

캡스톤 일지

범죄정보 과학융합 훈련 프로그램 개발 보고서

Science Convergence Police Training Program
Development Report

2020년 03월 16일 ~ 2020년 06월 19일

영산대학교
사이버경찰학과
김 태 룡

목차

제 1 장 서론	1
1.1 개발 배경 및 목적	1
1.2 보고서 내용 및 구성	2
제 2 장 프로그램 제작 계획 수립	3
2.1 훈련프로그램 설정	3
2.1.1 훈련프로그램 구상	3
2.1.2 훈련프로그램 제작 범위 설정	5
2.2 유사서비스, 연구자료, 데이터 분석	7
2.2.1 경찰청 내부 프로그램	7
2.2.2 생활안전지도	8
2.2.3 경찰청 브리핑	9
2.2.4 공공 데이터 분석	10
2.2.5 동료 개발자 자문	12
2.2.6 종합 결과	14
2.3 DB 설계	14
2.3.1 데이터 공통분모 찾기 (DataDB)	15
2.3.2 데이터 지표 설정 (ModelDB)	16
2.4 음향 시스템 설계	16
2.5 Map API 설정	17
2.5.1 Map API 제공처 설정	17
2.5.2 Map API 서버 설정	18
2.6 동작 설계	18
2.6.1 연산	18
2.6.2 시각화	18
2.7 기타 설계	19
2.7.1 범죄 발생 예측 모델 생성법 설계	19
2.7.2 예측 모델 설계	20
2.7.3 위험지역 예측	20
2.7.4 데이터와의 상관관계	21
2.7.5 합법적 데이터 수집 방법	21
2.8 GUI 설계 및 기초 기능 구현	21
2.8.1 메인 화면	22
2.8.2 환경설정 화면	23
2.8.3 도움말	24
2.8.4 알림 화면	25

2.8.5 범죄통계 분석 메인 화면	25
2.8.6 모델 편집 화면	27
2.8.7 데이터 편집 화면	28
2.8.8 클라이언트 사이드 웹 서버	28
2.8.9 스크린샷	29
제 3 장 타당성 검증, 데이터편집기 개발	30
3.1 데이터 구조	30
3.1.1 필자가 계획한 데이터 구조	30
3.1.2 실제 서비스 중인 프로그램의 데이터 구조 및 알고리즘	31
3.1.3 기타	33
3.2 데이터 구조 타당성 검증	35
3.2.1 타당성을 위해 유지되어야 할 사항	35
3.2.2 타당성 향상을 위해 변경되어야 할 사항	36
3.2.3 타당성을 위해 사용되어야 할 알고리즘	36
3.2.4 기타 타당성을 위해 고려할 사항	37
3.2.5 추가해야 할 데이터	38
3.3 데이터 마이닝	38
3.3.1 환경 데이터	38
3.3.2 범죄 데이터	39
3.4 기능 구현	42
3.4.1 임의 데이터 생성	42
3.4.2 데이터 모델 변경	44
3.4.3 CSV 파일 지원	44
제 4 장 데이터 연산식 개발	46
4.1 기능 구현	46
4.1.1 데이터 처리 속도 향상	46
4.1.2 Kakao map API	46
4.1.3 통계 분석화면 수정 및 시각화 대상 선정 GUI 추가	49
4.1.4 격자 크기 설정	50
4.2 통계 알고리즘 구현	51
4.2.1 통계 알고리즘 구상	51
4.2.2 통계 알고리즘 구현	56
제 5장 시각화	71
5.1 시각화 계획 구상	71
5.1.1 시각화 방식 선정	71
5.2 개발	72

5.2.1 지도	72
5.2.2 인접지역 보너스	75
5.2.3 표	76
5.2.4 파이 차트	81
5.2.5 라인 차트	82
5.2.6 브리핑 페이지	84
5.2.7 임의 데이터 생성기능 향상	85
5.3 실험	87
5.3.1 실험조건	87
5.3.2 시각화 및 평가	89
5.3.3 평등화 계산식 적용 시각화 및 평가	98
5.3.4 시각화 종합 평가	102
제 6 장 향후 개발방향	103
6.1 한계	103
6.2 향후 개발방향	103

표 목 차

[표 1-1] 보고서 흐름도	2
[표 2-1] 아이디어 실현 가치 분석 표	6
[표 2-2] 사용 중인 데이터	8
[표 2-3] Map API 비교	17
[표 2-4] 개발 환경 표	21
[표 3-1] 기본 제공 모델(카테고리/엘리먼트) 및 추가 데이터	30
[표 3-2] 생활안전지도에서 사용 중인 데이터	32
[표 3-3] 보도자료에서 사용된 머신러닝 알고리즘	35
[표 4-1] 위험 지수 계산 식	54

그 림 목 차

[그림 2-1] 서비스 중인 생활안전지도	8
[그림 2-2] 보도자료 속 군집지도	10
[그림 2-3] 보도자료 속 변수 중요도 표	10
[그림 2-4] 범죄 통계 조회 화면	11
[그림 2-5] 고정식 교통 단속 장비 운영현황 csv 파일	12
[그림 2-6] 파일 형태로 저장된 SQLite 데이터	13
[그림 2-7] DB 설계 다이어그램	15
[그림 2-8] 사운드 풀을 만든 모습	17
[그림 2-9] 게임 메뉴 바 형식 NuriCHAIN 메인 메뉴	22
[그림 2-10] 구현된 메인화면	23
[그림 2-11] 프로그램 설명이 나오는 모습	23
[그림 2-12] 환경설정 화면	24
[그림 2-13] 도움말 화면	24
[그림 2-14] 알림 화면	25
[그림 2-15] 전후 행동 처리가 가능하도록 구현된 코드	25
[그림 2-16] 범죄통계 분석 메인 화면	26
[그림 2-17] Pane에 fxml 파일을 읽어 들이도록 구현한 코드	27
[그림 2-18] 모델 편집 화면	27
[그림 2-19] 데이터 편집 화면	28
[그림 2-20] 서버 구동 코드	29
[그림 2-21] 스크린샷 저장	29
[그림 3-1] PredPol에서 사용 중인 특허 알고리즘	32
[그림 3-2] PredPol의 서비스 중인 모습	32
[그림 3-3] 생활안전지도의 서비스 중인 모습	33
[그림 3-4] 오규철 교수님의 강의 내용	34
[그림 3-5] 보도 자료에 게시된 절차 프로세스	37
[그림 3-6] 필자가 지정한 CSV 파일 데이터 양식	39
[그림 3-7] 수집된 환경 데이터	39
[그림 3-8] 공공데이터 포털은 범죄 세부 데이터를 다루지 않는 모습	40
[그림 3-9] 본인(고소인/피고소인/피해자) 사건이 아니면 열람 불가한 모습	40
[그림 3-10] 생활 안전지도	41
[그림 3-11] 지도를 구성하는 레이어	41
[그림 3-12] 직접 경찰청에 전화하라는 답변	42

[그림 3-13] 임의 데이터 생성 GUI	43
[그림 3-14] 위 - 기존 DB / 아래 - 변경 후 DB	44
[그림 3-15] CSV 파일로 출력된 데이터 모습	45
[그림 4-1] html 속 javascript를 실행시키는 모습	46
[그림 4-2] 데이터 분포 범위를 표시 한 모습	47
[그림 4-3] 모든 마커를 제거하는 함수	47
[그림 4-4] 시간 범위를 구하는 Query문	48
[그림 4-5] 훨씬 짧아진 Query문	48
[그림 4-6] 사용될 스레드 수 제어	49
[그림 4-7] 기존 Main 화면	49
[그림 4-8] 수정 이후 Main 화면	49
[그림 4-9] 시각화 대상 설정 GUI	50
[그림 4-10] 격자 생성 및 줌 인/아웃	50
[그림 4-11] 보도 자료에 게시된 시각화 프로세스	51
[그림 4-12] 시간별 순위 매김 과정	53
[그림 4-13] 격자별 순위매김 과정	54
[그림 4-14] 계층 합산 방식	55
[그림 4-15] 5점 척도 색상 표시	56
[그림 4-16] 다른 개발자분의 답변	56
[그림 4-17] DB에 기입된 절대 건수	57
[그림 4-18] 서브쿼리가 가능한 모습	58
[그림 4-19] 서브쿼리 통합 쿼리	58
[그림 4-20] 단계별 수집되는 포인트 값	59
[그림 4-21] 살인 ‘엘리먼트’ 뿐만 아니라 치안점수 ‘포인트’까지 등록 된 모습	59
[그림 4-22] 신뢰도와 정확도 향상을 위한 ‘발생 시간’ GROUP BY	59
[그림 4-23] 시간 순으로 나열한 모습. 2순위 까지 연산된 것이 눈에 띈다.	60
[그림 4-24] 모든 환경변수에 대한 시각화 대상의 시간별 절대건수 오차 값	60
[그림 4-25] 데이터 28,860개 기준 리팩토링 전 : 33초/후 : 28초	61
[그림 4-26] 시각화대상의 섹터/시간별 발생 건수를 저장한 모습	62
[그림 4-27] PC 스펙에 적합한 스레드 수를 구하는 함수	63
[그림 4-28] 데이터 28,860개 기준 수정 전 : 6분 10초/수정 후 : 5분 51초	63
[그림 4-29] 데이터 28,860개 기준 (좌-기준 : 6분 57초) (우-수정 : 6분 22초)	64
[그림 4-30] 계산 타임로그	65
[그림 4-31] 알고리즘 변형 이후 타임로그	65
[그림 4-32] 총 계산 시간	66
[그림 4-33] 하이브리드 형식 계산 결과	67

[그림 4-34] 3번째 실험 결과	68
[그림 4-35] 위도에 따른 다른 두 점 사이의 거리를 나타내는 Java 함수	68
[그림 4-36] 위도 기반 두 점 사이의 거리를 구하는 쿼리	69
[그림 4-37] 데이터 수가 늘어나자 Domain error가 발생한 화면	69
[그림 4-38] 보호 장치가 추가되어 2배로 길어진 쿼리문	70
[그림 4-39] YEAR 데이터 까지 제대로 들어간 모습	70
[그림 5-1] 절대 건수 시각화를 진행 시킨 모습	72
[그림 5-2] 영향력 조절 패널	74
[그림 5-3] 발생 가능성 시각화를 진행 시킨 모습	74
[그림 5-4] 경고 메시지를 띄워주는 모습	75
[그림 5-5] 인접지역 보너스를 적용한 모습	76
[그림 5-6] 전체지역 선택과 특정 섹터 선택이 적용되는 모습	77
[그림 5-7] 데이터 표시 제어	78
[그림 5-8] 데이터 표시 범위 제어	79
[그림 5-9] 계산 방식 선택	80
[그림 5-10] 시각화 대상 선택	81
[그림 5-11] 파이차트	82
[그림 5-12] 다양한 대상의 절대건수를 비교하는 모습	83
[그림 5-13] 특정 대상의 발생 가능성을 시각화한 모습	84
[그림 5-14] 브리핑 페이지	85
[그림 5-15] 짧은 기간 옵션이 추가된 모습	86
[그림 5-16] 환경변수 선택이 가능한 모습	86
[그림 5-17] 억제 옵션이 추가된 모습	87
[그림 5-18] 시각화 대상 설정	88
[그림 5-19] 격자 편성화면	89
[그림 5-20] 브리핑 페이지	90
[그림 5-21] 살인 사건 절대 건수 분포도	91
[그림 5-22] 성폭행 사건 절대 건수 분포도	92
[그림 5-23] 유괴 사건 절대 건수 분포도	92
[그림 5-24] 향후 살인 사건이 발생할 가능성이 높은 지역	93
[그림 5-25] 향후 성폭행 사건이 발생할 가능성이 높은 지역	94
[그림 5-26] 향후 유괴 사건이 발생할 가능성이 높은 지역	95
[그림 5-27] 살인 사건 발생률이 가장 높았던 155번 지역 차트	96
[그림 5-28] 성폭행 사건 발생률이 가장 높았던 175번 지역 차트	97
[그림 5-29] 유괴 사건 발생률이 가장 높았던 174번 지역 차트	98
[그림 5-30] 평등화 보너스를 가져오는 쿼리구문	99

[그림 5-31] 평등화 값을 각 보너스 수치에 곱하는 모습	99
[그림 5-32] 살인 사건 발생률이 가장 높았던 155번 지역 차트	99
[그림 5-33] 향후 성폭행 사건이 발생할 가능성이 높은 지역	100
[그림 5-32] 성폭행 사건 발생률이 가장 높았던 140번 지역 차트	101
[그림 5-33] 유괴 사건 발생률이 가장 높았던 174번 지역 차트	102

제 1 장 서론

1.1 개발 배경 및 목적

2019년 12월, 인공지능 국가전략안¹⁾이 발표되고 나서 가장 먼저 떠오른 것이 영화 ‘Minority Report(2002)’였습니다. 일부 초능력자들이 생성한 빅데이터를 AI가 종합 분석하고, 수사 네트워크망을 통해 형사가 미래의 범인을 잡아내는 영화로, DNA(Data·Network·AI)를 핵심동력으로 삼은 정부 정책과 일치하였습니다.

하지만 필자의 경우 사이버보안학과(구 사이버경찰학과) 학생으로, AI 와는 거리가 멀었으며, 실제 범죄 정보를 열람할 권한이 없기에 마이너리티 리포트를 실현하려면 더 많은 정보가 필요하였습니다. 때마침 1월 중순경, 경찰행정학과 오규철 교수님께서 경찰대학 및 치안정책 연구소 견학 인원을 모집하셨기에 다양한 질문거리를 안고 참여하였습니다.

물론 견학 내내 ‘일반인에게도 범죄정보 공개가 가능한가?’에 대한 답변은 No 였지만, 견학 일정 동안 경찰행정학, 사이버보안을 융합시킨 범죄정보과학융합 전공에 대하여 오규철 교수님과 토론하던 중, 융합전공 학과 학생들의 학습에도 도움이 되고, 실제 치안에도 도움이 될 만한 ‘범죄정보 과학융합 학습 프로그램’ 초안을 구상할 수 있었습니다.

프로그램의 목적은 앞서 언급하였듯 교육용도뿐만 아니라 시각화를 통하여 실제 치안에도 도움이 되는 프로그램을 제작하는 것이며, 범죄정보가 일반인에게 공개되지 않기에 ‘임의 범죄정보 생성’ 기능을 부착하여 학부생도 빅데이터를 직접 다뤄보고, 모의 정책 수립 훈련을 할 수 있도록 돋는 것입니다.

이를 통해 사용자의 빅데이터 분석, 통계 분석, 인공지능 사용 능력 등을 향상시키는 것을 주 목표로 하고 있으며, 향후 수사권에서 사용 될 경우, 범죄 가능성을 줄이고 실전 상황에 도움 될 수 있도록 하는 것이 최종 목표입니다.

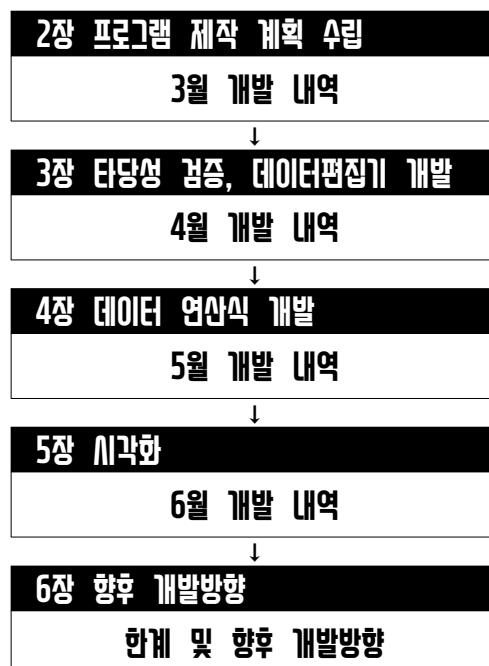
1) <http://www.korea.kr/news/pressReleaseView.do?newsId=156366736>

1.2 보고서 내용 및 구성

본 문서는 범죄정보 과학융합 프로그램의 개발 과정 및 결과를 중심으로 작성되었습니다.

2장에서는 개발 초기에 진행 되었던 계획 수립 및 GUI화면 설계 등에 대하여 다룰 것이며, 3장에서는 데이터 편집기를 제작함과 동시에 수립된 계획을 다른 서비스와 비교하며 타당성을 검증하는 내용에 대하여 다룰 것이며, 4장에서는 시각화에 사용 될 데이터 연산식 개발 과정에 대하여 다룰 것이며, 5장에서는 완성된 연산식을 이용하여 시각화 구현에 대한 주제를 다룰 것입니다.

이후 6장에서 한계와 향후개발 방향에 대해 알아보고 끝을 맺습니다.



[표 1-1] 보고서 흐름도

제 2 장 프로그램 제작 계획 수립

2.1 훈련프로그램 선정

Data, Network, AI 3가지 요건을 충족하면서도 수사대, 교수님 및 학생들에게 도움 줄 수 있는 프로그램을 제작해야 했기 때문에, 다양한 대상으로부터 조언을 구해보았으며, 제한된 시간, 자원 한계 내에서 제작 가능성이 높은 기능을 선정 해 보기로 하였습니다.

2.1.1 훈련프로그램 구상

① 범죄 통계 분석 프로그램

오규철 교수님 및 경찰행정학과 학생들과 함께 경찰대학 및 치안정책연구소 견학 당시 “서울청 여성 대상 범죄 빅데이터 분석”을 담당하셨던 김혜진 연구관님의 강연을 들을 수 있었습니다.

강연을 듣는 도중, 연구센터에서는 연구관님을 포함한 6명에서 전국 각 청의 요청사항에 대해 데이터를 분석하고, 결과를 도출하여 치안 정책 수립에 도움을 주는데, 이 때 경찰청 자체 통계 프로그램을 이용하신다고 하셨습니다.

영화 마이너리티 리포트와 달리 데이터가 충분하고, 통계를 구할 수만 있다면 예언자가 없어도 치안 정책에 큰 도움을 줄 수 있음을 알게 되어 ‘범죄 통계 분석 프로그램’을 구상하게 되었습니다.

이외에도 연구관님께서는 “데이터 분석은 선택과 집중이다. 데이터 분석은 쪼개기 작업으로, 절대건수만으로는 정책을 세울 수 없다. 범죄의 환경적, 사회적, 물리적 요인도 중요하다.” 등 데이터 분석법에 대한 팁을 주셨으며, 예산 문제로 경찰청 내부 프로그램에 새로운 컬럼을 추가하는 것이 쉽지 않다는 말씀을 하시며 유연한 데이터 관리가 뒤따라주면 좋다는 점을 일깨워 주셨습니다.

하지만 너무 통계 쪽으로만 치우칠 경우, 이용자가 직접 분석할 기회를 놓치면서 훈련용 프로그램으로써는 부적절하기에 “30대 중 후

반 여성 대상 성범죄를 막기 위해 CCTV를 어디에 설치하면 되는가?”와 같은 치안에 대한 질의를 내거나, 이용자가 해답을 내지 못할 경우, 프로그램 스스로 답을 내 놓는 기능을 부가적으로 생각해 보았습니다.

② 이동 경로 예상 프로그램

견학 당시 오규철 교수님께서 중국의 뛰어난 안면인식 기술과 인공지능에 대해 말씀하셨습니다. 안면인식 기술을 이용하여 공항뿐만 아니라 거리의 CCTV를 통해 개개인의 신원을 확인하고 있기 때문에, A라는 사람이 현상범인데 지금 어느 고속도로를 통해 어디로 가는지, B라는 사람은 어제 귀국하면서 누구와 같이 어디로 갔는지 1초 안에 모든 결과가 나온다고 하셨습니다.

교수님 말씀의 요지는 안면인식 기술에 대한 내용이었으나, 필자는 대화 도중 범인의 시간별 이동 데이터를 분석하여 미리 예상 위치에 잠복했다가 급습하여 체포하는 ‘잠복근무지 선정 프로그램’을 떠 올렸습니다. 물론 잠복근무 자체가 다양한 자료들을 바탕으로 이루어지고, 철저한 사전조사 끝에 이루어지는 것이나, 1초를 다투는 실제 상황에서 버튼 클릭 한번으로 대략적인 결과를 나타내어 준다면 수사관의 부담을 어느 정도 덜어 줄 수 있을 것 같았습니다.

③ 도주경로 차단 시뮬레이션

숙소 로비에서 오규철 교수님과의 대화를 이어 나가던 도중, 해외 경찰의 경우, 범인이 총기를 소지하고 있을 가능성이 높기 때문에 “○○ 상황에서는 □□로 단속해야 한다.”와 같은 규범이 잘 정리되어 있다고 하셨습니다. 여기서 필자는 중무장 한 채로 도주 중인 차량이 떠올리며 차량을 잡기 위해서는 어떤 전략을 세워야 할지 고민하던 도중, 도주경로 차단 시뮬레이션을 생각하게 되었습니다.

흉악범이 도로 위를 활개치고 다닐 경우, 예산과 인력, 장비를 아끼지 않고 투입하여 잡는 것은 당연하지만, 각 경찰서마다 보유 중인 인력, 장비, 작전 지역까지 도달하는 시간이 모두 다르고, 상공

에서 드론 혹은 헬리콥터 및 CCTV의 지원이 없는 지역도 존재하기 때문에 이를 모두 고려하여 작전을 세우려면 시간이 지체될 것이 당연하겠습니다.

때문에 미리 시뮬레이션을 통해 훈련을 하여 실전에 도움을 주고, 실제 상황이라면 판단 시간을 단축 시켜 줄 프로그램이 있다면 좋을 것 같다는 생각을 하게 되었습니다.

④ 위법성 판단 OX 퀴즈

AI와는 관계없지만, 오규철 교수님께서 요즘 경찰 시험을 치는 학생들이 문제(덤프)만 외워 치는 상황에 안타까워하시며, 프로그램 제작에 여유가 생긴다면 '위법성' 판단 OX 문제가 있으면 좋겠다는 의견을 전달 해 주셨습니다.

⑤ 음성 신고 도우미

견학을 마치고 사이버경찰학과 안미정 교수님께 자문을 구할 겸 필자의 프로젝트 계획을 설명 해 주었습니다. 교수님께서는 필자의 프로젝트에 긍정적인 반응을 보여 주시며 훈련프로그램 중 추가되었으면 하는 음성 인식 AI에 대한 아이디어를 보태 주셨습니다.

112 신고 전화 내용을 AI가 알아서 텍스트로 변환 시킨 다음, 형태소별 분리한 뒤, 주요 키워드만 수집하여 은어/유행어를 표준말로 변환시키고, 주변 상황 음성 또한 함께 분석하여(주변이 시끄럽다 = 사방이 트인 곳 등) 결과를 내어주는 음성 인식 및 단어 변환 프로그램에 대한 아이디어였습니다.

2.1.2 훈련프로그램 제작 범위 선정

다양한 훈련프로그램 아이디어가 모아졌지만, 교내에서는 최대 2만 원 까지만 지원되며(식대 포함), 3개월 만에 완성시켜야 하기에 유료 라이브러리 구매나 유료 논문 결제가 부담스럽고, 짧은 기간 내에 결과물이 나와야했습니다. 그 이외에도 충분히 교육적이어야 하며, 수사 이외 용도로도 사용할 수 있어야 했습니다.

아이디어	개발 난이도	교육성	AI 이용	DB이용	수사외 용도
범죄통계 분석	상	상	상	최상	통계 분석기
이동경로 예상	중	상	중	중	경로 예상기
도주경로 차단	최상	최상	중	상	게임
위법 판단 OX	하	상	하	중	게임
음성 인식	상	하	최상	상	번역기

[표 2-1] 아이디어 실현 가치 분석 표

범죄통계 분석 프로그램은 다양한 데이터가 필요하며, AI 및 Map API를 사용해야하는 등 시간이 많이 소요되는 요소이지만, 교육성도 짙고 Database, Network, Ai 모두 활용해야하기에 필자의 개발 기술 발전에 큰 도움을 줄 수 있으며, 통계를 내어 지도에 시각화시키는 동작 덕분에 다른 부/처에서도 다양하게 사용될 것 같았습니다.

이동경로 예상 프로그램은 ‘경로’에 대한 통계를 내기에 한 개체에 대한 시간별 이동 데이터만 있어도 되며, 높은 수준의 AI가 필요하지 않으나 범죄통계 분석 프로그램과 같이 Map API를 사용해야합니다. 시간과 위치 정보를 이용한 각종 교육에는 사용되겠지만 ‘경로 예측’이라는 틀을 벗어나지 못하기에 맷돼지 흔적을 찾는 산림청이나 기타 조난자 구조 등 특정 대상에 대한 경로 예측기기 정도로만 쓰일 것 같았습니다.

도주경로 차단 시뮬레이션은 Map API와 실제 경찰서 장비/인력 정보, 교통 정보 등 다양한 데이터가 필요하며, 도주 차량의 AI 등 제작 난이도가 상당히 높은 대신, 교육성은 최고인 장점을 가졌습니다. 하지만 도주경로 차단 시뮬레이션은 수사대 이외 부/처에서는 사용하기가 난해하며, 오히려 시뮬레이션 게임으로 Steam에 등록하면 꽤나 많은 다운로드 수를 기록 할 수 있는 프로그램 같았습니다.

위법판단 OX의 경우, 위법 상황을 임의로 만든 다음, 위법 판단의 키워드가 되는 단어만 적절히 배치하여 제작하면 되기에 필자의 개발 기술 발전에 전혀 도움이 되지 않으며, 경찰을 희망하는 취업 준비 학생들에게는 유용할지 몰라도 그 이외 사용자에게는 그저 게임으로

취급 될 확률이 높아 보였습니다.

음성인식 프로그램의 경우 은어/유행어가 매 달 업데이트 되어야 하며 AI 의존성이 높기에 AI 기술이 전무한 현재의 필자로써는 제대로 된 결과가 3개월 안에 나오기 힘들어 보였습니다. 수사 이외 용도로 쓰인다 해도 구글의 TTS 음성 변환 시스템에 주변 환경 음을 분석하고, 은어/유행어를 번역하는 ‘부가기능이 조금 더 붙은’ 음성 변환기에 그칠 것 같았습니다.

따라서 이번 4학년 1학기는 ‘범죄 통계 분석’ 훈련 프로그램만 제작하기로 하였습니다.

2.2 유사서비스, 연구자료, 데이터 분석

2.2.1 경찰청 내부 프로그램

견학 당시 김혜진 연구관님께서 경찰청 내부에는 범죄 통계 프로그램이 존재한다고 언급 하셨습니다. 경찰청 내부의 범죄 통계 프로그램은 실제 범죄 데이터를 다루기에 일반인에게 공개될 수는 없었지만, 방대한 양의 빅데이터를 구축 해 두었다고 하셨습니다.

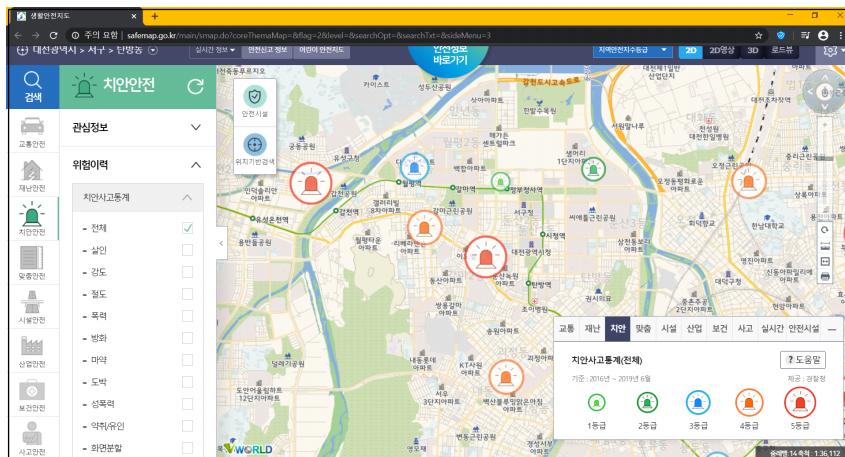
하지만 빅데이터를 쌓기 위해서는 0과 1처럼 수치화 가능한 데이터를 모아야 하는데, 수기로 작성된 사건일지가 “종로 3가에서 3시 20분경 절도사건 신고가 28세 여성 견주로부터 들어왔다. 절도 대상은 강아지고, 흰색에 포메라니안에 수컷이었다.” 일 경우, 현재 통계 프로그램으로 데이터 처리가 가능한 부분이 [종로 3가] [3시] [28세] [여성] [절도] [개] 정도뿐이라면, 개 절도범을 특정하기 위해 통계를 더욱 세밀하게 내고자 [색상] [품종] [성별] 데이터를 추가하려면 수사 프로그램 데이터 체계 자체를 갈아엎어야 하므로 몇십억이 들지 모른다고 하셨습니다.

이를 통해 다양한 데이터를 구하더라도 프로그램이 유동적이지 못하다면, 구해온 데이터를 100% 활용할 수 없기에 데이터의 유동성과 편집 기능이 필요 해 보였습니다.

따라서 실제 범죄 데이터를 사용하지만, 컬럼 추가/제거가 자유롭지

못 한(비용 문제) 이유로, 경찰청 내부 프로그램이 있더라도 작은 모집단을 특정하여 세부 통계를 내야 하는 상황에서는 컬럼 추가/제거가 자유로운 프로그램을 사용할 가능성이 있어보였습니다.

2.2.2 생활안전지도²⁾



[그림 2-1] 서비스 중인 생활안전지도

분류	서비스명	제공처
공통POI	치안안전	경찰청
공통POI	가로등	지자체
공통POI	방범등	지자체
맞춤안전	노인대상범죄현황	경찰청
맞춤안전	어린이대상범죄현황	경찰청
맞춤안전	여성범길치안안전	경찰청
치안안전	안전녹색길	경찰청
치안안전	치안사고발생현황	경찰청
치안안전	치안사고통계	경찰청

[표 2-2] 사용 중인 데이터

생활안전지도는 과거 발생했던 범죄/재해 등에 대한 통계치를 지도를 통해 보여주는 서비스입니다. 일반인들도 사용이 가능하며, 지도를 통한 시각화가 빠른 이해를 도왔습니다.

2) <https://www.safemap.go.kr>

과거사건 발생 통계를 표기 해 주는 서비스는 Data, Network측면을 포괄하고 있었지만, 다양한 주제를 다루다 보니, 수사에 국한된 심층 분석(범죄 발생률을 높이는 요인 분석 등)은 불가능 하였습니다.

따라서 지도를 통한 시각화를 지원 해 주되, AI가 데이터 분석을 도와주는 프로그램을 제작 한다면, 생활안전지도와 더불어 좋은 통계 서비스를 제공 해 줄 수 있을 것 같았습니다.

2.2.3 경찰청 브리핑³⁾

2019년 11월 29일 금요일자로 보도된 자료에 따르면, 치안정책의 패러다임을 전환하고 스마트 치안을 구현하기 위한 빅데이터 분석을 수행하였습니다. 수행 내역은 인천지역을 대상으로, 범죄·무질서 위험도 예측모델을 설계하여 월, 일, 2시간 단위로 범죄·무질서 발생 위험 지역을 예측하고, 발생 영향을 미치는 주요 환경적 요인을 파악한 것으로, 필자의 개발 방향을 지도 해 주는 좋은 자료였습니다.

또한 예측 모델을 현장에 적용하자 범죄 예방에 효과가 있는 것으로 나타났기에, 비슷한 프로그램을 만들 경우, 수사권에 직접적인 도움을 줄 수 있다는 희망을 안겨 주었습니다.

보도자료 중, 분석 대상 구역을 격자 형태로 나눈 다음, AI를 이용한 빅데이터 분석을 통해 5점 척도 군집화 시켜 지도상에 시각화 하여 결과를 출력하는 모습은 놀라웠으며, 이를 지표 삼기로 하였습니다.

3) <http://www.korea.kr/common/download.do?fileId=189406805&tblKey=GMN>



[그림 2-2] 보도자료 속 군집지도

이외에도 LSTM, 그래디언트 부스팅기반 Catboost 알고리즘을 사용하여 범죄 발생에 큰 영향을 미치는 환경 요인에 대한 분석을 통해 변수 중요도별 리스트링 해 놓은 자료를 보면 어떤 기능이 수사에 도움을 줄 수 있는지, 어떤 알고리즘을 사용해야 하는지 등 중요한 정보를 얻을 수 있었습니다.

□ 월 단위 예측 모델 변수 중요도(Catboost 모델)				
순위	환경기반 모델		건수기반 모델	
	변수명	중요도	변수명	중요도
1	음식-우홍주점	1.000	6개월 전 건수	1.000
2	음식-한식	0.846	11개월 전 건수	0.909
3	소매-종합소매점	0.141	12개월 전 건수	0.833
4	음식-커피점 카페	0.124	3개월 전 건수	0.807
5	숙박-모텔 여관여인숙	0.119	1개월 전 건수	0.796
6	비율(2개월 전 건수 / 2개월 전 남성 50대 유동인구)	0.097	5개월 전 건수	0.653
7	비율(2개월 전 건수 / 2개월 전 남성 40대 유동인구)	0.091	2개월 전 건수	0.600
8	거주인구 수	0.060	7개월 전 건수	0.593
9	비율(1개월 전 건수 / 1개월 전 남성 50대 유동인구)	0.054	9개월 전 건수	0.589
10	1개월 전 시간대별(2시~4시) 유동인구	0.048	10개월 전 건수	0.581

[그림 2-3] 보도자료 속 변수 중요도 표

2.2.4 공공 데이터 분석

경찰청 내부 데이터를 얻을 수는 없지만, 만일 얻었을 경우 필자의

고정식 무인단속장비 운영장소 (717개소) 2020.3.1.현재			
연번	서별	장 소 명	방 향
1	창원중부	창원시 성산구 성산동 남창원역 앞	신촌→안민
2	창원중부	창원시 성산구 양곡동 양곡삼거리	진해→창원
3	창원중부	창원시 성산구 가을정동 대우2차아파트 201동 앞	창원→장유
4	창원중부	창원시 성산구 성주동 천선삼거리	진해→공단
5	창원중부	창원시 성산구 양곡동 신촌삼거리	창원→마산
6	창원중부	창원시 성산구 남양동 불곡사거리	창원→진해
7	창원중부	창원시 성산구 귀산동 귀산터널 앞	창원→마산
8	창원중부	창원시 성산구 중앙동 시청사거리	명곡→중앙
9	창원중부	창원시 성산구 온난로 510(의회점곡 약)	서죽동→시촌동

[그림 2-5] 고정식 교통 단속 장비 운영현황 csv 파일

2.2.5 동료 개발자 자문

프로젝트를 진행하기에 앞서, 주위 의견과 조언을 들어보는 것이 좋겠다 생각되어, 교수님뿐만 아니라 동료 개발자에게 필자의 프로그램 제작 계획을 알려주고 의견을 물어 보았습니다.

첫 질문으로는 “사이버보안학과인데 보안 관련 프로그램(포렌식, 네트워크, 해킹 등) 안 만들고 경찰 프로그램을 왜 만드느냐”는 질문이 날아왔습니다.

보안학과도 궁극적인 목표는 사이버 치안이므로, 사이버 수사대를 돋는 프로그램을 만드는 것이 안 될 것 없으며, 현재 진행하려는 프로젝트는 준비해 왔던 다양한 프로젝트 계획서 중 일부일 뿐, 이 프로젝트가 끝나면 포렌식 툴킷 프로그램을 제작할 계획이 있다고 답변하였으며, 덧붙여 원래 학과명은 사이버경찰학과였지만, 복학했더니 사이버보안학과로 변경되어 있었다고 전하였습니다.

두 번째 질문은 “이미 경찰청에 더 좋은 통계 프로그램이 있고, 경찰청 브리핑을 보면 이미 프로그램이 만들어 진 것 같은데 굳이 왜 만드느냐”였습니다.

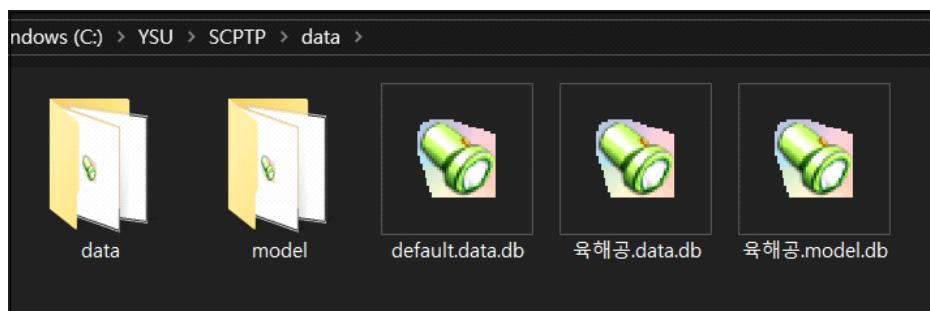
경찰청에 더 좋은 프로그램이 있어도 경찰을 희망하는 학생은 물론, 일반인들에게도 공개하지 않기 때문에 적어도 수사대를 희망하고, 빅데이터를 다루는 학생들을 위한 모의 훈련용 사제 프로그램 정도는 있으면 좋겠다고 생각하였으며, 김혜진 연구관님의 말씀에 따르면, 수

사대에서 사용하는 통계 프로그램은 새로운 모델 추가 및 편집 시, 비용이 많이 들기에 비록 성능은 떨어지더라도 필자가 만들 프로그램은 모델 및 데이터 편집을 유연하도록 제작하여 수사관들의 보조 수단으로 사용 될 수 있도록 개발을 진행 할 것이라 전하였습니다.

더하여 모델 및 데이터 편집이 유연하게 만듦으로써 범죄에 국한되지 않고, 경찰 통계 이외의 방면으로도 사용할 수 있도록 제작 할 것이며, 통계 프로그램이 있다는 것을 알고 입사 하는 것과 직접 다뤄보고 입사 하는 것, 직접 고안하여 만들어 보고 입사하는 것은 다르기 때문에, 후일 사이버 수사대 지원 시, 소프트웨어 개발 분야든 빅데이터 관리 분야든 다양한 분야에서 제작 경험을 발휘할 수 있을 것이라 전하였습니다.

세 번째 질문은 “Database는 MySQL, Oracle, MSSQL 등 다른 제품도 많은데 왜 굳이 SQLite를 사용하느냐”였습니다.

MySQL, Oracle, MSSQL 등은 클라이언트 단에서 설치가 필요하지만, SQLite는 설치 없이 바로 사용할 수 있기 때문이며, 이는 PC 사용에 익숙하지 못한 비전공자(경찰행정학과 등) 혹은 중장년층을 위한 배려로 작용될 것이라 전하였습니다.



[그림 2-6] 파일 형태로 저장된 SQLite 데이터

이외에는 하나의 스키마가 하나의 파일 데이터라 공유 및 백업하기가 쉽고, Android에서도 SQLite를 사용하기에 APP 버전 개발 시 데이터를 그대로 사용 할 수 있기 때문이라고 전하였습니다.

네 번째 질문은 “유해사이트 탐색 크롤러 NuriCHAIN⁶⁾과 같이 이번에도 JavaFX를 사용하여 개발 할 것인가”였습니다.

분명 Java보다 Kotlin이 대세이기도 하고, AI 혹은 빅데이터 관련은 Python, C++가 꽉 잡고 있어서 다른 언어로 사용할까 고민도 많았지만, Kotlin 기반 TornadoFX는 잔 버그가 많았고, Python과 C++ 대신 JavaFX로 작성하면 AndroidApp과 동작 방식이 유사하여 모바일 버전 제작 시 용이하며, 다양한 PC 환경에서 Java만 설치되어있다면 실행 가능하였습니다. 더하여 이전에 개발한 유해사이트 탐색 크롤러 NuriCHAIN 또한 JavaFX로 개발한지라 소스코드 일부를 재사용 할 수 있기 때문이라고 전하였습니다.

물론 JavaFX의 단점도 존재하는데, 연산속도가 Python, C++에 비하면 터무니없이 느리기 때문에 AI 사용이 힘들다는 것이었습니다. 때문에 빅데이터 통계를 내거나 AI 사용 시에는 외부 프로그램을 사용하여 계산 결과 값을 JavaFX 쪽으로 넘겨주는 방식을 고안할 것이라고 덧붙였습니다.

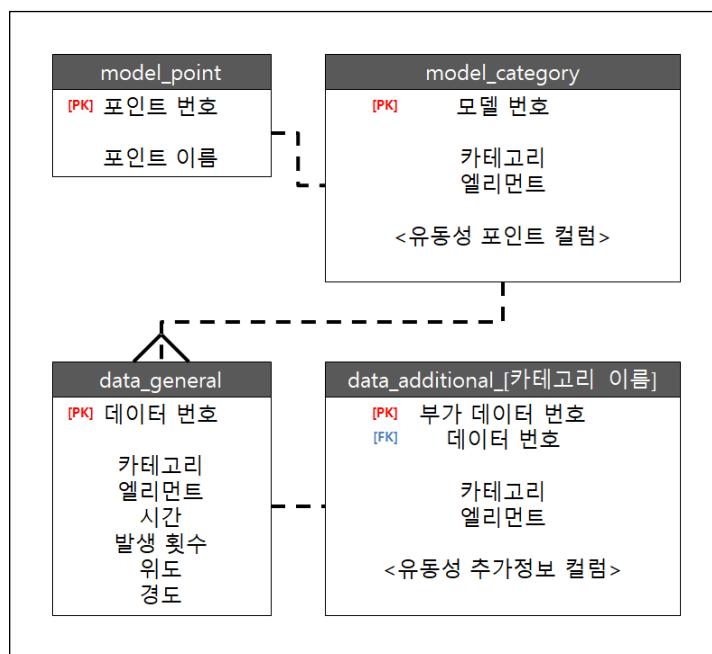
2.2.6 종합 결과

수사권 이외에도 도움을 줄 수 있는 ‘통계 프로그램’을 만들되 공공 데이터를 최대한 활용하여 데이터의 유기적 결합(컬럼 추가/제거가 자유로운)이 가능하게 만들고, Map API를 이용하여 지도를 통한 시각화를 지원 해 주되, 지역을 ‘특정 환경 변수’ 기준 5점 척도로 여러 구획으로 나누어 보기 쉽게 만들어 주도록 해야 하며, AI가 ‘특정 환경 변수’ 발생에 큰 영향을 미치는 환경 요인을 분석하도록 만들어야 한다는 결론이 나왔습니다. 여기에 ‘특정 환경 변수’에는 범죄가 들어갈 수도 있고, 물 소비량이 들어갈 수도 있는 등 제약이 없어야합니다.

2.3 DB 설계

6) <https://cafe.naver.com/goldbigdragon/85753>

통계 프로그램인 만큼 어떤 데이터를 수집할지, 수집된 데이터를 어떻게 저장하고 관리할 것인가에 대한 계획이 필요하였습니다. 어떤 계획을 세우느냐에 따라 프로그램의 정확성, 신속성이 판가름 나며 통계 프로그램의 존립 여부를 결정하기에 우선 통계에 사용될 데이터들의 공통 분모부터 찾은 다음, 나머지를 보조할 수단을 찾아보기로 하였습니다.



[그림 2-7] DB 설계 다이어그램

2.3.1 데이터 공통분모 찾기 (DataDB)

- 물리세계에서 발생하는 사건은 [시간, 장소, 횟수(개수)]가 남는다.
- 데이터에 따라 추가 정보가 다를 수도 있다.
 [살인 : 흉기, 피해자 나이, 피해자 성별] [CCTV : 모델명, 설치일자, 보유청사]
 - 성향이 비슷한 데이터는 엮을 수 있다.
 [강력범죄 [살인, 강도, 강간]] [경찰장비 [CCTV, 순찰차량, 순찰오토바이]]
 - 성향이 비슷한 데이터는 비슷한 부가정보를 가진다.
 [경찰장비 [CCTV : 모델명, 보유청사], [순찰차량 : 모델명, 보유청사]]

따라서 모든 데이터는 카테고리, 엘리먼트, 시간, 장소, 횟수에 대한 정보를 가지고, 이외 데이터는 부가정보 테이블에 넣기로 하였습니다. 이 때 각각의 데이터별로 부가정보 테이블을 가지게 만들 경우, 끔찍하게 많은 테이블이 생성되므로 동일 카테고리에 속한 데이터는 동일한 부가정보 테이블을 가지도록 설계하였습니다.

또한 데이터의 유기적인 결합/편집을 위하여 부가 데이터 테이블 속에는 부가 데이터 번호, 카테고리, 엘리먼트, 대상 데이터의 번호 컬럼만 고정하고, 이외 컬럼은 편집이 자유롭도록 설계하였습니다.

부가 데이터 테이블 속에 일반 데이터 테이블의 카테고리, 엘리먼트 컬럼이 중복되는 이유는 서브쿼리 시 속도 향상을 위함이며, 속도를 위해 용량 면에서 양보를 하였습니다.

2.3.2 데이터 지표 설정 (ModelDB)

카테고리와 엘리먼트가 많아진다면 이를 관리하는 DB도 필요해 보였으며, 데이터 DB 구조를 정한 후 확인해보니, 숫자를 계산하여 결과를 도출해야 하지만, 문자 정보가 많이 들어갔습니다.

이에 체스 말마다 점수가 달라, 동일 조건상 AI가 더 높은 점수의 말을 잡아내는 점에 착안하여, 각 데이터마다 가상 점수를 부여함으로써 높은 수준의 AI를 만들기보다는 부여된 점수 계산을 통해 ‘가장 최상의 점수를 얻는 쪽’을 선택할 수 있도록 각 데이터들의 가상 점수를 부여하고, 이를 관리할 수 있는 DB를 만들기로 하였습니다.

2.4 음향 시스템 설계

전자문서 서비스⁷⁾를 제작할 당시, TornadoFX를 이용하여 배달대행 서비스 프로그램 제작에 도움을 주면서 음향 재생 시스템에 문제가 있음을 발견한 적이 있었습니다. 동일 사운드에 대해 연속으로 재생 명령을 내릴 경우, 음이 중간에 끊기는 현상이 일어나는 것입니다.

7) <https://cafe.naver.com/goldbigdragon/84346>

따라서 통계 프로그램의 음향 시스템에는 사운드 풀을 생성하여 각 음원별 5개 조를 만들어 최대 5회 연속 재생 될 수 있도록 하며, 재생이 시작된 음원 오브젝트는 재생이 끝날 때 까지 호출할 수 없고, 재생이 끝난 음원 오브젝트는 호출 가능 상태로 전환시키기로 하였습니다.

```
public static Map<String, List<SoundObject>> musicMap = new HashMap<>();  
  
public SoundUtil(){  
    for(SoundType st : SoundType.values()) {  
        List<SoundObject> soundList = new ArrayList<>();  
        for(int count = 0; count < 5; count++)  
            soundList.add(new SoundObject(st));  
        musicMap.put(st.name(), soundList);  
    }  
}
```

[그림 2-8] 사운드 풀을 만든 모습

2.5 Map API 선정

2.5.1 Map API 제공처 선정

지도를 띄우고, 효과를 넣는 등 다양한 동작이 가능한 Map API를 제공해 주는 기업으로는 Naver, Kakao, Google이 있었습니다. 하지만 Google은 해외기업이라 그런지 한국 지도가 상세하게 나타나지 않았으며, 거리 뷰, 항공 뷰 등의 기능을 사용할 수 없었습니다.

때문에 Naver, Kakao 두 기업 중 하나를 선택해야 했으며, 우선 가격부터 비교해 보기로 하였습니다.

서비스 제공처	플랫폼	요금	최대 이용 한도	일단위 계산
Kakao	Web	무료	300,000 (일)	300,000 (일)
Naver	Web	무료	10,000,000 (월)	322,580 (일)

[표 2-3] Map API 비교

네이버 측에서 하루 2만 2천여 건을 더 검색할 수 있도록 지원해

주었지만, 회원 가입 시 결제 수단을 등록해야 했습니다. 결제 수단을 등록하면 혹시나 초과 한도가 일어 날 때 마다 결제가 이루어질까봐 안전하게 Kakao Map API를 사용하기로 하였습니다.

2.5.2 Map API 서버 설정

웹 서버를 만들어 지도를 띄운 후, 요청이 들어오면 지도에 표기하는 형식이 가장 이상적이었지만, 역시나 필자가 서버 컴퓨터를 가지고 있지 않았으며, 금전적 여유도 없었습니다.

또한 NuriCHAIN 개발 당시 교수님, 학교, 경찰서 등에서 서버 제공 의사를 밝혔었지만, 지금까지도 서버를 얻지 못하여 서버 의존적이던 NuriCHAIN 프로그램은 현재 전체 기능의 1/3 밖에 사용 할 수 없는 상태가 되었습니다. 이러한 시행착오를 겪은 탓에 이번 통계 프로그램은 메인 서버에 의존적이지 않도록 클라이언트가 자체 웹 서버를 구동 시키는 방식으로 제작하기로 하였습니다.

2.6 동작 설계

2.6.1 연산

[지역, 시간(년, 월, 일, 시간), Category, Element] 정보를 이용하여 데이터 범위를 설정하고, 절대건수 기반 단순 통계를 계산하거나, 환경변수별 영향력을 계산하여 특정 대상의 발생확률을 계산시킨 다음, 결과를 수치별 군집화 시켜 지도에 출력시키도록 합니다.

2.6.2 시각화

HttpServer 클래스를 이용하여 Web 서버를 구동 시킨 다음, JavaFX에서 WebView 태그를 통해 구동된 웹 서버 페이지를 띄울 계획을 세웠으며, 앞서 언급한 연산 규칙대로 계산 완료된 결과를 Web 서버에 전달하여 WebView 속 지도에 표기하기로 하였습니다.

[대상] 선택 시에는 Category, Element, Point 3가지 중 하나를 선택 할 수 있도록 하고, 그 중 Element 선택 시에는 Element 속 부

가정보(Additional data)를 설정하여 서브 쿼리가 가능하게 만들어 편리성을 향상시키기로 하였습니다.

① 표, 파이차트

[대상] 발생에 가장 큰 영향을 끼치는 [대상]을 찾아 순위별 나열.

EX) [강력범죄] 발생에 가장 큰 영향을 끼치는 [요소]를 30순위 까지 검색. 기준 점수로 '범죄 점수' AND 기준 점수로 '취약 점수'를 선정.

[전체/특정 지역]속 [대상]의 수치가 가장 높은 상위 10구역 통계.

EX) [위도:14.000 경도:32.31 ~ 위도:27.00 경도:42.32 AND 위도:15.00 경도:37.31 ~ 위도:21.00 경도:82.12]속 [범죄 점수]의 수치가 가장 높은 상위 10구역 통계

② 지도

[대상]의 절대 건수/수치가 높은 구역을 지도에 표시.

EX) [피해자 성별이 남성 AND 피해자 나이가 30 이상인 AND 발생 시각이 오후 3시 부터 오후 9시 사이인 성범죄]의 절대 건수가 높은 구역을 지도에 표시.

[대상]의 수치가 앞으로 상승할 구역을 지도에 표시.

EX) [피해자 성별이 남성 AND 피해자 나이가 30 이상인 성범죄]의 수치가 앞으로 상승 할 구역을 지도에 표시. 최선 대책 요소 선정 시 기준 점수로 '치안 점수'를 선정.

③ 꺾은 선 그래프

시간대별 [대상] 증감 수치 출력.

EX) [2010 ~ 2019]간 [피해자 성별이 남성 AND 피해자 나이가 30 이상인 AND 발생 시각이 오후 3시 부터 오후 9시 사이인 성범죄]의 증감 수치 출력.

2.7 기타 설계

2.7.1 범죄 발생 예측 모델 생성법 설계

1. 범죄 데이터를 모은다.
2. 환경 데이터를 모은다.
3. 범죄 데이터의 년/월 주기의 증감 수치에 대하여 통계를 낸다.
4. 환경 데이터를 바탕으로 범죄 발생 확률을 높이는 순위를 매긴다.
5. 다음 년도 증가할 범죄는 [범죄 발생 환경요소가 가장 많이 갖춰

진] AND [연도별 증감 수치가 높은] 연산을 한다. 다음 달 증가할 범죄는 [범죄 발생 환경요소가 가장 많이 갖춰진] AND [N월 증감 수치가 높았던] 연산을 한다.

2.7.2 예측 모델 설계

1. 모든 데이터에 [발생시각] [위도] [경도] 정보를 포함시키도록 한다.
2. AI 연산을 사용하지 않고 사용자 지정 연산 사용 시를 고려하여 모든 데이터는 고유 [기준 점수]를 가지도록 한다.

2.7.3 위험지역 예측

① 예측 기법

1. [예측 정확도 : 중간] [쉬움] 시간별 범죄의 절대 건수를 확인하고, 가장 많은 범죄가 일어났던 지역을 위험 지역으로 선정한다.
2. [예측 정확도 : 데이터 질에 따라 다름] [어려움] 범죄 발생 환경 요소를 범죄 발생 원인에 가까운 순으로 리스트하고, 시간별 증감 수치를 사용하여 범죄 발생 환경 요소가 가장 많이 밀집 되어있고, 시간대를 대조하여 가장 많은 위험 수치를 가진 지역을 위험 지역으로 선정하다.

② 타당성

1. [생활 안전 지도]가 [위험지역 예측 방법 1]과 비슷한 형태이며, 절대건수가 많을수록 해당 구역의 안전성이 떨어지는 것은 도시 개혁이 없는 한 유효한 결과입니다.
2. [경찰청 브리핑 2019.11.29(금) 조간 - 빅데이터를 통한 범죄 예측, 첫발을 내딛다]가 [위험지역 예측 방법 2]를 사용한 결과이며, 그 결과 신고 건수는 2018년 같은 기간 대비 666건에서 508건으로 23.7%, 범죄발생건수는 124건에서 112건으로 9.7% 감소한 실 사례가 있습니다.

③ AI / BigData 기술 사용처

- 통계결과 값으로 현재 어느 지역이 가장 위험한지는 알 수 있다.
- 하지만 예측은 단순 통계로 타당성을 입증 할 수 없다.
- 따라서 미래 범죄 예측 시에는 절대건수만을 이용한 계산 대신, 주위 환경 요소까지 파악하는 AI를 이용하여 야 한다.
- AI를 이용할 부분은 [범죄 환경 요소 리스트]이며, 그 이외 [범죄 발생 건수], [시간], [장소]는 AI동작이 필요 없다.

2.7.4 데이터와의 상관관계

- 범죄 데이터 이외 데이터를 '환경요소'로 분류.
- 범죄 데이터 분포에 따른 환경요소 분포도 작성.
- 특정 범죄에 대하여 많은 작용을 가하는 순서대로 환경요소 정렬.
- 모든 범죄 발생에 대한 모든 환경요소 정렬.
- 최후의 환경요소 정렬 결과가 범죄 발생에 가장 큰 영향을 주는 요인이며, 이를 기준 척도로 삼아 기준 점수를 매기면 범죄 발생 영향 평가가 가능하다.

2.7.5 합법적 데이터 수집 방법

- 범죄 데이터는 민간인의 신분으로 구할 수 없다.
- 환경 요소를 공공 데이터 포털에서 구한 다음, 범죄 데이터는 임의 데이터를 사용하여야 한다.

2.8 GUI 설계 및 기초 기능 구현

환경	값
OS	Windows 10 Home x64
IDE	IntelliJ IDEA 2018.3.4 (Ultimate Edition) JRE: 1.8.0_152-release-1343-b26 amd64 JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o
JDK	java version "1.8.0_121" Java(TM) SE Runtime Environment (build 1.8.0_121-b13)
Libraries	Gson 2.8.5, Jsoup 1.11.3, Sqlite-jdbc 3.23.1

[표 2-4] 개발 환경 표

2.8.1 메인 화면

범죄통계 분석 서비스 이외에도 이후 추가될 기능을 생각하면, 하나의 프로그램 안에 여러 서비스를 추가해야 하는 상황이기에 프로그램의 기본 프레임을 메뉴 형식으로 디자인 할 필요가 있어 보였습니다.

하지만 이전에 개발 했던 NuriCHAIN의 경우, 다양한 서비스를 한 프로그램 안에 넣은 게임 메뉴 바 형식의 디자인 때문에, 오히려 고연령층 사용자의 경우 사용이 불편하다는 의견을 들었습니다.



[그림 2-9] 게임 메뉴 바 형식 NuriCHAIN 메인 메뉴

또한 NuriCHAIN은 다양한 기능을 동시에 사용 가능하도록 설계하였기에 컴퓨터 자원을 모든 기능이 공유함으로써, 여러 기능을 동시에 실행 할 수 있었습니다.

하지만 범죄정보 과학융합 훈련 프로그램의 경우, 각각의 기능들이 컴퓨터 연산과 약한 수준의 인공지능, 빅데이터 처리 등 컴퓨터 자원이 많이 소모되는 기능들이며, 사이버 수사관 및 누리캅스 회원과 같이 하루 종일 컴퓨터를 접하는 사람 대신, 경찰행정학과 학생 혹은 교통경찰 등 컴퓨터를 활용할 일이 많이 없는 대상도 손쉽게 사용할 수 있어야 합니다.

때문에 내적으로는 한 가지 기능에 모든 자원을 몰아주는 형식으로 제작해야 하고, 외적으로는 메뉴 선택창에서 한 가지 기능만 선택할 수 있도록 제작하기로 하였습니다.



[그림 2-10] 구현된 메인화면

우측에는 환경설정 버튼 및 도움말, 개발자 Github 페이지로 갈 수 있는 버튼을 배치하였으며, 이용자가 선택 할 수 있는 모든 버튼에 ToolTip을 연결하여 버튼 위에 마우스를 3초 이상 올려둘 경우, 각 버튼에 대한 간단한 설명을 보여주도록 하였습니다.

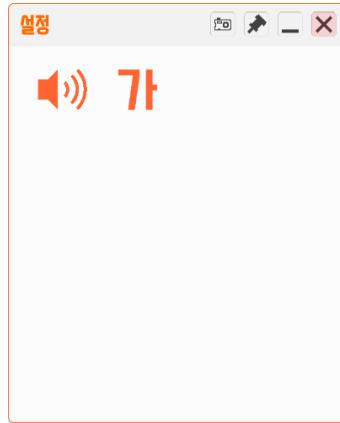
이외에는 선택 가능한 프로그램 위에 마우스를 올려 둘 경우, 하단 설명란에 해당 프로그램이 제공 해 주는 기능에 대하여 간략한 설명이 나오도록 하였습니다.



[그림 2-11] 프로그램 설명이 나오는 모습

2.8.2 환경설정 화면

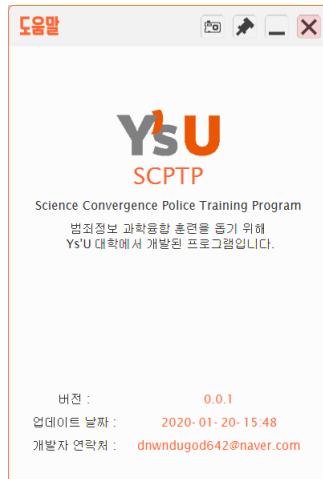
효과음 On/Off 버튼을 추가 해 두었으며, 아직 한국어 이외는 지원하고 있진 않지만, 후일을 위해 언어 선택 버튼도 추가하였습니다.



[그림 2-12] 환경설정 화면

2.8.3 도움말

현재 프로그램의 버전 및 업데이트 날짜를 확인 할 수 있고, SCPTP의 뜻과 함께 대학교 마크를 넣었습니다. 대학 마크 클릭 시, 영산대학교 홈페이지가 팝업되며, 개발자 연락처 클릭 시, 개발자 카페가 팝업 되도록 하였습니다.



[그림 2-13] 도움말 화면

2.8.4 알림 화면



[그림 2-14] 알림 화면

에러 및 작업 진행 중을 알리는 화면으로, 알림 이외에도 스레드를 이용하여 알림창이 떴을 때 실행시킬 작업과, 알림창이 사라진 다음 실행시킬 작업을 설정 할 수 있도록 하였습니다.

```
@Override  
public void initialize(URL location, ResourceBundle resources) {  
    Thread t = run() -> {  
        for(BeforeWorkType bwt : beforeWorkList) {  
            if(bwt == BeforeWorkType.CREATE_CRIMINALSTATISTICSANALYSIS_DEFAULT_MODEL) {  
                createDefaultModel();  
            } else if(bwt == BeforeWorkType.OVERWRITE_CRIMINALSTATISTICSANALYSIS_MODEL) {  
                overwriteModel();  
            } else if(bwt == BeforeWorkType.LOAD_CRIMINALSTATISTICSANALYSIS_DATABASE) {  
                loadData(param);  
            }  
        }  
        afterr();  
        interrupt();  
    };  
    Platform.runLater(() -> {  
        this.title.textProperty().setValue(titleText);  
        t.start();  
    });  
}
```

[그림 2-15] 전후 행동 처리가 가능하도록 구현된 코드

2.8.5 범죄통계 분석 메인 화면

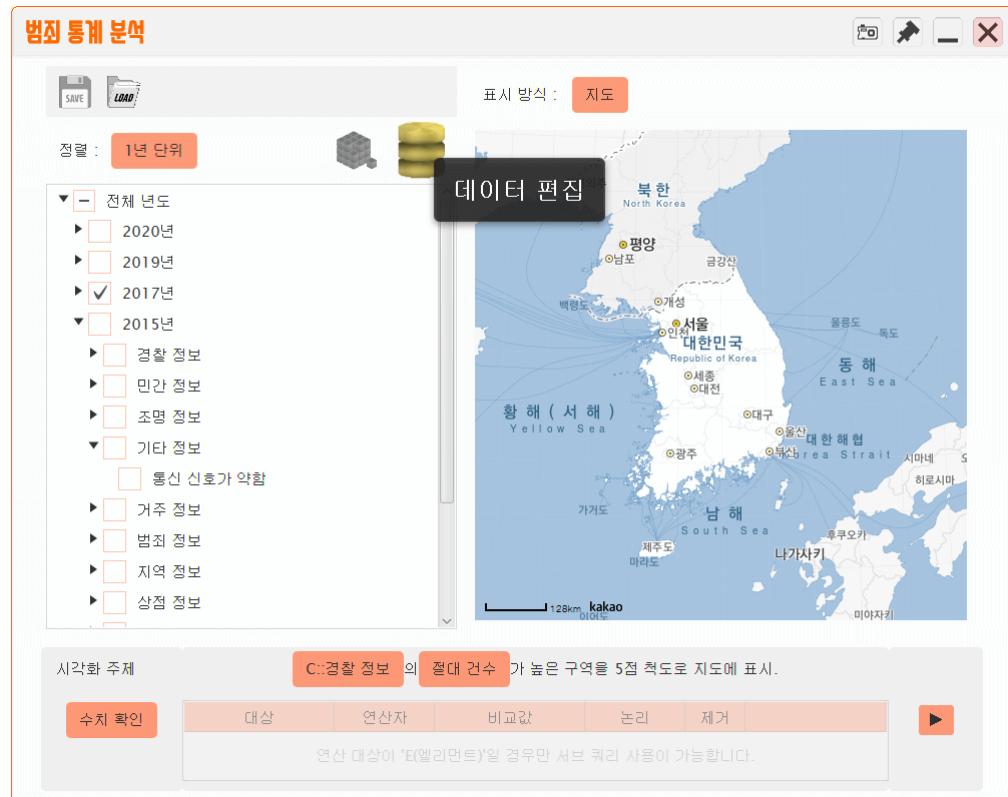
좌측에서 데이터를 선택하고, 우 측에서 연산 결과를 시각화 시키는 구조로 화면을 구성하였으며, 현재 불러온 데이터, 모델 셋을 다른 이름으로 저장하는 버튼과, 외부로부터 데이터 및 모델 셋을 불러올 수

있는 기능을 우측 상단에 배치하였습니다.

불러오기 선택 시, 다중 선택이 가능하게 하였으며, 데이터는 데이터끼리 합치고, 모델은 모델끼리 합쳐 동시에 불러 올 수 있도록 함으로써 편리함과 실용성을 동시에 잡았습니다.

하단부에서는 통계 및 시각화 주제를 선정할 수 있도록 하여 직접적인 쿼리문을 작성하지 않아도 되게끔 이용자들의 편의를 고려하였습니다. 다만 [검색] 버튼을 아직 꾸미지 않았기에 ► 기호를 임시로 사용하고 있습니다.

시각화 방식을 선택 할 수 있는 메뉴 버튼에는 지도 이외에도 표, 파이차트, 꺾은 선 그래프 등 다양한 시각화 방식을 선택 할 수 있도록 설계 해 두었습니다.



[그림 2-16] 범죄통계 분석 메인 화면

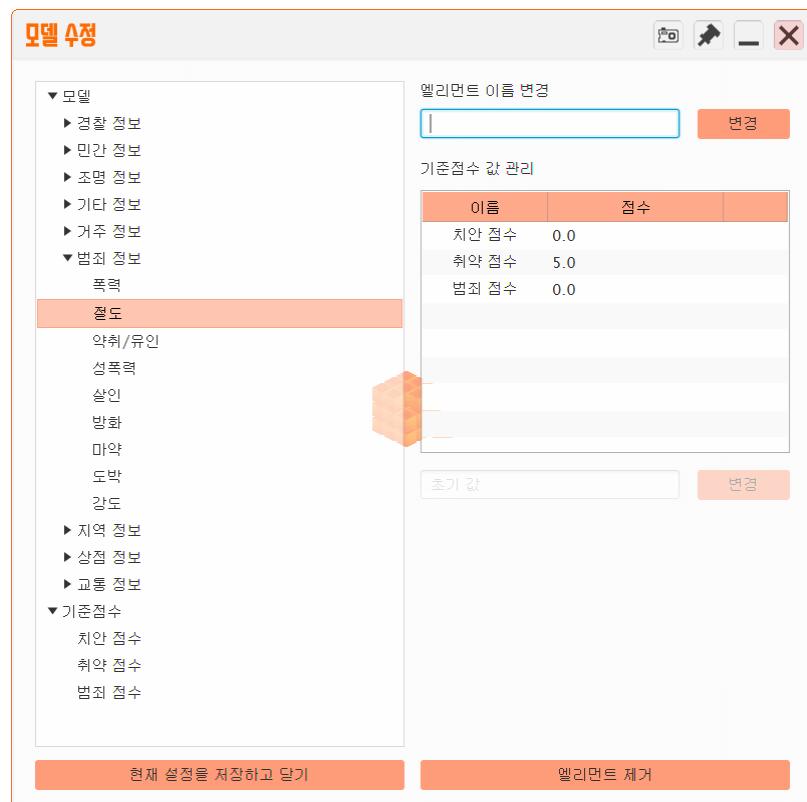
2.8.6 모델 편집 화면

기준점수 및 카테고리, 엘리먼트에 대한 설정을 할 수 있는 모델 편집 화면을 만들기 위해 좌측에서 대상을 선정하고, 우측에서 선택한 값에 대한 설정을 할 수 있도록 하였습니다.

이 과정에서 하나의 fxml 안에 우측에 나타날 다양한 기능을 집어 넣는다면 관리가 어려워 질 것이 분명하였기에, 우측의 편집 노드들을 기능별로 fxml 파일에 따로 작성하여 불러오게 하였습니다.

```
private void loadFXML(String file) {
    activePane.getChildren().clear();
    try {
        Parent root = FXMLLoader.load(getClass().getResource( name: "/resources/fxml/" + file + ".fxml"));
        activePane.getChildren().setAll(root);
    } catch(IOException e) {}
}
```

[그림 2-17] Pane에 fxml 파일을 읽어 들이도록 구현한 코드



[그림 2-18] 모델 편집 화면

2.8.7 데이터 편집 화면

데이터 검색/추가/제거/편집이 가능한 데이터 편집 화면을 구현하였으며, 이용자 편의를 위해 상단부에서 쿼리문 작성 없이 데이터 쿼리가 가능하게 하였습니다.

쿼리 결과는 중앙의 테이블에서 확인 할 수 있으며, 중앙 테이블 속 데이터 클릭 시, 하단 탭 중, ‘추가정보’ 탭이 활성화 되면서 해당 데이터의 추가정보를 편집 할 수 있게 됩니다.

이외에도 각 카테고리별로 추가정보를 등록/제거/편집 할 수 있는 탭을 생성 해 두었기 때문에 언제든 컬럼을 확장/제거 할 수 있도록 하였습니다.

The screenshot shows a 'Data Editor' window with the following components:

- Header:** Includes tabs for '모든 카테고리', '속', '모든 엘리먼트', '를', '번호', '기준', '오름차순', '정렬로', and '검색'.
- Main Table:** A grid showing 10 rows of data with columns: 번호, 카테고리, 엘리먼트, 시간, 개수, 위도, 경도, and 제거. Row 4 is highlighted.
- Pagination:** Shows page 1 of 2.
- Bottom Tabs:** Includes '데이터 추가', '추가정보 등록/제거', and '추가정보'.
- Additional Information Tab:** A table showing basic information for item ID 1.

이름	타입	기본 값	제거
모델	TEXT	NULL	☒
배치일	LONG	0	☒
업무	TEXT	NULL	☒

Below this table are input fields: '이름 : 추가정보 이름을 입력하세요', '기준 타입 : 문자열', '기본 값 : 공백 시 기본값 NULL', and a '등록' button.

[그림 2-19] 데이터 편집 화면

2.8.8 클라이언트 사이드 웹 서버

ApacheTomcat/NodeJS/SpringFramework 등을 쓸까 싶었지만

무겁기도 하고, 설치가 필요하여 편리성이 아쉬워도 Java에서 기본적으로 지원하는 HttpServer 클래스를 이용하기로 하였습니다.

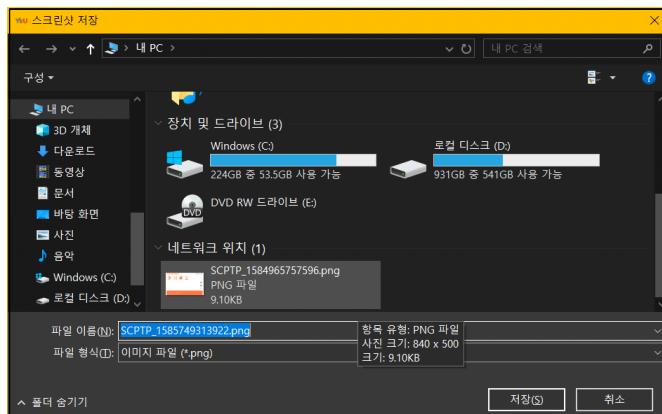
```
private HttpServer server = null;

public void start(int port, Map<String, HttpHandler> handler) {
    try{
        server = HttpServer.create(new InetSocketAddress(port), 0);
        for(String path : handler.keySet()) {
            server.createContext(s: "/" + path, handler.get(path));
        }
        server.setExecutor(null);
    } catch(Exception e) {
        new AlarmAPI().serverCreateError();
    }
    if(server != null) {
        server.start();
    } else {
        new AlarmAPI().serverCreateError();
    }
}
```

[그림 2-20] 서버 구동 코드

2.8.9 스크린샷

모든 화면에 ‘스크린샷’ 버튼을 추가하여 사용자 편의를 향상시켰습니다. 스크린샷 버튼 클릭 시, 카메라 사운드가 재생되면서 SCPTP_[UTC].png 형태의 이미지 파일을 생성하게 됩니다.



[그림 2-21] 스크린샷 저장

제 3 장 타당성 검증, 데이터편집기 개발

4월 초, DB속에 데이터를 입력하고 있을 무렵, 데이터 자체의 신뢰도도 중요하지만, 데이터를 어떻게 처리할 것인가에 대한 신뢰도가 매우 중요하다고 정민포 교수님께서 조언 해 주셨습니다.

때문에 실제 서비스 중인 프로그램과 필자의 프로그램을 비교 하면서 어떤 데이터를 사용 하여 어떤 가공을 거칠 것인지, 어떤 알고리즘을 통해 통계를 낼 것인가에 대한 고민을 하였습니다.

3.1 데이터 구조

3.1.1 필자가 계획한 데이터 구조

현재 필자의 프로그램은 [범죄 정보 카테고리 속 강도 엘리먼트] 형식으로 카테고리-엘리먼트 관계로 데이터를 다루고 있으며, 각각의 엘리먼트들은 [시간, 장소, 횟수(개수)] 3가지 정보를 공통적으로 가지며, 이외 정보들은 ‘추가 정보’로 별개취급하고 있습니다.

때문에 DB 상에서 [시간, 장소, 횟수(개수)]를 가진 공통 데이터 테이블에 컬럼 추가/제거가 불가능 하지만, ‘추가 정보’ 영역에는 컬럼 추가/제거가 자유롭기 때문에 다양한 정보를 담을 수 있습니다.

카테고리	엘리먼트	추가 데이터 컬럼
경찰 정보	헬리콥터/순찰 차량/드론/긴급 신고 버튼/교통 단속 카메라/경찰 인력/CCTV	모델/배치일/업무
민간 정보	높은 유동인구/낮은 유동인구	-
조명 정보	어두움/밝음	-
기타 정보	통신 신호가 약함	-
거주 정보	다인 가구/깨진 창문/1인 가구	거주자 성별/거주자 연령
범죄 정보	폭력/절도/약취, 유인/성폭력/살인/방화/마약/도박/강도	가해자 성별/가해자 연령/피해자 성별/피해자 연령/주거지/층수/대상/타입
지역 정보	주택가/논,밭/공업단지	-
상점 정보	한식/유흥/높은 매출/낮은 매출	-
교통 정보	상습 결빙구간	-

[표 3-1] 기본 제공 모델(카테고리/엘리먼트) 및 추가 데이터

3.1.2 실제 서비스 중인 프로그램의 데이터 구조 및 알고리즘⁸⁾

① PredPol⁹⁾

㉠ 사용 중인 데이터 필드

- 사건 ID : 범죄마다 고유 식별 코드 부여.
- 범죄/사건 타입 : 특정 범죄 혹은 사건에 대한 타입을 지칭.
- 발생 장소 : 최고의 정확성을 위해 위도/경도를 기입.
- 사건 시각 : 사건의 시작 및 종료 날짜. PredPol은 시작/종료 시간 사이의 중간 지점을 취하여 사고 발생 시간을 계산하므로 시작/종료 시간이 72 시간을 초과하는 사고는 예측 정확도가 저하되므로 제외시킴.
- 사고 기록 수정시간 : 이 필드는 선택 사항.

㉡ 사용 중인 알고리즘 (본문 내용)

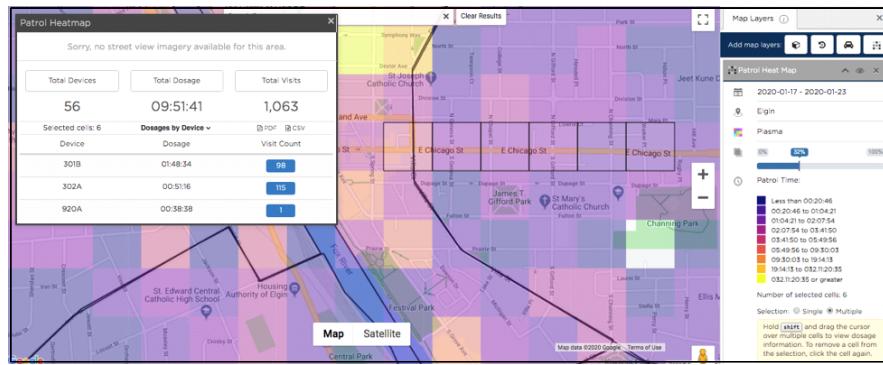
- PredPol은 범죄 패턴 형성의 원인에 대한 연구를 10년간 진행하였으며, 이를 기반으로 범죄 행위의 몇 가지 주요 원인을 찾아 수학적 구조와 성공적으로 연결시켰습니다.
 - ‘집에 사람이 있는지’ ‘값어치 있는 물품이 있는지’ 모르는 채로 알려지지 않은 집에 들어가는 것이 훨씬 위험하기 때문에 범죄자는 ‘합리적’으로 이전에 범죄에 성공한 장소로 돌아갑니다.
 - 반복되는 피해는 이웃집도 위험에 처해 있음을 알려줌. 이웃도 피해자와 매우 흡사하기 때문. 비슷한 사회 경제적 지위, 비슷한 출근 시간, 비슷한 주거 형태, 비슷한 재산. 이 때문에 피해자에게 사용한 ‘범죄 계획’이 이웃집에 거의 완벽하게 적용됩니다.
 - 지역 탐색도 중요한 요소 중 하나로, 가해자의 집은 주요 활동 지점과 멀리 떨어지지 않은 곳에 있으며, 한 지역에 집중되는 경향을 가지고 있기 때문입니다.

8) 영문사이트는 한글로 번역하였으며, 공개된 자료만 기입하였기에 실제 사용되는 데이터는 더 많을 수도 있음.

9) <https://www.predpol.com/technology>, <https://twitter.com/PredPol>

$$\frac{\partial A}{\partial t} = B + \frac{\eta D}{4} \nabla^2 A - \omega A + \theta \omega \delta$$

[그림 3-1] PredPol에서 사용 중인 특허 알고리즘



[그림 3-2] PredPol의 서비스 중인 모습

② 생활안전지도¹⁰⁾

① 사용 중인 데이터 (중복 및 기타 환경/통계 데이터 제외)¹¹⁾

분류	서비스명	제공처
공통POI	치안안전	경찰청
공통POI	가로등	지자체
공통POI	방범등	지자체
맞춤안전	노인대상범죄현황	경찰청
맞춤안전	어린이대상범죄현황	경찰청
맞춤안전	여성범길치안안전	경찰청
치안안전	안전녹색길	경찰청
치안안전	치안사고발생현황	경찰청
치안안전	치안사고통계	경찰청

[표 3-2] 생활안전지도에서 사용 중인 데이터

10) <https://www.safemap.go.kr/>

11) <https://www.safemap.go.kr/dvct/data/selectDataAPIList.do>

④ 사용 중인 알고리즘

- 이미 연산 작업이 끝난 통계 자료를 레이어 형태로 보여주기만 하기에 연산/알고리즘이 없다.



[그림 3-3] 생활안전지도의 서비스 중인 모습

3.1.3 기타

① 오규철 교수님 강의

- 뉴욕시와 마이크로소프트가 협력하여 Domain Awareness System(대테러 감지 시스템) 구축 시 지능형 CCTV, 총소리, 방사능 탐지센서, 자동차 번호판 인식장치, CCTV 감지망/센서 등을 이용하여 살인사건 75% 해결하였다.
- 캘리포니아 주 산타크루즈에서는 PredPol을 이용하여 범죄 신고지역-발생지역 분석, 위험도 예측하여 범죄 대응. 추정된 위험도를 순찰 차량 배치에 활용하여 2013 ~ 2014년 동안 범죄율 20%, 차량절도 20%, 강도사건 32%가 감소하였다.
- 경찰 데이터(수사기록, 범죄통계, 순찰, 정책) + 공공, 공개 정보(CCTV, 환경 정보, 소셜)을 이용한 스마트 경찰을 강조하였다.
- 지리적 범죄 분석 시스템(GeoPros)를 이용하여 범죄 발생 위치를 GIS-범죄학 이론과 결합하여 용의자 거점을 추적하며, 이를 확장 시켜 범죄 빈발지역 추출, 범죄 위험지수를 지도에 표현하는 것이 가능하다고 한다.

- 수사/내사 결과보고서에는 사건번호, 발생일시, 피해금액, 범행장소, 범행수법, 침입수단, 침입구, 침입 방법, 피해물품 등의 평균 20~40여개의 다양한 요소가 들어간다.
- 한국 경찰은 Geo, CSS, KICS 접수 데이터, 위험요소 정보(우범자 등) 등의 데이터를 활용, KCIS, I2, I3, GeoPros, Holmes와 연계하여 범인의 특정 단서를 제공할 수 있다.
- 분석 시에는 다음 3가지 절차를 따른다. 1. 데이터 분석 (관계망 형성) 2. 지리적 분석 3. 확률모델 생성 절차를 통해 분석.



[그림 3-4] 오규철 교수님의 강의 내용

② 보도자료¹²⁾

⑦ 사용된 데이터

- 범죄(살인·강도·성폭력·절도·폭력), 무질서(주취자·시비 등) 112 신고 10종
- 112신고·범죄통계 등의 치안데이터를 중심으로 소상공인시장진 흥공단의 소상공인 데이터(8만건), 인천시의 항공사진(16.2GB)뿐만 아니라, SK텔레콤의 유동인구(530만건)·신용카드 매출정보(521만 건) 등 민간과 공공의 다양한 데이터를 결합하여 활용. (총 사용된

12) <http://www.korea.kr/common/download.do?fileId=189406805&tblKey=GMN>

데이터 약 1억59만 건)

④ 데이터 가공법

- K-means 알고리즘을 활용하여 무질서·범죄 건수 및 비율을 기반으로 격자를 자동으로 군집화 분석을 위해 인천 지역을 가로 200m× 세로 200m 크기의 2만3천여 개 격자로 나누고 Long Short Term Memory(시계열 데이터를 처리하기 위한 딥러닝 모델) 알고리즘을 통해 5개의 군집으로 분류함.
- 신고·범죄 건수뿐만 아니라 환경적 요인을 결합하여 범죄 위험도 예측모델을 개발했으며, 이 과정에서 LSTM 딥러닝 알고리즘과 그래디언트 부스팅 기반의 Catboost 알고리즘을 활용함.

구분	선형회귀	GBM (Catboost)	딥러닝 (LSTM-Attention)
모델	변수들 간의 관계를 선형함수식으로 모형화	다수의 단순 모델(의사결정나무)을 순차적으로 결합	신경망을 사용하여 집중할 변수를 선별하도록 학습
특징	단순하고 학습 빠름, 과소적합 가능성 있음	매개변수 값에 따라 정확도, 과소·과대적합 등 성능 차이 존재	시계열 특화, 학습 속도가 느리고 학습에 많은 데이터 필요

[표 3-3] 보도자료에서 사용된 머신러닝 알고리즘

⑤ 성과

- 경찰청은 지난 10월 14일부터 6주간 범죄 예측 결과를 기반으로 인천시의 16개 지역에 경찰관과 순찰차를 집중 배치하여 신고 건수는 2018년 같은 기간 대비 666건에서 508건으로 23.7%, 범죄발생건수는 124건에서 112건으로 9.7% 감소했다.

3.2 데이터 구조 타당성 검증

3.2.1 타당성을 위해 유지되어야 할 사항

생활안전지도에서 데이터를 그룹화 시켜 사용하는 모습을 보아 [카테고리-엘리먼트]별 그룹을 지어 사용하는 필자의 처리방식은 그대로 유지하도록 했습니다.

이외에도 수사/내사 결과 보고서에는 평균 20~40여개의 다양한 요소가 들어간다는 사실에 따라, 모든 사건에 공통적으로 들어가는 시간, 장소, 횟수 컬럼 이외의 데이터는 기존 방침대로 일반 데이터에 함께 넣지 말고 ‘추가 데이터 Table’에 따로 저장하기로 하였습니다.

3.2.2 타당성 향상을 위해 변경되어야 할 사항

PredPol의 알고리즘을 보고, 모든 사건은 시작 시간과 종료 시간이 있었음을 깨닫게 되어, 이미 생성 해 두었던 ‘시간’ 컬럼을 제거하고, 그 자리에 ‘시작 시간’, ‘종료 시간’ 컬럼을 추가하기로 하였습니다.

또한 오규철 교수님의 강의를 통해 기본적으로 제공되는 데이터를 더욱 세분화시키기로 하였습니다.

예) ‘경찰 정보’ -> ‘치안 장비’ ‘순찰 장비’ ‘경찰 인력’ ‘경찰청’

3.2.3 타당성을 위해 사용되어야 할 알고리즘

보도 자료에서 실행한 대로 ‘범죄 데이터의 단순 수치(건수) 통계’를 이용하여 건수 기반 예측 모델을 만들고, ‘모든 데이터’를 이용하여 환경기반 예측 모델을 만든 다음, 각 모델을 통한 결과를 지도에 나타내면 데이터 처리절차의 타당성을 인정받을 수 있습니다.

하지만 모델 설계 단계에서 어떠한 방법을 이용하였는지 자료에서 공개하지 않았으므로 필자가 만들어 사용한 알고리즘은 알고리즘의 타당성을 검증 받을 수 없었습니다.



[그림 3-5] 보도 자료에 게시된 절차 프로세스

보도 자료에서 사용된 것과 같이 정확도가 상대적으로 낮지만 데이터가 x, y축 만으로 이루어 질 경우 빠른 결과를 나타내는 선형회귀 기법 및 Gradient Boosting Machine(Algorithm)은 ‘사용자 지정 점수’값을 분석 할 때 사용할 경우, 사용된 알고리즘에 대한 타당성을 인정받을 수 있어보였습니다.

예) 울산광역시에서 발생한 범죄 데이터, 환경 데이터를 모아 x축을 점수, y축을 횟수로 설정하여 범죄점수/치안점수/취약점수 등을 높이거나 내리는 요인대로 구분한 뒤, 범죄점수를 내리는데 가장 효율적인 환경을 선정하거나, 치안점수를 내리는 가장 큰 원인을 찾는다.

또한 오규철 교수님의 강의 속 실제 수사대가 사용 중인 [1. 데이터 분석 (관계망 형성) 2. 자리적 분석 3. 확률모델 생성절차를 통해 분석.] 절차를 따를 경우, 분석 알고리즘에 대한 타당성을 얻을 수 있어 보였습니다.

3.2.4 기타 타당성을 위해 고려할 사항

생활안전지도는 데이터마다 고유 위치를 가지고 있고, 출처가 분명하기 때문에 생활안전지도에서 사용된 데이터를 구할 수 있다면 사용하는 것이 좋아보였습니다.

경찰청 보도 자료에서 LSTM 딥러닝을 위해 사용된 환경 데이터 수가 약 1억59만 건임을 감안하여 상대적으로 적은 데이터 수를 다룰 필자의 경우, 데이터 수가 많을수록 정확도가 올라가는 LSTM 딥러닝

알고리즘을 사용하기에는 부적합 하므로, 사용자 측에서 어떤 알고리즘을 사용 할지 선택하게 하는 방안도 고려하는 것이 좋아보였습니다.

3.2.5 추가해야 할 데이터

넓은 범위는 찾기도 힘들고 정확성이 떨어지므로 ‘울산’ 지역의 공공/범죄 데이터만 수집하여 테스트 하도록 하였습니다.

그러기 위해서 범죄 데이터 최소 100개, 공공 데이터 최소 1000개 모으고, Domain Awareness System(대테러 감지 시스템) 구축 시 지능형 CCTV, 총소리, 방사능 탐지센서, 자동차 번호판 인식장치, CCTV 감지망/센서 등을 이용하여 살인사건 75%를 줄였기에 공공데이터 포털에서 CCTV, 감지망/센서 등의 정보도 추가하도록 하였습니다.

3.3 데이터 마이닝

3.3.1 환경 데이터

데이터의 신뢰도를 위해 모든 환경데이터는 공공데이터 포털¹³⁾에서 구하기로 하였으며, 전 지역의 데이터를 이용하면 계산도 어렵고, 구현에 시간이 많이 들기 때문에 ‘울산’에만 한정시키기로 하였습니다.

사이트에 공개된 모든 csv 파일 데이터는 각기 다른 양식을 가졌기에 프로그램에서 처리 가능하도록 공통 데이터 양식이 필요하였습니다. 때문에 파일 명을 하나의 카테고리로 취급하고, [엘리먼트][시작 시간][종료 시간][개수][시작 위도][시작 경도][종료 위도][종료 경도] 컬럼을 공통되게 가지도록 하되 이외의 추가 데이터는 TEXT, INTEGER, BOOLEAN 등의 데이터 타입을 기입한 채 공통 데이터 뒤에 올 수 있도록 하였습니다.

13) <https://www.data.go.kr/>

A	B	C	D	E	F	G	H	I	J	K
1 열리면트	시작 시간	종료 시간	개수	시작 위도	시작 경도	종료 위도	종료 경도	TEXT::관리	TEXT::기관전화번호	
2 건축물	2017-01-01 0:00	2019-01-01 0:00	1	35.56334	129.3439	35.56334	129.3439	울산광역시052-290-3816		
3 건축물	2017-01-01 0:00	2019-01-01 0:00	1	35.57167	129.35	35.57167	129.35	울산광역시052-290-3816		
4 건축물	2017-01-01 0:00	2019-01-01 0:00	1	35.56972	129.3463	35.56972	129.3463	울산광역시052-290-3816		
5 건축물	2017-01-01 0:00	2019-01-01 0:00	1	35.56973	129.3464	35.56973	129.3464	울산광역시052-290-3816		
6 건축물	2017-01-01 0:00	2019-01-01 0:00	1	35.57006	129.3466	35.57006	129.3466	울산광역시052-290-3816		
7 건축물	2017-01-01 0:00	2019-01-01 0:00	1	35.5698	129.3448	35.5698	129.3448	울산광역시052-290-3816		

[그림 3-6] 필자가 지정한 CSV 파일 데이터 양식.

데이터를 이용하여 CCTV, 교통 단속 카메라 등 카메라 장치의 유무, 조명의 유무, 비상벨 유무, 경찰관서까지의 거리 등 치안 요소를 통해 범죄 발생 원인을 알아보고, 공중화장실, 어린이집, 도시공원 등 장소에 따른 범죄 발생 원인을 알아보기로 하였기에 울산 지역 CCTV 위치 3,958건, 경찰관서 위치 36건, 공중 화장실 위치 366건, 도시 공원 위치 466건, 무료 와이파이 위치 412건, 무인 교통 단속 카메라 위치 49건, 보안등 위치 22,570건, 안전 비상벨 위치 353건, 어린이 집 위치 740건, 총 28,950건의 환경 데이터를 우선 수집하였습니다.

CCTV.csv	2020-04-23 오후 12:35	Microsoft Excel 워크북	555KB
경찰관서.csv	2020-04-24 오후 12:02	Microsoft Excel 워크북	5KB
공중화장실.csv	2020-04-23 오후 12:35	Microsoft Excel 워크북	56KB
도시공원정보.csv	2020-04-23 오후 12:34	Microsoft Excel 워크북	79KB
무료와이파이.csv	2020-04-24 오후 12:04	Microsoft Excel 워크북	57KB
무인교통단속카메라.csv	2020-04-23 오후 12:35	Microsoft Excel 워크북	7KB
방화.csv	2020-04-24 오후 12:37	Microsoft Excel 워크북	1KB
보안등.csv	2020-04-23 오후 8:50	Microsoft Excel 워크북	2,740KB
안전비상벨위치.csv	2020-04-24 오후 12:05	Microsoft Excel 워크북	49KB
어린이집.csv	2020-04-24 오전 11:27	Microsoft Excel 워크북	110KB

[그림 3-7] 수집된 환경 데이터

와이파이 위치 데이터는 범죄와 네트워크간의 상관관계가 있을지 궁금하여 실험적으로 수집 해 보았으며, 프로그램을 제작하는 동안 데이터를 더 모아 3만 건 이상의 데이터를 사용하기로 하였습니다.

3.3.2 범죄 데이터

필요한 범죄 데이터는 ‘사건 종류’, ‘사건 발생 시작’, ‘사건 종료 시작’, ‘사건 발생 위치(위/경도)’ 정보를 가지고 있어야 하며,

공공데이터포털을 뒤져보았으나, 연도별 살인/폭력 등 발생 횟수 등 위치 정보, 발생 시각을 제외한 단순 통계 데이터만 나왔습니다.

범죄 발생 위치 데이터 신청
요청자 익명 제공기관 행정안전부 조회수 45 등록일 2020-04-15
범죄가 일어난 장소의 위치정보(좌표 형태)를 api의 형태로(힘들다면 파일 데이터라도)얻고 싶습니다. 생활안전지도 사이트를 방문하였으나, 이 곳에서는 데이터를 png형태로 제공하여 활용에 어려움이 많습니다. 단순히 발생한 시와 구를 알려주는 데이터가 아니라, 범죄가 발생한 바로 그 장소의 데이터가 필요합니다.
답변
답변자명 공공데이터활용지원센터 답변일 2020-04-16
안녕하세요. 공공데이터활용지원센터입니다. 공공데이터포털은 공공기관이 생성 또는 취득하여 관리하고 있는 공공데이터를 한 곳에서 제공하는 통합 창구이며, 데이터 1번가 또한 대국민이 필요로 하는 공공데이터에 대한 의견을 자유롭게 나눌 수 있는 데이터 소통창구이므로 데이터를 직접 제공해드릴 수 없는 점 양해 부탁드립니다. 해당 정보는 공공데이터포털에 개방출인 행정안전부의 '생활안전정보'(https://www.data.go.kr/dataset/15004715/openapi.do)에서 해당 정보를 찾아보실 수 있습니다. 추가적인 내용의 데이터를 원하신다면 공식적인 절차인 공공데이터 '제공신청'을 해주시면 담당 기관으로부터 더욱 정확한 답변을 받으실 수 있습니다. 그 외 기타 문의사항이 있으신 경우 공공데이터포털 대표번호 (1566-0025)로 문의하여 주시기 바랍니다.

[그림 3-8] 공공데이터 포털은 범죄 세부 데이터를 다루지 않는 모습

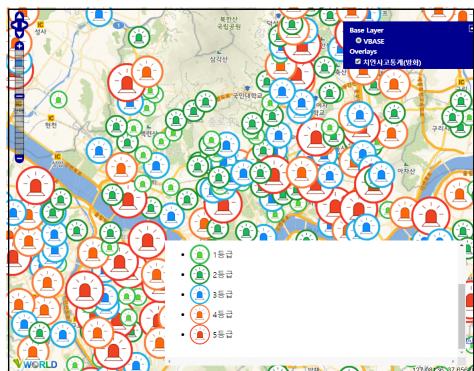
정부 24의 경찰사건 조회 서비스를 이용하려 하였으나, 조회 가능 대상자는 사건 당사자만 열람 가능하였기에 불가능하였습니다.

경찰 사건조회
최종수정일 2020.04.20 소관기관 법무부 기획조정실 정책기획관 협사사법금융시스템운영단
경찰에 입건된 사건에 대한 사건 진행 상황을 조회할 수 있는 서비스입니다.
 어떤 내용인지 궁금하시죠?
지원형태 정보제공
지원내용 <input checked="" type="radio"/> 경찰 사건 조회 - 경찰에 입건된 사건을 검색하여 진행 상황을 조회할 수 있습니다.
※ 피해자는 살인, 강도, 성범죄, 방화, 중상해 등 5개 중요범죄의 피해자 중 경찰, 검찰에서 조사받는 과정에서 서면으로 정보제공 동의를 하신 분만 조회 가능
※ 아래의 경우는 검색하실 수 없습니다. - 진정, 내사 사건 - 조회 가능 대상자가 아닌 경우 - 개인인증 로그인을 하지 않은 경우

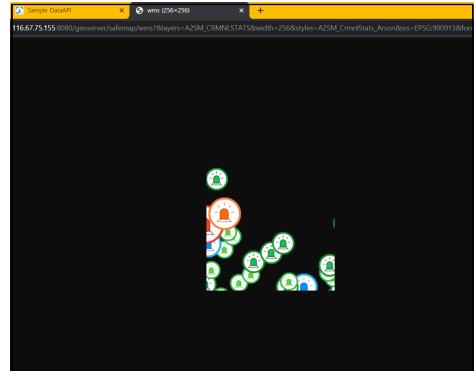
[그림 3-9] 본인(고소인/피고소인/피해자) 사건이 아니면 열람 불가한 모습

생활안전지도 API를 이용하여 지도상에 표시되는 범죄 데이터

정보를 모으기로 하였으나, [위도, 경도, 발생 횟수] 데이터를 이용하여 지도상에 핀(POI)을 꽂아 표시하는 방식이 아니라 지도 위에 이미지(레이어)를 추가 시키는 방법이었기에 사용할 수 없었습니다.

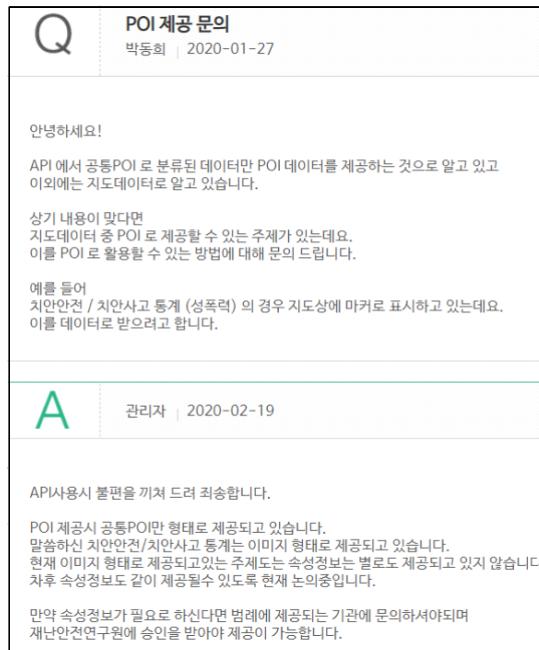


[그림 3-10] 생활 안전지도



[그림 3-11] 지도를 구성하는 레이어

이러한 처리 방식에 대하여 QnA 질문 카테고리에 글을 남기려 했으나, 이미 비슷한 경험을 한 사람이 글을 남긴 것을 발견하였습니다. 질문자 또한 이미지 레이어로 제공하는 데이터 이외 범죄 발생 위도, 경도, 횟수가 포함된 데이터 파일을 원하였지만, 답변에는 해당 사항은 경찰청에 연락 해 보라는 내용이 적혀 있습니다.



[그림 3-12] 직접 경찰청에 전화하라는 답변

때문에 ‘실제 범죄 데이터’를 분석하여 범죄율을 높이는 환경요소를 찾는 실험 대신 범죄가 일어날 환경을 미리 구성 해 두고, 임의 범죄 데이터를 주입하여 프로그램이 범죄 발생 환경 요소를 얼마나 잘 찾아내는지 실험을 하기로 테스트 계획을 바꾸었습니다.

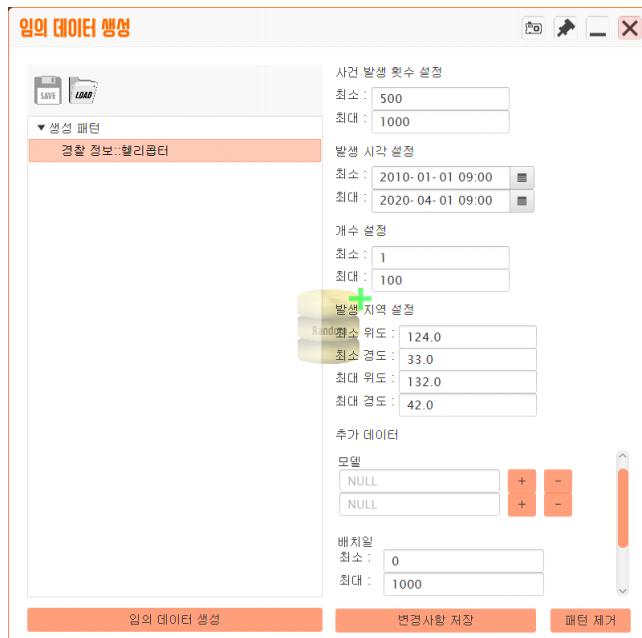
예) 어두운 조명 근처에 살인 사건을 몰아서 생성. 공중화장실 근처에 성폭행 사건을 몰아서 생성. CCTV와 가까워질수록 범죄 데이터 50% 씩 제거. 이후 프로그램이 ‘살인사건은 어두운 조명에서 자주 발생하며, 성폭행 사건은 공중화장실 근처에서 잘 발생하고, CCTV가 범죄율을 줄인다.’라는 결과를 도출하는지 알아보기.

3.4 기능 구현

3.4.1 임의 데이터 생성

위에서 했던 고민과 조언들을 토대로 DB 구조 및 제작 계획 수정을 마친 후, 임의 범죄 데이터 생성의 필요성을 느꼈기에 데이터의 틀, ‘모델’을 기반으로 임의 데이터를 생성 할 수 있는 기능을 구현하였습니다.

니다. 임의 데이터는 사용자가 지정한 ‘생성 패턴’에 맞게 생성되며, 사용된 패턴을 저장하거나 불러 올 수 있도록 하여 편의성을 더하였습니다.



[그림 3-13] 임의 데이터 생성 GUI

이외에도 공통 데이터인 시간, 위도, 경도, 발생 횟수뿐만 아니라 카테고리마다 상이한 추가데이터 까지 모두 임의 데이터 생성 패턴을 만들 수 있도록 하여 동일 데이터가 생성될 가능성을 낮추었습니다.

완성된 임의 데이터 생성 기능을 이용하여 10,000(경찰정보 2천, 범죄정보 8천)건의 데이터를 생성하는데 총 4분이 걸리는 것을 확인하였으며, 추후 속도를 향상시키기로 하였습니다.

하지만 임의 데이터 생성 패턴 파일을 불러올 때, 추가되어 있지 않은 모델이 존재 할 경우, 데이터 생성이 제대로 되지 않던 오류를 발견하였습니다. 이는 패턴 파일에 존재하지 않는 ‘추가 데이터’ 타입이 현재 개설된 DB에 존재할 경우와, 패턴 파일에는 존재하는데 현재 개설된 DB에는 존재하지 않을 경우 2가지였습니다.

때문에 해당 오류 사항은 패턴 파일 로딩 시 자체적으로 현재 개설된 DB와 동기화 시키는 코드를 작성함으로 써 해결하였습니다.

3.4.2 데이터 모델 변경

앞서 2장에서 PredPol, 보도자료, 생활안전지도 등 현재 서비스 중인 프로그램과 비교 했을 때, 필자의 데이터 처리 방식에 수정이 필요한 사항이 존재하는 것을 발견 하였습니다.

때문에 [발생 시각, 위도, 경도]만 공통 데이터로 지정하던 '시작'과 '끝'이 모호한 기존 방식 대신, [시작 시각] [종료 시각] [시작 위도] [시작 경도] [종료 위도] [종료 경도] 형태의 DB로 재구축 하였습니다.

num	main	sub	time	amount	latitude	longitude			
1	경찰 정보	교통 단속 카메라	924874522	2	35.1800496742018	129.076831926163			
2	경찰 정보	교통 단속 카메라	924874522	3	35.1807321425881	129.07223934113			
num	main	sub	startTime	endTime	amount	startLatitude	startLongitude	endLatitude	endLongitude
1	경찰 정보	교통 단속 카메라	1999-04-23 13:35:22	2020-04-23 13:35:22	2	35.1800496742018	129.076831926163	35.1800496742018	129.076831926163
2	경찰 정보	교통 단속 카메라	1999-04-23 13:35:22	2020-04-23 13:35:22	3	35.1807321425881	129.07223934113	35.1807321425881	129.07223934113
3	경찰 정보	교통 단속 카메라	1999-04-23 13:35:22	2020-04-23 13:35:22	1	35.1789930611852	129.074335474331	35.1789930611852	129.074335474331
4	경찰 정보	CCTV	1999-04-23 13:35:22	2020-04-23 13:35:22	11	35.1789930611852	129.074335474331	35.1789930611852	129.074335474331
5	경찰 정보	CCTV	1999-04-23 13:35:22	2020-04-23 13:35:22	3	35.1789930611852	129.074335474331	35.1789930611852	129.074335474331
6	경찰 정보	CCTV	1999-04-23 13:35:22	2020-04-23 13:35:22	7	35.1789930611852	129.074335474331	35.1789930611852	129.074335474331
7	경찰 정보	순찰 차량	1999-04-23 13:35:22	2020-04-23 13:35:22	8	35.1789930611852	129.074335474331	35.1789930611852	129.074335474331
8	경찰 정보	순찰 차량	1999-04-23 13:35:22	2020-04-23 13:35:22	2	35.1789930611852	129.074335474331	35.1789930611852	129.074335474331

[그림 3-14] 위 - 기존 DB / 아래 - 변경 후 DB

또한 [그림 3-14]와 같이 기존에는 시간 데이터를 Epoch 값으로 저장하여 INTEGER(LONG) 자료형을 사용하였었지만, 가독성을 증가시키기 위해 DB 재설계시 TEXT 형으로 교체하였습니다.

3.4.3 CSV 파일 지원

프로그램이 완성될 경우, 다양한 데이터가 사용 될 가능성이 높았으며, 이는 CSV 형식의 파일 데이터를 넣고 싶어 할 사용자가 생길 수 있다는 것을 의미하였습니다. 때문에 CSV 파일로 데이터 입/출력이 가능하도록 부가 기능을 추가하였습니다.

엘리먼트	시작 시간	종료 시간	개수	시작 위도	시작 경도	종료 위도	종료 경도	TEXT::모델	LONG::배가	TEXT::업무
CCTV	1999-04-23 13:35	1999-04-23 13:35	11	35.17899	129.0743	35.17899	129.0743	EOC4A8F6	1.26E+09	NULL
CCTV	1999-04-23 13:35	1999-04-23 13:35	3	35.17899	129.0743	35.17899	129.0743	T0442	1.26E+09	NULL
CCTV	1999-04-23 13:35	1999-04-23 13:35	7	35.17899	129.0743	35.17899	129.0743	A7104	1.26E+09	NULL
경찰 인력	1997-04-23 13:35	1997-04-23 13:35	32	35.17899	129.0743	35.17899	129.0743	NULL	1.56E+09	생활안전
경찰 인력	1997-04-23 13:35	1997-04-23 13:35	42	35.17899	129.0743	35.17899	129.0743	NULL	1.56E+09	경비
경찰 인력	1997-04-23 13:35	1997-04-23 13:35	52	35.17899	129.0743	35.17899	129.0743	NULL	1.56E+09	수사
경찰 인력	1997-04-23 13:35	1997-04-23 13:35	52	35.17899	129.0743	35.17899	129.0743	NULL	1.56E+09	행사
경찰 인력	1997-04-23 13:35	1997-04-23 13:35	12	35.17899	129.0743	35.17899	129.0743	NULL	1.56E+09	사이버안전
경찰 인력	1997-04-23 13:35	1997-04-23 13:35	42	35.17899	129.0743	35.17899	129.0743	NULL	1.56E+09	교통
교통 단속	1999-04-23 13:35	1999-04-23 13:35	2	35.18005	129.0768	35.18005	129.0768	CVT-0487f	1.26E+09	NULL
교통 단속	1999-04-23 13:35	1999-04-23 13:35	3	35.18073	129.0722	35.18073	129.0722	CVT-0487f	1.26E+09	NULL

[그림 3-15] CSV 파일로 출력된 데이터 모습

이 기능을 통해 공공데이터가 저장된 10개의 CSV파일 속 28,950건의 데이터를 읽어 들여 DB에 등록하는데 14분 5초가 소요되었습니다.

제 4 장 데이터 연산식 개발

4.1 기능 구현

4.1.1 데이터 처리 속도 향상

제 3장에서 나오듯, 28,950건의 데이터를 읽어 들이는데 14분 5초가 소요되는 것은 솔직히 옳지 못하였기에, 기존 개별 INSERT 방식을 Batch를 이용하여 그룹 INSERT 방식으로 변경하여 동일 조건에서 1초 만에 완료되도록 하였습니다.

더하여 경찰 정보 2천 건, 범죄정보 8천 건, 총 1만 건의 데이터를 생성하는데 4분이 걸리던 문제도 1초 안에 생성되도록 하였습니다.

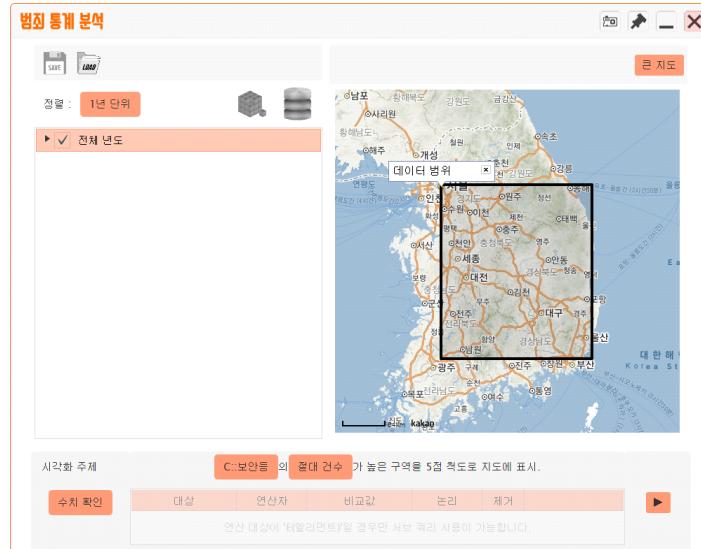
4.1.2 Kakao map API

GET/POST를 이용한 웹 페이지 명령 전달 체계는 개발 속도를 늦추는 바람에, 이를 폐기하고, WebView의 WebEngine을 이용하여 html속 내장된 javascript를 바로 실행시켜 사용하기로 하였습니다.

```
map.getEngine().executeScript("drawRectangle(" + range.get("startLatitude") + ", " + range.get("endLongitude") + ", 4, '#0000FF');  
map.getEngine().executeScript("createInfoWindow(" + range.get("startLatitude") + ", " + range.get("endLongitude") + ", 150, 100);  
map.getEngine().executeScript("infoWindow.open());");
```

[그림 4-1] html 속 javascript를 실행시키는 모습

이후 HTML 문서 안에 데이터 영역 표시를 위한 사각형, 원, 선, 다각형 그리기 함수를 넣고, 메인 화면에서 프로그램에 주입 시킨 데이터의 분포 범위를 표시 해 보았습니다.



[그림 4-2] 데이터 분포 범위를 표시 한 모습

분포 범위를 표기하고, 다시 지우려 했으나, 함수가 제공되어 있지 않아 marker.setMap(map); 구문을 통해 지도에 올라가는 점에 착안하여 obj.setMap(null);을 하였더니 성공적으로 표식이 사라졌으며, 이를 clear 함수로 묶어 사용하기로 하였습니다.

```
markers = [];

function clear() {
    var count = 0;
    for (; count < markers.length; count++) {
        markers[count].setMap(null);
    }
    markers = [];
}
```

[그림 4-3] 모든 마커를 제거하는 함수

데이터 분포 범위를 시간 단위로 구하기 위해, SQLITE에서 지원하는 %Y, %m, %w, %d 등의 플레이스홀더와 strftime()함수를 활용하여, 특정 시간대의 데이터 분포 범위를 구하는 함수를 만들었습니다.

```

if(timeEnum == TimeEnum.YEAR) {
    sql.append("startTime >= '" + startTime + "' AND endTime < '" + endTime + "'");
} else {
    // 시작 및 종료 일자가 일치 할 경우
    sql.append("(" + strftime("'" + timeEnum.symbol + "'", startTime) + ">'" + startTime + "' AND " + strftime("'" + timeEnum.symbol + "'", endTime) + "<'" + endTime + "'");
    sql.append("OR ");
    sql.append("(" + strftime("'" + timeEnum.symbol + "'", endTime) + ">'" + startTime + "' AND " + strftime("'" + timeEnum.symbol + "'", startTime) + "<'" + endTime + "'");
    sql.append("OR ");
}
if(timeEnum == TimeEnum.MONTH) {
    // 시작 연도와 종료 연도가 1년 이상 차이 날 경우
    sql.append("((" + strftime("%Y", endTime) - strftime("%Y", startTime)) > 0)");
} else if(timeEnum == TimeEnum.HOUR) {
    // 시작 일자와 종료 일자가 1일 이상 차이 날 경우
    sql.append("((" + strftime("%s", endTime) - strftime("%s", startTime)) > 86400)");
} else if(timeEnum == TimeEnum.WEEK) {
    // 시작 일자와 종료 일자가 1 주일 이상 차이 날 경우
    sql.append("((" + strftime("%s", endTime) - strftime("%s", startTime)) > 604800)");
}
}

```

[그림 4-4] 시간 범위를 구하는 Query문

하지만 쿼리문이 너무 긴 것 같아 새로운 방법을 고안하던 중, 기존 방식이 시작 시간과 종료 시간을 경우의 수 만큼 판별하여 계산 속도가 떨어졌다면, 새로운 방식은 약 2개의 질의 블록만 사용하면 모든 경우의 수를 맞출 수 있도록 설계하였습니다.

```

// 시작 및 종료 일자가 일치 할 경우
sql.append("((" + strftime("%Y", startTime) + "<=" + endTime + " AND " + strftime("%Y", endTime) + ">=" + startTime + "))");
if(timeEnum != TimeEnum.YEAR) {
    sql.append("OR ");
    if(timeEnum == TimeEnum.MONTH) {
        // 시작 연도와 종료 연도가 1년 이상 차이 날 경우
        sql.append("((" + strftime("%Y", endTime) - strftime("%Y", startTime)) > 0)");
    } else if(timeEnum == TimeEnum.HOUR) {
        // 시작 일자와 종료 일자가 1일 이상 차이 날 경우
        sql.append("((" + strftime("%s", endTime) - strftime("%s", startTime)) > 86400)");
    } else if(timeEnum == TimeEnum.WEEK) {
        // 시작 일자와 종료 일자가 1 주일 이상 차이 날 경우
        sql.append("((" + strftime("%s", endTime) - strftime("%s", startTime)) > 604800)");
    }
}

```

[그림 4-5] 훨씬 짧아진 Query문

쿼리문은 짧아졌지만 특정 시간대의 데이터 범위 표기 대상을 사용자가 원하는 만큼 짚을 수 있는 노릇이었기에, 1건만 선택할 수도, 1000건을 선택할 수도 있었습니다. 때문에 스레드 사용에 앞서, 작업 큐의 크기에 따라 스레드 수를 유동적으로 설정하도록 하여 적은 작업일 경우, 적은 스레드로, 많은 작업의 경우, 많은 스레드를 이용하여 계산하도록 하였습니다.

```

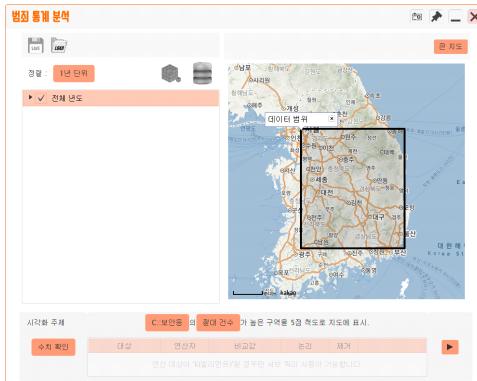
int querySize = DataRangeCalculateThread.queryList.size();
if(querySize > 1000) {
    threadAmount = querySize/18;
} else if(querySize > 100) {
    threadAmount = querySize/20;
} else if(querySize > 50) {
    threadAmount = 3;
} else if(querySize > 10) {
    threadAmount = 2;
}

```

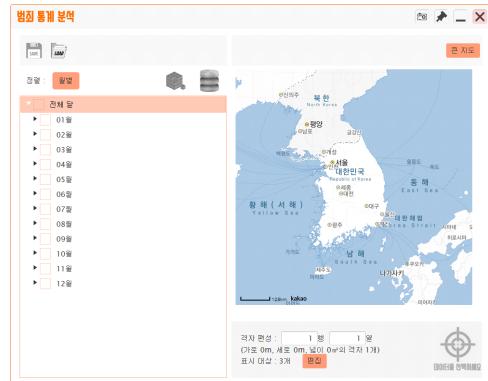
[그림 4-6] 사용될 스레드 수 제어

4.1.3 통계 분석화면 설정 및 시각화 대상 선정 GUI 추가

기존 통계 분석화면 디자인의 시각화 대상을 선정하는 영역이 좁고 보기 좋지 못하여 아예 시각화 대상을 선정하는 영역을 새 창으로 분리시키고, 각 Pane간 배치를 넓게 하였습니다.



[그림 4-7] 기존 Main 화면



[그림 4-8] 수정 이후 Main 화면

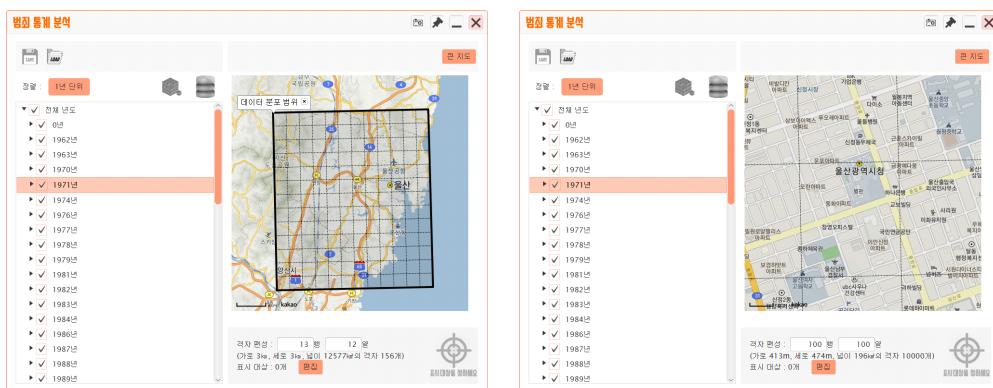
덕분에 검색 버튼을 꾸미고, 데이터 범위를 대상으로 생성 시킬 격자 개수를 지정할 수 있는 옵션을 추가시킬 수 있었습니다.



[그림 4-9] 시각화 대상 설정 GUI

4.1.4 격자 크기 설정

메인 디자인을 변경한 덕분에 데이터 분포 격자를 나눌 수 있는 영역이 생겼습니다. 이를 통해 행과 열을 입력 할 경우, 시작 위도/시작 경도/종료 위도/종료 경도를 실제 m 단위로 계산하여 가로, 세로, 넓이를 이용자에게 보여줄 수 있게 되어 편리성이 향상되었습니다.



[그림 4-10] 격자 생성 및 줌 인/아웃

하지만 격자가 촘촘해 질수록 랜더링 시간이 오래 걸렸습니다. 이는 불행히도 Kakao Map API가 담당하는 영역이기에 필자가 어찌 수정할 방법이 없었습니다. 때문에 추후 격자를 구성하는 ‘선’ 오버레이를 API에서 제공해 주는 다른 오버레이 중, 대체 가능한 것이 있는지 찾아보기로 하였습니다.

4.2 통계 알고리즘 구현

4.2.1 통계 알고리즘 구상

통계에서 가장 중요한 키워드는 ‘신뢰성’이기에 이미 비슷한 실험을 한 연구 자료나 서비스의 통계 절차 및 알고리즘을 모방하는 것이 좋아 보였습니다.

PredPol에서는 시작/종료 시간이 72 시간을 초과하는 범죄 데이터는 예측 정확도가 저하되므로 연산 시 제외¹⁴⁾시키는 방안을 사용 중이며, 보도자료¹⁵⁾에서는 범죄 데이터의 단순 수치(건수) 기반으로 초기 모델을 생성한 다음, 모든 데이터를 이용하여 환경 모델을 구성하고, 생성된 두 모델의 통계를 다시 내어 결과를 얻고 있었습니다.



[그림 4-11] 보도 자료에 게시된 시각화 프로세스

14) <https://www.predpol.com/technology>

15) <http://www.korea.kr/common/download.do?fileId=189406805&tblKey=GMN>

하지만 어떠한 연구자료/서비스에서도 자신만의 ‘모델 설계’ 알고리즘을 공개하고 있지 않았기 때문에, 모델 설계는 직접 할 수밖에 없어보였습니다.

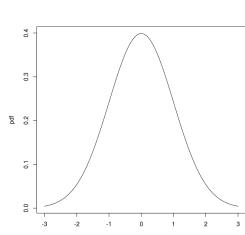
때문에 오규철 교수님의 “범죄정보과학의 이해” 강의를 듣게 되어, 수사대가 사용 중인 [1. 데이터 분석 (관계망 형성) 2. 지리적 분석 3. 확률모델 생성 절차를 통해 분석.] 절차를 인용하여 알고리즘을 생각해 보기로 하였습니다.

[사용자 조기 설정 단계]

- ① 범죄 데이터 및 환경 데이터를 프로그램에 입력합니다.
- ② 통계에 사용될 데이터 범위를 좌측 TreeView에서 선택합니다.
- ③ 격자 행 열 지정을 통해 범위를 설정합니다.
- ④ 시각화 대상을 선정합니다. (예 : 18세 이상 남성을 대상으로 한 살인 범죄)

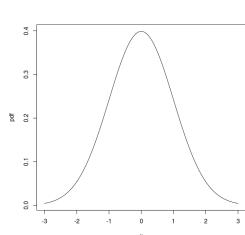
[분석 단계 - 데이터 분석(관계망 형성)]

⑤ 시각화 대상인 살인 범죄가 발생한 년도의 범죄/환경 데이터, 발생한 시각의 범죄/환경 데이터, 발생한 요일의 범죄/환경 데이터를 구하여 각각 어느 시간대에서 가장 높은 발생률을 보이는지 순위를 매깁니다. 이 때, 요일 수가 7이므로 순위는 7위 까지만 매기도록 코드에 안전장치를 겁니다.



순위	년	월	시	요일
1	2018	04	01	화
2	2016	06	00	일
3	2019	05	03	금
4	2020	07	22	목
5	2005	09	20	토
...				

[▲ 살인 범죄 발생 시각 표준 정규 분포 및 순위 표]



순위	년	월	시	요일
1	2016	05	01	월
2	2017	04	03	화
3	2010	08	00	일
4	2019	06	12	수
5	2018	10	13	토
...				

[▲ 강간 범죄 발생 시각 정규 분포 및 순위 표]

년도 차	월 차	시차	요일차	평균오차
2018-2016 = 2	04-05 = 1	01-01 = 0	화-월 = 1	1
2016-2017 = 1	06-04 = 2	03-00 = 3	화-일 = 5	2.75
2019-2010 = 9	05-08 = 3	03-00 = 3	금-일 = 2	4.25
2020-2019 = 1	07-06 = 1	22-12 = 10	목-수 = 1	3.25
2005-2018 = 13	09-10 = 1	20-13 = 7	토-토 = 0	5.25
오차 평균			3.3	

[그림 4-12] 시간별 순위 매김 과정

매겨진 오차 값은 추후 범죄 발생 예측에 사용되므로, 오차 평균값이 낮을수록 특정 범죄 발생 확률 상승에 높은 영향력을 발휘하는 변수가 됩니다.

[분석 단계 - 지리적 분석]

- ⑥ 격자 내에서 시각화 대상인 살인 범죄가 발생한 위치를 기준으로 연도별, 월별, 요일별, 시간별 범죄/환경 데이터의 누적 분포 데이터를 구합니다.
- ⑦ 격자 내에서 시각화 대상인 살인 범죄가 발생한 연도의 범죄/환경 데이터, 발생한 시각의 범죄/환경 데이터, 발생한 요일의 범죄/환경 데이터를 구하고, 가장 많이 발생한 순으로 나열합니다.
- ⑧ 모든 격자로 부터 시각화 대상인 살인 범죄가 발생한 횟수를 연도별, 월별, 요일별, 시간별로 나열합니다.

순위	년	평균 거리
1	2018	3.1 km
2	2012	4.5 km
3	2019	4.6 km
4	2020	5.8 km
5	2005	9.2 km
평균		5.4 km

[▲ 섹터 A1 구역 이내의 연도 별 살인 범죄에 대한 강간 범죄 발생 거리 분석 표]

순위	년	살인	강간	오차
1	2016	43회	32회	11
2	2017	42회	48회	6
3	2010	24회	34회	10
4	2019	33회	43회	10
5	2018	29회	30회	1
평균			7.6	

[▲ 섹터 A1 구역 이내의 살인 범죄 발생에 대한 강간 범죄 발생 횟수 비교 표]

순위	격자	발생 건수
1	A8	98회
2	A6	78회
3	U3	64회
4	O3	58회
...		

[▲ 2018년도 살인 범죄가 발생한 격자 및 발생 절대 건수]

[그림 4-13] 격자별 순위매김 과정

[분석 단계 - 확률모델 생성 절차를 통해 분석]

⑨ 연도별 확률모델 :

9-1.

연도별 살인 발생 절대 건수가 가장 높은 순으로 제 1 계층을 생성합니다. 이 후 연도별 살인 발생 건수에 가장 높은 영향을 끼치는 변수를 절대 점수 $n(1 \div [\text{오차 평균}])$ 점에 발생 건수를 곱하여 2계층 데이터를 생성합니다.

순위	변수 강간	발생 건수 98회	절대 점수 0.303	위험 지수 29.694	상승치
2	절도	78회	0.271	21.138	
3	납치	64회	0.267	17.088	

[표 4-1] 위험 지수 계산 식

9-2.

절대 건수가 적힌 제 1계층 위에 환경변수 위험 지수를 적은 제 2계층을 더 합니다.

98	78	12	0
64	58	16	0
23	21	11	2
6	1	3	8

제 1 계층 (발생 절대건수)

42.1	34.6	10.1	5.2
51.2	32.9	9.7	14.3
16.2	10.8	11.0	2.4
1.8	2.4	2.2	3.1

환경 변수(위험 지수) 총 합

140.1	112.6	22.1	5.2
115.2	90.9	25.7	14.3
25.2	31.8	22.0	4.4
7.8	3.4	5.2	11.1

계층 합산

[그림 4-14] 계층 합산 방식

9-3.

[가장 높은 점수 - 가장 낮은 점수] \div 5 계산을 통해 5점 척도를 구하여 색을 칠합니다.

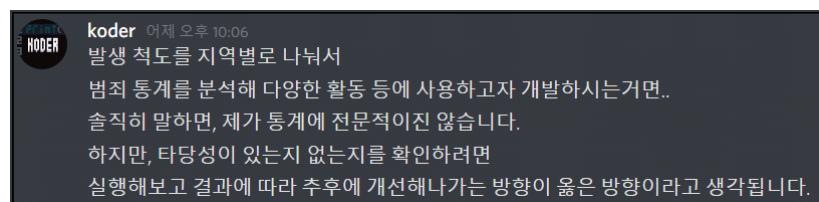
140.1	112.6	22.1	5.2
115.2	90.9	25.7	14.3
25.2	31.8	22.0	4.4
7.8	3.4	5.2	11.1

2018년도 5점 척도 지도 표시

[그림 4-15] 5점 척도 색상 표시

- ⑩ 월별 확률모델 : 연도별 확률모델과 동일한 과정을 거쳐 지도에 표시합니다.
- ⑪ 요일별 확률모델 : 연도별 확률모델과 동일한 과정을 거쳐 지도에 표시합니다.
- ⑫ 시간별 확률모델 : 연도별 확률모델과 동일한 과정을 거쳐 지도에 표시합니다.
- ⑬ 통합 확률모델 : 연도별, 월별, 요일별, 시간별 확률모델의 지도 결과를 구역별로 합산하여 5점 척도로 나누어 지도에 표시합니다.

위 알고리즘은 필자가 고안한 방식이기 때문에 신뢰성 및 타당성이 의심되어 교수님 및 각종 개발자 커뮤니티에 질문 게시글을 작성하여 의견을 물어보았습니다.



[그림 4-16] 다른 개발자분의 답변

4.2.2 통계 알고리즘 구현

- ① 환경변수 대상 절대건수 구하기

앞 장에서 설명한 통계 알고리즘대로 절대건수를 기반 한 통계 수

치를 지도에 표기하기 위해, 그 첫 번째 단계인 모든 환경변수 데이터의 시간별 절대 건수를 얻기로 하였으며, 계산이 다 끝나 반환된 절대 건수 값을 메모리에 저장 해 두었다가, 한 번에 INSERT 하는 알고리즘을 사용할 경우, 메모리가 적은 컴퓨터에서는 응답 없음 창이 뜰 수도 있기에, 계산이 끝난 함수는 즉시 DB에 INSERT 요청을 날리도록 하여 속도는 조금 양보하되 안전성과 메모리를 중시하기로 하였습니다.

num	category	element	timeType	targetTime	timeCount
1	보안등	한전주	MONTH	01	21799
30	보안등	한전주	HOUR	00	21799
33	보안등	한전주	WEEK	2	21799
1008	보안등	한전주	YEAR	2019	21799
31	보안등	한전주	WEEK	0	21349
32	보안등	한전주	WEEK	1	21349
34	보안등	한전주	WEEK	3	21349
35	보안등	한전주	WEEK	4	21349
36	보안등	한전주	WEEK	5	21349

[그림 4-17] DB에 기입된 절대 건수

하지만 연산 량이 많은 탓인지 계산하는 과정이 데이터 30건 기준 0.5초, 데이터 28,860건 기준 26초나 걸렸습니다.

② 시각화 대상 절대건수 구하기

앞 장에 언급 된 환경변수의 절대 건수를 구하는 코드를 기반으로 시각화 대상에 대한 절대건수를 구하기로 하였습니다.

시각화 대상 절대건수는 서브쿼리 뿐만 아니라 각 시각화 대상마다 부여되는 포인트(예 : 치안 점수)에 대한 연산까지 지원해야 하므로 다소 성가신 부분이 많았습니다.



[그림 4-18] 서브쿼리가 가능한 모습

서브 쿼리가 등록 된 시각화 대상의 경우, IN (SELECT ~) 문을 이용하여 서브쿼리 결과 값이 포함 된 num 값을 가진 데이터만 불러오도록 처리하였으며, 포인트 계산의 경우, 포인트 값이 0이 아닌 모든 엘리먼트를 구한 다음, 각 시간별 포인트 값의 총 합을 구하여 Map에 <시간:포인트값> 형태로 저장시켜 한 번에 INSERT 하기로 하였습니다.

```
sql.append("AND num IN (");
sql.append("SELECT target FROM 'data.ad");
if(element != null){
    sql.append("sub == '" + element + "' AND");
}
sql.append(subQuery);
sql.append(") "));
```

[그림 4-19] 서브쿼리 통합 쿼리

```

totalCount = MainController.dataDb.getAbsoluteCount(te, category,
    timeKey = te.name() + Main.SPLIT_SYMBOL+stringTimeCount;
    if(totalPointValue.containsKey(timeKey)){
        timeValue = totalPointValue.get(timeKey);
    } else {
        timeValue = 0;
    }
    timeValue += (totalCount * hasPointElement.get(key));
    totalPointValue.put(timeKey, timeValue);
}

```

[그림 4-20] 단계별 수집되는 포인트 값

num	category	element	subQuery	timeType	targetTime	timeCount
20	범죄 정보	살인	'가해자 성별' == '남' AND '가해자 연령' >= 42 AND '피해자 성별' == '남' OR '피해자 연령' == 1	MONTH	06	1.0
21	범죄 정보	살인	'가해자 성별' == '남' AND '가해자 연령' >= 42 AND '피해자 성별' == '남' OR '피해자 연령' == 1	WEEK	4	1.0
22	범죄 정보	살인	'가해자 성별' == '남' AND '가해자 연령' >= 42 AND '피해자 성별' == '남' OR '피해자 연령' == 1	HOUR	18	1.0
48	범죄 정보	살인	'가해자 성별' == '남' AND '가해자 연령' >= 42 AND '피해자 성별' == '남' OR '피해자 연령' == 1	YEAR	2015	1.0
1	치안 점수			MONTH	09	51.0
2	치안 점수			MONTH	11	51.0

[그림 4-21] 살인 ‘엘리먼트’ 뿐만 아니라 치안점수 ‘포인트’까지 등록 된 모습

③ 시각화대상 및 환경변수 간 절대건수 평균 오차 값 구하기

시각화 대상의 절대건수가 가장 많았던 연/월/요일/시를 구하여 순서대로 메모리에 올린 뒤, 각 환경변수의 절대건수가 가장 많았던 연/월/요일/시 값과 차를 구하여 평균을 내 보기로 하였습니다.

하지만 1997 ~ 2020 까지 발생한 이벤트만 100개를 다 채운 상태 일 경우, 가장 많이 발생 한 순서가 1997 - 1998 - 1999 - 2000 - 2001... 순으로 나오기 때문에 신뢰도와 정확도가 많이 떨어질 것 같았기에, GROUP BY를 이용하여 동일한 발생 건수를 가지는 ‘발생 시간’을 하나로 묶기로 하였습니다.

```

    ,
    sql.append("timeType == '"+timeType+"' GROUP BY timeCount ORDER BY timeCount DESC LIMIT 7;");
try {
    PreparedStatement pstmt = conn.prepareStatement(sql.toString());
    ResultSet rs = pstmt.executeQuery();
    int timeCount = 0;
    while (rs.next()) {
        timeCount = Integer.parseInt(rs.getString( columnLabel: "targetTime"));
        targetTimeList.add(timeCount);
    }
}

```

[그림 4-22] 신뢰도와 정확도 향상을 위한 ‘발생 시간’ GROUP BY

GROUP BY로 묶인 탓에 7순위 까지 나와야 할 데이터가 1개 혹은 2순위 까지만 나오기도 하였기에 ‘가장 적은 순위를 가진’ 통계를 기준으로 연산하도록 하였습니다.

```
ECo보안등Co한전주 : MONTH - [1, 12]
ECo보안등Co한전주 : YEAR - [2019, 2018, 2017, 2016, 2015, 2014, 2013]
ECo보안등Co한전주 : HOUR - [0, 24]
ECo보안등Co한전주 : WEEK - [2, 6]
```

[그림 4-23] 시간 순으로 나열한 모습. 2순위 까지 연산된 것이 눈에 띈다.

num	category	element	targetDataType	targetCategory	targetElement	yearCount	monthCount	hourCount	weekCount	totalAverage
17	범죄 정보	철도	E	경찰 정보	헬리콥터	6.0	9.0	16.0	0.0	7.75
20	범죄 정보	악취/유인	E	경찰 정보	헬리콥터	6.0	9.0	16.0	0.0	7.75
21	범죄 정보	방화	E	경찰 정보	헬리콥터	6.0	9.0	16.0	0.0	7.75
23	범죄 정보	마약	E	경찰 정보	헬리콥터	6.0	9.0	16.0	0.0	7.75
18	범죄 정보	폭력	E	경찰 정보	헬리콥터	5.0	10.0	12.0	1.0	7.0
19	범죄 정보	성폭력	E	경찰 정보	헬리콥터	5.0	10.0	12.0	1.0	7.0
22	범죄 정보	살인	E	경찰 정보	헬리콥터	5.0	6.0	6.0	2.0	4.75
25	범죄 정보	강도	E	경찰 정보	헬리콥터	4.0	3.0	4.0	3.0	3.5
28	지역 정보	논/밭	E	경찰 정보	헬리콥터	4.0	3.0	4.0	3.0	3.5
30	상점 정보	유홍	E	경찰 정보	헬리콥터	4.0	3.0	4.0	3.0	3.5
32	상점 정보	낮은 매출	E	경찰 정보	헬리콥터	4.0	3.0	4.0	3.0	3.5
33	교통 정보	사고 다발지역	E	경찰 정보	헬리콥터	4.0	3.0	4.0	3.0	3.5
1	경찰 정보	헬리콥터	E	경찰 정보	헬리콥터	0.0	0.0	0.0	0.0	0.0
2	경찰 정보	수찰 바이크	E	경찰 정보	헬리콥터	0.0	0.0	0.0	0.0	0.0

[그림 4-24] 모든 환경변수에 대한 시각화 대상의 시간별 절대건수 오차 값

오차 값을 구해 보니, totalAverage 값이 0인 데이터가 눈에 띠었습니다. totalAverage 낮을수록 시각화 대상 발생 요인에 가까운 데이터인데, 모델만 등록 되어 있고 데이터는 등록되어 있지 않는 경우나, 시각화 대상일 경우 값이 0으로 출력되었습니다.

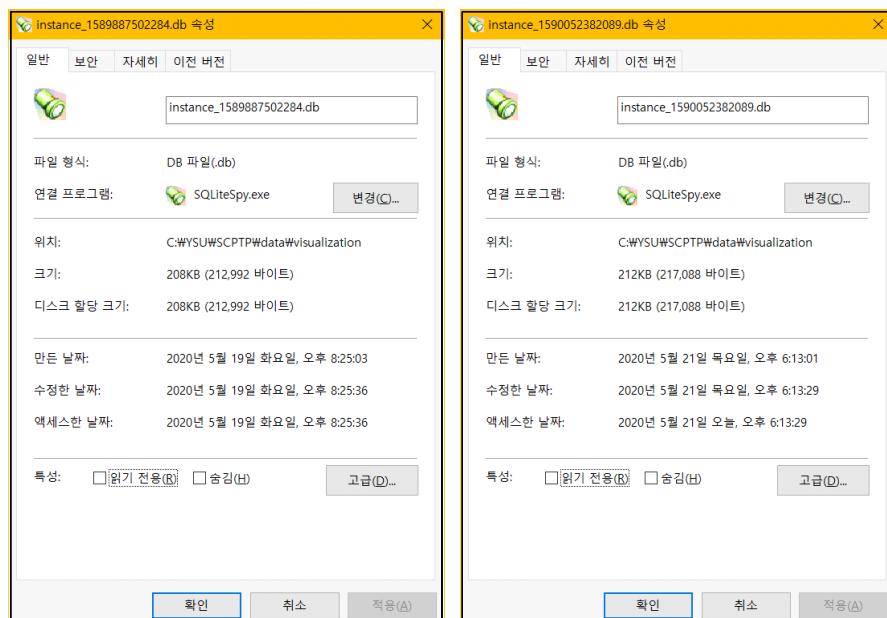
때문에 시각화 단계에서 totalAverage가 0인 항목은 연산하지 않기로 하였습니다.

④ 리팩토링

전체 환경변수, 시각화 대상에 대한 절대건수 계산이 완벽하게 이루어 졌으나, 곧 만들어야 할 섹터별 계산에서 비슷한 알고리즘이 섹터 단위로 반복될 것이 분명하였습니다.

때문에 지금까지 사용된 모든 스레드, 계산식, 전역 변수 등을 체계화 시켜 섹터별 환경 요소 계산 시에도 코드가 재사용될 수 있도록 리팩토링 하기로 하였으며, 우선 QueryObject를 만들어 계산 시 사용되는 모든 변수 값을 통합하였으며, 이후 스레드 간 공통 계산 요소를 줄여 나가면서 총 4개였던 연산 클래스를 2개까지 줄여나갔습니다.

이후 실행시킨 결과, 코드 재사용성이 향상되었을 뿐만 아니라, 실행 속도가 소폭 향상된 모습을 볼 수 있었습니다.



[그림 4-25] 데이터 28,860개 기준 리팩토링 전 : 33초/후 : 28초

⑤ 섹터별 시각화 대상의 시간대별 절대건수 구하기

시각화 대상의 절대건수를 구하던 함수를 복제한 뒤, 사용자가 지정한 행/열에 맞게 위/경도를 나누어 각 섹터별 발생 건수를 구하는 함수를 만들었습니다.

num	sector	data...	cate...	element	subQ...	timeT...	targe...	ti...	▲
495	8	C	보안들			YEAR	1997	1.0	
497	8	C	보안들			YEAR	1998	1.0	
498	8	C	보안들			YEAR	2000	1.0	
499	8	C	보안들			YEAR	1999	1.0	
501	8	C	보안들			YEAR	2001	1.0	
567	10	C	보안들			YEAR	2000	1.0	
625	11	C	보안들			YEAR	1970	1.0	
636	11	C	부와드			YEAR	1971	1.0	

[그림 4-26] 시각화대상의 섹터/시간별 발생 건수를 저장한 모습

리팩토링을 아무리 거쳐도 섹터 계산으로 넘어가니 연산 시간이 점점 늘어지는 모습이 보였으며, 데이터 28,860개, 25섹터 기준, 섹터별 시각화 대상 절대 건수 계산 까지 35초가 소요되었습니다.

⑥ 섹터별 환경변수의 시간대별 절대건수 구하기

모든 환경변수에 대한 모든 시간, 모든 섹터별 계산을 진행한 결과, 데이터 28,860개, 25섹터(5행 5열) 기준 14,237개의 섹터 데이터가 생성되었으며, 이 단계까지 총 6분 10초가 소요되었습니다.

아무리 연산 건수가 많다 해도 시간이 너무 오래 걸리는 듯해서 스레드 수를 변경 해 보기로 하였습니다.

기존 방침은 쿼리 건수가 많아질수록 스레드 수가 덩달아 늘어나는 형태였는데, NuriCHAIN¹⁶⁾ 개발 당시에도 스레드 수가 너무 적으면 컴퓨터 성능을 최대로 뽑아 쓰지 못하고, 스레드 수가 너무 많으면 단일 스레드보다 자원 효율이 좋지 못하여 적정 수치를 항상 유지해야했었기에 만들었던 적정 스레드 수를 지정하는 함수를 사용하기로 하였습니다.

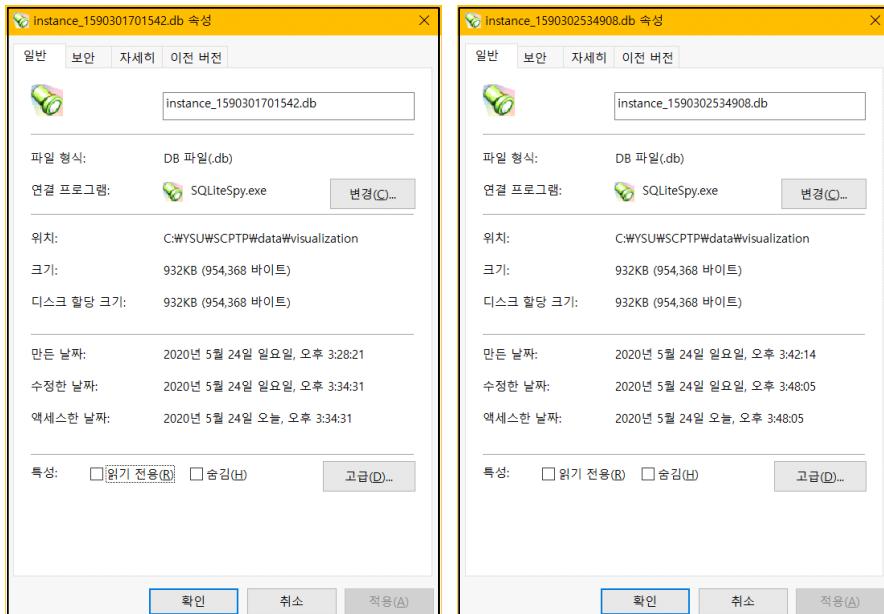
16) <https://cafe.naver.com/goldbigdragon/85753>

```

int processors = Runtime.getRuntime().availableProcessors();
maxThreads = (processors * processors) + (processors * 8);
if(maxThreads > 200)
    maxThreads = maxThreads/25;
else if(maxThreads > 100)
    maxThreads = maxThreads/15;
else if(maxThreads > 50)
    maxThreads = maxThreads/8;
else if(maxThreads > 20)
    maxThreads = maxThreads/4;
else
    maxThreads = 2;

```

[그림 4-27] PC 스펙에 적합한 스레드 수를 구하는 함수



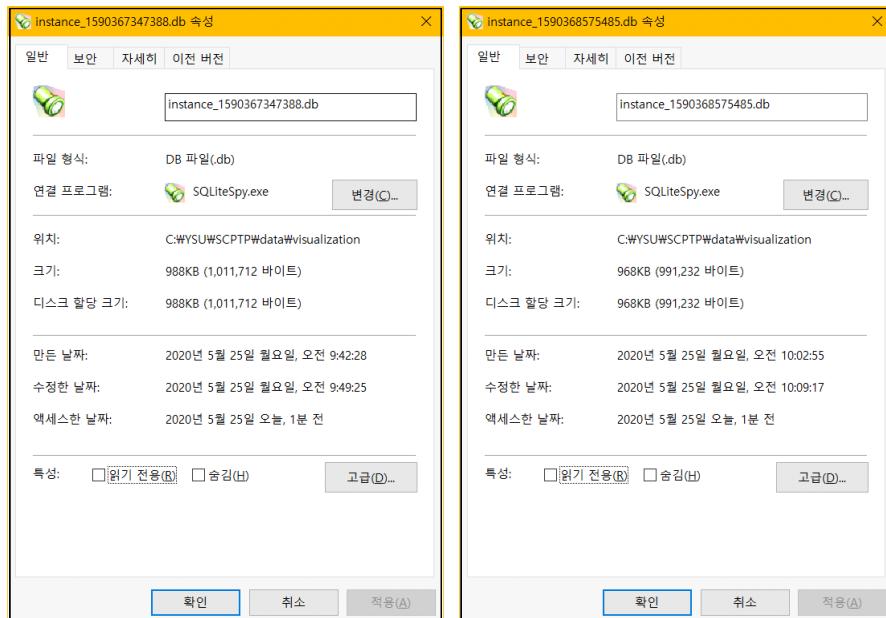
[그림 4-28] 데이터 28,860개 기준 수정 전 : 6분 10초/수정 후 : 5분 51초

약 20초가량 빨라진 모습을 보니 크롤링 용도로 설계된 스레드 수도출 공식이 일반 연산 식에도 효과가 있는 듯 보였습니다.

- ⑦ 섹터별 시각화대상 및 환경변수의 평균 오차 값 구하기
시각화 대상과 환경변수의 섹터별 발생 건수 평균 오차 값을 구하

기로 하고, 데이터 28,860개, 25섹터(5행 5열) 기준 평균 오차 값을 구했더니, 6분 57초가 걸렸습니다.

역시나 처리 시간이 마음에 들지 않아 스레드 수도 바꿔 보고 알고리즘도 변경하던 도중, 스레드 생성 위치를 전 처리가 아닌 후 처리로 변경하였더니 시간이 소폭 향상된 모습을 볼 수 있었습니다.



[그림 4-29] 데이터 28,860개 기준 (좌-기준 : 6분 57초) (우-수정 : 6분 22초)

⑧ 계산속도 향상

시각화 대상과 환경변수간 거리 계산만 남겨 둔 상태이므로, 막간을 이용해 계산 시간을 조금 더 줄여 보기로 했습니다.

우선 어느 구간에서 시간이 가장 오래 걸리는지 알아 볼 필요가 있기 때문에 각 과정마다 타임 로그를 찍어보았습니다.

```
[시각화] : 시각화 계산 시작
[SQLITE] : 94개의 AbsoluteTargetCountData가 추가됨. [연산 시작 2초 609% 이후]
[SQLITE] : 3186개의 AbsoluteCountData가 추가됨. [연산 시작 20초 335% 이후]
[SQLITE] : 47개의 AbsoluteDifferenceCountData가 추가됨. [연산 시작 72% 이후]
[SQLITE] : 1129개의 AbsoluteGeoTargetCountData가 추가됨. [연산 시작 12초 404% 이후]
[SQLITE] : 5001개의 AbsoluteGeoCountData가 추가됨. [연산 시작 2분 2초 이후]
[SQLITE] : 5001개의 AbsoluteGeoCountData가 추가됨. [연산 시작 3분 30초 이후]
[SQLITE] : 4235개의 AbsoluteGeoCountData가 추가됨. [연산 시작 6분 31초 이후]
[SQLITE] : 431개의 AbsoluteGeoDifferenceCountData가 추가됨. [연산 시작 6초 826% 이후]
[시각화] : 시각화 계산 완료 [연산 시작 7분 14초 이후]
```

[그림 4-30] 계산 타임로그

확인해 본 결과, 전체 7분 14초 중, 6분 31초(약 90%) 의 시간이 세터별 환경 변수 계산 (AbsoluteGeoCountData) 과정에서 소요되는 모습을 관찰 할 수 있었습니다.

때문에 알고리즘을 변형하고, 노파심에 설치 해 두었던 Null Exception을 대부분의 클래스에서 제거하는 대신, 계산 이후 DB 속이 null일 경우, 강제 중단 시키는 코드를 작성하였습니다.

이외 일부 데이터 처리 구문에는 하드코딩을 하여 시간을 살짝 단축시키기로 하였으며, 연관성이 높은 계산 식끼리 연달아 시작 되도록 계산 순서를 바꿔 주었습니다.

```
[시각화] : 시각화 계산 시작
[SQLITE] : 94개의 AbsoluteTargetCountData가 추가됨. [연산 시작 2초 506% 이후]
[SQLITE] : 3186개의 AbsoluteCountData가 추가됨. [연산 시작 17초 806% 이후]
[SQLITE] : 1129개의 AbsoluteGeoTargetCountData가 추가됨. [연산 시작 10초 893% 이후]
[SQLITE] : 5001개의 AbsoluteGeoCountData가 추가됨. [연산 시작 1분 35초 이후]
[SQLITE] : 5001개의 AbsoluteGeoCountData가 추가됨. [연산 시작 2분 38초 이후]
[SQLITE] : 4235개의 AbsoluteGeoCountData가 추가됨. [연산 시작 5분 5초 이후]
[SQLITE] : 47개의 AbsoluteDifferenceCountData가 추가됨. [연산 시작 62% 이후]
[SQLITE] : 501개의 AbsoluteGeoDifferenceCountData가 추가됨. [연산 시작 4초 556% 이후]
[시각화] : 시각화 계산 완료 [연산 시작 5분 42초 이후]
```

[그림 4-31] 알고리즘 변형 이후 타임로그

결과는 7분 14초에서 5분 42초로, 1분 32초가량 시간을 절약 할 수 있게 되었습니다.

하지만 5월 4주차 주간 보고서를 읽으신 교수님이 강력한 쿼리문을 놔두고 너무 프로그램 단에서만 계산을 시키고 있다고 지적하셨기에, 지금 까지 개발 되었던 대부분의 구문을 DB단에서 자체 처리

하도록 수정하기로 하였습니다.

하지만 DB단에서 연산 및 입력 처리를 진행 시키는 것은 SQLite가 파일 단위로 스키마가 나뉘기 때문에 현재 스키마 구성으로는 불가능하였습니다. 그래서 데이터가 담긴 data.db 스키마 파일을 복제하여 모든 연산이 끝나고 나면 데이터를 날리기로 하였습니다.

결과적으로 아래와 같이 SELECT 내장형 ISNERT문이 만들어 졌으며, 파란색은 고정된 문장, 빨간색은 프로그램 상에서 FOR문을 통해 값이 계속 바뀌면서 생성되는 부분입니다.

```
INSERT INTO 'absolute_data_count' (category, element, timeType,
targetTime, timeCount) VALUES ('범죄 정보', '폭력', 'YEAR', ?,
(SELECT COUNT(*) FROM 'data_general' WHERE main == '범죄 정보'
AND sub == '폭력' AND NOT startTime IS NULL AND NOT endTime IS
NULL AND ((strftime('%Y', startTime) <= ? AND strftime('%Y',
endTime) >= ?))));
```

[시각화] : 시각화 계산 시작
[시각화] : 시각화 계산 완료 [연산 시작 5분 54초 이후]

[그림 4-32] 총 계산 시간

웬걸, 4주차와 동일한 실험 환경인 데이터 28,860개, 25섹터 기준 쿼리문을 통한 계산 결과, 총 5분 54초로, 5분 42초였던 4주차보다 12초 더 많이 걸렸습니다.

시간이 줄어들긴 커녕 더 늘어났고, 쿼리 계산의 가장 큰 문제점은, 동일한 쿼리문으로 동작하지 않는 ‘시각화 대상’에 대한 INSERT 문을 생성할 수 없는 것이었습니다.

때문에 다양성에 강한 Java의 힘은 ‘시각화 대상’에 쓰고, 일목요연한 데이터 연산에 강한 Query의 힘은 ‘환경 변수 대상’에 사용하여 전기차 마냥 상황에 맞게 두 힘을 바꿔서 사용하는 하이브리드 형식으로 변경하기로 하였습니다.

우선 Query의 힘을 더 향상시켜 주기 위해 배치 수를 더 줄이고

자 고정된 문장을 더 늘려 배치 수를 줄였습니다.

```
INSERT INTO 'absolute_data_count' (category, element, timeType,
targetTime, timeCount) VALUES (?, ?, 'YEAR', ?, (SELECT COUNT(*)
FROM 'data_general' WHERE main == ? AND sub == ? AND NOT
startTime IS NULL AND NOT endTime IS NULL AND ((strftime('%Y',
startTime) <= ? AND strftime('%Y', endTime) >= ?))));
```

이후 통합 연산식에서 ‘시각화 대상’ 따로, ‘환경 변수’ 따로 계산식을 분리하여 실행 시켜 보았습니다.

```
[SQLITE] : 2043개의 AbsoluteCountData가 추가됨.
[SQLITE] : 1790개의 AbsoluteGeoCountData가 추가됨.
[SQLITE] : 1129개의 AbsoluteGeoTargetCountData가 추가됨.
[SQLITE] : 41개의 AbsoluteDifferenceCountData가 추가됨.
[SQLITE] : 448개의 AbsoluteGeoDifferenceCountData가 추가됨.
[시각화] : 시각화 계산 완료 [연산 시작 5분 45초 이후]
```

[그림 4-33] 하이브리드 형식 계산 결과

이전과 비교하면 약 10초 정도 빨라진 모습을 확인 할 수 있었지만, 여전히 Java만 사용 했을 때 보다는 3초가량 늦었고, SELECT 한 대상을 바로 INSERT 하는 탓에 값이 0인 경우도 모두 들어가서 계산 초기 용량을 많이 잡아먹을 뿐만 아니라, 계산 이후 DELETE 명령어로 값이 0인 데이터를 잘라 내야했습니다.

때문에 INSERT-SELECT 문을 또 다시 두 쪽 내어 SELECT 따로 INSERT 따로 구문을 만들고, SELECT 문에서 모든 경우의 수를 판별하고, INSERT는 단순히 INSERT만 하도록 기능을 완전 분리 시켜 보았습니다.

```
[시각화] : 계산 시작
[SQLITE] : 94개의 AbsoluteTargetCountData가 추가됨. [연산 시작 2초 314% 이후]
[SQLITE] : 5001개의 AbsoluteGeoCountData가 추가됨. [연산 시작 1분 6초 이후]
[SQLITE] : 5001개의 AbsoluteGeoCountData가 추가됨. [연산 시작 3분 3초 이후]
[SQLITE] : 3186개의 AbsoluteCountData가 추가됨. [연산 시작 4분 57초 이후]
[SQLITE] : 4235개의 AbsoluteGeoCountData가 추가됨. [연산 시작 4분 57초 이후]
[SQLITE] : 1129개의 AbsoluteGeoTargetCountData가 추가됨. [연산 시작 12초 480% 이후]
[SQLITE] : 47개의 AbsoluteCountDifferenceData가 추가됨. [연산 시작 5초 447% 이후]
[SQLITE] : 199개의 AbsoluteGeoCountDifferenceData가 추가됨. [연산 시작 5초 455% 이후]
[시각화] : 계산 완료 [연산 시작 5분 18초 이후]
```

[그림 4-34] 3번째 실험 결과

모든 작업을 마치고 나니, Java에게 모든 연산을 맡겼을 때 보다 24초가량 빨라진 모습을 확인 할 수 있었습니다.

⑨ 섹터별 시각화 대상과 환경변수 간 평균 거리 구하기

이름부터 각종 수학 공식이 난무할 것만 같았는데, 실제로도 수학 공식이 난무하여 난감하였습니다. 두 점 사이의 거리를 피타고拉斯의 정의를 통해 구해야 하는데, 지구가 둥글기 때문에 이를 위도에 따라 다른 거리 값으로 연산해야 하는 점이 머리를 때렸습니다.

다행히 일전에 위도에 따른 거리 계산 함수를 공수 해 왔었기에 해당 함수를 쿼리문으로 풀어쓰기로 하였습니다.

```
private double distanceByMeter(double startLatitude, double startLongitude, double endLatitude, double endLongitude) {
    double theta = startLongitude - endLongitude;
    double dist = Math.sin(deg2rad(startLatitude)) * Math.sin(deg2rad(endLatitude))
        + Math.cos(deg2rad(startLatitude)) * Math.cos(deg2rad(endLatitude)) * Math.cos(deg2rad(theta));
    dist = Math.acos(dist);
    dist = rad2deg(dist);
    dist = dist * 60 * 1.1515;
    dist = dist * 1609.344;
    return (dist);
}
private double areaByMeter(double width, double height, int row, int column) {
    return (width / column) * (height / row);
}
private double deg2rad(double deg) {
    return (deg * Math.PI / 180.0);
}
private double rad2deg(double rad) {
    return (rad * 180 / Math.PI);
}
```

[그림 4-35] 위도에 따른 다른 두 점 사이의 거리를 나타내는 Java 함수

위 함수를 쿼리문으로 풀어쓰기 위해 SQLite에서 지원하는 함수가 있는지 찾아보았으며, 다행히 SIN, COS, ACOS, PI, POW, SQRT

모두 존재하였습니다.

```

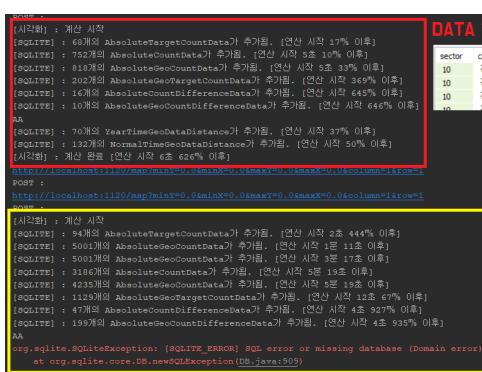
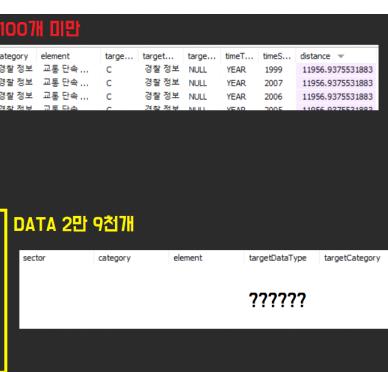
SELECT main, sub, AVG((ACOS(SIN(
CASE WHEN 35.5624698 > ((startLatitude+endLatitude) / 2)
    THEN ((startLatitude+endLatitude) / 2) * PI() / 180)
    ELSE (35.5624698 * PI() / 180)
END) *
SIN(
CASE WHEN 35.5624698 > ((startLatitude+endLatitude) / 2)
    THEN (35.5624698 * PI() / 180)
    ELSE (((startLatitude+endLatitude) / 2) * PI() / 180)
END) +
COS(
CASE WHEN 35.5624698 > ((startLatitude+endLatitude) / 2)
    THEN ((startLatitude+endLatitude) / 2) * PI() / 180)
    ELSE (35.5624698 * PI() / 180)
END) *
COS(
CASE WHEN 35.5624698 > ((startLatitude+endLatitude) / 2)
    THEN (35.5624698 * PI() / 180)
    ELSE (((startLatitude+endLongitude) / 2) * PI() / 180)
END) *
COS(
CASE WHEN 129.3504332 > ((startLongitude+endLongitude) / 2)
    THEN (((startLongitude+endLongitude) / 2) - 129.3504332)
    ELSE (129.3504332 - ((startLongitude+endLongitude) / 2))
END) *180 / PI())* 60 * 1.1515 * 1609.344)) AS distance FROM data_general GROUP BY sub

```

main	sub	distance
우로와이파이	고율시설	19933.359802608
어린이집	장애아동합+시간연장형	27927.6597917219
어린이집	시간연장형+휴일보육+24시간	117863.895525651
어린이집	일반+시간체보육	118975.434987198
어린이집	영어전달+시간연장형	122830.014941546
보안등	간죽률	125055.224765785
보안등	전용주	137896.471931258
어린이집	장애인아동합+시간연장형+휴일보육	142590.457843918
어린이집	장애인아동합+시간연장형+폐밀봉	142544.379934733

[그림 4-36] 위도 기반 두 점 사이의 거리를 구하는 쿼리

테스트 코드는 문제없이 평균 거리가 계산된 데이터가 DB에 정확히 잘 입력되었으나, 2만 9천 건의 실제 데이터를 입력 했더니, 어떠한 정보도 저장되지 않았습니다.

sector	category	element	targetDataType	targetCategory
10	경찰 정보	교통 단속...	C	경찰 정보
10	경찰 정보	교통 단속...	C	경찰 정보
10	경찰 정보	교통 단속...	C	경찰 정보
10	경찰 정보	교통 단속...	C	경찰 정보

[그림 4-37] 데이터 수가 늘어나자 Domain error가 발생한 화면

이에 모든 쿼리를 평문화 시켜 모니터링 해 본 결과, 2만 9천개 데이터 중에서 한 섹터 내에 거리를 구할 수 있는 대상이 없는 경우가 딱 1건 존재했습니다.

해당 값은 distance 값을 NULL로 반환하는데, 이를 INTEGER 컬럼인 distance가 받아내지 못하여 전체 데이터가 날아가는 것으로 확인되었으며, 그로인해 원래 길고 복잡했던 쿼리문이 보호 장치(CASE-END문)를 추가하는 바람에 2배로 더 길어졌습니다.

```

prefix.append("INSERT INTO " + AbsoluteDataAPI.TABLE_GEO_DATA_TEMP_DISTANCE + " " +
    "(sector, category, element, targetType, targetCategory, targetElement, timeType, timeString, distance) ");
prefix.append("SELECT ? AS sector, main AS category, sub AS element, " +
    "? AS targetType, ? AS targetCategory, ? AS targetElement, ? AS timeType, " +
    "? AS timeString, " +
    "CASE WHEN (" +
        "AVG((ACOS(SIN(CASE WHEN ? > ((startLatitude+endLatitude) / 2) THEN ((startLatitude+endLatitude) / 2) * PI() / 180) " +
        "*PI() / 180) END) * SIN(CASE WHEN ? > ((startLatitude+endLatitude) / 2) THEN (? * PI() / 180) " +
        "+ELSE (((startLatitude+endLatitude) / 2) * PI() / 180) END) + COS(CASE WHEN ? > ((startLatitude+endLatitude) / 2) " +
        "+THEN ((startLatitude+endLatitude) / 2) * PI() / 180) ELSE (? * PI() / 180) END) * " +
        "COS(CASE WHEN ? > ((startLatitude+endLatitude) / 2) THEN (? * PI() / 180) " +
        "+ELSE (((startLatitude+endLatitude) / 2) * PI() / 180) END) * COS(CASE WHEN ? > ((startLongitude+endLongitude) / 2) " +
        "+THEN ((startLongitude+endLongitude) / 2) - ?) ELSE (? - ((startLongitude+endLongitude) / 2)) " +
        "+END)*180 / PI())* 60 * 1.1515 * 1609.344) " +
    ") IS NULL THEN -1 ELSE (" +
        "AVG((ACOS(SIN(CASE WHEN ? > ((startLatitude+endLatitude) / 2) THEN ((startLatitude+endLatitude) / 2) * PI() / 180) " +
        "*PI() / 180) END) * SIN(CASE WHEN ? > ((startLatitude+endLatitude) / 2) THEN (? * PI() / 180) " +
        "+ELSE (((startLatitude+endLatitude) / 2) * PI() / 180) END) + COS(CASE WHEN ? > ((startLatitude+endLatitude) / 2) " +
        "+THEN ((startLatitude+endLatitude) / 2) * PI() / 180) ELSE (? * PI() / 180) END) * " +
        "COS(CASE WHEN ? > ((startLatitude+endLatitude) / 2) THEN (? * PI() / 180) " +
        "+ELSE (((startLatitude+endLatitude) / 2) * PI() / 180) END) * COS(CASE WHEN ? > ((startLongitude+endLongitude) / 2) " +
        "+THEN ((startLongitude+endLongitude) / 2) - ?) ELSE (? - ((startLongitude+endLongitude) / 2)) " +
        "+END)*180 / PI())* 60 * 1.1515 * 1609.344) " +
    ") END " +
    "AS distance " +
    "FROM data_general ");
prefix.append("WHERE ");
prefix.append("(startLatitude >= ? AND endLatitude <= ?) AND ");
prefix.append("(startLongitude >= ? AND endLongitude <= ?) AND ");
prefix.append("NOT startTime IS NULL AND NOT endTime IS NULL AND (");
prefix.append("strftime(?, startTime) <= ? AND strftime(?, endTime) >= ? )");

```

[그림 4-38] 보호 장치가 추가되어 2배로 길어진 쿼리문

sector	category	element	targe...	targe...	targe...	timeT...	timeS...	distance
3	도시공원정보	근린공원	C	보안등	NULL	YEAR	2017	403260.912442984
3	도시공원정보	근린공원	C	보안등	NULL	MONTH	10	403260.912442984
3	도시공원정보	근린공원	C	보안등	NULL	MONTH	09	403260.912442984
3	도시공원정보	근린공원	C	보안등	NULL	MONTH	11	403260.912442984
3	도시공원정보	근린공원	C	보안등	NULL	MONTH	12	403260.912442984

[그림 4-39] YEAR 데이터 까지 제대로 들어간 모습

제 5장 시각화

5.1 시각화 계획 구상

5.1.1 시각화 방식 선정

모든 시각화 옵션은 전체/연도/월/요일/시간 단위로 표시할 수 있도록 만들고, 전체 시간을 표시할 경우, 모든 시간의 확률모델을 구역별 재 합산하여 5점 척도로 나누기로 하였습니다.

또한 시간 세부 설정(오후 3시부터 오후 6시 까지 등)이 가능하게 만들되, 지도를 제외한 모든 시각화 옵션은 전체/섹터 단위 설정이 가능하도록 만들기로 하였습니다.

① 지도

- 선택된 시각화 대상의 절대 발생 건수를 지도에 표시.
- 개별 환경변수의 발생 건수를 지도에 표시. (레이어 선택 형식)
- 시각화 대상의 발생에 영향을 끼치는 환경변수의 계산 값을 이용하여 구획별 시각화 대상 발생 가능성이 높은 지역 순으로 색을 다르게 표시. (발생 가능성 예측)

② 표

- 시각화 대상의 절대 발생 건수를 순위별로 테이블에 표시.
- 개별 환경변수의 절대 발생 건수를 순위별로 테이블에 표시.

③ 파이 차트

- 대상(전체/섹터) 선택이 가능하도록 옵션을 만들기.
- 시각화 대상의 절대 발생 건수를 파이 차트에 표시.
- 개별 환경변수의 절대 발생 건수를 파이 차트에 표시.
- 시각화 대상 발생률을 가장 많이 높이는 환경변수에 대한 지분을 파이 차트에 표시.

④ 꺾은 선 그래프

- 시간 변화에 따른 개별/다중 시각화 대상의 절대 발생 건수 증감 수치 표시.

- 시간 변화에 따른 개별/다중 환경 변수의 절대 발생 건수 증감 수치 표시.

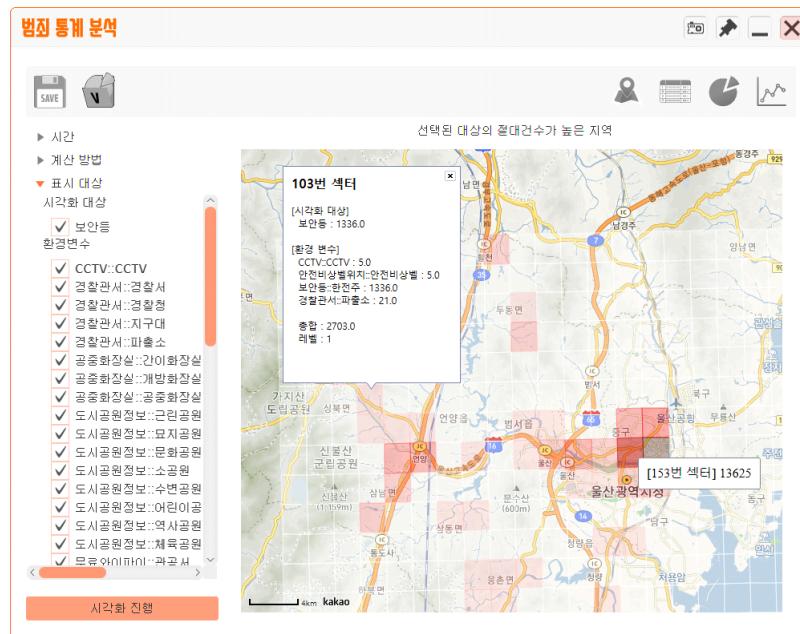
5.2 개발

5.2.1 지도

- 전체/연도/월/요일/시간 단위 설정을 좌측 옵션 패널에서 선택 가능하게 하였다.
- 8가지 색상으로 섹터별 색상을 입혀 줌으로 써 시각 효과를 더했다.

① 절대 건수기반 시각화

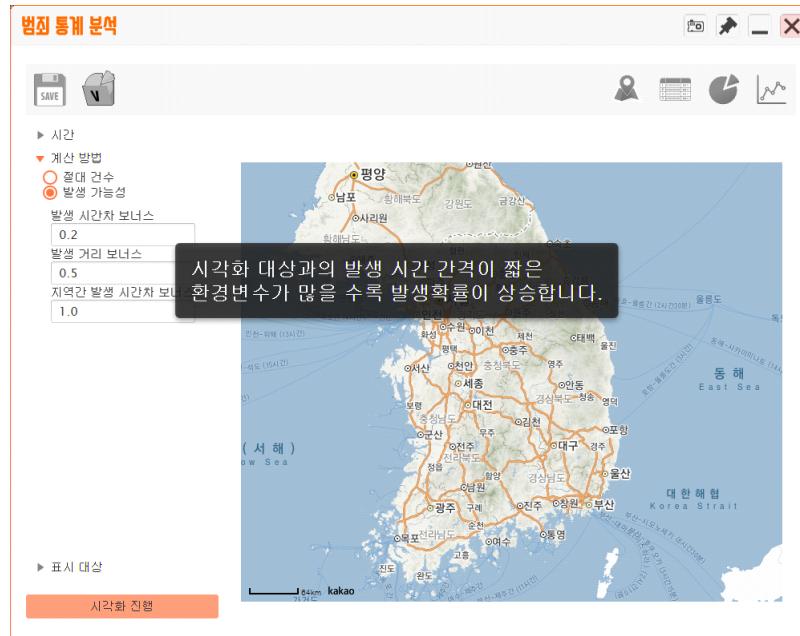
- 선택된 시각화 대상의 절대 발생 건수를 지도에 표시.
- 개별 환경 변수의 절대 발생 건수를 지도에 표시. (레이어 선택 형식)



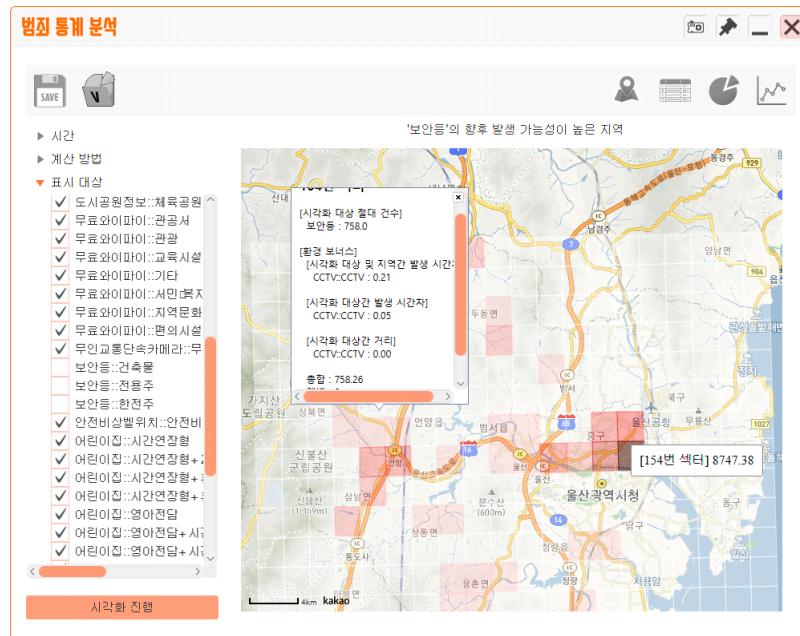
[그림 5-1] 절대 건수 시각화를 진행 시킨 모습

② 발생가능성 기반 시각화 (발생 예측)

- 선택된 시각화 대상을 선택된 환경변수를 통해 계산하여 발생위험이 높은 순으로 지도에 표시.
- 발생가능성 연산 방식은 아래 순서를 따른다.
 1. 섹터별 시각화 대상의 발생 절대 건수를 구한다.
 2. 섹터별 환경변수의 발생 절대 건수를 구한다.
 3. 1을 [섹터별 환경변수와 시각화 대상 간 시간대별 발생 오차 값]으로 나눈다.
 4. 0.2를 [전체 환경변수의 발생 횟수와 전체 시각화 대상 간 발생 오차 값]으로 나눈다.
 5. 0.5를 [섹터별 각 환경변수와 시각화 대상 간 거리 평균]으로 나눈다.
 6. 각 3,4,5번 결과 값을 2번 결과 값과 곱하고, 곱한 3가지 값을 모두 더한 다음, 1번과 더한다.
 7. 모든 섹터 중 가장 작은 값을 가진 섹터가 시각화 대상의 발생 확률이 가장 낮은 곳이며, 가장 높은 값을 가진 섹터가 시각화 대상의 발생 확률이 가장 높은 섹터가 된다.
- 섹터별 오차 값을 1, 전체 대상 오차 값을 0.2, 거리 대상 오차 값을 0.5로 준 것은 임의로 섹터별 오차가 가장 영향력을 크게 갖고, 그 다음이 섹터 간 시각화 대상간의 거리 값을 영향력을 크게 갖도록 필자 임의로 조정 한 것이므로, 영향력 수치를 사용자가 직접 조정 가능한 설정 패널을 [계산 방법] 카테고리에 추가하였다.

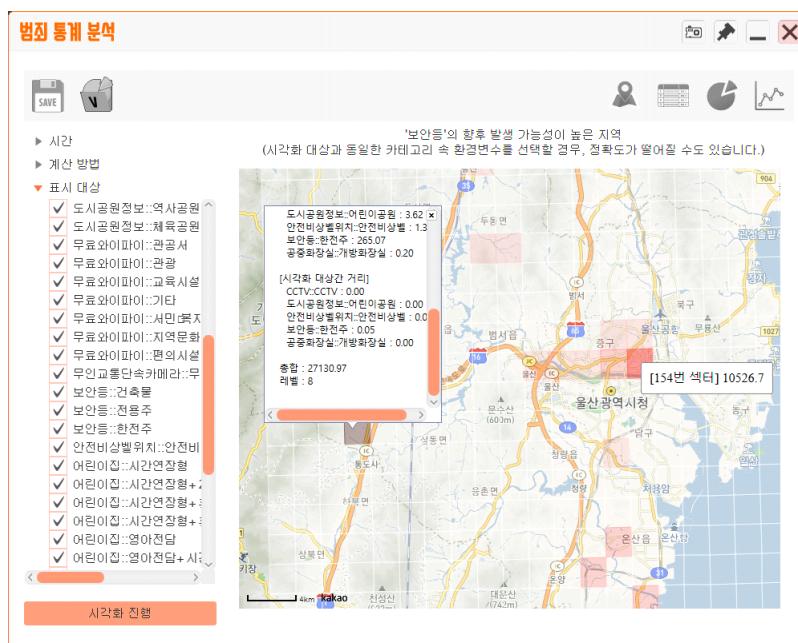


[그림 5-2] 영향력 조절 패널



[그림 5-3] 발생 가능성 시각화를 진행 시킨 모습

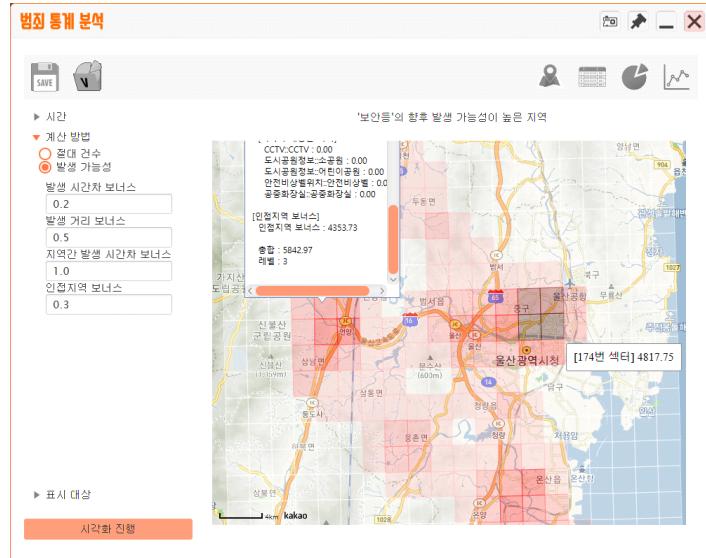
- 수차례 실험 결과, 시각화 대상과 동일한 요소를 환경변수 상에서 선택할 경우, 계산 오차가 크게 나지 못하여 결과가 만족스럽지 못하였다. 때문에 시각화 대상과 동일한 요소를 환경변수 상에서 선택한 채로 시각화를 진행시키려 할 경우, 경고 메시지를 띠워주기로 했다.



[그림 5-4] 경고 메시지를 띠워주는 모습

5.2.2 인접지역 보너스

- PredPol에서 그랬듯, 범죄자는 이전 범죄에 성공한 장소로 돌아가며, 반복되는 피해는 이웃에게도 피해가 가고, 가해자의 집은 주요 활동지점과 멀리 떨어지지 않은 곳에 있는 점을 고려해야합니다.
- 때문에 발생확률이 높은 섹터일수록 더 많은 발생확률 보너스를 적용하고, 주변 8블록 이내 발생확률에 영향을 끼치도록 하였습니다.

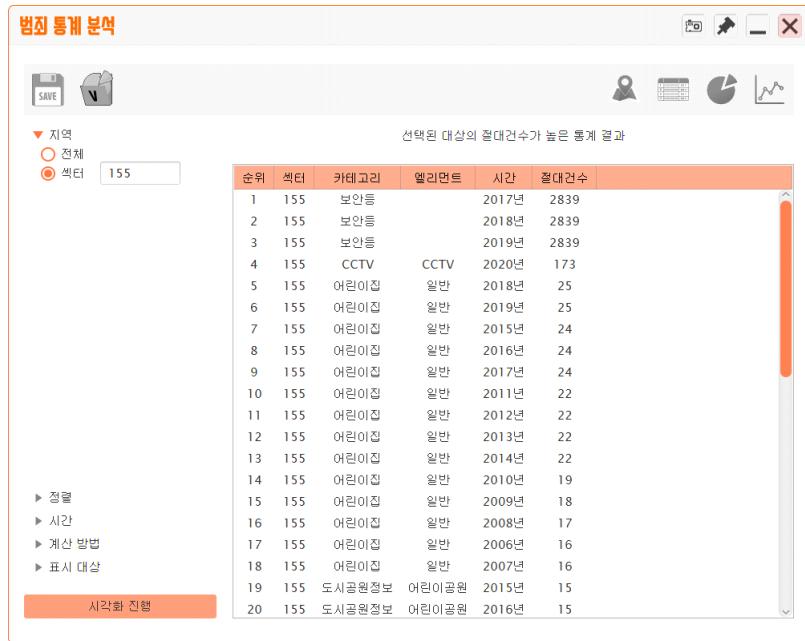


[그림 5-5] 인접지역 보너스를 적용한 모습

- 인접지역 보너스 적용이후, 더욱 부드러운 타일 배치가 이루어졌습니다.

5.2.3 표

- [지역] 카테고리에서 <전체>대상 혹은 특정 <섹터> 대상으로 표시할지 선택 가능하도록 하여 사용자의 편의성을 높였습니다.



[그림 5-6] 전체지역 선택과 특정 섹터 선택이 적용되는 모습

- [정렬] 카테고리에서 <오름차순> 혹은 <내림차순> 정렬 설정이 가능하며, 도표에 표시될 데이터 최대치를 지정할 수 있습니다. (최소 1, 최대 100)

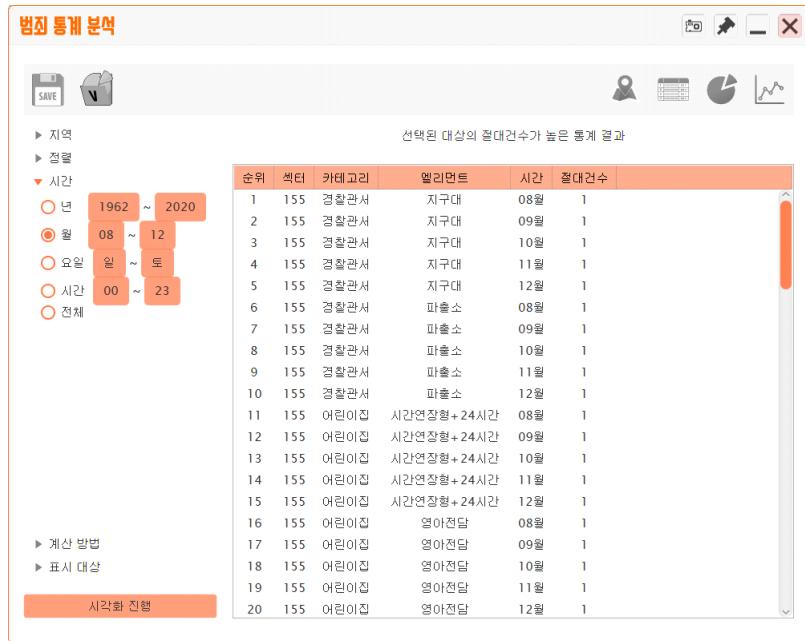
범죄 통계 분석

선택된 대상의 결과 건수가 높은 통계 결과

순위	브터	카테고리	옐리먼트	시간	결과건수
1	155	보안등		1997년	1
2	155	보안등		1998년	1
3	155	보안등		1999년	1
4	155	보안등		2000년	1
5	155	보안등		2001년	1
6	155	보안등		2002년	1
7	155	보안등		2003년	1
8	155	보안등		2004년	1
9	155	보안등		2005년	1
10	155	보안등		2006년	1
11	155	보안등		2007년	1
12	155	보안등		2008년	1
13	155	보안등		2009년	1
14	155	보안등		2010년	1
15	155	보안등		2011년	1
16	155	보안등		2012년	1
17	155	보안등		2013년	1
18	155	보안등		2014년	1
19	155	보안등		2015년	1
20	155	보안등		2016년	1

[그림 5-7] 데이터 표시 제어

- [시간]카테고리에서 검색할 시간대를 설정할 수 있습니다.



[그림 5-8] 데이터 표시 범위 제어

- [계산방법]카테고리에서 <절대 건수> 기반으로 표시할지, 다양한 환경변수 영향에 따른 <발생 가능성> 기반으로 표기할지 설정이 가능하며, <발생 가능성>계산 시, 사용자 임의로 보너스 수치를 조절 할 수 있습니다.



[그림 5-9] 계산 방식 선택

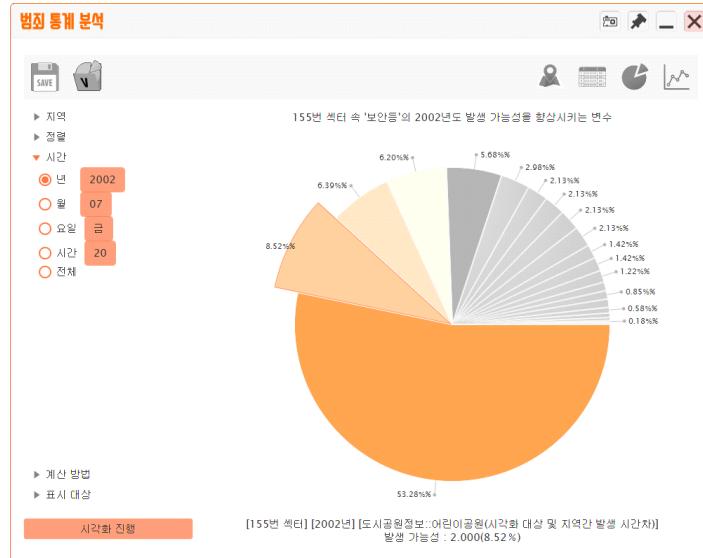
- [표시 대상]카테고리에서 시각화 시킬 대상을 선택할 수 있으며, 저번 주에 없던 [모두 선택] 버튼과 [선택 취소] 버튼을 추가하여 편의성을 더하였습니다.



[그림 5-10] 시각화 대상 선택

5.2.4 파일 차트

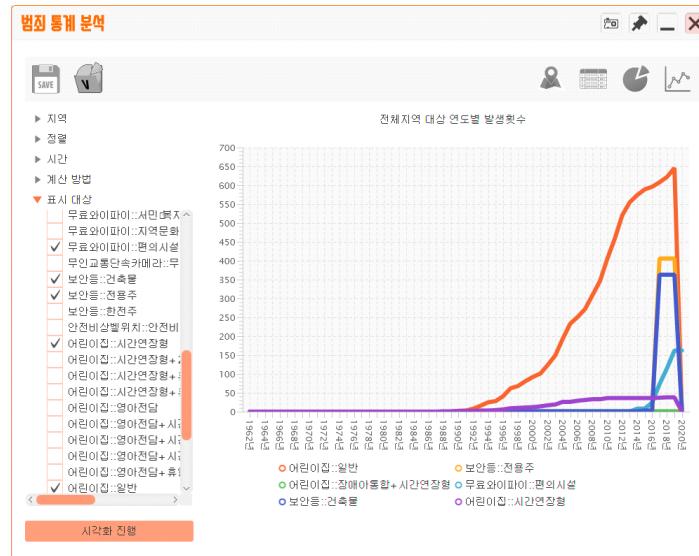
- 앞서 나온 [표]의 모든 기능을 추가하였으며, 마우스 이벤트를 걸어 각 영역 위에 마우스를 올릴 경우, 해당 영역이 확대되면서 화면 하단부에 상세 수치가 나타나도록 하였습니다.



[그림 5-11] 파이차트

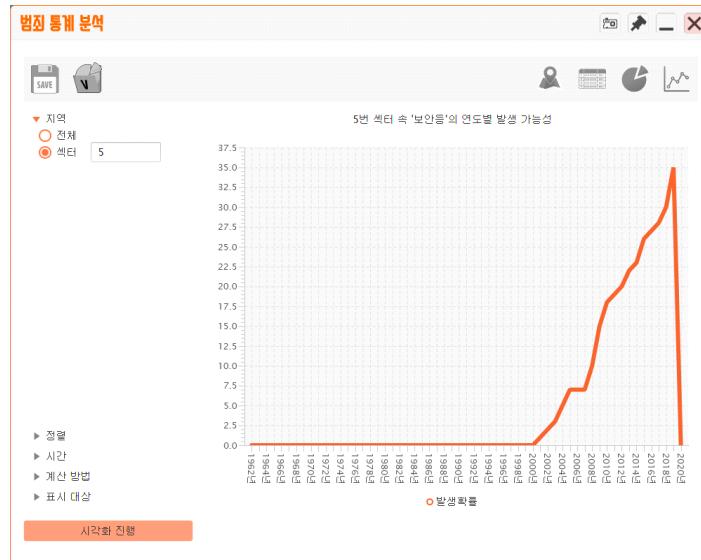
5.2.5 라인 차트

- 라인 차트는 그 특성이 ‘시간대별’ 비교하는 것이기에 한 번에 다량의 정보를 시간대별로 분리해야 했습니다. 때문에 성능 이슈가 발생하지 않도록 [절대 건수] 비교의 경우, 최대 10개 항목만 선택 가능하도록 하였으며, 표, 파이차트와 같이 지역별, 시간별 다른 값을 지정할 수 있도록 하여 분석 가능 범위를 넓혀주었습니다.



[그림 5-12] 다양한 대상의 절대건수를 비교하는 모습

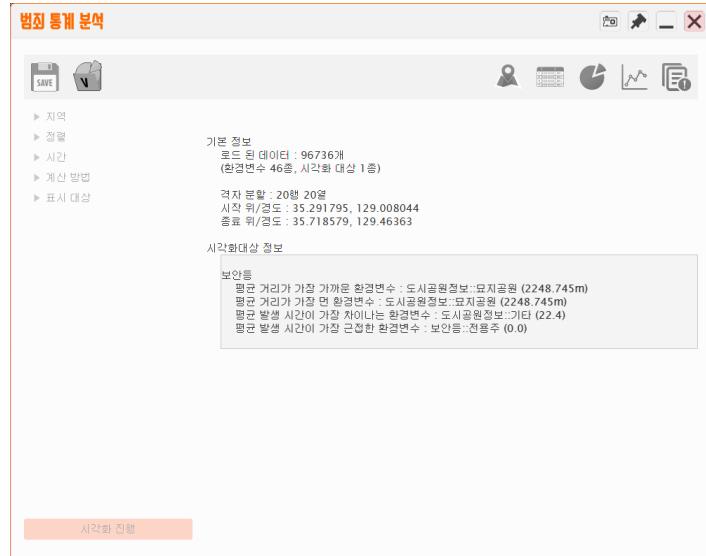
- 라인 차트에서 발생 가능성을 확인한다면, 필시 시간대별 발생 가능성이 어떻게 늘고 줄었는지 확인하려는 행위이기에, 보너스 수치를 여러 라인으로 복잡하게 표시하지 않고, 라인 1줄로 통합하여 직감적으로 볼 수 있도록 하였습니다.



[그림 5-13] 특정 대상의 발생 가능성을 시각화한 모습

5.2.6 브리핑 페이지

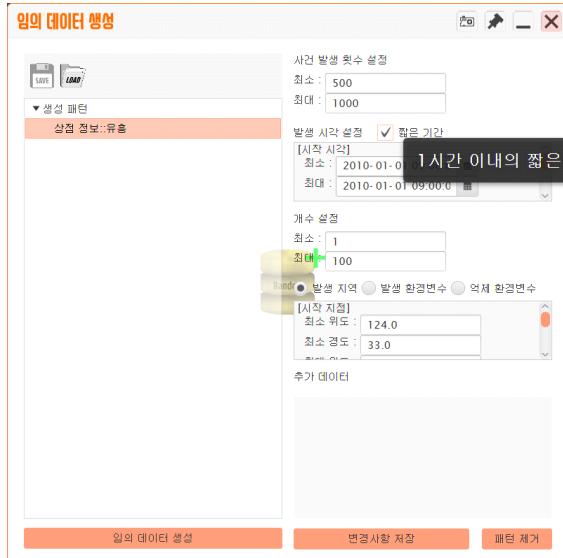
- 데이터에 대한 시각화는 모두 완성하였지만, 그래도 사용자에게 더 많은 정보를 제공하고 싶었기에 브리핑 페이지를 따로 생성하였습니다.
- 브리핑 페이지는 현재 시각화에 사용 중인 DB에 대한 정보와 각종 시각화대상들의 절대 정보(DB내에서 변동사항이 없는)를 제공하게 됩니다.
- 시각화대상의 절대정보로는 [평균 거리가 가장 가까운/먼 환경변수], [평균 발생 시간이 가장 근접한/차이나는 환경변수] 총 4가지로, 연산 파라미터로 어떤 값을 넣느냐에 따라 달라지는 [발생 가능성]에 대한 정보는 제공하지 않습니다.



[그림 5-14] 브리핑 페이지

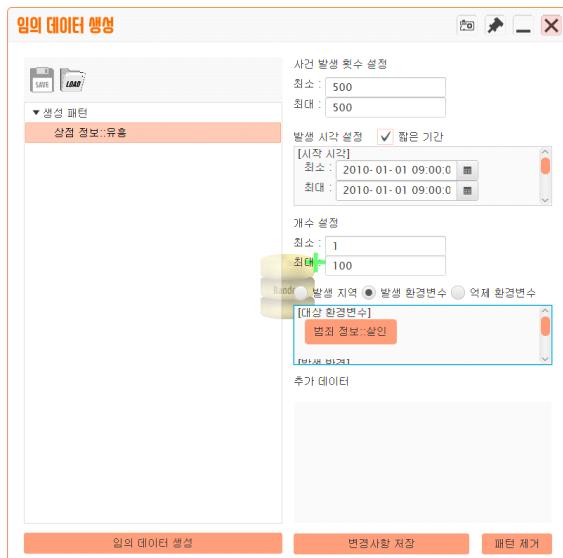
5.2.7 임의 데이터 생성기능 향상

- 신뢰도 검증 및 테스트를 위해 가짜 범죄 사건을 데이터 셋에 추가해야하지만, 어두운 조명 근처에 살인 사건을 몰아서 생성하거나, 공중화장실 근처에 성폭행 사건을 몰아서 생성 및 CCTV와 가까워질수록 범죄 데이터 제거 등, 질의다운 질의 형식을 사용하는 기능이 부족하여 임의 데이터 생성 기능을 손보기로 하였습니다.
- 우선 발생 시각 설정 시 완전한 임의 값을 가져오기 때문에 1999년에 일어난 범죄가 2020년 까지 이어질 수도 있었습니다. 때문에 1시간 이내의 [짧은 기간]생성 옵션을 추가하였습니다.



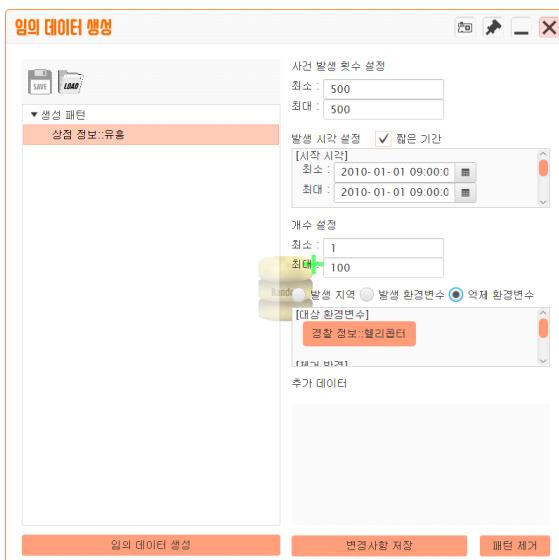
[그림 5-15] 짧은 기간 옵션이 추가된 모습

- 특정 환경변수 인근에 사건을 생성시키는 [발생 환경변수] 옵션을 넣었습니다. 대상 환경변수를 설정하고, 발생 최소 거리와 최대 거리를 입력할 경우, 자동으로 근방에 사건을 생성합니다.



[그림 5-16] 환경변수 선택이 가능한 모습

- 특정 환경변수 인근에 위치한 사건을 소멸시키는 [억제 환경변수] 옵션을 넣었습니다. 제거 대상 환경변수를 설정하고, 억제 최소 거리와 최대 거리를 입력할 경우, 자동으로 근방에 위치한 사건 중, 제거하고 싶은 종류의 사건을 제거합니다.



[그림 5-17] 억제 옵션이 추가된 모습

5.3 실험

5.3.1 실험조건

범죄 데이터는 정상적인 방법으로 입수할 수 없기에, ‘특정 환경 요소 주변에 임의 범죄 데이터를 생성시킨 뒤, 프로그램이 해당 특정 환경 요소를 얼마나 잘 찾아내는가?’를 실험하기로 했으며, 범죄 생성 패턴은 다음과 같이 지정하였습니다.

유괴 18000개 (어린이집으로 부터 0.002(대략 200m) 이내에 생성)

CCTV 0.00005(대략 5m) 이내 최대 600개 제거

성폭행 3000개 (공중화장실로 부터 0.00001(대략 1m) 이내에 생성)

CCTV 0.00005(대략 5m) 이내 최대 600개 제거

살인 10000개 (울산시 전역에 임의 생성)

파출소 0.005(대략 500m) 이내 최대 1000개 제거

지구대 0.008(대략 800m) 이내 최대 1000개 제거

경찰서 0.01(대략 1000m) 이내 최대 1500개 제거

경찰청 0.015(대략 1500m) 이내 최대 2000개 제거

안전비상벨 0.00005 이내 최대 500개 제거

CCTV 0.00005(대략 5m) 이내 최대 4000개 제거

무인교통단속카메라 0.00005(대략 5m) 이내 최대 600개 제거

보안등(3종)으로 부터 0.00005(대략 5m) 이내 최대 250개 제거

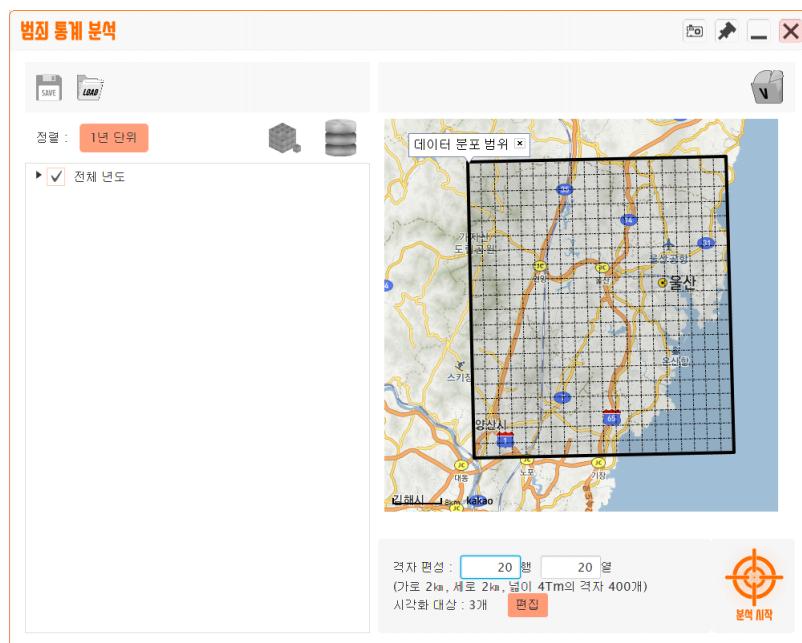
위 스크립트대로 임의 범죄 데이터 생성 결과, 기존 사용하던 환경변수와 합쳐 총 49,753건의 데이터를 얻게 되었습니다.

생성된 데이터를 통해 시각화 대상을 지정하기로 하고, 시각화 대상으로는 ‘유괴’사건과 ‘20세 이상 30세 미만인 여성을 대상으로 한 성폭행’, ‘가해자 연령이 30세 이상, 40세 미만인 남성인 살인’사건을 선택하였습니다.



[그림 5-18] 시각화 대상 설정

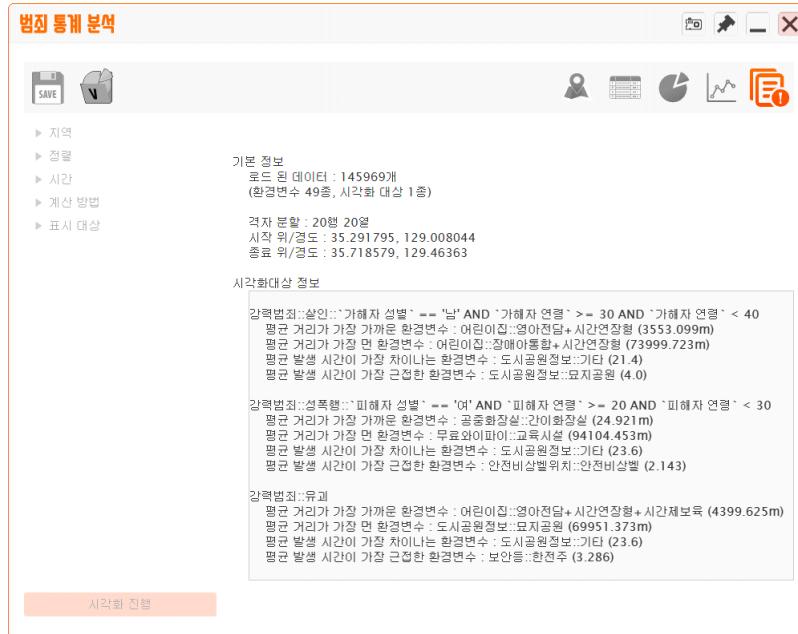
시각화 범위는 울산 전역이며, 20×20 총 400개의 섹터로 분할하여 약 2km 넓이로 일대를 나누었습니다.



[그림 5-19] 격자 편성화면

5.3.2 시각화 및 평가

실험에 앞서, 생성된 범죄 데이터는 ‘특정 환경변수 주변’에 ‘임의’시간대로 생성한 것이기에 ‘시간대별 증감 수치’는 전혀 믿을만한 자료가 되지 않으므로 [발생 시간차 보너스], [지역 간 발생 시간차 보너스] 수치를 모두 0으로 설정하고, [발생 거리 보너스 : 1000]와 [인접지역 보너스 : 0.3]만 적용시키기로 하였습니다.



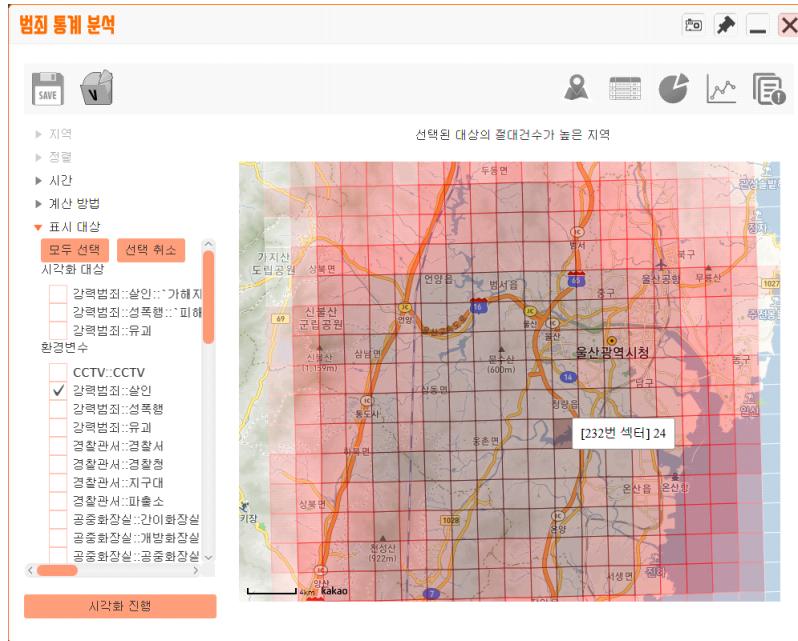
[그림 5-20] 브리핑 페이지

우선 브리핑 페이지를 확인 해 보니, [유괴]사건은 어린이집(영아전담) 인근에서 가장 많이 발생하였으며, 평균 4,399m 이내에서 발생하였습니다. 사건 발생에 가장 영향을 적게 미치는 환경변수는 도시공원(묘지)로, 유괴사건 발생 지역으로부터 평균 6,9km 떨어 져 있었습니다.

[20세 이상 30세 미만의 여성 대상 성폭행]은 공중화장실(간이화장실) 인근에서 가장 많이 발생하였으며, 평균 24m 이내에서 발생하였습니다. 반면 사건 발생에 가장 영향을 적게 미치는 환경 변수는 무료와이파이(교육시설)근처이며, 사건 발생지역에서 평균 9,4km 떨어 져 있었습니다.

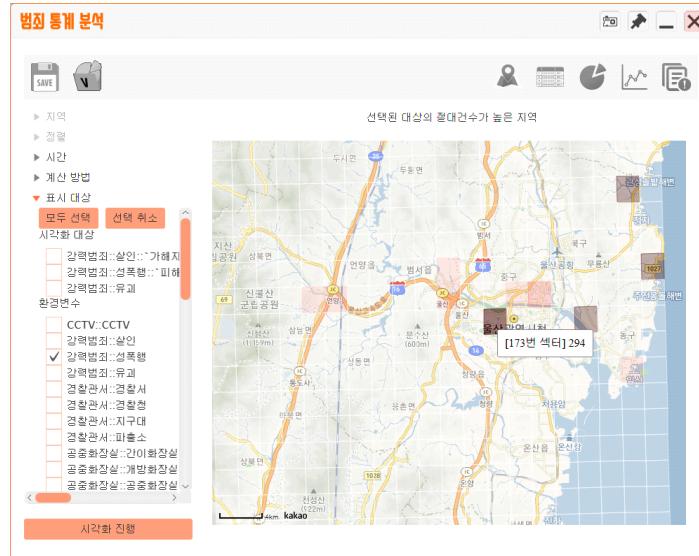
[30세 이상 40세 미만의 남성 가해자 살인]은 어린이집(영아전담) 인근에서 가장 많이 발생하였으며, 평균 3,553m 이내에서 발생하였습니다. 가장 영향을 적게 미치는 환경 변수는 어린이집(장애아동통합)이었으며, 평균 7.4km 떨어 져 있었습니다. 브리핑 페이지에서 획득한 정보를 통해, 특정 환경변수 인근에만

발생 시킨 유괴 및 성폭행 스크립트를 잘 간파했음을 알 수 있었습니다. 하지만 전역에 걸쳐 발생시킨 살인 사건을 파악하기 위해서는 브리핑 페이지 이외 추가 분석이 필요 해 보였습니다. 때문에 각 사건별 절대 건수 분포도를 확인하여, 어떤 섹터에 얼마나 분포하고 있는지 확인 해 보기로 하였습니다.



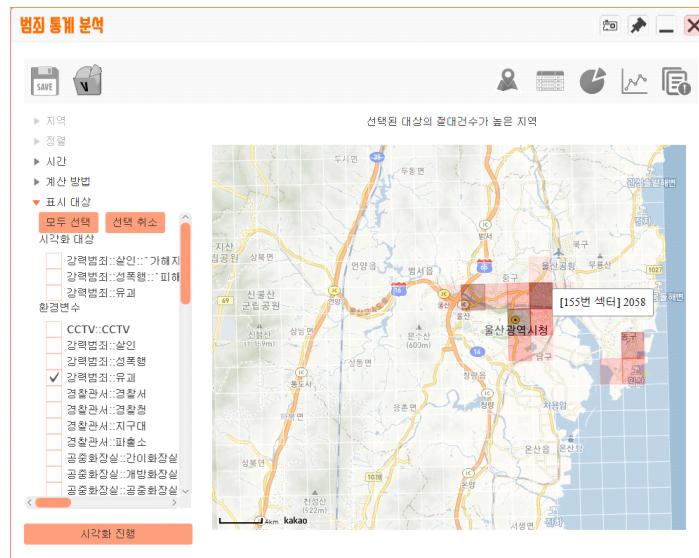
[그림 5-21] 살인 사건 절대 건수 분포도

살인사건은 스크립트대로 전 섹터에 걸쳐 고루 분포되었으며, 억제 수단을 많이 추가시킨 만큼, 그 절대 건수가 적기에 가장 심각한 레벨 8 기준은 그저 연, 월, 일, 시별 절대 건수 수치가 24만 넘으면 되었습니다.



[그림 5-22] 성폭행 사건 절대 건수 분포도

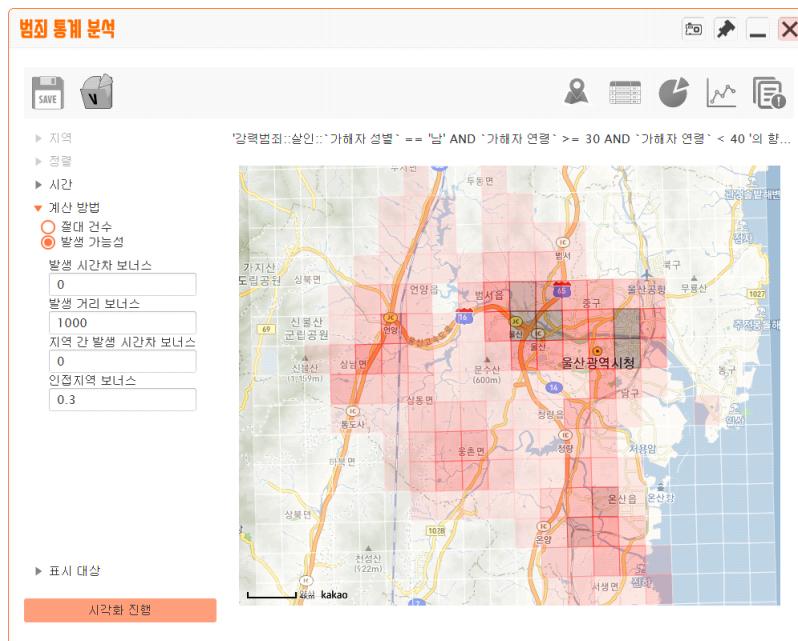
성폭행 사건은 스크립트 상 공중화장실이 있는 지역에 발생하게 하였는데, 주로 공중화장실이 많은 해수욕장 인근이나 도심지에서 발생하였습니다.



[그림 5-23] 유괴 사건 절대 건수 분포도

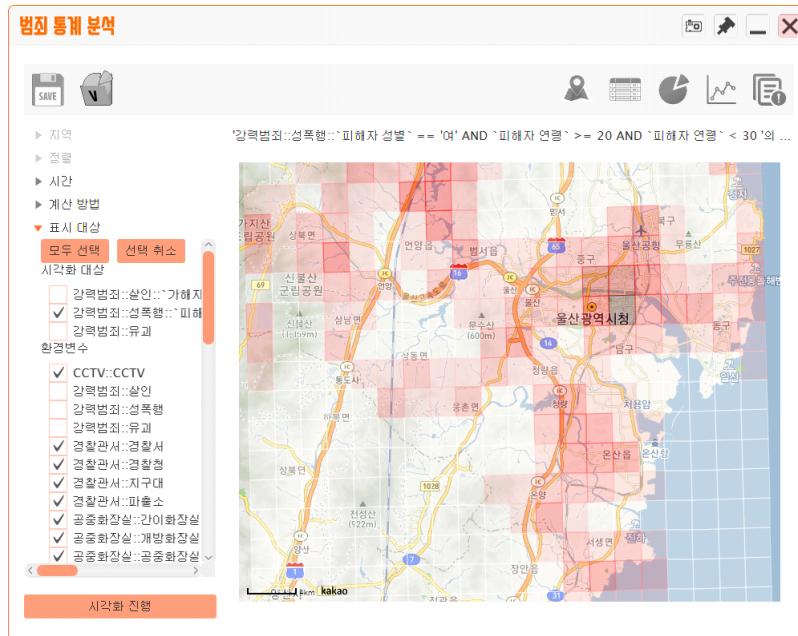
유괴사건은 스크립트 상 어린이집 인근에 발생하게 하였었는데, 주로 도심지에서 발생한 모습을 확인 할 수 있었습니다.

각 사건별 발생 집중구역을 지도로 확인하였으므로, 거리 보너스와 인접지역 보너스를 적용시켜 향후 발생 가능성이 높은 지역을 출력해 보기로 하였습니다.



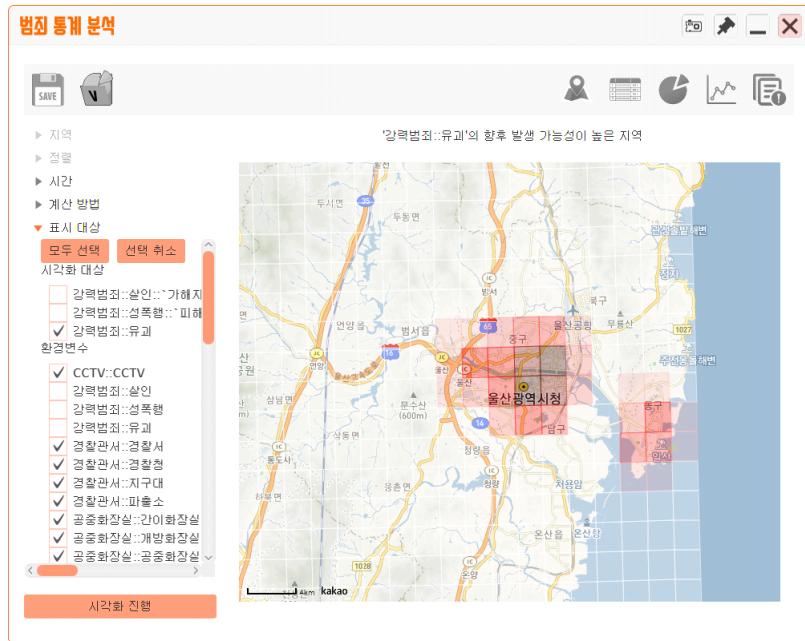
[그림 5-24] 향후 살인 사건이 발생할 가능성이 높은 지역

절대 건수로 확인 했을 땐 바다 한 가운데서도 많은 살인 사건이 찍혔었지만, 향후 발생 가능성에 대한 계산 결과, 사람이 실제 거주하는 구역에 집중되었으며, 인구 밀집도가 낮은 지역(산)에선 실제로 발생 가능성이 낮게 나왔습니다.



[그림 5-25] 향후 성폭행 사건이 발생할 가능성이 높은 지역

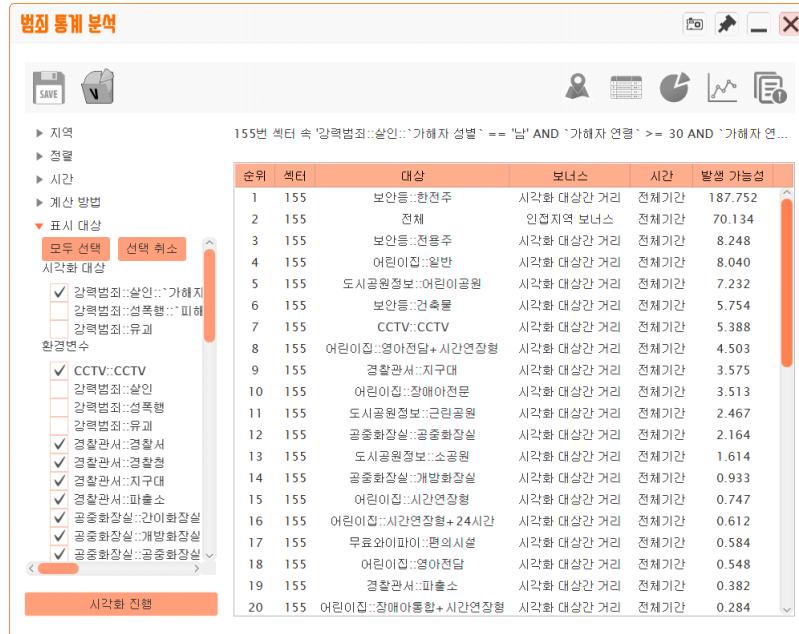
성폭행 사건의 경우, 절대 건수로 확인 했을 때는 해수욕장 인근과 도심지 몇몇 곳에 발생한 것으로 확인되었지만, 발생 가능성 계산 이후, 살인사건과 비교될 정도로 해안가 지역의 발생 가능성이 뚜렷하게 많아졌으며, 비슷한 도시라도 강가, 하천을 주변에 둔 곳이 더 높은 발생 가능성을 보여주었습니다.



[그림 5-26] 향후 유괴 사건이 발생할 가능성이 높은 지역

유괴 사건의 경우, 공업단지가 있는 온산읍이나 인구밀도가 적은 지역은 발생 가능성이 현저히 낮았지만, 학교, 유치원, 인프라시설 등이 모인 도심지에 집중적으로 발생 가능성이 높게 나타난 모습을 볼 수 있었습니다.

각 사건별 발생 가능성이 가장 높은 섹터를 알게 되었으므로 어떤 환경변수가 발생 가능성을 높이는지 확인 해 보기로 하였습니다.



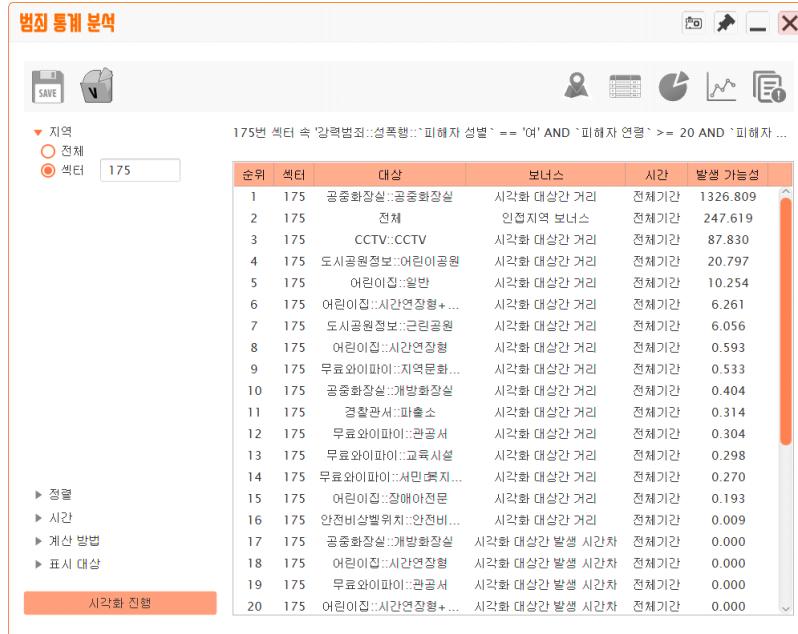
[그림 5-27] 살인 사건 발생률이 가장 높았던 155번 지역 차트

살인의 경우, 살인 사건을 발생시키는 환경변수 지정 없이, 전국적으로 골고루 생성하였었으며, 파출소, 지구대, 경찰청, 경찰서, 안전비상벨, CCTV, 무인교통단속카메라, 보안등 근처에서 일부 살인 사건을 억제하도록 스크립트를 작성하였습니다.

그 영향 탓인지 전체적인 환경변수가 고루 영향을 끼쳤다고 나와 있었으며, 살인사건 장소와 가장 가깝고, 가장 개체 수가 많았지만, 억제율이 가장 낮았던(250개 억제) 보안등이 발생 가능성을 높여주는 원인으로 1, 3, 6위를 차지하고 있었습니다.

이후 아무런 억제/발생 능력을 가지지 않은 어린이집(일반)이 살인사건 가까이에 많이 위치하였기 때문에 4순위를 차지하였고, 뒤이어 도시공원이 5위를 차지하였습니다.

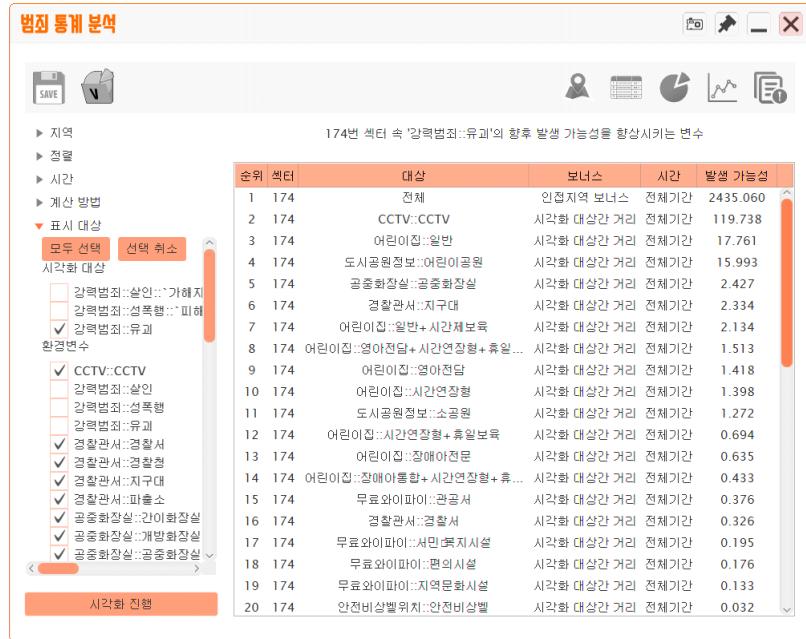
CCTV의 경우, 보안등 보다는 개체수가 적지만 4천개를 억제하도록 설정 한 덕에 7순위를 이어받았으며, 지구대, 파출소, 경찰청 등의 가장 높은 억제율을 보여주던 환경변수는 모두 수치가 4점미만으로 매우 낮게 나오는 모습을 보여주었습니다.



[그림 5-28] 성폭행 사건 발생률이 가장 높았던 175번 지역 차트

성폭행 사건의 경우, 공중화장실이 1326점을 기록하며 사건을 발생시키는 환경변수를 지정하지 않았던 살인사건의 1위(187점)와 매우 큰 차이를 보여주었습니다.

스크립트 상 CCTV 인근 성폭행 사건을 억제 시켰음에도 불구하고, 억제 범위(약 5m)와 억제량(최대 600개)을 너무 작게 설정한데다, CCTV가 총 3,957개나 비치되어 있는 탓에 성폭행 발생의 원인 3순위로 잡힌 모습을 볼 수 있었습니다.



[그림 5-29] 유괴 사건 발생률이 가장 높았던 174번 지역 차트

유괴사건은 어린이집과 어린이공원이 각 17, 15점을 기록하였으며, 이외 환경변수는 3 이하 낮은 수치를 보였습니다. 하지만 성폭행 사건과 마찬가지로 CCTV의 경우, 억제량, 억제 범위는 낮은데, 분포는 많이 되어 있어서 발생 순위 2위를 차지하고 있었습니다.

이처럼 CCTV의 수가 전체 영역에 걸쳐 압도적으로 많은 수를 가지고 있는 탓에, 애초 설정했던 범죄 발생패턴과는 관계없이 사건 발생율을 상승시키는 대상으로 지목되고 있었습니다.

덕분에 개수가 적을수록 개체 하나하나가 큰 영향력을 발휘하고, 개수가 많을수록 개체의 영향력이 줄어드는 평등화 계산식(개수에 반비례하는 계산법)을 떠올릴 수 있었습니다.

5.3.3 평등화 계산식 적용 시각화 및 평가

평등화를 위해 개수에 반비례하는 값을 반환하는 쿼리문을 작성하고, 결과를 거리 계산식, 시간차 계산식 등에 곱해주었습니다.

```

public Map<String, Double> getEqualizer() {
    Map<String, Double> returnMap = new HashMap<>();
    String environmentQuery = "SELECT (category||\">Main.SPLIT_SYMBOL+\"||element) AS key, (1.0/SUM(timeCount)) AS bonus " +
        "FROM absolute_data_count WHERE timeType = 'YEAR' GROUP BY element";
}

```

[그림 5-30] 평등화 보너스를 가져오는 쿼리구문

```

geoDataDifferenceMap = Absolute.absoluteDB.getGeoAbsoluteDifference(selectedTime, sectorTime,
    sectorTarget);
dataDifferenceMap = Absolute.absoluteDB.getAbsoluteDifference(selectedTime, timeAvgPower, ar-
    gentTarget);
geoDistanceMap = Absolute.absoluteDB.getGeoDistance(selectedTime, time, distanceAvgPower, pa-
    rtmentTarget);
if(equalizer && !equalizerMap.isEmpty()){
    for(String ele : equalizerMap.keySet()){
        for(int sector : geoDataDifferenceMap.keySet()){
            if(geoDataDifferenceMap.get(sector).containsKey(ele)){...}
        }
        for(int sector : geoDistanceMap.keySet()){
            if(geoDistanceMap.get(sector).containsKey(ele)){...}
        }
        if(dataDifferenceMap.containsKey(ele)){
            dataDifferenceMap.put(ele, dataDifferenceMap.get(ele) * equalizerMap.get(ele));
        }
    }
}

```

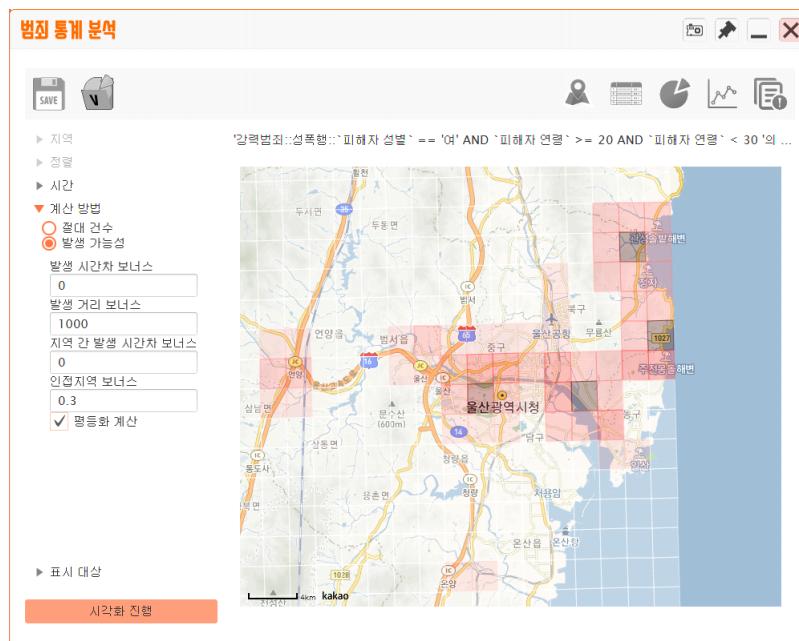
[그림 5-31] 평등화 값을 각 보너스 수치에 곱하는 모습

이후 [평등화 수치 적용]을 검색 옵션에 추가시킨 뒤, 앞선 실험에서 사용 한 데이터를 그대로 적용시켜 동일한 실험을 반복하였습니다.



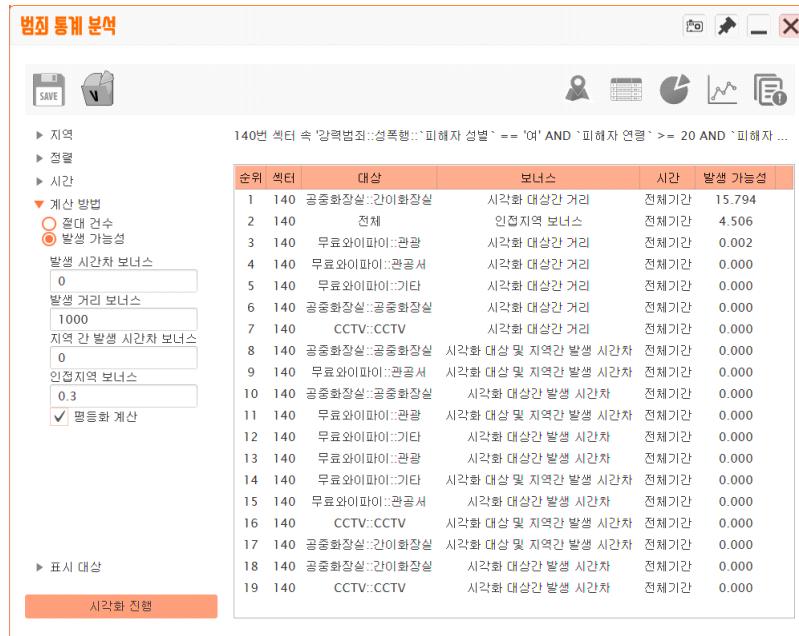
[그림 5-32] 살인 사건 발생률이 가장 높았던 155번 지역 차트

앞서 진행했던 실험 결과와는 다르게 살인 사건 발생에 특별히 영향을 주는 환경변수를 찾지 못한 모습을 보였으며, 살인 사건 억제율이 가장 높았던 경찰청, 경찰서는 아예 순위권에 들지도 못하였고, 안전 비상벨, CCTV, 무인교통단속카메라 등은 0.000 미만 수치를 기록하는 등, 영향을 끼치는 환경변수가 없는 점을 보아 전국적으로 고루 사건을 발생 시켰으며, 경찰청, 경찰서, CCTV 등이 살인사건을 억제하는 환경변수임을 알 수 있는 등, 일부 스크립트를 유추할 수 있는 결과를 보여주었습니다.



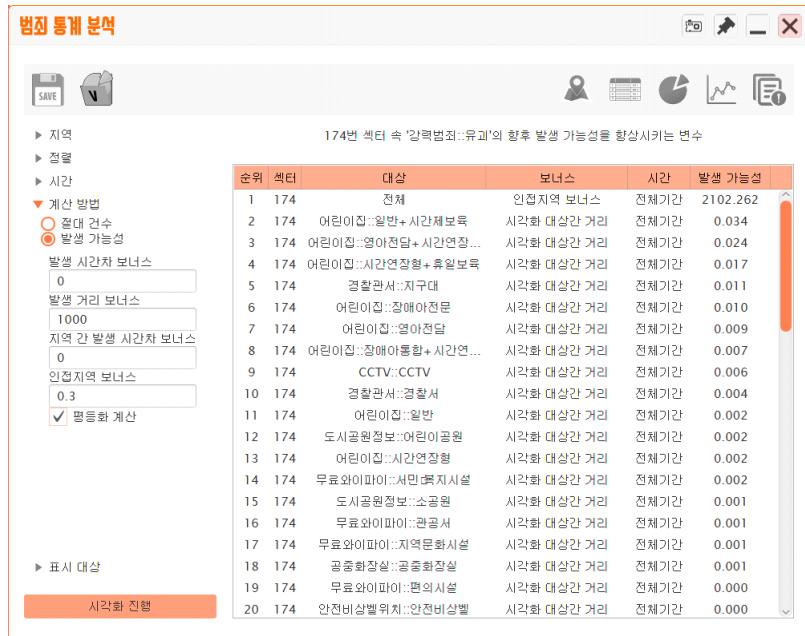
[그림 5-33] 향후 성폭행 사건이 발생할 가능성이 높은 지역

성폭행 가능성이 높은 지역은 앞선 실험과 다르게 해안가 및 도심지역을 집중적으로 조명하고 있었습니다. 특히 주전 해수욕장 인근인 140번 섹터가 가장 높은 수치를 기록하고 있었습니다.



[그림 5-32] 성폭행 사건 발생률이 가장 높았던 140번 지역 차트

놀랍게도 공중화장실 이외 다른 모든 변수들은 0 혹은 순위 밖으로 밀려 난 모습을 보였으며, 무료와이파이(관광)만이 수치 0.002를 기록하고 있을 뿐이었기에, 공중화장실 인근에 성폭행 사건을 발생 시킨 스크립트를 유추 할 수 있었습니다.



[그림 5-33] 유괴 사건 발생률이 가장 높았던 174번 지역 차트

유괴사건의 경우, 10위 안에 어린이집 항목이 6개를 차지하고 있었으며, 인접지역 보너스를 제외할 경우, 1, 2, 3위가 모두 어린이집이었습니다.

이외 유괴 발생률을 억제하는 CCTV 환경변수는 수치가 0.006으로, 앞전 실험과 마찬가지로 유괴 발생 위험을 줄이는데 별 다른 공헌을 하지 못하는 것을 볼 수 있었습니다.

이를 통해 CCTV가 유괴 사건을 줄여주는 억제 변수임은 알 수 없었지만, 유괴 사건이 어린이집을 대상으로 일어나는 스크립트임을 알 수 있었습니다.

5.3.4 시각화 종합 평가

평등화 계산 적용 이후, 시각화 결과를 통해 스크립트 내용을 대부분 유추 가능하게 되었습니다만, 실험 조건 설정 시 억제 강도를 10배 정도 더 높였더라면 더욱 선명한 결과를 얻을 수 있지 않았을까 하는 아쉬움이 남아 있습니다.

제 6 장 향후 개발방향

6.1 한계

실제 범죄 데이터를 구하지 못하여 임의 범죄를 생성하고, 생성 스크립트를 유추 해 나가는 실험밖에 할 수 없었기에 시간 오차 계산식을 사용하지 못하였습니다. 때문에 사건 발생 시각을 분석하여 연/월/일/시 단위로 분리하고, 특정 시간 혹은 특정 요일 어떤 사건이 어디서 발생 할 것인가에 대한 실험을 하지 못한 것이 아쉬웠습니다.

만일 실제 범죄 데이터를 구하게 된다면 시간대별 분석 기능을 사용함에 따라 사건 발생원인 유추 방법이 더욱 풍성해 질 것이며, 이를 통해 발생원인 환경변수 유추 및 억제 환경변수 적소 배치, 시간대별 위험 분포 확인 및 위험지수가 높은 골목 우회경로 생성 등 각종 데이터 분석 훈련에 도움 되는 프로그램이 될 수 있을 것입니다.

또한 보고서 제출 기한으로 인해 더욱 다양한 실험을 하지 못하였기에, 이후 실험에서는 앞서 언급한대로 억제 강도를 10배 이상 높여 뚜렷한 결과가 나오는지 살펴보고, 2000년부터 2020년 까지 해마다 범죄 발생 수치를 점차 높여 보면서 시간차 계산식이 제대로 작동하는지 확인 해 볼 것입니다.

6.2 향후 개발방향

Data, Network, AI 3가지 요건을 충족시키기 위해 더 많은 실험과 자문, 데이터 수집 등을 통하여 계산식을 수정/보완 해 나가 정확도를 상승시킬 계획이며, AI 부문으로 약한 수준의 인공지능을 개발하여 어디에 어떤 환경변수를 두어 발생확률을 높이거나 줄일지 스스로 계산하고 답을 도출하는 기능을 개발 할 것입니다.

Network 부문으로는 프로그램 결과물 및 데이터를 원격으로 공유하거나, 네트워크를 통해 계산거리를 서로 나눔으로 써 분산 연산을 통해

느린 시각화 계산 속도를 보완할 계획입니다.

모든 기능이 완벽하게 작동한다면, 소스코드를 모바일 버전으로 이식
할 계획입니다.