범죄율 예측을 위한 간단한 모델

STA617 Final Project Report

통계학과 2021021352 윤아연

- 목차 -

1. 서론

본 프로젝트에서는 각 지역의 주어진 정보를 사용해 범죄율을 예측할 수 있는 모델을 구축해 범죄율 예측하고 범죄 예방에 도움이 되고자 한다. 더불어 해석이 쉬운 간단한 모델을 만듦으로써 범죄율에 영향을 미치는 요인을 발견하고 해당 요인이 범죄율에 어떤 영향을 미치는지 파악하고자 한다.

2. 데이터 소개 및 전처리

본 프로젝트에서 모델 구축을 위해 사용한 데이터는 'Communities and Crime' 데이터이다. Communities and Crime 데이터는 2215개의 관측치와 124개의 설명변수, 8개의 반응변수, 5개의 id 변수로 구성된다. 해당 데이터의 관측치는 미국의 각 지역(community)를 의미한다. 124개의 설명변수는 각 지역의 인종별 비율, 가구 소득, 교육, 이민자 비율 등 지역사회 전반에 관련된 변수들이며 18개의 반응변수는 살인, 강간, 강도, 폭행, 빈집털이, 자동차 털이, 절도, 방화 8개 범죄에 대한 범죄 건수와 10만 명당 범죄 건수, 10만 명당 강력범죄(살인, 강간, 강도, 폭행) 건수, 10만 명당 비폭력 범죄(빈집털이, 자동차 털이, 절도, 방화) 건수이다. 본 프로젝트에서는 범죄의 심각성을 고려해 10만 명당 강력범죄 건수만을 반응변수로 사용했다.

데이터 분석 및 모델 구축에 앞서 해당 데이터의 결측치를 제거했다. 먼저 124개의 설명변수 중 결측치 비율이 50%가 넘는 22개의 설명변수를 삭제한 후 여전치 결측치가 존재하는 관측치 314개를 삭제했다. 결측치 처리 후 분석에 사용되지 않을 5개의 id 변수와 17개의 반응변수 역시 제거했다. 이러한 전처리 과정을 거쳐 최종적으로 분석에 사용한 데이터는 1901개의 관측치, 102개의 설명변수, 그리고 1개의 반응변수로 이루어진 데이터이다. 전처리 후 분석에 사용된 최종 데이터의 자세한 변수 설명은 부록을 참고하기 바란다.

전처리가 끝난 데이터는 모델 적합에 사용할 train set(n=1401)과 최종적으로 모델의 성능을 확인하는데 사용할 test set(n=500)으로 나누어주었다. 모델 적합을 위해 train set에 대해서는 추가적인 처리를 진행했는데, 첫 번째로 설명변수를 표준화했으며 두 번째로 10개의 fold로 나눠 분석 과정에서 10-fold cross validation이 가능하도록 했다.

3. 분석 방법론

분석에 사용될 데이터는 총 102개의 설명변수를 포함하고 있다. 만약 102개의 설명변수가 모델에 모두 사용될 경우 모델이 복잡해 해석이 어려울 수 있고 불필요한 설명변수가 포함될 경우 오히려 모델의 성능이 저하될 수 있다. 따라서 본 프로젝트에서는 LASSO(Least Absolute Shrinkage and Selection Operator)를 사용한 변수 선택으로 모델을 간단히 하고자 한다. 또한 반응변수와 설명변수의 관계를 고려한 차원축소 방법인 SIR(Sliced Inverse Regression)을 통해 모델의 설명력을 최대한 잃지 않고 설명변수의 차원을 축소하는 방법도 시도한다. 이러한 방법론들을 통해 본 프로젝트에서는 모델에 사용되는 설명변수를 10개 이하로 줄여 간단한 모델을 만들고자 한다.

강력범죄율에 영향을 미치는 요인 및 영향력 파악을 위해 해석이 용이한 모델을 사용하고자 한다. 따라서 본 프로젝트에서 사용될 모델은 쉽게 해석이 가능한 GLM(Generalized Linear Model)과 GLM보다 유연해(flexible) 예측성능을 더 높일 수 있는 GAM(Generalized Additive Model)을 사용한다. GLM과 GAM에서 반응변수의 분포는 포아송 분포로 가정했으며 link function은 square root link를 사용했다. 포아송 분포 가정을 위해 반응변수를 반올림해

자연수로 만들어주었다.

본 프로젝트에서는 데이터 분석 및 모델 적합을 위한 프로그램으로 R을 사용했다. LASSO 방법론을 적용하기 위해 R glmnet 패키지의 glmnet 함수를 사용했다(argument setting : alpha = 1). SIR 방법론을 적용하기 위해 R dr 패키지의 dr 함수를 사용했다(argument setting : method = 'sir'). GLM을 사용하기 위해서는 R stats 패키지의 glm 함수를, GAM을 사용하기 위해서는 R gam 패키지의 gam 함수를 사용했다(공통 argument setting : family = poisson('sqrt')). GAM에 사용될 비선형 함수에 대해서는 natural splines는 R splines 패키지의 ns 함수를, smoothing splines는 R gam 패키지의 s 함수를 사용했다.

4. 분석 과정

4-1. LASSO 모델

간단한 모델을 구축하기 위해 첫 번째로 LASSO 변수 선택 방법을 사용하였다. LASSO를 적용해 GLM을 적합하였으며, 해당 모델에서 shrinkage penalty인 λ값에 대한 tuning이 필요하므로 10-fold cross validation을 통한 grid search parameter tuning을 진행하였다. Grid search에는 (-4, 1) 사이에서 등간격으로 생성된 100개의 $\log_{10}\lambda$를 사용했다.

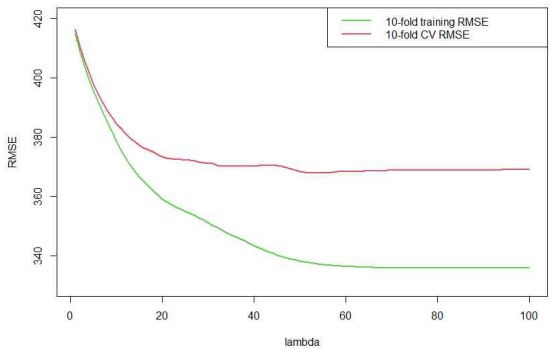

Figure 1. LASSO GLM의 10-fold cross validation을 통한 parameter tuning

Cross validation 결과 $\lambda = 0.0210$일 때 가장 작은 CV RMSE(=367.87)을 얻을 수 있었다. 하지만 해당 λ값을 사용한 모델은 102개의 설명변수 중 86개의 설명변수를 선택해 여전히 복잡한 모델임을 확인할 수 있었다.

더 간단한 모델을 만들기 위해 LASSO GLM에서 추가적인 변수선택을 진행하였다. Grid search에 사용된 λ값 중 크기가 큰 첫 9개 λ값을 사용했을 경우 Table 1.과 같이 설명변수가 10개 이하로 선택되었다.

Table 1.에서 선택된 3~9개의 설명변수들은 큰 shrinkage penality에도 회귀계수가 0으로 shrink되지 않았으므로 반응변수에 중요한 영향을 미치는 핵심 설명변수일 것이다. 추가적 변수 선택을 위해 해당 설명변수들 중 몇 개의 설명변수를 사용하는 것이 적절한지 10-fold cross validation을 통해 확인해 보았다. 여기서는 GLM보다 예측 성능을 높일 수 있도록 GAM을 사용하였다. GAM에는 effective degrees of freedom(=df_λ)가 4인 smoothing splines과 degrees of freedom(=df)가 4인 natural cubic splines를 전체 설명변수에 비선형 함수로 적용하였다.

Table 2.를 통해 10-fold cross validation 결과를 확인할 수 있다. GAM의 비선형 함수가 smoothing splines일 때는 6개의 설명변수를 사용한 모델이, natural splines일 때는 8개의

설명변수를 사용한 모델이 가장 작은 CV RMSE를 보여주었다. 앞서 86개의 설명변수를 사용했던 LASSO GLM과 이 두 모델을 비교했을 때, 설명변수의 수는 훨씬 줄어들었으나 CV RMSE는 더 작아졌으므로 변수 선택과 예측성능 향상이 성공적으로 이루어졌음을 확인할 수 있었다.

1) + 표시는 위의 행들에 적힌 모든 설명변수도 포함한다는 의미

λ	p	선택된 설명변수
$\lambda_1 = 10.00$	3	racePctWhite (백인 인구 비율) PctKids2Par (양부모 가정에 속한 아동 비율) PctKidsBornNeverMar (미혼 가정에 속한 아동 비율)
$\lambda_2 = 8.90$ $\lambda_3 = 7.92$ $\lambda_4 = 7.05$ $\lambda_5 = 6.28$	4	+ FemalePctDiv (여성 이혼율)[1]
$\lambda_6 = 5.59$	6	+ PctPersDensHouse (거주 인구 대비 좁은 집의 비율) TotalPctDiv (전체 이혼율)
$\lambda_7 = 4.89$ $\lambda_8 = 4.43$	8	+ MalePctDivorce (남성 이혼율) HouseVacant (빈 집의 수)
$\lambda_9 = 3.94$	9	+ MedRentPctHousInc (가구 소득에서 임대료가 차지하는 비율의 중앙값)

Table 1. LASSO GLM에서 설명변수가 10개 이하로 선택된 λ값과 그에 해당하는 설명변수

	smoothing splines ($df_\lambda = 4$)	natural cubic splines ($df = 4$)
p = 3	383.07	384.69
p = 4	376.65	379.41
p = 6	367.05	371.49
p = 8	437.24	365.32
p = 9	444.03	367.55

Table 2. 추가적 변수 선택을 위한 GAM의 10-fold cross validation 결과(CV RMSE)

연구자는 Table 2.의 smoothing splines와 natural cubic splines에서 가장 좋은 성능을 보여준 두 모델을 선택해 각 설명변수에 대한 비선형 함수의 parameter(df_λ, df) tuning 및 세부적인 모델 수정을 통해 'LASSO GAM'이라는 최종 모델을 선택하였다.

Table 3.을 통해 LASSO GAM을 살펴보면 총 6개의 설명변수가 사용되었고 PctKidsBornNeverMar 변수를 제외한 5개의 설명변수에는 natural cubic splines가 비선형 함수로 사용되었다. LASSO GAM의 10-fold cross validation RMSE는 356.27로 Table 2.에 나타난 모델들보다 세부적 수정을 통해 예측성능이 향상되었다.

설명변수	비선형 함수
racePctWhite (백인 인구 비율)	natural cubic splines ($df = 3$)
PctKids2Par (양부모 가정에 속한 아동 비율)	natural cubic splines ($df = 5$)
PctKidsBornNeverMar (미혼 가정에 속한 아동 비율)	Identity (=linear)
MalePctDivorce (남성 이혼율)	natural cubic splines ($df = 6$)
HouseVacant (빈 집의 수)	natural cubic splines ($df = 6$)
PctPersDensHouse (거주 인구 대비 좁은 집의 비율)	natural cubic splines ($df = 2$)

Table 3. LASSO 방법론에서 최종적으로 선택된 모델 - LASSO GAM

4-2. SIR 모델 (1)

간단한 예측 모델을 위해 연구자가 두 번째로 적용한 방법론은 SIR이다. SIR 차원축소를 진행해 간단한 모델을 만든 후 10-fold cross validation을 통해 설명변수를 1~10차원 중 몇 차원으로 축소하는 것이 적절한지 확인했다. 여기서는 GLM과 GAM을 모두 사용했으며 GAM의 모든 설명변수에 smoothing splines($df_\lambda = 4$)를 적용했다.

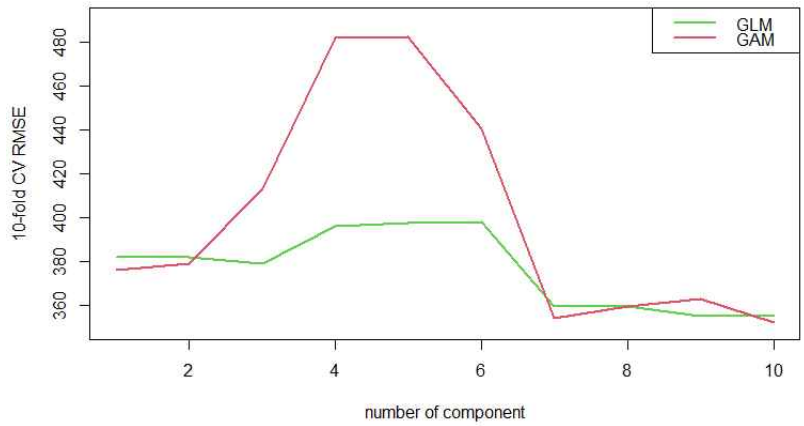

Figure 2. SIR 차원축소 후 10-fold cross validation 결과 - GLM, GAM

Figure 2.를 통해 10-fold cross validation의 결과를 살펴볼 수 있다. GLM의 경우는 설명변수로 9개의 component를 사용했을 때 가장 낮은 CV RMSE(=355.05)를 보였고 GAM의 경우는 설명변수로 10개의 component를 사용했을 때 가장 낮은 CV RMSE(=351.81)를 보였다. 또한 GAM에서는 3~6개의 component를 사용했을 때 GLM보다 특히 높은 CV RMSE를 보였다.

GLM과 GAM을 통틀어 가장 낮은 CV RMSE를 보여준 모델은 10개의 component를 설명변수로 사용한 GAM이었으므로 연구자는 해당 모델을 선택해 모델 개선을 진행하였다. 모델의 개선은 각 10개의 설명변수에 적용되는 비선형 함수의 종류 및 parameter를 조정하는 방법으로 진행되었다. 그 결과 '1차 SIR GAM'이라는 최종 모델을 선택하였다.

설명변수	비선형 함수	설명변수	비선형 함수
PC1	smoothing splines($df_\lambda = 4$)	PC6	smoothing splines($df_\lambda = 4$)
PC2	smoothing splines($df_\lambda = 2$)	PC7	smoothing splines($df_\lambda = 4$)
PC3	smoothing splines($df_\lambda = 4$)	PC8	smoothing splines($df_\lambda = 2$)
PC4	smoothing splines($df_\lambda = 3$)	PC9	smoothing splines($df_\lambda = 2$)
PC5	smoothing splines($df_\lambda = 2$)	PC10	smoothing splines($df_\lambda = 4$)

Table 4. SIR 방법론(1)에서 최종적으로 선택된 모델 - 1차 SIR GAM

Table 4.을 통해 1차 SIR GAM의 설명변수와 그에 해당하는 비선형 함수를 살펴볼 수 있다. 1차 SIR GAM의 10-fold cross validation RMSE는 349.00이었다.

4-3. SIR 모델 (2)

연구자는 앞서 1차 SIR GAM을 적합하는 과정에서 SIR 차원축소 시 계산된 각 eigenvector에서의 loading값에 주목했다. Table 5.에서 확인할 수 있듯이 각 eigenvector에서 102개의 설명변수 중 대부분의 설명변수에 대한 loading의 절댓값이 0.1보다 작게 나타났다.

eigenvector	1st	2nd	3rd	4th	5th
# of |loading| ≧ 0.1	10	12	22	11	15
eigencvector	6th	7th	8th	9th	10th
# of |loading| ≧ 0.1	20	9	7	15	13

Table 5. 1차 SIR GAM의 각 eigenvector에서 loading의 절댓값이 0.1을 넘는 변수의 수

Eigenvector에서 loading 절댓값은 각 설명변수가 차원축소에 얼마나 영향을 미치는지 알 수 있는 지표이다. Table 5.에서와 같이 대부분의 설명변수가 무시할 수 있을 정도의 작은 loading값을 갖는다는 것은 소수의 설명변수만이 데이터의 차원축소에 관여한다는 것이다. 이런 경우 loading의 절댓값이 작은 설명변수를 모델에서 제외하더라도 불필요한 정보이므로 정보의 손실이 크게 일어나지 않을 것이다. 따라서 연구자는 적은 양의 정보로 1차 SIR GAM과 비슷한 성능을 가진 모델을 만들기 위해 차원축소에 큰 영향을 미치는 소수의 변수들을 선택해 SIR 차원축소를 다시 진행하였다. 영향력 있는 변수들을 선택하는 방법은 다음과 같다.

<SIR 변수 선택>

n개의 관측치와 p개의 설명변수를 가진 데이터에서

Step 1. SIR 차원축소 방법을 사용해 eigenvector를 계산

Step 2. 첫 $k(<p)$개의 eigenvector를 선택 후, 각 eigenvector마다 loading의 절댓값이 큰 상위 $m(<n)$개의 설명변수를 추출. 이때, 추출된 $m \times k$개의 설명변수 중 중복을 제거한 설명변수의 개수를 p_1이라고 한다.

Step 3. 다음과 같은 방법으로 각 설명변수의 포함비율 α_i를 계산한다. ($i = 1, \cdots, p_1$)

: i번째 설명변수가 k개의 설명변수 그룹 중 u개의 설명변수 그룹에 속한다고 하자.

그렇다면 $\alpha_i = \frac{u}{k}$를 i번째 설명변수의 포함비율이라고 한다. $(u = 1, \cdots, k)$

Step 4. $\alpha_i > 0.5$인 설명변수만을 최종적으로 선택하고 이때 최종 선택된 설명변수의 수를 p_2라고 한다.

본 프로젝트에서는 $k = m = 30$으로 SIR 변수 선택을 진행하였다. 그 결과 Step 2.에서는 $p_1 = 75$개의 설명변수가 선택되었으며 Step 4.에서는 $p_2 = 27$개의 설명변수가 선택되었다. 최종적으로 선택된 27개의 변수를 사용해 SIR 차원축소를 진행한 후 4-2.에서와 같이 10-fold cross validation을 진행하였다.

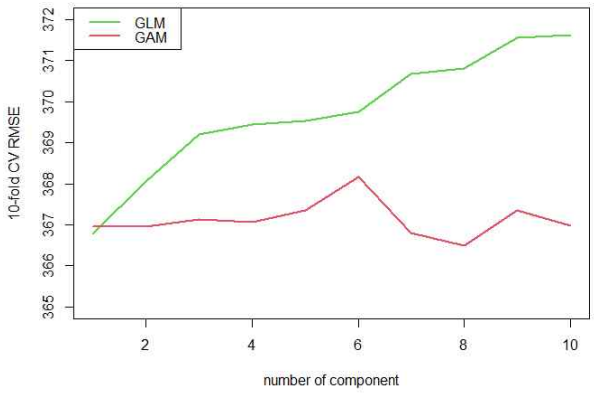

Figure 3. SIR 변수 선택 후 SIR 차원축소 10-fold cross validation 결과 - GLM, GAM

Figure 3.을 통해 10-fold cross validation 결과를 살펴볼 수 있다. GLM의 경우 설명변수로 1개의 component를 사용했을 때 가장 낮은 CV RMSE(=366.79)를 보였고, GAM의 경우 8개의 component를 사용했을 때 가장 낮은 CV RMSE(=366.49)를 보였다. 1~10개의 component에서 비슷한 성능을 보이는 GAM과 달리 GLM의 경우 component의 수가 많아질수록 모델의 성능이 떨어지는 경향을 보였다.

GLM과 GAM을 통틀어 8개의 component를 사용한 GAM이 가장 낮은 CV RMSE를 보여주었으므로 연구자는 해당 모델을 선택해 비선형 함수 및 parameter 조정을 진행하였다. 모델 개선 과정을 거쳐 '2차 SIR GAM'이라는 모델이 최종적으로 선택되었다.

설명변수	비선형 함수	설명변수	비선형 함수
PC1	smoothing splines($df_\lambda = 4$)	PC5	natural cubic splines($df = 4$)
PC2	smoothing splines($df_\lambda = 6$)	PC6	natural cubic splines($df = 4$)
PC3	smoothing splines($df_\lambda = 6$)	PC7	smoothing splines($df_\lambda = 8$)
PC4	smoothing splines($df_\lambda = 3$)	PC8	natural cubic splines($df = 5$))

Table 5. SIR 방법론(2)에서 최종적으로 선택된 모델 - 2차 SIR GAM

Table 5.를 통해 2차 SIR GAM의 설명변수와 그에 해당하는 비선형 함수를 살펴볼 수 있으며 2차 SIR GAM의 10-fold cross validation RMSE는 367.72였다. SIR 변수 선택을

진행하지 않은 모델인 1차 SIR GAM과 비교했을 때, 1차 SIR GAM의 CV RMSE(=349.00)가 더 낮았다.

5. 분석 결과

앞선 분석 과정에서 최종적으로 다음과 같은 3개의 모델이 선택되었다. - LASSO GAM, 1차 SIR GAM, 2차 SIR GAM. 3가지 모델의 예측성능을 확인하기 위해 test RMSE와 범죄 위험 단계 예측률을 사용하고자 한다. Test set을 모델에 사용할 수 있도록 표준화 및 차원축소를 진행할 때에는 train set에서 얻은 정보를 사용했다.

5-1. Test RMSE

모델	10-fold cross validation RMSE	Test RMSE
LASSO GAM	356.27	362.67
1차 SIR GAM	349.00	326.21
2차 SIR GAM	362.72	372.29

Table 6. LASSO GAM, 1차 SIR GAM, 2차 SIR GAM의 test RMSE

Table 6.에는 3가지 모델의 CV RMSE와 test RMSE가 나타나있다. 결과를 살펴보면 가장 낮은 CV RMSE를 보여준 1차 SIR GAM이 test RMSE에서도 가장 낮은 값을 보여주었다. 이를 통해 1차 SIR GAM의 성능이 가장 좋다는 결론을 내릴 수 있다. 또한 2차 SIR GAM은 모델 성능이 가장 떨어지는 것을 볼 수 있는데, 이를 통해 SIR 변수 선택에서 생각보다 많은 정보가 손실되었음을 짐작할 수 있다.

5-2. 범죄 위험 단계 예측

범죄 예측 모델의 목적은 해당 지역이 얼마나 위험한지 예측하는 것이므로 실질적으로 10만 명당 강력범죄 건수를 정확히 예측할 필요는 없다. 핵심은 '안전 지역을 안전 지역으로, 위험 지역을 위험 지역으로' 잘 예측하는 것이다. 따라서 연구자는 범죄 위험 단계를 Table 7.과 같이 총 4가지 단계로 나누어 각 모델의 범죄 위험 단계 예측성능을 알아보고자 했다. 이때 범죄 위험 단계의 기준은 train set의 10만 명당 강력범죄 건수에 대한 25%, 50%, 75% 분위수를 사용했다.

10만 명당 강력범죄 건수(= Y)	$Y < 161$	$161 \leq Y < 374$	$374 \leq Y < 783$	$783 \leq Y$
범죄 위험 단계	**1 (매우 안전)**	**2 (안전)**	**3 (위험)**	**4 (매우 위험)**

Table 7. 범죄 위험 단계

Table 8.을 통해 3가지 모델의 범죄 위험 단계 예측 결과를 살펴볼 수 있다. 4단계 매우 위험 단계인 4단계에 대한 예측성능은 1차 SIR이 133개의 4단계 지역 중 91개를 맞게 분류해 가장 좋았다. LASSO GAM과 1차 SIR GAM은 4단계를 1단계로 예측하거나 1단계를 4단계로 예측하는 경우는 발생하지 않았지만 2차 SIR GAM은 실제 4단계인 지역을 1단계로 예측하는 경우가 발생했다. 비록 1건이지만 매우 위험한 지역을 매우 안전한 지역으로 분류하는 것은 좋은 범죄 예측 모델이라고 볼 수 없으므로 test RMSE에서와 같이 2차 SIR GAM의 예측성능이 좋지 않다는 결론을 내릴 수 있었다.

		예측 위험 단계														총 합
		LASSO GAM					1차 SIR GAM					2차 SIR GAM				
		1	2	3	4		1	2	3	4		1	2	3	4	
실제 위험 단계	1	**57**	57	5	0	1	**57**	54	8	0	1	**53**	60	6	0	119
	2	19	**75**	37	5	2	21	**75**	35	5	2	20	**71**	36	9	136
	3	1	17	**63**	31	3	1	18	**66**	27	3	1	24	**61**	26	112
	4	0	6	36	**91**	4	0	2	37	**94**	4	1	4	39	**89**	133
총 합		77	155	141	127		79	149	146	126		75	159	142	124	500

Table 8. 3가지 모델의 범죄 위험 단계 예측 결과

5-3. 모델의 해석

본 프로젝트의 목적은 예측성능이 좋은 범죄 예측 모델을 구축하는 것과 더불어 범죄율에 영향을 미치는 요인을 파악하는 것이었다. 이는 모델 해석을 통해 접근할 수 있다.

1차 SIR GAM과 2차 SIR GAM의 경우 GAM plot을 통해 설명변수로 사용된 각 component와 반응변수 사이의 관계를 시각적으로 파악할 수 있었다. 하지만 각 component에 대한 해석이 쉽지 않아 강력범죄율에 영향을 미치는 요인이 무엇인지 파악하기는 힘들다는 단점이 있었다.

반면 LASSO GAM의 경우 차원축소가 아닌 변수 선택을 통해 만들어진 모델이므로 설명변수가 명확했고, 따라서 GAM plot을 통한 모델 해석이 어렵지 않았다.

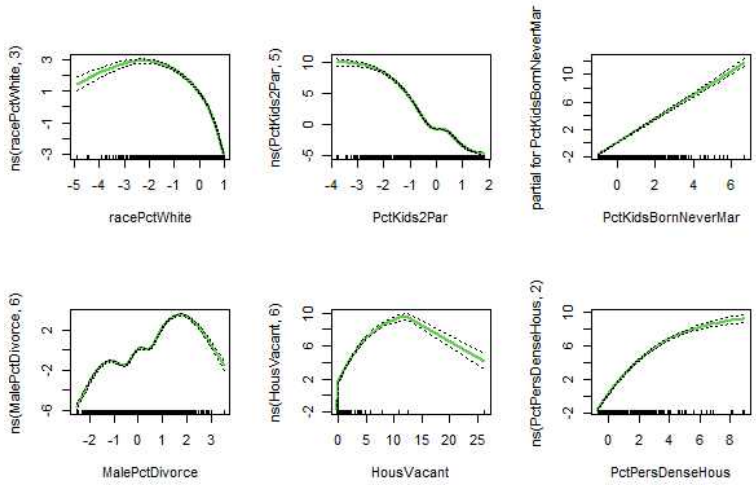

Figure 4. LASSO GAM의 GAM plot

Figure 4.을 통해 LASSO GAM을 해석할 수 있다. 백인비율, 양부모 가정에 속한 아동 비율이 증가할수록 강력범죄율이 감소하는 경향을 보인다. 반면 미혼 가정에 속한 아동 비율, 남성 이혼율, 거주 인구 대비 좁은 집의 비율이 증가할수록 강력범죄율도 함께 증가하는 것을 알 수 있다. 빈집의 수의 경우, 빈집의 수가 증가할수록 강력범죄율도 증가하다가 10~15 사이에서 강력범죄율이 급감하는 모습을 보인다. 하지만 이는 그래프의 x축에 나타난 관측치의 분포로 볼 때 아웃라이어의 영향이 강한 것으로 판단된다. 즉, 혼인을 통해 가정이 안정된 사회일수록 강력범죄율이 줄어드는 경향이 있으며 거주공간의 상황도 범죄율에 영향을 미치는 것으로 나타났다.

6. 결론

본 프로젝트에서는 범죄 예측 모델을 만들어 범죄율에 영향을 미치는 요인을 파악하고 각 지역의 범죄율을 예측해 범죄 예방에 도움이 되고자 했다. Communities and Crime 데이터를 분석한 결과 최종적으로 LASSO GAM, 1차 SIR GAM, 2차 SIR GAM의 3가지 모델을 선택할 수 있었다. 예측성능은 1차 SIR GAM이 가장 좋았으며 모델 해석의 측면에서는 LASSO GAM이 좋았다.

본 프로젝트의 의의는 102개의 설명변수를 모델에 그대로 사용하지 않고 변수 선택 및 차원축소 과정을 거쳐 최종 모델에 사용되는 설명변수를 10개 이하로 줄인 것에 있다. LASSO 방법론에서는 2번에 걸친 변수 선택을 진행하였고 SIR을 사용해 차원축소뿐만 아니라 변수 선택도 진행했다는 점이 본 프로젝트의 핵심이다.

하지만 이상치 제거를 하지 못한 점, 1차 SIR GAM과 2차 SIR GAM의 해석이 어려운 점, 그리고 2차 SIR GAM의 성능이 생각보다 떨어지는 점을 본 프로젝트의 한계점으로 볼 수 있다. 이러한 점을 보완한다면 더 좋은 범죄 예측 모델을 구축할 수 있을 것이다.

---

7. 참고문헌

1. James, G., Witten, D., Hastie, T.,, Tibshirani, R. (2013). An Introduction to Statistical Learning: with Applications in R . Springer.
2. 허명회. (2014). 응용데이터분석. 자유아카데미
3. Sanford Weisberg (2002). Dimension Reduction Regression in R. Journal of Statistical Software, 7, 1-22 URL: http://www.jstatsoft.org
4. Trevor Hastie (2020). gam: Generalized Additive Models. R package version 1.20. https://CRAN.R-project.org/package=gam
5. Jerome Friedman, Trevor Hastie, Robert Tibshirani (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1) 1-22. URL https://www.jstatsoft.org/v33/i01/.
6. R Core Team (2021). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

8. 부록

8-1. 데이터 변수 설명

전처리를 거쳐 분석에 사용된 최종 데이터는 다음과 같은 103개의 변수를 포함한다.

<설명변수 : 102개 (모두 수치형 변수)>

1. population: 지역의 인구
2. householdsize: 가구당 평균 인구
3. racepctblack: 아프리카계 미국인 인구 비율
4. racePctWhite: 백인 인구 비율
5. racePctAsian: 아시아계 인구 비율
6. racePctHisp: 히스패닉계 인구 비율
7. agePct12t21: 12~21세 인구 비율
8. agePct12t29: 12~29세 인구 비율
9. agePct16t24: 16~24세 인구 비율
10. agePct65up: 65세 이상 인구 비율
11. numbUrban: 도시로 분류된 지역에 거주하는 인구
12. pctUrban: 도시로 분류된 지역에 거주하는 인구 비율
13. medIncome: 중위 소득
14. pctWWage: 1989년 기준 임금/급여소득이 있는 가구 비율
15. pctWFarmSelf: 1989년 기준 농업소득/사업소득이 있는 가구 비율
16. pctWInvInc: 1989년 기준 투자/임대소득이 있는 가구 비율
17. pctWSocSec: 1989년 기준 사회보장소득이 있는 가구 비율
18. pctWPubAsst: 1989년 기준 공공부조 소득이 있는 가구 비율
19. pctWRetire: 1989년 기준 은퇴소득(연금)이 있는 가구 비율
20. medFamInc: 중위 가구소득
21. perCapInc: 1인당 소득
22. whitePerCap: 백인의 1인당 소득
23. blackPerCap: 아프리카계 미국인의 1인당 소득
24. indianPerCap: 인디언의 1인당 소득
25. AsianPerCap: 아시아계 미국인의 1인당 소득
26. OtherPerCap: '기타'인종의 1인당 소득
27. HispPerCap: 히스패닉계 미국인의 1인당 소득
28. NumUnderPov: 빈곤 인구
29. PctPopUnderPov: 빈곤 인구 비율
30. PctLess9thGrade: 25세 이상 인구 중 최종 학력이 9학년 미만인 비율
31. PctNotHSGrad: 25세 이상 인구 중 최종 학력이 고졸 미만인 비율
32. PctBSorMore: 25세 이상 인구 중 최종 학력이 학사 이상인 비율
33. PctUnemployed: 16세 이상 인구 중 실업 인구 비율
34. PctEmploy: 16세 이상 인구 중 고용된 인구 비율
35. PctEmplManu: 16세 이상 인구 중 제조업 종사자 비율
36. PctEmplProfServ: 16세 이상 인구 중 서비스직 종사자 비율

37. PctOccupManu: 16세 이상 인구 중 제조업 종사 경험이 있는 인구 비율
38. PctOccupMgmtProf: 16세 이상 인구 중 관리직/전문직 종사자 비율
39. MalePctDivorce: 남성 이혼율
40. MalePctNevMarr: 남성 미혼율
41. FemalePctDiv: 여성 이혼율
42. TotalPctDiv: 전체 이혼율
43. PersPerFam: 가족 당 평균 인원
44. PctFam2Par: 양부모 가정 비율
45. PctKids2Par: 양부모 가정에 속하는 아동 비율
46. PctYoungKids2Par: 양부모 가정이면서 4세 이하 자녀가 있는 가정 비율
47. PctTeen2Par: 양부모 갖ㅇ에 속하는 12-17세 비율
48. PctWorkMomYoungKids: 6세 이하 자녀를 가진 직장인 어머니의 비율
49. PctWorkMom: 18세 이하 자녀를 가진 직장인 어머니의 비율
50. NumKidsBornNeverMar: 미혼 가정에 속하는 아동의 수
51. PctKidsBornNeverMar: 미혼 가정에 속하는 아동 비율
52. NumImmig: 총 이민자 수
53. PctImmigRecent: 이민자 중 지난 3년 안에 이민 온 이민자 비율
54. PctImmigRec5: 이민자 중 지난 5년 안에 이민 온 이민자 비율
55. PctImmigRec8: 이민자 중 지난 8년 안에 이민 온 이민자 비율
56. PctImmigRec10: 이민자 중 지난 10년 안에 이민 온 이민자 비율
57. PctRecentImmig: 전체 인구 중 지난 3년 안에 이민 온 이민자 비율
58. PctRecImmig5: 전체 인구 중 지난 5년 안에 이민 온 이민자 비율
59. PctRecImmig8: 전체 인구 중 지난 8년 안에 이민 온 이민자 비율
60. PctRecImmig10: 전체 인구 중 지난 10년 안에 이민 온 이민자 비율
61. PctSpeakEnglOnly: 영어만 구사하는 인구 비율
62. PctNotSpeakEnglWell: 영어를 잘 구사하지 못하는 인구의 비율
63. PctLargHouseFam: 대가족 비율 (가구 인구가 6명 이상)
64. PctLargHouseOccup: 자가에 거주하는 대가족 비율 (가구 인구가 6명 이상)
65. PersPerOccupHous: 평균 가구 인구
66. PersPerOwnOccHous: 자가에 사는 가구의 평균 가구 인구
67. PersPerRentOccHous: 임대 주택에 사는 가구의 평균 가구 인구
68. PctPersOwnOccup: 자가에 사는 인구 비율
69. PctPersDenseHous: 거주 인구 대비 좁은 주택 비율 (한 방에 한 명 이상)
70. PctHousLess3BR: 침실이 3개 미만인 주택 비율
71. MedNumBR: 침실 수의 중앙값
72. HousVacant: 빈 주택 비율
73. PctHousOccup: 주택 거주율
74. PctHousOwnOcc: 자가 거주 가구 비율
75. PctVacantBoarded: 판자로 마감된 빈 주택의 비율
76. PctVacMore6Mos: 6개월 이상 비어있는 빈 주택의 비율
77. MedYrHousBuilt: 주택 건설 연도의 중앙값

78. PctHousNoPhone: 전화기가 없는 주택의 비율
79. PctWOFullPlumb: 완전한 배관시설이 없는 주택의 비율
80. OwnOccLowQuart: 주택 매매가격 하한 사분위수
81. OwnOccMedVal: 주택 매매가격 중앙값
82. OwnOccHiQuart: 주택 매매가격 상한 사분위수
83. OwnOccQrange: 주택 매매가격 상한-하한 사분위수의 차이
84. RentLowQ: 주택 임대료 하한 사분위수
85. RentMedian: 주택 임대료 중앙값
86. RentHighQ: 주택 임대료 상한 사분위수
87. RentQrange: 주택 임대료 상한-하한 사분위수의 차이
88. MedRent: 총 주택 임대료 중앙값
89. MedRentPctHousInc: 가구 소득에서 주택 임대료가 차지하는 비율의 중앙값
90. MedOwnCostPctInc: 주택 담보 대출이 있는 소유주의 경우 가구 소득에서 소유주 비용(owner cost)이 차지하는 비율
91. MedOwnCostPctIncNoMtg: 주택 담보 대출이 없는 소유주의 경우 가구 소득에서 소유주 비용(owner cost)이 차지하는 비율
92. NumInShelters: 노숙자 쉼터에 거주하는 인구
93. NumStreet: 노숙자 수
94. PctForeignBorn: 출생지가 외국인 인구
95. PctBornSameState: 지금 거주하는 주(state)에서 태어난 인구 비율
96. PctSameHouse85: 5년 전과 같은 집에 거주하는 인구 비율
97. PctSameCity85: 5년 전과 같은 도시에 사는 인구 비율
98. PctSameState85: 5년 전과 같은 주에 사는 인구 비율
99. LandArea: 평방마일 당 택지 면적
100. PopDens: 평방마일 당 인구밀도
101. PctUsePubTrans: 통근 시 대중교통을 이용하는 인구 비율
102. LemasPctOfficDrugUn: 마약반에 배정된 경찰의 비율

<반응변수 : 1개>
103. ViolentCrimesPerPop: 10만 명당 강력범죄 건수 (자연수)

8-2. Source Code (R)

```
library(dplyr)
library(tidyr)
library(dr)
library(gam)
library(glmnet)
library(splines)

crime <- read.csv('crimedata.csv')

#####################
####  1. 전처리  ####
```

```
#####################
# 해당 데이터는 결측치가 '?'로 표시되어있음
NA.ind <- rep(0, ncol(crime))
for(i in 1:ncol(crime)){
  NA.ind[i] <- is.character(crime[,i])
}
NA.ind <- which(NA.ind == TRUE)
NA.rate <- apply(crime[,NA.ind], 2, function(x) mean(x == '?'))
round(NA.rate[NA.rate <= 0.5], 3)

# 결측치 비율이 50%가 넘는 변수는 삭제
round(NA.rate[NA.rate > 0.5], 3)
remove.var <- names(NA.rate)[NA.rate > 0.5]
crime.1 <- crime %>% dplyr::select(-c(remove.var))
str(crime.1)
# 24개 변수 제거(id 변수 2개, 설명변수 22개) -> 123개의 변수가 남음

# 결측치가 존재하는 observation 삭제
NA.obs <- which(apply(crime.1, 1, function(x) any(x == '?')) == TRUE)
length(NA.obs)
crime.2 <- crime.1[-NA.obs,]
str(crime.2) # 314개의 관측치 제거 -> 1901개의 관측치가 남음
summary(crime.2)

# 의미없는 id variable 제거
# + 수치형 변수인데 문자형 변수로 존재하는 변수의 형변환
crime.3 <- as.data.frame(apply(crime.2[,-c(1:3)], 2, as.numeric))
str(crime.3)
summary(crime.3)
crime <- crime.3

# train-test split
500/nrow(crime) # test set size = 500
set.seed(2021021302)
train.ind <- sample(nrow(crime), nrow(crime) - 500)
crime.train <- crime[train.ind,]; raw.train <- crime.train
crime.test <- crime[-train.ind,]; raw.test <- crime.test

# parameter tuning 및 validation을 위한 train/validation set split
set.seed(2021021352)
index <- sample(nrow(crime.train), nrow(crime.train))
cv.index <- list()
for(i in 1:10){
  cv.index[[i]] <- index[(140*(i-1)+1):(140*i)]
}
cv.index[[10]] <- c(cv.index[[10]], index[1401])
names(cv.index) <- paste0('fold', 1:10)

# standardize
crime.train.std <- scale(crime.train[,1:102], scale = T, center = T)
crime.train <- as.data.frame(cbind(crime.train.std, round(crime.train$ViolentCrimesPerPop)))
```

```
colnames(crime.train)[103] <- 'ViolentCrimesPerPop'
raw.train.std <- crime.train


# test set standardize (train set의 mean과 sd 사용)
# crime.test <- raw.test
mean <- matrix(apply(raw.train[,1:102], 2, mean), nrow(crime.test), 102, byrow = T)
sd <- matrix(apply(raw.train[,1:102], 2, sd), nrow(crime.test), 102, byrow = T)
crime.test.std <- (crime.test[,1:102]-mean)/sd
crime.test <- as.data.frame(cbind(crime.test.std, round(crime.test$ViolentCrimesPerPop)))
colnames(crime.test)[103] <- 'ViolentCrimesPerPop'
raw.test.std <- crime.test

########################
#### 2. DR by LASSO ####
########################
# 해당 데이터는 총 102개의 설명변수가 존재하므로 차원축소가 필요함
crime.train <- raw.train.std

#### lasso glm ####
y <- crime.train$ViolentCrimesPerPop

grid <- 10^seq(1, -4, length = 100)
train.ls.error <- val.ls.error <- rep(0, 100)
ftrain.ls.error <- fval.ls.error <- rep(0, 10)
for(i in 1:100){
  for(j in 1:10){
    # train-valid set split
    val.ind <- cv.index[[j]]
    crime.val <- as.matrix(crime.train[val.ind,-103]); y.val <- y[val.ind]
    crime.cv.train <- as.matrix(crime.train[-val.ind,-103]); y.train <- y[-val.ind]

    # fitting lm
    crime.lasso <- glmnet(crime.cv.train, y.train, alpha = 1, lambda = grid[i],
                          family = poisson('sqrt'))
    ftrain.ls.error[j] <- mean((y.train - predict(crime.lasso, crime.cv.train, type = 'response'))^2)
    fval.ls.error[j] <- mean((y.val - predict(crime.lasso, crime.val, type = 'response'))^2)
  }
  train.ls.error[i] <- mean(ftrain.ls.error)
  val.ls.error[i] <- mean(fval.ls.error)
}

train.ls.error
val.ls.error

windows(10, 10)
plot(1:100, sqrt(val.ls.error), type = 'l', lwd = 2, col = 2, ylab = 'RMSE', xlab = 'lambda',
     ylim = c(330, 420))
lines(1:100, sqrt(train.ls.error), type = 'l', lwd = 2, col = 3)
legend('topright', c('10-fold training RMSE', '10-fold CV RMSE'), lwd = c(2, 2), col = 3:2)
```

```
best.lambda <- grid[which.min(val.ls.error)] # 54번째가가 best 135326.8
best.crime.lasso <- glmnet(as.matrix(crime.train[, -103]), y, alpha = 1, lambda = best.lambda,
                           family = poisson('sqrt'))
sum(best.crime.lasso$beta != 0) # 86개의 변수. 16개의 변수는 0으로 shrink되었다.

sqrt(min(val.ls.error))

grid.crime.lasso <- glmnet(as.matrix(crime.train[,-103]), y, alpha = 1, lambda = grid,
                           family = poisson('sqrt'))
not.zero <- apply(grid.crime.lasso$beta, 2, function(x) sum(x != 0))
# 사용된 변수 개수와 validation RMSE
rbind(not.zero, sqrt(val.ls.error))
# 3~9개의 변수가 사용된 모델에서 GAM을 적용해보자!

round(grid[1:9], 4)

#### lasso gam ####
imp.var.coef <- list()
for(i in 1:9){
  imp.var.coef[[i]] <- grid.crime.lasso$beta[which(grid.crime.lasso$beta[,i] != 0), i]
}
imp.var.coef

imp.var.list <- list()
for(i in 1:9){
  j <- length(imp.var.list)
  imp.var <- names(grid.crime.lasso$beta[which(grid.crime.lasso$beta[,i] != 0), i])
  if(i != 1){
    if(length(imp.var.list[[j]]) != length(imp.var)) {imp.var.list[[j+1]] <- imp.var}
  }
  if(i == 1){imp.var.list[[1]] <- imp.var}
}
imp.var.list

########################
## model selection 1 ##
########################
## s.s
formula.list <- list()
for(i in 1:5){
  var.list <- imp.var.list[[i]]; n.var <- length(var.list)
  p <- ''
  for(j in 1:(n.var-1)){
    p <- paste(p, 's(', var.list[j], ')',' + ', sep = '')
  }
  p <- paste(p, 's(', var.list[n.var], ')',sep = '')
  formula.list[[i]] <- as.formula(paste('ViolentCrimesPerPop ~', p))
}

train.gamls.error <- val.gamls.error <- rep(0, 5)
```

```
ftrain.gamls.error <- fval.gamls.error <- rep(0, 10)
for(i in 1:5){
  f <- formula.list[[i]]
  for(j in 1:10){
    # train-valid set split
    val.ind <- cv.index[[j]]
    crime.val <- crime.train[val.ind,]
    crime.cv.train <- crime.train[-val.ind,]

    # fitting lm
    crime.gam.lasso <- gam(f, data = crime.cv.train, family = poisson('sqrt'))
    ftrain.gamls.error[j] <- mean((crime.cv.train$ViolentCrimesPerPop - fitted(crime.gam.lasso))^2)
    fval.gamls.error[j] <- mean((crime.val$ViolentCrimesPerPop -
                                   predict(crime.gam.lasso, crime.val, type = 'response'))^2)
  }
  train.gamls.error[i] <- mean(ftrain.gamls.error)
  val.gamls.error[i] <- mean(fval.gamls.error)
}
train.gamls.error
sqrt(val.gamls.error)

f <- formula.list[[3]]
crime.gam.lasso <- gam(f, data = crime.train, family = poisson('sqrt'))

windows(10, 10)
par(mfrow = c(2, 3))
plot(crime.gam.lasso, se = T, col = 3)

### n.s
formula.list <- list()
for(i in 1:5){
  var.list <- imp.var.list[[i]]; n.var <- length(var.list)
  p <- ''
  for(j in 1:(n.var-1)){
    p <- paste(p, 'ns(', var.list[j], ',4)',' + ', sep = '')
  }
  p <- paste(p, 'ns(', var.list[n.var], ',4)',sep = '')
  formula.list[[i]] <- as.formula(paste('ViolentCrimesPerPop ~', p))
}

train.gamls.error <- val.gamls.error <- rep(0, 5)
ftrain.gamls.error <- fval.gamls.error <- rep(0, 10)
for(i in 1:5){
  f <- formula.list[[i]]
  for(j in 1:10){
    # train-valid set split
    val.ind <- cv.index[[j]]
    crime.val <- crime.train[val.ind,]
    crime.cv.train <- crime.train[-val.ind,]

    # fitting lm
```

```
    crime.gam.lasso <- gam(f, data = crime.cv.train, family = poisson('sqrt'))
    ftrain.gamls.error[j] <- mean((crime.cv.train$ViolentCrimesPerPop - fitted(crime.gam.lasso))^2)
    fval.gamls.error[j] <- mean((crime.val$ViolentCrimesPerPop -
                                   predict(crime.gam.lasso, crime.val, type = 'response'))^2)
  }
  train.gamls.error[i] <- mean(ftrain.gamls.error)
  val.gamls.error[i] <- mean(fval.gamls.error)
}
train.gamls.error
sqrt(val.gamls.error)

f <- formula.list[[4]]
crime.gam.lasso <- gam(f, data = crime.train, family = poisson('sqrt'))

windows(10, 10)
par(mfrow = c(2, 4))
plot(crime.gam.lasso, se = T, col = 3)

########################
## model selection 2 ##
########################
## s.s df cv
s.df <- 2:10
train.sls.error <- val.sls.error <- rep(0, 9)
ftrain.sls.error <- fval.sls.error <- rep(0, 10)
for(i in 1:9){
  d <- s.df[i]
  for(j in 1:10){
    val.ind <- cv.index[[j]]
    crime.val <- crime.train[val.ind,]
    crime.cv.train <- crime.train[-val.ind,]

    # fitting lm

    crime.gam.lasso <- gam(ViolentCrimesPerPop ~ s(racePctWhite, d) + s(FemalePctDiv, d) +
                             s(TotalPctDiv, d) + s(PctKids2Par, d) + s(PctKidsBornNeverMar, d) +
                             s(PctPersDenseHous, d), data = crime.cv.train, family = poisson('sqrt'))
    ftrain.sls.error[j] <- mean((crime.cv.train$ViolentCrimesPerPop - fitted(crime.gam.lasso))^2)
    fval.sls.error[j] <- mean((crime.val$ViolentCrimesPerPop -
                                 predict(crime.gam.lasso, crime.val, type = 'response'))^2)
    predict(crime.gam.lasso, crime.val)
  }
  train.sls.error[i] <- mean(ftrain.sls.error)
  val.sls.error[i] <- mean(fval.sls.error)
}
train.sls.error
val.sls.error # df = 9일때 가장 좋음 132143.2
sqrt(min(val.sls.error))


## n.s eff.df cv
```

```
n.df <- 2:10
train.nls.error <- val.nls.error <- rep(0, 9)
ftrain.nls.error <- fval.nls.error <- rep(0, 10)
for(i in 1:9){
  d <- n.df[i]
  for(j in 1:10){
    val.ind <- cv.index[[j]]
    crime.val <- crime.train[val.ind,]
    crime.cv.train <- crime.train[-val.ind,]

    # fitting lm
    crime.gam.lasso <- gam(ViolentCrimesPerPop ~ ns(racePctWhite, d) + ns(MalePctDivorce, d) +
                           ns(FemalePctDiv, d) + ns(TotalPctDiv, d) + ns(PctKids2Par, d) +
                           ns(PctKidsBornNeverMar, d) + ns(PctPersDenseHous, d) +
                           ns(HousVacant, d), data = crime.cv.train, family = poisson('sqrt'))
    ftrain.nls.error[j] <- mean((crime.cv.train$ViolentCrimesPerPop - fitted(crime.gam.lasso))^2)
    fval.nls.error[j] <- mean((crime.val$ViolentCrimesPerPop -
                               predict(crime.gam.lasso, crime.val, type = 'response'))^2)
  }
  train.nls.error[i] <- mean(ftrain.nls.error)
  val.nls.error[i] <- mean(fval.nls.error)
}
train.nls.error
val.nls.error # df = 6일때 가장 좋음 128429.7
sqrt(min(val.nls.error))

windows(10, 10)
plot(2:10, sqrt(val.sls.error), type = 'l', lwd = 2, col = 3,
     ylab = '10-fold CV RMSE', xlab = 'df', ylim = c(356, 372))
lines(2:10, sqrt(val.nls.error), type = 'l', lwd = 2, col = 2)
legend('topright', c('smoothing splines', 'natural cubic splines'), lwd = c(2, 2), col = 3:2)


crime.gams.lasso <- gam(ViolentCrimesPerPop ~ s(racePctWhite, 9) + s(FemalePctDiv, 9) +
                        s(TotalPctDiv, 9) + s(PctKids2Par, 9) + s(PctKidsBornNeverMar, 9) +
                        s(PctPersDenseHous, 9), data = crime.train, family = poisson('sqrt'))

crime.gamn.lasso <- gam(ViolentCrimesPerPop ~ ns(racePctWhite, 6) + ns(MalePctDivorce, 6) +
                        ns(FemalePctDiv, 6) + ns(TotalPctDiv, 6) + ns(PctKids2Par, 6) +
                        ns(PctKidsBornNeverMar, 6) + ns(PctPersDenseHous, 6) +
                        ns(HousVacant, 6), data = crime.train, family = poisson('sqrt'))

windows(10, 10)
par(mfrow = c(2, 3))
plot(crime.gams.lasso, se = T, col = 3)

windows(10, 10)
par(mfrow = c(2, 4))
plot(crime.gamn.lasso, se = T, col = 4)

#######################
```

```
## model selection 3 ##
#######################
key.var <- imp.var.list[[1]]
div.var <- c("FemalePctDiv", "MalePctDivorce", "TotalPctDiv")
hs.var <- c("HousVacant", "PctPersDenseHous")
var.list <- list()

for(i in 1:3){
  div.comb <- combn(3, i); div.n <- ncol(div.comb)
  for(j in 1:div.n){
    div <- div.var[div.comb[,j]]
    for(k in 1:2){
      hs.comb <- combn(2, k); hs.n <- ncol(hs.comb)
      for(l in 1:hs.n){
        hs <- hs.var[hs.comb[,l]]
        var <- c(key.var, div, hs)
        if(length(var.list) != 0){n.list <- length(var.list); var.list[[n.list+1]] <- var}
        if(length(var.list) == 0){var.list <- list(var)}
      }
    }
  }
}

formula.list <- list()
for(i in 1:21){
  var.list.1 <- var.list[[i]]; n.var <- length(var.list.1)
  p <- ''
  for(j in 1:(n.var-1)){
    p <- paste(p, 'ns(', var.list.1[j], ',6)',' + ', sep = '')
  }
  p <- paste(p, 'ns(', var.list.1[n.var], ',6)',sep = '')
  formula.list[[i]] <- as.formula(paste('ViolentCrimesPerPop ~', p))
}


train.cv.error <- val.cv.error <- rep(0, 21)
for(i in 1:21){
  f <- formula.list[[i]]
  ftrain.cv.error <- fval.cv.error <- rep(0, 10)
  for(j in 1:10){
    val.ind <- cv.index[[j]]
    crime.val <- crime.train[val.ind,]
    crime.cv.train <- crime.train[-val.ind,]

    # fitting gam
    crime.gam.lasso <- gam(f, data = crime.cv.train, family = poisson('sqrt'))
    ftrain.cv.error[j] <- mean((crime.cv.train$ViolentCrimesPerPop - fitted(crime.gam.lasso))^2)
    fval.cv.error[j] <- mean((crime.val$ViolentCrimesPerPop -
                                predict(crime.gam.lasso, crime.val, type = 'response'))^2)
  }
  train.cv.error[i] <- mean(ftrain.cv.error)
```

```
  val.cv.error[i] <- mean(fval.cv.error)
}
train.cv.error
val.cv.error
# full natural spline model보다 더 좋은 모델! 변수 개수도 8개 -> 6개로 줄었음
formula.list[[which.min(val.cv.error)]]
# ViolentCrimesPerPop ~ ns(racePctWhite, 6) + ns(PctKids2Par, 6) + ns(PctKidsBornNeverMar, 6)
# + ns(MalePctDivorce, 6) + ns(HousVacant, 6) + ns(PctPersDenseHous, 6)

f <- formula.list[[which.min(val.cv.error)]]
crime.gam.lasso <- gam(f, data = crime.train, family = poisson('sqrt'))
windows(10, 10)
par(mfrow = c(2, 3))
plot(crime.gam.lasso, se = T, col = 3)

########################
## model selection 4 ##
########################
df.mat <- matrix(0, 3^6, 6)
i <- 1
for(a in 2:4){for(b in 4:6){for(c in 2:4){for(d in 4:6){for(e in 4:6){for(f in 2:4){
  df <- c(a, b, c, d, e, f); df.mat[i,] <- df; i <- i+1
}}}}}}

train.cv.error <- val.cv.error <- rep(0, 3^6)
for(i in 1:(3^6)){
  df <- df.mat[i,]
  ftrain.cv.error <- fval.cv.error <- rep(0, 10)
  for(j in 1:10){
    val.ind <- cv.index[[j]]
    crime.val <- crime.train[val.ind,]
    crime.cv.train <- crime.train[-val.ind,]

    # fitting gam
    crime.gam.lasso <- gam(ViolentCrimesPerPop ~ ns(racePctWhite, df[1]) + ns(PctKids2Par, df[2]) +
                             ns(PctKidsBornNeverMar, df[3]) + ns(MalePctDivorce, df[4]) +
                             ns(HousVacant, df[5]) + ns(PctPersDenseHous, df[6]),
                           data = crime.cv.train, family = poisson('sqrt'))
    ftrain.cv.error[j] <- mean((crime.cv.train$ViolentCrimesPerPop - fitted(crime.gam.lasso))^2)
    fval.cv.error[j] <- mean((crime.val$ViolentCrimesPerPop -
                                predict(crime.gam.lasso, crime.val, type = 'response'))^2)
  }
  train.cv.error[i] <- mean(ftrain.cv.error)
  val.cv.error[i] <- mean(fval.cv.error)
}
train.cv.error
val.cv.error
min(val.cv.error) # 127299.8
df.mat[which.min(val.cv.error),] # 3 5 2 6 6 2
```

```
crime.gam.lasso <- gam(ViolentCrimesPerPop ~ ns(racePctWhite, 3) + ns(PctKids2Par, 5) +
                         ns(PctKidsBornNeverMar, 2) + ns(MalePctDivorce, 6) +
                         ns(HousVacant, 6) + ns(PctPersDenseHous, 2),
                       data = crime.train, family = poisson('sqrt'))
windows(10, 10)
par(mfrow = c(2, 3))
plot(crime.gam.lasso, se = T, col = 2)


for(j in 1:10){
  val.ind <- cv.index[[j]]
  crime.val <- crime.train[val.ind,]
  crime.cv.train <- crime.train[-val.ind,]

  # fitting gam
  crime.gam.lasso <- gam(ViolentCrimesPerPop ~ ns(racePctWhite, 3) + ns(PctKids2Par, 5) +
                           PctKidsBornNeverMar + ns(MalePctDivorce, 6) +
                           ns(HousVacant, 6) + ns(PctPersDenseHous, 2),
                         data = crime.cv.train, family = poisson('sqrt'))
  ftrain.cv.error[j] <- mean((crime.cv.train$ViolentCrimesPerPop - fitted(crime.gam.lasso))^2)
  fval.cv.error[j] <- mean((crime.val$ViolentCrimesPerPop -
                              predict(crime.gam.lasso, crime.val, type = 'response'))^2)
}
mean(ftrain.cv.error)
mean(fval.cv.error) # 126928.3
sqrt(126928.3)

# 최종 모델
crime.gam.lasso <- gam(ViolentCrimesPerPop ~ ns(racePctWhite, 3) + ns(PctKids2Par, 5) +
                         PctKidsBornNeverMar + ns(MalePctDivorce, 6) +
                         ns(HousVacant, 6) + ns(PctPersDenseHous, 2),
                       data = crime.train, family = poisson('sqrt'))
windows(10, 10)
par(mfrow = c(2, 3))
plot(crime.gam.lasso, se = T, col = 2)


######################
#### 3. DR by SIR ####
######################
# 해당 데이터는 총 102개의 설명변수가 존재하므로 차원축소가 필요함
crime.train <- raw.train.std
set.seed(123); add <- sample(1:102, 2)
x.mat.std <- crime.train[,1:102]
sir.vc.std <- dr(crime.train$ViolentCrimesPerPop ~ as.matrix(x.mat.std), method = 'sir')

sir.vc.1 <- sir.vc.std$evectors[,1:10]
x.mat.sir <- as.matrix(x.mat.std[,1:100]) %*% as.matrix(sir.vc.1)
dim(x.mat.sir) # 10개의 변수로 축약
crime.sir.train <- as.data.frame(cbind(x.mat.sir, crime.train[,103]))
crime.train <- crime.sir.train
```

```
colnames(crime.train)[11] <- 'ViolentCrimesPerPop'
str(crime.train)


##### GLM 변수 개수별로 비교해보려고 함 ####
formula.list <- list()
for(i in 1:10){
  p <- ''
  for(j in 1:(i-1)){
    if(i == 1) break
    p <- paste(p, 'Dir', j, ' + ', sep = '')
  }
  p <- paste(p, 'Dir', i, sep = '')
  formula.list[[i]] <- as.formula(paste('ViolentCrimesPerPop ~', p))
}


train.MSE.lm <- val.MSE.lm <- rep(0, 10)
ftrain.MSE.lm <- fval.MSE.lm <- rep(0, 5)
for(i in 1:10){
  for(j in 1:10){
    # train-valid set split
    val.ind <- cv.index[[j]]
    crime.val <- crime.train[val.ind,]
    crime.cv.train <- crime.train[-val.ind,]

    # fitting gam
    crime.lm.sir <- glm(formula.list[[i]], data = crime.cv.train, family = poisson('sqrt'))
    ftrain.MSE.lm[j] <- mean((crime.cv.train$ViolentCrimesPerPop - fitted(crime.lm.sir))^2)
    fval.MSE.lm[j] <- mean((crime.val$ViolentCrimesPerPop -
                                predict(crime.lm.sir, crime.val, type = 'response'))^2)
  }
  train.MSE.lm[i] <- mean(ftrain.MSE.lm)
  val.MSE.lm[i] <- mean(fval.MSE.lm)
}

train.MSE.lm
# 145000.1 144729.5 141337.5 141363.6 141291.8 141054.8 126470.0 126204.3 123209.8 122817.9
val.MSE.lm
# 145952.5 145894.1 143467.9 156873.6 157832.5 158465.4 129042.9 129067.5 126058.3 126173.7


##### GAM 변수 개수별로 비교해보려고 함 ####
formula.list <- list()
for(i in 1:10){
  p <- ''
  for(j in 1:(i-1)){
    if(i == 1) break
    p <- paste(p, 's(Dir', j, ') + ', sep = '')
  }
  p <- paste(p, 's(Dir', i, ')', sep = '')
```

```
  formula.list[[i]] <- as.formula(paste('ViolentCrimesPerPop ~', p))
}


train.MSE.gam <- val.MSE.gam <- rep(0, 10)
ftrain.MSE.gam <- fval.MSE.gam <- rep(0, 10)
for(i in 1:10){
  for(j in 1:10){
    # train-valid set split
    val.ind <- cv.index[[j]]
    crime.val <- crime.train[val.ind,]
    crime.cv.train <- crime.train[-val.ind,]

    # fitting gam
    crime.gam.sir <- gam(formula.list[[i]], data = crime.cv.train,
                         family = poisson('sqrt'))
    ftrain.MSE.gam[j] <- mean((crime.cv.train$ViolentCrimesPerPop - fitted(crime.gam.sir))^2)
    fval.MSE.gam[j] <- mean((crime.val$ViolentCrimesPerPop -
                               predict(crime.gam.sir, crime.val, type = 'response'))^2)
  }
  train.MSE.gam[i] <- mean(ftrain.MSE.gam)
  val.MSE.gam[i] <- mean(fval.MSE.gam)
}

train.MSE.gam

val.MSE.gam



windows(10, 10)
plot(1:10, sqrt(val.MSE.lm), col = 3, type = 'l', lwd = 2,
     ylab = '10-fold CV RMSE', xlab = 'number of component', ylim = c(350, 490))
lines(1:10, sqrt(val.MSE.gam), col = 2, lwd = 2)
legend('topright', c('GLM', 'GAM'), col = 3:2, lwd = c(2, 2))

sqrt(min(val.MSE.lm))
sqrt(min(val.MSE.gam))

# GAM 모델 10 선택

#### smoothing spline ####
ef.df <- 2:8

# 모델 10
train.MSE.s <- val.MSE.s <- rep(0, length(ef.df))
for(i in 1:length(ef.df)){
  df.cv <- ef.df[i]
  ftrain.MSE.s <- fval.MSE.s <- rep(0, 10)
  for(j in 1:10){
    # train-valid set split
```

```
    val.ind <- cv.index[[j]]
    crime.val <- crime.train[val.ind,]
    crime.cv.train <- crime.train[-val.ind,]

    # fitting GAM
    crime.gam.s <- gam(ViolentCrimesPerPop ~ s(Dir1, df = df.cv) + s(Dir2, df = df.cv) +
                         s(Dir3, df = df.cv) + s(Dir4, df = df.cv) + s(Dir5, df = df.cv) +
                         s(Dir6, df = df.cv) + s(Dir7, df = df.cv) + s(Dir8, df = df.cv) +
                         s(Dir9, df = df.cv) + s(Dir10, df = df.cv),
                       data = crime.cv.train, family = poisson('sqrt'))
    ftrain.MSE.s[j] <- mean((crime.cv.train$ViolentCrimesPerPop - fitted(crime.gam.s))^2)
    fval.MSE.s[j] <- mean((crime.val$ViolentCrimesPerPop -
                             predict(crime.gam.s, crime.val, type = 'response'))^2)
  }
  train.MSE.s[i] <- mean(ftrain.MSE.s)
  val.MSE.s[i] <- mean(fval.MSE.s)
}
train.MSE.s
val.MSE.s; min(val.MSE.s) # 123771.1, df = 4


#### natural spline ####
n.df <- 2:8

# 모델 10
train.MSE.n <- val.MSE.n <- rep(0, length(n.df))
for(i in 1:length(n.df)){
  df.cv <- n.df[i]
  ftrain.MSE.n <- fval.MSE.n <- rep(0, 10)
  for(j in 1:10){
    # train-valid set split
    val.ind <- cv.index[[j]]
    crime.val <- crime.train[val.ind,]
    crime.cv.train <- crime.train[-val.ind,]

    # fitting GAM
    crime.gam.n <- gam(ViolentCrimesPerPop ~ ns(Dir1, df = df.cv) + ns(Dir2, df = df.cv) +
                         ns(Dir3, df = df.cv) + ns(Dir4, df = df.cv) + ns(Dir5, df = df.cv) +
                         ns(Dir6, df = df.cv) + ns(Dir7, df = df.cv) + ns(Dir8, df = df.cv) +
                         ns(Dir9, df = df.cv) + ns(Dir10, df = df.cv),
                       data = crime.cv.train, family = poisson('sqrt'))
    ftrain.MSE.n[j] <- mean((crime.cv.train$ViolentCrimesPerPop - fitted(crime.gam.n))^2)
    fval.MSE.n[j] <- mean((crime.val$ViolentCrimesPerPop -
                             predict(crime.gam.n, crime.val, type = 'response'))^2)
  }
  train.MSE.n[i] <- mean(ftrain.MSE.n)
  val.MSE.n[i] <- mean(fval.MSE.n)
}
train.MSE.n
val.MSE.n; min(val.MSE.n) # 127170.5, df = 4
```

```
windows(10, 10)
plot(2:8, sqrt(val.MSE.s), type = 'l', col = 3, lwd = 2,
      ylim = c(349, 450), ylab = '10-fold CV RMSE', xlab = 'df')
lines(2:8, sqrt(val.MSE.n), type = 'l', col = 2, lwd = 2)
legend('topleft', c('smoothing splines', 'natural cubic splines'), lwd = c(2, 2), col = 3:2)

crime.gam.s <- gam(ViolentCrimesPerPop ~ s(Dir1, 4) + s(Dir2, 4) +
                       s(Dir3, 4) + s(Dir4, 4) + s(Dir5, 4) + s(Dir6, 4) + s(Dir7, 4) + s(Dir8, 4) +
                       s(Dir9, 4) + s(Dir10, 4), data = crime.train, family = poisson('sqrt'))
windows(10, 10)
par(mfrow = c(2, 5))
plot(crime.gam.s, se = T, col = 3)


# 자유도 조절
df.mat <- matrix(0, 8, 10)
i <- 1
for(a in 2:3){for(b in 3:4){for(c in 4:5){
  df.mat[i, c(1, 3, 4, 9)] <- rep(a, 4)
  df.mat[i, 6:7] <- rep(b, 2)
  df.mat[i, c(2, 5, 8, 10)] <- rep(c, 4)
  i <- i+1
}}}

train.MSE.s <- val.MSE.s <- rep(0, nrow(df.mat))
for(i in 1:nrow(df.mat)){
  d <- df.mat[i,]
  ftrain.MSE.s <- fval.MSE.s <- rep(0, 10)
  for(j in 1:10){
    # train-valid set split
    val.ind <- cv.index[[j]]
    crime.val <- crime.train[val.ind,]
    crime.cv.train <- crime.train[-val.ind,]

    # fitting GAM
    crime.gam.s <- gam(ViolentCrimesPerPop ~ s(Dir1, df = d[1]) + s(Dir2, df = d[2]) +
                         s(Dir3, df = d[3]) + s(Dir4, df = d[4]) + s(Dir5, df = d[5]) +
                         s(Dir6, df = d[6]) + s(Dir7, df = d[7]) + s(Dir8, df = d[8]) +
                         s(Dir9, df = d[9]) + s(Dir10, df = d[10]),
                       data = crime.cv.train, family = poisson('sqrt'))
    ftrain.MSE.s[j] <- mean((crime.cv.train$ViolentCrimesPerPop - fitted(crime.gam.s))^2)
    fval.MSE.s[j] <- mean((crime.val$ViolentCrimesPerPop -
                             predict(crime.gam.s, crime.val, type = 'response'))^2)
  }
  train.MSE.s[i] <- mean(ftrain.MSE.s)
  val.MSE.s[i] <- mean(fval.MSE.s)
}
train.MSE.s
val.MSE.s; min(val.MSE.s) # 124996.9
df.mat[which.min(val.MSE.s),] # 3 4 3 3 4 4 4 4 3 4
```

```
# 성능 향상이 전혀 되지 않음...
# 자유도를 임의로 조정해보자

for(j in 1:10){
  # train-valid set split
  val.ind <- cv.index[[j]]
  crime.val <- crime.train[val.ind,]
  crime.cv.train <- crime.train[-val.ind,]

  # fitting GAM
  crime.gam.s <- gam(ViolentCrimesPerPop ~ s(Dir1, 4) + s(Dir2, 2) +
                       s(Dir3, 4) + s(Dir4, 3) + s(Dir5, 2) + s(Dir6, 4) + s(Dir7, 4) +
                       s(Dir8, 2) + s(Dir9, 2) + s(Dir10, 4),
                     data = crime.cv.train, family = poisson('sqrt'))
  ftrain.MSE.s[j] <- mean((crime.cv.train$ViolentCrimesPerPop - fitted(crime.gam.s))^2)
  fval.MSE.s[j] <- mean((crime.val$ViolentCrimesPerPop -
                           predict(crime.gam.s, crime.val, type = 'response'))^2)
}
mean(fval.MSE.s)
sqrt(121799.5)
# 121799.5
# 가장 좋은 성능! 최종 모델로 선택


# 최종 모델
crime.gam.s <- gam(ViolentCrimesPerPop ~ s(Dir1, 4) + s(Dir2, 2) +
                     s(Dir3, 4) + s(Dir4, 3) + s(Dir5, 2) + s(Dir6, 4) + s(Dir7, 4) +
                     s(Dir8, 2) + s(Dir9, 2) + s(Dir10, 4),
                   data = crime.train, family = poisson('sqrt'))
windows(10, 10)
par(mfrow = c(2, 5))
plot(crime.gam.s, se = T, col = 3)


##### 2차 SIR #####
crime.train <- raw.train.std
x.mat.std <- cbind(crime.train[,1:102], crime.train[,add])
sir.vc.std <- dr(crime.train$ViolentCrimesPerPop ~ as.matrix(x.mat.std), method = 'sir')
round(cumsum(sir.vc.std$evalues/sum(sir.vc.std$evalues)), 3)
# 5개의 변수를 선택하면 전체 분산의 약 23%만 설명 가능함
# 30개의 변수를 사용해야 전체 분산의 약 70%를 설명 가능
# -> 차원을 좀 더 줄이면서 전체 분산을 더 많이 설명할 수 없을까?

# 코드 설명 : 첫 30개의 eigenvectors(eigenvale가 큰 순서대로)에서 loading 절댓값 상위 30개를 뽑았다.
# 즉, 각 PC에 기여도가 높은 상위 30개 변수를 뽑은 것
# -> 30개의 vectors에서 뽑은 30개의 loading 상위 변수 중 unique한 변수만 찾는다
# -> 각 변수가 30개의 eigenvector 중 몇개의 벡터에서 뽑혔는지 비율을 확인
# 즉, 전체 분산의 70%를 설명하는 30개의 PC에서 기여도가 높은 변수들이 무엇인지 확인하는 작업!
impV <- apply(sir.vc.std$evectors[,1:30], 2,
              function(x) colnames(x.mat.std[head(order(abs(x), decreasing = T), 30)]))
```

```
impV.uni <- unique(as.vector(impV)); length(impV.uni) # unique한 변수 : 75개
imp.p <- rep(0, length(impV.uni)); names(imp.p) <- impV.uni
for(i in 1:length(imp.p)){
  var <- impV.uni[i]
  where.v <- which(as.vector(impV) == var)
  imp.p[i] <- length(where.v)/30
}
sort(round(imp.p, 4), decreasing = T)

impV.1 <- names(imp.p)[which(imp.p >= 0.5)]; length(impV.1)
# 각 PC의 기여도 상위 20개 변수 그룹 중 절반 이상의 그룹에 속한 변수 : 27개
# 해당 27개 변수만을 사용해 SIR 방법을 다시 사용해본다.

x.mat.1.std <- crime.train[,impV.1]
sir.vc.1.std <- dr(crime.train$ViolentCrimesPerPop ~ as.matrix(x.mat.1.std), method = 'sir')
round(cumsum(sir.vc.1.std$evalues/sum(sir.vc.1.std$evalues)), 4)
# SIR 결과, 첫 10개의 PC를 사용하면 전체 분산의 약 90%가 설명된다.
# 첫 5개 PC -> 77%

sir.vc.2 <- sir.vc.1.std$evectors[,1:10]
x.mat.sir.1 <- as.matrix(x.mat.1.std) %*% as.matrix(sir.vc.2)
dim(x.mat.sir.1) # 10개의 변수로 축약
crime.sir.train <- as.data.frame(cbind(x.mat.sir.1, crime.train[,103]))
crime.train <- crime.sir.train
colnames(crime.train)[11] <- 'ViolentCrimesPerPop'
str(crime.train)

ev <- round(apply(sir.vc.2, 2, function(x) ifelse(abs(x) < 0.2, 0, x)), 2)
PC.list <- list()
for(i in 1:10){
  pc <- ev[which(ev[,i] != 0),i]; names(pc) <- impV.1[which(ev[,i] != 0)]
  PC.list[[i]] <- pc
}
PC.list

##### GLM 변수 개수별로 비교해보려고 함 ####
formula.list <- list()
for(i in 1:10){
  p <- ''
  for(j in 1:(i-1)){
    if(i == 1) break
    p <- paste(p, 'Dir', j, ' + ', sep = '')
  }
  p <- paste(p, 'Dir', i, sep = '')
  formula.list[[i]] <- as.formula(paste('ViolentCrimesPerPop ~', p))
}

train.MSE.lm.1 <- val.MSE.lm.1 <- rep(0, 10)
ftrain.MSE.lm.1 <- fval.MSE.lm.1 <- rep(0, 10)
for(i in 1:10){
  for(j in 1:10){
```

```
    # train-valid set split
    val.ind <- cv.index[[j]]
    crime.val <- crime.train[val.ind,]
    crime.cv.train <- crime.train[-val.ind,]

    # fitting lm
    crime.lm.sir.1 <- glm(formula.list[[i]], data = crime.cv.train,
                          family = poisson('sqrt'))
    ftrain.MSE.lm.1[j] <- mean((crime.cv.train$ViolentCrimesPerPop - fitted(crime.lm.sir.1))^2)
    fval.MSE.lm.1[j] <- mean((crime.val$ViolentCrimesPerPop -
                              predict(crime.lm.sir.1, crime.val, type = 'response'))^2)
  }
  train.MSE.lm.1[i] <- mean(ftrain.MSE.lm.1)
  val.MSE.lm.1[i] <- mean(fval.MSE.lm.1)
}
train.MSE.lm.1

val.MSE.lm.1

which.min(val.MSE.lm.1)
sqrt(134533.2)

##### GAM 변수 개수별로 비교해보려고 함 ####
formula.list <- list()
for(i in 1:10){
  p <- ''
  for(j in 1:(i-1)){
    if(i == 1) break
    p <- paste(p, 's(Dir', j, ') + ', sep = '')
  }
  p <- paste(p, 's(Dir', i, ')', sep = '')
  formula.list[[i]] <- as.formula(paste('ViolentCrimesPerPop ~', p))
}


train.MSE.gam.1 <- val.MSE.gam.1 <- rep(0, 10)
ftrain.MSE.gam.1 <- fval.MSE.gam.1 <- rep(0, 10)
for(i in 1:10){
  for(j in 1:10){
    # train-valid set split
    val.ind <- cv.index[[j]]
    crime.val <- crime.train[val.ind,]
    crime.cv.train <- crime.train[-val.ind,]

    # fitting gam
    crime.gam.sir.1 <- gam(formula.list[[i]], data = crime.cv.train,
                           family = poisson('sqrt'))
    ftrain.MSE.gam.1[j] <- mean((crime.cv.train$ViolentCrimesPerPop - fitted(crime.gam.sir.1))^2)
    fval.MSE.gam.1[j] <- mean((crime.val$ViolentCrimesPerPop -
                               predict(crime.gam.sir.1, crime.val, type = 'response'))^2)
  }
```

```
  train.MSE.gam.1[i] <- mean(ftrain.MSE.gam.1)
  val.MSE.gam.1[i] <- mean(fval.MSE.gam.1)
}

train.MSE.gam.1
# 132431.7 129909.8 128375.6 127502.4 127189.6 126664.1 125672.6 124778.7 124133.2 123716.4
val.MSE.gam.1
# 134660.6 134650.9 134782.1 134738.9 134953.6 135543.9 134547.7 134312.0 134943.5 134665.3
which.min(val.MSE.gam.1)
sqrt(134312.0)

# GLM vs GAM 비교
windows(10, 10)
plot(1:10, sqrt(val.MSE.lm.1), type = 'l', lwd = 2, col = 3,
     ylab = '10-fold CV RMSE', xlab = 'number of component', ylim = c(365, 372))
lines(1:10, sqrt(val.MSE.gam.1), lwd = 2, col = 2)
legend('topleft', c('GLM', 'GAM'), lwd = c(2, 2), col = 3:2)

# GLM 1차 SIR vs 2차 SIR 비교
windows(20, 10)
par(mfrow = c(1, 2))
plot(1:10, sqrt(val.MSE.lm), type = 'l', lwd = 3, col = 3, ylim = c(350, 490),
     ylab = '10-fold CV RMSE', xlab = 'number of component', main = 'GLM', xaxt="n",yaxt="n")
axis(side=1,at=1:10)
axis(side=2,at=1:10)
lines(1:10, sqrt(val.MSE.lm.1), lwd = 3, col = 2)
abline(v = 7, lwd = 2, col = 'gray', lty = 2)
legend('topright', c('1차 SIR', '2차 SIR'), lwd = c(3, 3), col = 3:2)
# GAM 1차 SIR vs 2차 SIR 비교
plot(1:10, sqrt(val.MSE.gam), type = 'l', lwd = 3, col = 3, ylim = c(350, 490),
     ylab = '10-fold CV RMSE', xlab = 'number of component', main = 'GAM', xaxt="n",yaxt="n")
axis(side=1,at=1:10)
axis(side=2,at=1:10)
lines(1:10, sqrt(val.MSE.gam.1), lwd = 3, col = 2)
abline(v = 7, lwd = 2, col = 'gray', lty = 2)
legend('topright', c('1차 SIR', '2차 SIR'), lwd = c(3, 3), col = 3:2)


# GAM모델 8 선택

#### smoothing spline ####
ef.df <- 2:10

# 모델 8
train.MSE.s <- val.MSE.s <- rep(0, length(ef.df))
for(i in 1:length(ef.df)){
  df.cv <- ef.df[i]
  ftrain.MSE.s <- fval.MSE.s <- rep(0, 10)
  for(j in 1:10){
    # train-valid set split
    val.ind <- cv.index[[j]]
```

```
    crime.val <- crime.train[val.ind,]
    crime.cv.train <- crime.train[-val.ind,]

    # fitting GAM
    crime.gam.s <- gam(ViolentCrimesPerPop ~ s(Dir1, df = df.cv) + s(Dir2, df = df.cv) +
                         s(Dir3, df = df.cv) + s(Dir4, df = df.cv) + s(Dir5, df = df.cv) +
                         s(Dir6, df = df.cv) + s(Dir7, df = df.cv) + s(Dir8, df = df.cv),
                       data = crime.cv.train, family = poisson('sqrt'))
    ftrain.MSE.s[j] <- mean((crime.cv.train$ViolentCrimesPerPop - fitted(crime.gam.s))^2)
    fval.MSE.s[j] <- mean((crime.val$ViolentCrimesPerPop -
                             predict(crime.gam.s, crime.val, type = 'response'))^2)
  }
  train.MSE.s[i] <- mean(ftrain.MSE.s)
  val.MSE.s[i] <- mean(fval.MSE.s)
}
train.MSE.s
val.MSE.s; min(val.MSE.s) # 133884, df = 6
sqrt(133884)



windows(10, 10)
plot(ef.df, val.MSE.s, type = 'l', lwd = 2, col = 'blue', ylim = c(124000, 160000),
     xlab = 'effective df', ylab = '', main = '10-fold CV MSE of smoothing spline')
lines(ef.df, train.MSE.s, lwd = 2, col = 'red')
legend('topleft', c('training MSE','validation MSE'), lwd = c(2, 2), col = c('red', 'blue'))
which.min(val.MSE.s)+1 # effective df = 8일때가 가장 좋음
min(val.MSE.s)


#### natural spline ####
n.df <- 2:10

# 모델 8
train.MSE.n <- val.MSE.n <- rep(0, length(n.df))
for(i in 1:length(n.df)){
  df.cv <- n.df[i]
  ftrain.MSE.n <- fval.MSE.n <- rep(0, 10)
  for(j in 1:10){
    # train-valid set split
    val.ind <- cv.index[[j]]
    crime.val <- crime.train[val.ind,]
    crime.cv.train <- crime.train[-val.ind,]

    # fitting GAM
    crime.gam.n <- gam(ViolentCrimesPerPop ~ ns(Dir1, df = df.cv) + ns(Dir2, df = df.cv) +
                         ns(Dir3, df = df.cv) + ns(Dir4, df = df.cv) + ns(Dir5, df = df.cv) +
                         ns(Dir6, df = df.cv) + ns(Dir7, df = df.cv) + ns(Dir8, df = df.cv),
                       data = crime.cv.train, family = poisson('sqrt'))
    ftrain.MSE.n[j] <- mean((crime.cv.train$ViolentCrimesPerPop - fitted(crime.gam.n))^2)
    fval.MSE.n[j] <- mean((crime.val$ViolentCrimesPerPop -
```

```
                                predict(crime.gam.n, crime.val, type = 'response'))^2)
  }
  train.MSE.n[i] <- mean(ftrain.MSE.n)
  val.MSE.n[i] <- mean(fval.MSE.n)
}
train.MSE.n
val.MSE.n; min(val.MSE.n) # 136237.8, df = 3


windows(10, 10)
plot(2:10, sqrt(val.MSE.s), type = 'l', col = 3, lwd = 2,
     ylim = c(363, 377), ylab = '10-fold CV RMSE', xlab = 'df')
lines(2:10, sqrt(val.MSE.n), type = 'l', col = 2, lwd = 2)
legend('topleft', c('smoothing splines', 'natural cubic splines'), lwd = c(2, 2), col = 3:2)


########################
## model selection 2 ##
########################
crime.gam.s <- gam(ViolentCrimesPerPop ~ s(Dir1, df = 6) + s(Dir2, df = 6) +
                     s(Dir3, df = 6) + s(Dir4, df = 6) + s(Dir5, df = 6) +
                     s(Dir6, df = 6) + s(Dir7, df = 6) + s(Dir8, df = 6),
                   data = crime.train, family = poisson('sqrt'))

windows(10, 10)
par(mfrow = c(2, 4))
plot(crime.gam.s, se = T, col = 3)
# 해당 모델의 성능이 가장 좋게 나왔다. 여기서 더 발전시켜본다.
# 각 변수마다 비선형 함수 및 자유도를 달리 적용해본다.

df.mat <- matrix(0, ((2^5)*(3^3)), 8)
i <- 1
for(a in 2:4){for(b in 4:6){for(c in 4:6){for(d in 2:4){
  for(e in 4:6){for(f in 4:6){for(g in 4:6){for(h in 4:6){
    df <- c(a, b, c, d, e, f, g, h); df.mat[i,] <- df; i <- i+1
  }}}}}}}}

formula.list <- list()
for(i in 1:nrow(df.mat)){
  p <- ''
  for(j in 1:7){
    if(j %in% 5:6){p <- paste(p, 'ns(Dir', j, ', ', df.mat[i, j], ')',' + ', sep = '')}
    if(!(j %in% 5:6)){p <- paste(p, 's(Dir', j, ', ', df.mat[i, j], ')',' + ', sep = '')}
  }
  p <- paste(p, 'ns(Dir', j, ', ', df.mat[i, 8], ')',sep = '')
  formula.list[[i]] <- as.formula(paste('ViolentCrimesPerPop ~', p))
}

train.MSE.cv <- val.MSE.cv <- rep(0, nrow(df.mat))
for(i in 1:nrow(df.mat)){
  f <- formula.list[[i]]
```

```
  ftrain.MSE.cv <- fval.MSE.cv <- rep(0, 10)
  for(j in 1:10){
    # train-valid set split
    val.ind <- cv.index[[j]]
    crime.val <- crime.train[val.ind,]
    crime.cv.train <- crime.train[-val.ind,]

    # fitting GAM
    crime.gam.cv <- gam(f, data = crime.cv.train, family = poisson('sqrt'))
    ftrain.MSE.cv[j] <- mean((crime.cv.train$ViolentCrimesPerPop - fitted(crime.gam.cv))^2)
    fval.MSE.cv[j] <- mean((crime.val$ViolentCrimesPerPop -
                                predict(crime.gam.cv, crime.val, type = 'response'))^2)
  }
  train.MSE.cv[i] <- mean(ftrain.MSE.cv)
  val.MSE.cv[i] <- mean(fval.MSE.cv)
}
train.MSE.cv
val.MSE.cv; min(val.MSE.cv) # 성능 그다지 향상되지 않음. 임의로 자유도 조정해보기

ftrain.MSE.n <- fval.MSE.n <- rep(0, 10)
for(j in 1:10){
  # train-valid set split
  val.ind <- cv.index[[j]]
  crime.val <- crime.train[val.ind,]
  crime.cv.train <- crime.train[-val.ind,]

  # fitting GAM
  crime.gam.n <- gam(ViolentCrimesPerPop ~ s(Dir1, 4) + s(Dir2, 6) +
                       s(Dir3, 6) + s(Dir4, 3) + ns(Dir5, 4) +
                       ns(Dir6, 4) + s(Dir7, 8) + ns(Dir8, 5),
                     data = crime.cv.train, family = poisson('sqrt'))
  ftrain.MSE.n[j] <- mean((crime.cv.train$ViolentCrimesPerPop - fitted(crime.gam.n))^2)
  fval.MSE.n[j] <- mean((crime.val$ViolentCrimesPerPop -
                           predict(crime.gam.n, crime.val, type = 'response'))^2)
}
mean(ftrain.MSE.n)
mean(fval.MSE.n)
sqrt(131568.5)
# 131568.5 성능 가장 좋음
# ViolentCrimesPerPop ~ s(Dir1, 4) + s(Dir2, 6) + s(Dir3, 6) + s(Dir4, 3) + ns(Dir5, 4) +
#   ns(Dir6, 4) + s(Dir7, 8) + ns(Dir8, 5)


# 최종 모델
crime.gam.s <- gam(ViolentCrimesPerPop ~ s(Dir1, 4) + s(Dir2, 6) + s(Dir3, 6) + s(Dir4, 3) +
                     ns(Dir5, 4) + ns(Dir6, 4) + s(Dir7, 8) + ns(Dir8, 5),
                   data = crime.train, family = poisson('sqrt'))

windows(10, 10)
par(mfrow = c(2, 4))
plot(crime.gam.s, se = T, col = 3)
```

```
#### test!!! ####
#### 1차 SIR ####
crime.test <- raw.test.std
crime.train <- raw.train.std

x.mat.sir <- as.matrix(crime.train[,1:100]) %*% as.matrix(sir.vc.1)
dim(x.mat.sir) # 10개의 변수로 축약
crime.sir.train <- as.data.frame(cbind(x.mat.sir, crime.train[,103]))
crime.train <- crime.sir.train
colnames(crime.train)[11] <- 'ViolentCrimesPerPop'
str(crime.train)
crime.train.SIR.1 <- crime.train


x.mat.sir <- as.matrix(crime.test[,1:100]) %*% as.matrix(sir.vc.1)
dim(x.mat.sir) # 10개의 변수로 축약
crime.sir.test <- as.data.frame(cbind(x.mat.sir, crime.test[,103]))
crime.test <- crime.sir.test
colnames(crime.test)[11] <- 'ViolentCrimesPerPop'
str(crime.test)
crime.test.SIR.1 <- crime.test

crime.gam.SIR.1 <- gam(ViolentCrimesPerPop ~ s(Dir1, 4) + s(Dir2, 2) +
                         s(Dir3, 4) + s(Dir4, 3) + s(Dir5, 2) + s(Dir6, 4) + s(Dir7, 4) +
                         s(Dir8, 2) + s(Dir9, 2) + s(Dir10, 4),
                         data = crime.test.SIR.1, family = poisson('sqrt'))

mean((crime.test.SIR.1$ViolentCrimesPerPop -
        predict(crime.gam.SIR.1, crime.test.SIR.1, type = 'response'))^2) # 106412
sqrt(106412) # 326.2085

windows(10, 10)
par(mfrow = c(2, 5))
plot(crime.gam.SIR.1, se = T, col = 3)

#### 2차 SIR ####
crime.test <- raw.test.std
crime.train <- raw.train.std


x.mat.sir <- as.matrix(crime.train[,impV.1]) %*% as.matrix(sir.vc.2)
dim(x.mat.sir) # 10개의 변수로 축약
crime.sir.train <- as.data.frame(cbind(x.mat.sir, crime.train[,103]))
crime.train <- crime.sir.train
colnames(crime.train)[11] <- 'ViolentCrimesPerPop'
str(crime.train)
crime.train.SIR.2 <- crime.train


x.mat.sir <- as.matrix(crime.test[,impV.1]) %*% as.matrix(sir.vc.2)
```

```
dim(x.mat.sir) # 10개의 변수로 축약
crime.sir.test <- as.data.frame(cbind(x.mat.sir, crime.test[,103]))
crime.test <- crime.sir.test
colnames(crime.test)[11] <- 'ViolentCrimesPerPop'
str(crime.test)
crime.test.SIR.2 <- crime.test


crime.gam.SIR.2 <- gam(ViolentCrimesPerPop ~ s(Dir1, 4) + s(Dir2, 6) + s(Dir3, 6) + s(Dir4, 3) +
                       ns(Dir5, 4) + ns(Dir6, 4) + s(Dir7, 8) + ns(Dir8, 5),
                       data = crime.train.SIR.2, family = poisson('sqrt'))

mean((crime.test.SIR.2$ViolentCrimesPerPop -
        predict(crime.gam.SIR.2, crime.test.SIR.2, type = 'response'))^2) # 138597.3
sqrt(138597.3) # 372.2866


#### LASSO ####
crime.test.ls <- crime.test <- raw.test.std
crime.train.ls <- crime.train <- raw.train.std

crime.gam.ls <- gam(ViolentCrimesPerPop ~ ns(racePctWhite, 3) + ns(PctKids2Par, 5) +
                      PctKidsBornNeverMar + ns(MalePctDivorce, 6) +
                      ns(HousVacant, 6) + ns(PctPersDenseHous, 2),
                      data = crime.train.ls, family = poisson('sqrt'))

mean((crime.test.ls$ViolentCrimesPerPop -
        predict(crime.gam.ls, crime.test.ls, type = 'response'))^2) # 131527.5
sqrt(131527.5) # 362.6672

windows(10, 10)
par(mfrow = c(2, 3))
plot(crime.gam.ls, se = T, col = 3, lwd = 2)


#### 범죄단계 예측 성능 ####
crime.q <- quantile(crime.train$ViolentCrimesPerPop, c(0.25, 0.5, 0.75))

test.set <- crime.test$ViolentCrimesPerPop
pred.SIR.1 <- predict(crime.gam.SIR.1, crime.test.SIR.1, type = 'response')
pred.SIR.2 <- predict(crime.gam.SIR.2, crime.test.SIR.2, type = 'response')
pred.ls <- predict(crime.gam.ls, crime.test.ls, type = 'response')

test.grade <- ifelse(test.set < crime.q[1], 1,
                      ifelse(crime.q[1] <= test.set & test.set < crime.q[2], 2,
                             ifelse(crime.q[2] <= test.set & test.set < crime.q[3], 3, 4)))

pred.grade.SIR.1 <- ifelse(pred.SIR.1 < crime.q[1], 1,
                      ifelse(crime.q[1] <= pred.SIR.1 & pred.SIR.1 < crime.q[2], 2,
                             ifelse(crime.q[2] <= pred.SIR.1 & pred.SIR.1 < crime.q[3], 3, 4)))
```

```
pred.grade.SIR.2 <- ifelse(pred.SIR.2 < crime.q[1], 1,
                           ifelse(crime.q[1] <= pred.SIR.2 & pred.SIR.2 < crime.q[2], 2,
                                  ifelse(crime.q[2] <= pred.SIR.2 & pred.SIR.2 < crime.q[3], 3, 4)))

pred.grade.ls <- ifelse(pred.ls < crime.q[1], 1,
                        ifelse(crime.q[1] <= pred.ls & pred.ls < crime.q[2], 2,
                               ifelse(crime.q[2] <= pred.ls & pred.ls < crime.q[3], 3, 4)))


addmargins(table(test.grade, pred.grade.SIR.1))
addmargins(table(test.grade, pred.grade.SIR.2))
addmargins(table(test.grade, pred.grade.ls))
```