16기 정규세션

ToBig's 15기 강의자

이성범

Ensemble

앙상블

Contents

Unit 01 | Intro

Unit 02 | Voting

Unit 03 | Bagging

Unit 04 | Boosting

Unit 05 | Stacking

01. Intro

No Free Lunch Theorem

We have dubbed the associated results “No Free Lunch” theorems
because they demonstrate that if an algorithm performs
well on a certain class of problems
then it necessarily pays for that with degraded performance
on the set of all remaining problems.

No Free Lunch Theorem은 간단히 말해 특정한 문제에 최적화된 알고리즘은
다른 문제에서는 그렇지 않다는 것을 수학적으로 증명한 이론이다.

-> **즉 경험하기 전에 더 좋다고 할 수 있는 모델은 없다!**

따라서 다양한 모델을 실험해보고

최적의 모델을 찾아야 한다!

그런데 사람도 각자의 장점이 있는데
모델도 모델마다 장점이 있지 않을까?

그러면 다양한 모델의 장점을 하나로 묶어서 예측을 해보자!

그것이 바로

Ensemble

Ensemble 이란?

개념

- 앙상블 학습은 **여러 개의 모델을 결합**하여 하나의 모델보다 더 좋은 성능을 내는 머신러닝 기법
- 앙상블 학습의 핵심은 **여러 개의 약 분류기(Weak Classifier)를 결합**하여 **강 분류기(Strong Classifier)로 만드는 것**

기대 효과

- 과적합 및 과소적합 방지
- 분산과 편향 감소
- 모델 성능 향상

한마디로 앙상블은

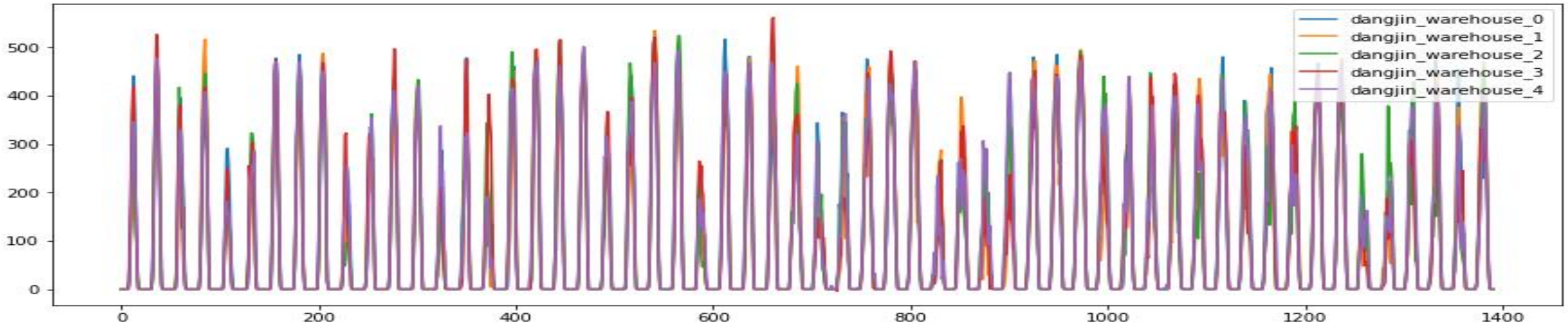
집단지성을 활용하는 방법!

하지만 **집단지성(앙상블)**이

무조건 **한 명의 완벽한 전문가(개별 모델)**를

이긴다는 보장은 없다.

즉, NFL 이론에 따라서 다양한 실험을 해봐야 한다!



DACON 동서 발전 태양광 발전량 예측 AI 경진대회 에서 5개의 모델을 앙상블 하여 6위의 성적

Kaggle, DACON 등에서 앙상블을 활용해 좋은 성적을 많이 낸다.

Ensemble은 왜 높은 성능을 가져다 줄까?

가정

- 앞면(정답)이 51%, 뒷면(오답)이 49%가 나오는 동전이 있다고 가정

실험 (각 시행은 서로 독립)

- 동전을 1000번 던진다면 앞면이 510번, 뒷면은 490번이 나온다. (대수의 법칙)
- 하지만 뒷면이 더 많이 나올 수도 있다.
- 이항분포의 확률 질량 함수로 계산을 하면 1,000번을 던진 후 앞면이 다수가 될 확률은 약 75%, 10,000번을 던진 후 앞면이 다수가 될 확률은 약 97%
- **즉, 시행 횟수를 증가시킬수록 앞면이 다수가 될 확률은 점점 더 커진다!**

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

시행 횟수를 모델의 개수로

앞면이 다수가 될 확률을 앙상블 모형의 정확도로 본다면

Ensemble이 왜 성능이 높은지 알 수 있다!

Unit 01 | Intro

우리는 **동전을 던질 때 각각의 시행은 독립적이라는 가정**을 세웠다.

따라서 Ensemble을 통하여 좋은 성능을 내기 위해서는

모든 **분류기가 완벽하게 독립이고 오차에 상관관계가 없어야** 한다.

즉, **독립적인 다양한 분류기**를 얻기 위해서는

각기 다른 알고리즘으로 또는 **다른 데이터 셋**으로 학습을 시켜야 한다.

-> 이는 모든 Ensemble 방법에 적용!

앙상블을 배우기전

Decision Tree, 분산, 편향

에 대하여 알고 들어가자!

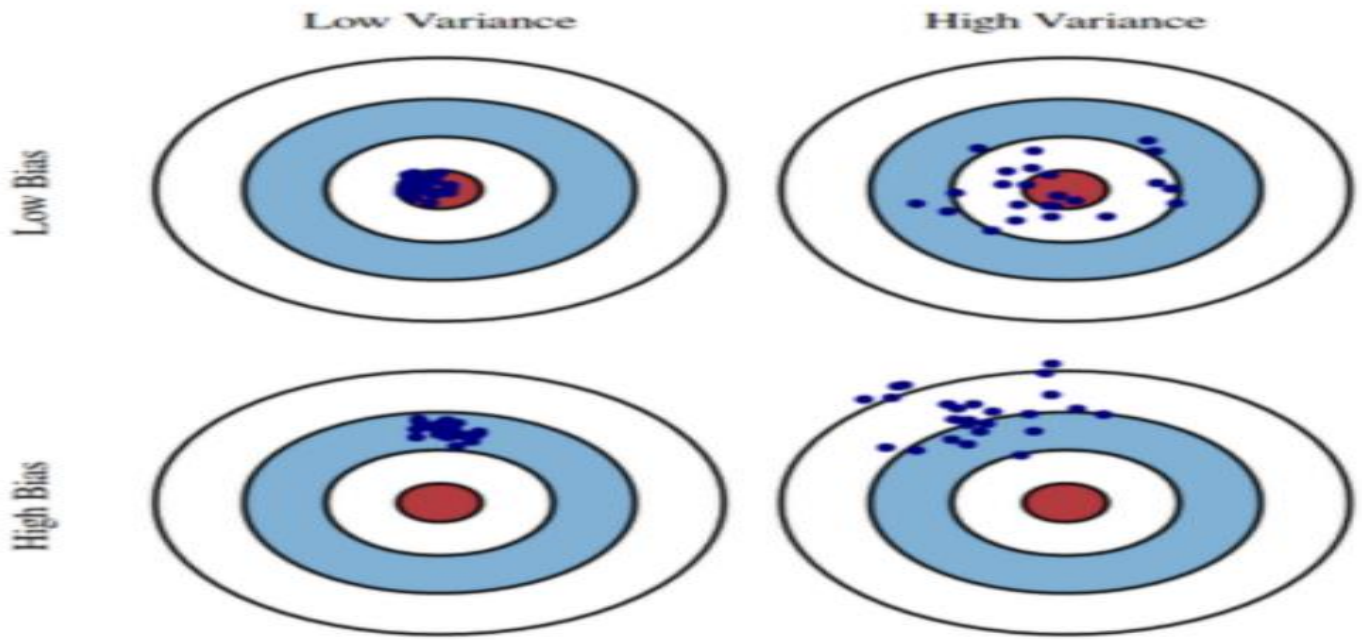
Decision Tree는 앙상블의 Base Model로 많이 쓰인다!

왜? 분산과 편향을 조절하기 쉽다!

Ex) 모델의 깊이를 깊게 하면 분산이 높아지고

모델의 깊이를 얕게 하면 편향이 높아진다!

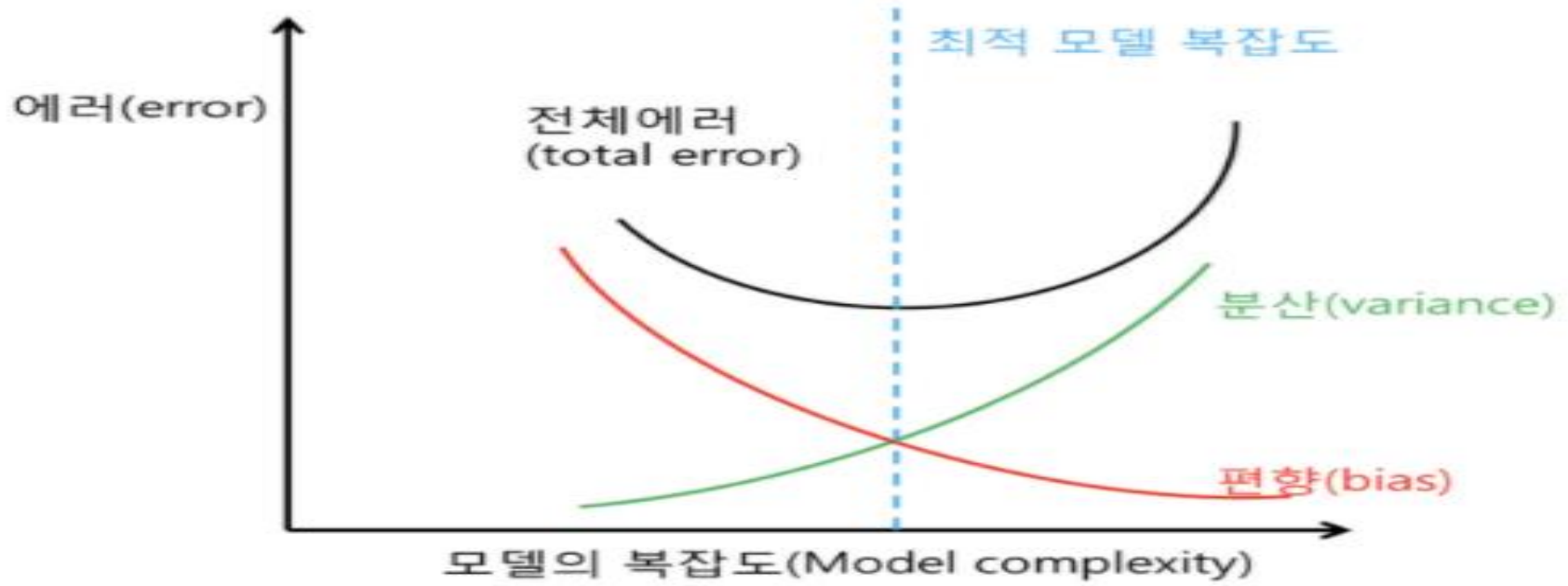
Unit 01 | Intro



분산은 예측의 변동폭이 얼마나 큰지를 반영

편향은 예측이 정답에서 얼마나 떨어져 있는지를 반영

Unit 01 | Intro



High Variance -> Overfitting(과적합)

High Bias -> Underfitting(과소적합)

High Variance or **High Bias Model**은 **Weak Classifier** 이라고 할 수 있다.

앙상블이란? **Weak Classifier을 결합**하여 **Strong Classifier**로 만드는 것!

근데 **DT는 Variance와 Bias를 조정하기 쉽다!**

따라서 **Ensemble의 Base Model**로 많이 쓰인다!

Voting <- Vanllila Ensemble

Bagging <- High Variance Model의 성능을 높이자!

Boosting <- High Bias Model의 성능을 높이자!

Stacking <- 예측을 취합하는 함수를 Model로 만들자!

02. Voting

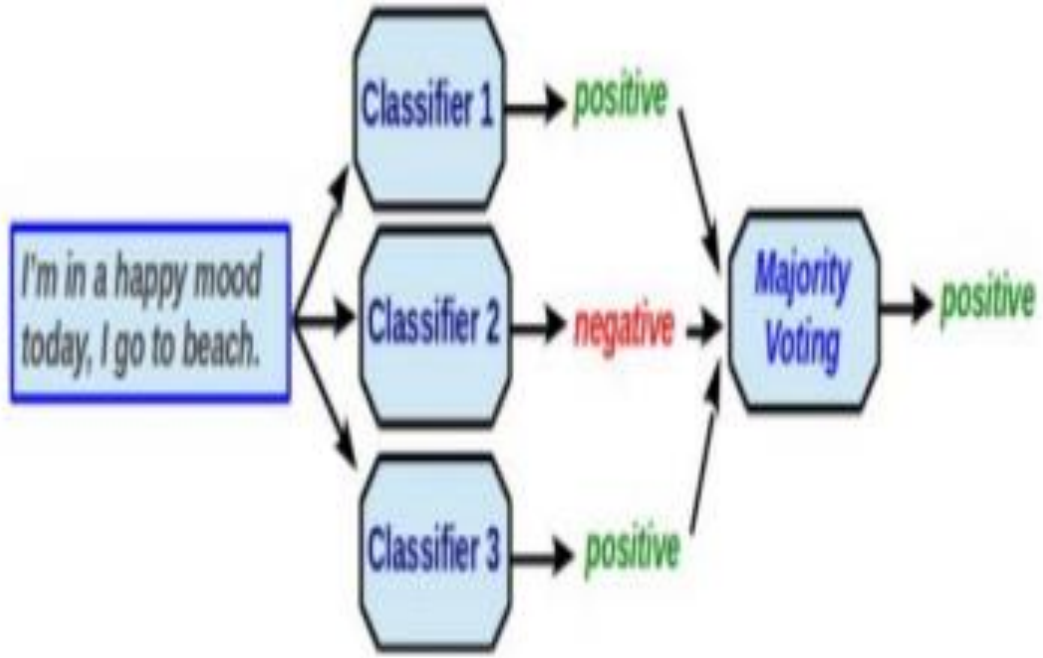
Voting 이란?

- 가장 기본적인 Ensemble 방법
- 여러 개의 **서로 다른 모델**을 결합하여 예측을 하는 방법 (2가지)

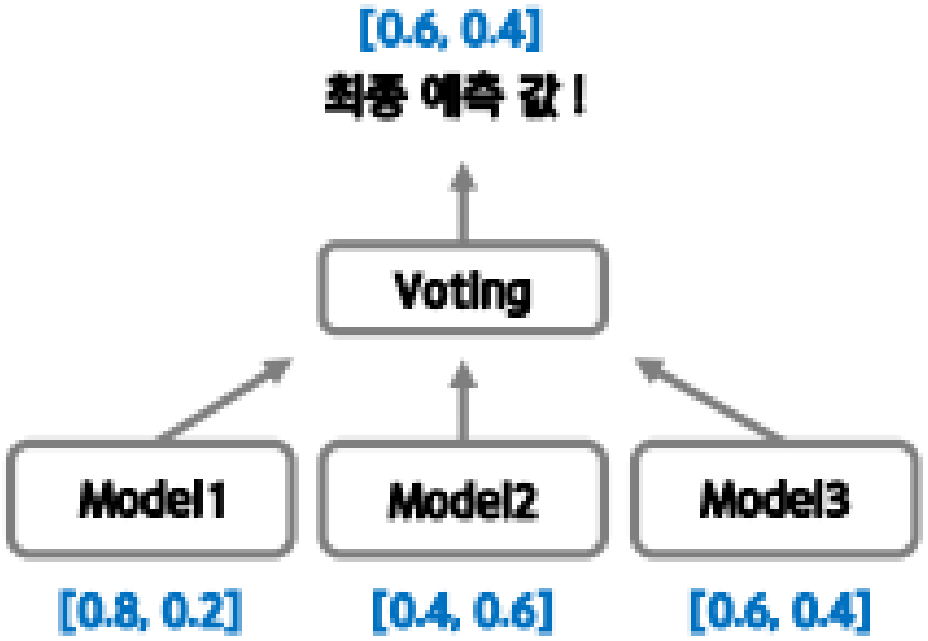
Hard Voting : 직접 투표, 다수결의 원칙

Soft Voting : 간접 투표, 확률(분류) 또는 값(회귀)의 평균

Voting 이란?



Hard Voting



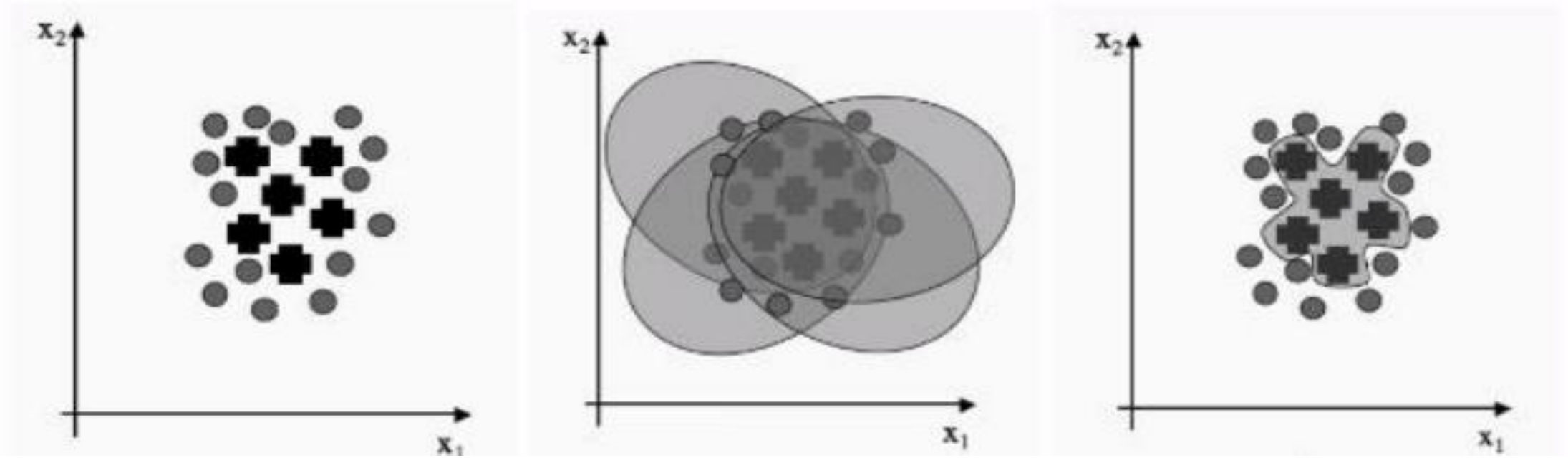
Soft Voting

03. Bagging

Bagging 이란?

- Bootstrap Aggregating의 줄임말
- Bootstrap의 Subset Sample로 **동일한 모델** N개를 학습하여 Voting하는 방법
- High Variance 모델에 적합
- 회귀(평균 or 중위수), 분류(다수결 or 평균)

왜 High Variance Model에 적합하고 분산이 감소될까?



서로 다른 Data의 Overfitting된 모델의 **평균**을 내면 Overfitting이 줄어들기 때문에!

Bagging을 왜 쓸까?

- 단순히 **같은 Dataset**으로 만든 **동일한 Classifier**는 **의미가 없다.**

  Ex) 같은 Dataset으로 만든 여러 개의 Tree는 매우 비슷하기 때문에 성능에 영향을 주지 않는다.

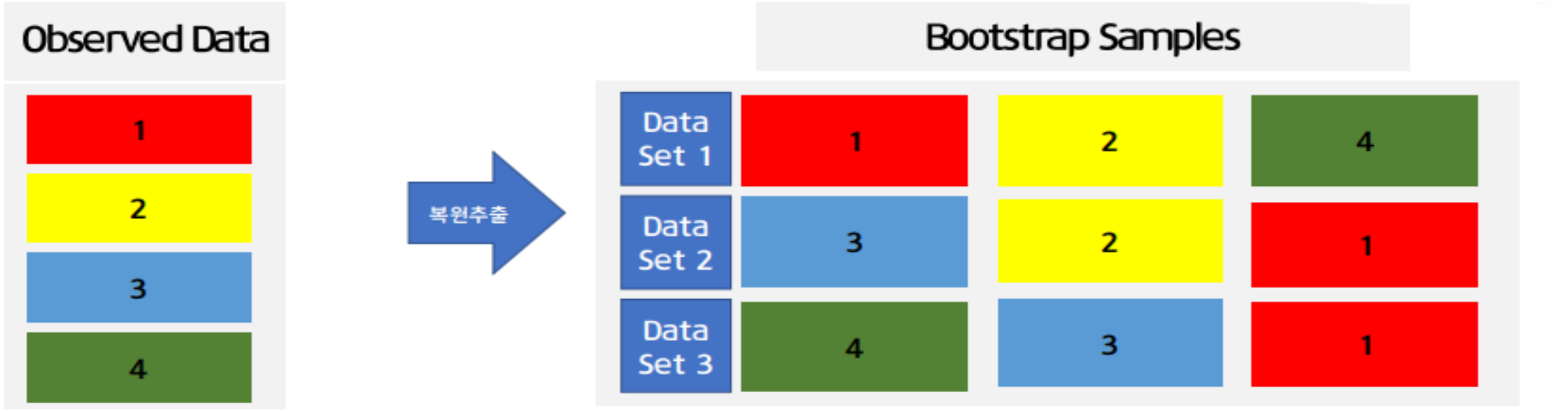
- 우리가 가지고 있는 데이터는 매우 큰 부분의 일부이다. 마찬가지로 우리가 갖고 있는 샘플 자체를 하나의 모집단이라고 생각하고 **샘플링을 한다면 다양한 데이터 셋**을 만들 수 있다.
- 즉, 다양한 Sampling Dataset으로 다양한 Classifier을 만들자!

-> **다양한 Sampling Dataset을 만드는 방법**이 바로 **Bootstrapping!**

-> **Bootstrapping을 활용**한 **다양한 Classifier**을 만드는 방법이 바로 **Bagging!**

Bootstrapping이란?

- **중복을 허용**하여 데이터를 샘플링하는 방법(복원 추출)
- 학습 데이터 Subset N개를 추출하자!



Bootstrap .632

- Bootstrapping을 통하여 데이터 셋을 추출하면 전체 데이터의 **63%**만 샘플링

한 데이터가 뽑힐 확률: $\frac{1}{d}$ **/ 뽑히지 않을 확률:** $1 - \frac{1}{d}$

적어도 한번 이상 뽑힐 확률:

$$1 - \prod(1 - \frac{1}{d}) = 1 - (1 - \frac{1}{d})^d \approx 1 - e^{-1}$$

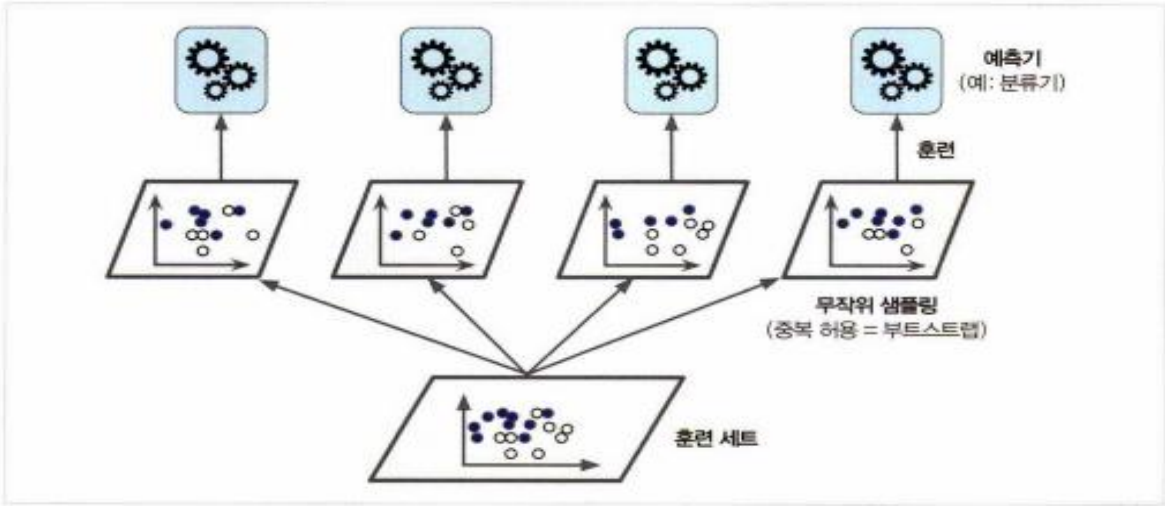
= 0.632

Out-of-Bag Error = OOB Error

- 샘플링이 되지 않은 **37%** Data를 OOB 샘플이라고 함
- Bagging 시 OOB 샘플을 통해서 모델의 성능을 평가
- Validation set, K-fold와 처리하는 방법이 유사
- OOB Error는 Bagging 성능 측정을 위한 굉장히 좋은 지표

정리

- Bootstrapping을 통해서 서로 다른 Sub Dataset이 만들어짐
- 즉, OOB 샘플도 달라진다는 것!
- 따라서 **다양한 Dataset으로 다양한 Classifier**를 만들 수 있음
- 단순히 다양한 데이터 셋으로 다양한 모델을 만들기 때문에 병렬처리가 가능
- 병렬 학습이 가능하여 모델의 학습 속도가 빠름!
- 분산을 낮춰 강건한 모델이 만들어짐!



번외

- 복원 추출이 아닌 비복원 추출을 통한 Data Sampling 방법을 Pasting 이라고 함
- Data의 feature(특성)도 샘플링 할 수 있음!
- Bootstrap X / Bootstrap_feature X 하는 것을 Random Patches Method
- Bootstrap X / Bootstrap_feature O 하는 것을 Random Subspaces Method
- **특성 샘플링 방식 또한 다양한 예측기를 만들며 분산을 낮춰주는 효과**를 가져다 줌
- 앞으로 다룰 RandomForest Model도 Hyperparameter Tunning을 통해서 특성 샘플링 가능

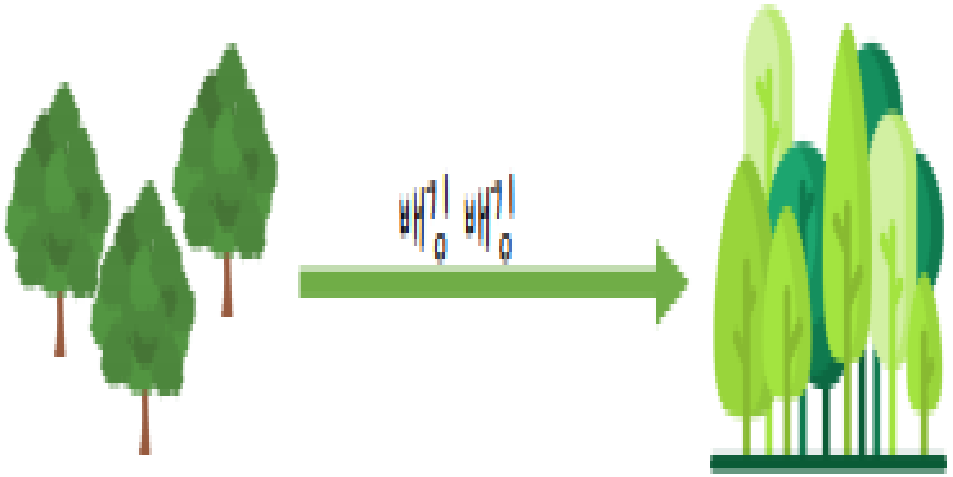
Bagging 방식을 활용한

Decision Tree Base의 대표적인 Ensemble Model이

Random Forest

RandomForest 이란?

- Bagging + Randomized Decision Tree
- **High Variance DT들의 Ensemble**
- 관측치에 비해 변수의 수가 많은 고차원 데이터에서 중요 변수 선택 기법으로 활용도가 높음



나무가 모여 숲을 이룬다

Main Idea : Diversity(다양성), Random(무작위)

- 성능은 좋지만 과적합이 심한 DT의 분산을 줄이자!

  -> **High Variance DT의 Ensemble!**

- 단순한 Ensemble은 의미가 없어(동전 던지기) 서로 다른 다양한 모델을 만들어야 한다!

  -> **Bagging 방식을 활용!**

- 만들어진 다양한 모델에서 더 다양한 모델을 만들 수 없을까?

  -> **feature도 랜덤하게 선택!**

Random Forest의 중요 파라미터

- n_estimators : DT의 개수
  - ✓기본 값은 100
- max_features : 노드를 나눌 때 선택 할 feature의 개수
  - ✓분류는 sqrt(n_features) / 회귀는 n_features 가 기본값
- max_depth : 뿌리의 깊이
  - ✓High Variance DT를 만들기 위해 기본 값은 None
- random_state : Seed 고정
  - ✓RandomForest 모델은 무작위성이 강조된 모델이기 때문에 학습 마다 동일한 모델을 만들기 위해서는 꼭 고정이 필요

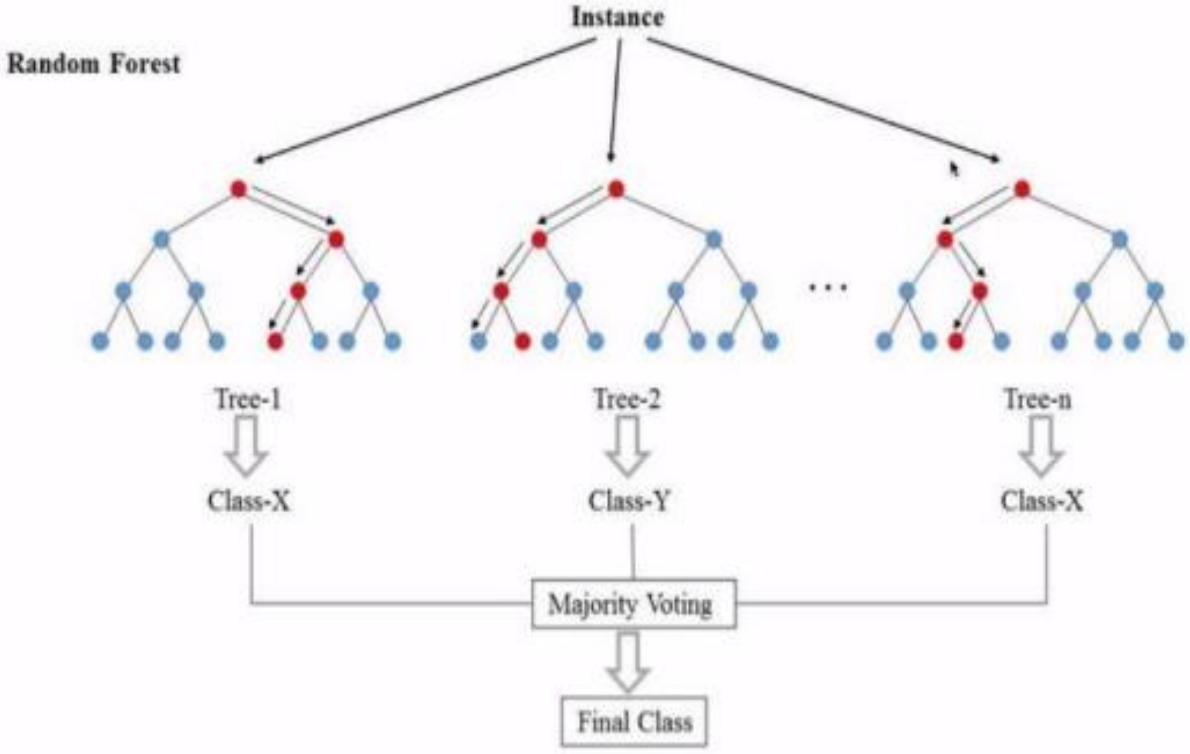
Data feature의 개수는 25개

기본 파라미터를 사용한 Random Forest의

분류 문제의 학습 과정을 예시로 다룸!

n_estimators = 100, max_features = 5, max_depth = None

Random Forest의 학습 과정



1. Bootstrap 기법을 이용하여 100개의 Data set을 구성
2. 100개의 max_depth가 None인 DT를 준비
3. 100개의 DT를 학습
4. 각 DT 학습 시 노드를 분할 할 때 마다 5개의 feature를 무작위로 선택 후 가장 불순도를 낮추는 feature를 기준으로 분할
5. 4의 과정을 더 이상 분할이 불가능 할 때 까지 반복
6. 100개 DT의 결과의 class 확률 값을 평균 내어 예측

Random Forest의 변수의 중요도

- 랜덤 포레스트는 선형 회귀모델 / 로지스틱 회귀모델과는 달리 개별 변수가 통계적으로 얼마나 유의한지 알 수 없음(확률 분포 가정을 하지 않기 때문에)
- 따라서 개별 변수의 중요도를 알기 위해 전체 **Tree에서 각각의 변수가 불순도를 얼마나 감소시키는지를 평균 계산하여 중요도를 계산**
- 불순도를 낮추는 변수는 class가 많은 범주형 변수가 뽑힐 확률이 높음
- 상관관계가 높은 변수들 중 하나가 선택된다면, 나머지는 동일한 기능을 하기 때문에 선택될 확률이 감소함. **즉, 상관관계가 높은 변수들 끼리 중요도의 왜곡이 생길 수 있음**

feature_importances_ 가 만능은 아니다!

Random Forest의 Hyperparameter

sklearn.ensemble.RandomForestClassifier¶

```
class sklearn.ensemble.RandomForestClassifier(n_estimators=100, *, criterion='gini',
max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0,
max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None,
bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False,
class_weight=None, ccp_alpha=0.0, max_samples=None)                                    [source]
```

sklearn.ensemble.RandomForestRegressor

```
class sklearn.ensemble.RandomForestRegressor(n_estimators=100, *, criterion='mse',
max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0,
max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None,
bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False,
ccp_alpha=0.0, max_samples=None)                                                       [source]
```

04. Boosting

배깅은 DT 1...DT N개가 서로 **독립적**으로 결과를 예측한다.

N개의 독립된 DT의 결과를 집계해 최종 결과 값을 예측!

그러나 부스팅은 독립이 아니다!

Unit 04 | Boosting

부스팅은 모델이 **연속적(모델간 팀워크)**으로 만들어진다!

처음 모델이 예측을 하면 그 예측 결과에 따라 데이터에 가중치가 부여되고,

부여된 가중치가 **다음 모델에 영향**을 준다!

잘못 분류된 데이터에 집중(가중치 부여)하여 새로운 분류 규칙을 만드는 단계를 반복한다!

배깅은 독립된 DT의 결합 / 부스팅은 연속된 DT간의 결합

Boosting 이란?

- 부스팅은 **가중치를 활용**하여 약 분류기(High Bias Model)를 강 분류기로 만드는 방법
- 학습 Round를 진행하면서 모델을 생성

- 해당 모델들로 앙상블 모델을 만듦
- 잘못 분류된 데이터를 더 잘 분류해보는 것이 목적!
- 부스팅의 종류는 다양하지만 대표적으로 2가지 방법이 많이 쓰임

Adaptive Boosting : AdaBoost

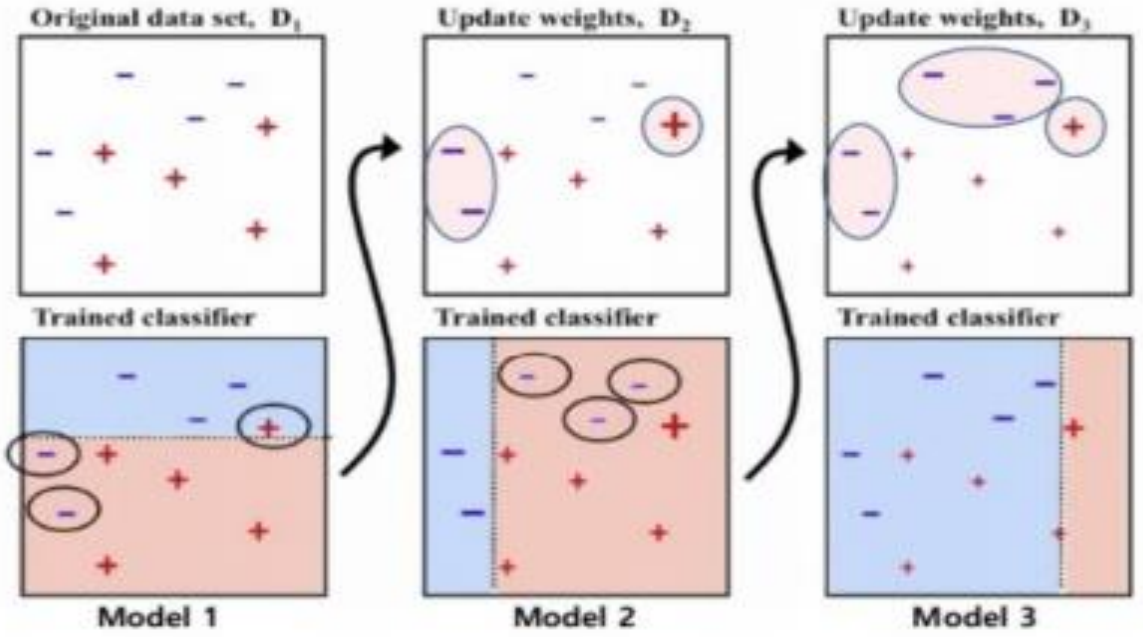
Gradient Boosting : GBM, XGBoost, LGBM, CatBoost 등등

AdaBoost(Adaptive Boosting) 이란?

- AdaBoost == Adaptive Boosting
- 매 Round마다 Observation의 Weight 값을 계산
- **틀리는 Observation의 Weight Up -> Weight 기준 Resampling(중복허용)**
- 약한 분류기에 입력 값을 변화시켜가며 강한 분류기를 만듦
- High-Depth DT, NN (High Variance Model)에는 적합하지 않음
- **약 분류기로 stump(depth가 1인 매우 작은 DT)를 사용**

AdaBoost 학습 과정

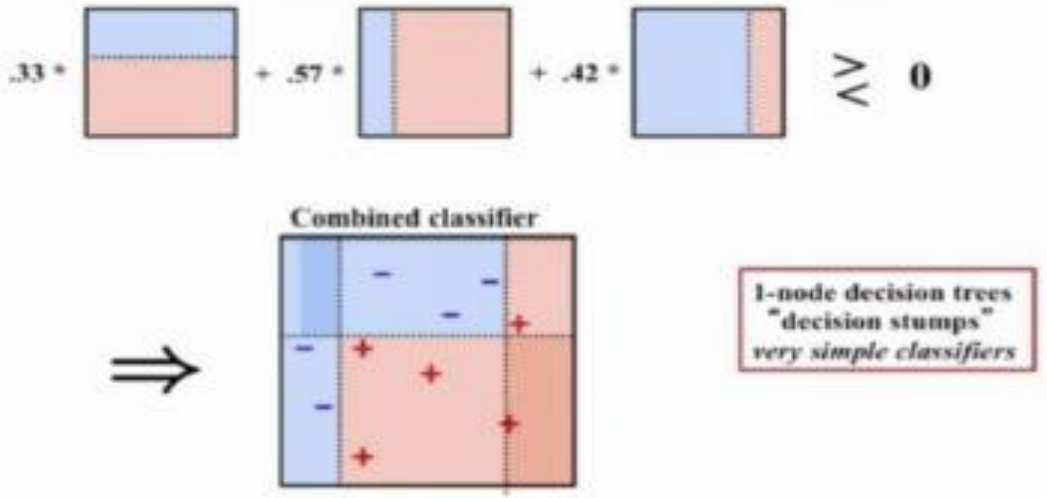
- Model1에서 잘못 예측한 데이터에 가중치를 부여
- Model2는 잘못 예측한 데이터를 분류하는데 더 집중
- Model3는 Model1, 2가 잘못 예측한 데이터를 분류하는데 집중



- Cost Function : 가중치(W)를 반영하여 계산

$$J(\theta) = \sum_i w_i J_i(\theta, x^{(i)})$$

- 3개의 모델별로 계산된 가중치를 합산하여 최종 모델을 생성



https://www.youtube.com/watch?v=ix6IvwbVpw0&list=PL4zv4UkoVTPflyFDJdJtz248-8Wdz_nct&index=3

AdaBoost 학습 과정

Algorithm 10.1 *AdaBoost.M1.*

값 샘플의 weight를 1/N로 초기화

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.

N은 데이터의 갯수

2. For $m = 1$ to M: M은 모델의 갯수

Weight값을 기준으로 분류기 생성(resampling)

   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.

   (b) Compute 해당 분류기의 에러 계산

$$I(y_i, G_m(x_i)) = \begin{cases} 0 & \text{if } y_i = G_m(x_i) \\ 1 & \text{if } y_i \neq G_m(x_i) \end{cases}$$

$$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$

   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$. 해당 분류기의 분류기 가중치 생성

   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.

Instance의 weight 업데이트

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

AdaBoost 학습 과정

값 샘플의 weight를 1/N로 초기화

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N.$

N은 데이터의 갯수

모든 관측치들의 Weight를 동일하게 초기화

AdaBoost 학습 과정

2. For $m = 1$ to M: M은 모델의 갯수

Weight값을 기준으로 분류기 생성(resampling)

(a) Fit a classifier $G_m(x)$ to the training data using weights w_i.

이제 M개의 모델이 돌아감 (for문 반복)

관측치들의 Weight 값을 기준으로 Sampling을 진행

최초의 샘플링 - 모두 뽑힐 확률이 동일

n_round 샘플링 - 틀린 관측치는 가중치가 높아지므로 뽑힐 확률이 높아짐

AdaBoost 학습 과정

(b) Compute 해당 분류기의 에러 계산

$$I(y_i, G_m(x_i)) = \begin{cases} 0 & \text{if } y_i = G_m(x_i) \\ 1 & \text{if } y_i \neq G_m(x_i) \end{cases}$$

$$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$

예측값과 실제값이 같으면 0이 되고, 다르면 1이 되게 함.

분자 - 틀린 예측 값의 instance의 weigh만 남게 되어 모두 더함

분모 - 모든 가중치를 더 함(Normalizer 역할)

따라서 err의 범위는 0 < err < 1 이다.

AdaBoost 학습 과정

(c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$. 해당 분류기의 분류기 가중치 생성

에러가 높다면 m번째 모델은 좋지 않은 모델

에러가 낮다면 m번째 모델은 좋은 모델

반비례 관계 이므로 (1-err)/err를 통해 반대방향으로 바꿔 모델의 가중치를 계산

모델의 가중치는 전체 앙상블에서 해당 모델에 역할의 비중을 의미! (높을 수록 앙상블 결과에 많은 반영)

AdaBoost 학습 과정

$$(\mathrm{d})\ \text{Set } w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N.$$

Instance의 weight 업데이트

모델의 가중치의 exp를 하여 틀린 값들의 가중치만 업데이트!

지금과 같은 방식을 for문을 통해 M개의 모델 만큼 반복!

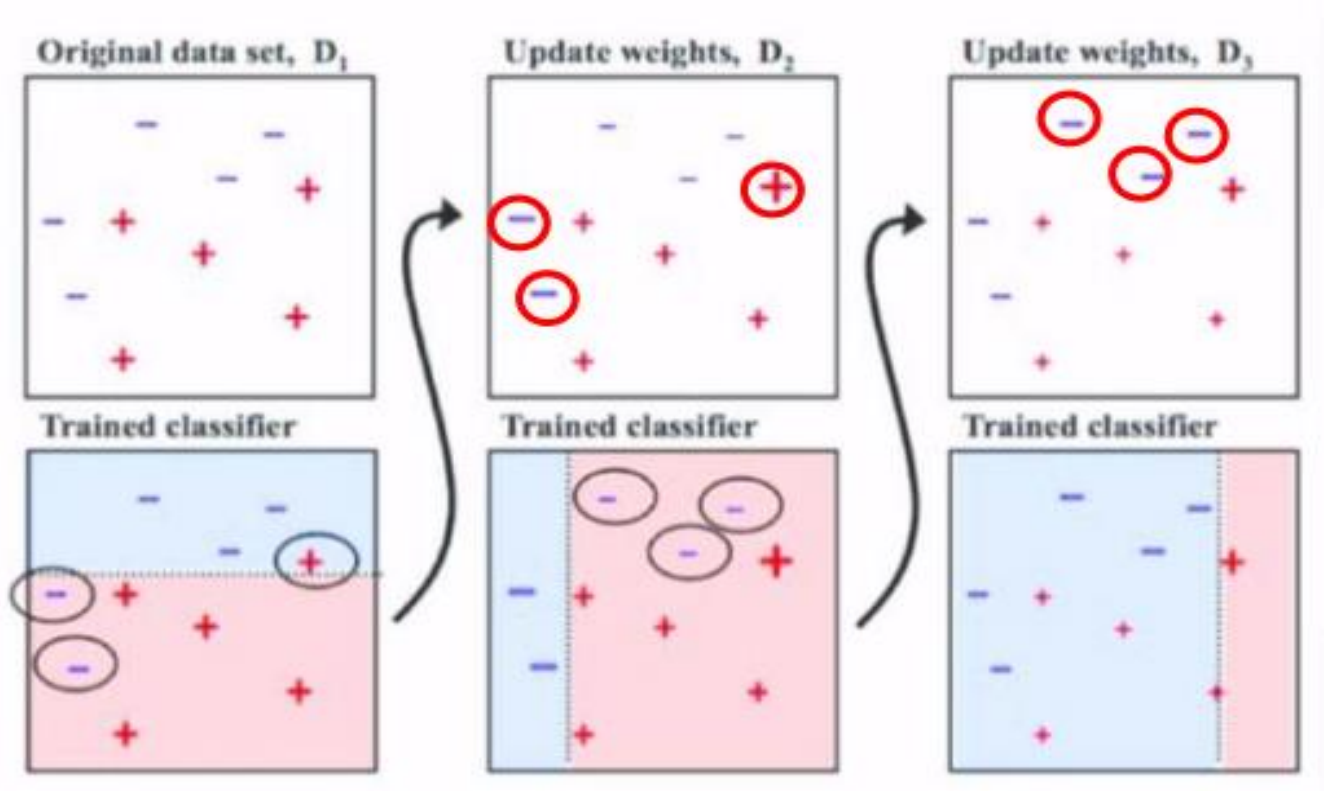
AdaBoost 학습 과정

$$3.\ \text{Output}\ G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right].$$

M개의 모델을, 각각의 모델의 예측치에 모델의 가중치를 곱한 후

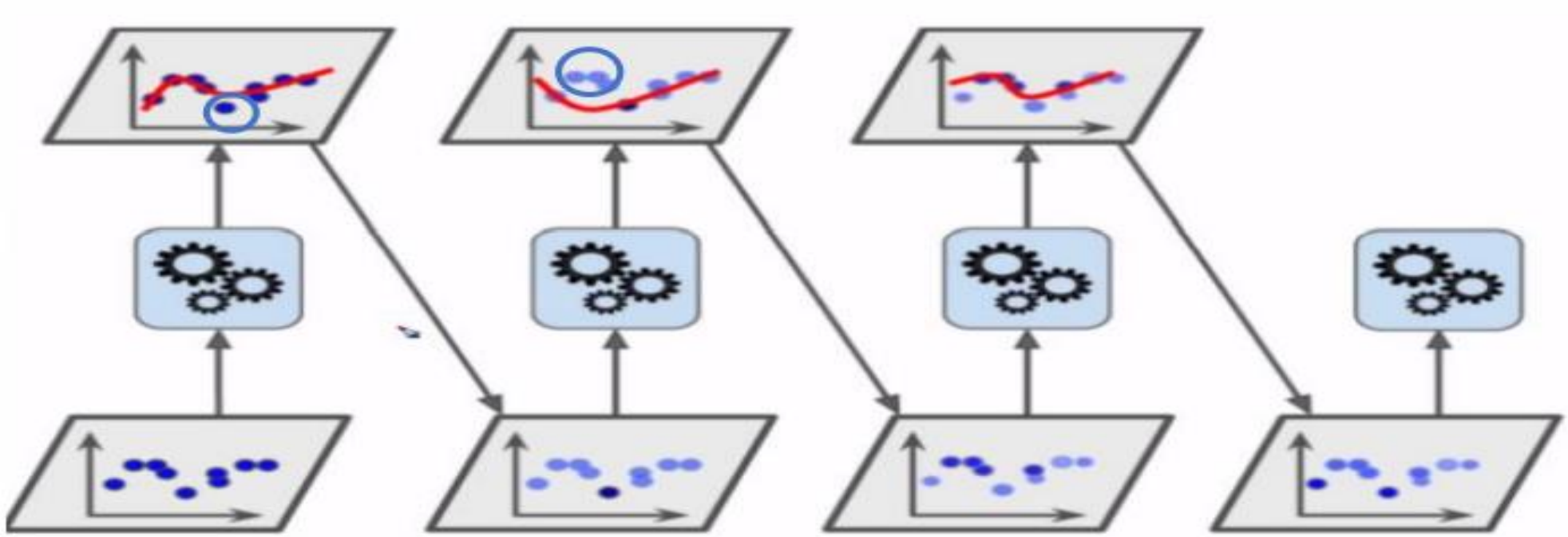
Voting 방식을 통하여 최종 예측

AdaBoost - Classfication



오분류된 Instance에 가중치를 줘서 완벽해져가는 모델.

AdaBoost - Regression



AdaBoost의 Hyperparameter

sklearn.ensemble.AdaBoostClassifier¶

class sklearn.ensemble.**AdaBoostClassifier**(*base_estimator=None, *, n_estimators=50, learning_rate=1.0, algorithm='SAMME.R', random_state=None*) [source]

AdaBoost는 이전 모델의 단점을 데이터에서 찾는 방식

따라서 **사용할 수 있는 손실함수가 한정**되기 때문에

모델의 유연성이 떨어짐

그래서 **모델의 잔여 오차(residual error)를 최소화**하는

Gradient Descent 알고리즘을 활용하는

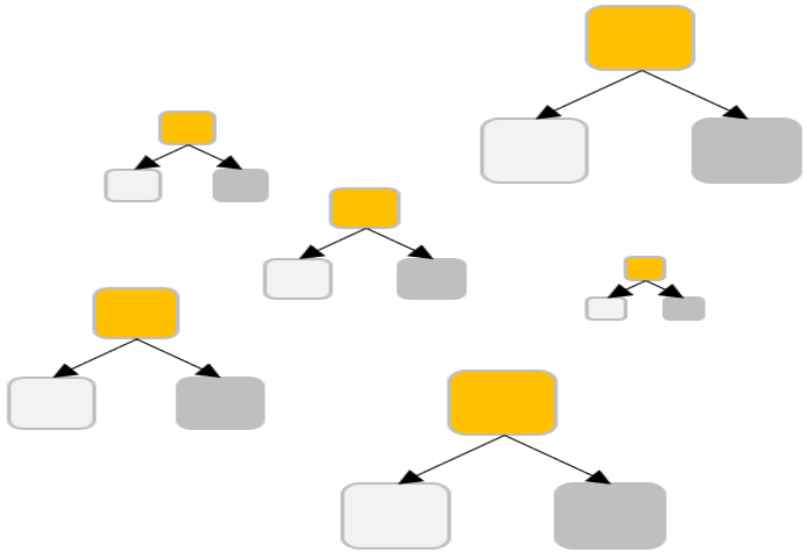
Gradient Boosting 방법이 있음!

(이전 모델의 단점을 기울기로 통해서 찾는 방식)

AdaBoost VS Gradient Boosting

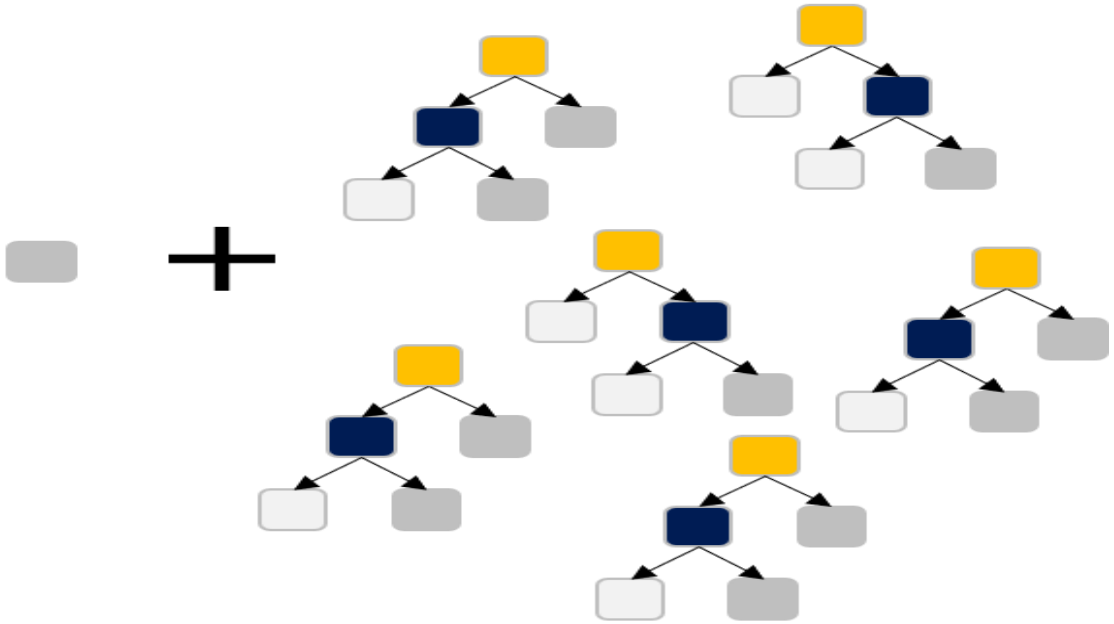
AdaBoost

- Stump
  - With different amount of say



Gradient Boosting

- A leaf and restricted tress
  - With equal learning rate



Gradient Boosting 이란?

- Sequential + Additive Model
- 매 Round마다 잔여 오차(residual error)값을 계산
- **Residual을 가지고 Model을 학습**
- 전에 학습된 모델의 오차를 보완하는 방향으로 학습
- Residual을 예측하여 발전하는 Weak learner
- **약 분류기로 restricted tree(depth 2이상인 성장에 제한을 둔 DT)를 사용**

Gradient Boosting for Regression 학습 과정

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

Step 2: for $m = 1$ to M:

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

(B) Fit a regression tree to the r_{im} values and create terminal regions R_{jm}, for $j = 1 \ldots J_m$

(C) For $j = 1 \ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

(D) Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

식은 다음과 같지만.....(머리 아프네요)

1. first Model의 output을 구해 평균 : F

2-1. y – F를 구해 Residual를 구하고 : r

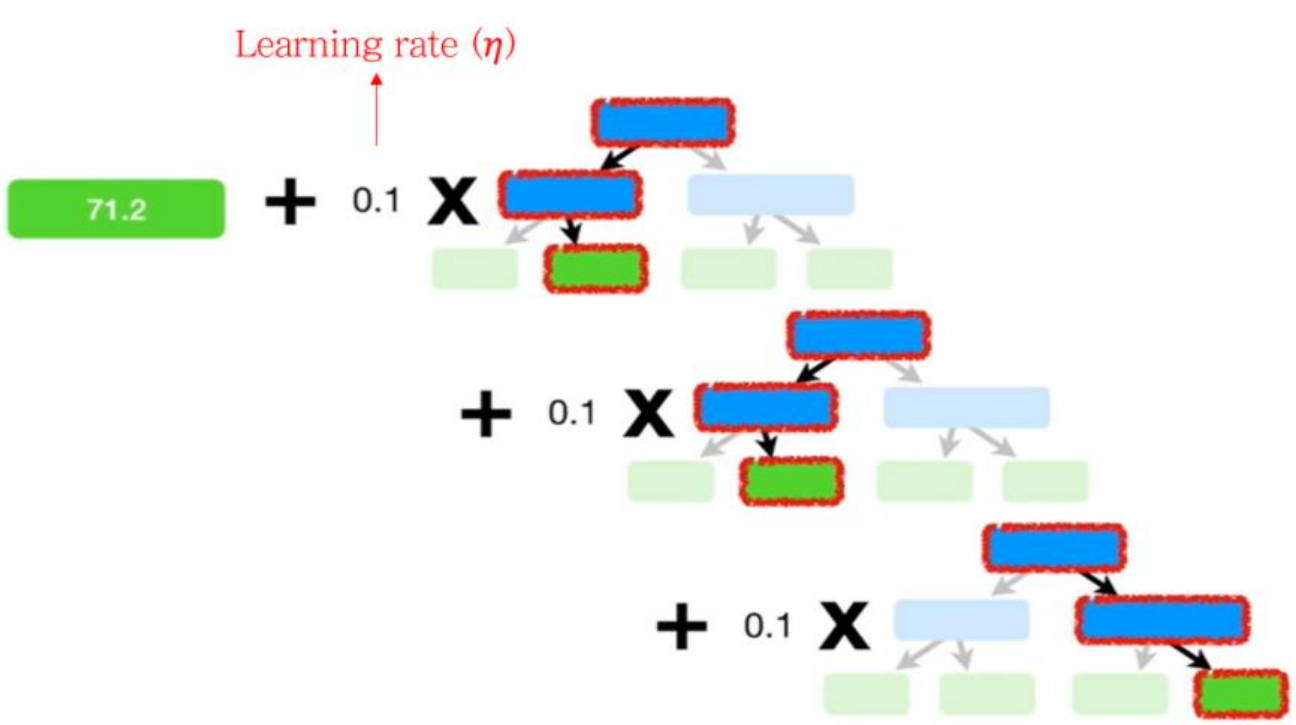
2-2. r을 예측하는 모델을 학습 후 X를 넣어 예측: gamma

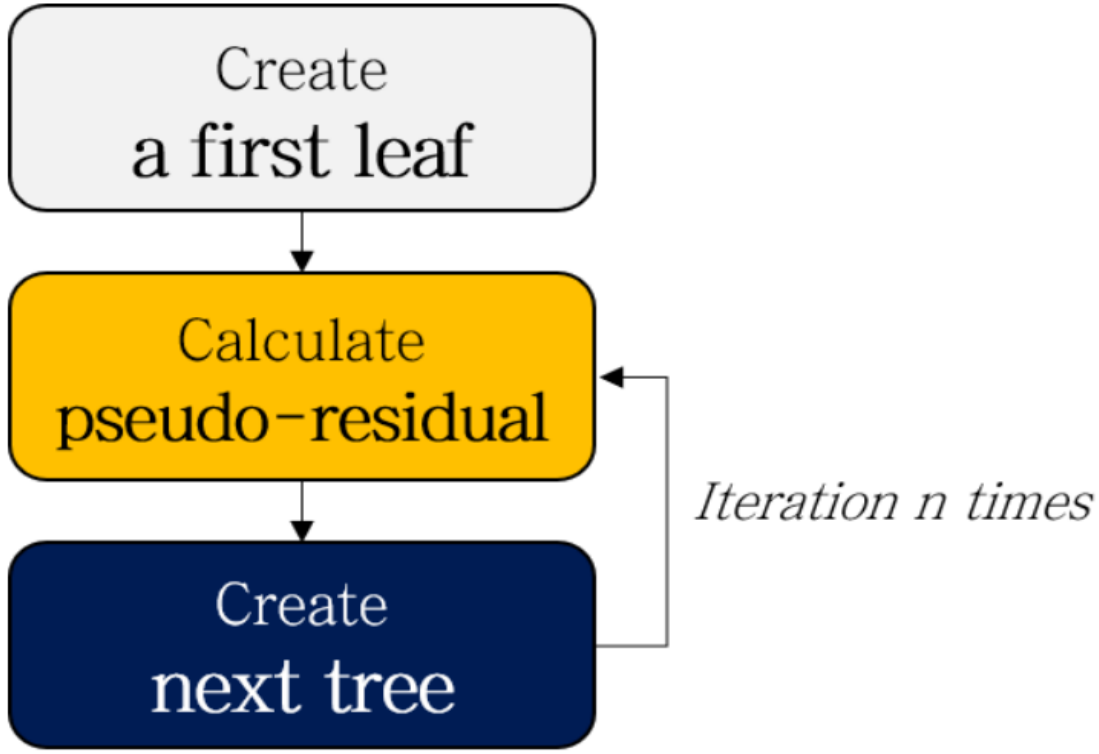
2-3. F + lr(학습률) * gamma 을 통해 예측 값을 구함 : F

2-1 ~ 2-3 의 과정을 계속 반복해가면서 F를 업데이트

한다고 이해하시면 됩니다!

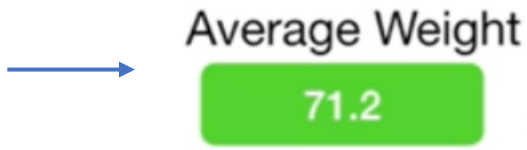
Gradient Boosting for Regression 학습 과정





Gradient Boosting for Regression 학습 과정

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57



first DT의 Output을 구해 평균 값을 계산

Gradient Boosting for Regression 학습 과정

Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	4.8
1.5	Blue	Female	56	-15.2
1.8	Red	Male	73	1.8
1.5	Green	Male	77	5.8
1.4	Blue	Female	57	-14.2

Calculate
pseudo-residual

실제 값 – 예측 값을 통해 Residual을 계산

Gradient Boosting for Regression 학습 과정

Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	4.8
1.5	Blue	Female	56	-15.2
1.8	Red	Male	73	1.8
1.5	Green	Male	77	5.8
1.4	Blue	Female	57	-14.2

Create
next tree

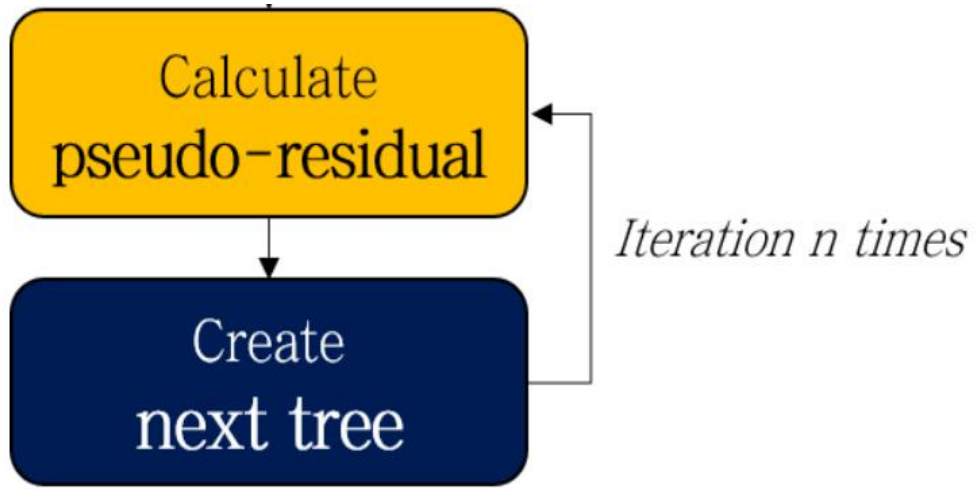
Residual을 예측하는 DT를 만든다.

Gradient Boosting for Regression 학습 과정

Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	4.8
1.5	Blue	Female	56	-15.2
1.8	Red	Male	73	1.8
1.5	Green	Male	77	5.8
1.4	Blue	Female	57	-14.2

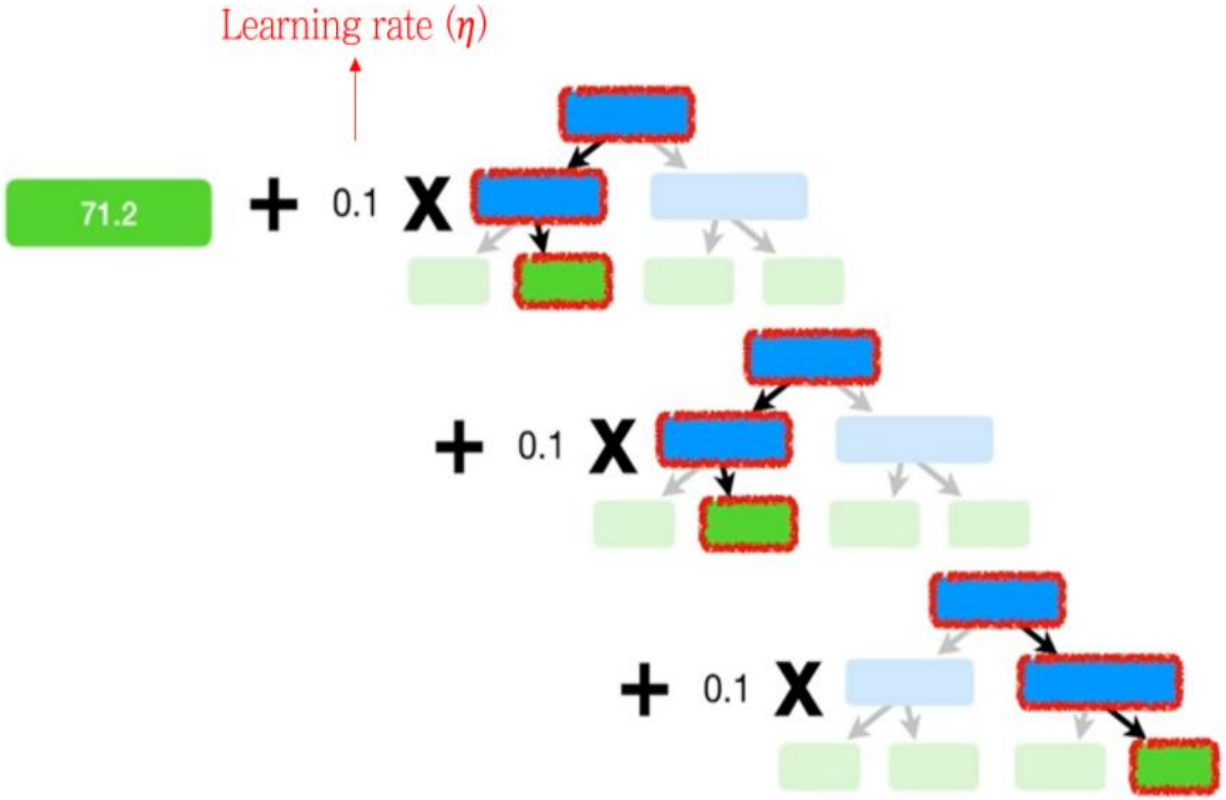
Less Residual

Residual
15.1
4.3
-13.7
1.4
5.4
-12.7



반복을 통해 Residual은 점점 감소

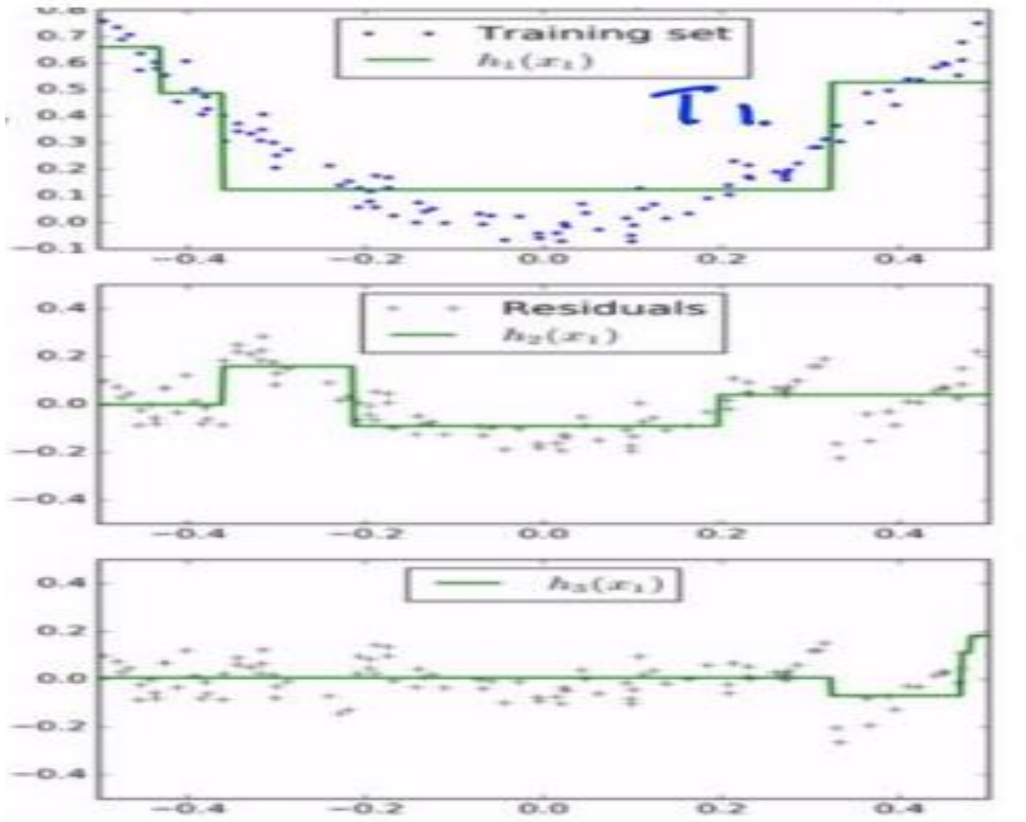
Gradient Boosting for Regression 학습 과정



만들어진 DT에 학습률을 곱하여

모두 더하면 최종 예측 값이 나온다

Gradient라고 부르는 이유!



Gradient Boosting은 잔차를 감소시키는 방향으로 학습을 하고,

이때 학습 방법을 Gradient Method를 쓰기 때문에,

Gradient라고 부른다!

Gradient Boosting의 Hyperparameter

sklearn.ensemble.GradientBoostingClassifier¶

class sklearn.ensemble.**GradientBoostingClassifier**(*, *loss='deviance', learning_rate=0.1, n_estimators=100, subsample=1.0, criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_decrease=0.0, min_impurity_split=None, init=None, random_state=None, max_features=None, verbose=0, max_leaf_nodes=None, warm_start=False, validation_fraction=0.1, n_iter_no_change=None, tol=0.0001, ccp_alpha=0.0*) [source]

sklearn.ensemble.GradientBoostingRegressor

class sklearn.ensemble.**GradientBoostingRegressor**(*, *loss='ls', learning_rate=0.1, n_estimators=100, subsample=1.0, criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_decrease=0.0, min_impurity_split=None, init=None, random_state=None, max_features=None, alpha=0.9, verbose=0, max_leaf_nodes=None, warm_start=False, validation_fraction=0.1, n_iter_no_change=None, tol=0.0001, ccp_alpha=0.0*) [source]

그런데 Gradient Boost 는 굉장히 잘 만들어진 알고리즘이지만,

연속된 Model의 결합이기 때문에 **속도가 느리고 비효율적**

그래서 Microsoft에서는 이를 개선하기 위해

Regularization을 사용하고, 분산처리 지원을 통해 속도를 향상시키고,

Package화를 한 Model을 제공해줌!

그것이 바로!

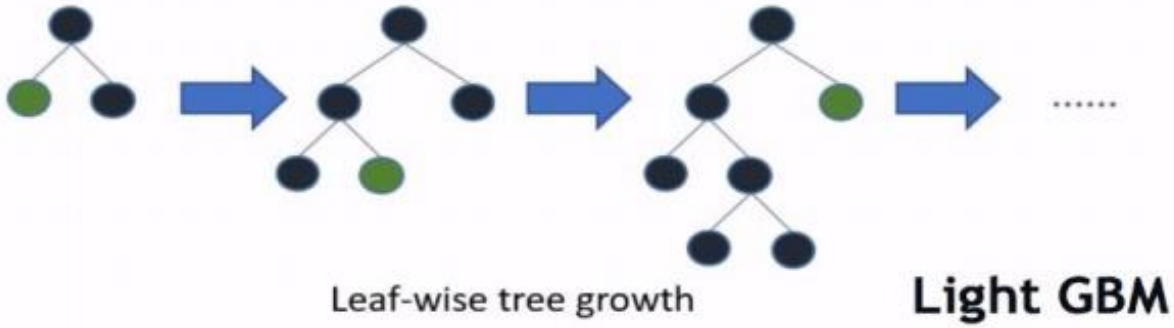
XGBoost 와 LightGBM

이 밖에도 다양한 GBM 계열의 Model을 제공해줌(CatBoost, EBM 등)

XGBoost and LightGBM

- XGBoost - eXtreme Gradient Boosting
- LightGBM – Light Gradient Boosting Machine
- 구현 알고리즘은 일부 다르지만 목표는 비슷함
- **둘 다 Tree기반의 Gradient Boosting임**
- 병렬 처리를 통한 속도 향상

XGBoost and LightGBM



- XGBoost
  - ✓ **균형적으로 성장**
  - ✓ LightGBM 보다 느린 속도
  - ✓ LightGBM 보다 과적합에 강건
- LightGBM
  - ✓ **한쪽으로만 성장**
  - ✓ XGBoost보다 빠른 속도
  - ✓ XGBoost보다 과적합이 쉽게 발생

Bagging과 Boosting 의 차이

비교	Bagging	Boosting
특징	병렬 앙상블 모델 (각 모델은 서로 독립적)	연속 앙상블 (이전 모델의 오류를 고려)
목적	Variance 감소	Bias 감소
적합한 상황	복잡한 모델 (High variance, Low bias)	Low variance, High bias 모델
대표 알고리즘	Random Forest	Gradient Boosting, AdaBoost
Sampling	Random Sampling	Random Sampling with weight on error

Bagging과 Boosting 의 차이

부스팅은 배깅에 비해 error가 작다. 즉 성능이 더 좋다!

하지만 속도가 느리고 오버 피팅이 될 가능성이 존재!

그러면 배깅과 부스팅 중 무엇이 더 좋을까? 상황에 따라 다르다!

개별 Model의 **낮은 성능**이 문제라면 **부스팅**이 적합!

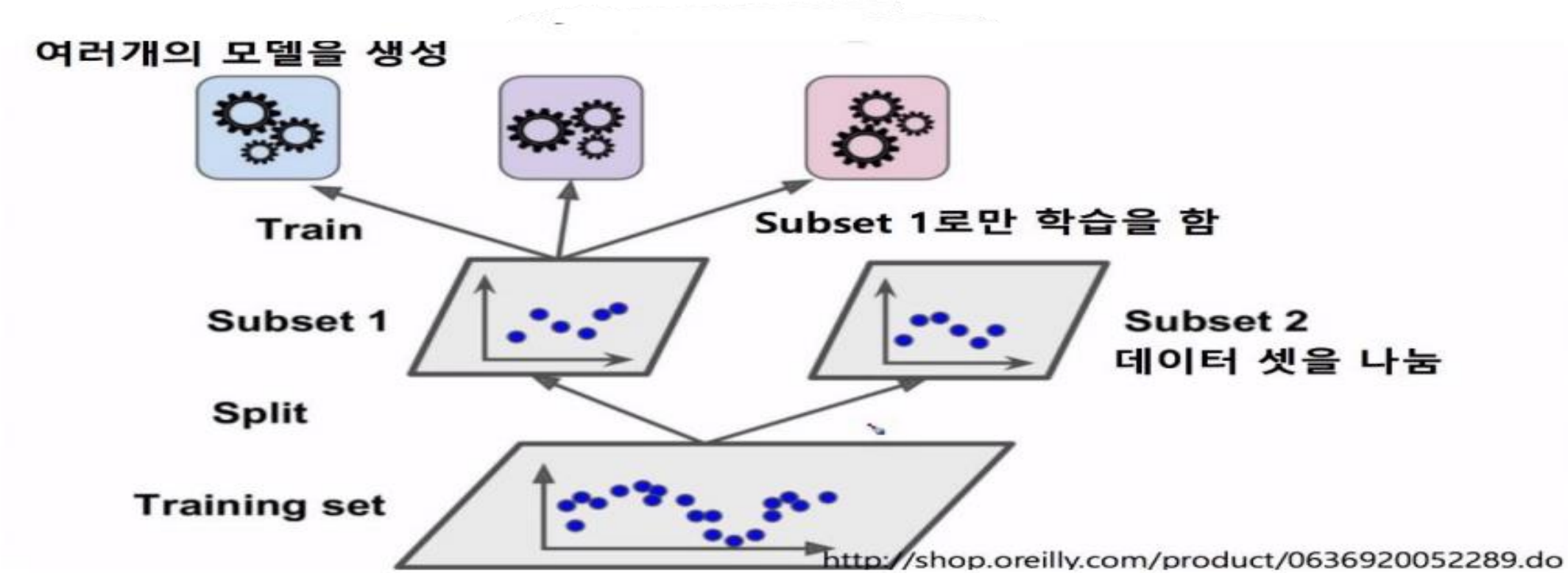
오버 피팅이 문제라면 **배깅**이 적합!

05. Stacking

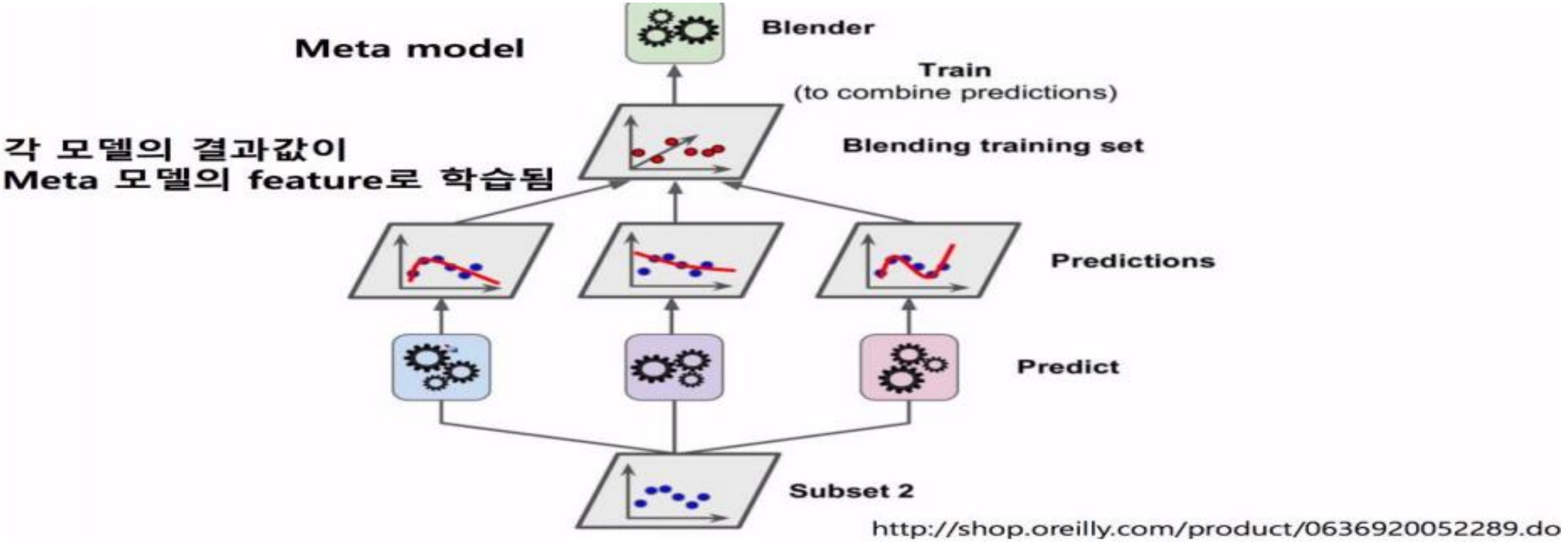
Stacking 이란?

- 앙상블에 속한 모든 예측기의 예측을 취합하는 간단한 함수 대신 **취합하는 모델을 훈련시키자** 라는 아이디어에서 시작
- Stacking은 서로 다른 모델들을 조합해서 최고의 성능을 내는 모델을 생성
- **개별 모델은** SVM, RandomForest, KNN 등 **다양한 알고리즘을 사용**
- **여러 모델의 결과를 묶어서 예측**하는 Meta Model을 만듦
- 조합을 통해 서로의 장점은 취하고 약점을 보완

Stacking의 학습 과정



Stacking의 학습 과정



지금까지 배운 모든 내용 Code 실습

Week4_Ensemble_BaseLine_Model.ipynb

코드를 Colab에서 실행해주세요

들어주셔서 감사합니다.

Assignment : 캐글 경진대회 참여!

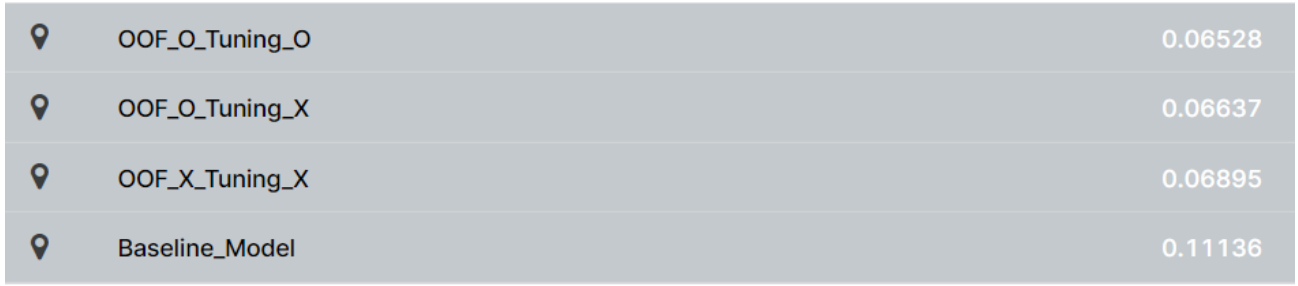
캐글 경진 대회에 참여하여 가장 좋은 Model 을 만들어 보세요!

채점 기준은 리더보드 + EDA + 모델의 결과에 대한 설명 입니다!

Feature Engineering과 Hyperparameter Tunning을 통해서 최적의 모델을 찾아보세요

오늘까지 배운 모든 모델을 모두 활용해보세요~

BaseLine Model은 모두 넘으셔야 합니다!!



https://www.kaggle.com/t/cd058141be084c4f81c9df010c587ae4

Unit | 참고 자료

- 핸즈온 머신러닝 2판
- 10기 이준걸님 Ensemble 강의 자료
- 12기 배유나님 Ensemble 강의 자료
- 13기 이예진님 Ensemble 강의 자료
- https://tyami.github.io/machine%20learning/ensemble-4-boosting-gradient-boosting-regression/
- https://assaeunji.github.io/ml/2020-09-05-gbm/
- https://tensorflow.blog/%ED%8C%8C%EC%9D%B4%EC%8D%AC-%EB%A8%B8%EC%8B%A0%EB%9F%AC%EB%8B%9D/2-3-6-%EA%B2%B0%EC%A0%95-%ED%8A%B8%EB%A6%AC%EC%9D%98-%EC%95%99%EC%83%81%EB%B8%94/
- https://becominghuman.ai/ensemble-learning-bagging-and-boosting-d20f38be9b1e
- https://soobarkbar.tistory.com/21