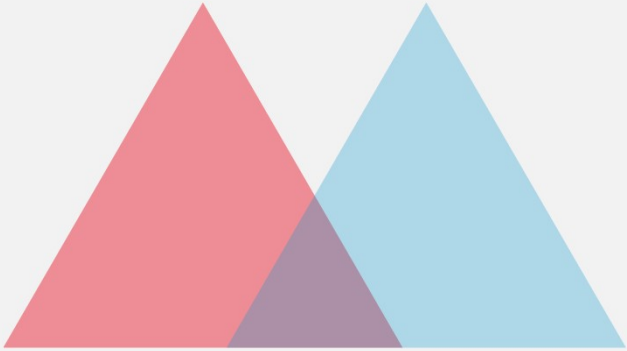
L1,L2 정규화
엘라스틱 넷
Stacking
Blending

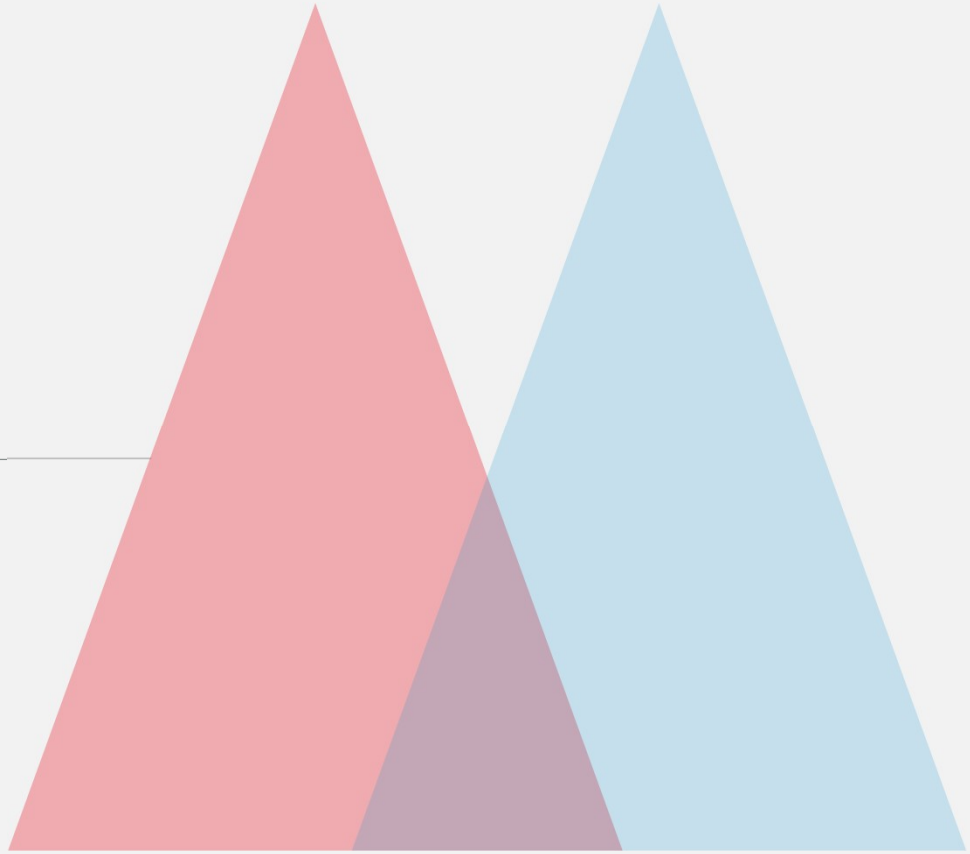


프로젝트2팀
송태인 김준호 박성민 서승수 최영현 한형진

Contents

001

L1 / L2 정규화
엘라스틱 넷



1

L1 정규화 (Lasso)

1

L2 정규화 (Ridge)

1

엘라스틱 넷 (Elastic Net)

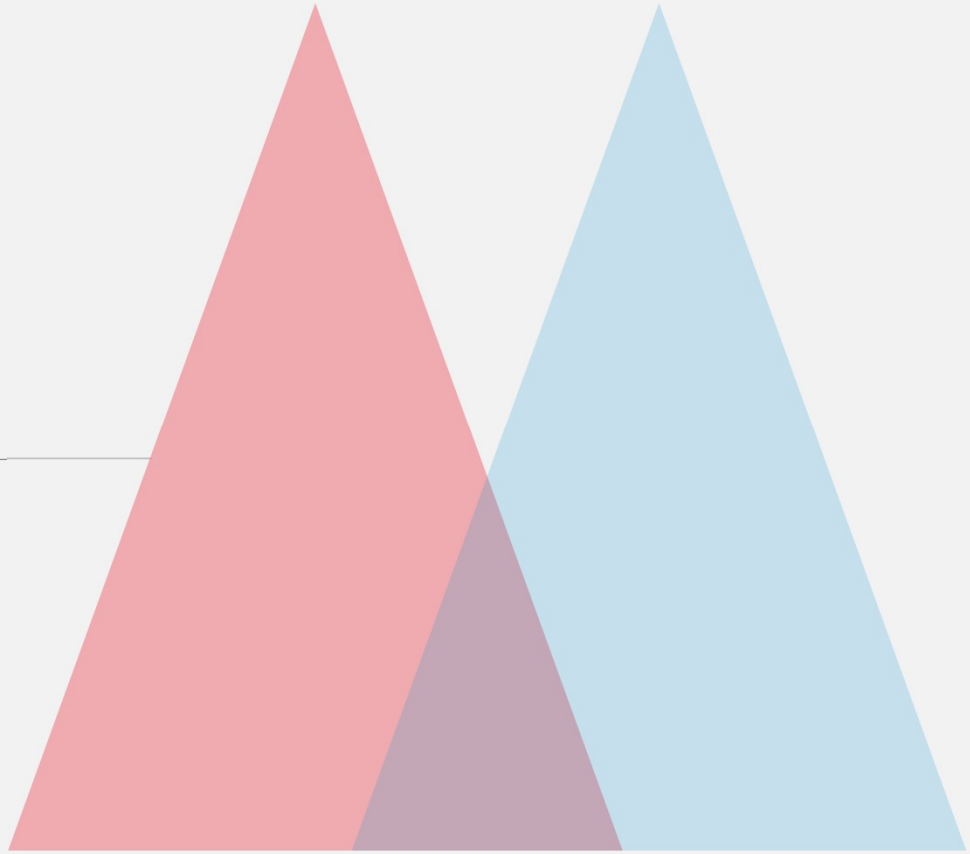
1

정규화 방식의 차이점

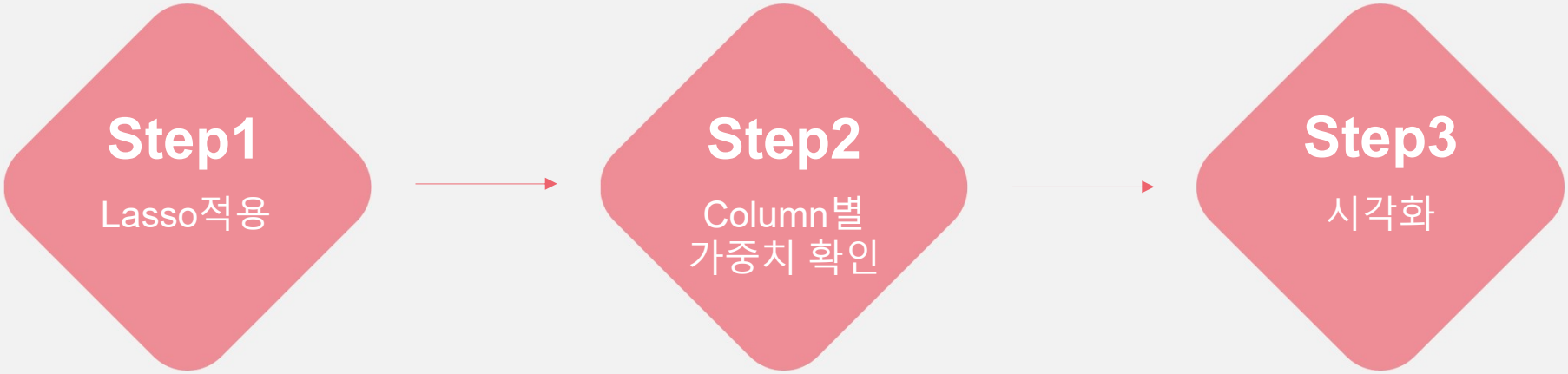
	L1	L2	Elastic Net
특징 1	변수 상관관계가 높을 때 성능 ↓	변수 상관관계가 높아도 좋은 성능	변수상관관계를 반영한 정규화
특징 2	비중요 변수를 우선적 줄임	중요 변수를 우선적 줄임	상관관계가 큰 변수를 선택/배제
특징 3	특성 선택에 유리	다중공산성 문제 해결	수행시간이 상대적 오래 걸림
overfitting을 방지하고 일반화 성능을 향상시키기 위함			

002

L1 / L2 정규화
엘라스틱 넷 적용하기



2 L1 적용



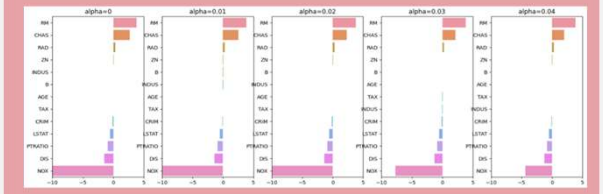
2 L1 적용

```
from sklearn.linear_model import Lasso
lasso = Lasso(alpha = 0.1)
```

Lasso 객체 형성

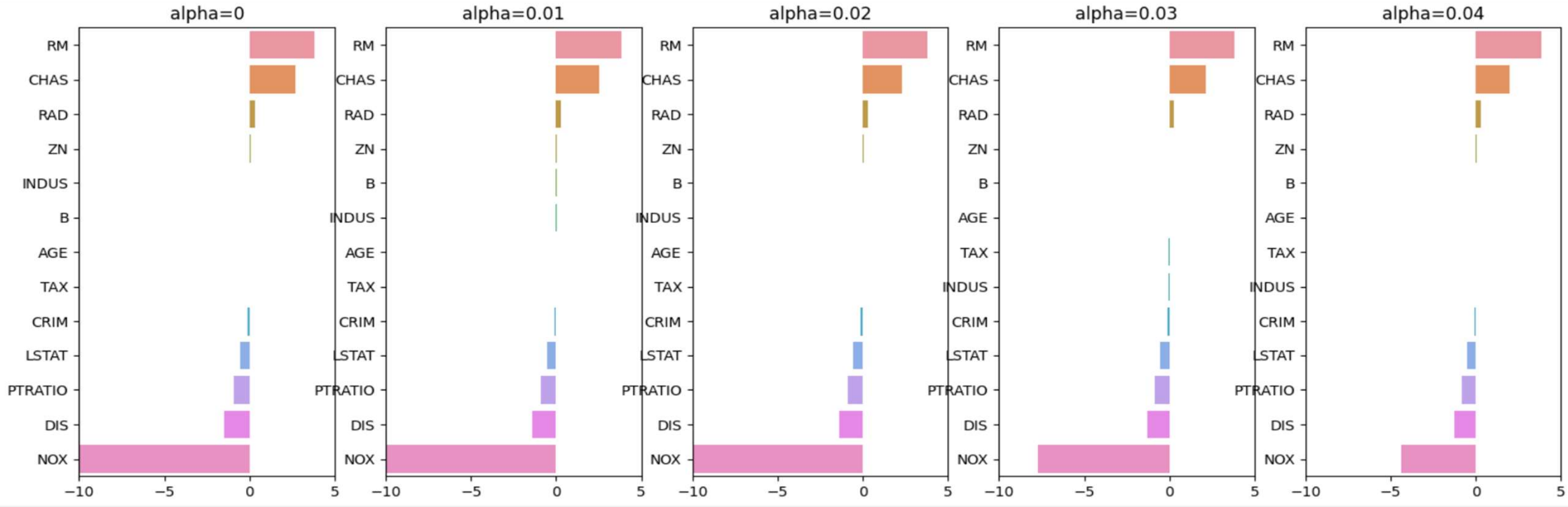
	alpha:0	alpha:0.01	alpha:0.02	alpha:0.03	alpha:0.04
RM	3.809865	3.814186	3.820592	3.825939	3.831277
CHAS	2.686734	2.504190	2.316725	2.131692	1.946662
RAD	0.306049	0.298526	0.292030	0.285037	0.278040
ZN	0.046420	0.046860	0.047382	0.047863	0.048345
INDUS	0.020559	0.006474	-0.004214	-0.016592	-0.028973
B	0.009312	0.009485	0.009664	0.009840	0.010017
AGE	0.000692	-0.001818	-0.004329	-0.006839	-0.009348
TAX	-0.012335	-0.012627	-0.013011	-0.013351	-0.013689
CRIM	-0.108011	-0.106228	-0.104380	-0.102566	-0.100752
LSTAT	-0.524758	-0.530481	-0.536422	-0.542254	-0.548088
PTRATIO	-0.952747	-0.916369	-0.880927	-0.845025	-0.809118
DIS	-1.475567	-1.422155	-1.366426	-1.311850	-1.257277
NOX	-17.766611	-14.394478	-11.078060	-7.733933	-4.389675

Alpha 값에 따른 가중치 변화



시각화

	alpha:0	alpha:0.01	alpha:0.02	alpha:0.03	alpha:0.04
RM	3.809865	3.814186	3.820592	3.825939	3.831277
CHAS	2.686734	2.504190	2.316725	2.131692	1.946662
RAD	0.306049	0.298526	0.292030	0.285037	0.278040
ZN	0.046420	0.046860	0.047382	0.047863	0.048345
INDUS	0.020559	0.006474	-0.004214	-0.016592	-0.028973
B	0.009312	0.009485	0.009664	0.009840	0.010017
AGE	0.000692	-0.001818	-0.004329	-0.006839	-0.009348
TAX	-0.012335	-0.012627	-0.013011	-0.013351	-0.013689
CRIM	-0.108011	-0.106228	-0.104380	-0.102566	-0.100752
LSTAT	-0.524758	-0.530481	-0.536422	-0.542254	-0.548088
PTRATIO	-0.952747	-0.916369	-0.880927	-0.845025	-0.809118
DIS	-1.475567	-1.422155	-1.366426	-1.311850	-1.257277
NOX	-17.766611	-14.394478	-11.078060	-7.733933	-4.389675



2 L2 적용

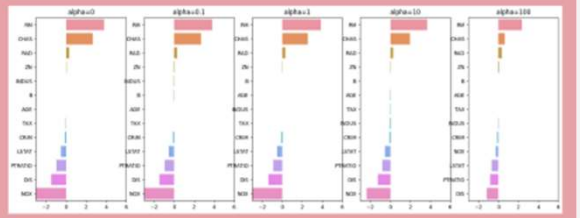
2 L2 적용

```
from sklearn.linear_model import Ridge
ridge=Ridge(alpha=10)
```

Ridge 객체 생성

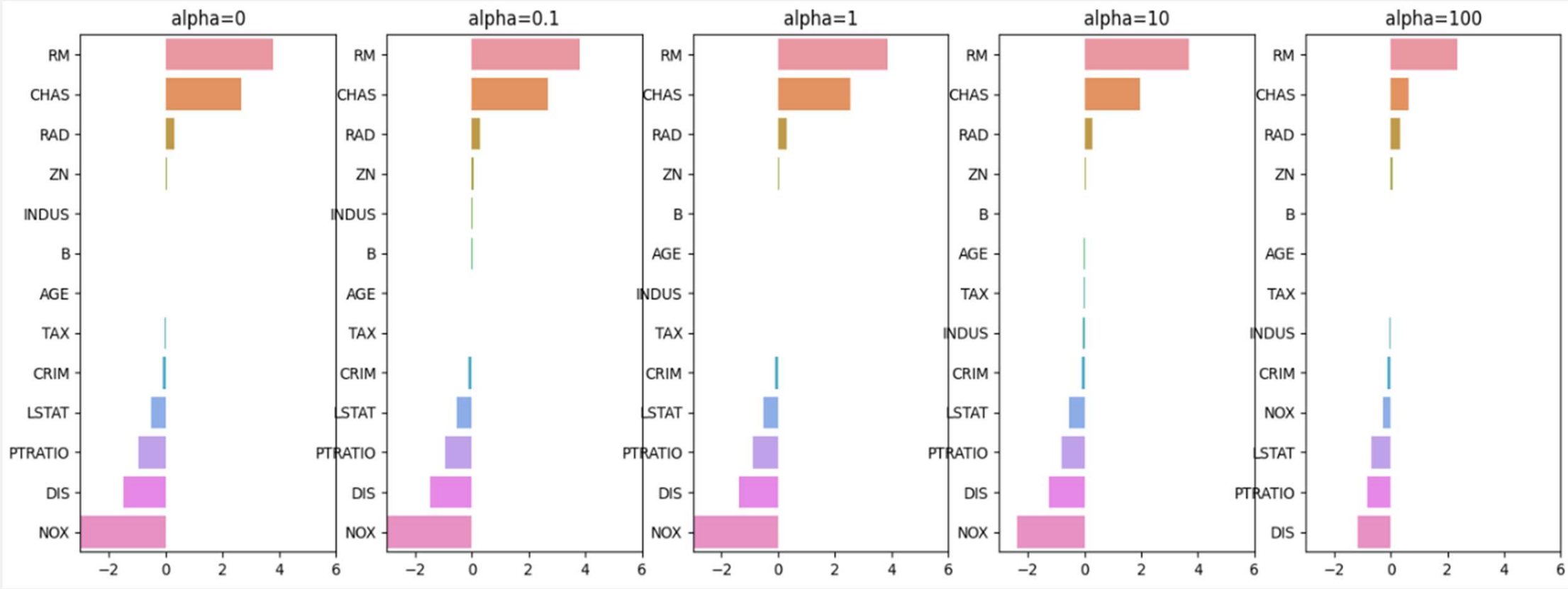
	alpha=0	alpha=0.1	alpha=1	alpha=10	alpha=100
RM	3.809865	3.818233	3.854000	3.702272	2.334536
CHAS	2.686734	2.670019	2.552393	1.952021	0.638335
RAD	0.306049	0.303515	0.290142	0.279596	0.315358
ZN	0.046420	0.046572	0.047443	0.049579	0.054496
INDUS	0.020559	0.015999	-0.008805	-0.042962	-0.052826
B	0.009312	0.009368	0.009673	0.010037	0.009393
AGE	0.000692	-0.000269	-0.005415	-0.010707	0.001212
TAX	-0.012335	-0.012421	-0.012912	-0.013993	-0.015856
CRIM	-0.108011	-0.107474	-0.104595	-0.101435	-0.102202
LSTAT	-0.524758	-0.525966	-0.533343	-0.559366	-0.660764
PTRATIO	-0.952747	-0.940759	-0.876074	-0.797945	-0.829218
DIS	-1.475567	-1.459626	-1.372654	-1.248808	-1.153390
NOX	-17.766611	-16.684645	-10.777015	-2.371619	-0.262847

Alpha 값에 따른 가중치 변화



시각화

	alpha=0	alpha=0.1	alpha=1	alpha=10	alpha=100
RM	3.809865	3.818233	3.854000	3.702272	2.334536
CHAS	2.686734	2.670019	2.552393	1.952021	0.638335
RAD	0.306049	0.303515	0.290142	0.279596	0.315358
ZN	0.046420	0.046572	0.047443	0.049579	0.054496
INDUS	0.020559	0.015999	-0.008805	-0.042962	-0.052826
B	0.009312	0.009368	0.009673	0.010037	0.009393
AGE	0.000692	-0.000269	-0.005415	-0.010707	0.001212
TAX	-0.012335	-0.012421	-0.012912	-0.013993	-0.015856
CRIM	-0.108011	-0.107474	-0.104595	-0.101435	-0.102202
LSTAT	-0.524758	-0.525966	-0.533343	-0.559366	-0.660764
PTRATIO	-0.952747	-0.940759	-0.876074	-0.797945	-0.829218
DIS	-1.475567	-1.459626	-1.372654	-1.248808	-1.153390
NOX	-17.766611	-16.684645	-10.777015	-2.371619	-0.262847



2

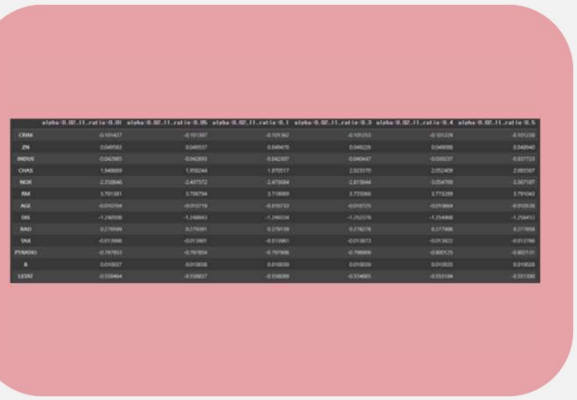
Elastic Net적용

2

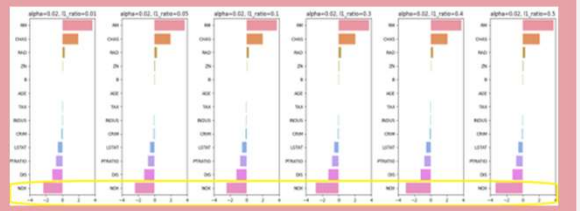
Elastic Net적용

```
from sklearn.linear_model import ElasticNet
elasticnet = ElasticNet(alpha=0.1, l1_ratio=0.5)
```

Elastic Net 객체 생성

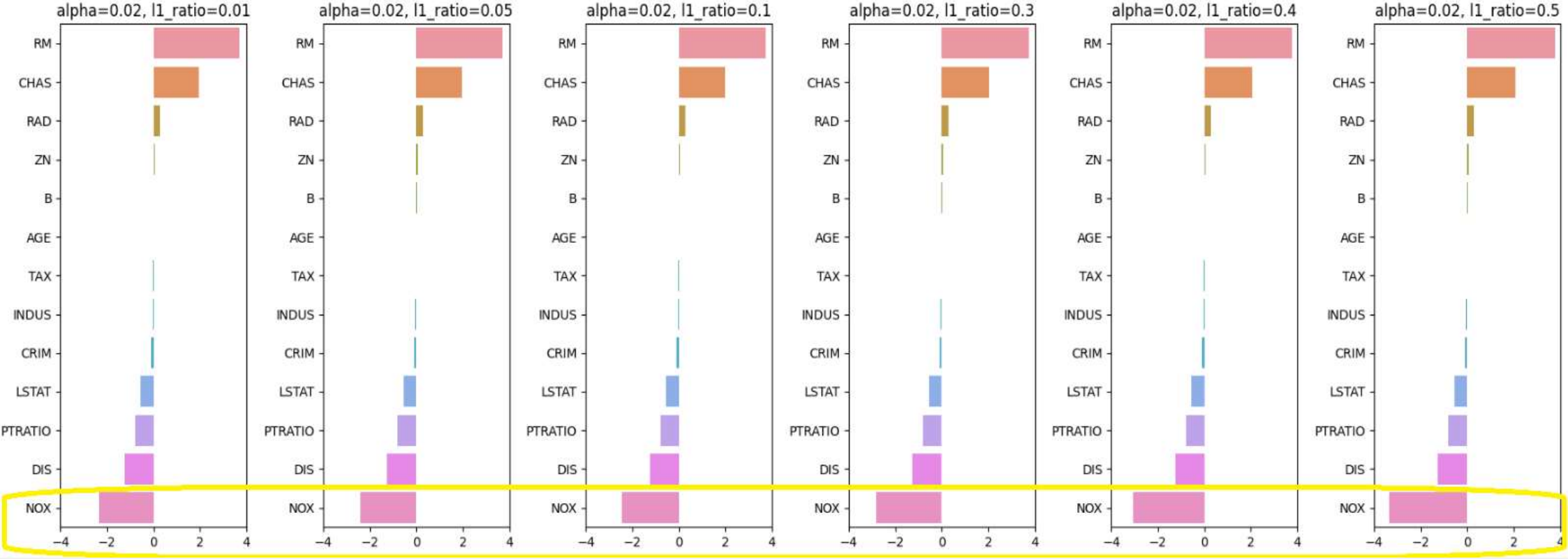


Alpha 값과 l1_ratio 값에 따른 가중치 변화



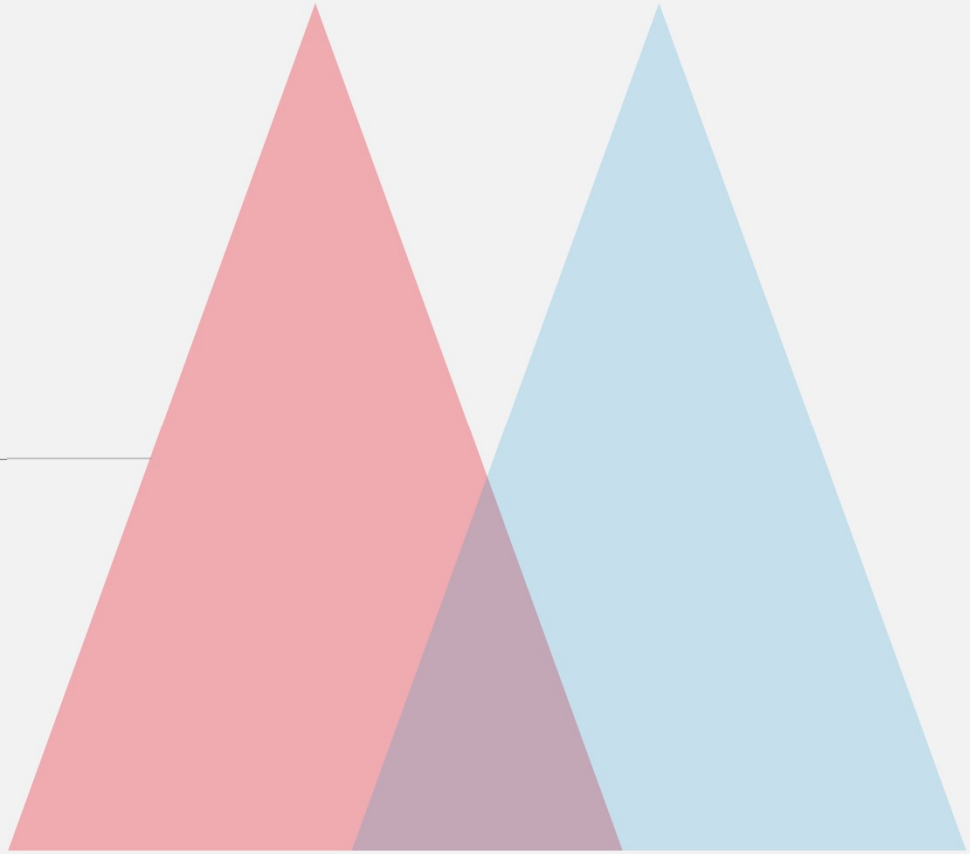
시각화

	alpha:0.02_l1_ratio:0.01	alpha:0.02_l1_ratio:0.05	alpha:0.02_l1_ratio:0.1	alpha:0.02_l1_ratio:0.3	alpha:0.02_l1_ratio:0.4	alpha:0.02_l1_ratio:0.5
CRIM	-0.101427	-0.101397	-0.101362	-0.101253	-0.101229	-0.101239
ZN	0.049583	0.049537	0.049478	0.049226	0.049088	0.048940
INDUS	-0.042985	-0.042693	-0.042307	-0.040447	-0.039237	-0.037723
CHAS	1.948669	1.958244	1.970517	2.023370	2.052409	2.083567
NOX	-2.358846	-2.407572	-2.473684	-2.815644	-3.054769	-3.367187
RM	3.701381	3.708794	3.718069	3.755066	3.773289	3.791043
AGE	-0.010704	-0.010719	-0.010733	-0.010725	-0.010664	-0.010538
DIS	-1.248508	-1.248843	-1.249334	-1.252379	-1.254868	-1.258412
RAD	0.279599	0.279391	0.279139	0.278278	0.277986	0.277856
TAX	-0.013996	-0.013981	-0.013961	-0.013873	-0.013822	-0.013766
PTRATIO	-0.797853	-0.797854	-0.797906	-0.798909	-0.800125	-0.802131
B	0.010037	0.010038	0.010039	0.010039	0.010035	0.010028
LSTAT	-0.559464	-0.558857	-0.558089	-0.554885	-0.553184	-0.551390



003

모델 앙상블



3

모델 앙상블 - Stacking

스태킹(Stacking)

▲원리)

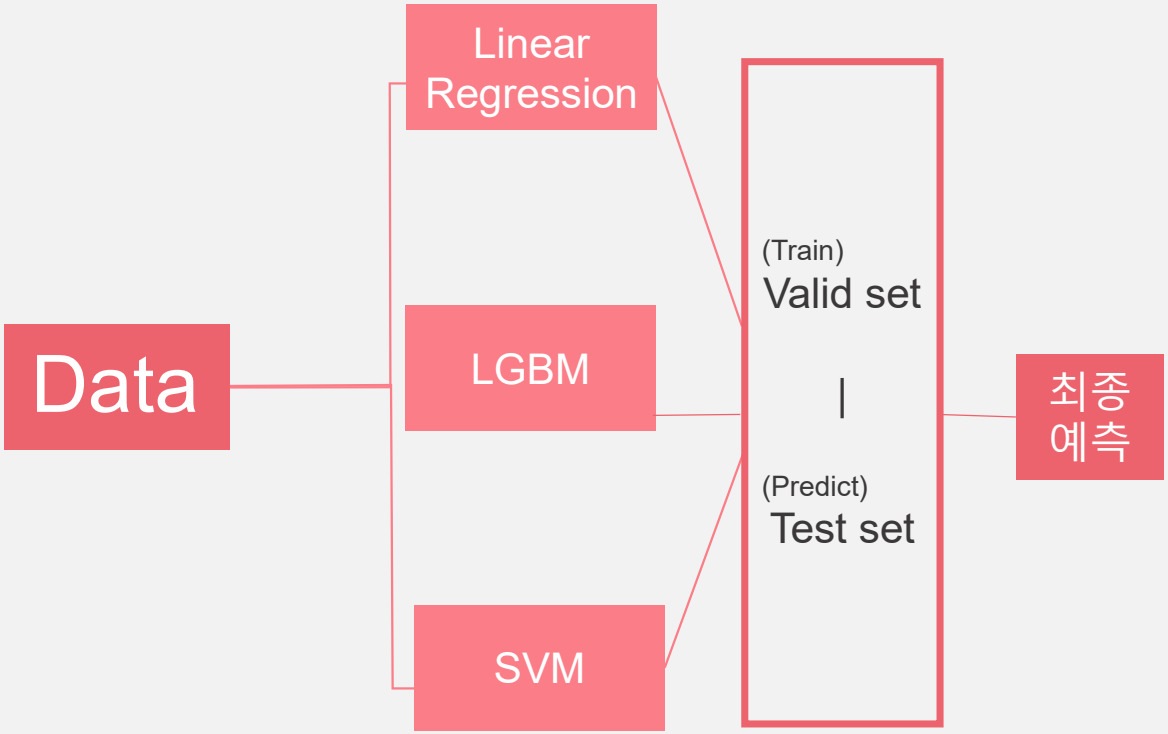
1가지 데이터를 가지고 n개의 모델을 학습 시킨 후
n개의 예측값을 새로운 모델(Meta Model) 학습데이터로
새로운 예측값을 확인

▲특징

1. 사이킷 런에서 모듈을 제공
2. 개별 모델에 대한 약점을 보안
3. 기존 학습 모델과 최종 모델을 선택
4. 모델의 복잡성이 증가
5. 다양한 모델을 사용할 수 있기에 섬세한 조정 필요
6. 계산 비용 증가(메모리사용량, 연산량...etc)

3

모델 앙상블 - Blending



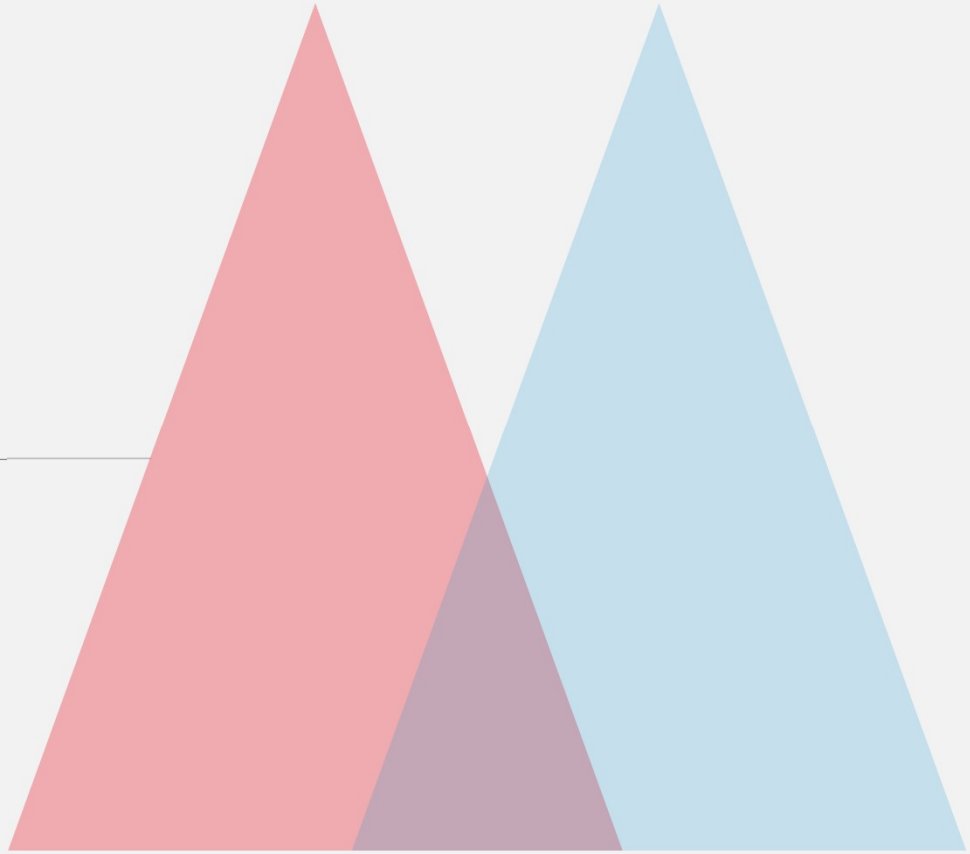
블랜딩 (Blending)

▲ 원리)

1. Data를 3가지로 나눈다 (Train, Test, Valid)
2. 각각의 모델의 Train으로 학습
3. 각각의 모델의 predict를 Test와 Valid를 적용
4. 각각 모델의 Test predict을 메타 모델에 train
6. Valid set로 예측 값의 결과 확인

004

모델 앙상블 사용하기



4

Sklearn 유방암 데이터 셋

Target	종양의 악성, 양성 여부
1	의심세포의 질감, 크기,너비 등
2	질감,크기,너비 오차
3	암 세포들의 최대 크기,넓이,질감

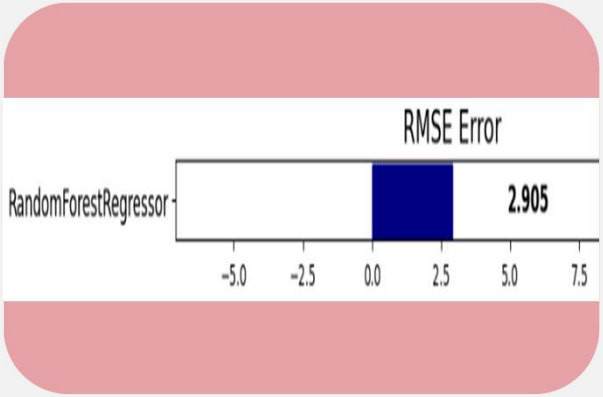
총 30개의 독립변수와 Target의 관계를 확인한다.

4

모델 앙상블 Stacking 사용하기

4

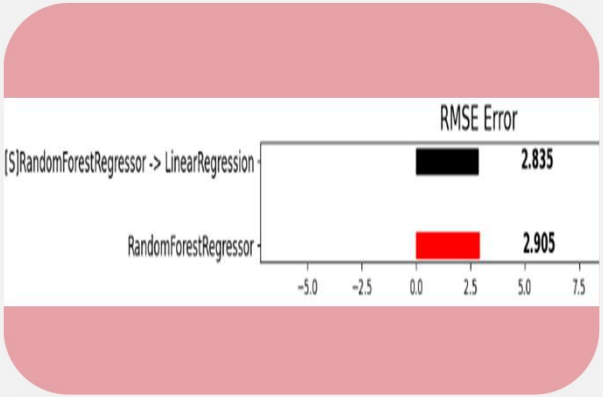
Stacking 적용



1개 모델의 RMSE

```
lr = LinearRegression()
stack = StackingRegressor([('rf',rf)],
                          final_estimator=lr,
                          n_jobs=-1)
```

Stacking 객체 설정

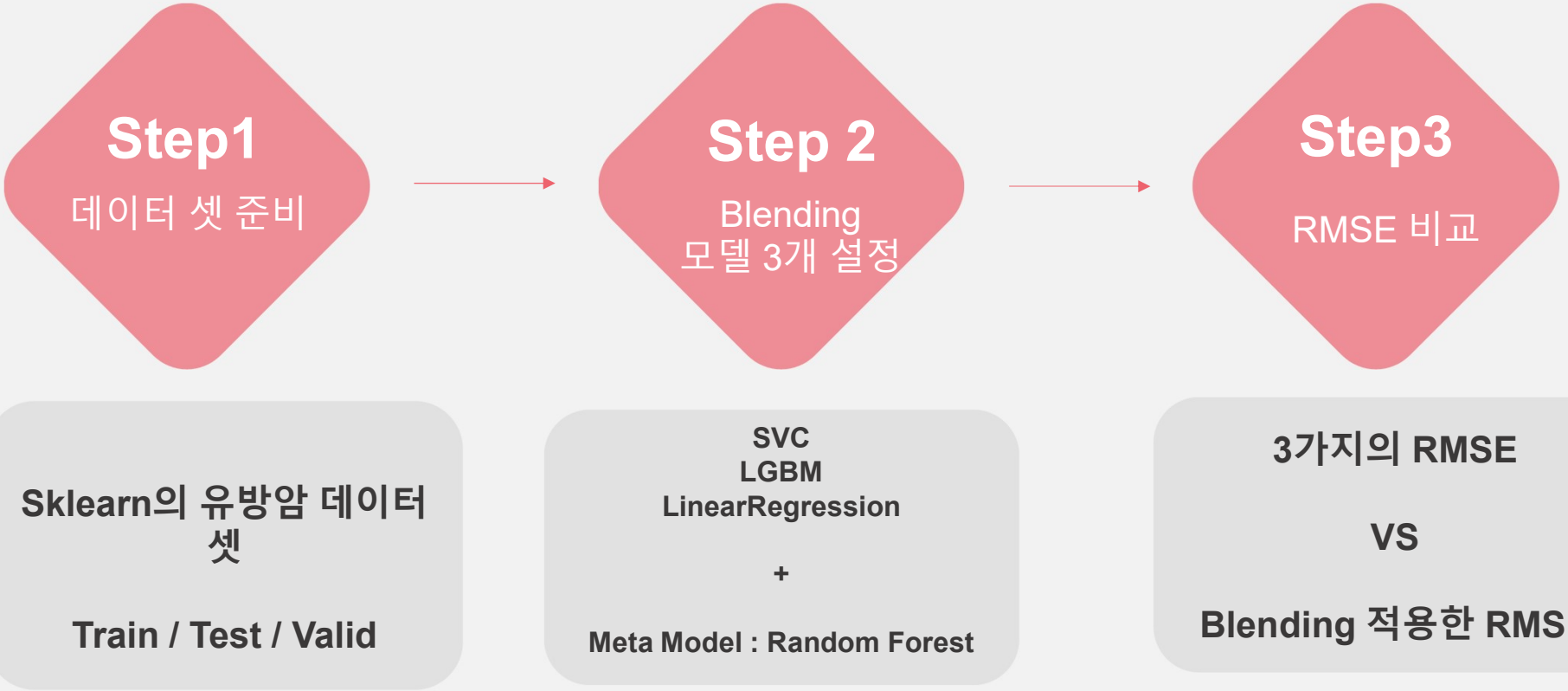


RMSE 비교

1. Stacking은 sklearn에서 제공하는 모듈
2. 기존 학습한 모델을 stack_models에 저장
3. stacking 모듈에 기본모델 + 새로운 모델 입력

4

모델 앙상블 Blending 사용하기



4

Blending 적용

```
x_train,x_test,y_train,y_test = train_test_split(df.drop('target', 1),
                                                 df['target'],
                                                 test_size = 0.2,
                                                 random_state=17)
x_train,x_val,y_train,y_val = train_test_split(x_train,
                                               y_train,
                                               test_size=0.1,
                                               random_state=17)
```

데이터 셋 분리

전체 데이터의 72% train

전체 데이터의 20% test

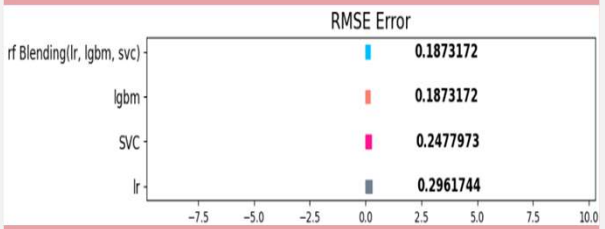
전체 데이터의 8% Valid

```
SVC RMSE: 0.22941573387056177
LightGBM RMSE: 0.1873171623163388
LogisticRegression RMSE: 0.24779731389167603
```

3개 모델의 RMSE

Train – Train Data set

Predict – Test / Valid



RMSE 비교

Blending을 적용한 모델의 RMSE: 0.18

LGBM RMSE: 0.18

SVC RMSE: 0.24

Lr RMSE: 0.29

QnA


Thanks