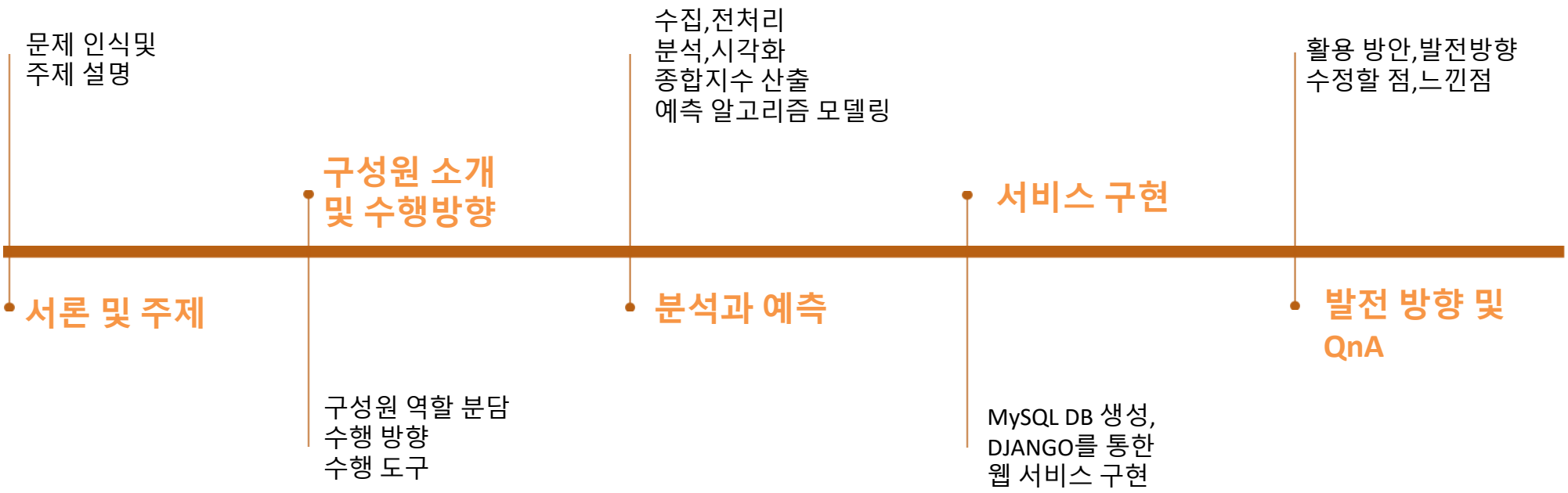
BEACONOMIC PROJECT

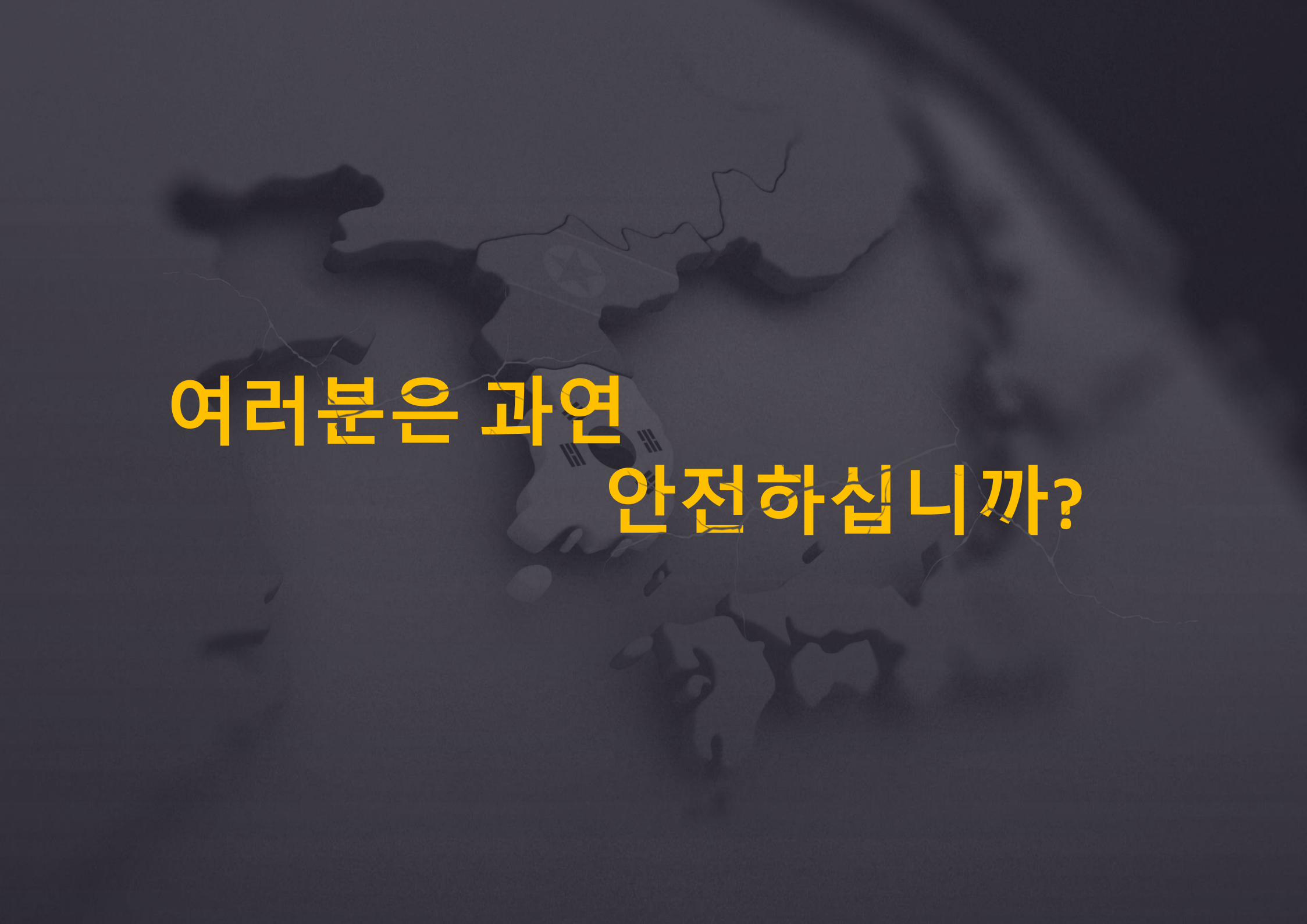
서울시 치안 종합지도와 112 신고건수 예측을 통한 범죄수요 예측

박준영

허지훈, 이원우, 장민준, 박창훈

목차


문제 인식및
주제 설명
서론 및 주제
구성원 소개
및 수행방향
구성원 역할 분담
수행 방향
수행 도구
수집,전처리
분석,시각화
종합지수 산출
예측 알고리즘 모델링
분석과 예측
서비스 구현
MySQL DB 생성,
DJANGO를 통한
웹 서비스 구현
활용 방안,발전방향
수정할 점,느낀점
발전 방향 및
QnA


여러분은 과연
안전하십니까?

신당역
살인 사건

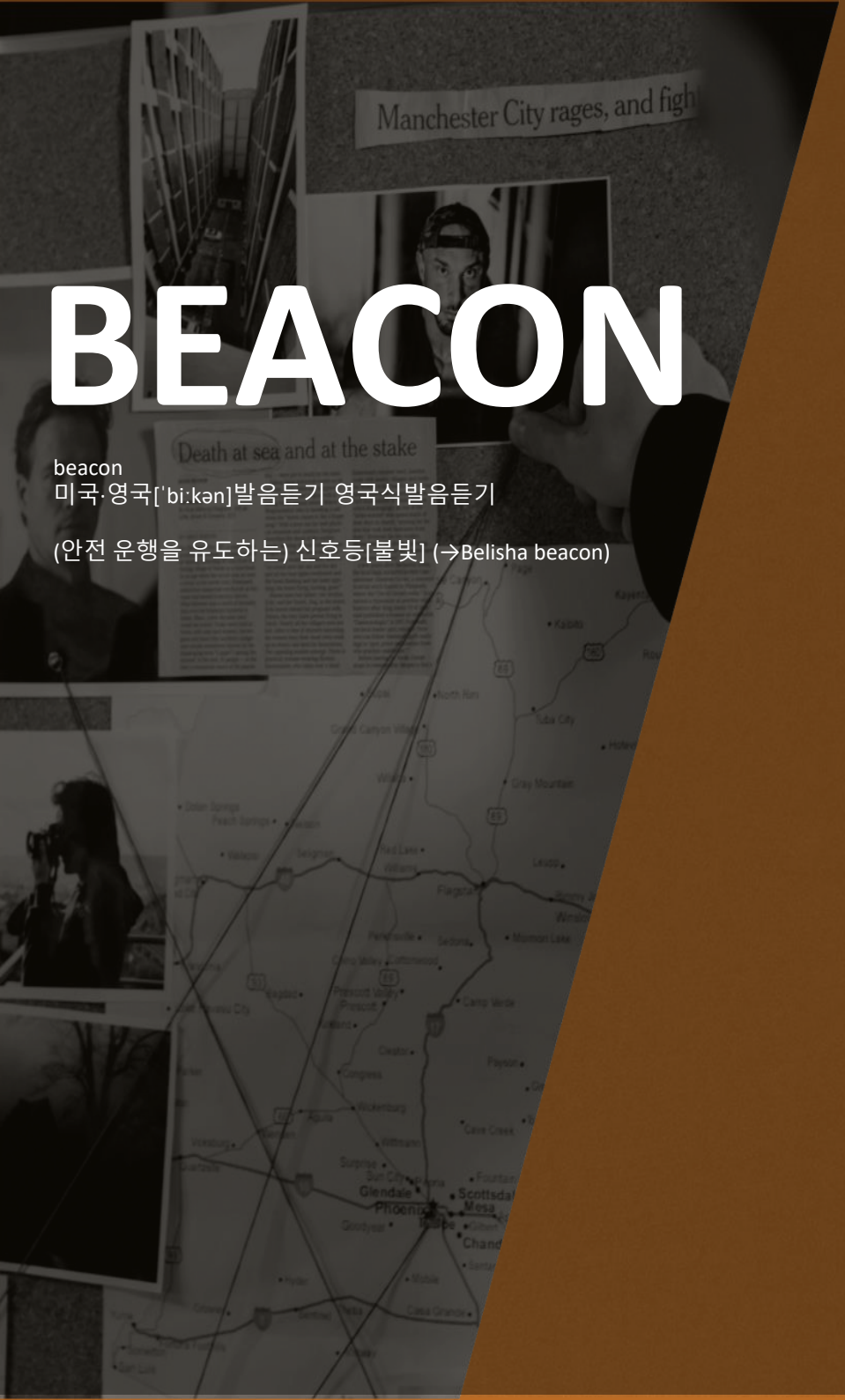
당진 자매
살인사건

송파
전 여자친구
가족 살인사건

가장 가깝지만
가장 무서운

CAUTION

BEACONOMIC

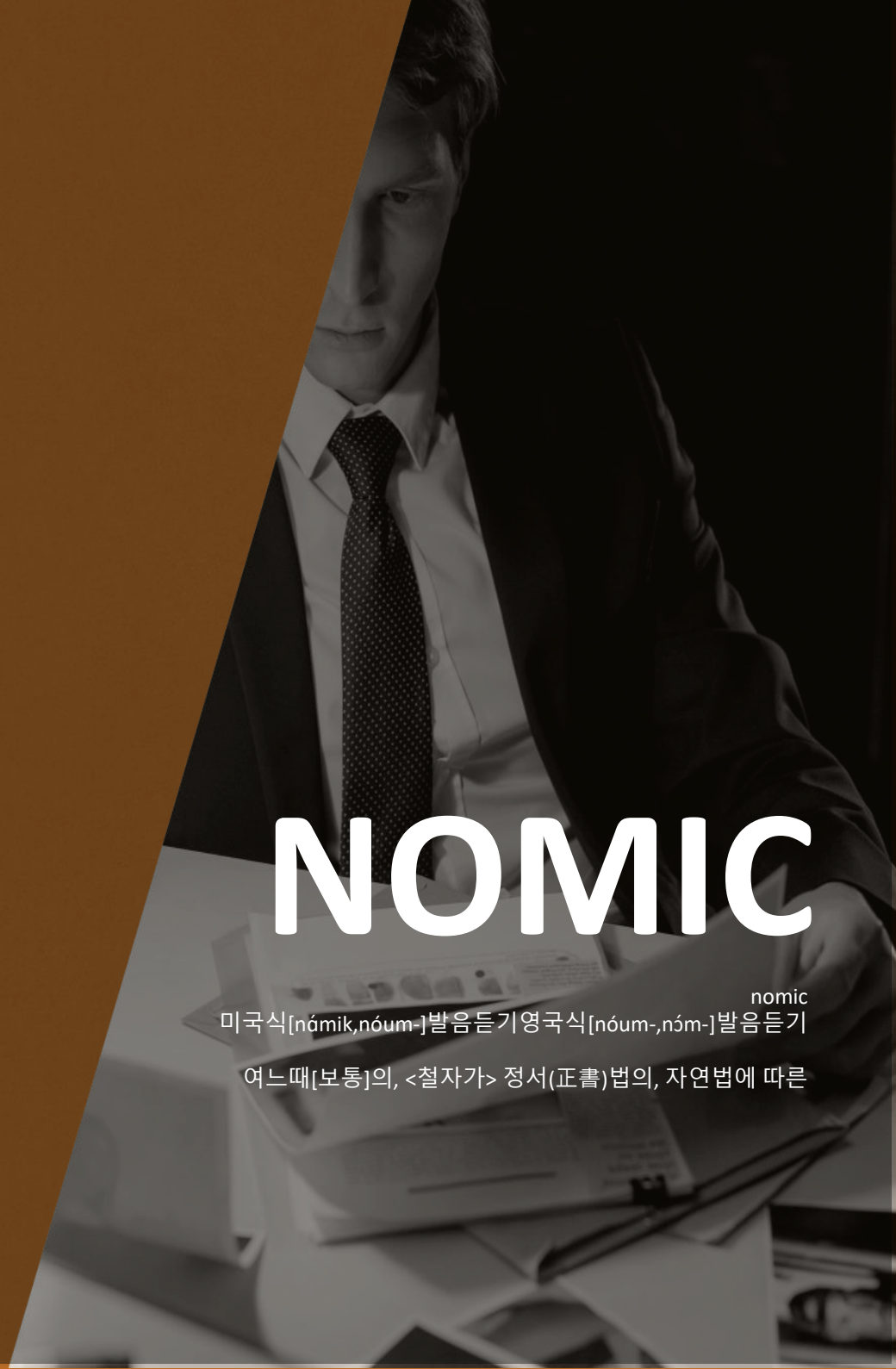


BEACON

beacon
미국·영국['biːkən]발음듣기 영국식발음듣기

(안전 운행을 유도하는) 신호등[불빛] (→Belisha beacon)

+



NOMIC

nomic
미국식[nάmik,nóum-]발음듣기영국식[nóum-,nɔ́m-]발음듣기

여느때[보통]의, <철자가> 정서(正書)법의, 자연법에 따른

치안 안전 지도

치안환경 데이터를 분석하고
군집화해 종합 치안지수를 산출해
치안 관련 정보를 제공하는 종합
치안 안전 지도 서비스를 제공한다

치안 안전 지도
&
112 신고건수 예측

112 신고 건수 예측

회귀 분석 알고리즘을 통해
앞으로 발생할 112 신고 건수를
예측하는 알고리즘을 모델링해
범죄 수요를 예측한다.

수행 방향,
구성원 소개

구성원 소개

조장: 박준영 – ppt 작성 및 발표, 전처리, 시각화, 데이터 모델링
조원1: 허지훈 - 전처리, 시각화, 데이터 모델링
조원2: 이원우 - 전처리, 시각화, 데이터 모델링
조원3: 장민준 - 전처리, 시각화, 데이터 모델링
조원4: 박창훈 - 전처리, 시각화, 데이터 모델링

공통 – 기획, 자료 조사, DB 구축, 웹서버, front 개발

수행방향

치안 안전 지도

서울시 행정구별 치안 데이터를 수집했습니다.
위의 데이터를 전처리하고 시각화 합니다. 치안 안전 지수를 산출하는 계산식을 파이썬 함수로 만듭니다.
그리고 전처리한 변수들의 상관관계를 계산해 변수로 사용해 종합 치안 지수를 산출합니다.

112 신고 건수를 통해 범죄수요 예측

회귀 분석 모델을 이용해 112 신고 건수의 증감을 예측하는 모델링을 할 예정입니다.

그렇게 예측한 값으로 각 행정구별 범죄 수요를 예측할 수 있습니다.

수행도구

수집,전처리
분석,시각화

예측 알고리즘 모델링

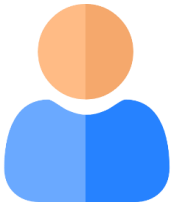
DataBase

서비스 구현

분석,
예측

수집 데이터

서울시 행정구별
인구밀도 와 변화



면적당
여성인구 수

경찰서(파출소)의 수

범죄 발생 수

가로등,
CCTV 수

지역구별
112 신고건수

전처리

자치구별(1)	자치구별(2)	2014											
		합계											
		소계		살인		강도		강간·강제추행		절도		폭력	
		발생	검거	발생	검거	발생	검거	발생	검거	발생	검거	발생	검거
합계	소계	#####	####	158	152	343	299	5,462	4,957	####	####	####	####
	종로구	5,021	4,610	3	7	12	15	226	948	2,272	1,281	2,508	2,359
	중구	5,231	3,188	6	5	13	13	221	161	2,576	887	2,415	2,122
	용산구	3,799	2,340	1	2	7	7	213	169	1,560	522	2,018	1,640
	성동구	3,582	2,048	1	1	5	6	141	103	1,753	529	1,682	1,409
	광진구	6,268	3,531										
	동대문구	4,363	2,882										
	중랑구	5,353	3,259										
	성북구	4,606	3,028										
	강북구	4,030	2,806										
	도봉구	3,124	1,986										
	노원구	5,312	3,300										
	은평구	5,431	3,325										
	서대문구	4,194	2,588										

A1 5대범죄

	A	B	C	D	E	F	G	H
1	5대범죄	CCTV	인구밀도	가로등	경찰서수	여성인구	112신고건	인구
2	8617	830	14728	8988	25	282,162	169303	581760
3	5244	154	18842	8651	20	223,788	94193	463321
4	4257	124	14172	6977	13	163,606	79634	334426
5	5585	199	14376	5299	16	292,237	116561	595691
6	6345	609	17891	4981	11	257,655	113627	529031
7	5909	98	21987	7890	23	187,552	92547	375180
8	5646	268	22596	5392	19	221,847	89610	454604
9	3781	361	19675	5301	26	122,654	68027	256167
10	5130	514	16317	4360	16	288,868	32992	578221
11	2664	172	17056	4400	11	173,908	96246	353241
12	4720	115	26301	8745	20	183,598	58027	373824
13	4074	128	25240	6835	22	207,292	71189	412774
14	5854	185	16706	5467	12	198,673	110374	398351
15	4029	137	18351	7428	19	160,812	68579	323105
16	5444	562	9601	7209	19	218,963	99429	451258
17	3358	325	18093	6540	14	148,674	58252	305065
18	4154	322	19104	7365	19	234,983	79826	469560
19	6778	214	19704	4736	13	325,605	134780	667480
20	4528	246	28098	8679	17	235,634	79155	489010
21	6867	209	17033	4641	17	202,727	118640	417811
22	3820	221	11337	4964	21	117,302	73050	247909
23	4745	329	16920	11143	21	245,665	88986	502578
24	4705	150	6851	12184	25	82,011	77520	163822
25	4954	236	13486	11225	22	65,260	132963	134329
26	5193	132	22633	7232	18	203,301	92500	418620
27	8149	1293	14484	8988	25	277,824	167991	572140
28	4462	202	18238	8621	21	216,517	87446	448471
29	4229	251	14015	7206	13	161,805	76723	330704
30	5450	168	14531	5712	16	294,766	111972	602104
31	5678	622	17776	4925	11	255,824	108733	525607
32	5322	52	21807	8019	22	186,121	88093	372104
33	5366	326	22347	5682	19	220,938	86042	449600

쓸모없는 행렬을 삭제한다.

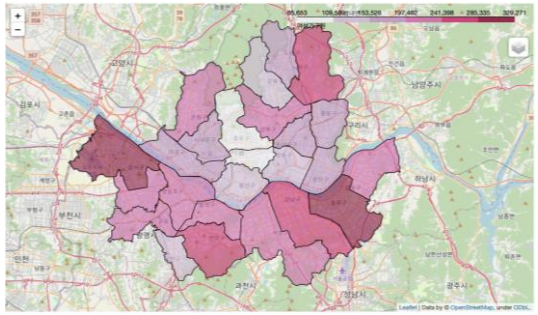
수집한 자료들의 기간이 전부 달라
공통으로 수집할 수 있던
2015년도 부터 2021년도 까지
정리했다.

각 지역구별로 구분 할 수 있도록
모두 전처리 했다.

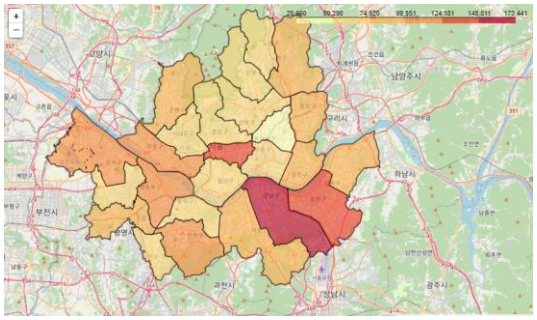
시각화

Python의 Folium 라이브러리를 통해 각각 수집한 데이터들을 시각화

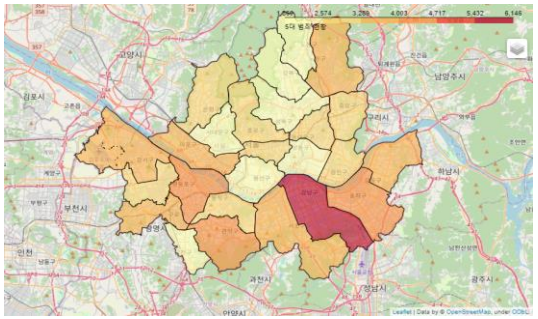
여성인구 밀집도 시각화



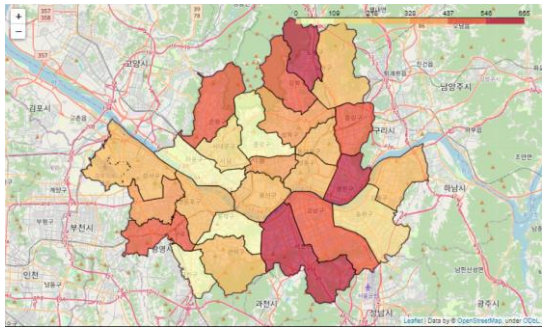
범죄(살인,강도,강간 및 추행,절도,폭력) 발생 시각화



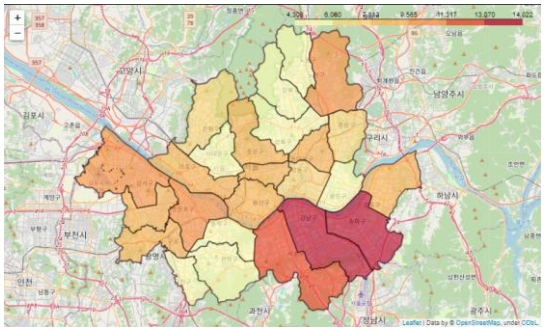
112 신고 건수 시각화



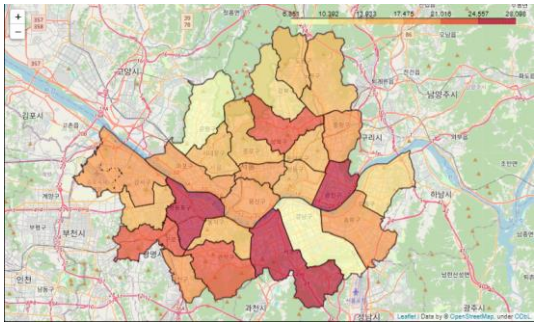
CCTV 설치 시각화



가로등 개수 시각화



인구 밀집도 시각화



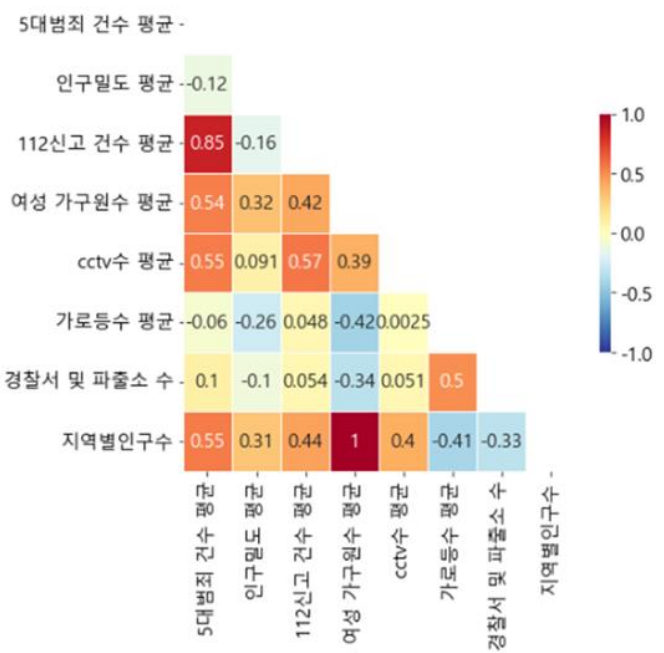
상관 관계 분석

```
# 그림 사이즈 지정
fig, ax = plt.subplots( figsize=(7,7) )

# 삼각형 마스크를 만든다(위 쪽 삼각형에 True, 아래 삼각형에 False)
mask = np.zeros_like(df2, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# 히트맵을 그린다
sns.heatmap(df2,
            cmap = 'RdYlBu_r',
            annot = True,   # 실제 값을 표시한다
            mask=mask,      # 표시하지 않을 마스크 부분을 지정한다
            linewidths=.5,  # 경계면 실선으로 구분하기
            cbar_kws={"shrink": .5},# 컬러바 크기 절반으로 줄이기
            vmin = -1,vmax = 1   # 컬러바 범위 -1 ~ 1
           )
plt.show()
```

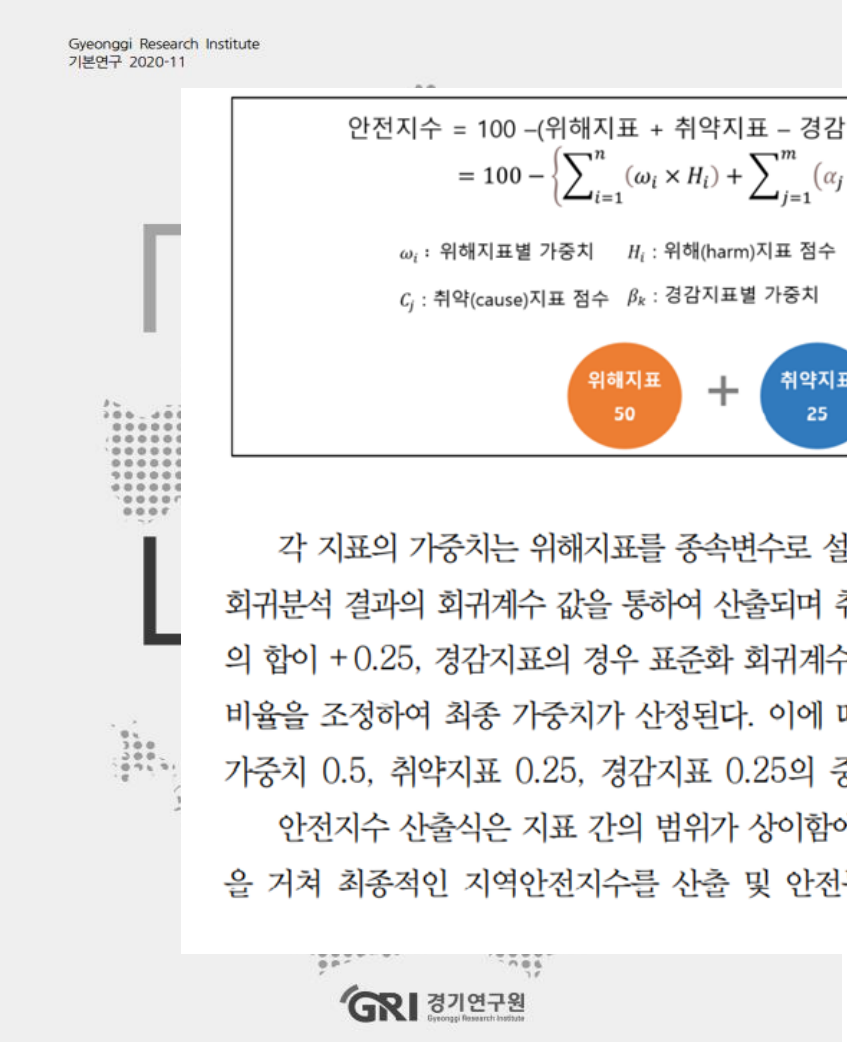
```
C:\Users\user\AppData\Local\Temp\ipykernel_23412\2248707057.py:5: DeprecationWarning: `np.bool` is a deprecated alias for the bu
ltin `bool`. To silence this warning, use `bool` by itself. Doing this will not modify any behavior and is safe. If you specific
lly wanted the numpy scalar type, use `np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  mask = np.zeros_like(df2, dtype=np.bool)
```

```
df2=df1.corr()
df2
```

	5대범죄 건수 평균	인구밀도 평균	112신고 건수 평균	여성 가구원수 평균	cctv수 평균	가로등수 평균	경찰서 및 파출소 수	지역별인구수
5대범죄 건수 평균	1.000000	-0.122679	0.853244	0.541711	0.551223	-0.059686	0.100614	0.553393
인구밀도 평균	-0.122679	1.000000	-0.157382	0.321665	0.090934	-0.258817	-0.099768	0.312262
112신고 건수 평균	0.853244	-0.157382	1.000000	0.415883	0.568105	0.047604	0.054176	0.437484
여성 가구원수 평균	0.541711	0.321665	0.415883	1.000000	0.389118	-0.423861	-0.338928	0.997800
cctv수 평균	0.551223	0.090934	0.568105	0.389118	1.000000	0.002527	0.051148	0.397457
가로등수 평균	-0.059686	-0.258817	0.047604	-0.423861	0.002527	1.000000	0.499306	-0.411624
경찰서 및 파출소 수	0.100614	-0.099768	0.054176	-0.338928	0.051148	0.499306	1.000000	-0.330807
지역별인구수	0.553393	0.312262	0.437484	0.997800	0.397457	-0.411624	-0.330807	1.000000



상관 관계 분석



변수들간의 상관관계를 통해 종합안전지수를 구한다.

상관관계

0.7이상 :위해

0.7~0.5: 취약

0.5 미만: 경감

상관 관계 분석

```
In [27]: df4['안전등급'] = np.where(df4['안전지수(100-(위해지수+취약지수-경감지수)*100']<74.79, 1,
                                np.where(df4['안전지수(100-(위해지수+취약지수-경감지수)*100']<77.31, 2,
                                        np.where(df4['안전지수(100-(위해지수+취약지수-경감지수)*100']<79.06, 3, 4)))

         df4.head(10)
```

	지역	안전지수(100-(위해지수+취약지수-경감지수)*100	안전등급
0	강남구	72.513376	1
1	강동구	78.313422	3
2	강북구	75.902832	2
3	강서구	78.076987	3
4	관악구	76.710396	2
5	광진구	75.474702	2
6	구로구	77.945891	3
7	금천구	74.793921	2
8	노원구	79.539811	4
9	도봉구	79.935653	4

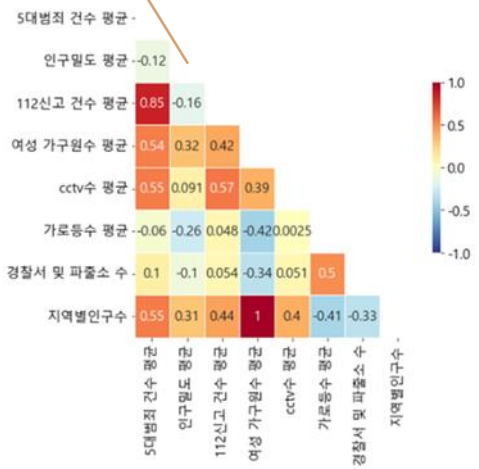
안전지수 = 100 −(위해지표 + 취약지표 − 경감지표)

$$= 100 - \left\{ \sum_{i=1}^{n} (\omega_i \times H_i) + \sum_{j=1}^{m} (\alpha_j \times C_j) - \sum_{k=1}^{o} (\beta_k \times M_k) \right\}$$

ω_i : 위해지표별 가중치 H_i : 위해(harm)지표 점수 α_j : 취약지표별 가중치

C_j : 취약(cause)지표 점수 β_k : 경감지표별 가중치 M_k : 경감(mitigation)지표 점수

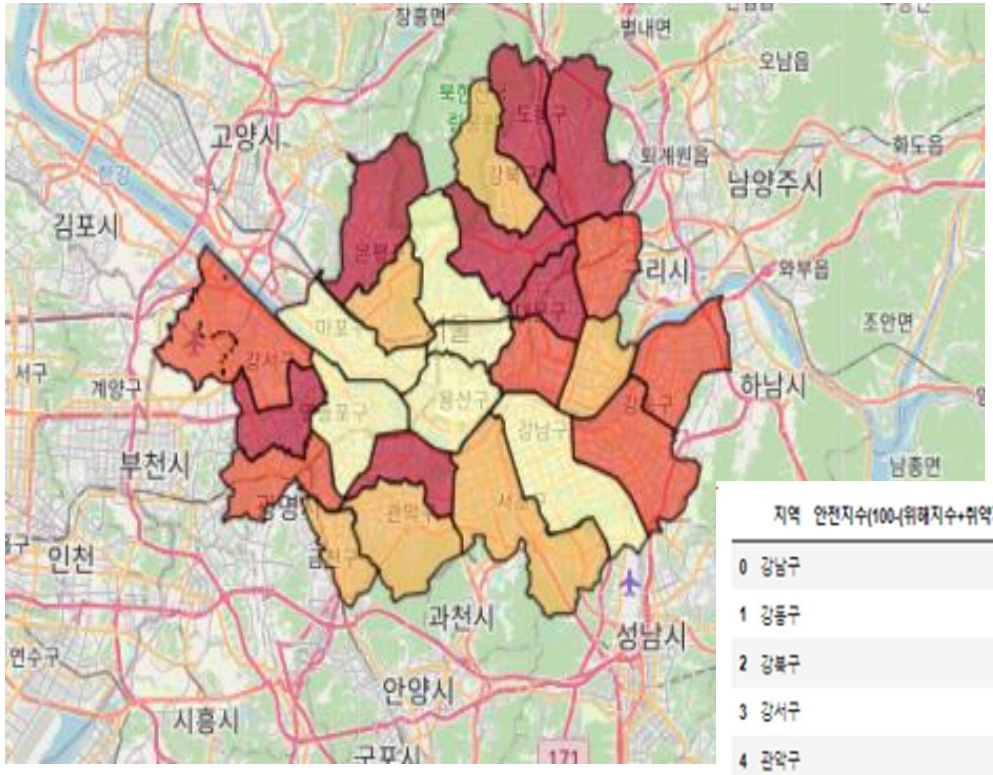
위해지표 50 + 취약지표 25 + 경감지표 25



각 지표의 가중치는 위해지표를 종속변수로 설정하고 이에 대한 세부 지표들의 회귀분석 결과의 회귀계수 값을 통하여 산출되며 취약지표의 경우 표준화 회귀계수의 합이 +0.25, 경감지표의 경우 표준화 회귀계수의 합이 -0.25가 되도록 상대적 비율을 조정하여 최종 가중치가 산정된다. 이에 따라 안전지수 산출 시 위해지표 가중치 0.5, 취약지표 0.25, 경감지표 0.25의 중요도가 반영된다.

안전지수 산출식은 지표 간의 범위가 상이함에 따라 다음과 같은 상세화 과정을 거쳐 최종적인 지역안전지수를 산출 및 안전등급을 산정하게 된다.

종합지수



	지역	안전지수(100-(위해지수+취약지수-경감지수)*100	안전등급
0	강남구	72.513376	1
1	강동구	78.313422	3
2	강북구	75.902832	2
3	강서구	78.076987	3
4	관악구	76.710396	2
5	광진구	75.474702	2
6	구로구	77.945891	3
7	금천구	74.793921	2
8	노원구	79.539811	4
9	도봉구	79.935653	4

Python의 Folium 라이브러리를 통해 치안 종합지수를 시각화해 한번에 위험도를 확인할 수 있게 했다.

예측 알고리즘 모델링

SPLIT DATA

01 **XGB Regression**

02 **Random Forest Regression**

03 **AdaBoost Regression**

04 **Linear Regression**

05 **Decision Tree Regression**

회귀 분석 모델링

사이킷런의 회귀 분석 모델들을 적용해 112 신고건수를 예측하는 모델을 구상

112 신고 건수를 종속 변수로 다른 수집한 데이터들을 독립 변수로 앞으로 일어날 신고 건수를 예측

예측 알고리즘 모델링

01 XGB Regression

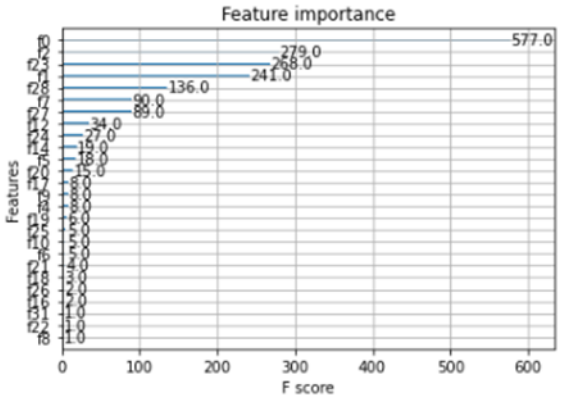
```
In [4]:
from sklearn.model_selection import train_test_split
train_input,test_input,train_output,test_output=train_test_split(a,output) # 스케일링된 값과
```

```
In [5]:
import xgboost
from sklearn.metrics import explained_variance_score

xgb_model = xgboost.XGBRegressor(n_estimators=100, learning_rate=0.08, gamma=0, subsample=0.75,
                                 colsample_bytree=1, max_depth=7)
xgb_model.fit(train_input,train_output)
```

```
Out[5]:
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.08, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=7, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, ...)
```

```
In [6]:
xgboost.plot_importance(xgb_model)
```

```
Out[6]:
<AxesSubplot:title={'center':'Feature importance'}, xlabel='F score', ylabel='Features'>
```




```
In [7]:
predictions = xgb_model.predict(test_input)
predictions
```

```
Out[7]:
array([ 84098.57 ,  64655.32 ,  40076.562,  76312.99 ,  71213.664,
        77562.484,  71999.59 , 119305.86 ,  91670.22 , 109197.664,
       116633.766, 126877.945,  68714.22 ,  75892.41 ,  85614.53 ,
        77552.01 ,  87662.55 ,  99793.   ,  84748.32 ,  70604.53 ,
       119514.38 ,  66018.71 , 110056.13 ,  76013.94 , 127763.88 ,
       118382.18 ,  86664.07 , 111906.85 ,  61772.727,  87826.68 ,
        73230.7  ,  47160.234,  84684.23 ,  72904.15 ,  99683.01 ,
        69721.625,  75695.3  , 110330.   , 116550.38 , 101656.25 ,
        52574.   , 100372.82 , 125818.74 ,  84907.586], dtype=float32)
```

```
In [8]:
r_sq = xgb_model.score(test_input, test_output)
print(r_sq)
print(explained_variance_score(predictions,test_output))
```

```
0.7969370823300117
0.6365102502312034
```

전처리된 데이터들을 통해 112 신고건수를 Xgboost로 예측하였으나 학습데이터가 적고 손봐야할 파라미터 값이 너무 많아서 모두 고려하지 않고 진행하여 낮은 정확도를 보였다.
그래프에서 나타나듯이 각 Feature 들의 F-score 값을 보았을 때도 원하는 Feature의 영향이 아닌 다른 Feature들의 값이 높게 왔다

예측 알고리즘 모델링

02 Random Forest Regression

```
from sklearn.ensemble import RandomForestRegressor

regr = RandomForestRegressor(max_depth=100, random_state=0)
regr.fit(X_train, y_train)

y_test_pred=regr.predict(X_test)
print(regr.score(X_test,y_test))

scores = cross_val_score(regr, X, y)
print(scores.mean())
```

```
0.9151286327015289
0.8597896081198488
```

03 AdaBoost Regression

```
from sklearn.ensemble import AdaBoostRegressor

regr = AdaBoostRegressor(random_state=0, n_estimators=100)
regr.fit(X_train, y_train)

y_test_pred=regr.predict(X_test)
print(regr.score(X_test,y_test))

scores = cross_val_score(regr, X, y)
print(scores.mean())
```

```
0.8411066178606017
0.816778334895018
```

04 Linear Regression

```
from sklearn.linear_model import LinearRegression

regr = LinearRegression()
regr.fit(X_train, y_train)

y_test_pred=regr.predict(X_test)
print(regr.score(X_test,y_test))

scores = cross_val_score(regr, X, y)
print(scores.mean())
```

```
0.4813831349136145
0.42379440063179324
```

05 Decision Tree Regression

```
from sklearn.tree import DecisionTreeRegressor

regr = DecisionTreeRegressor(random_state=0)
regr.fit(X_train, y_train)

y_test_pred=regr.predict(X_test)
print(regr.score(X_test,y_test))

scores = cross_val_score(regr, X, y)
print(scores.mean())
```

```
0.6125243270070375
0.6779929569145245
```

1차학습(지역무시)

학습데이터를
최대한으로 부여하기
위해 자치구를 무시하고
학습을 진행하였다.

각 회귀모델의 성능을
교차평가하기 위해
사이킷런의
cross_val_score을
사용하였다.

예측 알고리즘 모델링

df_total_onehot

Python

	5대범죄	CCTV	인구밀도	가로등	경찰서수	여성인구	112신고건수	인구	강남구	강동구	...	성동구	성북구	송파구	양천구	영등포구	용산구	은평구	종로구	중구	중랑구
0	8617	830	14728	8988	25	282,162	169303	581760	1	0	...	0	0	0	0	0	0	0	0	0	0
1	5244	154	18842	8651	20	223,788	94193	463321	0	1	...	0	0	0	0	0	0	0	0	0	0
2	4257	124	14172	6977	13	163,606	79634	334426	0	0	...	0	0	0	0	0	0	0	0	0	0
3	5585	199	14376	5299	16	292,237	116561	595691	0	0	...	0	0	0	0	0	0	0	0	0	0
4	6345	609	17891	4981	11	257,655	113627	529031	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
170	2381	228	10852	5789	19	114,778	70794	237285	0	0	...	0	0	0	0	0	1	0	0	0	0
171	3244	471	16061	12593	21	239,230	91838	477173	0	0	...	0	0	0	0	0	0	1	0	0	0
172	2712	0	6431	14351	23	77,490	69511	153789	0	0	...	0	0	0	0	0	0	0	1	0	0
173	2861	403	13231	14822	22	65,653	127597	131787	0	0	...	0	0	0	0	0	0	0	0	1	0
174	3210	448	21188	8126	17	193,943	84004	391885	0	0	...	0	0	0	0	0	0	0	0	0	1

지역적 특성을 반영하기 위해 원핫인코딩을 사용하였다.

scikit learn 예측 알고리즘 모델링

02 Random Forest Regression

```
from sklearn.ensemble import RandomForestRegressor

regr = RandomForestRegressor(max_depth=100, random_state=0)
regr.fit(X_train, y_train)

y_test_pred=regr.predict(X_test)
print(regr.score(X_test,y_test))

scores = cross_val_score(regr, X, y)
print(scores.mean())
```

```
0.9151286327015289
0.8597896081198488
```

03 AdaBoost Regression

```
from sklearn.ensemble import AdaBoostRegressor

regr = AdaBoostRegressor(random_state=0, n_estimators=100)
regr.fit(X_train, y_train)

y_test_pred=regr.predict(X_test)
print(regr.score(X_test,y_test))

scores = cross_val_score(regr, X, y)
print(scores.mean())
```

```
0.8406206327000445
0.8643023637379811
```

04 Linear Regression

```
from sklearn.linear_model import LinearRegression

regr = LinearRegression()
regr.fit(X_train, y_train)

y_test_pred=regr.predict(X_test)
print(regr.score(X_test,y_test))

scores = cross_val_score(regr, X, y)
print(scores.mean())
```

```
0.9882202279625926
0.9884395404401545
```

05 Decision Tree Regression

```
from sklearn.tree import DecisionTreeRegressor

regr = DecisionTreeRegressor(random_state=0)
regr.fit(X_train, y_train)

y_test_pred=regr.predict(X_test)
print(regr.score(X_test,y_test))

scores = cross_val_score(regr, X, y)
print(scores.mean())
```

```
0.9361874636618323
0.7684170277172179
```

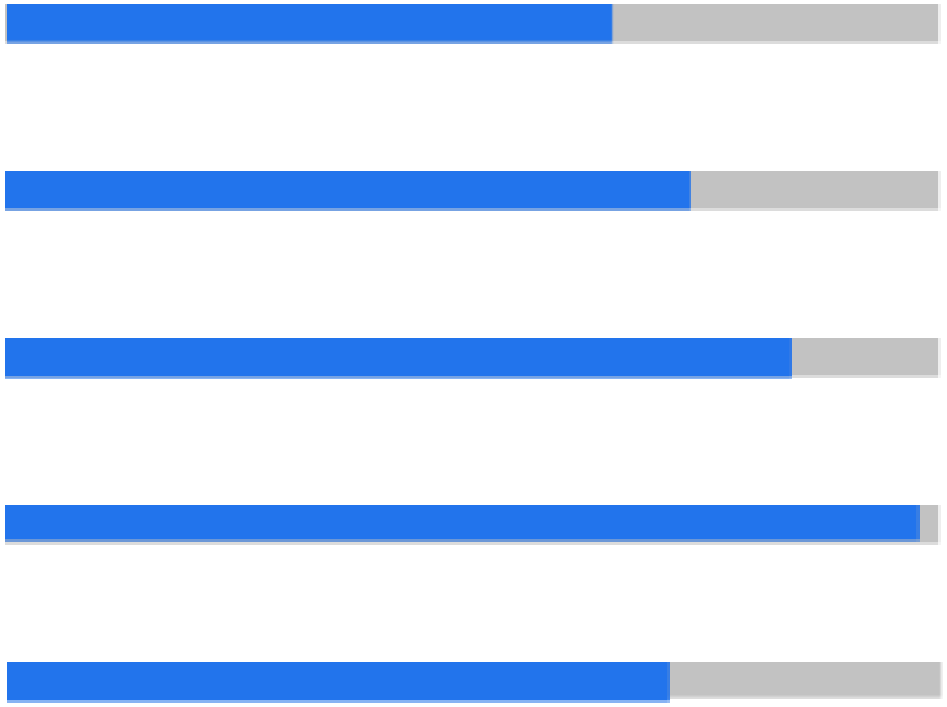
2차학습(지역반영)

지역적 특성을 반영하기 위해 원핫인코딩을 사용하였다.
원핫인코딩을 해준 후 학습을 진행하였을 때, **LinearRegression**모델이 가장 높은 성능을 보여주는 것을 확인 할 수 있었다.

scikit learn 예측 알고리즘 모델링

적합도 평가

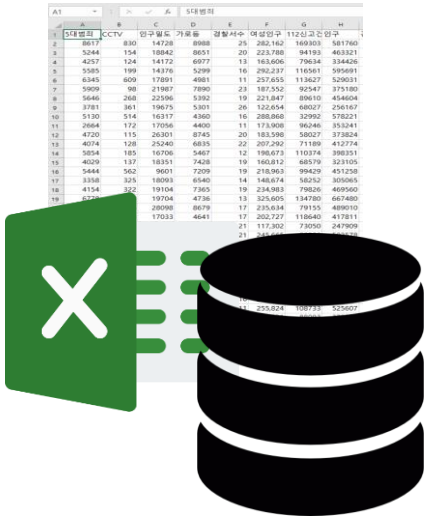
01	XGB Regression
02	Random Forest Regression
03	AdaBoost Regression
04	Linear Regression
05	Decision Tree Regression



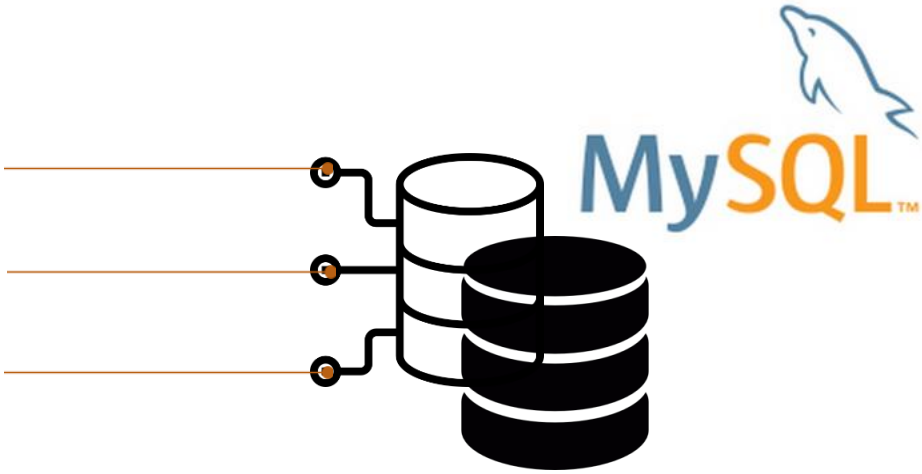
서비스 구현

MySQL

수집 데이터 변수 CSV

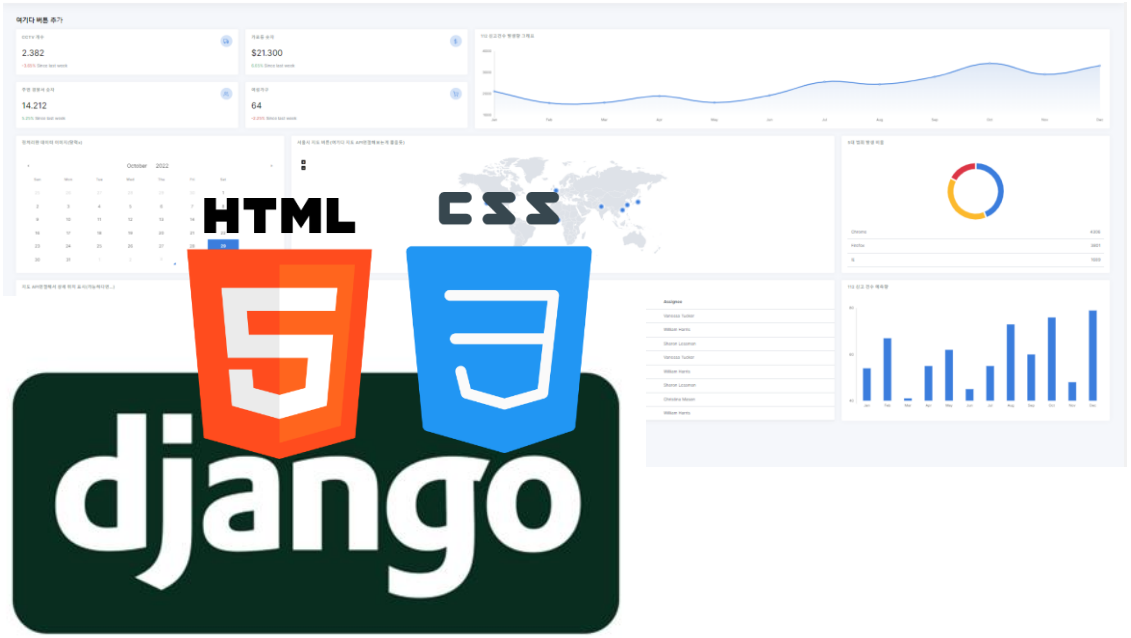


통합 데이터 변수 CSV



SQL 테이블로 변환해
DB 구축

서비스 구현



MySQL DB를 Django에 연동했으며
HTML 템플릿을 통해 서비스 화면을 구현했다

링크: https://beconomic.netlify.app/

Beaconomic - 강남구

강남구 강동구 강북구 강서구 관악구 광진구 구로구 금천구 노원구 도봉구 동대문구 동작구 마포구 서대문구 서초구 성동구 성북구 송파구 양천구 영등포구 용산구 은평구 종로구 중구 중랑구

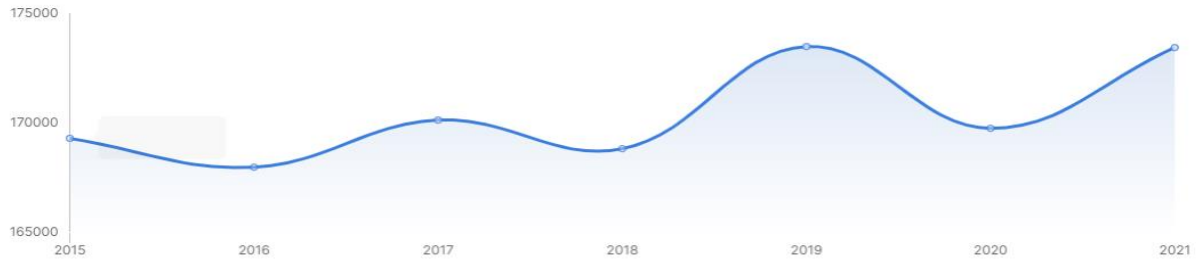
CCTV 개수
361
전년도 대비 +516개

가로등 숫자
14184
전년도 대비 -871개

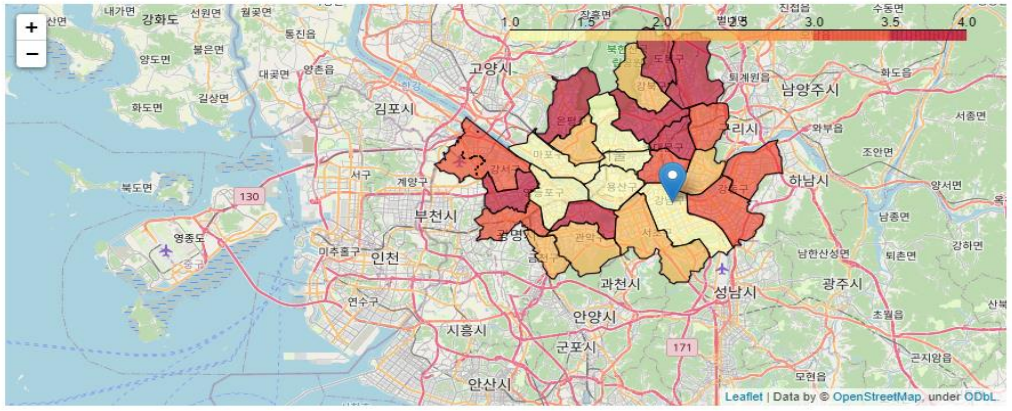
주변 경찰서 숫자
22
전년도 대비 +0개

여성가구
66597
전년도 대비 -2636명

112 신고건수 발생량 그래프



경찰서 상세 위치 성범죄자 알림e 실시간 cctv 위치 심리상담센터 행정안전부 홈페이지



5대 범죄 발생 비율

살인	12
강도	25
강간,강제추행	578
절도	2372
폭력	3159

112 신고 건수 예측량:122966

종합 치안 안전 지수:72.5

안전종합지수 등급

A등급	✓
B등급	×
C등급	×
D등급	×

발전방향,
개발후기

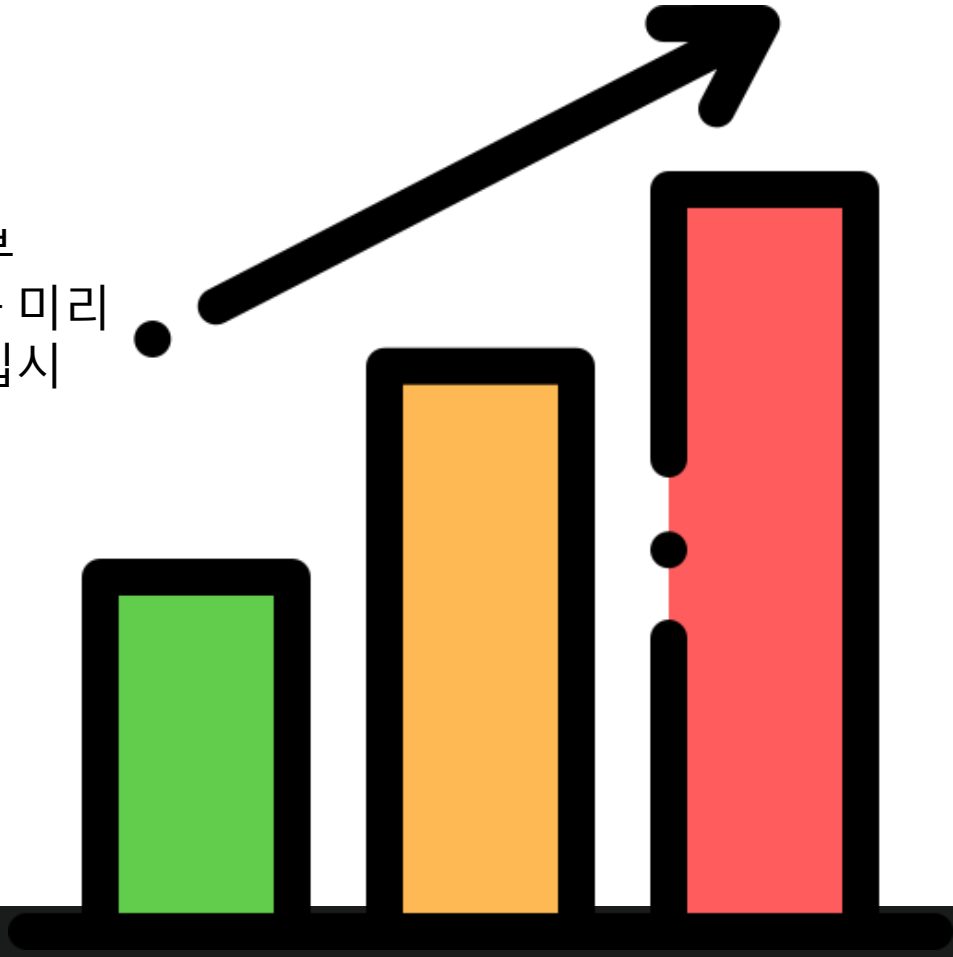
미래 발전,활용방안

안전 귀가 경로 추천 서비스

실시간 데이터를 수집하고 지역당 세세한 세부 데이터를 다양하게 수집해서 안전 귀가 경로를 추천해주는 알고리즘을 모델링

위험지역 미리 알림 서비스

실시간 데이터를 수집하고 지역당 세세한 세부 데이터를 다양하게 수집해서 세부 위험지역을 미리 예측하는 알고리즘을 모델링해 그 근처에 진입시 미리 알림을 해주는 서비스



개발 후기

박준영

직접적으로 서비스를 구현하는데 있어서 모든게 처음이였기에 쉬운 것이 하나 없었다. DB를 서버에 연동하는 것부터 심지어 Django 프로젝트를 만드는 파일을 만들어 설정하는 것 까지,,,
구현하지 못한 기능 또한 많았다. 고객들이 클릭한 정보를 모아 선호도 분석과 같은 것들은 구현할 물리적 시간조차 없었다. 하지만 서비스를 기획부터 구현까지 모든 과정에 참여해 볼 수 있어 의미가 있었다.

이원우

느낀 점 - 막상 프로젝트를 시작하려니 간단한 것들에 헤메는 경우가 많았다. 그래도 프로젝트 전에 비해 성장한 것 같아서 기분이 좋다. 특히 지도모양 버튼을 구현하는데에 꽤 많은 시간을 소비했지만 실패했다. 더욱 성장해서 원하는 아이디어를 그대로 구현해낼수있는 능력을 갖추고 싶다.

박창훈

회귀를 통해 112신고건수를 예측하는 프로젝트를 진행하면서 결과물을 구현해내기까지 많은 시행착오가 있었지만 문제를 해결해가면서 성취감을 느낄 수 있었다.
짧은 시간동안 웹 프로세스의 전반적인 과정을 다룰 수 있어서 많은 것을 배울 수 있는 기회였다.

장민준

무엇인가를 해보면서 느끼는 점이 많았고, 내가 좀 더 잘했으면 더 쉽게 프로젝트를 하지 않았을까 하는 아쉬움이 들었다.

허지훈

구현하기 전에는 이것 저것 많이 생각을 해놨는데 막상 구현하려고 하니까 잘되지 못하는 경우가 많았다. 스파크 , 웹크롤링 등 다양한 방법을 이번 프로젝트때 해보고 싶었지만 시간적 한계로 인해 그러지 못했다. 이번 프로젝트에서 엔지니어링 부분으로 보다 발전하고 싶었는데 프론트적인 부분에 포커싱이 맞추어져서 다소 아쉬웠다. 그래도 하나의 웹사이트를 구현하는데 있어서 많은 자신감을 가지게 되어서 좋았다.

경청해주셔서
감사합니다.

자료 출처

서울시 열린 데이터 광장

공공 데이터 포탈

경기 연구원
경기도 지역안전지수분석 및 개선 방안Analysis
of Regional Safety Index and
Improvement Plan in Gyeonggi Province