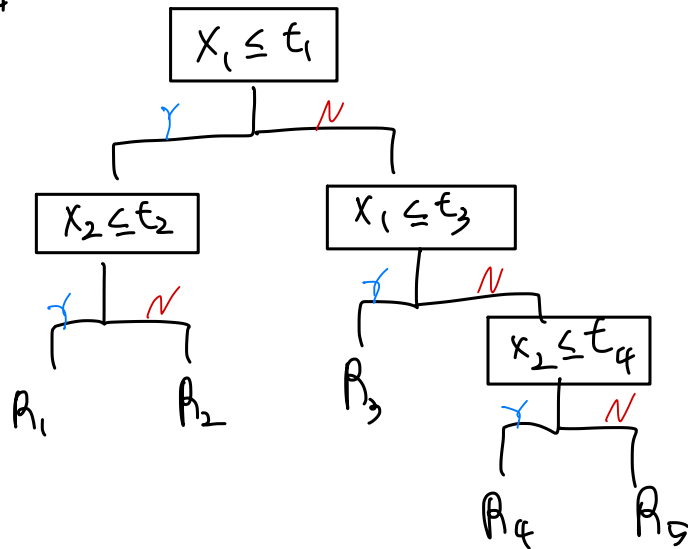
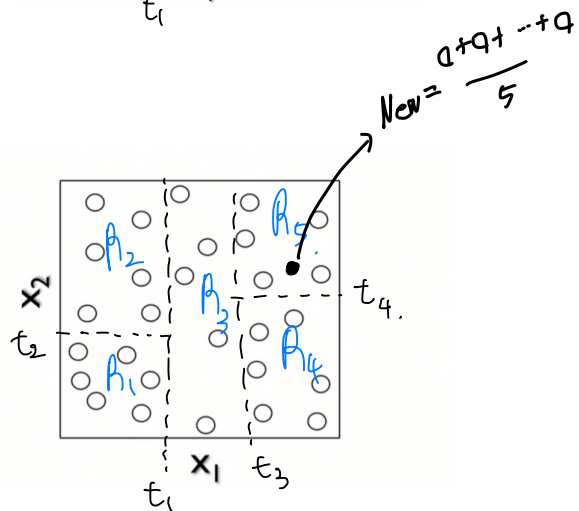
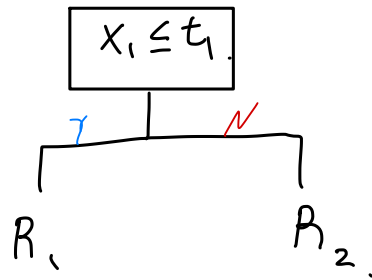
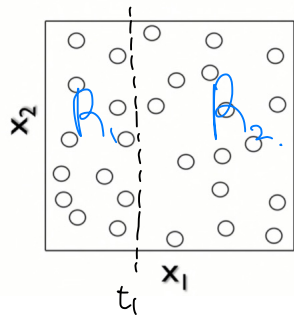


#결정 트리



· 분할 변수와 분할 지점은  $\text{Criterion} = \begin{cases} \text{gini} \\ \text{entropy} \end{cases}$

"분할도"

$$\text{지니불순도} = 1 - (\text{음성클래스 비율}^2 + \text{양성클래스 비율}^2)$$

$$1 - \left( \left( \frac{1259}{5199} \right)^2 + \left( \frac{3939}{5199} \right)^2 \right) = 0.369$$

$$1 - \left( \left( \frac{50}{100} \right)^2 + \left( \frac{50}{100} \right)^2 \right) = 0.5$$

$$1 - \left( \left( \frac{0}{100} \right)^2 + \left( \frac{100}{100} \right)^2 \right) = 0$$

· 결정 트리는 모든 데이터를 하나하나 분할할 수 있음.

기본적으로 과대적합 위험성을 가지고 있음.

⇒ 랜덤성을 추가하거나 → 앙상블.

트리의 성장을 제한. → 가지 치기.

## 결정 트리의 장점.

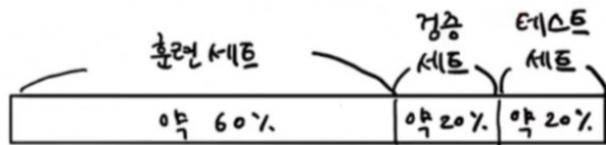
- 이해하기 쉬움.
- 데이터 전체가 필요 없음.
- 특성 중요도를 구할 수 있음.

## # 교차 검증.

- 성능 테스트를 위해 테스트 세트를 계속 사용하면 의미가 퇴색됨.

따라서 over-fitting, under-fitting 을 판단할 "검증 세트" 가 필요함.

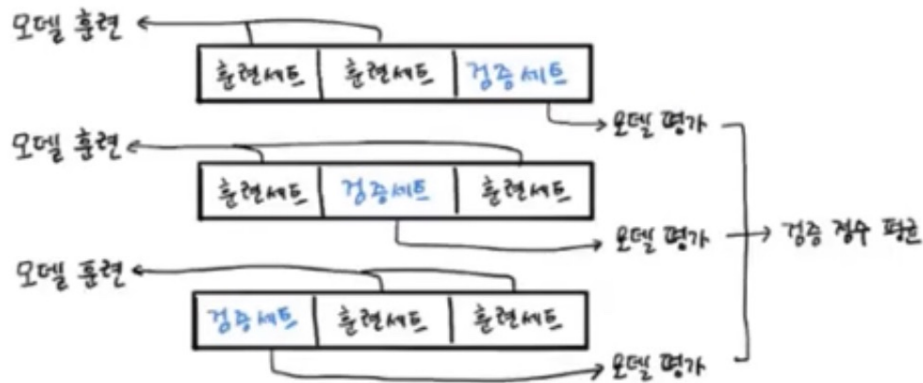
=> 훈련세트를 한번 더 나눔.



충분한 양의 데이터를 검증에 쓰고

=> K-Fold 교차 검증.

훈련세트를 크게 작아지 않는 방향으로?



```
from sklearn.model_selection import StratifiedKFold

scores = cross_validate(dt, train_input, train_target, cv=StratifiedKFold())
print(np.mean(scores['test_score']))
0.855300214703487

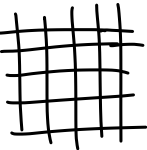
splitter = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
scores = cross_validate(dt, train_input, train_target, cv=splitter)
print(np.mean(scores['test_score']))
0.8574181117533719
```

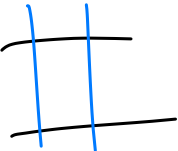
## #하이퍼 파라미터 튜닝.

ex. 결정 트리 - 여러개의 하이퍼 파라미터. (사용자 지정 파라미터)

트리의 깊이, 분할도 차이

↙  
특정적이지 않음. (동시에 바뀌어야 함.)

⇒ { 그리드 서치 :  x folds의 갯수. (정확도 ↑)

랜덤 서치 :  → 랜덤하게 여러번. (속도 ↑)

정형 데이터

- CSV
- 엑셀
- ⋮

VS 비정형 데이터

- 텍스트
- 이미지
- 음악
- ⋮

앙상블 학습

ensemble learning

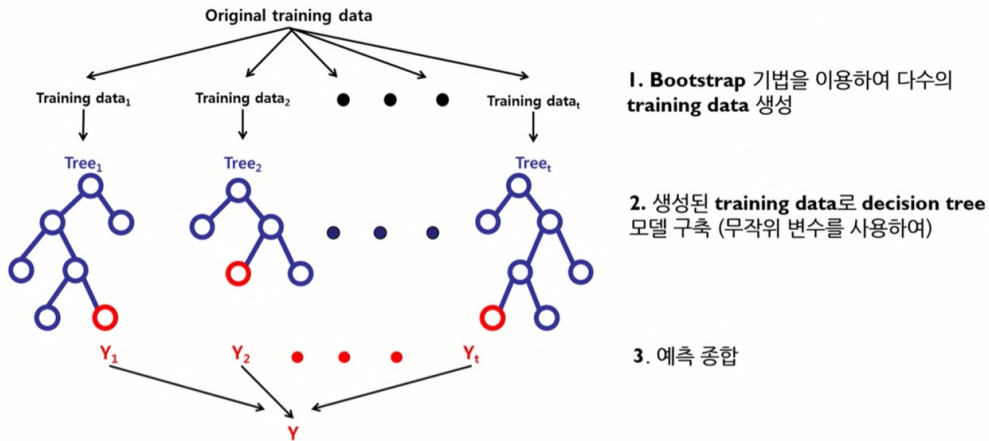


신경망 알고리즘

Neural network

## 랜덤 포레스트 개요

- 다수의 의사결정나무모델에 의한 예측을 종합하는 앙상블 방법
- 일반적으로 하나의 의사결정나무모델 보다 높은 예측 정확성을 보여줌
- 관측치 수에 비해 변수의 수가 많은 고차원 데이터에서 중요 변수 선택 기법으로 널리 활용됨





핵심 아이디어 : Diversity, Random 확보.

1. 여러개의 Training data를 생성하여 각 데이터마다 결정트리 구축.

=> Bagging.

2. 결정트리 구축 시 변수 무작위로 선택.

=> Random subspace.

# · Bagging (Bootstrap Aggregating)

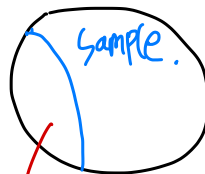
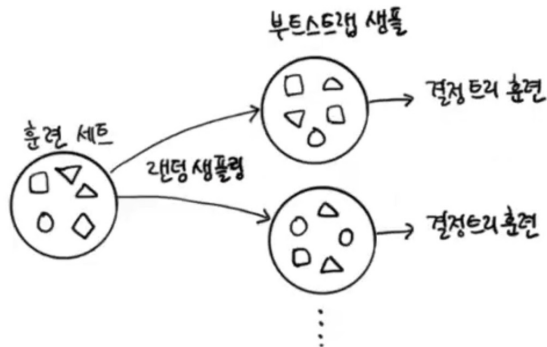
↓ 샘플링      ↗ 결과 종합.

# Bootstrap Sample.

↗ 복원 추출.

이론적으로 한 번도 선택되지 않을 확률.

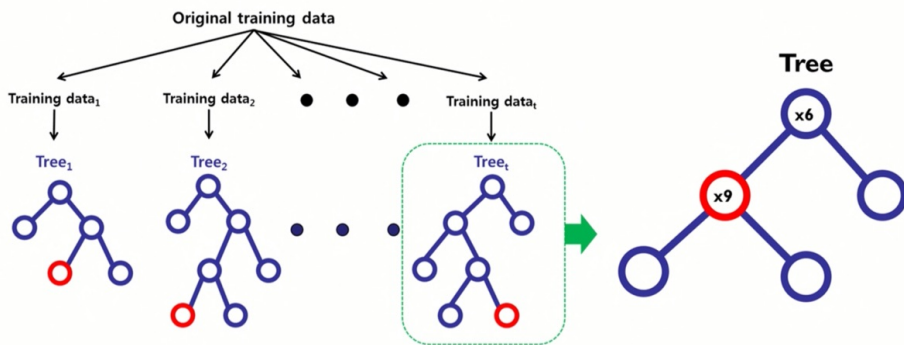
$$\left(1 - \frac{1}{N}\right)^N \Rightarrow \lim_{N \rightarrow \infty} \left(1 - \frac{1}{N}\right)^N = e^{-1} = 0.368$$



OOB. (out of bag)  
검증 세트로 활용 가능.

# # Random Subspace

## Random Subspace



Original Features	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16
Input Features	x1				x5				x9	x10						

3. 이러한 과정을 **full-grown tree**가 될 때까지 반복

# # Result Aggregating.

라벨의 비율.

1 - 6개 → 1 예측.  
0 - 4개

Training Accuracy	Ensemble population	P(y=1) for a test instance	Predicted class label
0.80	Model 1	0.90	1
0.75	Model 2	0.92	1
0.88	Model 3	0.87	1
0.91	Model 4	0.34	0
0.77	Model 5	0.41	0
0.65	Model 6	0.84	1
0.95	Model 7	0.14	0
0.82	Model 8	0.32	0
0.78	Model 9	0.98	1
0.83	Model 10	0.57	1

정확도 가중치 적용

$$\frac{1 \times \boxed{\phantom{0.80}} + 1 \times \boxed{\phantom{0.80}} + \dots}{10}$$

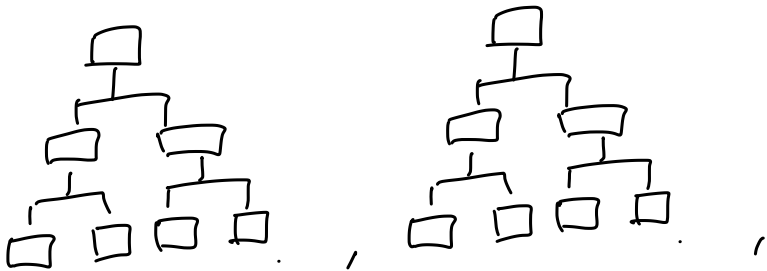
확률의 평균 값.

## # 엑스트라 트리 (Extra Trees)

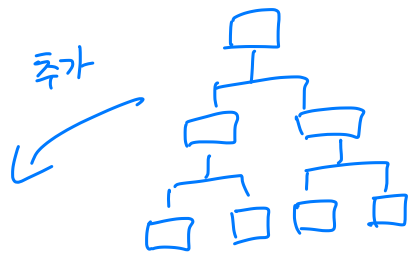
- 랜덤 포레스트와 달리 전체 훈련데이터를 사용함. (부트스트랩 샘플링 X)
- 노드를 분할할 때 '랜덤 서치' 사용.

⇒ 무작위성이 더 크기때문에 깊이는 더 깊어짐.  
그러나 빠르다.

# # Gradient Boosting.

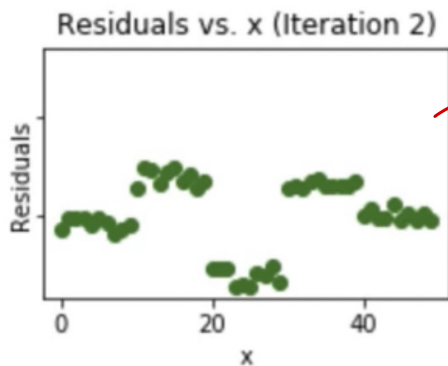
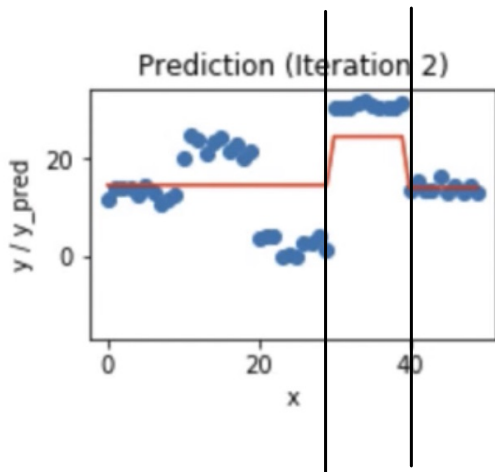


...



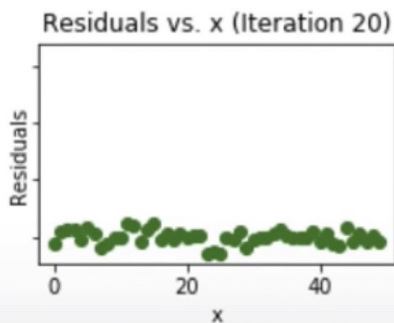
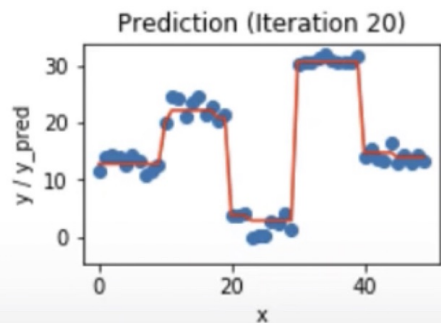
깊이가 얕은 트리를 사용하여 이전 트리의 오차를 보완하는 방식으로 앙상블  
like Gradient descent.

( default : depth 3  
tree 100개. )



잔차들을 이용해  
트리 만들기.

증가.



가장 성능이 좋은 앙코리즘.

⇒ 히스토그램 기반 그라디언트 부스팅 ? ...

XGBoost, LightGBM. <sup>-ms</sup>  
전용 라이브러리 존재.

앙상블

- 랜덤 포레스트
- 엑스트라 트리
- 그라디언트 부스팅

→ 여러개의 트리를 동시에 (병렬적으로) 성장시킬 수 있음.  
( $n-jobs = 1$ ).

→ 조금 더 느리지만 성능이 뛰어나함.



Q & A.

중재.

[https://www.youtube.com/watch?v=Moz8i-tKurk&list=PLJN246IAkhQjoU0C4v8FgtbjOIXxSs\\_4Q&index=13](https://www.youtube.com/watch?v=Moz8i-tKurk&list=PLJN246IAkhQjoU0C4v8FgtbjOIXxSs_4Q&index=13)

<https://www.youtube.com/watch?v=d6nRgztYWQM>

<https://www.youtube.com/watch?v=xki7zQDf74I>