

엔지니어링 DB 및 실습 2조

코로나 바이러스 확진자 데이터를 이용한
집단 감염 분석 및 지역 별 대책 예측



아주대학교 산업공학과

201620233 이정운

201620170 최정훈

201620199 김태우

201720218 편재관



■ 목차

1. 문제 정의
2. 프로젝트 목적
3. 데이터베이스 설계
4. 상관 분석
5. 딥러닝
6. 시각화
7. 데이터베이스 구현
8. 결론 및 기대효과

1. 문제 정의

■ 문제 정의

1. 늘어나는 코로나 확진자 발생으로 사회적 혼란 야기
2. 전례 없는 감염병 유행으로 사례별 분석 필요성 대두
3. 코로나 바이러스 대처 의료 장비 및 시설 확보 필요
4. 한정된 의료인력 및 물자로 우선순위 및 예측 필요
5. 빠른 상황판단을 위한 정확한 DB 구축 필요



코로나로 변한 일상

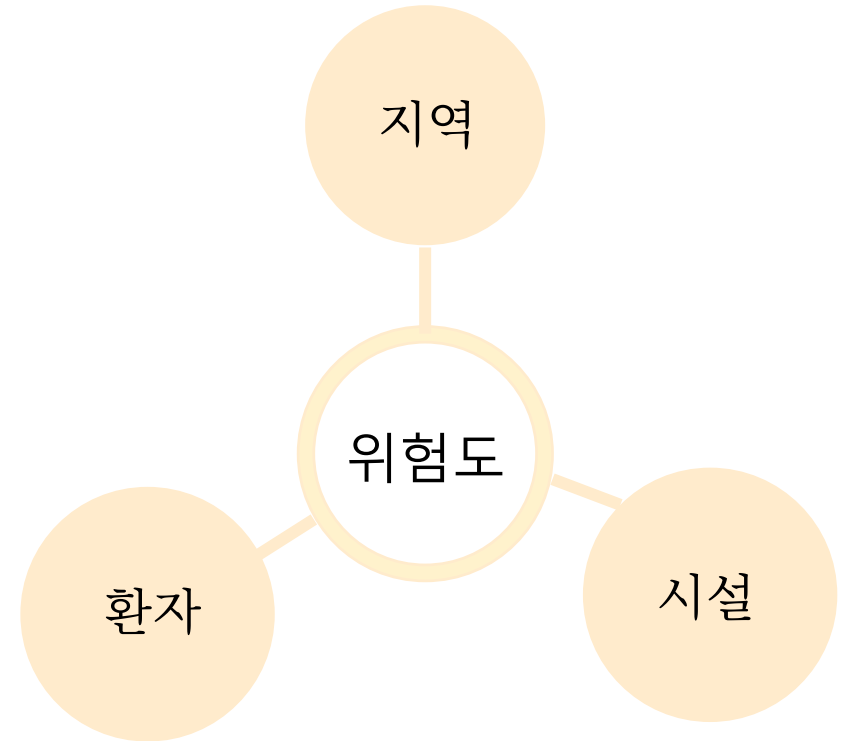
1. 문제 정의

■ 데이터 선별

개인과 지역의 특성과 전염과의 상관성
(환자의 나이, 동선, 지역별 통행량 등)

병원 등의 시설 정보를 이용해 대책 마련 시 사용

데이터들 간의 상관관계 및 규칙 규명



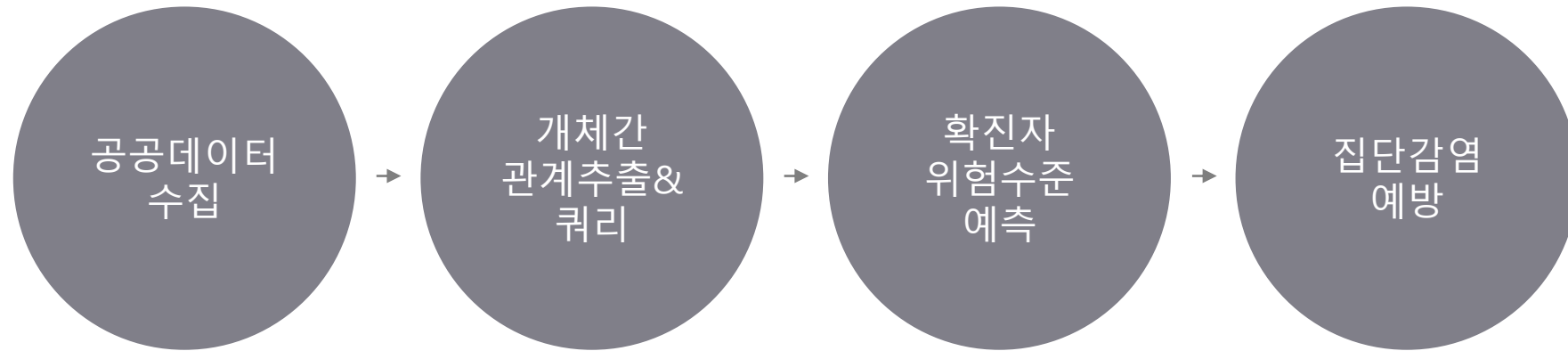
2. 프로젝트 목적

■ 해결아이디어 및 프로젝트 목적

1. 전국의 코로나 바이러스 확진자 데이터를 같은 포맷으로 입력 받아 통일된 포맷의 데이터를 사용하는 DB 구축
2. 신속하고 정확하게 상황파악을 할 수 있도록 분석을 위한 쿼리 작성
3. 딥러닝을 이용한 새로운 확진자의 지역의 위험도를 예측
4. 시각화를 이용하여 직관적으로 상황 전달
5. 예측한 위험도를 토대로 병실 수 확보 가능

2. 프로젝트 목적

■ 프로젝트 구상도



“

기존 코로나 바이러스 확진자들의 개인 및 지역별 특성을 파악하여
새로운 확진자의 해당 거주 지역에 최적의 대처를 할 수 있다.

”

3. 데이터베이스 설계

■ 공공 데이터 수집



kaggle™

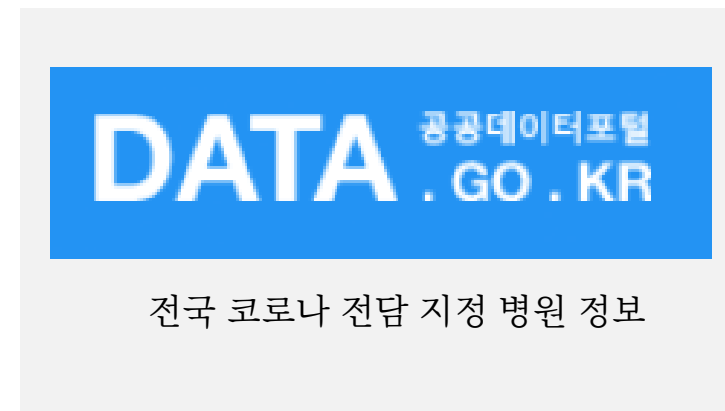
Data Science for COVID-19 (DS4C)
DS4C: Data Science for COVID-19 in South Korea

약 3518개의 확진자 개인 정보
약 7279개의 확진자 이동 경로
대한민국 지역 정보



새로운 경기 ▶ 공정한 세상
경기도교통정보센터

수도권 통행량 정보



DATA . GO . KR 공공데이터포털

전국 코로나 전담 지정 병원 정보

3. 데이터베이스 설계

■ 요구사항 분석

→ 명사(빨간색), 동사(초록색)을 찾는다.

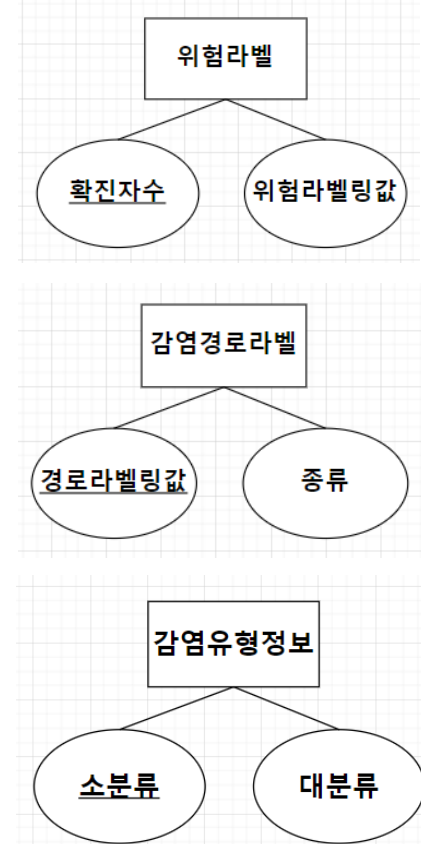
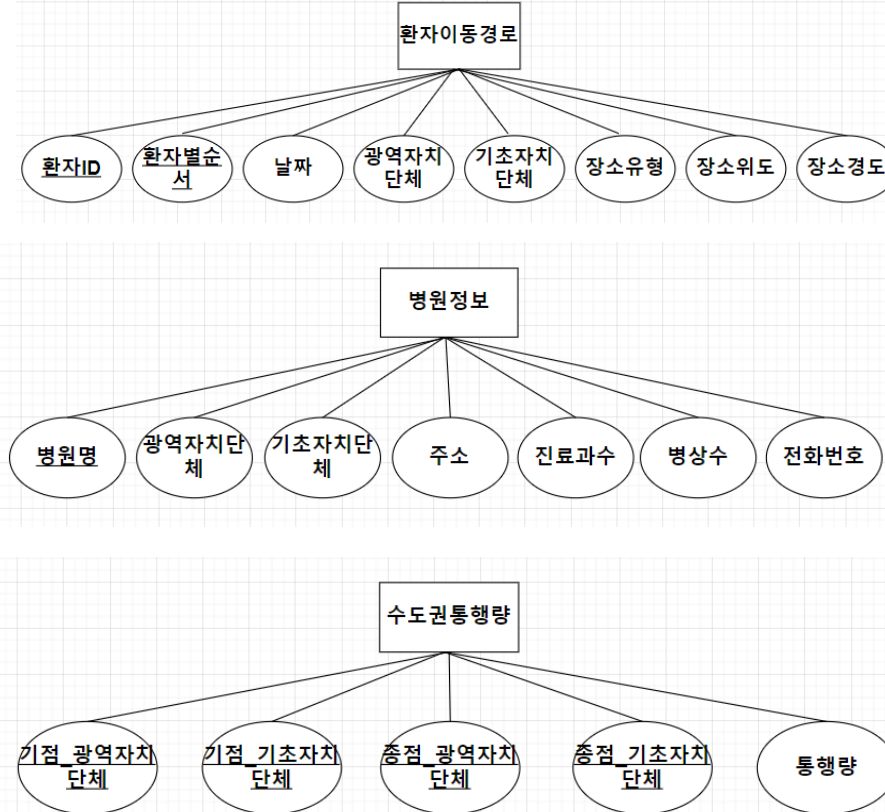
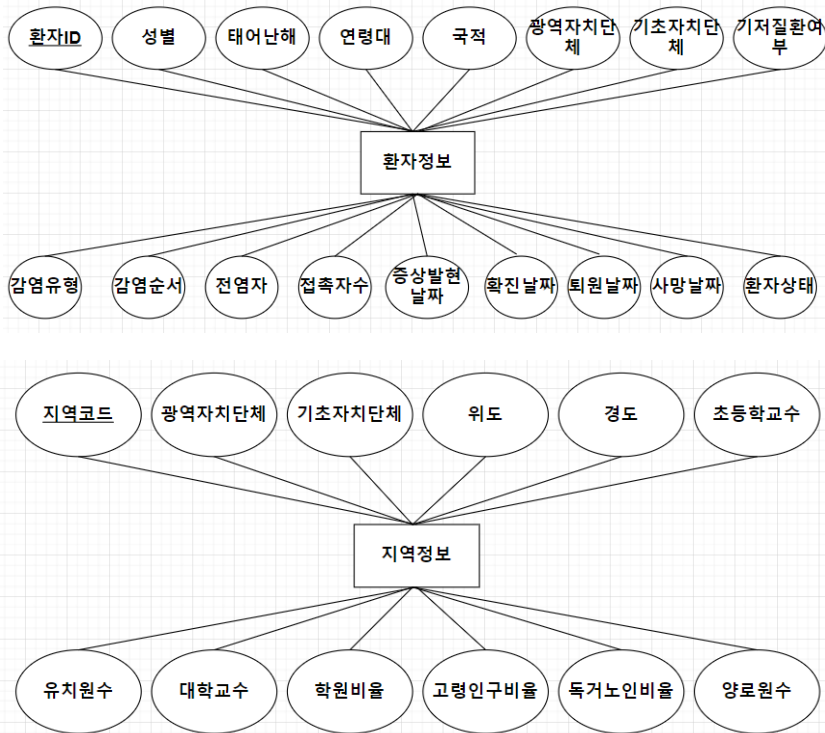
[요구사항 명세서]

- (1) 환자 명부에 등록되기 위해서는 기본정보로 환자ID, 성별, 태어난 해, 연령대, 국적, 기저질환여부가 입력되어야 한다.
- (2) 환자는 환자 ID로 식별한다.
- (3) 환자가 코로나에 걸린 사례(유형)와 검사를 받은 지역(광역시, 기초), 증상 발현 날짜, 확진받은 날짜, 확진자로 판정되기 전과 그 후에 접촉한 사람의 수, 집단 감염이 아니라 특정인에게 감염된 경우 감염자와, 몇 차 감염인지를 입력한다.
- (4) 환자의 상태 (자가격리, 격리 해제, 사망)와 격리되었다면 격리해제 날짜, 사망했다면 사망 날짜를 추가적으로 입력한다.
- (5) 환자 명부에는 지역정보를 포함해야 한다.
- (6) 따라서 환자ID, 검사 받은 장소(광역시), 상태는 필수적으로 유지하며, 나머지 속성들은 가능한 한 입력할 수 있도록 한다.
- (7) 감염유형의 체계적으로 관리하기 위해 감염유형사례를 관리하는 테이블을 만들어야 한다.
- (8) 감염의 구체적인 사례, 그 사례를 포괄할 수 있는 유형이 입력되어야 한다. (A교회, 교회), (B요양시설, 요양원)이 위의 예시이다.
- (9) 환자명부의 감염유형은 감염유형사례의 소분류만 가질 수 있다.
- (10) 여러 장소를 돌아다닌 환자의 정보를 관리하기 위해 경로 테이블을 생성해 환자 ID, 날짜, 장소 (광역시, 기초, 방문 목적)가 필요하며, 장소에 해당하는 위도, 경도도 추가적으로 입력한다.
- (11) 장소의 특정 이름이 공개되면 사생활의 피해가 발생할 수 있기에 유형(방문 목적)으로 대체한다.
- (12) 경로의 모든 값들은 유지되어야 한다.
- (13) 환자는 여러 지역을 방문할 수 있어서 환자 ID당 장소에 대해 일련번호를 부여해 환자 ID와 일련번호 두 개를 통해 정보들을 식별한다.
- (14) 지역에 대한 정보들은 광역, 기초, 그 지역을 대표하는 건물(시청 등)의 위도, 경도와 지역의 다중이용시설의 수들을 나타내는 정보로 구성된다.
- (15) 지역의 모든 값들은 유지되어야 한다.
- (16) 지역에서 광역, 기초에 해당하는 코드를 부여해 이 코드로 정보들을 식별한다.
- (17) 병원 정보는 기관명, 지역(광역시, 기초) 진료과 수, 병상 수, 전화번호, 주소가 입력되어야 한다.
- (18) 모든 값들은 유지되어야 한다.
- (19) 기관명으로 정보들을 식별하며, 필요한 경우 병원 코드를 부여해서 활용하도록 한다.
- (20) 이 경우 하나의 지역에 여러 병원을 갖게 된다.
- (21) 수도권의 지역별 통행량을 알기 위해 기점, 종점, 통행량 정보를 입력한다.
- (22) 기점, 종점을 통해 통행량을 식별할 수 있으며, 모든 정보들은 유지되어야 한다.
- (23) 수도권의 특정 지역에서, 나머지의 지역의 통행량을 갖고 있어야 한다.
- (24) 위험 라벨은 확진자 수를 통해 위험도를 식별한다.
- (25) 감염경로 라벨은 라벨링 번호로 종류를 식별한다.
- (26) 환자정보, 지역정보, 위험라벨, 감염경로라벨을 통해 필요한 정보들을 추출해 분석을 진행한다.
- (27) 분석 진행에 앞서 환자정보, 지역정보를 통해 지역 당 확진자 수를 구해서 지역이 갖는 고유한 속성들이 확진자 수에 영향을 미치는지 판단해야 한다.

3. 데이터베이스 설계

■ 요구사항 분석

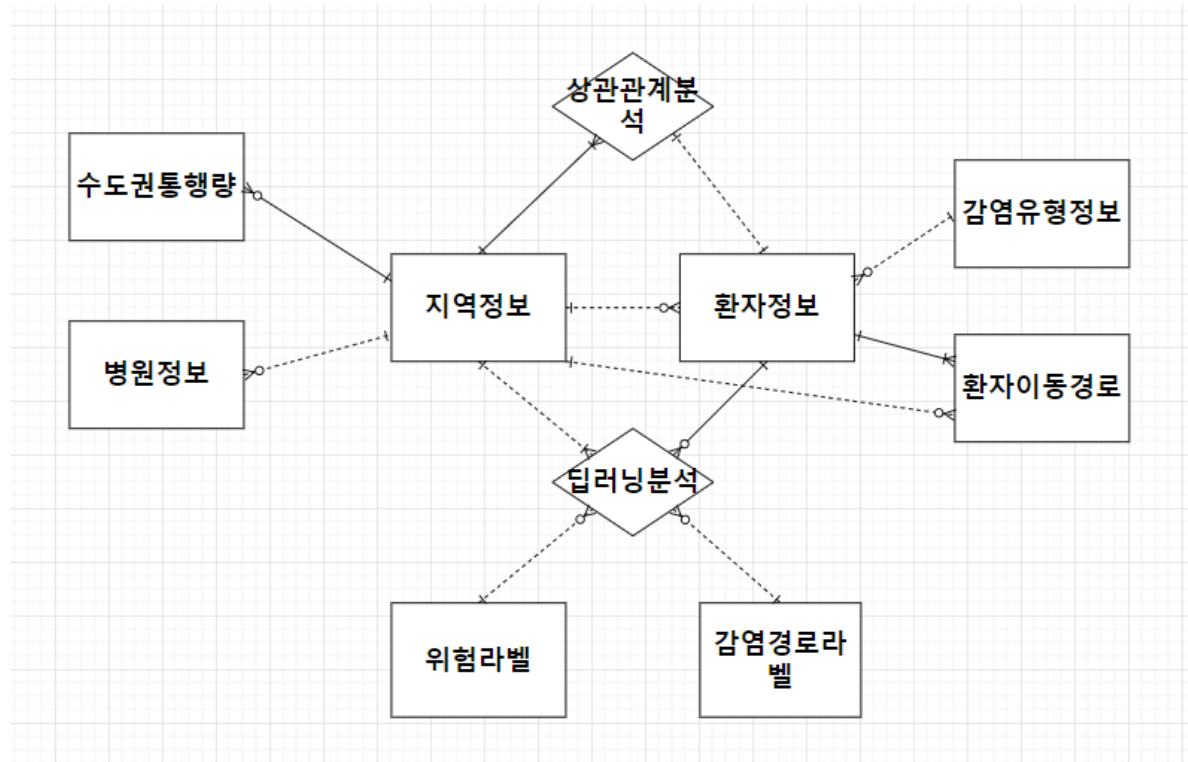
→ 개념적 설계: E-R 다이어그램



3. 데이터베이스 설계

■ 요구사항 분석

→ 개념적 설계: E-R 다이어그램



3. 데이터베이스 설계

■ 논리적 모델링

* 5가지 규칙에 따라 릴레이션으로 변환하는 작업

- # 규칙 1 : 모든 개체는 릴레이션으로 변환한다.
- # 규칙 2 : 다대다(N:M) 관계는 릴레이션으로 변환한다.
- # 규칙 3 : 일대다(1:n) 관계는 외래키로 표현한다.
- # 규칙 4 : 일대일(1:1) 관계는 외래키로 표현한다.
- # 규칙 5 : 다중 값 속성은 릴레이션으로 변환한다.

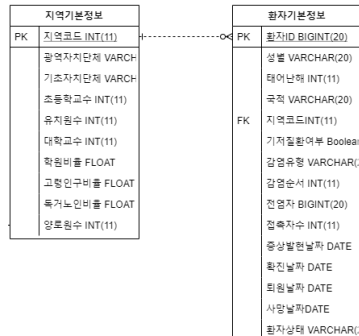


* 모든 테이블 제 4 정규형까지 만족

- # 제 1 정규형
- # 제 2 정규형
- # 제 3 정규형
- # BCNF 정규형
- # 제 4 정규형

예시

개체1	관계			개체2
지역기본정보	1	(1:N)	1	환자정보
PK(지역코드)	강한 비식별			PK(환자ID) FK(지역코드)



예시

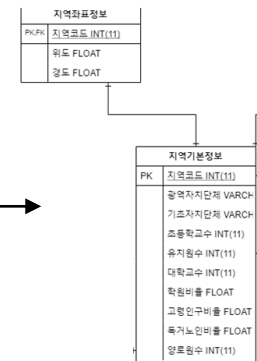
이행적 함수 종속 (제 3 정규형 위반)

(지역 코드) → (위도, 경도)

(광역자치단체, 지역자치단체) → (위도, 경도)

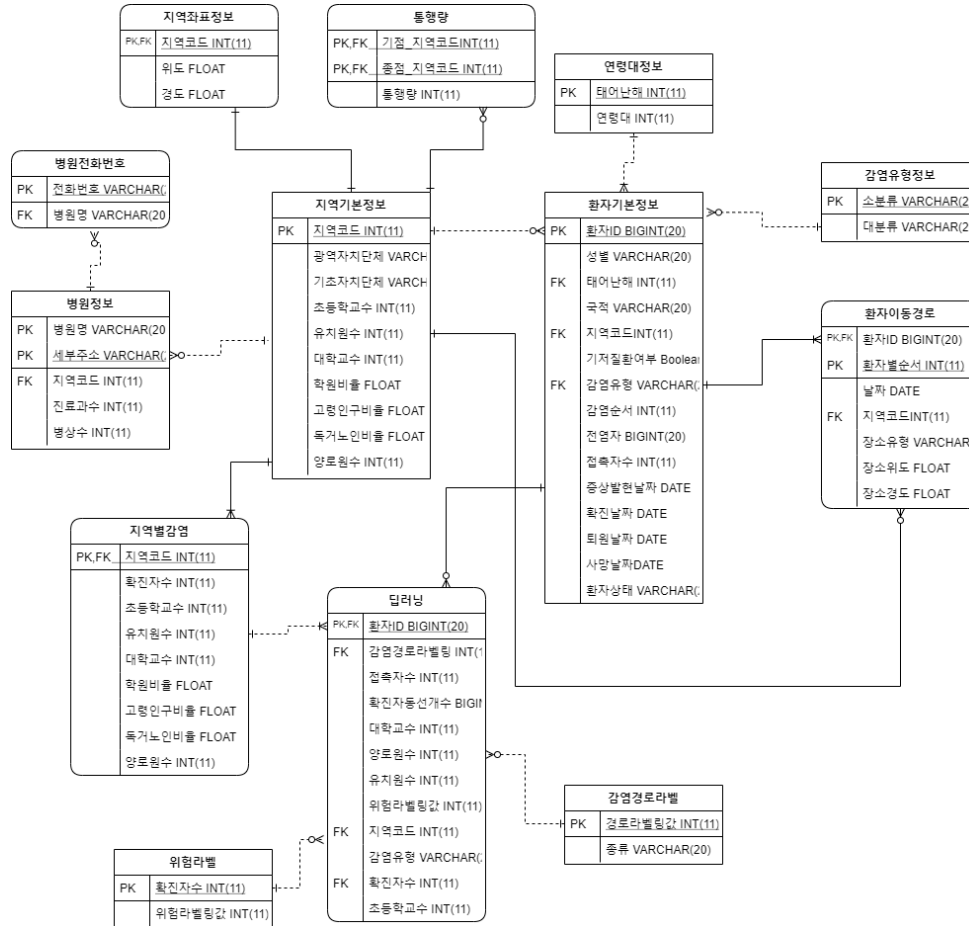
(지역 코드) → (광역자치단체, 지역자치단체)

지역코드	위도	경도	초등학교수	유치원수	대학교수	병원비율	노인인구비율	독거노인비율	알로원수
10000	37.566953	126.977977	607	830	48	1.44	15.38	5.8	22739
10010	37.518421	127.047222	33	38	0	4.18	13.17	4.3	3088
10020	37.530492	127.123837	27	32	0	1.54	14.55	5.4	1023
10030	37.639938	127.025508	14	21	0	0.67	19.49	8.5	628
10040	37.551166	126.849506	36	56	1	1.17	14.39	5.7	1080
10050	37.47829	126.951502	22	33	1	0.89	15.12	4.9	909
10060	37.538712	127.082366	22	33	3	1.16	13.75	4.8	723
10070	37.495632	126.88765	26	34	3	1	16.21	5.7	741
10080	37.456852	126.895229	18	19	0	0.96	16.15	6.7	475
10090	37.654259	127.056294	42	66	6	1.39	15.4	7.4	952
10100	37.668952	127.047082	23	26	1	0.95	17.89	7.2	485
10110	37.574552	127.039721	21	31	4	1.06	17.26	6.7	832
10120	37.510571	126.963604	21	34	3	1.17	15.85	5.2	762
10130	37.566283	126.901644	22	24	2	1.83	14.05	4.9	929
10140	37.579428	126.936771	19	25	6	1.12	16.77	6.2	587
10150	37.483804	127.032693	24	27	1	2.6	13.39	3.8	1465
10160	37.563277	127.036647	21	30	2	0.97	14.76	5.3	593
10170	37.589562	127.0167	29	49	6	1.02	16.15	6	729
10180	37.51462	127.106141	40	51	1	1.65	13.1	4.1	1527



3. 데이터베이스 설계

■ 논리적 모델링



[지역기본정보 - 환자기본정보]

#규칙3	지역기본정보	1	(1:N)	0	환자기본정보
	PK(지역코드)		강한 비식별		PK(환자ID) FK(지역코드)

[환자기본정보-환자이동경로]

#규칙3	환자정보	1	(1:N)	1	환자이동경로
	PK(환자ID)		약한 식별		PK[FK(환자ID), 환자별 순서]

[지역기본정보-지역별감염]

#규칙4	지역기본정보	1	(1:1)	1	지역별감염
	PK(지역코드)		약한 식별		PK[FK(지역코드)]

[지역기본정보-지역좌표정보]

제 3정규형 위반 → 제 4정규형 만족	지역기본정보	1	(1:1)	1	지역좌표정보
	PK(지역코드)		약한 식별		PK[FK(지역코드)]

[논리적 모델링 후 최종 구현할 DB 정보]



3. 데이터베이스 설계

물리적 모델링

Create Table을 사용하여 테이블 생성

```
40 • create table 환자기본정보(  
41   환자ID bigint not null,  
42   성별 varchar(20),  
43   태어난해 int,  
44   국적 varchar(20),  
45   광역자치단체 varchar(20) not null,  
46   기초자치단체 varchar(20),  
47   거주질현여부 boolean,  
48   감염유형 varchar(20),  
49   감염순서 int,  
50   전염자 bigint,  
51   접촉자수 int,  
52   증상발현날짜 date,  
53   확진날짜 date not null,  
54   퇴원날짜 date,  
55   사망날짜 date,  
56   환자상태 varchar(20) not null,  
57   primary key (환자ID),  
58   check (성별 = 남자 or 성별 = 여자 or 성별 = ''),  
59   check (환자상태 = 입원 or 환자상태 = 퇴원 or 환자상태 = 사망),  
60   foreign key (감염유형) references 감염유형정보 (소분류),  
61   foreign key (태어난해) references 연령대정보 (태어난해)  
62 );
```

#지역코드 부여 및 삭제, 외래키 설정

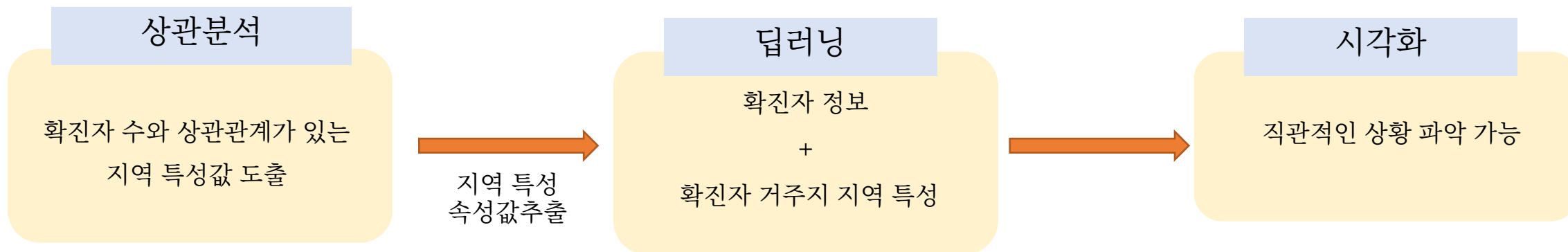
```
5 • alter table 환자기본정보 add 지역코드 int;  
6  
7 • alter table 환자기본정보 add constraint 지역코드외래키  
8   foreign key (지역코드) references 지역기본정보(지역코드);  
9  
10 • update 환자기본정보 p  
11   join 지역기본정보 r on r.광역자치단체 = p.광역자치단체 and r.기초자치단체 = p.기초자치단체  
12   set p.지역코드 = r.지역코드;  
13  
14 • alter table 환자기본정보 modify 지역코드 int not null;  
15  
16 • alter table 환자기본정보 drop 광역자치단체;  
17 • alter table 환자기본정보 drop 기초자치단체;  
18  
19 • alter table 환자기본정보 modify 지역코드 int after 국적;  
20  
23 • alter table 병원정보 add 지역코드 int;  
24  
25 • alter table 병원정보 add constraint 병원_지역코드외래키  
26   foreign key (지역코드) references 지역기본정보(지역코드);  
27  
28 • update 병원정보 h  
29   join 지역기본정보 r on r.광역자치단체 = h.광역자치단체 and r.기초자치단체 = h.기초자치단체  
30   set h.지역코드 = r.지역코드;  
31  
32 • alter table 병원정보 modify 지역코드 int not null;  
33  
34 • alter table 병원정보 drop 광역자치단체;  
35 • alter table 병원정보 drop 기초자치단체;  
36  
37 • alter table 병원정보 modify 지역코드 int after 병원명;  
38
```

#SQL JOIN을 이용하여 데이터 검색 후 테이블 생성

```
10 • create table 지역별감염 as  
11   SELECT Count(환자ID) AS 확진자수,  
12   환자기본정보.지역코드, 지역기본정보.초등학교수, 지역기본정보.유치원수, 지역기본정보.대학교수, 지역기본정보.학원비율,  
13   지역기본정보.고령인구비율, 지역기본정보.독거노인비율, 지역기본정보.양로원수  
14   FROM 환자기본정보 INNER JOIN 지역기본정보 ON (환자기본정보.지역코드 = 지역기본정보.지역코드)  
15   where 환자기본정보.지역코드 not in (10000, 11000, 12000, 13000, 14000, 15000, 16000,  
16   30000, 40000, 41000, 50000, 51000, 60000, 61000, 80000)  
17   GROUP BY 환자기본정보.지역코드, 지역기본정보.초등학교수, 지역기본정보.유치원수,  
18   지역기본정보.대학교수, 지역기본정보.학원비율, 지역기본정보.고령인구비율, 지역기본정보.독거노인비율, 지역기본정보.양로원수;  
19  
20 • alter table 지역별감염 modify 확진자수 int;  
21  
22 • alter table 지역별감염 add  
23   primary key (지역코드);  
24  
25 • alter table 지역별감염 add constraint 지역별감염_지역외래키  
26   foreign key (지역코드) references 지역기본정보 (지역코드);  
27  
28 • create table 입력정보 as  
29   SELECT 위항라벨,위항라벨링값, 감염경로라벨,경로라벨링값 AS 감염경로라벨링, 환자기본정보.접촉자수,  
30   Count(환자ID동경로,환자ID) AS 확진자동선개수, 지역별감염.대학교수, 지역별감염.양로원수, 지역별감염.유치원수, 환자기본정보.환자ID,  
31   환자기본정보.지역코드, 환자기본정보.감염유형, 지역별감염.확진자수 AS 확진자수, 지역별감염.초등학교수  
32   FROM (((지역별감염 INNER JOIN  
33   (환자ID동경로 RIGHT JOIN 환자기본정보 ON 환자ID동경로.환자ID = 환자기본정보.환자ID)  
34   ON (지역별감염.지역코드 = 환자기본정보.지역코드))  
35   INNER JOIN 감염유형정보 ON 환자기본정보.감염유형 = 감염유형정보.소분류)  
36   INNER JOIN 감염경로라벨 ON 감염유형정보.대분류 = 감염경로라벨.종류)  
37   INNER JOIN 위항라벨 ON 지역별감염.확진자수 = 위항라벨.확진자수)  
38   GROUP BY 위항라벨,위항라벨링값, 감염경로라벨,경로라벨링값, 환자기본정보.접촉자수, 지역별감염.대학교수, 지역별감염.양로원수,  
39   지역별감염.초등학교수, 지역별감염.유치원수, 환자기본정보.지역코드, 환자기본정보.환자ID, 환자기본정보.감염유형, 지역별감염.확진자수  
40   HAVING (((환자기본정보.감염유형) <> 해외유입))  
41   ORDER BY 환자기본정보.환자ID, 지역별감염.확진자수 DESC;  
42
```

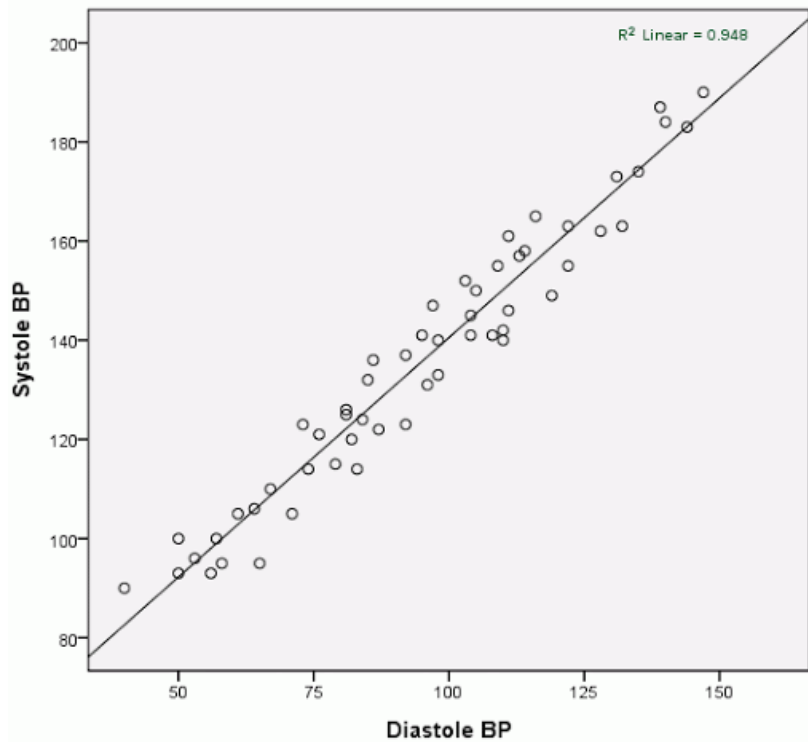
4. 분석

■ 분석 구상도



■ 상관분석

→ 변수 간의 밀접한 정도, 즉 상관관계를 분석하는 통계적 분석 방법



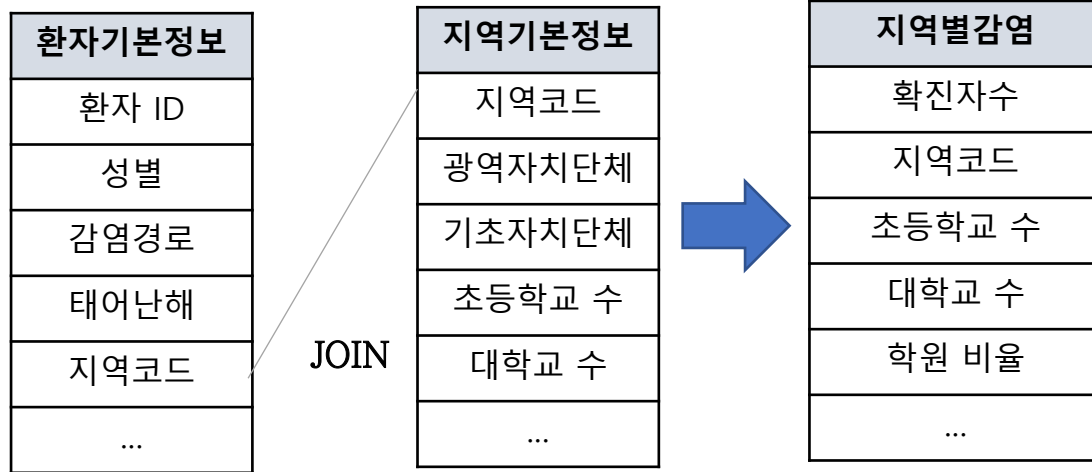
[상관분석 해석(KSI 통계 연구 기반)]

상관계수	상관관계
± 0.9 이상	상관관계가 아주 높다
$\pm 0.7 \sim 0.9$	상관관계가 높다
$\pm 0.4 \sim 0.7$	상관관계가 있다
$\pm 0.2 \sim 0.4$	상관관계가 있으나 낮다
± 0.2 미만	상관관계가 거의 없다

*상관계수의 절댓값 > 0.2 사용

4. 상관분석

전처리



	확진 자수	지역 코드	초등학 교수	유치 원수	대학 교수	학원 비율	고령인구 비율	독거노인 비율	양로 원수
▶	46	10000	607	830	48	1.44	15.38	5.8	22739
	69	10010	33	38	0	4.18	13.17	4.3	3088
	17	10020	27	32	0	1.54	14.55	5.4	1023
	8	10030	14	21	0	0.67	19.49	8.5	628
	31	10040	36	56	1	1.17	14.39	5.7	1080
	52	10050	22	33	1	0.89	15.12	4.9	909
	11	10060	22	33	3	1.16	13.75	4.8	723
	34	10070	26	34	3	1	16.21	5.7	741
	13	10080	18	19	0	0.96	16.15	6.7	475

[상관분석에 사용될 최종 테이블]

[SQL 쿼리]

```

10 • create table 지역별감염 as
11 SELECT Count(환자기본정보.환자ID) AS 확진자수,
12 환자기본정보.지역코드, 지역기본정보.초등학교수, 지역기본정보.유치원수, 지역기본정보.대학교수, 지역기본정보.학원비율,
13 지역기본정보.고령인구비율, 지역기본정보.독거노인비율, 지역기본정보.양로원수
14 FROM 환자기본정보 INNER JOIN 지역기본정보 ON (환자기본정보.지역코드 = 지역기본정보.지역코드)
15 where 환자기본정보.지역코드 not in (10000, 11000, 12000, 13000, 14000, 15000, 16000,
16 30000, 40000, 41000, 50000, 51000, 60000, 61000, 80000)
17 GROUP BY 환자기본정보.지역코드, 지역기본정보.초등학교수, 지역기본정보.유치원수,
18 지역기본정보.대학교수, 지역기본정보.학원비율, 지역기본정보.고령인구비율, 지역기본정보.독거노인비율, 지역기본정보.양로원수;

```


4. 상관분석

상관분석 결과

Code

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sat Jun 20 11:43:08 2020
4
5 @author: Lee_j
6 """
7
8 import pandas as pd
9
10 data1 = pd.read_csv('지역별강원_수정.csv')
11 print(data1)
12
13 del data1['지역코드']
14
15 corr = data1.corr(method = 'pearson')
16 print(corr)
17
```

[상관분석 코드]

*가설 설계

- 귀무가설(H0) : 모 집단에서 두 변수 사이에 상관성이 없다.
- 대립가설(H1) : 모 집단에서 두 변수 사이에 상관성이 있다.

```
17 dropnan = data1.dropna(axis = 0)
18 import scipy.stats as stats
19 corr1 = stats.pearsonr(dropnan.대학교수, dropnan.확진자수)
20 corr2 = stats.pearsonr(dropnan.양로원수, dropnan.확진자수)
21 corr3 = stats.pearsonr(dropnan.유치원수, dropnan.확진자수)
22 print("\n\n(상관계수, p-value)")
23 print("1. 확진자 수 : 대학교 수")
24 print(corr1)
25 print("2. 확진자 수 : 양로원 수")
26 print(corr2)
27 print("3. 확진자 수 : 유치원 수")
28 print(corr3)
29
```

[P value 검증 코드]

Index	확진자수	초등학교수	유치원수	대학교수	학원비율	고령인구비율	독거노인비율	양로원수
확진자수	1	0.17052	0.211439	0.424429	0.0743238	-0.110211	-0.12058	0.209945
초등학교수	0.17052	1	0.960761	0.616137	0.413299	-0.477675	-0.438806	0.683729
유치원수	0.211439	0.960761	1	0.586487	0.405071	-0.533751	-0.489665	0.694326
대학교수	0.424429	0.616137	0.586487	1	0.271466	-0.37866	-0.38025	0.413611
학원비율	0.0743238	0.413299	0.405071	0.271466	1	-0.553284	-0.507151	0.565625
고령인구비율	-0.110211	-0.477675	-0.533751	-0.37866	-0.553284	1	0.975833	-0.538232
독거노인비율	-0.12058	-0.438806	-0.489665	-0.38025	-0.507151	0.975833	1	-0.560337
양로원수	0.209945	0.683729	0.694326	0.413611	0.565625	-0.538232	-0.560337	1

(상관계수, p-value)

1. 확진자 수 : 대학교 수
(0.4244292333941744, 9.834390703720952e-09)
2. 확진자 수 : 양로원 수
(0.20994484209258235, 0.006306801028135561)
3. 확진자 수 : 유치원 수
(0.21143914096613325, 0.005936293155719786)

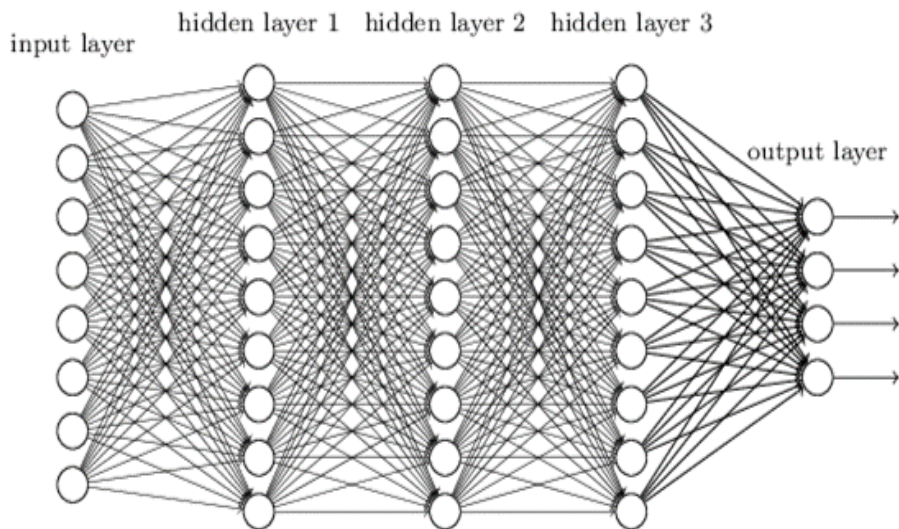
→ 상관계수의 절댓값이 0.2보다 큰 변수인 유치원수, 대학교수, 양로원수에 대해서 P value의 값이 0.05 이하의 값을 가져 유의 수준보다 작으므로 상관성이 있음을 알 수 있다.

→ 유치원 수, 대학교 수, 양로원 수를 딥러닝의 특성값으로 사용한다.

■ 딥러닝

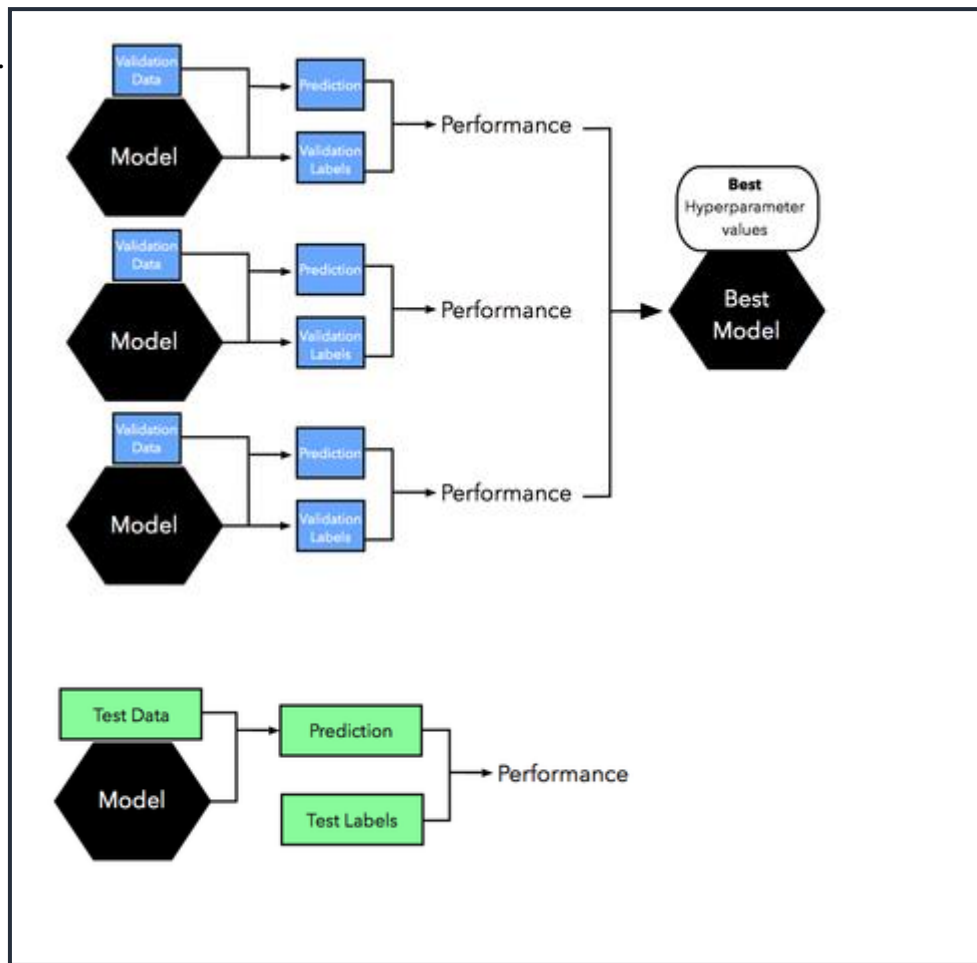
→ 샘플들의 데이터에서 통계적 구조를 찾아 작업을 자동화 하기 위한 규칙을 만든다.

Deep neural network

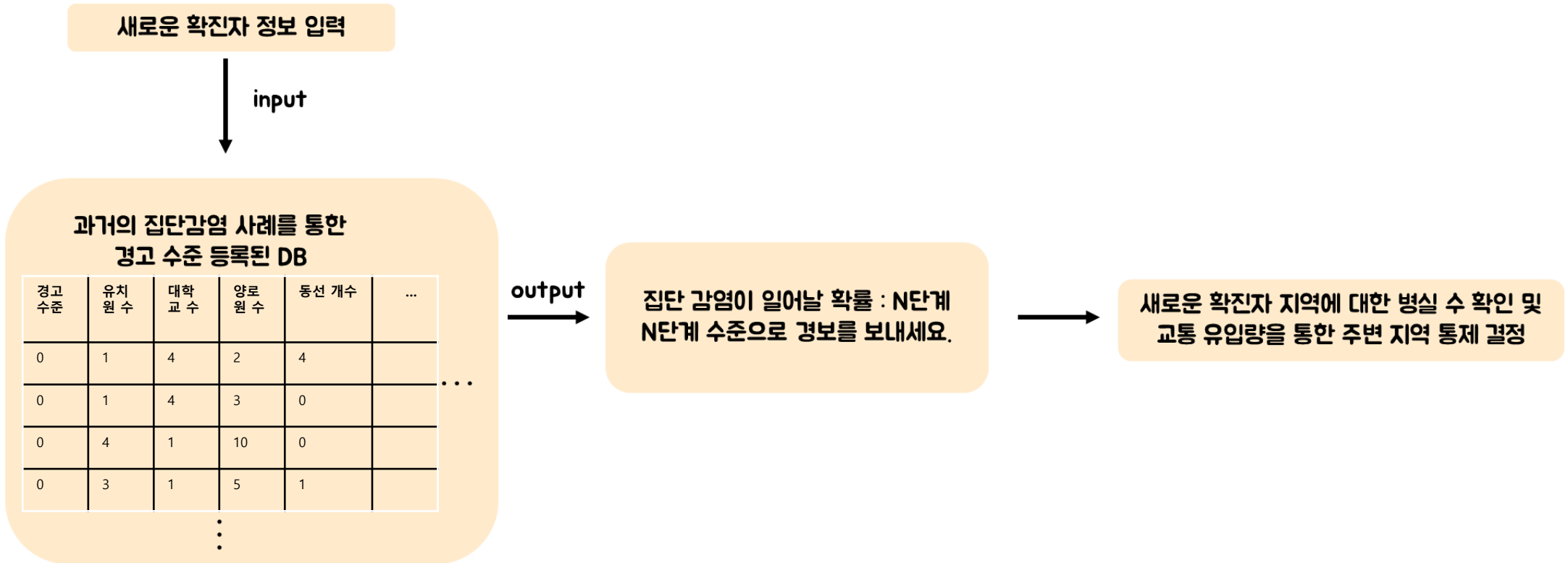


→ 연속된 Layer에서 점진적으로 의미 있는 표현 학습

→ 데이터로부터 표현을 학습



■ 시스템 구상도



■ 사용된 테이블 및 전처리

*전염에 영향을 끼치는 이전 확진자 정보

환자이동경로
환자 ID
환자별순서
날짜
지역코드
장소
...

환자기본정보
환자ID
성별
국적
감염유형
지역코드
...

JOIN

지역별감염
지역코드
확진자수
초등학교수
유치원수
대학교수
...

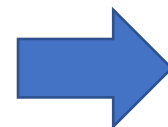
위험라벨
확진자수
위험라벨링값

*확진자 수와 상관성이 있는 지역 특성

감염유형정보
소분류
대분류

감염경로라벨
경로라벨링
종류

* 딥러닝에 사용하기 위하여 속성 값을 연속형 범주로 변경



딥러닝분석테이블
위험라벨링값
감염경로라벨링
접촉자수
확진자동선개수
대학교수
양로원수
유치원수
...

■ 최종 딥러닝 테이블

[SQL 쿼리]

```
22 • create table 딥러닝 as
23 SELECT 위험라벨, 위험라벨링값, 감염경로라벨, 경로라벨링값 AS 감염경로라벨링, 환자기본정보, 접촉자수,
24 Count(환자이동경로, 환자ID) AS 확진자동선개수, 지역별감염, 대학교수, 지역별감염, 양로원수, 지역별감염, 유치원수, 환자기본정보, 환자ID,
25 환자기본정보, 지역코드, 환자기본정보, 감염유형, 지역별감염, 확진자수 AS 확진자수, 지역별감염, 초등학교수
26 FROM (((지역별감염 INNER JOIN
27 (환자이동경로 RIGHT JOIN 환자기본정보 ON 환자이동경로, 환자ID = 환자기본정보, 환자ID)
28 ON (지역별감염, 지역코드 = 환자기본정보, 지역코드))
29 INNER JOIN 감염유형정보 ON 환자기본정보, 감염유형 = 감염유형정보, 소분류)
30 INNER JOIN 감염경로라벨 ON 감염유형정보, 대분류 = 감염경로라벨, 종류)
31 INNER JOIN 위험라벨 ON 지역별감염, 확진자수 = 위험라벨, 확진자수)
32 GROUP BY 위험라벨, 위험라벨링값, 감염경로라벨, 경로라벨링값, 환자기본정보, 접촉자수, 지역별감염, 대학교수, 지역별감염, 양로원수,
33 지역별감염, 초등학교수, 지역별감염, 유치원수, 환자기본정보, 지역코드, 환자기본정보, 환자ID, 환자기본정보, 감염유형, 지역별감염, 확진자수
34 HAVING (((환자기본정보, 감염유형) <> 제외유입))
35 ORDER BY 환자기본정보, 환자ID, 지역별감염, 확진자수 DESC;
```

[딥러닝에 사용될 최종 테이블]

	위험라벨 링값	감염경로라 벨링	접촉 자수	확진자동선 개수	대학 교수	양로 원수	유치 원수	환자ID	지역 코드	감염유형	확진 자수	초등학 교수
▶	0	3	17	2	3	668	17	1000000003	10230	환자 접촉	17	13
	0	3	2	1	6	729	49	1000000005	10170	환자 접촉	27	29
	0	3	43	1	3	668	17	1000000006	10230	환자 접촉	17	13
	0	3	0	1	3	668	17	1000000007	10230	환자 접촉	17	13
	0	3	6	4	6	729	49	1000000010	10170	환자 접촉	27	29
	0	3	117	19	3	668	17	1000000013	10230	환자 접촉	17	13
	0	3	27	13	3	668	17	1000000014	10230	환자 접촉	17	13
	0	4	8	9	2	593	30	1000000015	10160	성동구 아파트	22	21
	0	3	0	2	3	668	17	1000000016	10230	환자 접촉	17	13
	0	3	0	1	3	668	17	1000000017	10230	환자 접촉	17	13
	0	3	0	0	3	668	17	1000000019	10230	환자 접촉	17	13

■ 모델 설계

Drop out을 사용한 Neural Network
(실행 환경 : 구글 colab)

[Input Data 정보]

Data 수 : 총 2796개

Feature 수 : 6개

Label 수 : 3개

Train Data : Test Data = 7:3

<모델 환경설정>

- Hyperparameter

Learning Rate : 0.0005

Batch size : 3

Epochs : 130

Nb_classes : 3

Drop rate : 0.005

- Layer

Relu 4개

Softmax 1개

Optimizer = Adam 사용

```
import numpy as np
import random
import tensorflow as tf
import pandas as pd
from sklearn.model_selection import train_test_split

random.seed(777) # for reproducibility
#하이퍼 파라미터 설정
learning_rate = 0.0005
batch_size = 3
training_epochs = 130
nb_classes = 3
drop_rate = 0.005

data1 = pd.read_excel('지역감염현황분석용.xlsx')
print(data1)

x = data1.iloc[:,1:7]
y = data1[['label']]
x_train, x_test, y_train, y_test = train_test_split(x,y)
x_train.shape, x_test.shape, y_train.shape, y_test.shape

x_train = x_train.to_numpy()
print(type(x_train))
x_test = x_test.to_numpy()
print(type(x_test))
y_train = tf.keras.utils.to_categorical(y_train, nb_classes)
y_test = tf.keras.utils.to_categorical(y_test, nb_classes)

tf.model = tf.keras.Sequential()

tf.model.add(tf.keras.layers.Dense(input_dim=6, units=70, kernel_initializer='glorot_normal', activation='relu'))
tf.model.add(tf.keras.layers.Dropout(drop_rate))
tf.model.add(tf.keras.layers.Dense(units=70, kernel_initializer='glorot_normal', activation='relu'))
tf.model.add(tf.keras.layers.Dropout(drop_rate))
tf.model.add(tf.keras.layers.Dense(units=70, kernel_initializer='glorot_normal', activation='relu'))
tf.model.add(tf.keras.layers.Dropout(drop_rate))
tf.model.add(tf.keras.layers.Dense(units=70, kernel_initializer='glorot_normal', activation='relu'))
tf.model.add(tf.keras.layers.Dropout(drop_rate))
tf.model.add(tf.keras.layers.Dense(units=nb_classes, kernel_initializer='glorot_normal', activation='softmax'))
tf.model.compile(loss='categorical_crossentropy',
                  optimizer=tf.keras.optimizers.Adam(lr=learning_rate), metrics=['accuracy'])
tf.model.summary()

history = tf.model.fit(x_train, y_train, epochs = training_epochs, batch_size = batch_size, validation_data=(x_test, y_test))
```

[딥러닝 학습]

```
import matplotlib.pyplot as plt

history_dict = history.history
loss = history_dict['loss']
val_loss = history_dict['val_loss']

epochs = range(1, len(loss) + 1)

plt.plot(epochs, loss, 'bo', label='Training loss') # 'bo'는 파란색 점을 의미
plt.plot(epochs, val_loss, 'r', label='Validation loss') # 'r'는 파란색 실선을 의미
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.rcParams['figure.figsize'] = (30,20)
plt.legend()

plt.show()

plt.clf() # 그래프를 초기화합니다.
acc = history_dict['accuracy']
val_acc = history_dict['val_accuracy']

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'r', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

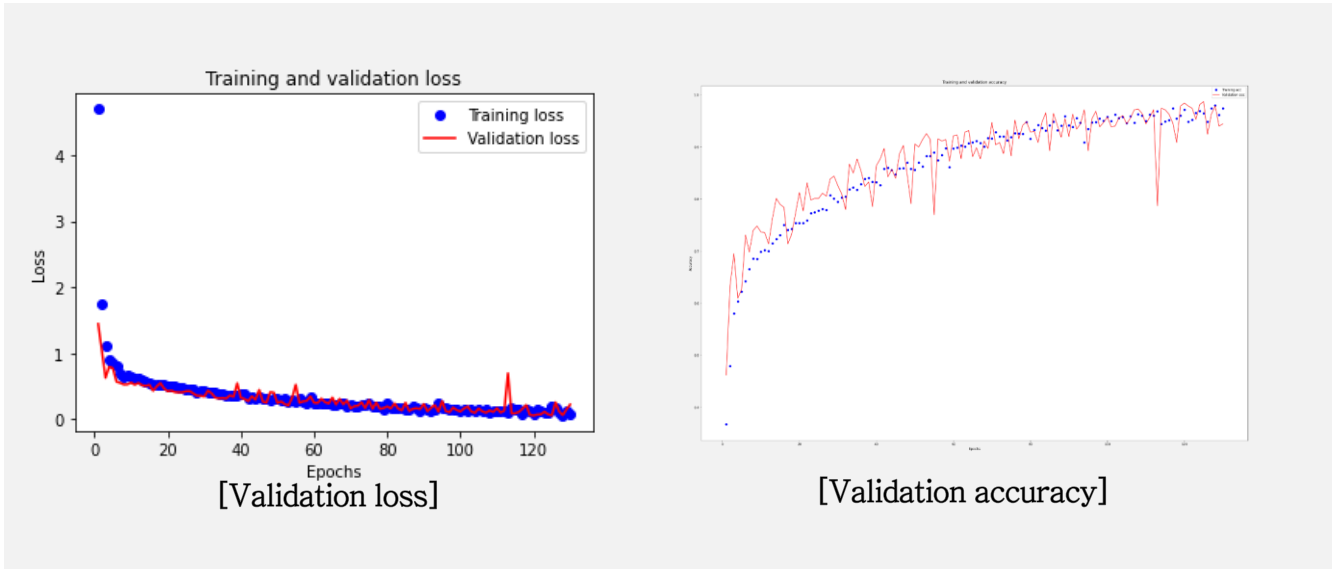
plt.show()

# predict 10 random hand-writing data
y_predicted = tf.model.predict(x_test)
for x in range(0, 10):
    random_index = random.randint(0, x_test.shape[0]-1)
    print("index: ", random_index,
          "actual y: ", np.argmax(y_test[random_index]),
          "predicted y: ", np.argmax(y_predicted[random_index]))

# evaluate test set
evaluation = tf.model.evaluate(x_test, y_test)
print('loss: ', evaluation[0])
print('accuracy', evaluation[1])
```

[Validation 그래프 출력]

■ 딥러닝 학습 결과



```

index: 234 actual y: 2 predicted y: 2
index: 456 actual y: 1 predicted y: 1
index: 456 actual y: 1 predicted y: 1
index: 377 actual y: 0 predicted y: 0
index: 589 actual y: 2 predicted y: 2
index: 276 actual y: 0 predicted y: 0
index: 344 actual y: 2 predicted y: 2
index: 550 actual y: 2 predicted y: 2
index: 40 actual y: 1 predicted y: 1
index: 456 actual y: 1 predicted y: 1
22/22 [=====] - 0s 1ms/step - loss: 0.2197 - accuracy: 0.9442
loss: 0.21970495581626892
accuracy 0.9442059993743896
    
```

[Test 결과]

```

Epoch 114/130
699/699 [=====] - 1s 2ms/step - loss: 0.1653 - accuracy: 0.9437 - val_loss: 0.0836 - val_accuracy: 0.9742
Epoch 115/130
699/699 [=====] - 1s 2ms/step - loss: 0.1487 - accuracy: 0.9490 - val_loss: 0.0885 - val_accuracy: 0.9714
Epoch 116/130
699/699 [=====] - 1s 2ms/step - loss: 0.1397 - accuracy: 0.9513 - val_loss: 0.1060 - val_accuracy: 0.9628
Epoch 117/130
699/699 [=====] - 1s 2ms/step - loss: 0.0770 - accuracy: 0.9742 - val_loss: 0.1575 - val_accuracy: 0.9456
Epoch 118/130
699/699 [=====] - 1s 2ms/step - loss: 0.1545 - accuracy: 0.9547 - val_loss: 0.2127 - val_accuracy: 0.9084
Epoch 119/130
699/699 [=====] - 1s 2ms/step - loss: 0.1176 - accuracy: 0.9604 - val_loss: 0.0617 - val_accuracy: 0.9785
Epoch 120/130
699/699 [=====] - 1s 2ms/step - loss: 0.0802 - accuracy: 0.9714 - val_loss: 0.0540 - val_accuracy: 0.9643
Epoch 121/130
699/699 [=====] - 1s 2ms/step - loss: 0.1504 - accuracy: 0.9485 - val_loss: 0.0673 - val_accuracy: 0.9785
Epoch 122/130
699/699 [=====] - 1s 2ms/step - loss: 0.1266 - accuracy: 0.9518 - val_loss: 0.0711 - val_accuracy: 0.9742
Epoch 123/130
699/699 [=====] - 1s 2ms/step - loss: 0.1013 - accuracy: 0.9652 - val_loss: 0.1112 - val_accuracy: 0.9528
Epoch 124/130
699/699 [=====] - 1s 2ms/step - loss: 0.0948 - accuracy: 0.9690 - val_loss: 0.0715 - val_accuracy: 0.9814
Epoch 125/130
699/699 [=====] - 1s 2ms/step - loss: 0.1828 - accuracy: 0.9647 - val_loss: 0.0542 - val_accuracy: 0.9871
Epoch 126/130
699/699 [=====] - 1s 2ms/step - loss: 0.1582 - accuracy: 0.9485 - val_loss: 0.2536 - val_accuracy: 0.9242
Epoch 127/130
699/699 [=====] - 1s 2ms/step - loss: 0.0938 - accuracy: 0.9738 - val_loss: 0.1135 - val_accuracy: 0.9628
Epoch 128/130
699/699 [=====] - 1s 2ms/step - loss: 0.0637 - accuracy: 0.9795 - val_loss: 0.0585 - val_accuracy: 0.9785
Epoch 129/130
699/699 [=====] - 1s 2ms/step - loss: 0.1185 - accuracy: 0.9614 - val_loss: 0.1448 - val_accuracy: 0.9399
Epoch 130/130
699/699 [=====] - 1s 2ms/step - loss: 0.0745 - accuracy: 0.9738 - val_loss: 0.2197 - val_accuracy: 0.9442
    
```

[Train 결과]

→ Test Set 학습 결과

loss : 0.219

Accuracy : 0.9442(약 94%)

■ 모델 테스트

[새로운 확진자 데이터]

	A	B	C	D	E	F	G	H	I	J	K	
1	label	감염경로라벨링	접촉자수	확진자동선개수	대학교수	양로원수	유치원수	환자ID	광역자치단체	기초자치단체	감염경로	
2			1	50	10	3	2095	127	2000000100	경기도	성남시	A교회

→ 새로운 확진자 정보
 감염경로 라벨링 : 1 (교회)
 접촉자 수 : 50
 확진자 동선 개수 : 10
 광역자치단체 : 경기도
 기초자치단체 : 성남시

[검증 코드]

```
data1 = pd.read_excel('검증.xlsx')
print(data1)
x = data1.iloc[:,1:7]
x_data = x.to_numpy()
x_data.shape
print(x_data)

predict = tf.model.predict(x_data)

# 위험도0 위험도1 위험도2
zero= 0
one= 0
two= 0

for i in range(len(predict)):
    a = np.argmax(predict[i])

    if a == 0:
        zero += 1
    elif a == 1:
        one += 1
    elif a == 2:
        two += 1

print("위험도:",a, ", 확률:",predict[i][a]*100,"%")
```

[검증 결과]

```
label 감염경로라벨링 접촉자수 확진자동선개수 대학교수 ... 유치원수 환자ID
0 NaN 1 50 10 3 ... 127 2000000100 경기도 성남시 A교회

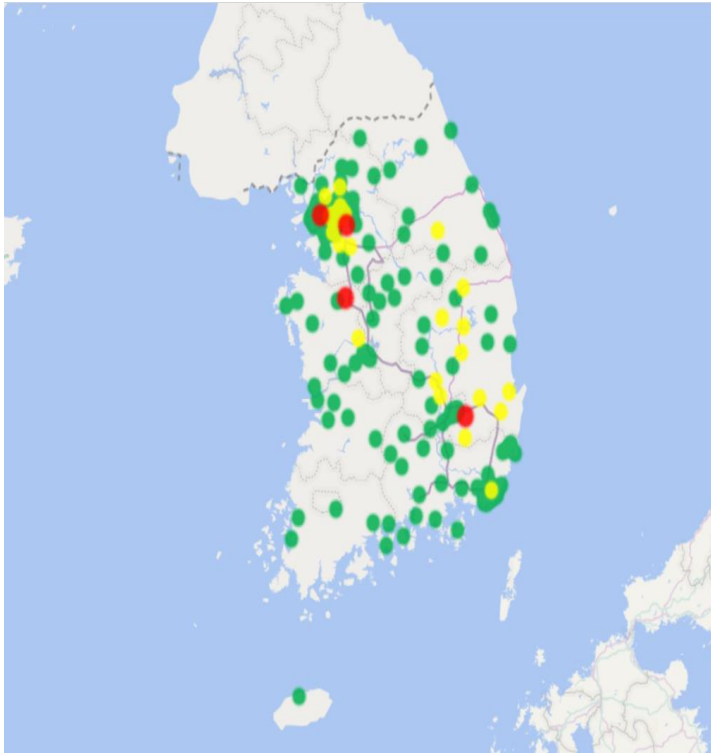
[1 rows x 11 columns]
[[ 1 50 10 3 2095 127]]
위험도: 2 , 확률: 99.98307228088379 %
```

→ 검증 결과

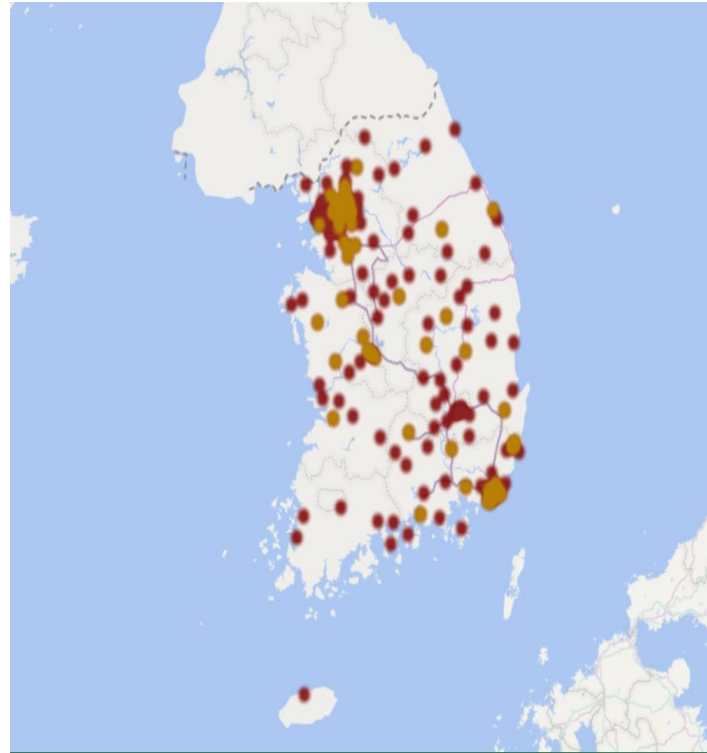
새로운 확진자의 위험도는 99%의 확률로 위험정도 2단계인 가장 높은 위험도를 예측하고 있다.

→ 따라서, 새로운 확진자의 지역에 강한 대비 경보를 내려야 하며, 병실 수를 체크해야 한다.

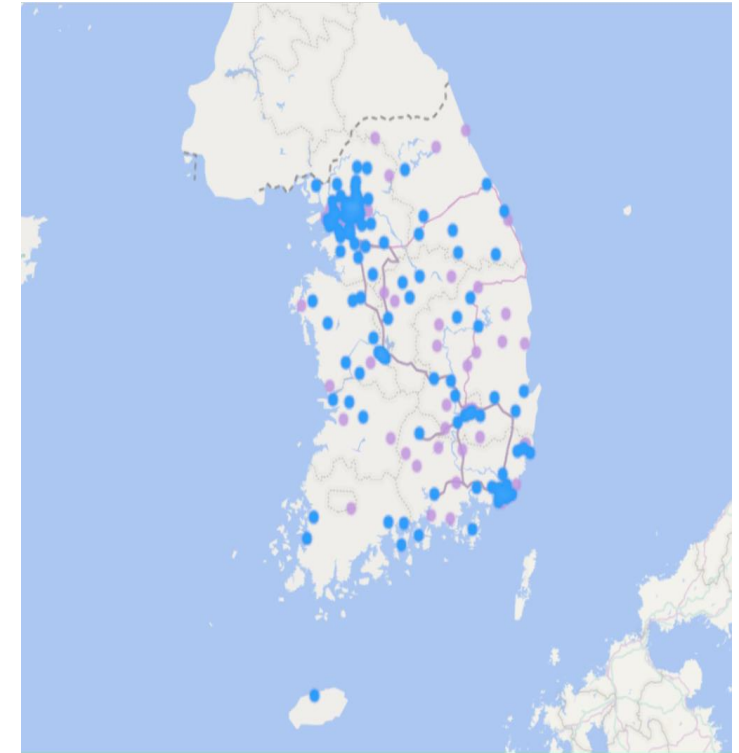
■ 시각화 모델



위험 지역 라벨링에 따라 나눈 모델



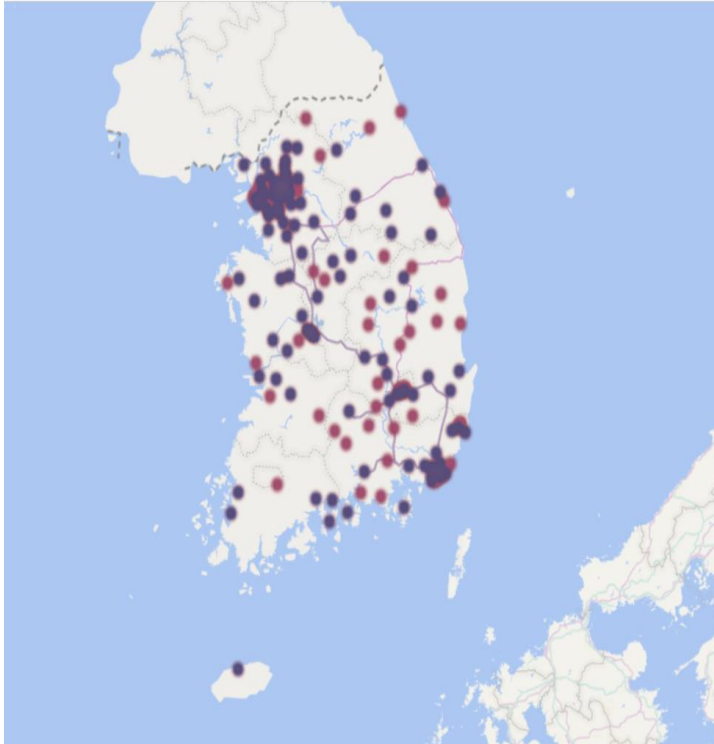
1차 감염과 N차 감염에 따라 나눈 모델



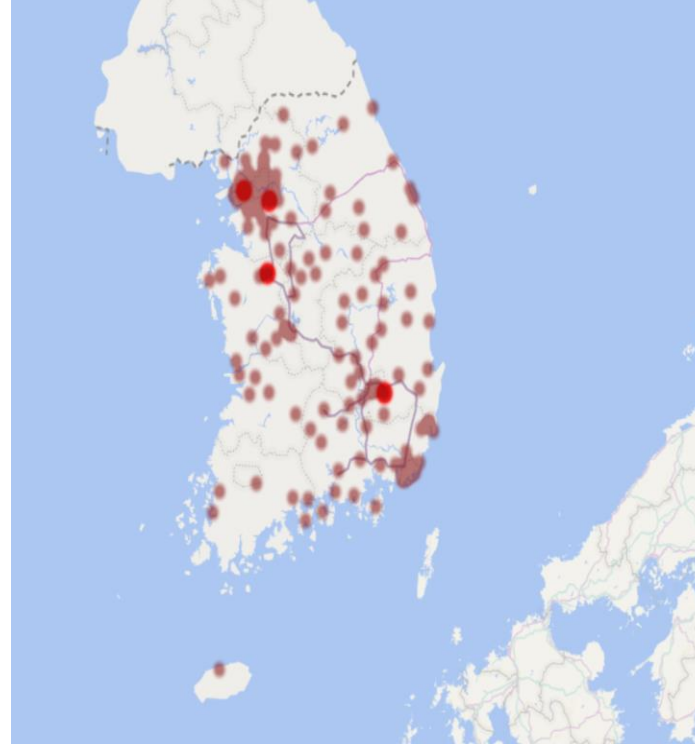
집단 지역의 종류에 따라 나눈 모델



■ 시각화 모델



N차 감염과 집단 단체의 관계를 나타내는 모델



N차 감염과 위험지역 Label 2의 관계를 나타내는 모델



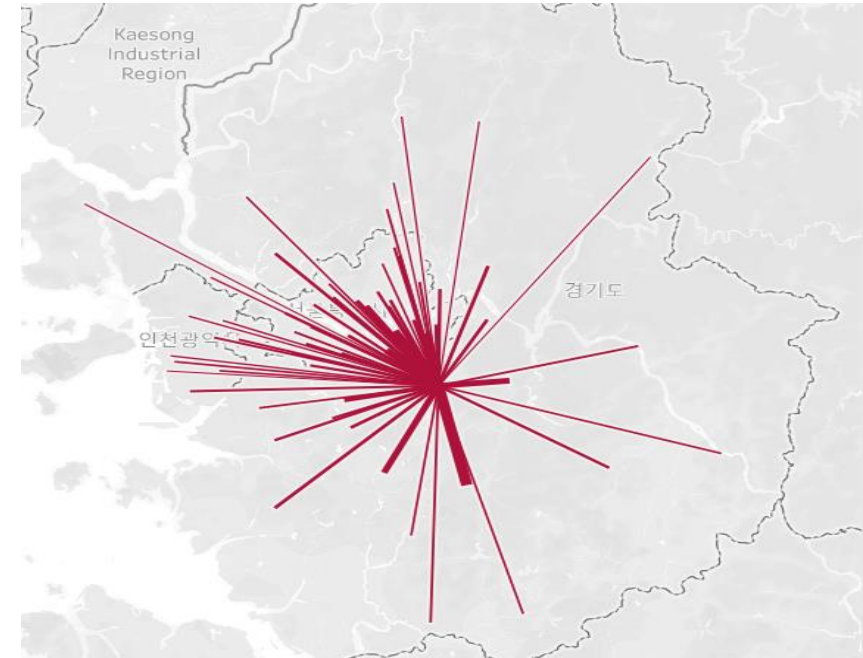
사용 코드

```
1 install.packages("ggplot2")
2 install.packages("ggmap")
3 library(ggmap)
4
5 ggmap(get_map(location='south korea', zoom=7))
6
7 wifi <- read.csv("Region_case.csv", header=T, as.is=T)
8
9 ggmap(map) + geom_point(data=Region_case, aes(x=lon, y=lat, color=label))
10 ggmap(map) + geom_point(data=Region_case, aes(x=lon, y=lat, color=1차 감염))
11 ggmap(map) + geom_point(data=Region_case, aes(x=lon, y=lat, color=N차 감염))
12
```

→ 지역의 데이터들을 시각화하여 값들을 확인하고 비교하여 예상 결과값과 비교해 볼 수 있었다.

6. 시각화

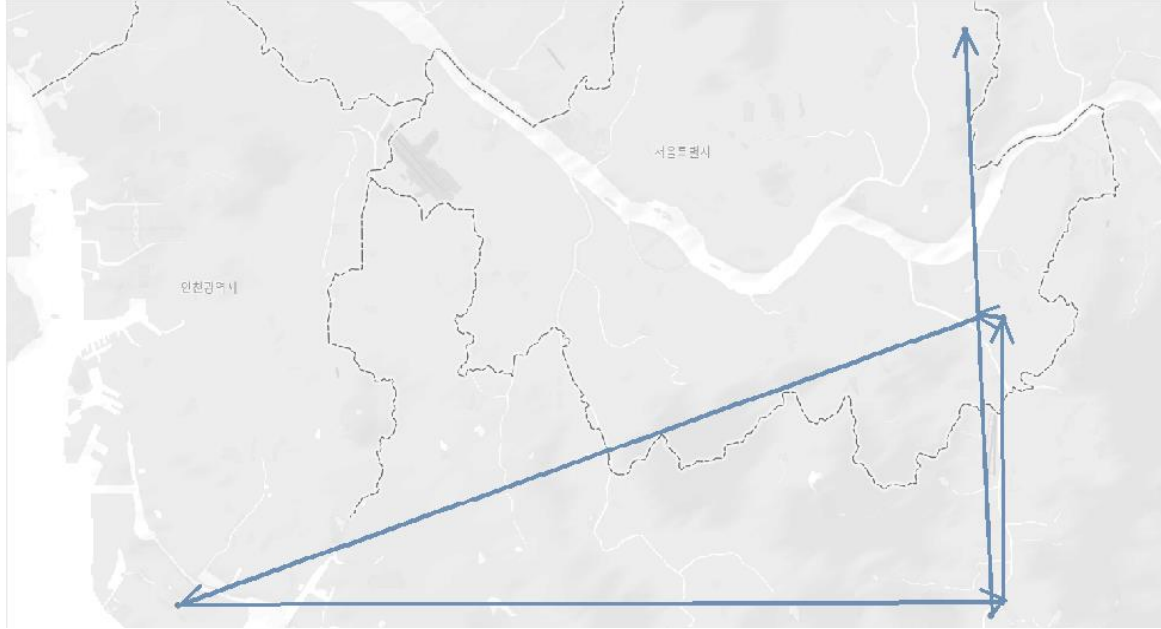
■ 시각화 모델



경기도 성남시 기준의 이동 경로를 나타낸 모델
통행량의 양에 따라 선의 굵기 표시

6. 시각화

■ 시각화 모델



	A	B	C	D	E	F	G	H
1	환자ID	순서	날짜	광역자치단	기초자치단	유형	위도	경도
2	1E+09	1	#####	경기도	성남시	etc	37.3827	127.1189
3	1E+09	2	#####	경기도	성남시	etc	37.3827	127.1189
4	1E+09	3	#####	서울	송파구	etc	37.49845	127.1073
5	1E+09	4	#####	서울	송파구	음식점	37.49695	127.12
6	1E+09	5	#####	서울	송파구	etc	37.49845	127.1073
7	1E+09	6	#####	인천	연수구	상점	37.38148	126.6572
8	1E+09	7	#####	인천	연수구	상점	37.38148	126.6572
9	1E+09	8	#####	경기도	성남시	etc	37.3827	127.1189
10	1E+09	9	#####	경기도	성남시	음식점	37.37714	127.1127
11	1E+09	10	#####	서울	중랑구	병원	37.61277	127.0982

성남시의 확진자의 이동경로 시각화한 모델

통행량과 확진자의 경로를 시각화하여, 앞으로의 환자 케이스가 생겼을 때 대처할 수 있는 정보를 알려줄 수 있다.

사용 코드

```

START 0 성남시 기점종점 ×
MAKEPOINT ([거점 (위도)], [거점 (경도)])
LAST 0 성남시 기점종점 ×
MAKEPOINT ([종점 (위도)], [종점 (경도)])
Route 0 성남시 기점종점 ×
MAKELINE ([START], [LAST])
Route_Point 0 성남시 환자 루트 ×
MAKEPOINT ([위도], [경도])
    
```

MY-SQL 질의

질의 1. 새로운 확진자 지역에서 타 지역으로의 확산을 파악하기 위해 사용

[경기도 성남시 기준 도착지의 통행량 내림차순 검색]

```
1 #모델 검증에 사용한 경기도 성남시 기준 도착지의 통행량
2 • select 통행량 as 도착지통행량, 중점_지역코드 as 도착지, 기초자치단체
3 from 통행량, 지역기본정보
4 where 기점_지역코드 = 20120
5 AND 통행량.중점_지역코드 = 지역기본정보.지역코드
6 ORDER BY 통행량 desc;
```

도착지통행량	도착지	기초자치단체
1524993	20120	성남시
146025	20230	용인시
98448	10010	강남구
87470	10180	송파구
57803	10150	서초구
49635	20050	광주시
42519	20130	수원시
23690	20170	안양시

[경기도 성남시 기준 출발지의 통행량 내림차순 검색]

```
8 #모델 검증에 사용한 경기도 성남시 기준 출발지의 통행량
9 • select 통행량 as 출발지통행량, 기점_지역코드 as 출발지, 기초자치단체
10 from 통행량, 지역기본정보
11 where 중점_지역코드 = 20120
12 AND 통행량.기점_지역코드 = 지역기본정보.지역코드
13 ORDER BY 통행량 desc;
```

출발지통행량	출발지	기초자치단체
1524993	20120	성남시
144718	20230	용인시
105130	10010	강남구
101295	10180	송파구
55241	20050	광주시
53971	10150	서초구
45286	20130	수원시
23288	10230	종로구

질의2. 새로운 확진자 지역의 코로나 전담 병원의 병실 수 확보에 사용

[경기도 성남시의 코로나 전담 병원의 병실 수 확인]

```
15 #경기도 성남시의 병원 수
16 • select 병원정보, 병원명, 병원정보.병상수, 병원전화번호,전화번호
17 from 병원정보, 병원전화번호
18 where 지역코드 = 20120
19 AND 병원전화번호.병원명 = 병원정보.병원명;
```

병원명	병상수	전화번호
성남중앙병원	230	031)743-3000
성모월병원	160	1833-7512
연세제일내과의원	0	031)755-4243

[경기도 성남시의 코로나 입원 환자 수 확인]

```
21 #경기도 성남시의 병실에 있는 환자 수
22 • select count(환자ID) as 현재_성남시_코로나입원환자수
23 from 환자기본정보
24 where 지역코드 = 20120
25 AND 환자상태 = '입원';
```

현재_성남시_코로나입원환자수
127

MY-SQL 질의

질의 3. 새로운 확진자 추가를 위한 필수 속성값 검색

[확진자 추가를 위한 속성 값 검색]

```
27 #확진자 추가를 위한 필수 요소 검색
28 • desc 환자기본정보;
29 • desc 환자이동경로;
```

Field	Type	Null	Key	Default
환자ID	bigint(20)	NO	PRI	NULL
성별	varchar(20)	YES		NULL
태어난해	int(11)	YES	MUL	NULL
국적	varchar(20)	YES		NULL
지역코드	int(11)	YES	MUL	NULL
기저질환여부	tinyint(1)	YES		NULL
감염유형	varchar(20)	YES	MUL	NULL
감염순서	int(11)	YES		NULL
전염자	bigint(20)	YES		NULL

Field	Type	Null	Key	Default
환자ID	bigint(20)	NO	PRI	NULL
환자별순서	int(11)	NO	PRI	NULL
날짜	date	NO		NULL
광역자치단체	varchar(20)	NO		NULL
기초자치단체	varchar(20)	NO		NULL
장소유형	varchar(20)	NO		NULL
장소위도	float	NO		NULL
장소경도	float	NO		NULL

질의 4. 새로운 확진자 정보 삽입

[새로운 확진자 정보 삽입]

```
31 #확진자 추가
32 • set autocommit = 0;
33 • start transaction;
34 • savepoint A;
35 • insert into 환자기본정보 (환자ID, 성별, 지역코드, 접촉자수, 확진날짜, 환자상태)
36 values (2000000100, '여자', 20120, 50, '2020-06-22', '입원');
37 • insert into 환자이동경로 (환자ID, 환자별순서, 날짜, 광역자치단체, 기초자치단체, 장소유형, 장소위도, 장소경도)
38 values (2000000100, 1, '2020-06-22', '경기도', '성남시', 'A교회', 1701919, 10981092);
39 • rollback to savepoint A; #데이터를 잘못 삽입할 경우를 대비한 rollback 쿼리

42 #확진자 추가 확인
43 • select *
44 from 환자기본정보
45 where 환자ID = 2000000100;
46 • select *
47 from 환자이동경로
48 where 환자ID = 2000000100;
```

환자ID	성별	태어난해	국적	지역코드	기저질환여부	감염유형	감염순서	전염자	접촉자수	증상발현날짜	확진날짜	퇴원날짜	사망날짜	환자상태
2000000100	여자	20120		20120					50		2020-06-22			입원

환자ID	환자별순서	날짜	광역자치단체	기초자치단체	장소유형	장소위도	장소경도
2000000100	1	2020-06-22	경기도	성남시	A교회	1701920	10981100

MY-SQL 질의

질의5. 참조 무결성 조건 확인

[확진자 정보를 지울 때 참조 무결성 확인]

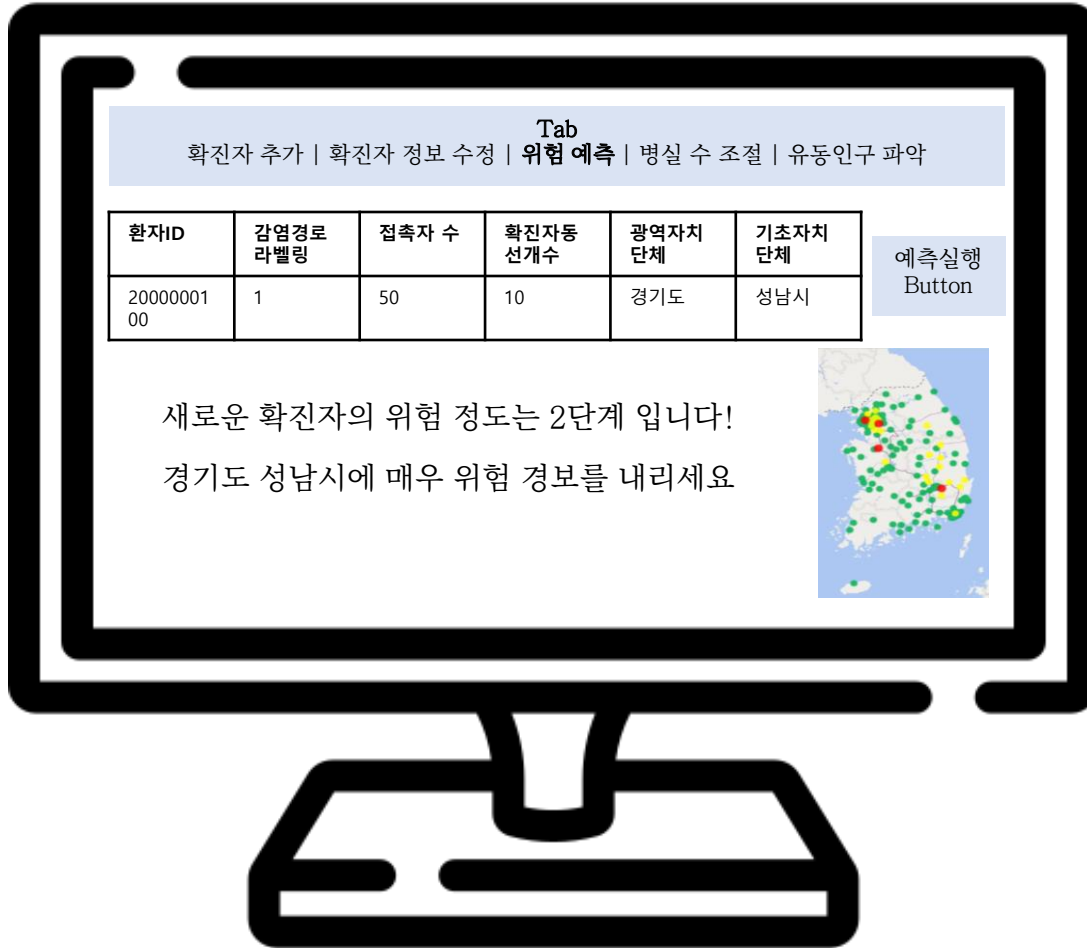
```
49
50 #참조 무결성 조건으로 인해 삭제 불가
51 • delete from 환자기본정보
52 where 환자ID = 2000000100;
```

299 02:33:14 delete from 환자기본정보 where 환자ID = 2000000100 Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails ('edb_project'. '환자이동... 0.000 sec

```
54 #자식 테이블에서 먼저 삭제 후 부모 테이블에서 삭제 가능
55 • delete from 입려닝
56 where 환자ID = 2000000100;
57 • delete from 환자이동경로
58 where 환자ID = 2000000100;
59 • delete from 환자기본정보
60 where 환자ID = 2000000100;
61
62 • select *
63 from 환자기본정보
64 where 환자ID = 2000000100;
65
```

환자 ID	성별	태어 난해	국적	지역 코드	기저질환 여부	감염 유형	감염 순서	진료과	검측 자수	증상발현 날짜	확진 날짜	퇴원 날짜	사망 날짜	환자 상태
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

8. 결론 및 기대효과



- 새로운 확진자 발생 시, 그 지역의 위험 정도를 정확하고 빠르게 예측할 수 있고, 알맞은 상황 대처 가능
- 위험 지역으로 예측 시, 해당 지역의 병실 수를 미리 체크하여 상황에 대비 가능
- 관련 데이터들을 한번에 시각화하여, 앞으로 생길 감염자에 대처할 수 있는 정보 제공
- 전국의 모든 관련 기관들이 동일한 정보 포맷을 이용하여 입력하기 때문에, 후에 데이터 가공 시간을 절약할 수 있음
- 전국의 모든 관련 기관들이 동일하게 상황파악 가능

□ □ 감사합니다