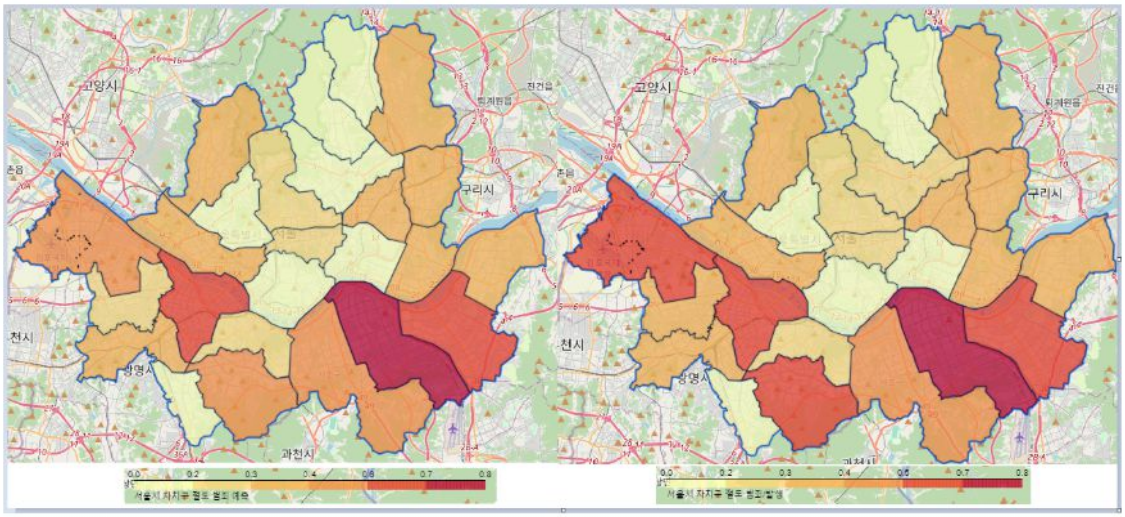


강력 범죄 발생 예측 데이터와 실제 데이터 비교

왼쪽이 예측한 데이터이고, 오른쪽은 실제 데이터 이다. 직관적인 비교를 진행하였을 때, 서로 비슷한 수치의 그래프를 보이고 있다. 예측 모델이 적합하게 진행되었다.

절도 범죄 발생 예측

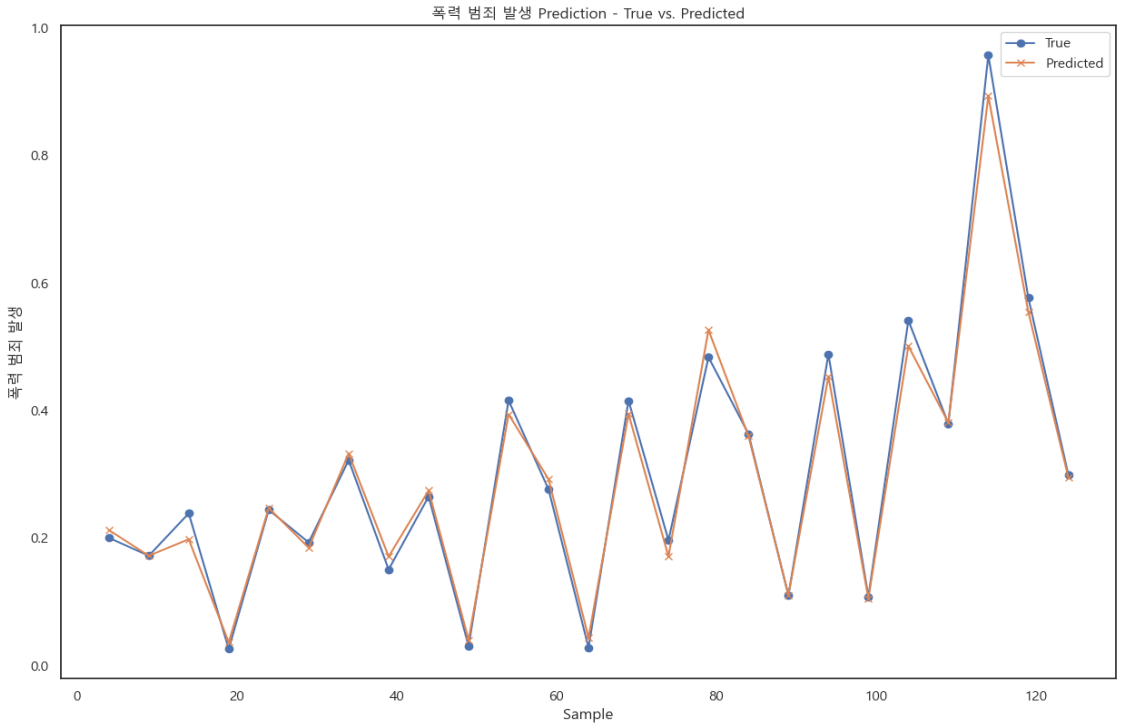


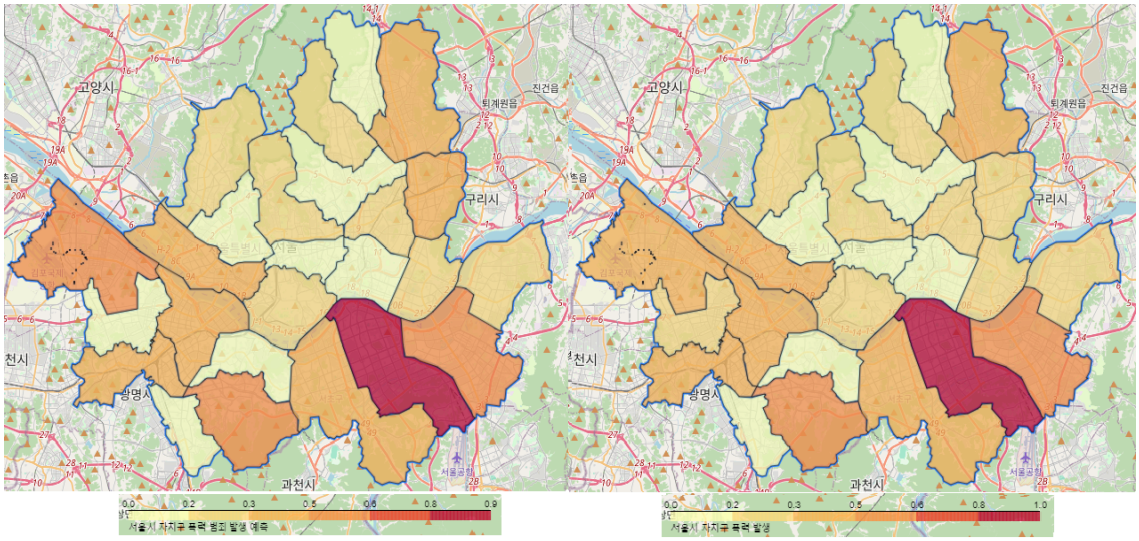


절도 범죄 발생 예측 데이터와 실제 데이터 비교

왼쪽이 예측한 데이터이고, 오른쪽은 실제 데이터 이다. 직관적인 비교를 진행하였을 때, 서로 비슷한 수치의 그래프를 보이고 있다. 예측 모델이 적합하게 진행되었다.

폭력 범죄 발생 예측



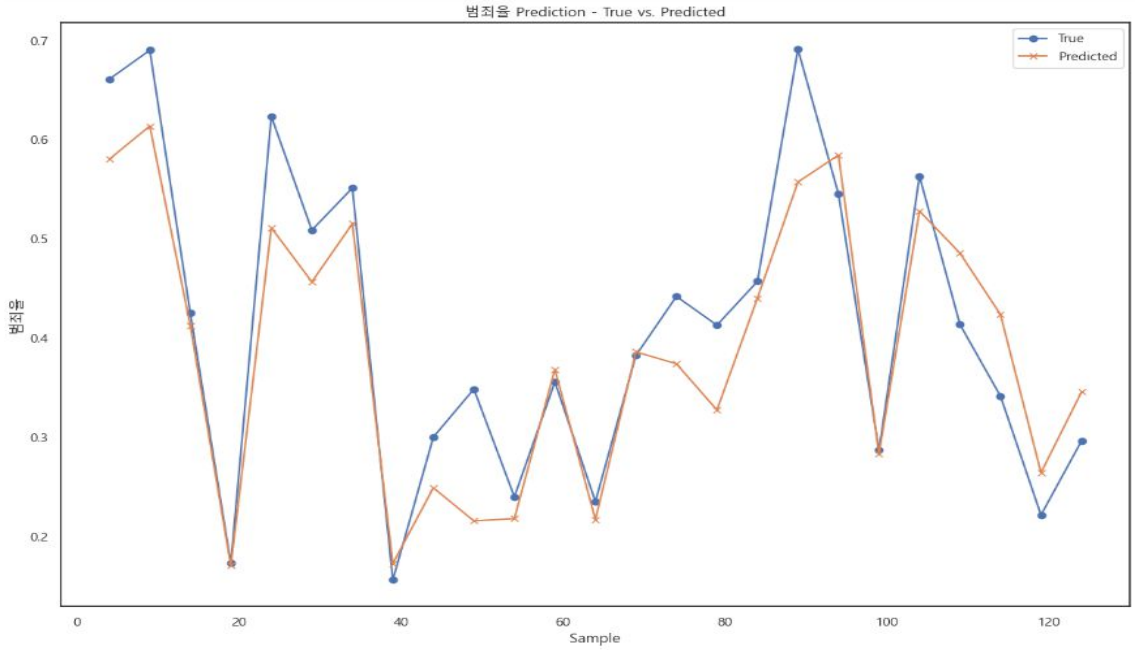


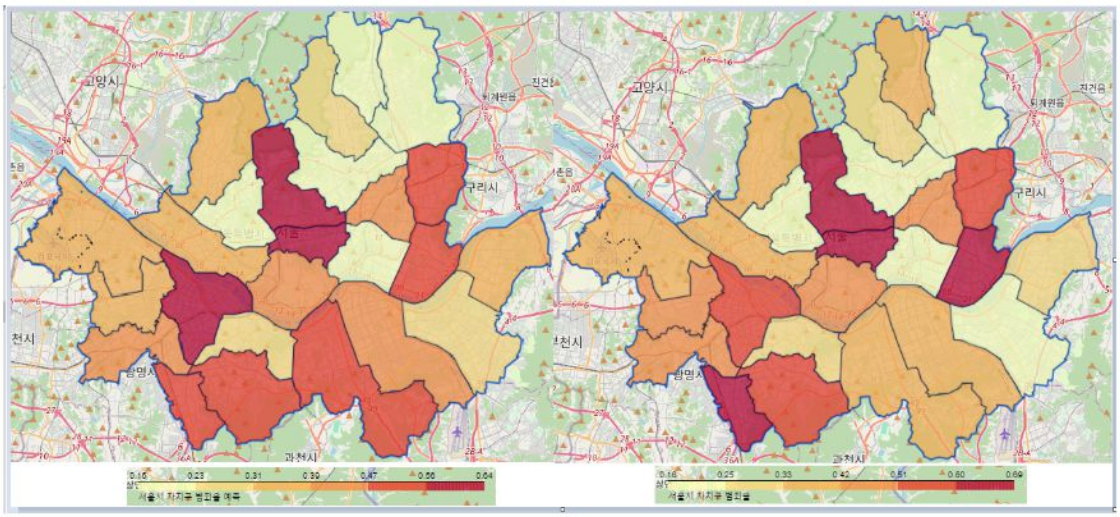
폭력 범죄 발생 예측 데이터와 실제 데이터 비교

왼쪽이 예측한 데이터이고, 오른쪽은 실제 데이터이다. 직관적인 비교를 진행하였을 때, 서로 비슷한 수치의 그래프를 보이고 있다. 예측 모델이 적합하게 진행되었다.

범죄율 예측하기

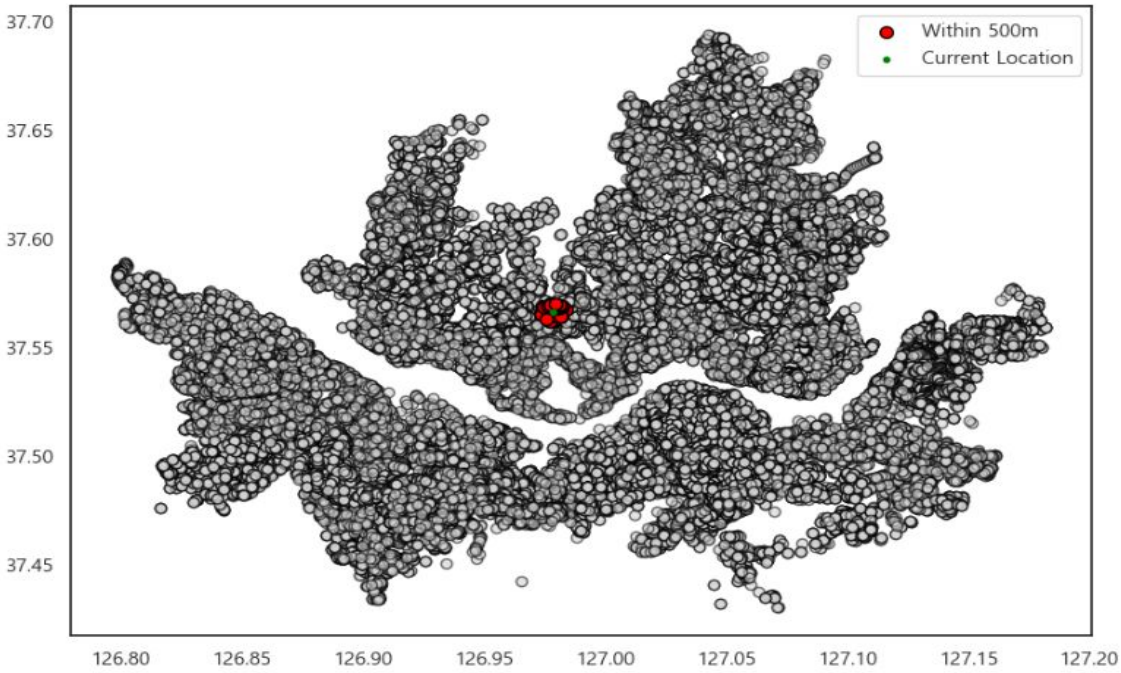
모든 범죄를 통합하고 생활 인구 평균에 따른 범죄와의 비교 임의 데이터를 만들어 '범죄율' 이라는 컬럼을 생성했었다. 이것에 대한 예측을 진행하자.





범죄율 예측 데이터와 실제 데이터 비교
왼쪽이 예측한 데이터이고, 오른쪽은 실제 데이터이다. 직관적인 비교를 진행하였을 때, 서로 비슷한 수치의 그래프를 보이고 있다. 예측 모델이 어느정도는 비슷하게 진행되었다.

5. 서비스 배포를 위한 장고



좌표 찾기 성공
CCTV의 좌표를 구해서 반경 m 내에 존재하는 CCTV 좌표만을 찾아줄 수 있는, 장고 서비스화를 위한 준비를 한다. 나의 현재위치를 기반으로 500m 내에 CCTV 좌표만을 찾아주는 코드를 만들었을 때, 성공적으로 붉은색으로 표시가 되는 것을 보니, 장고 서비스화 구현에 충분히 쓸 수 있다는 사실을 알게 되었다. 그렇다면 이제 장고 서비스화 구현을 시작해 보자.

장고 서비스 구현

장고 서비스 구현에 필요한 데이터는 CCTV, 경찰서, 편의점, 유흥시설, 지킴이 집, 음주 운전 다발 구역 등의 서울시의 존재하는 시설의 좌표가 필요하다. 그 자료들의 정리를 시작해 보자. 편의상 Jupyter에서 작업하지 않고, 먼저 CSV에서 정리를 하였다.

```python
# 각좌표간의 거리 반경 계산기 함수 (직선 거리 공식)
def calculate_distance(lat1, lon1, lat2, lon2):
    # 두 지점 간의 직선 거리를 계산하는 함수
    # 두 지점 간의 x, y 좌표 차이를 사용하여 계산
    dx = lon2 - lon1
    dy = lat2 - lat1
    distance = sqrt(dx**2 + dy**2) * 111000  # 경도와 위도의 차이를 고려하여 거리를 계산 (단위: 미터)
    return distance
```

좌표간의 거리계산 함수

이제 얻어낸 좌표 데이터를 가지고 서비스 웹페이지에서 해당 좌표 마킹을 할 수 있도록 코드를 만들어서 동작하게 만들어 주자. 현재 서비스 구현의 최종 목표는 우리가 원하는 좌표 입력을 받고, 그 좌표와 현재 위의 시설 좌표 간의 거리 계산을 해서 반경 m 내에 존재 하는 시설의 좌표만 마킹을 해주는 작업을 하게 될 것이다. 이 코드는 그 반경에 대한 거리를 계산해주는 함수이다.

```python
# CSV을 읽어오고, 입력한 좌표의 반경 내에 있는 좌표만 전달하는 함수
def read_csv(file_path, reference_lat, reference_lon, radius):
    coordinates = []
    with open(file_path, 'r', encoding='cp949') as file:
        reader = csv.reader(file)
        next(reader)  # Skip header if exists
        for row in reader:
            district = row[0]  # Assuming the first column is district name
            latitude = float(row[1])  # Assuming the second column is latitude
            longitude = float(row[2])  # Assuming the third column is longitude

            # 참조 지점에서의 거리 계산
            distance = calculate_distance(reference_lat, reference_lon, latitude, longitude)

            # 거리가 지정된 반경 내에 있는지 확인
            if distance <= radius:
                coordinates.append({'district': district, 'latitude': latitude, 'longitude': longitude})
    return coordinates
```

좌표가 저장된 CSV 파일 불러오기

시설 좌표가 저장된 CSV 파일을 읽어 들여서, 위에서 계산된 반경 내에 존재하는 좌표만을 남겨서, 딕셔너리 형태로 저장하고, 반환을 해주는 함수이다.

```python
# 좌표값 구분 후에 해당 좌표를 딕셔너리로 저장하는 함수
def mark_district_coordinates(api_key, coordinates):
    markers = []
    for coordinate in coordinates:
        district = coordinate['district']
        latitude = coordinate['latitude']
        longitude = coordinate['longitude']

        # Call Kakao Maps API to get more information if needed
        # Example: https://dapi.kakao.com/v2/local/geo/coord2regioncode.json?x={longitude}&y={latitude}
        marker = {
            'title': district,
            'latlng': {'lat': latitude, 'lng': longitude},
        }
        markers.append(marker)
    return markers
```

위도와 경도를 구분해서 정리하는 함수

이제 저장이 완료된 딕셔너리 형태의 데이터를 다시 한번, 구분하기 쉬운 이름으로 바꿔주고 'latlng'이라는 Key 값을 생성해서, 하나의 'title'에 위도와 경도가 구분되서 들어가게끔 바꿔주는 함수를 만든다.

```python
# 가장 가까운 지역구를 찾아주는 함수
def distance_gu(reference_lat, reference_lng):
    # 좌표값
    latitude = reference_lat
    longitude = reference_lng

    # GeoJSON 파일에서 지역구 경계 읽기 (파일 경로는 실제 경로로 변경 필요)
    geojson_file_path = 'hangjeongdong_서울특별시.geojson'
    gdf = gpd.read_file(geojson_file_path)

    # 좌표를 Shapely Point로 변환
    point = Point(longitude, latitude)

    # 지역구 경계 중 가장 가까운 것 찾기
    min_distance = float('inf')
    nearest_district = None

    for index, row in gdf.iterrows():
        district_shape = shape(row['geometry'])
        distance = point.distance(district_shape)

        if distance < min_distance:
            min_distance = distance
            nearest_district = row['sggnm']

    return nearest_district
```

가장 가까운 지역구 찾아주는 함수

우리가 받은 좌표를 기준으로 서울시 지역구 경계 geojson 파일을 이용하여, 그 좌표와 가장 가까운 지역구를 찾아주고, 그 지역구를 반환해 주는 함수이다.

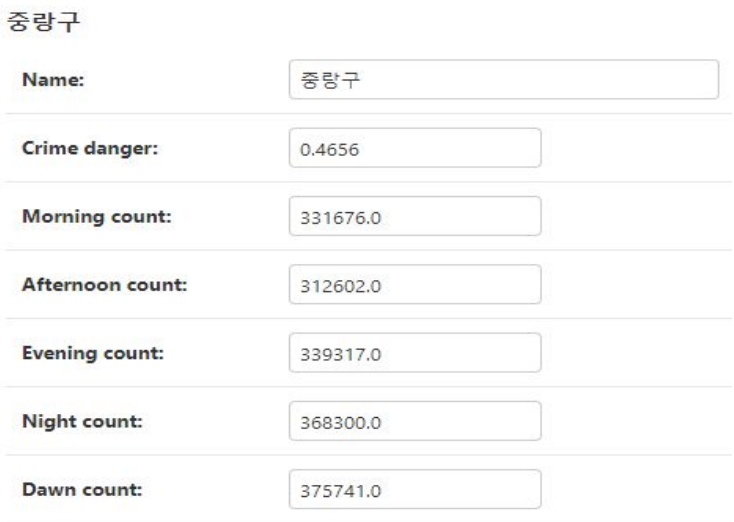
```python
# Y 좌표 할당해주는 함수
def addr_lat(addr):
    url = 'https://dapi.kakao.com/v2/local/search/address.json?query={address}'.format(address=addr)
    headers = {"Authorization": "KakaoAK " + "943e784f9ac8621fd887cd598cea9f18"}
    result = json.loads(str(requests.get(url, headers=headers).text))
    match_first = result['documents'][0]
    return float(match_first['y'])

# X 좌표 할당해주는 함수
def addr_lng(addr):
    url = 'https://dapi.kakao.com/v2/local/search/address.json?query={address}'.format(address=addr)
    headers = {"Authorization": "KakaoAK " + "943e784f9ac8621fd887cd598cea9f18"}
    result = json.loads(str(requests.get(url, headers=headers).text))
    match_first = result['documents'][0]
    return float(match_first['x'])
```

검색어의 좌표값을 반환해주는 함수
웹페이지에서 먼저 검색어를 받아올 것인데, 그 검색어를 카카오 API를 통해서 해당 되는 검색어에 매칭이 되는 좌표를 가지고 와서 반환해 주는 함수이다.

```python
# 여러 계산을 진행 후에 페이지를 만드는 함수
def map_detail(request):
    search_keyword = request.GET.get('search_keyword', '')
    reference_lng = addr_lng(search_keyword)
    reference_lat = addr_lat(search_keyword)
    gu_name = distance_gu(reference_lat, reference_lng)
```

이제 가장 중요한 함수가 있다. 현재 위의 모든 함수 들을 일괄적으로 돌리면서, 만든 값을 HTML로 보내주어서, 웹페이지에 서비스 구현을 할 수 있게 만들어 주는 함수이다. 우선 검색어를 먼저 받고, 그 검색어에 해당되는 좌표를 가져온다. 그 후, 그 얻어낸 좌표를 통해서, 그 좌표와 가장 가까운 지역구를 찾고 반환받는다. DB에 사전에 등록해 놓은 범죄율과 시간대별 유동 인구 데이터가 존재하는데



```
location_instance = Location.objects.get(name=gu_name)
crime_danger = location_instance.crime_danger

# 거리 단위를 가져오기
distance_unit = request.GET.get('distance', '')
# 거리 단위에 따라 radius 값 설정 (기본값은 500m)
radius = int(distance_unit) if distance_unit.isdigit() else 500

csv_file_path = '서울시 통합 좌표.csv'
api_key = '5b2f66c85f3743a2bc9b7c204f357893'
```

찾아낸 가장 가까운 지역구와 DB를 매칭시켜서 해당 DB에 모든 오브젝트를 불러오게 된다. 그리고 검색창에서 주소 이외에도 검색할 반경을 HTML로부터 데이터를 받게 되고, 그 반경 값을 변수 이름에 저장해 두어 반경 거리 함수에 변수 이름을 보내주어, 반경을 계산하게 된다. CSV 파일은 시설의 좌표가 저장된 파일을 보내줘 계산하게 될 것이고, API KEY는 카카오 API 승인을 받기 위한 API KEY 값이다.

```
search_time = request.GET.get('search_time')
# 시간대에 따른 점수 계산
if search_time == 'morning':
    score = 0.3
    count = location_instance.morning_count
elif search_time == 'afternoon':
    score = 0.25
    count = location_instance.afternoon_count
elif search_time == 'evening':
    score = 0.2
    count = location_instance.evening_count
elif search_time == 'night':
    score = 0.1
    count = location_instance.night_count
elif search_time == 'dawn':
    score = 0.3
    count = location_instance.dawn_count
else:
    score = 0.3
    count = location_instance.morning_count
```

시간 가중치값 계산

거리 계산 외에도, 시간 값도 따로 받을 것인데, 우리의 서비스 구현 목표는 안전도 수치를 나타내는 것이 목표이기 때문에, 시간대에 따라서 가중치를 다르게 주어서 안전도를 계산할 수 있도록 한다. 그리고 시간대별 비교 후에, DB에 알맞은 데이터를 찾아서 반환해준다.

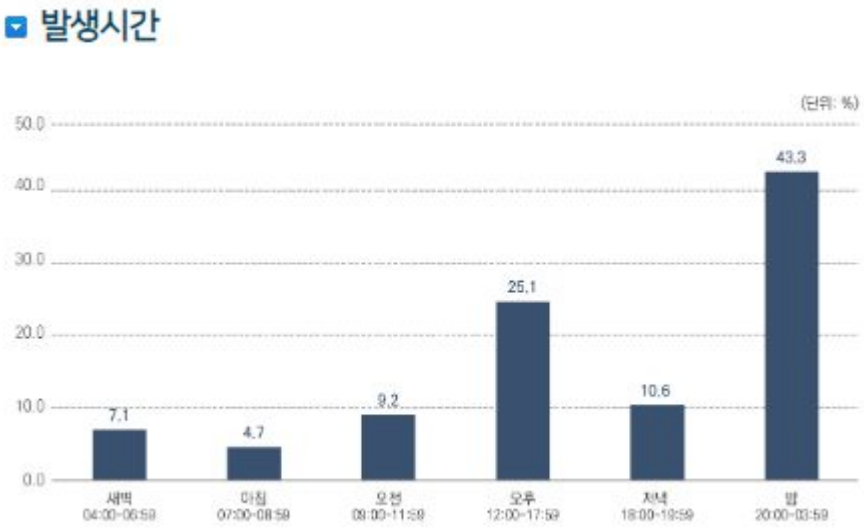


그림 39 성폭력범죄의 범죄발생시간

시간 가중치값이 들어간 이유

위의 예시는 성폭력 범죄의 발생 시간에 따른 통계표지만, 모든 범죄의 발생 통계가 밤이 가장 높았고, 다음은 오후이며 오전/아침은 오전으로 통일하고, 밤과 오후 외에는 가중치 값을 똑같게 주어서, 밤의 경우 안전도가 낮음을 나타낼 수 있는 가중치를 준다.

```
filtered_coordinates = read_csv(csv_file_path, reference_lng, reference_lat, radius)
markers = mark_district_coordinates(api_key, filtered_coordinates)

context = {'markers': markers, 'x' : reference_lng, 'y' : reference_lat, 'search_keyword': search_keyword, 'score': score,
           'search_time': search_time, 'gu_name':gu_name, 'crime_danger':crime_danger, 'count':count, 'radius':radius}
return render(request, 'projects/map_detail.html', context)
```

HTML로 데이터 보내주기

이제 CSV 파일을 보내서 좌표를 계산하고, 딕셔너리로 받아낸 데이터를 마지막 HTML 페이지로 보내주도록 한다.

```
<td>
    <select name="search_time" style ="width:100%;height:100%;text-align: center;">
        <option value="morning">오전</option>
        <option value="afternoon">오후</option>
        <option value="evening">저녁</option>
        <option value="night">밤</option>
        <option value="dawn">새벽</option>
        <!-- 다른 시간대 옵션들을 추가할 수 있습니다. -->
    </select>
</td>
<td>
    <select name="distance" style ="width:100%;height:100%;text-align: center;">
        <option value="300">300m</option>
        <option value="500">500m</option>
        <option value="700">700m</option>
        <option value="1000">1000m</option>
        <!-- 다른 거리 옵션들을 추가할 수 있습니다. -->
    </select>
</td>
```

시간대 및 반경 선택

이제 검색받을 검색어와 시간대, 거리를 기본 'map.html'에서 받아서 'map_detail'로 보내줄 수 있도록 세팅을 해놓는다. 그리고 가장 중요한 'script'가 있다.

```
<div class = "map"  id="map" style="width:100%;height:800px;">
    <script>
        function currentLocation() {
            // HTML5의 geolocation으로 사용할 수 있는지 확인합니다
            if (navigator.geolocation) {
                // GeoLocation을 이용해서 접속 위치를 얻어옵니다
                navigator.geolocation.getCurrentPosition(function(position) {

                    var lat = position.coords.latitude, // 위도
                        lon = position.coords.longitude; // 경도

                    var container = document.getElementById('map'); //지도를 담을 영역의 DOM 레퍼런스
                    var options = { //지도를 생성할 때 필요한 기본 옵션
                        center: new kakao.maps.LatLng(lat, lon), //지도의 중심좌표.
                        level: 3 //지도의 레벨(확대, 축소 정도)
                    };

                    var map = new kakao.maps.Map(container, options); //지도 생성 및 객체 리턴
                    // 현재 위치로 지도의 중심을 이동
                });
            } else { // HTML5의 GeoLocation을 사용할 수 없을때 마커 표시 위치와 인포윈도우 내용을 설정합니다

                var container = document.getElementById('map'); //지도를 담을 영역의 DOM 레퍼런스
                var options = { //지도를 생성할 때 필요한 기본 옵션
                    center: new kakao.maps.LatLng(33.450701, 126.570667), //지도의 중심좌표.
                    level: 3 //지도의 레벨(확대, 축소 정도)
                };
            }
            return true
        }
        currentLocation();
    </script>
</div>
```

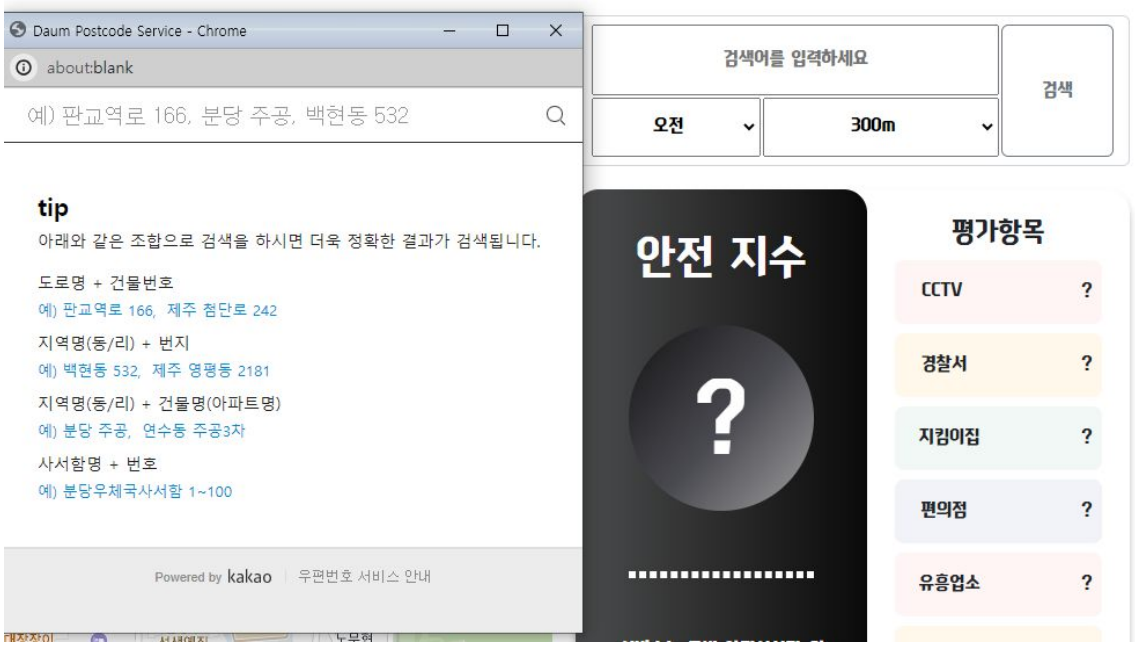
현재위치 기반 서비스

우선 기본적인 'map.html'을 열어서 카카오 API에서 기본적으로 제공하는 'geolocation'으로 실시간 현재위치로 지도의 좌표를 지정해 주도록 한다. 다만, 학원의 문제인지 종로3가가 아닌 안국역보다 살짝 위쪽으로 잡히는 문제가 있는데, 이 문제는 해결하지 못했다.

```
<script src="//t1.daumcdn.net/mapjsapi/bundle/postcode/prod/postcode.v2.js"></script>
<script>
    window.onload = function(){
        document.getElementById("address").addEventListener("click", function(){
            //카카오 지도 발생
            new daum.Postcode({
                oncomplete: function(data) {
                    document.getElementById("address").value = data.address;
                    var addressDetailInput = document.querySelector("input[name=address_detail]");
                    if (addressDetailInput) {
                        addressDetailInput.focus();
                    }
                }
            }).open();
        });
    }
</script>
```

카카오 API 도로명 주소 받기

동작시키는 함수는 카카오 API에서 제공하는 주소지를 입력해야만 그에 해당하는 좌표를 받아올 수 있게 설계가 되어, 마우스를 클릭하게 되면, 카카오 API를 연결해서 주소지를 선택하고 선택한 주소지의 좌표를 받아올 수 있도록 카카오에서 제공하는 기능을 적극 활용한다.



기본 map.html 페이지

기본 'map.html'은 이렇게 구성이 되어 있다. 'keyword' 값을 받고 싶다면, 검색어를 입력하는 창에 마우스 클릭하게 되면 새 창이 열려서 주소지를 선택할 수 있는 창이 나온다. 주소지를 클릭해서 누르면 검색어에 자동으로 해당하는 주소지가 입력되도록 하였다. 그리고 원하는 시간대와 반경값을 받은 후에 'map_detail.html'로 넘겨줄 수 있도록 세팅이 되어 있다.

```
var positions = [
    // Iterate through markers and add positions
    {% for marker in markers %}
    {
        title: '{{ marker.title }}',
        latlng: new kakao.maps.LatLng({{ marker.latlng.lng }}, {{ marker.latlng.lat }})
    },
    {% endfor %}
];
```

마킹을 하기 위한 좌표값 선택

'map_detail.html' 쪽을 살펴보게 되면 함수로 넘겨준 주소지의 좌표를 지도의 중심 좌표로 설정하게 되고, 마킹이 필요한 좌표에 해당 되는 딕셔너리 데이터를 for문을 통해서 html에서 값을 저장하게 된다.

```
for (var i = 0; i < positions.length; i++) {
    // 마커 이미지의 이미지 크기 입니다
    var imageSize = new kakao.maps.Size(50, 50);
    if (positions[i].title == 'CCTV'){
        // 마커 이미지를 생성합니다
        cctv_count += 1;
        var markerImage = new kakao.maps.MarkerImage(imageSrc_cctv, imageSize);}
    else if (positions[i].title == '유흥시설'){
        // 마커 이미지를 생성합니다
        drunk_count += 1;
        var markerImage = new kakao.maps.MarkerImage(imageSrc_drunk, imageSize);}
    else if (positions[i].title == '경찰서'){
        // 마커 이미지를 생성합니다
        police_count += 1;
        var markerImage = new kakao.maps.MarkerImage(imageSrc_police, imageSize);}
    else if (positions[i].title == '음주 운전 다발구역'){
        // 마커 이미지를 생성합니다
        crash_count += 1;
        var markerImage = new kakao.maps.MarkerImage(imageSrc_crash, imageSize);}
    else if (positions[i].title == '편의점'){
        // 마커 이미지를 생성합니다
        conv_count += 1;
        var markerImage = new kakao.maps.MarkerImage(imageSrc_conv, imageSize);}
    else if (positions[i].title == '지킴이집'){
        // 마커 이미지를 생성합니다
        safe_count += 1;
        var markerImage = new kakao.maps.MarkerImage(imageSrc_safe, imageSize);}
```

마킹을 하기 위한 시설 구분

'title'의 이름을 비교한 후에, 시설마다 구분 지어서 카운팅을 진행하고, 그에 해당하는 이미지를 불러와서 지도에 마킹할 수 있도록 해준다.

```
<script>
    var  totalScore = cctv_count + drunk_count + police_count + crash_count + conv_count + safe_count
    var averageScore = ((cctv_count / totalScore * 3 / 10) -
                        (drunk_count / totalScore * 2 / 10) +
                        (police_count / totalScore * 5) -
                        (crash_count / totalScore * 5) +
                        (conv_count / totalScore * 2) +
                        (safe_count / totalScore * 2) -
                        ({{ crime_danger }} / 10) -
                        ({{ count }} / 10000000));
    var safety = parseFloat((averageScore + {{ score }}).toFixed(4)) * 100;
</script>
```

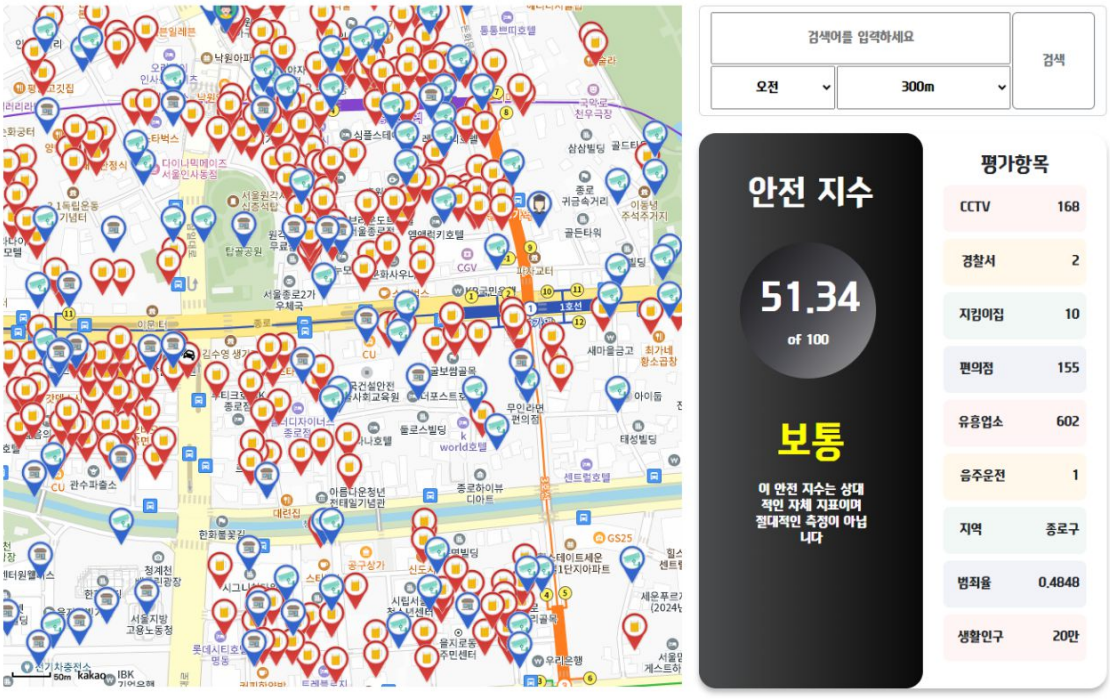
안전 수치 계산 방식

안전 수치 계산을 하기 위해서 안전 수치에 위협이 되는 시설들은 그 값만큼 - 가중치를 부여하고 안전 수치를 늘려줄 수 있는 값에는 + 가중치를 부여해서, 안전 수치를 나타낼 수 있도록 한다. CCTV와 유흥업소의 경우에는 개수가 워낙에 많아서 10개당의 가중치를 부여해서 큰 차이가 나지 않도록 값을 설정하도록 한다. 그리고 각각의 개수를 모든 개수를 더한 만큼 나눠주는 코드를 만들어서, 반경이 넓어져도 안전 수치의 큰 차이가 없도록 만들어 주도록 한다. DB에서 사전에 만들었던 범죄율과 시간대별 유동 인구는 좌표의 해당하는 지역구의 데이터를 가지고 안전 수치에 반영할 수 있도록 만들어 주고, 만들어진 값을 가지고 마지막에 파싱을 통해서 시간대에 대한 가중치 값을 더해준 이후에 백분위를 만들기 위해 100을 곱해서 최종적인 안전 수치를 만들 수 있도록 한다.

```
ection class="sec">
  <div class="left-card">
    <h1 class="lc-h1">안전 지수</h1>
    <div class="lc-score">
      <script>
          document.write("<p class='lc-result'>" + safety.toFixed(2) + "</p>");
      </script>
      <p class="lc-total">of 100</p>
    </div>
      <script>
          if (safety < 30) {
              safety_text = "매우 낮음";
          } else if (safety >= 30 && safety < 50) {
              safety_text = "낮음";
          } else if (safety >= 50 && safety < 70) {
              safety_text = "보통";
          } else if (safety >= 70 && safety < 100) {
              safety_text = "높음";
          } else if (safety >= 100) {
              safety_text = "매우 높음";
          }
          var color = (safety < 30) ? 'red' : (safety >= 30 && safety < 50) ? 'orange' : (safety >= 50 &
          document.write("<h2 class='lc-grade' style='color: " + color + ";'>" + safety_text + "</p>");
      </script>
    <p class="lc-remarks">
      이 안전 지수는 상대적인 자체 지표이며 절대적인 측정이 아닙니다
    </p>
  </div>
```

안전 수치값에 따라 색깔과 안전 수치 구분

각각의 안전 수치에 해당하는 값을 표기 값으로 만들도록 한다. 색깔 구분도 지어주고 웹페이지에는 소수점 두자리 수 까지만 나타낼 수 있도록 fix 작업을 진행한다.



최종 서비스 구현

평가 항목으로 데이터를 정렬시키고 각각의 시설의 개수를 웹페이지에 나타낼 수 있도록 구현하도록 한다. 최종적으로 웹페이지에서 나타내주는 페이지를 구성한다.

6. 프로젝트 결론 및 느낀점

어려운 작업이었지만, 생각보다 순조롭게 진행되어 문제들을 성공적으로 해결할 수 있었습니다. 범죄 데이터의 난해한 구조와 연단위로 정리된 데이터의 한계로 인해 몇 가지 어려움이 있었지만, 여러 가지 컬럼을 추가하고 새로운 데이터를 만들어내며 상관관계를 찾아내는 작업은 새로운 도전이었습니다. 범죄 분석은 데이터 뿐만 아니라 각 개인의 심리적인 요소도 고려해야 하기 때문에 예측이 쉽지 않았습니다.

CCTV와 검거율 간의 상관관계를 찾아보려 했으나 예상과는 달리 큰 연관성이 없었습니다. 그러나 서울시의 CCTV 개수를 살펴보면 우리나라의 치안 수준이 나쁘지 않다는 인식을 얻을 수 있었습니다. 상관관계가 크게 나타나지 않는 데이터로 예측 모델을 구축하려는 시도에서 여러 의문과 고민이 생기게 되었습니다.

장고 서비스를 구현하는 과정에서는 처음에 어려움을 느낄 수 있을 것이라 생각했지만, 여러 함수를 조합하여 좌표 중심으로 반경을 설정하고 마킹하는 작업을 성공적으로 마무리할 수 있었습니다. 앞으로도 이와 관련된 코딩 작업은 더 수월하게 해낼 수 있을 것으로 기대됩니다.

팀원들과 이전에 다양한 프로젝트를 수행했을 때, 팀 내 의사소통은 매우 원활하게 이루어졌습니다. 하지만 의견이 충돌하는 상황도 가끔 발생했습니다. 이러한 경우에는 정기적인 회의를 통해 의견을 조율하고 문제를 해결하는 방식으로 대응했습니다. 프로젝트 도중 발생한 문제나 의견 차이를 주기적으로 피드백하면서 의사소통을 강화했습니다. 의견 충돌을 조율하고 해결해 나가는 과정을 통해 프로젝트의 진행이 원활하게 이뤄지고, 마지막 단계에 도달하여 현재의 결과를 도출할 수 있었습니다. 이 경험은 효과적인 팀 협업과 의사소통의 중요성을 배울 수 있는 기회가 되었습니다.

이제 학원의 마지막 프로젝트를 통해 초보자에서 코드를 읽을 수 있는 수준으로 성장한 것 같아 뿌듯합니다. 그러나 이 분야에서는 지속적인 공부와 노력이 필요하며 계속해서 스스로 발전해 나가야 함을 깨닫게 되었습니다. 학습의 마무리가 아니라 시작에 불과하며, 더 나은 개발자가 되기 위해 끊임없이 노력하고자 합니다.