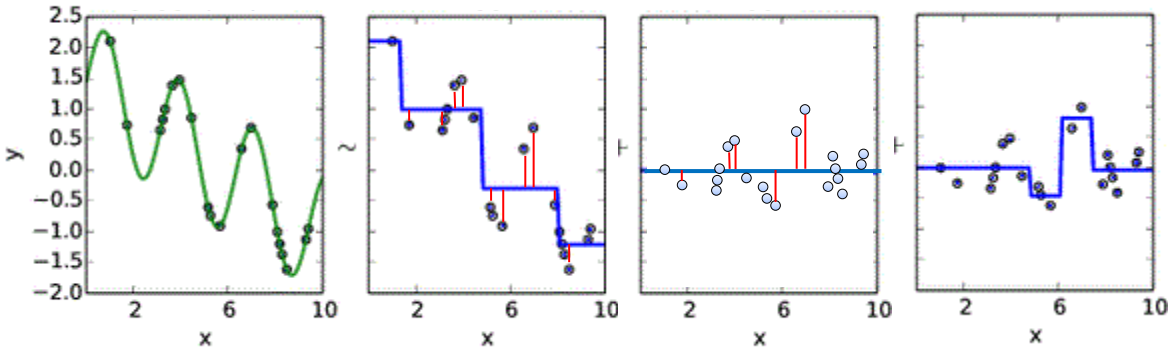


Boost _ ADA _ GBM

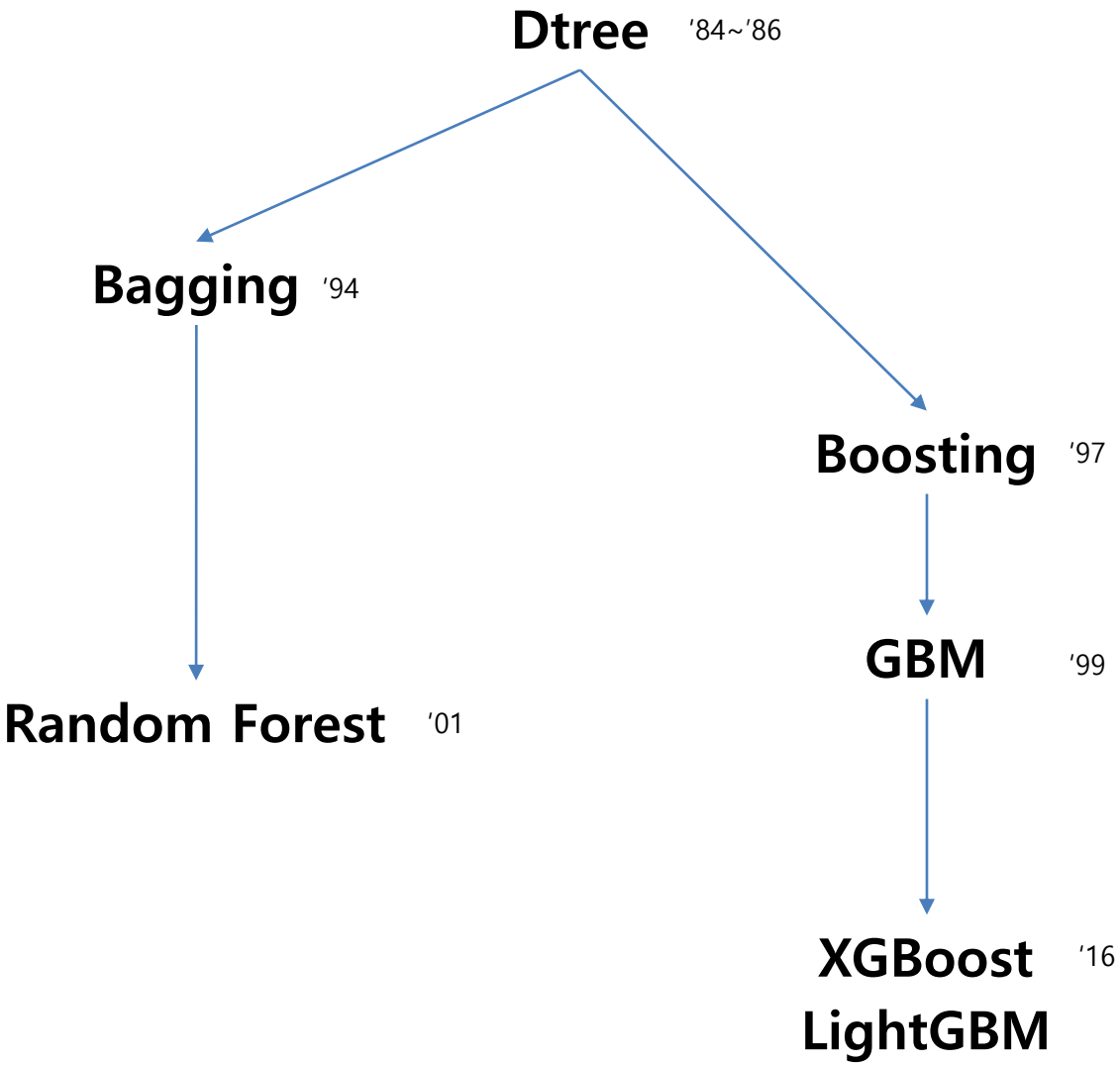
구름

도시공학과 일반대학원

한양대학교

1. **Adaboost**
2. GBM
3. 종합

의사결정나무 알고리즘 확장



Adaboost (Adaptive Boosting)

약한 분류기(weak learner)를 여러 개 연결하여 강한 분류기(Strong Learner)를 생성하는 메타 알고리즘

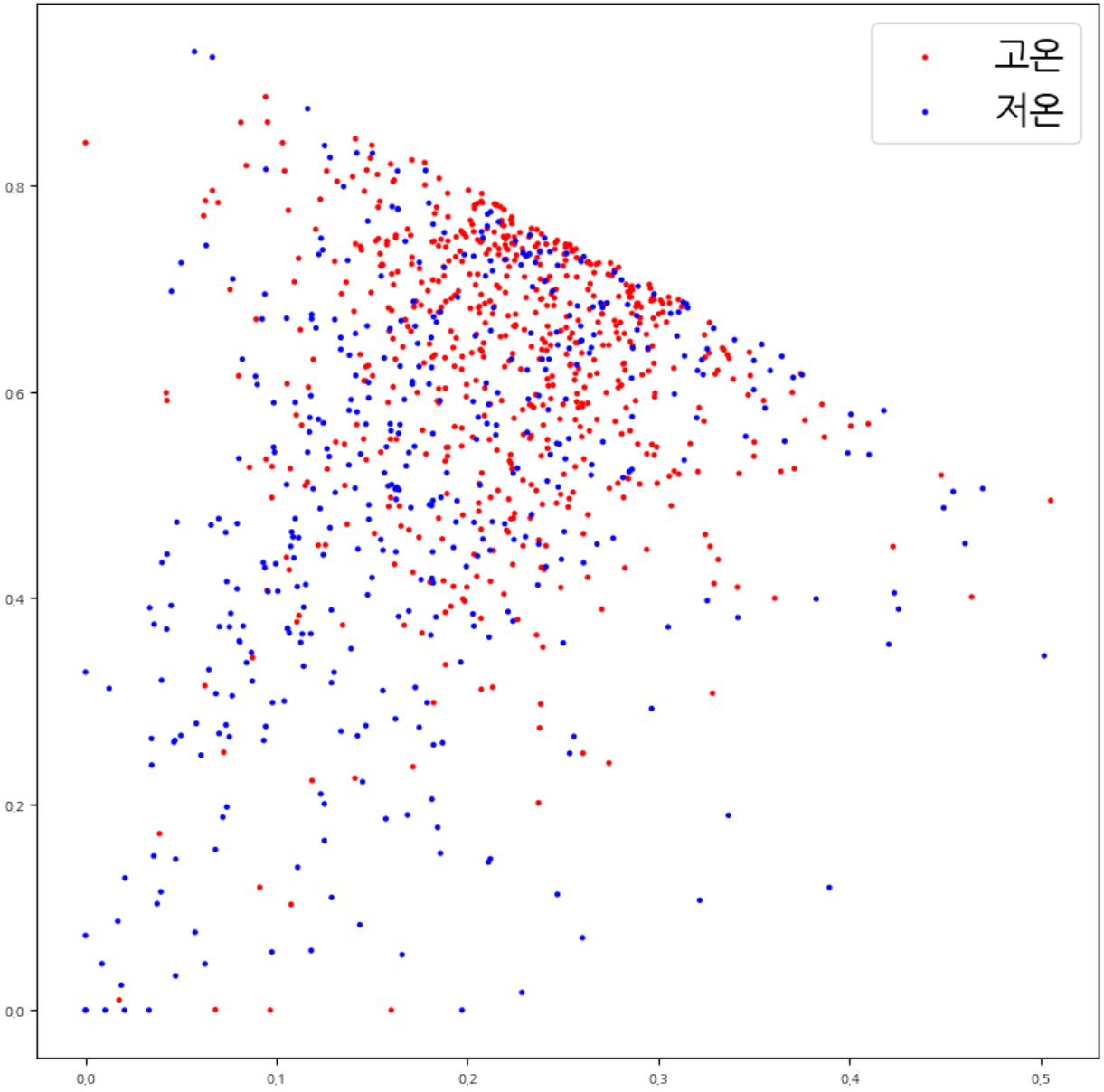
1. 학습데이터 세트 내 모든 데이터의 선택 확률 동일하게 초기화 (w_i)
2. 선택확률을 이용해 데이터 샘플 복원 추출
3. Weak Learner 모델을 이용하여 학습 (G_m)
4. 사용한 데이터 샘플로 학습 정확도 계산 (err_m)
5. 모델 가중치를 계산 (α_m)
6. 학습데이터 선택 확률 갱신 (w_i)
7. 2~6번 반복 (M)
8. 최종 모델 (G)

Adaboost (Adaptive Boosting)

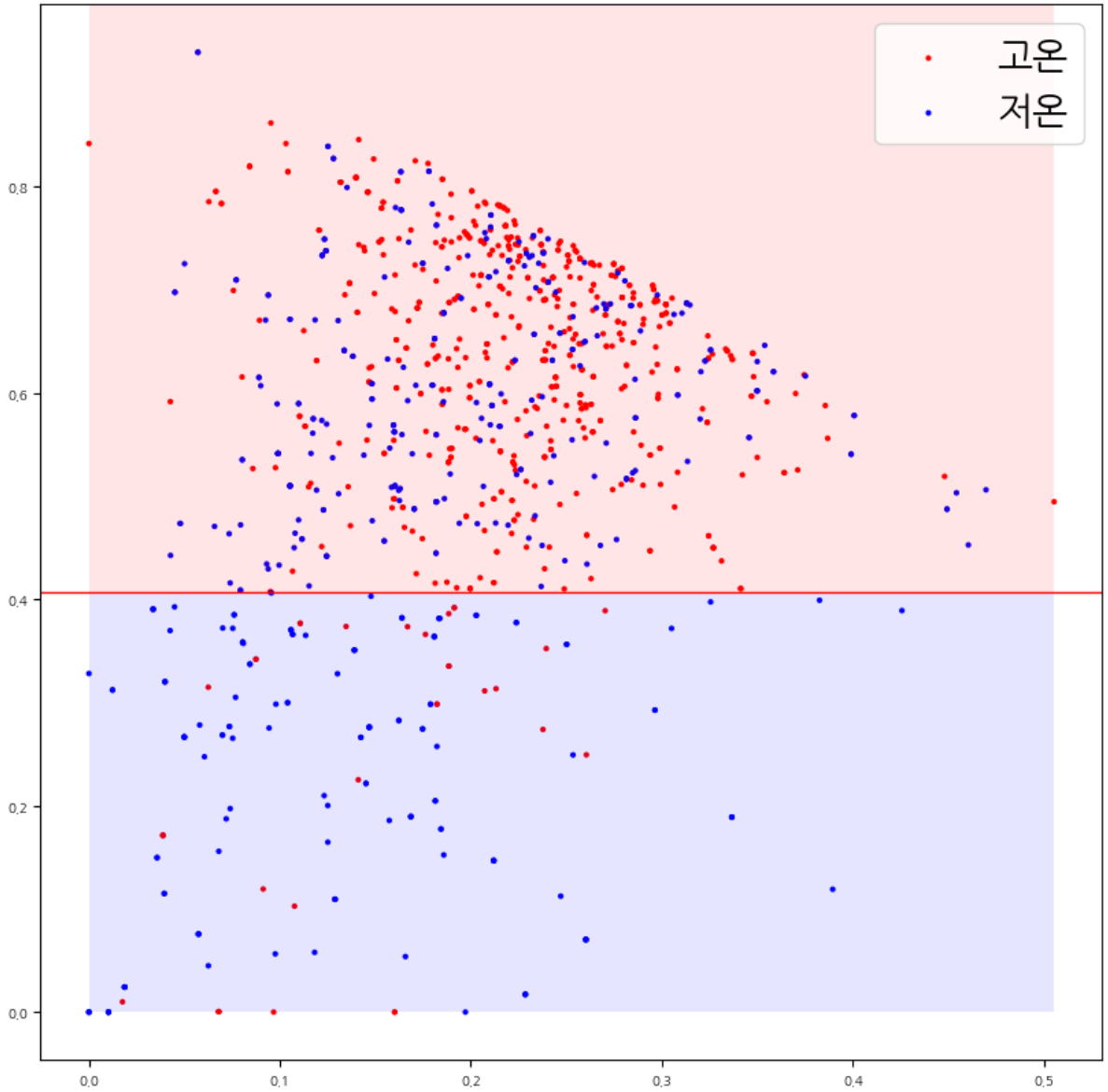
Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

S-dot Data (x = 도로, y = 대지)



Stump Tree

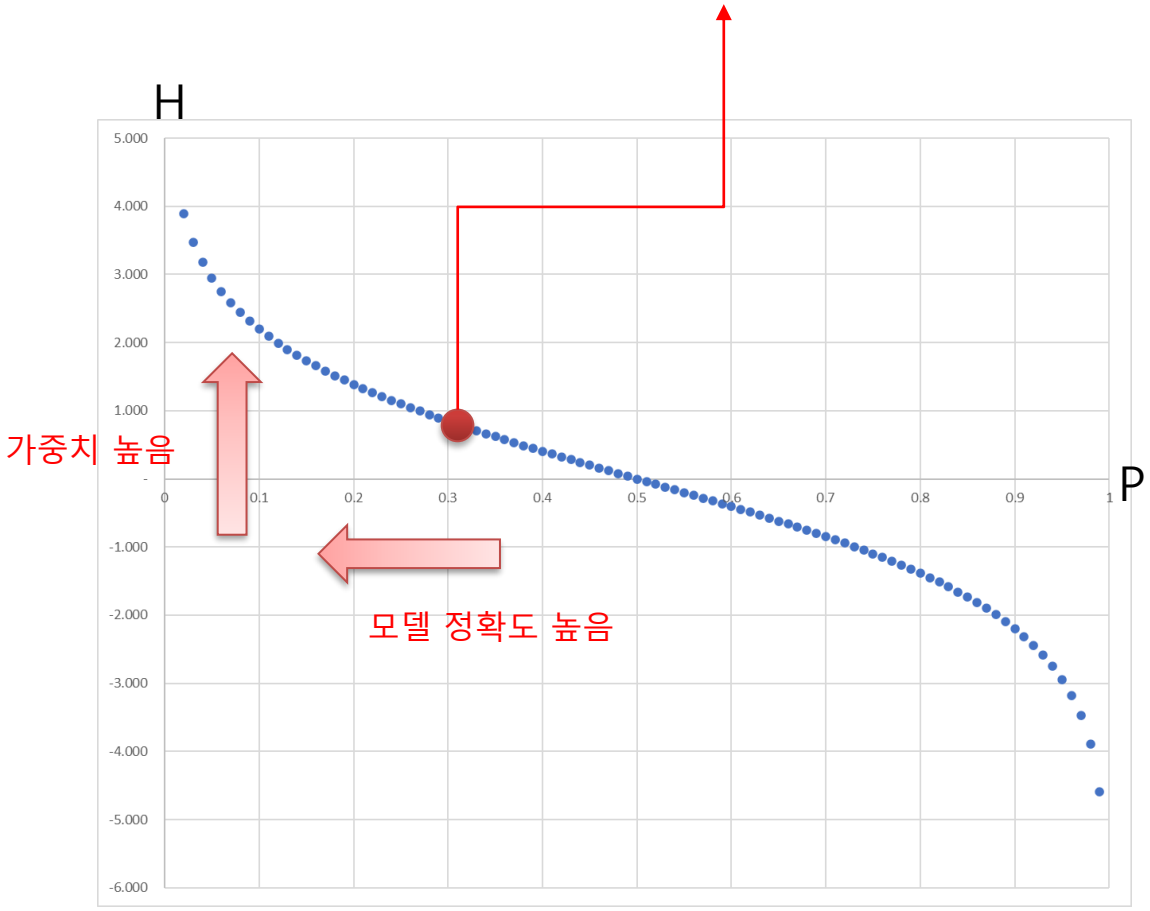


대 > 0.4067인 경우
데이터 수 = 854,
고온 수 : 560(0.66),
저온 수 : 294(0.34)

대 < 0.4067인 경우
데이터 수 = 177,
고온 수 : 34(0.19),
저온 수 : 143(0.81)

err_m = 예측값과 실제값이 다를 확률 = 0.3181

$\alpha_m = \ln\left(\frac{1 - err_m}{err_m}\right)$ = 모델 가중치 = 0.7623



$$w_i = w_i \times exp[\alpha_m \times I(y_i \neq G_m(x_i))]$$

y_i $G_m(x_i)$ $exp[\alpha_m \times I(y_i \neq G_m(x_i))]$

y - NumPy object array

	0
0	1
1	-1
2	-1
3	-1
4	1
5	-1
6	1
7	1
8	-1
9	-1
10	-1
11	-1
12	-1
13	-1

y_pred - NumPy object

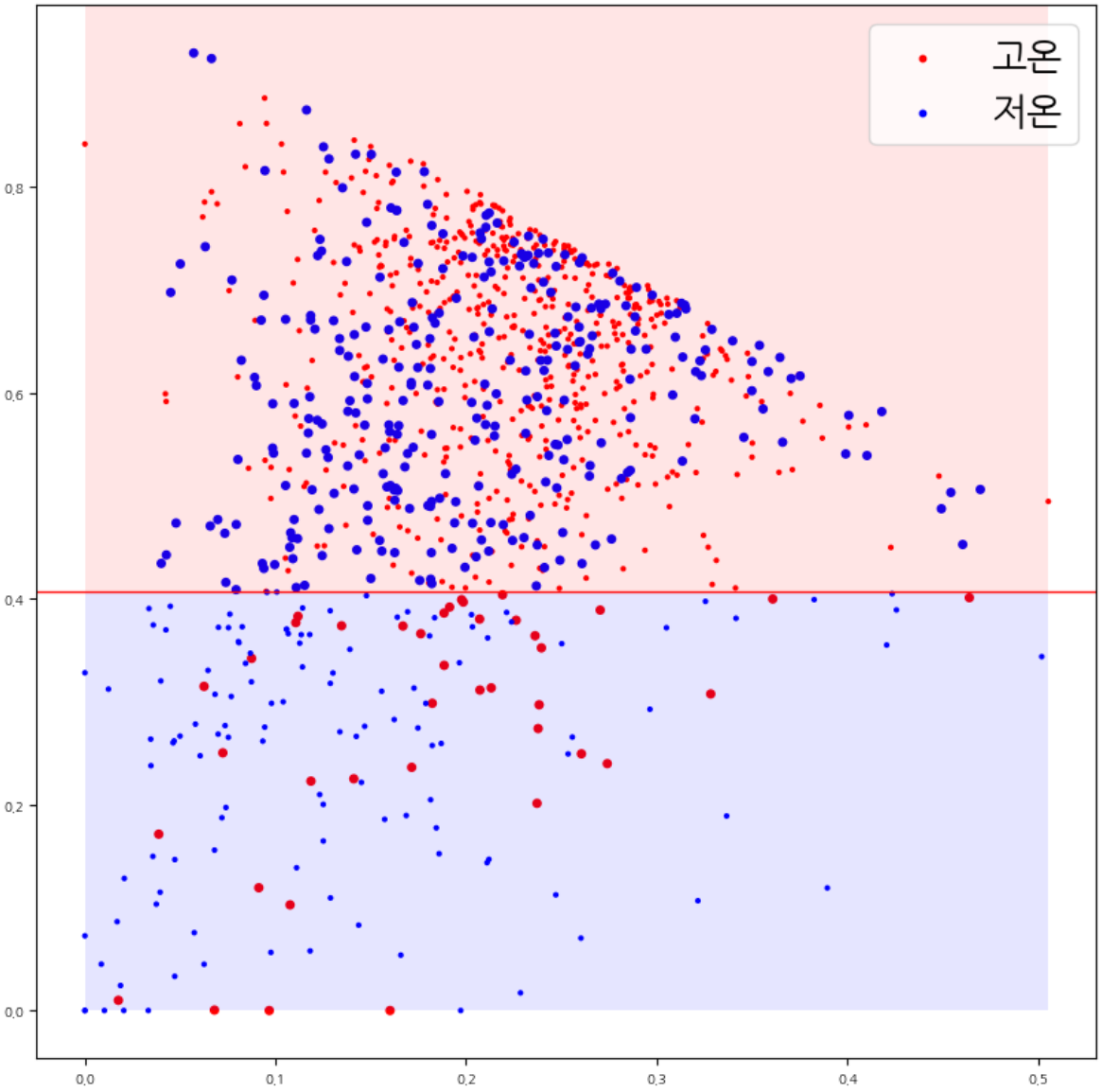
	0
0	1
1	1
2	1
3	1
4	1
5	-1
6	1
7	1
8	1
9	1
10	1
11	1
12	1
13	1

tmp - NumPy object array

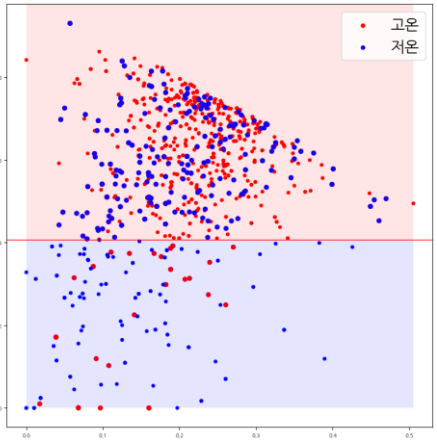
	0
0	1
1	2.14329
2	2.14329
3	2.14329
4	1
5	1
6	1
7	1
8	2.14329
9	2.14329
10	2.14329
11	2.14329
12	2.14329
13	2.14329

틀린 데이터셋만 2배씩

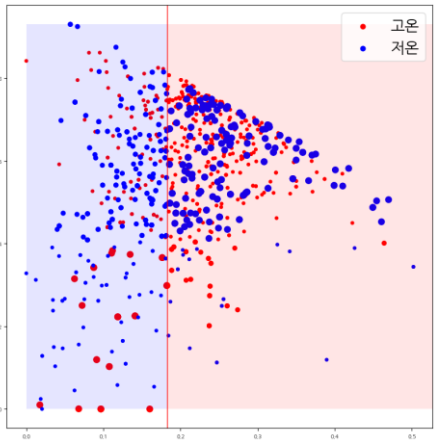
오답 데이터의 가중치 증가



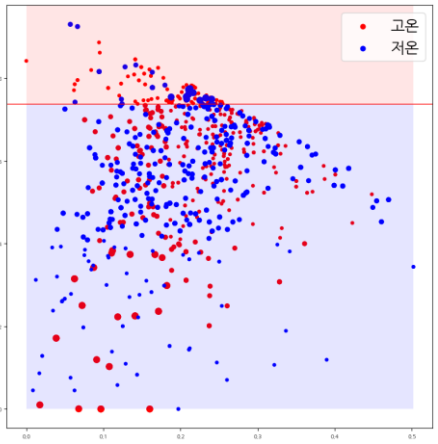
100 번 반복 학습 진행



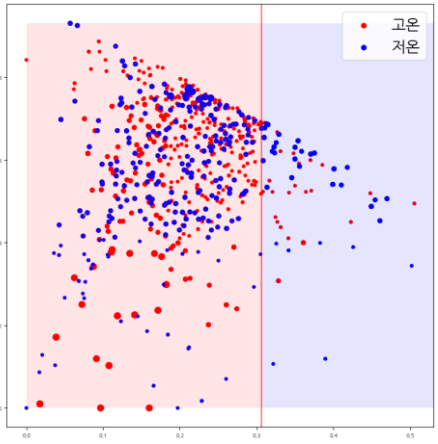
가중치 : 0.7623432834265215



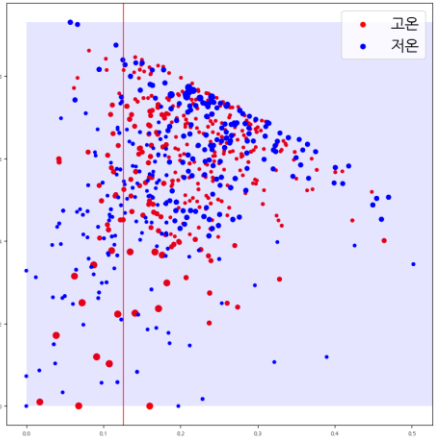
가중치 : 0.5634282210700787



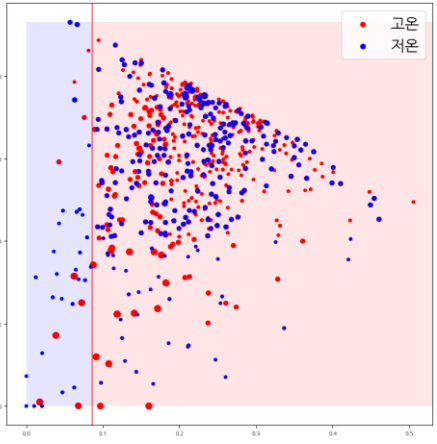
가중치 : 0.350789111761471



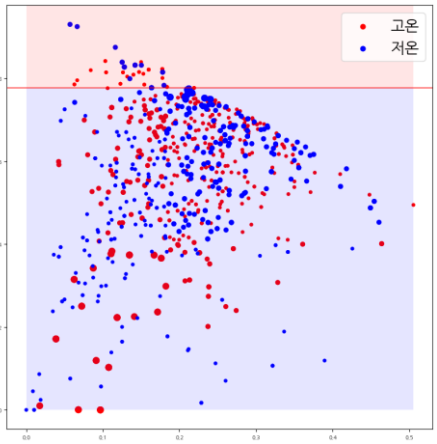
가중치 : 0.1848113322220434



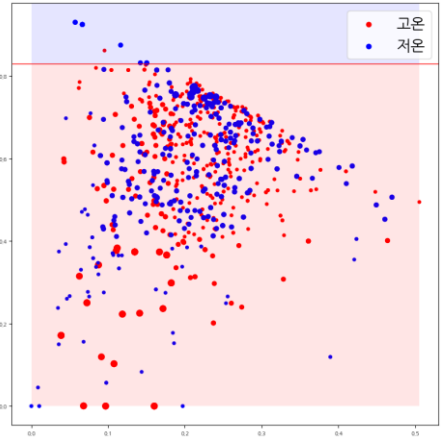
가중치 : 0.2043943646349172



가중치 : 0.10290351196477415



가중치 : 0.1808990814474119

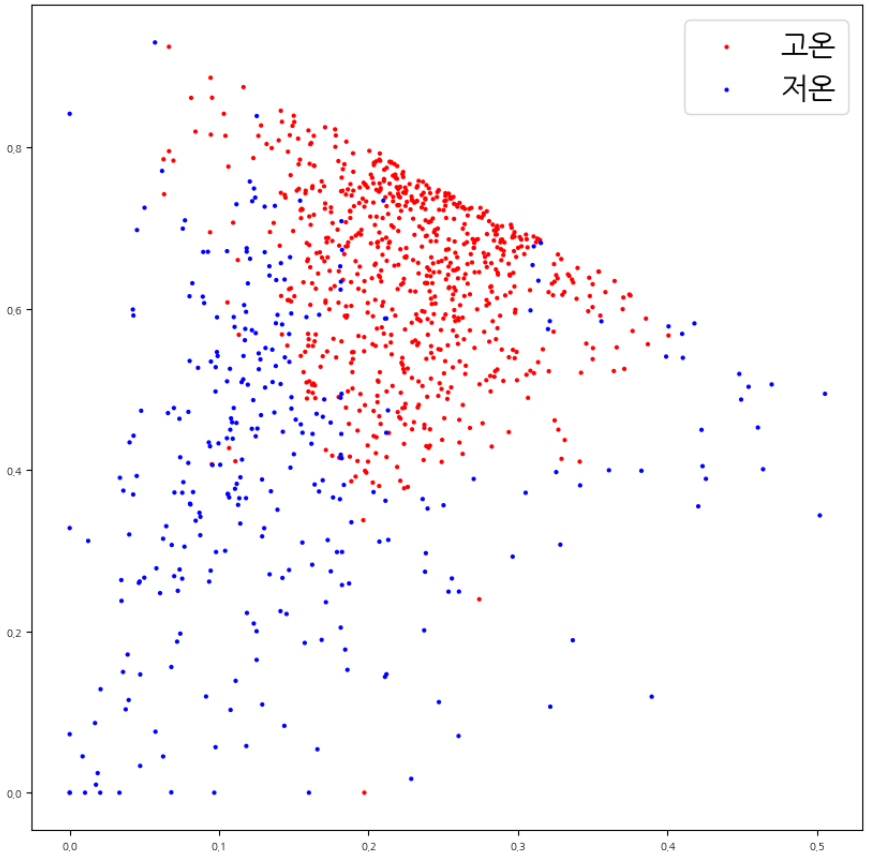


가중치 : 0.11847008480742165

최종 모델

$$\text{Output } G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$$

개별 모델의 예측 값(+1, -1) 에 각 모델의 가중치를 곱하여 Sum
0 기준으로 크면 +1, 작으면 -1로 예측



Adaboost feature importance

= 개별모델의 feature importance * 가중치

Adaboost Learning Rate

가중치 = $\text{LearningRate} * \ln\left(\frac{1 - err_m}{err_m}\right)$

1. Adaboost
2. **GBM**
3. 종합

GBM (Gradient Boosting Machine)

Adaboost

1. Weak learner 모델을 연결하여 예측
2. 이전 모델의 오분류 데이터에 가중치를 부여하여 학습데이터셋 추출

GBM

1. Weak learner 모델을 연결하여 예측
2. Gradient Descent(경사 하강) 개념을 통해 다음 학습데이터셋 생성

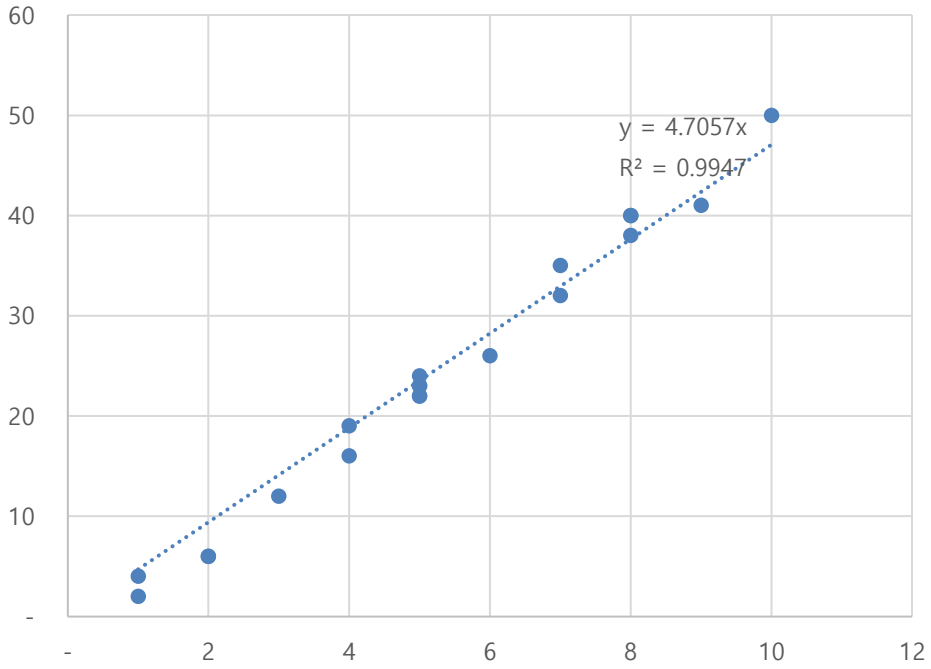
Gradient Descent (경사하강법)

선택 가능 변수 1개 가정 (2차원 공간)

$$f(x) = \alpha x$$

임의의 값
2

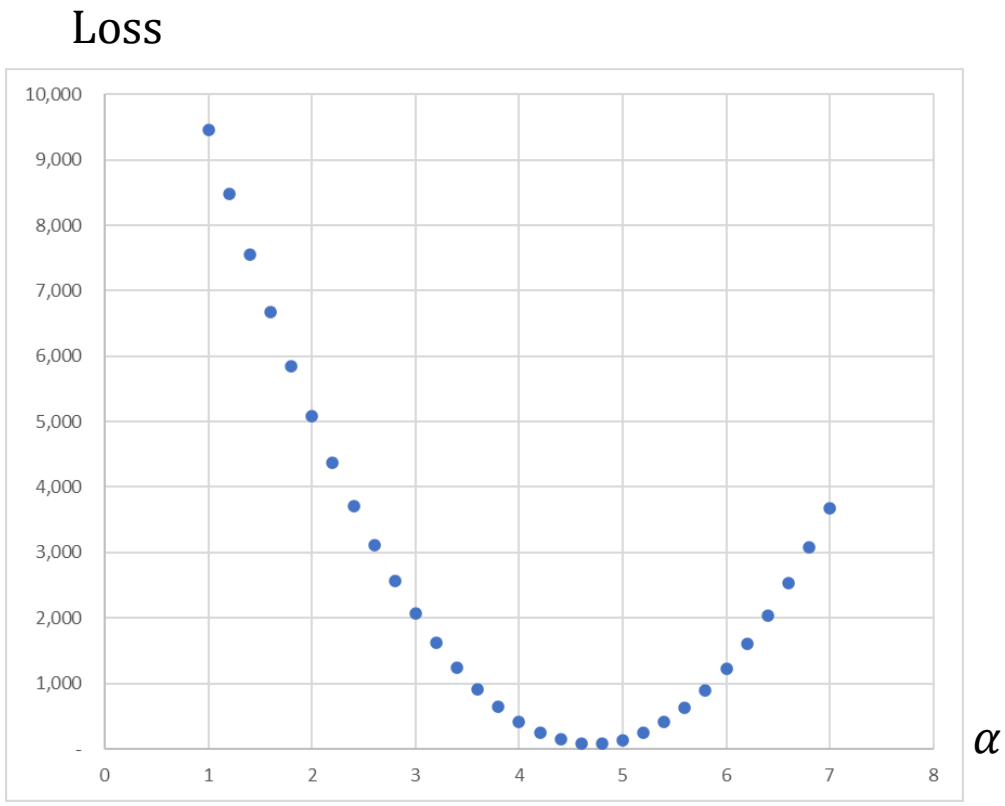
x	y
2	6
6	26
5	23
3	12
5	22
5	24
2	6
10	50
9	41
1	2
5	23
4	16
7	32
5	22
8	40
1	4
8	40
7	35
4	19
8	38



Gradient Descent (경사하강법)

$f(x) = 2x$

x	y	y_hat	loss
2	6	4	4
6	26	12	196
5	23	10	169
3	12	6	36
5	22	10	144
5	24	10	196
2	6	4	4
10	50	20	900
9	41	18	529
1	2	2	-
5	23	10	169
4	16	8	64
7	32	14	324
5	22	10	144
8	40	16	576
1	4	2	4
8	40	16	576
7	35	14	441
4	19	8	121
8	38	16	484
			5,081



$$\text{Loss} = \frac{1}{2}\sum(y_i - (2x_i))^2$$

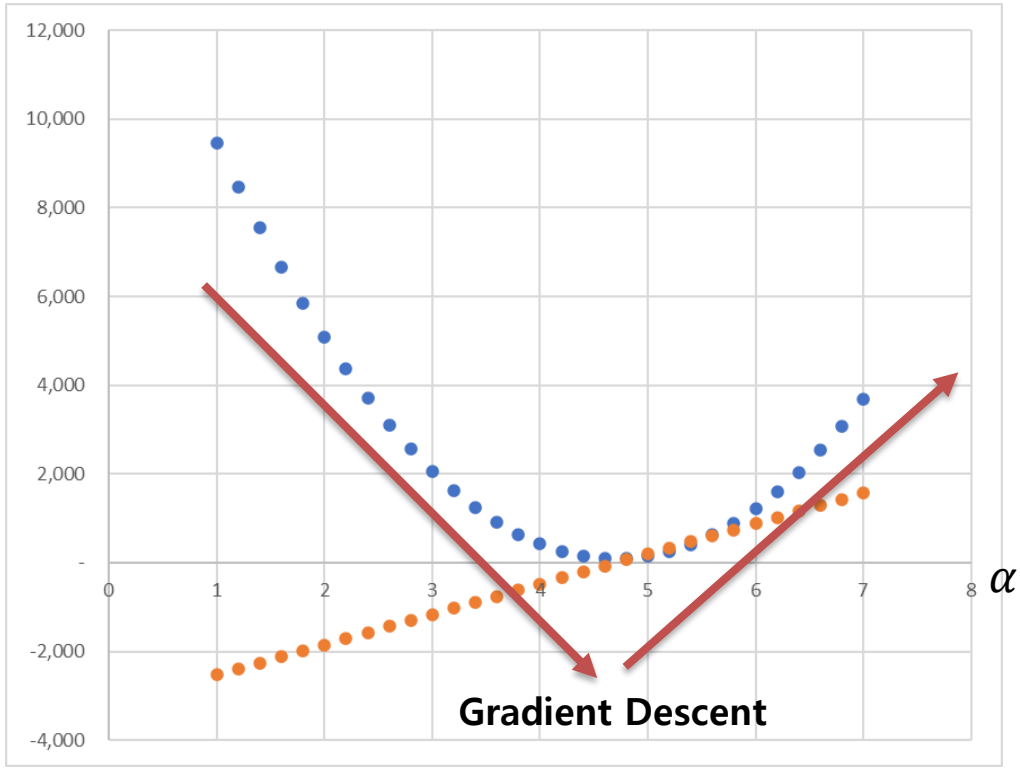
Gradient Descent (경사하강법)

$$f(x) = \alpha x$$

$$\text{Loss} = \frac{1}{2}\sum\left(y_i - (\alpha x_i)\right)^2$$

$$\frac{\partial \text{Loss}}{\partial \alpha} = \sum {x_i}^2 \alpha - \sum y_i x_i$$

경사의 반대로 이동 : $-\frac{\partial \text{Loss}}{\partial \alpha}$



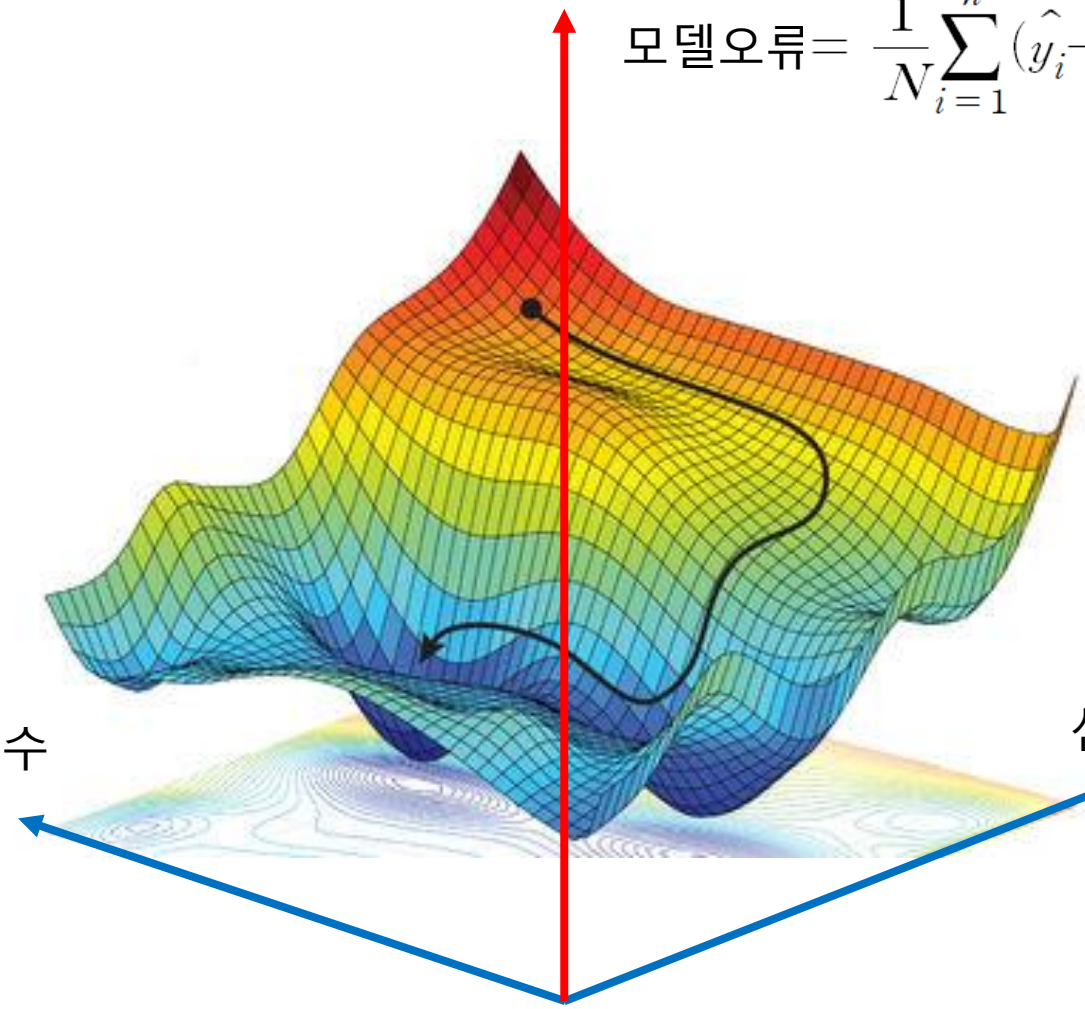
Loss

$\frac{\partial \text{Loss}}{\partial \alpha}$

Gradient Descent (경사하강법)

선택 가능 변수 2개 가정 (3차원 공간)

모델오류= $\frac{1}{N}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$



GBM에서의 경사 하강

1. Loss Function의 정의

$$\min L = \frac{1}{2}\sum \left(\mathrm{y_i} - f(x_\mathrm{i})\right)^2$$

2. 각 데이터셋의 예측 값의 Loss Function에 대한 기울기

$$\frac{\partial \mathrm{L}}{\partial f(x_\mathrm{i})} = f(x_\mathrm{i}) - \mathrm{y_i}$$

3. Loss를 줄이기 위해 잔차를 추가로 예측하는 모델이 필요

$$- \frac{\partial \mathrm{L}}{\partial f(x_\mathrm{i})} = \mathrm{y_i} - f(x_\mathrm{i}) = \text{잔차}$$

GBM Algorithm

Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$, number of iterations M.

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For m = 1 to M:
    1. Compute so-called *pseudo-residuals*:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

    2. Fit a base learner (or weak learner, e.g. tree) closed under scaling $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.
    3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

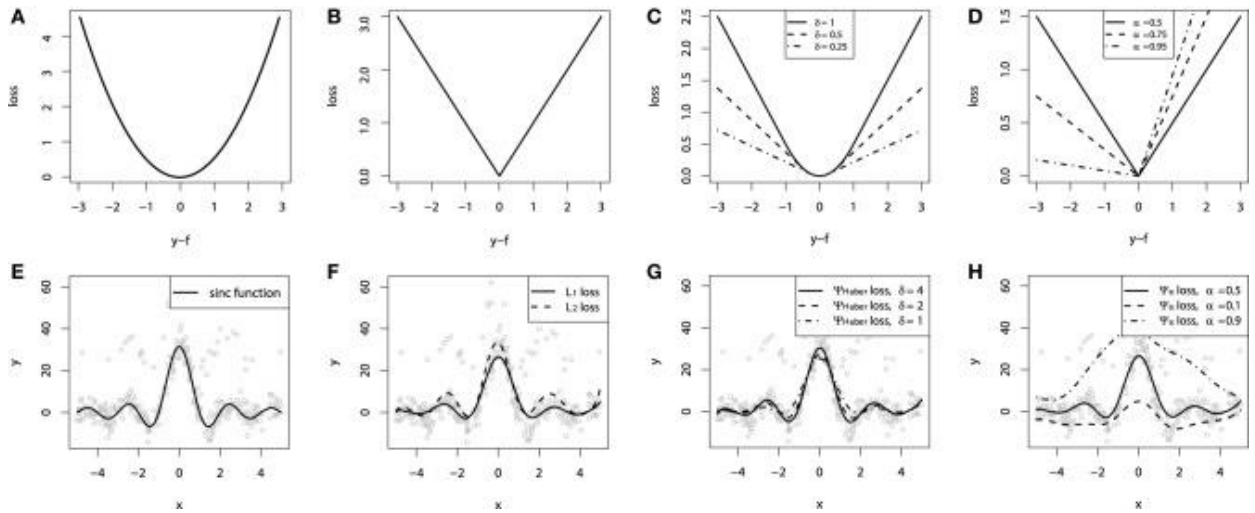
$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

    4. Update the model:

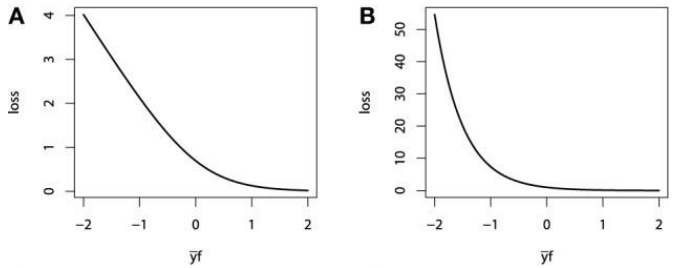
$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

다양한 Loss function을 이용한 경사 하강



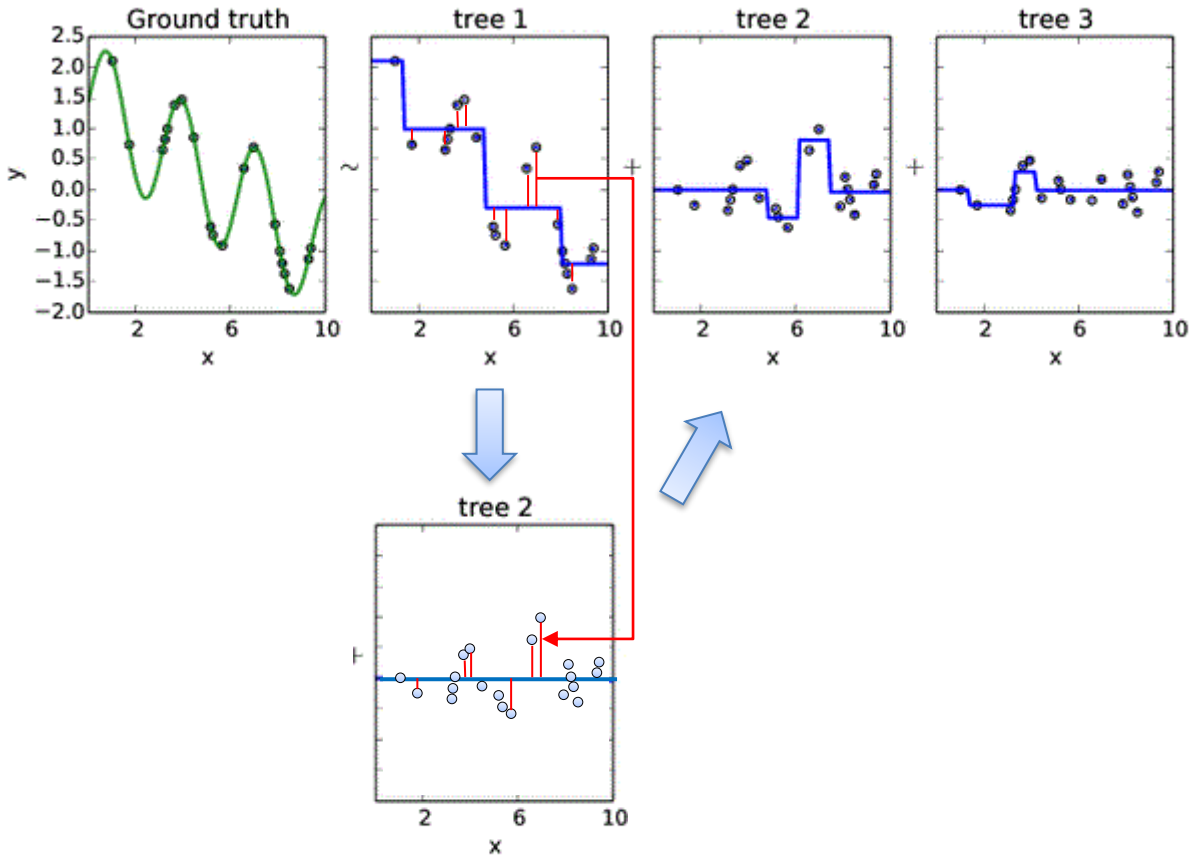
Continuous loss functions: (A) L2 squared loss function; (B) L1 absolute loss function; (C) Huber loss function; (D) Quantile loss function. Demonstration of fitting a smooth GBM to a noisy sinc(x) data: (E) original sinc(x) function; (F) smooth GBM fitted with L2 and L1 loss; (G) smooth GBM fitted with Huber loss with δ = {4, 2, 1}; (H) smooth GBM fitted with Quantile loss with α = {0.5, 0.1, 0.9}.



Categorical loss function : (A) Bernoulli loss function. (B) Adaboost loss function.

Gradient boosting machines, a tutorial
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3885826/#!po=18.5315

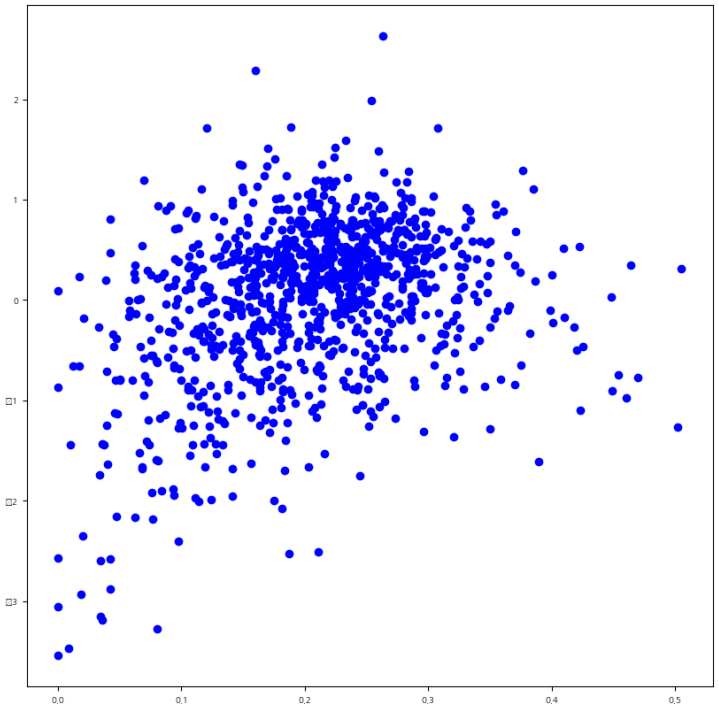
단순 평균값을 이용한 회귀 나무 모형 GBM



S-Dot 온도차이 vs 도로면적 비율

실제 데이터 1031개

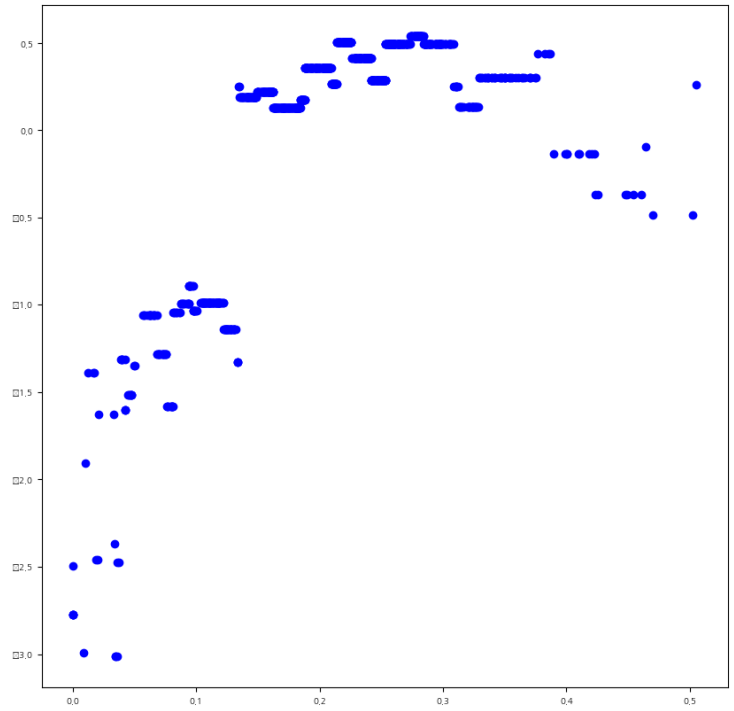
온도차이



도로비율

GBM 활용

온도차이



도로비율

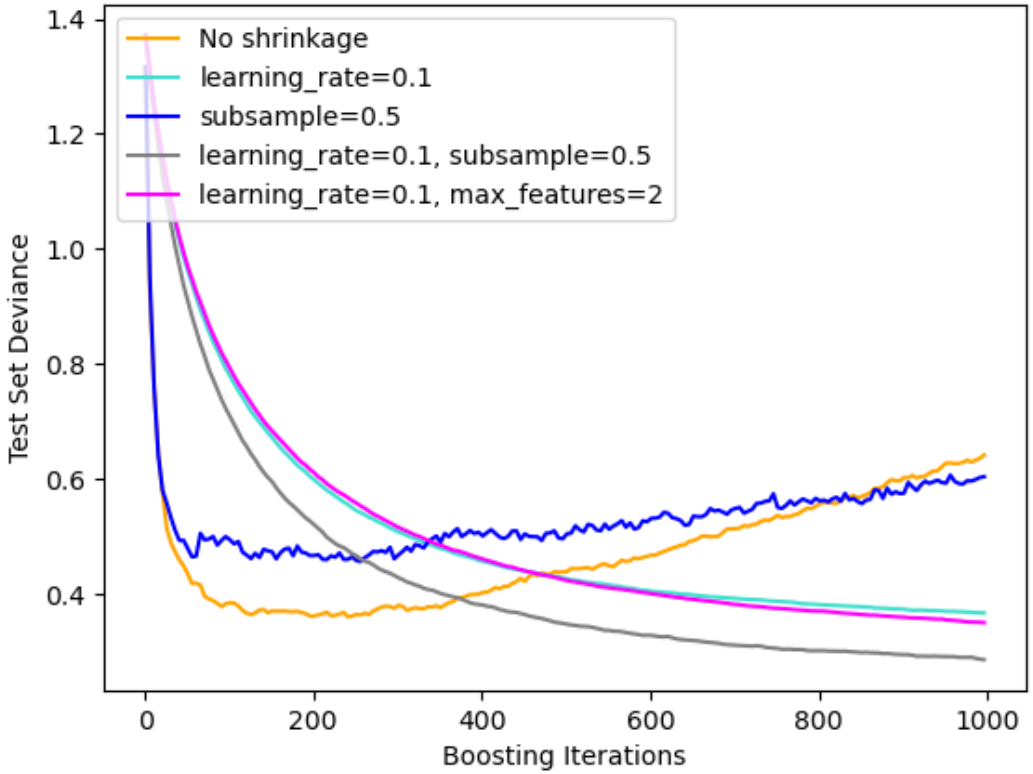
Learning Rate

경사도의 반대 방향으로 Loss 함수의 최소값을 향해 이동하더라도
이동하는 간격에 대한 고민이 필요.

Overfitting

GBM은 과적합 가능성이 높아짐

1. 과적합을 줄이기 위해 학습률을 통해 모델이 더해질수록 영향력을 shrinkage(수축)
2. Bagging 방식을 통해 Subsample을 이용해 학습 데이터의 과적합 제어
3. Random Forest처럼 Feature subsampling을 통해 변수 과적합 제어 (max_features)



XGBoost

2016년 대용량 데이터에서 GBM등의 알고리즘을 빠르게 병렬 수행이 가능하도록 만들어진 오픈소스

https://xgboost.readthedocs.io/en/latest/index.html

https://github.com/dmlc/xgboost/releases/tag/v1.4.0

Light GBM

2016년 MS에서 대용량 GBM 등의 알고리즘을 빠르게 병렬 수행이 가능하도록 만들어진 프레임워크

https://www.microsoft.com/en-us/research/project/lightgbm/

https://github.com/microsoft/LightGBM

Catboost

2017년 러시아 Yandex에서 카테고리컬 데이터의 Gradient Boost를 지원하는 오픈소스

https://catboost.ai/

https://github.com/catboost/catboost

1. Adaboost
2. GBM
3. **종합**

Dtree , Bagging, Boosting 모델 동시 학습

```
#%%

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import confusion_matrix
from sklearn.inspection import permutation_importance

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

분석 결과 종합

구분	Dtree	Bagging Tree	Random Forest	AdaBoost	GBM
학습시간	1.34	3.22	0.46	0.67	1.34
Train ACC	95.98%	100%	100%	89.65%	95.08%
Test ACC	69.76%	68.60%	72.86%	73.64%	69.76%