

---



# ML SESSION

#6 Boosting

---

# INDEX

1<sup>st</sup> Bagging

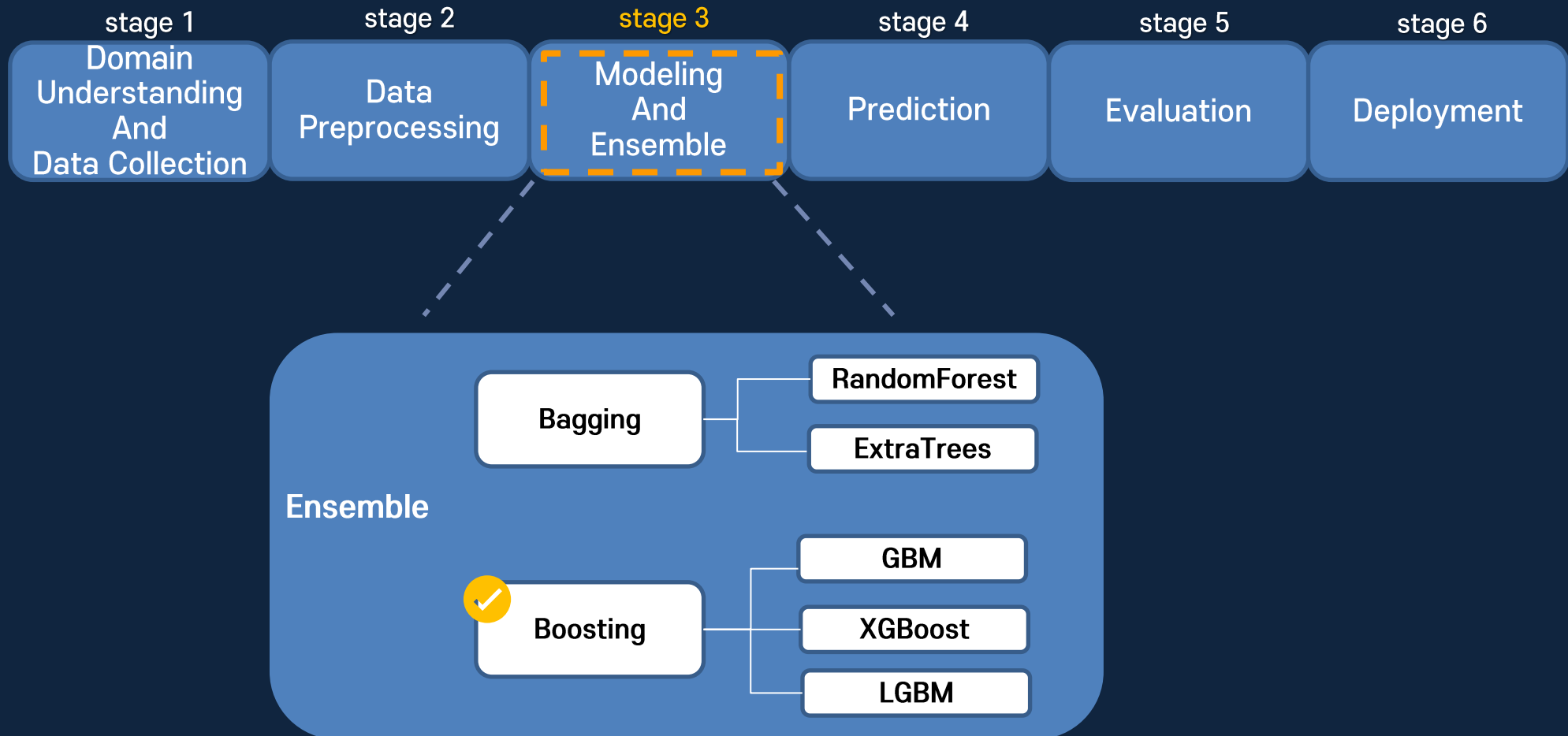
2<sup>nd</sup> Boosting

3<sup>rd</sup> GBM

4<sup>rd</sup> XGB

5<sup>th</sup> LGBM

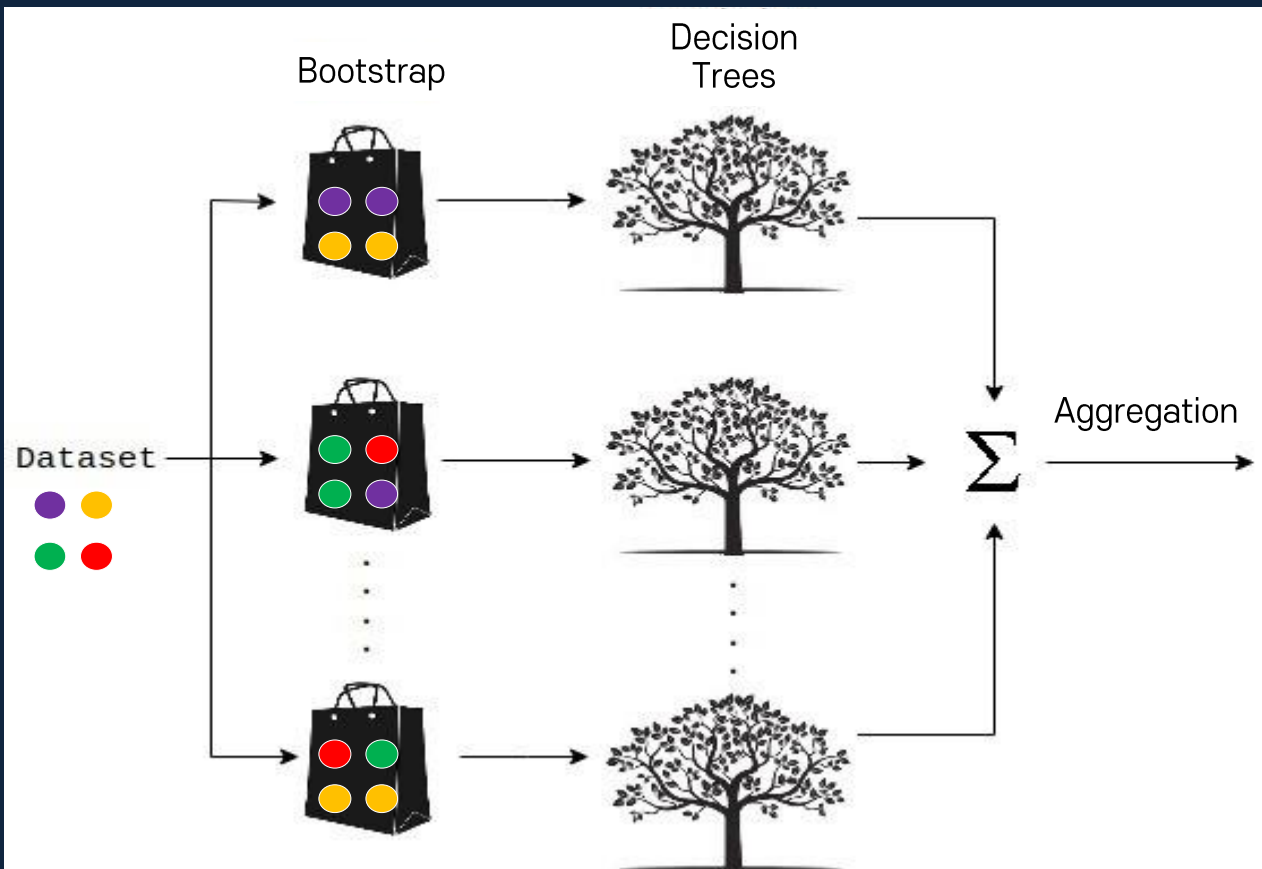
# 0 ML FLOW



# 1 Bagging

## Bagging = Bootstrap + Aggregating

※ bootstrap : 통계학에서는 **중복을 허용한 리샘플링(resampling)**을 부트스트래핑(bootstrapping)이라고 함



분류의 경우 : 최빈값

회귀의 경우 : 평균값





- ① Row data에서 bootstrap 데이터 추출
- ② 추출을 반복하여 n개의 데이터 생성
- ③ 각 데이터를 각각 모델링 하여 모델 생성
- ④ 단일 모델들의 예측 값을 결합

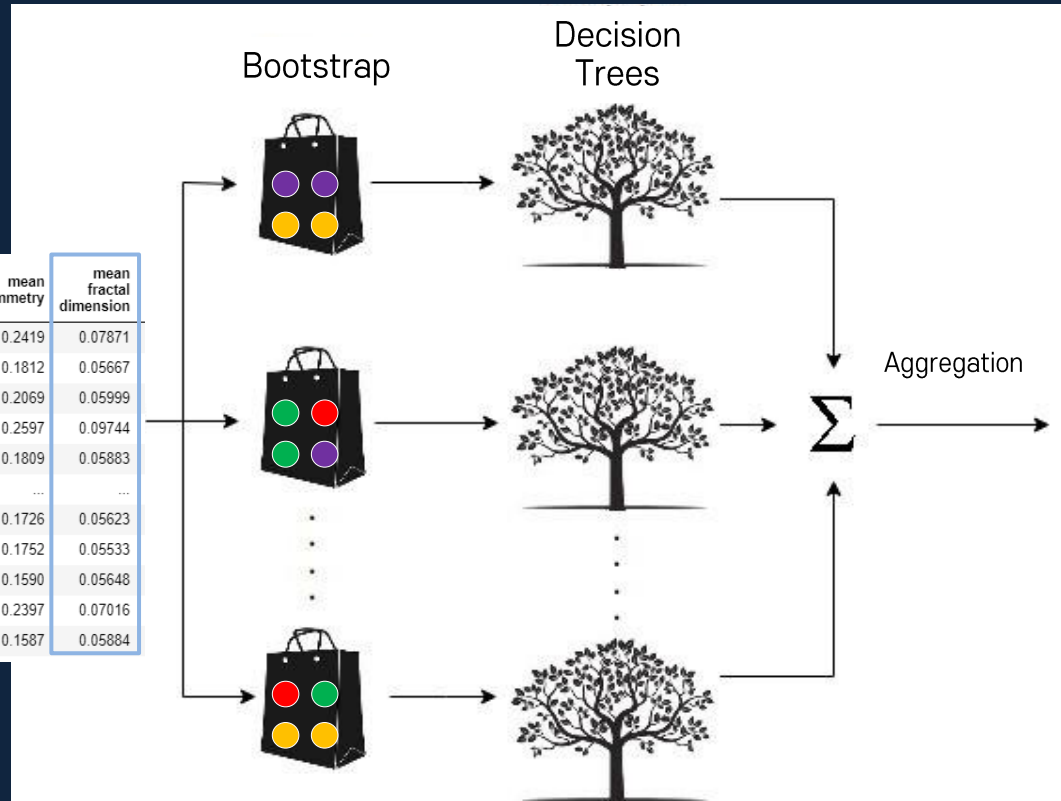
# 1 Bagging

## RandomForest

'max\_features'인자를 통한 변수 무작위성

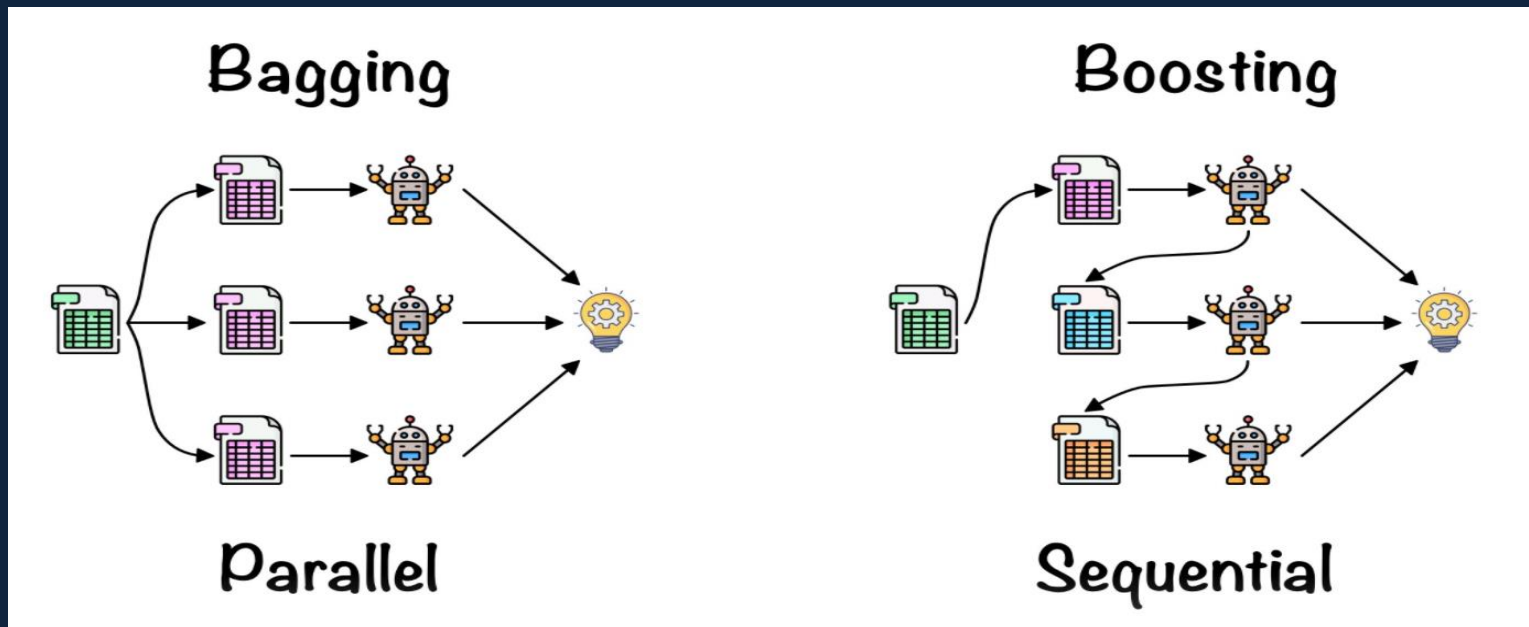
Dataset

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension
	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871
	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667
	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999
	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883
...	...	...	...	...	...	...	...	...	...	...
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884



# 2 Boosting

## Bagging vs Boosting



Bagging	Boosting
병렬적	순차적
모델들이 학습 시 상호 영향을 주지 않음	학습 시 앞의 모델의 결과가 뒤에 모델에 영향을 줌



면접 단골문제

# 2 Boosting

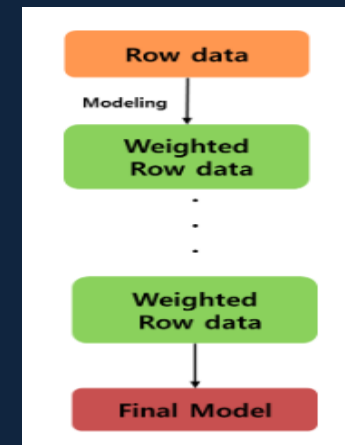
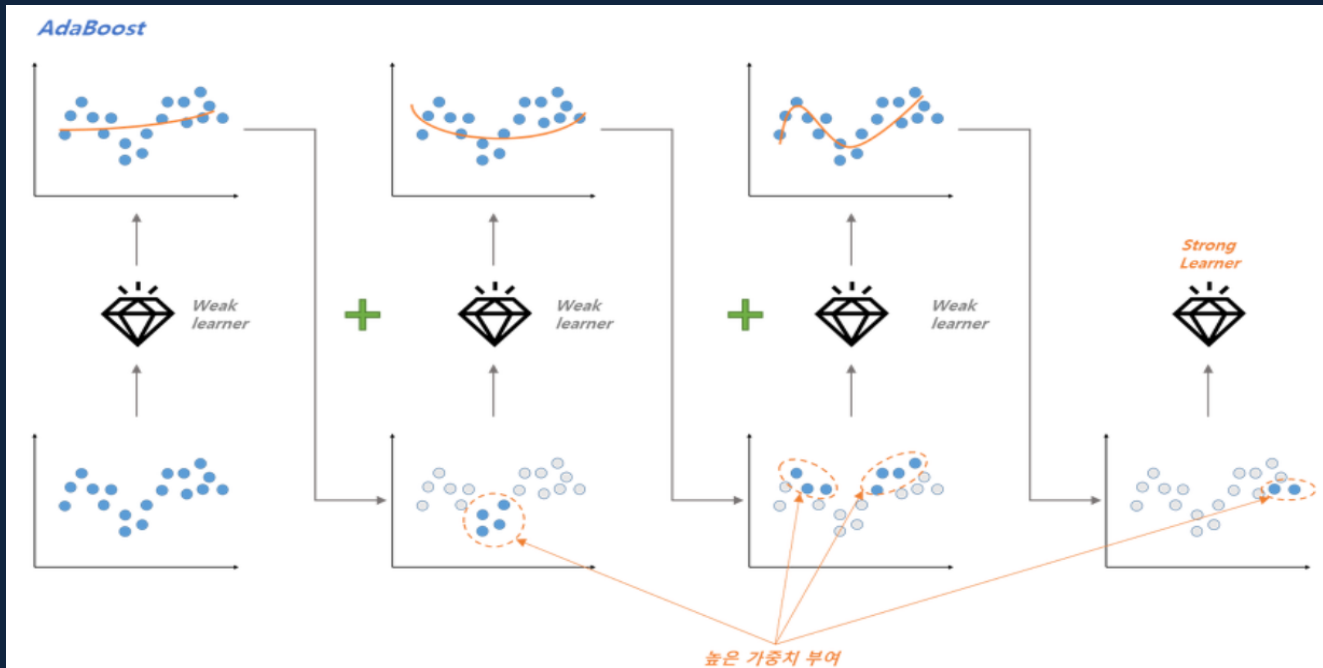
## Boosting이란?

약한 학습기를 여러 개 연결하여 강한 학습기를 만드는 앙상블 방법

학습된 모델을 보완해 나가면서 더 나은 모델로 학습시켜가는 방법

종류 : AdaBoost, GradientBoost

AdaBoost: 이전 분류기의 학습 결과를 토대로 다음 분류기의 **학습 데이터의 샘플가중치**를 조정해 학습을 진행하는 방법



- ① Row data에 동일가중치로 모델 생성
- ② 생성된 모델로 인한 오분류 데이터 수집
- ③ 오분류 데이터에 높은 가중치 부여
- ④ 과정 반복을 통하여 모델의 정확도 향상

Break Time

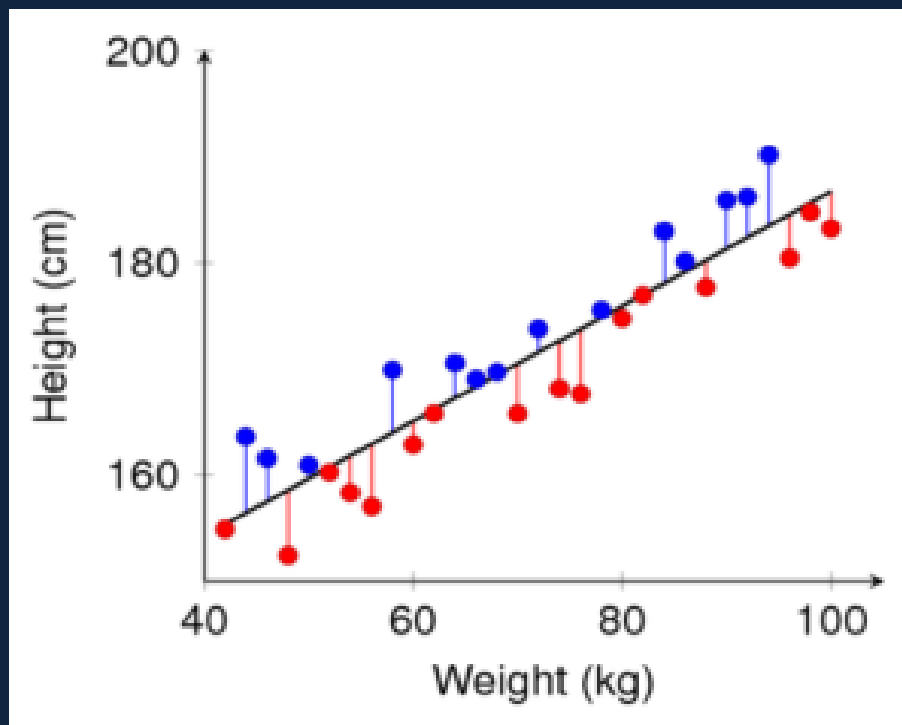




# 3 GBM

Gradient Boosting = Gradient Descent + Boosting

※ Gradient Boosting은 잔차를 활용하여 다음 학습기에 영향을 미친다



$$\hat{y} = f_1(x)$$

$$y - \hat{y} = y - f_1(x)$$

(residual)

$$y - f_1(x) = r_1 = f_2(x)$$

$$y - f_1(x) - f_2(x) = r_2 = f_3(x)$$

• Main idea

Original Dataset

$x^1$	$y^1$
$x^2$	$y^2$
$x^3$	$y^3$
$x^4$	$y^4$
$x^5$	$y^5$
$x^6$	$y^6$
$x^7$	$y^7$
$x^8$	$y^8$
$x^9$	$y^9$
$x^{10}$	$y^{10}$

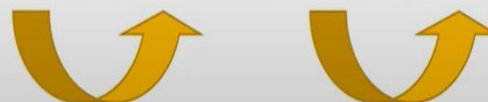
Modified Dataset 1

$x^1$	$y^1 - f_1(x^1)$
$x^2$	$y^2 - f_1(x^2)$
$x^3$	$y^3 - f_1(x^3)$
$x^4$	$y^4 - f_1(x^4)$
$x^5$	$y^5 - f_1(x^5)$
$x^6$	$y^6 - f_1(x^6)$
$x^7$	$y^7 - f_1(x^7)$
$x^8$	$y^8 - f_1(x^8)$
$x^9$	$y^9 - f_1(x^9)$
$x^{10}$	$y^{10} - f_1(x^{10})$

Modified Dataset 2

$x^1$	$y^1 - f_1(x^1) - f_2(x^1)$
$x^2$	$y^2 - f_1(x^2) - f_2(x^2)$
$x^3$	$y^3 - f_1(x^3) - f_2(x^3)$
$x^4$	$y^4 - f_1(x^4) - f_2(x^4)$
$x^5$	$y^5 - f_1(x^5) - f_2(x^5)$
$x^6$	$y^6 - f_1(x^6) - f_2(x^6)$
$x^7$	$y^7 - f_1(x^7) - f_2(x^7)$
$x^8$	$y^8 - f_1(x^8) - f_2(x^8)$
$x^9$	$y^9 - f_1(x^9) - f_2(x^9)$
$x^{10}$	$y^{10} - f_1(x^{10}) - f_2(x^{10})$

...



$$\hat{y} = f_1(x) \quad \hat{y} - f_1(x) = f_2(x) \quad \hat{y} - f_1(x) - f_2(x) = f_3(x)$$

## 3

## GBM

## Gradient Descent

$$W := W - \alpha \frac{\partial}{\partial W} \text{cost}(W, b) \quad * \alpha = \text{학습률 (learning rate)}$$

- How is this idea related to the gradient?

- ✓ Loss function of the ordinary least square (OLS)

$$\min L = \frac{1}{2} \sum_{i=1}^n (y_i - f(\mathbf{x}_i))^2$$

- ✓ Gradient of the Loss function  $\partial L = (f(x_i) - y_i) \partial f(x_i)$

$$\frac{\partial L}{\partial f(\mathbf{x}_i)} = f(\mathbf{x}_i) - y_i$$

- ✓ Residuals are the negative gradient of the loss function

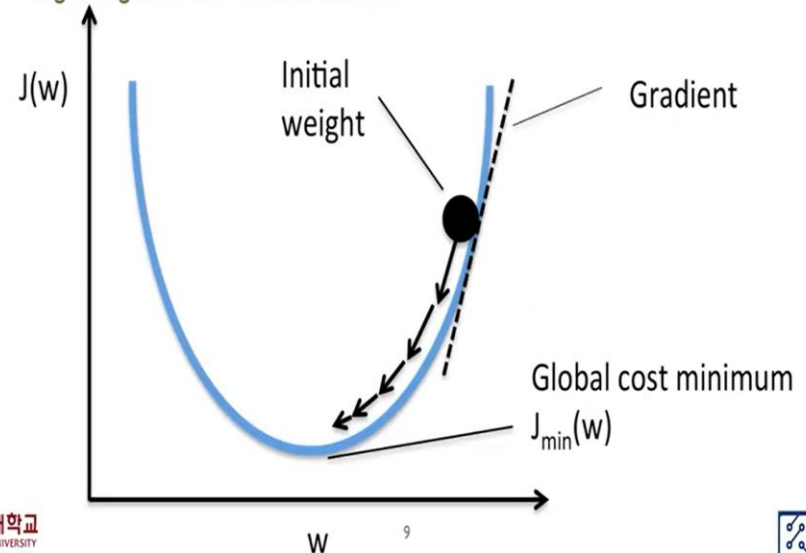
$$y_i - f(\mathbf{x}_i) = -\frac{\partial L}{\partial f(\mathbf{x}_i)}$$

- Gradient Descent Algorithm

- ✓ Blue line: value of loss function with a given parameter

- ✓ Black point: current state

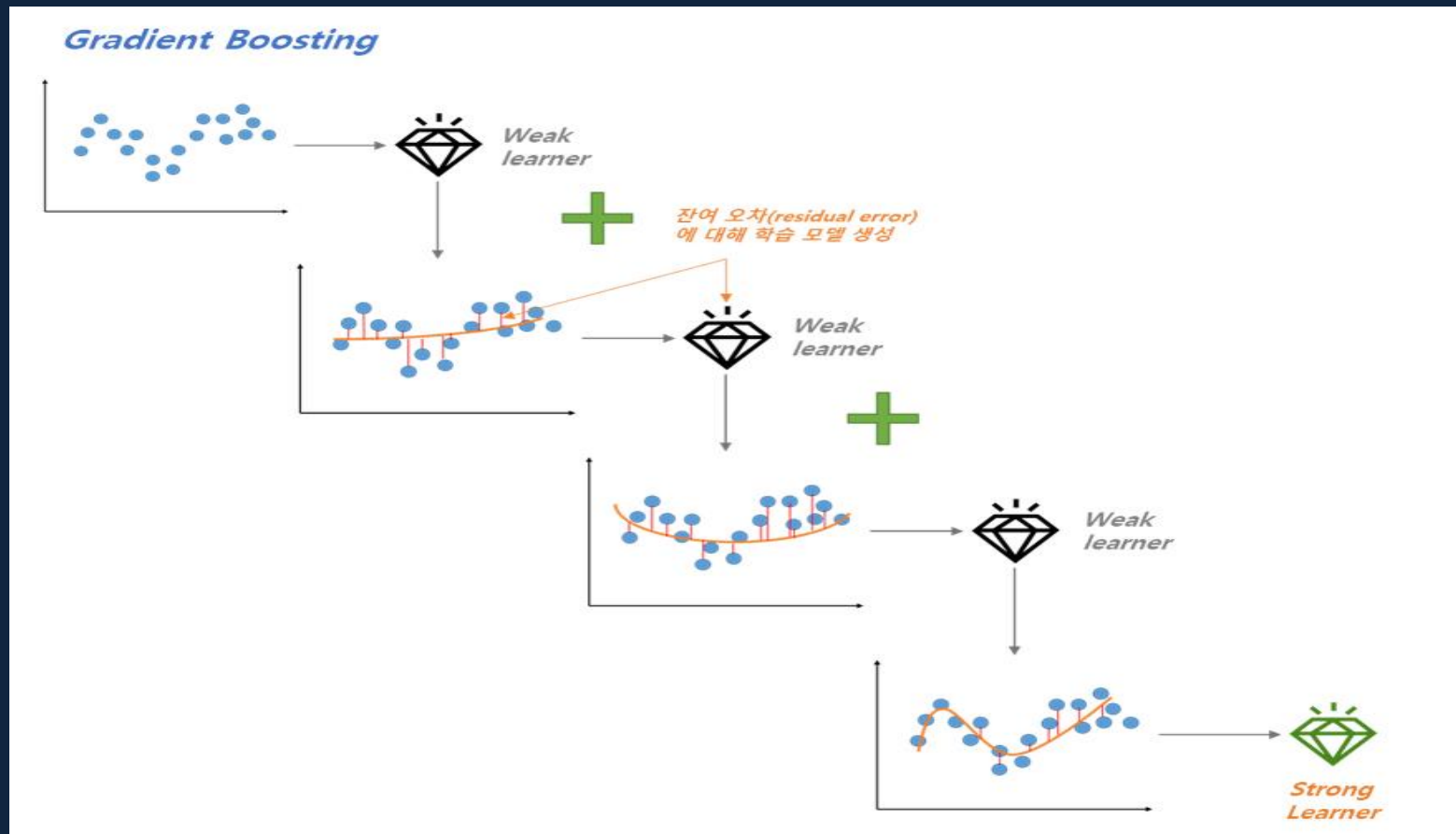
- ✓ Arrows: the direction that the parameter should follow to minimize the loss function  
= negative gradient of the loss function



## 3

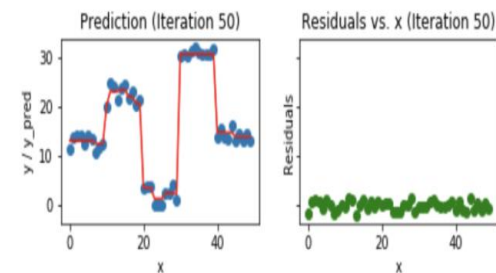
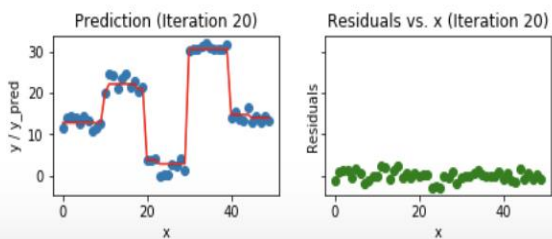
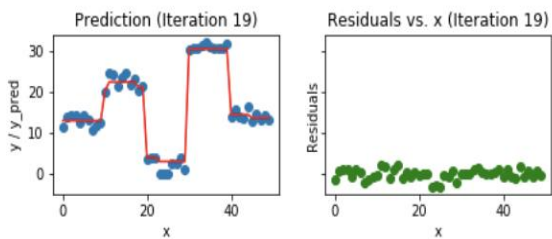
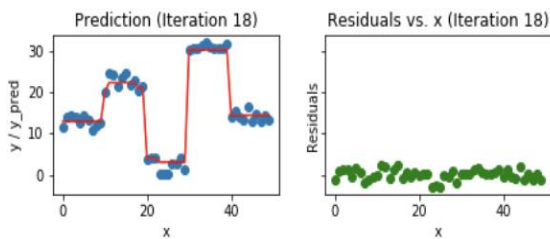
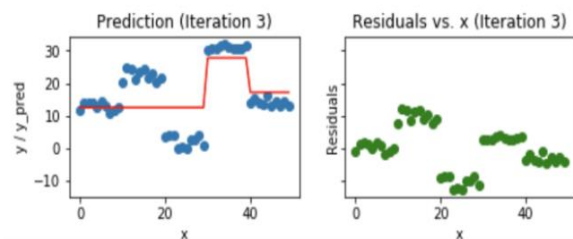
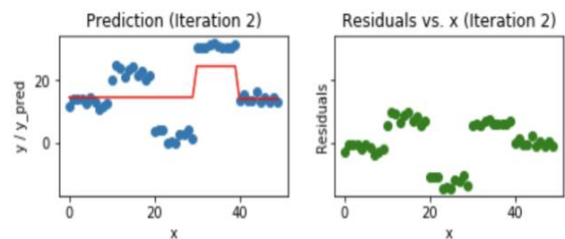
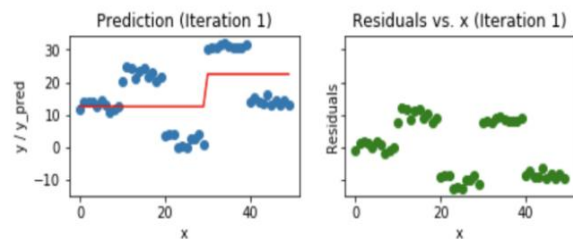
## GBM

## Gradient Boosting Machine



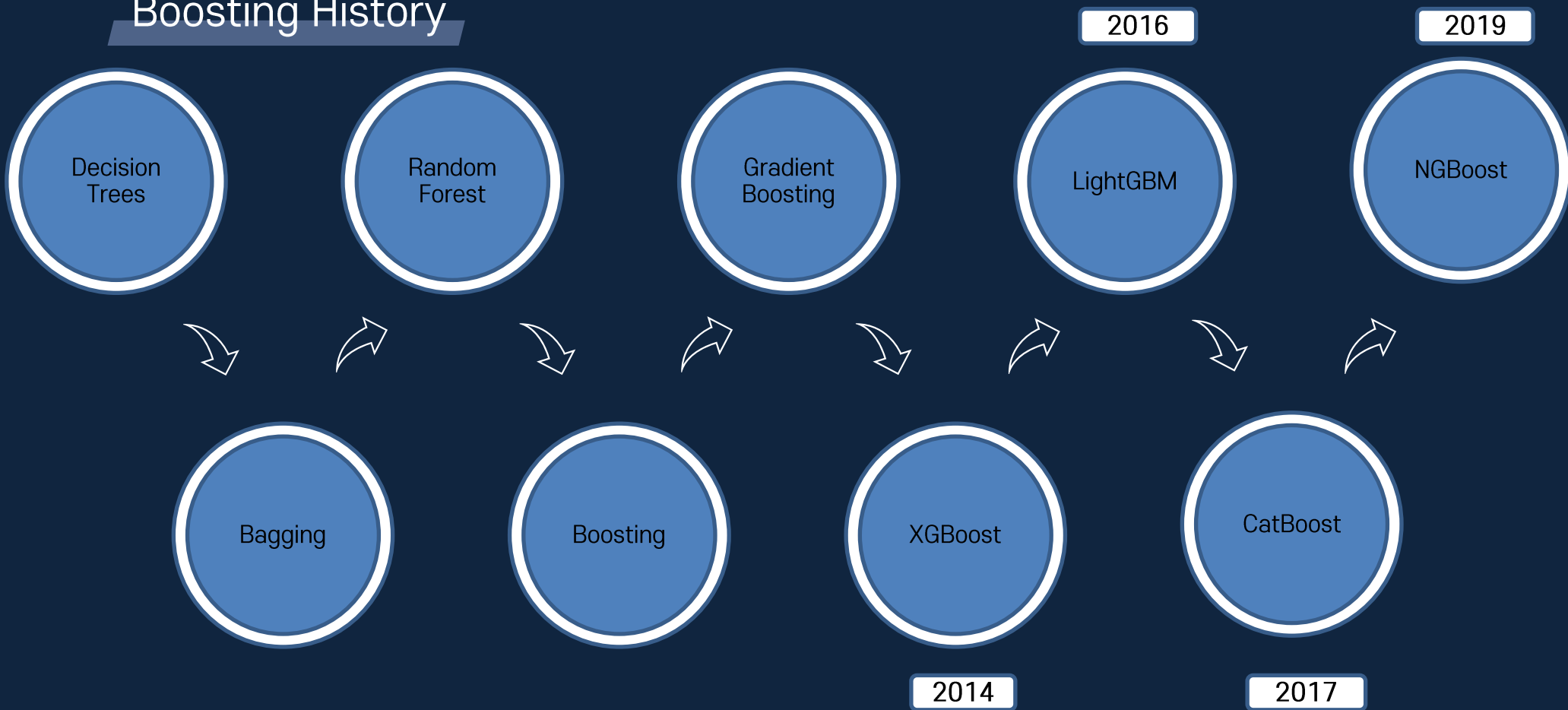
# 3 GBM

## GBM Example



# 4 XGB

## Boosting History



# 4 XGB

## eXtreme Gradient Boost

학습속도 *up* 성능 *up*

### Features of XGBoost

- Regularized boosting (prevents overfitting) → 정규화
- Can handle missing values automatically → Missing Value
- Parallel processing → 병렬처리
- Can cross-validate at each iteration → CV, 조기종료
  - Enables early stopping, finding optimal number of iterations
- Incremental training → 연이어 추가 학습
- Can plug in your own optimization objectives → Objective 설정
- Tree pruning → 가지치기
  - Generally results in deeper, but optimized, trees

# 4 XGB

## XGB Parameters



종류	파라미터명	default값	설명
부스터 파라미터	learning_rate	0.1	- 학습률 - 범위: 0 ~ 1
	n_estimators	100	- 생성할 약한 학습기 개수
	max_depth	3	- 트리의 최대 깊이 - 보통 3~10 사이의 값 적용
	min_child_weight	1	- 트리에서 추가적으로 가지를 나눌지 결정하기 위해 필요한 데이터들의 가중치 총합 - 과적합 조절 용도 - 범위: 0 ~ Inf
	gamma	0	- 트리의 리프 노드를 추가적으로 나눌지 결정할 최소 손실 감소값 - 해당 값보다 큰 손실이 감소된 경우에 리프 노드 분리 - 값이 클수록 과적합 방지
	early_stopping_rounds	None	- 조기 종단을 위한 반복 횟수. N번 반복하는 동안 성능 평가 지표가 향상되지 않으면 반복이 멈춤
	subsample	1	- 트리가 커져 과적합되는 것을 제어하기 위해 데이터를 샘플링하는 비율을 지정 - sub_sample = 0.5면 전체 데이터의 절반을 트리 생성에 사용 - 보통 0.5 ~ 1 사이의 값 적용 - 범위: 0 ~ 1
	colsample_bytree	1	- 트리 생성에 필요한 피처를 임의로 샘플링
	reg_lambda	1	- L2 Regularization 적용 값
	reg_alpha	0	- L1 Regularization 적용 값
학습 태스크 파라미터	scale_pos_weight	1	- 특정값으로 치우친 비대칭한 클래스로 구성된 데이터 세트의 균형을 유지하기 위한 파라미터
	objective	reg:linear	손실함수 - binary:logistic: 이진 분류, 확률 반환 - multi:softmax: 다중 분류, num_class 파라미터 지정 필요 - multi:softprob: 개별 레이블 클래스에 해당하는 예측 확률 반환
	eval_metric	- 회귀: rmse - 분류: error	검증에 사용하는 함수 정의 - rmse: Root Mean Square Error - mae: Mean Absolute Error - logloss: Negative log-likelihood - merror: Multiclass classification error rate - mlogloss: Multiclass logloss - auc: Area under the curve
	eval_set	None	- 성능 평가에 사용하는 데이터세트 (evaluation set)

5

# LGBM

LightGBM

GBM

속도 느림

오버피팅 가능성

XGB

속도 보통

오버피팅 규제

LGB

속도 빠름

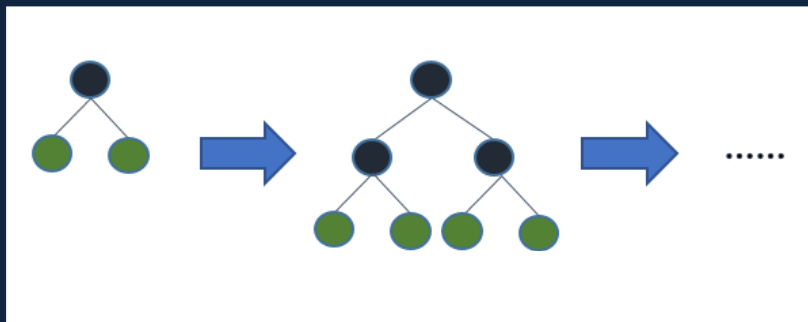
적으면 오버피팅



# 5 LGBM

## XGB vs LGBM

Level – wise tree

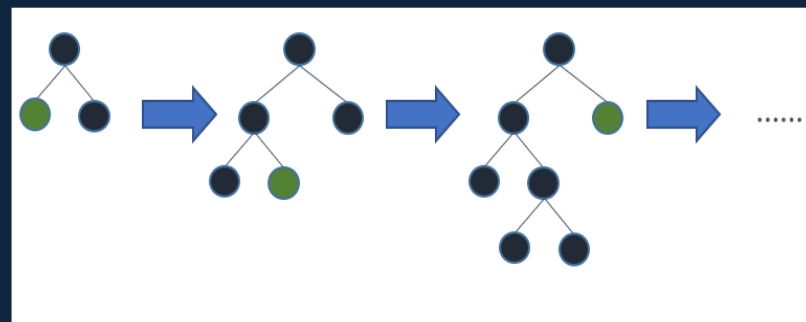


Level – wise tree : 균형 트리 분할

사용 모델 : RF, XGB

성장 방법 : 균형을 맞춤, 연산추가, **수평성장**

Leaf – wise tree



Leaf – wise tree : 리프 중심 트리 분할

사용 모델 : LGBM

성장 방법 : 균형 맞추지 않음, 손실 값이 큰 리프 노드에서 분할, **수직성장**

# 5 LGBM

## LGBM 특징

- 속도가 빠르고, 성능이 좋다.
- 저장 공간을 덜 차지한다.
- 병렬적인 학습(동시적)이 가능하다.
- Overfitting에 민감하다.

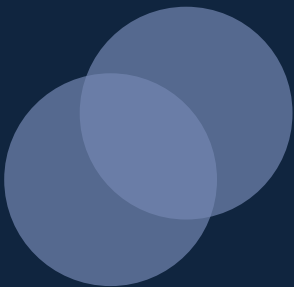


대용량 data에 적합  
(적어도 10,000건 이상)

# 5 LGBM

## LGBM Parameters

종류	파라미터명	default값	설명
부스터 파라미터	learning_rate	0.1	- 학습률 - 범위: 0 ~ 1
	n_estimators	100	- 생성할 약한 학습기 개수
	max_depth	3	- 트리의 최대 깊이 - 보통 3~10 사이의 값 적용
	min_child_samples	20	- 최종 리프 노드가 되기 위해 최소한으로 필요한 레코드 수 - 과적합 제어를 위한 파라미터
	num_leaves	31	- 하나의 트리가 가질 수 있는 최대 리프 개수
	subsample	1	- 트리가 커져 과적합되는 것을 제어하기 위해 데이터를 샘플링하는 비율을 지정 - sub_sample = 0.5면 전체 데이터의 절반을 트리 생성에 사용 - 보통 0.5 ~ 1 사이의 값 적용 - 범위: 0 ~ 1
	colsample_bytree	1	- 트리 생성에 필요한 피처를 임의로 샘플링
	reg_lambda	1	- L2 Regularization 적용 값
	reg_alpha	0	- L1 Regularization 적용 값
학습 태스크 파라미터	objective	reg:linear	손실함수 - binary:logistic: 이진 분류, 확률 반환 - multi:softmax: 다중 분류, num_class 파라미터 지정 필요 - multi:softprob: 개별 레이블 클래스에 해당하는 예측 확률 반환
	eval_metric	- 회귀: rmse - 분류: error	검증에 사용하는 함수 정의 - rmse: Root Mean Square Error - mae: Mean Absolute Error - logloss: Negative log-likelihood - merror: Multiclass classification error rate - mlogloss: Multiclass logloss - auc: Area under the curve
	eval_set	None	- 성능 평가에 사용하는 데이터세트 (evaluation set)



# 과제

첫번째  
Baseline을 넘어라 !

THANK YOU

