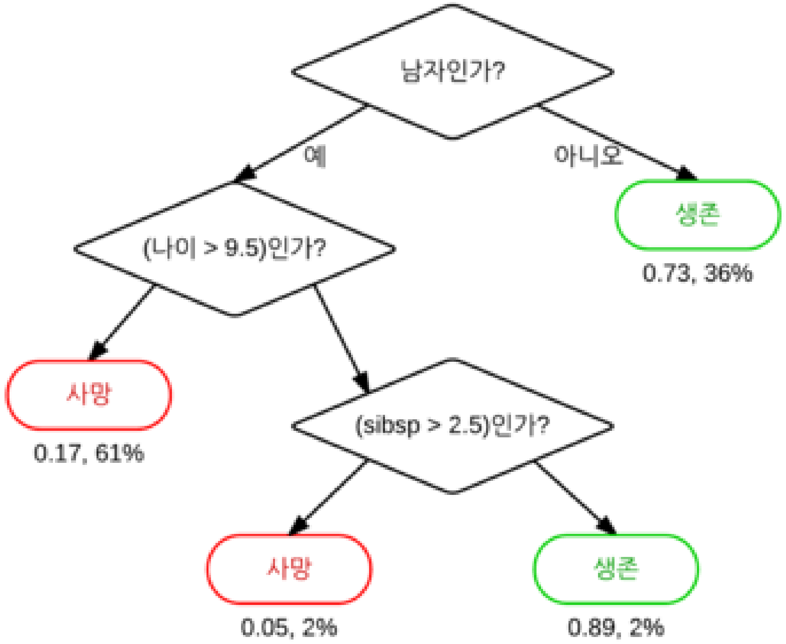
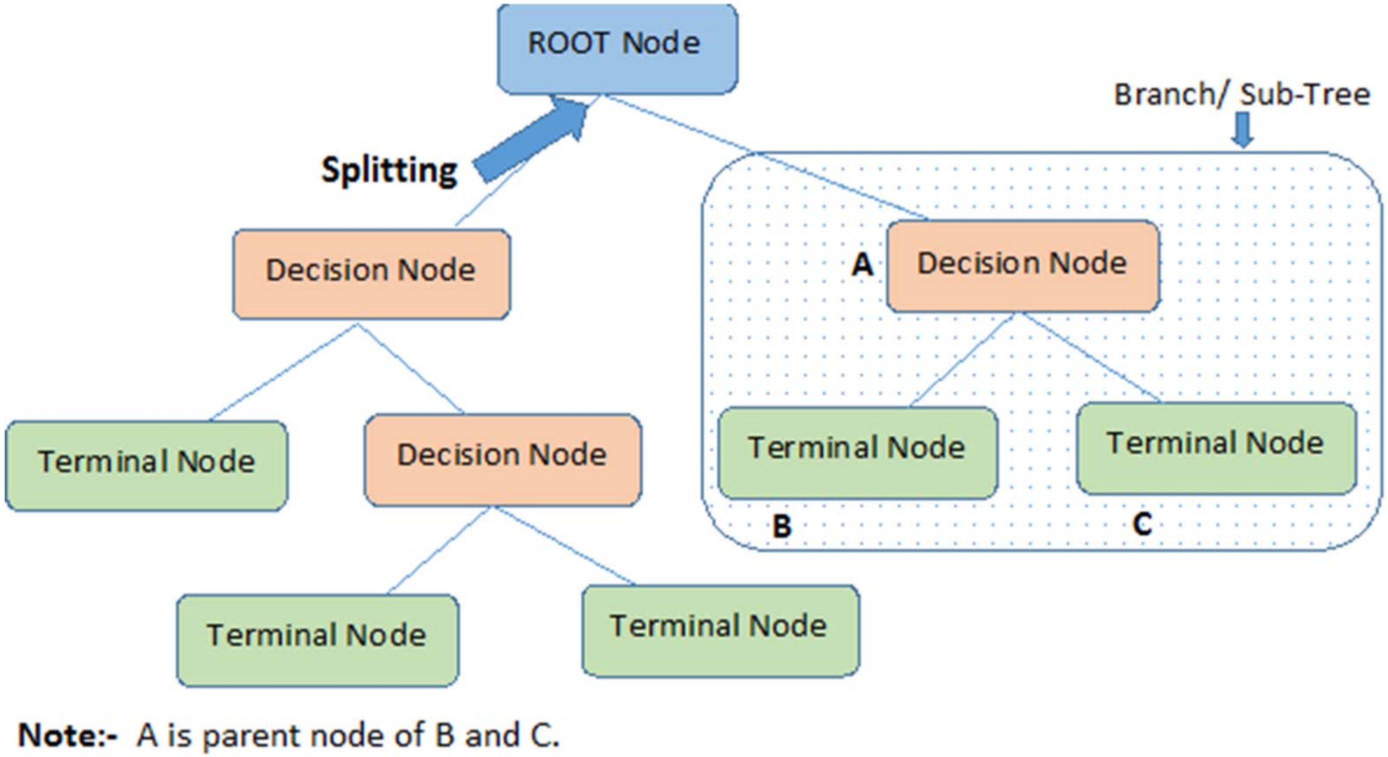
DECISION TREE & RANDOM FOREST

DECISION TREE



- 질문을 던지며 대상을 좁혀가는 스무고개와 비슷
- 결정에 다다르기 위해 Y/N 질문을 이어나가며 학습
- Decision Tree를 학습한다는 것은 정답에 가장 빨리 도달하는 Y/N 질문 목록을 학습한다는 것
- 데이터를 가장 잘 구분할 수 있는 Decision Tree를 생성

DECISION TREE 용어



GROWING A DECISION TREE

- Decision Tree를 만드는 알고리즘이 필요
- 어떻게 해야 가장 잘 분류할 수 있는가? = 어떤 Attribute를 선택해서 나눠야 하는가?
- “어떤 Attribute가 더 확실한 정보를 제공하고 있는가?” 로 분기 Attribute를 선택
- 분기 기준 (Split Criterion)
  -Categorical Response : Gini Index, Entropy , $\chi^2 - statistics$
  -Numerical Response : RSS

CART Algorithm

- 한번에 하나의 변수를 사용
- Binary decision Rule
- Numerical attribute : test whether value is greater or less than constant.
- Nominal attribute : test whether value belongs to subset.
- 불순도(impurity) 를 사용한 Greedy search

CART Algorithm’s split idea

- Parent Node 보다 Child Node의 impurity가 낮은 split을 선택
- 가능한 split들 중에서 impurity의 개선 수준이 높은 attribute를 선택

Why Binary decision rule

- 2개 이상의 그룹으로 분리할 수도 있지 않을까?
- Multiway split을 하면, 다음 단계에 insufficient data가 남을 수도 있음
- Multiway split은 a series of binary splits을 통해 얻을 수 있음

Greedy Search

- 가능한 모든 경우의 수를 따져보아 최선의 방법을 찾아냄
- 연속형 변수 : unique한 관찰값 m개 → m-1회의 greedy search
- 범주형 변수 : q개의 범주 → $2^{q-1} - 1$
- 불순도 개선이 가장 높은 것을 선택

Gini Index(Impurity)

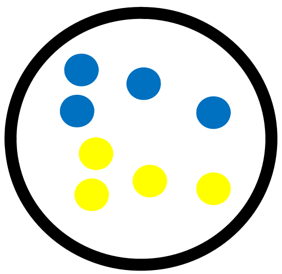
- CART 알고리즘의 Split Criterion

$$Gini\ index = \sum_{i=1}^{m} p_i(1 - p_i) = 1 - \sum_{i=1}^{m} p_i^2$$

- Gini 값이 가장 작은 분류를 선택

Gini Index(Impurity)

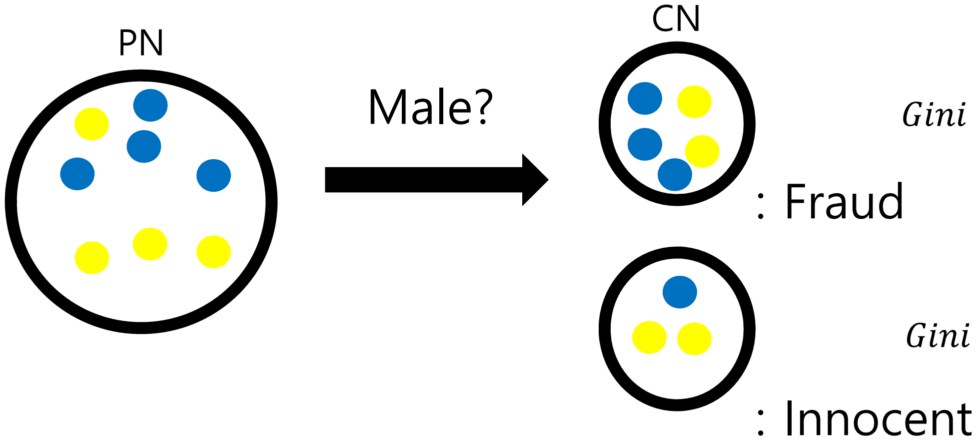
PN



$$Gini\ index = 1 - \left(\frac{4}{8}\right)^2 - \left(\frac{4}{8}\right)^2 = 0.5$$

: Fraud

: Innocent

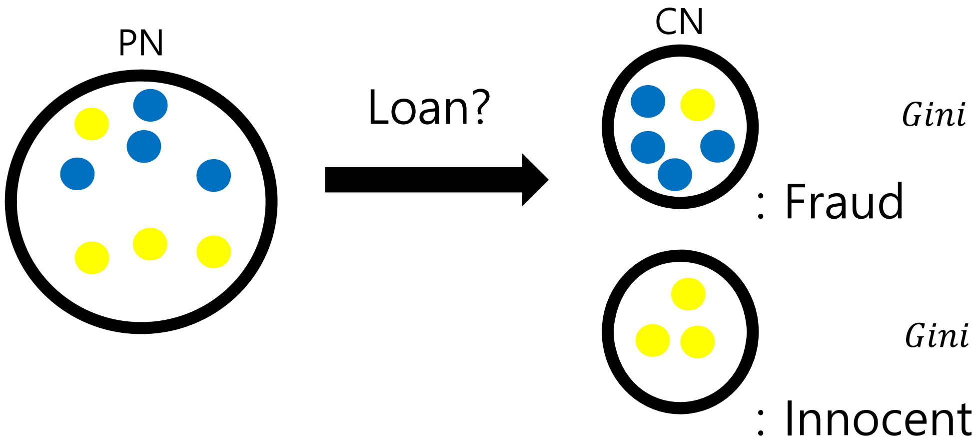


$$Gini\ index = 1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2 = 0.48$$

$$Gini\ index = 1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2 = 0.44$$

$$weighted\ average\ = 0.48 * \frac{5}{8} + 0.44 * \frac{3}{8} = 0.465$$

$$Goodness\ of\ split\ = 0.5 - 0.465 = 0.035$$

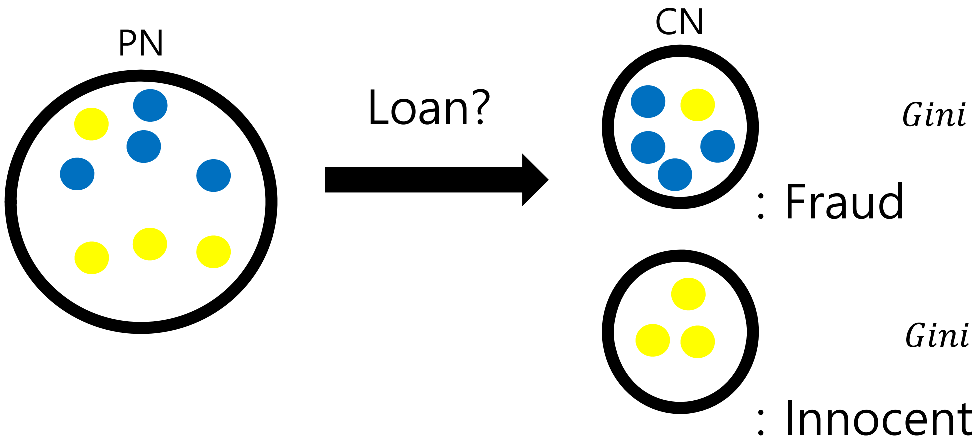


$$Gini\ index = 1 - \left(\frac{4}{5}\right)^2 - \left(\frac{1}{5}\right)^2 = 0.32$$

$$Gini\ index = 1 - \left(\frac{3}{3}\right)^2 = 0$$

$$weighted\ average = 0.32 * \frac{5}{8} + 0 * \frac{3}{8} = 0.2$$

$$Goodness\ of\ split = 0.5 - 0.2 = 0.3$$



$$Gini\ index = 1 - \left(\frac{4}{5}\right)^2 - \left(\frac{1}{5}\right)^2 = 0.32$$

$$Gini\ index = 1 - \left(\frac{3}{3}\right)^2 = 0$$

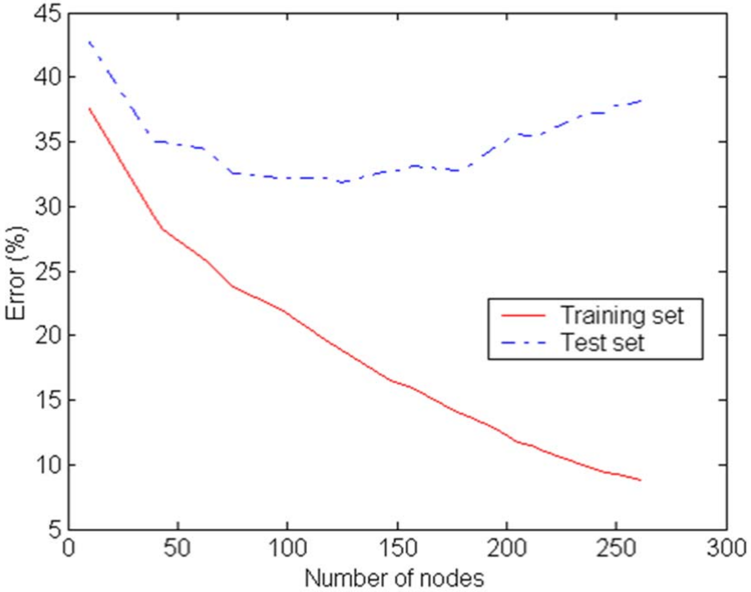
$$weighted\ average\ = 0.32 * \frac{5}{8} + 0 * \frac{3}{8} = 0.2$$

$$Goodness\ of\ split\ = 0.5 - 0.2 = 0.3$$

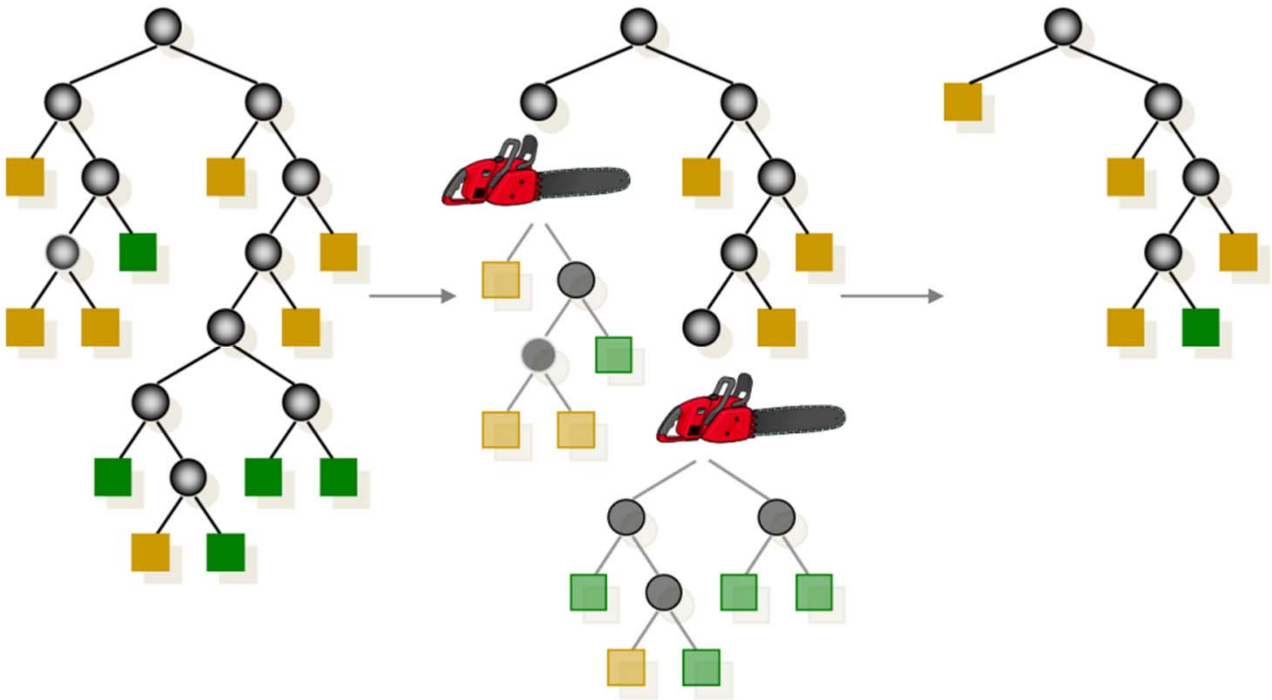
Better Split !

Overfitting

- 모든 terminal node의 순도 100% → Fully grown tree
- 지나친 split은 training data에 overfitting 될 수 있음
- Training data 오분류는 감소
- test data 오분류는 감소하다가 증가



Pruning



Pruning

- 사전 가지치기(pre-pruning)
  - 트리 성장 작업 중 실시
  - 어느 정도 불순한 Leaf node가 있어도 성장 조기 종료
  - 트리의 최대 깊이(max_depth), Leaf node의 최대 개수를 제한
  - Node 분할을 위한 최소 포인트 개수를 지정

- 사후 가지치기(post-pruning)
  - 트리 성장 작업 완료 후 실시
  - 완성된 트리에서 일부 Leaf node를 제거

Cost Complexity pruning

- 비용함수를 최소화 시키는 split을 찾아내어 pruning을 결정

- Cost complexity function

$$CC(T) = Err(T) + \alpha \cdot L(T)$$

- CC(T) : Decision tree의 cost complexity (오류가 적으면서 terminal node의 수가 작은 것이 작은 값을 가진다)
- Err(T) : test data에 대한 오분류율
- L(T) : Terminal node의 수
- α : 가중치 (0.01~0.1)

Cost Complexity pruning

- 비용함수를 최소화 시키는 split을 찾아내어 pruning을 결정

- Cost complexity function

$$CC(T) = Err(T) + \alpha \cdot L(T)$$

- 새로운 split을 함으로써 생기는 error 감소 이득이 penalty 증가보다 크지 못하면 더 이상 split을 하지 않음

CHAID

- 이산형 목표변수일 때, 카이제곱 검정
- 연속형 목표변수일 때, F-검정
- multiway split을 수행

CHAID

- 카이제곱 통계량 : 관측도(frequency)로 이루어진 분할표(contingency table)로 계산

	범주1	범주2	…	범주c	계
범주1	f_{11}	f_{12}	…	f_{1c}	f_{1+}
범주2	f_{21}	f_{22}	…	f_{2c}	f_{2+}
…	…	…	…	…	…
범주r	f_{r1}	f_{32}	…	f_{rc}	f_{r+}
계	f_{+1}	f_{+2}	…	f_{+c}	f_{++}

CHAID

- Pearson 카이제곱 통계량

$$\chi^2 = \sum_{i,j} \frac{(f_{ij} - e_{ij})^2}{e_{ij}}$$

- 통계량 자유도는 (r-1)(c-1)

- e_{ij} 분포의 동일성 또는 독립성 가설 하에서 계산된 기대도수 $e_{ij} = \frac{f_{i+} * f_{+j}}{f_{++}}$

CHAID

- 카이제곱 통계량이 자유도에 비해 매우 작다 → 예측변수의 각 범주에 따른 목표변수 분포가 서로 동일, 따라서 예측변수가 목표변수 분류에 영향을 주지 않음

- 자유도에 비해 카이제곱 통계량이 크고 작음은 p-value로 표현 가능, 카이제곱 통계량이 자유도에 비해 작으면 p-value ↑

- Split criterion을 카이제곱 통계량으로 하는 것 = p-value값이 가장 작은 예측변수와 그 때의 최적 split에 의해서 child node를 형성시키는 것

DECISION TREE 특징

- 간단하고 직관적인 결과 표현이 가능, 인간의 의사결정 과정과도 유사
- Numerical, Categorical variable 모두를 다룰 수 있음
- 질적 변수를 더미 변수를 생성하지 않고 쉽게 다룰 수 있음
- Attribute의 scaling이 필요 없음
- 관측치의 절대값이 아닌 순서가 중요 → Outlier에 Robust

DECISION TREE 특징

- Missing values 쉽게 다룰 수 있음

Categorical predictor : “missing " 이란 make a new category

Surrogate split : 대체 변수(surrogate variables) 사용하여 split 진행

- Greedy 알고리즘 → 부분 최적화, Global optimization 아닌 local optimization

DECISION TREE 특징

TABLE 10.1. *Some characteristics of different learning methods. Key:* ▲*= good,* ◆*=fair, and* ▼*=poor.*

Characteristic	Neural Nets	SVM	Trees	MARS	k-NN, Kernels
Natural handling of data of "mixed" type	▼	▼	▲	▲	▼
Handling of missing values	▼	▼	▲	▲	▲
Robustness to outliers in input space	▼	▼	▲	▼	▲
Insensitive to monotone transformations of inputs	▼	▼	▲	▼	▼
Computational scalability (large N)	▼	▼	▲	▲	▼
Ability to deal with irrelevant inputs	▼	▼	▲	▲	▼
Ability to extract linear combinations of features	▲	▲	▼	▼	◆
Interpretability	▼	▼	◆	▲	▼
Predictive power	▲	▲	▼	◆	▲

DECISION TREE 특징

- Pruning을 해도 overfitting을 완벽하게 해결하지 못함
- 다른 방법들에 비해 예측 정확도가 떨어짐

Instability of Trees

- 데이터가 조금만 바뀌어도 split이 상당히 많이 달라질 수 있음 (instability)
- Tree는 high variance 이고 이는 tree 방법의 major problem
- Major reason for instability is the hierarchical nature of the process
- 상위 split에서의 effect of an error가 그 아래 모든 split으로 전파됨

그래서 tree'들'을 모아 예측 정확도를 높일 수 있는 방법을 알아보자

Ensemble Method

- 여러가지 Algorithm을 모아 성능을 향상
- 그 결과 Better Accuracy & More Stability
- Bias가 크다 :실제 값과 예측 값의 차이가 크다.
- Variance가 크다 : 다른 데이터셋에서 모델 적합이 일관적이지 못하다.

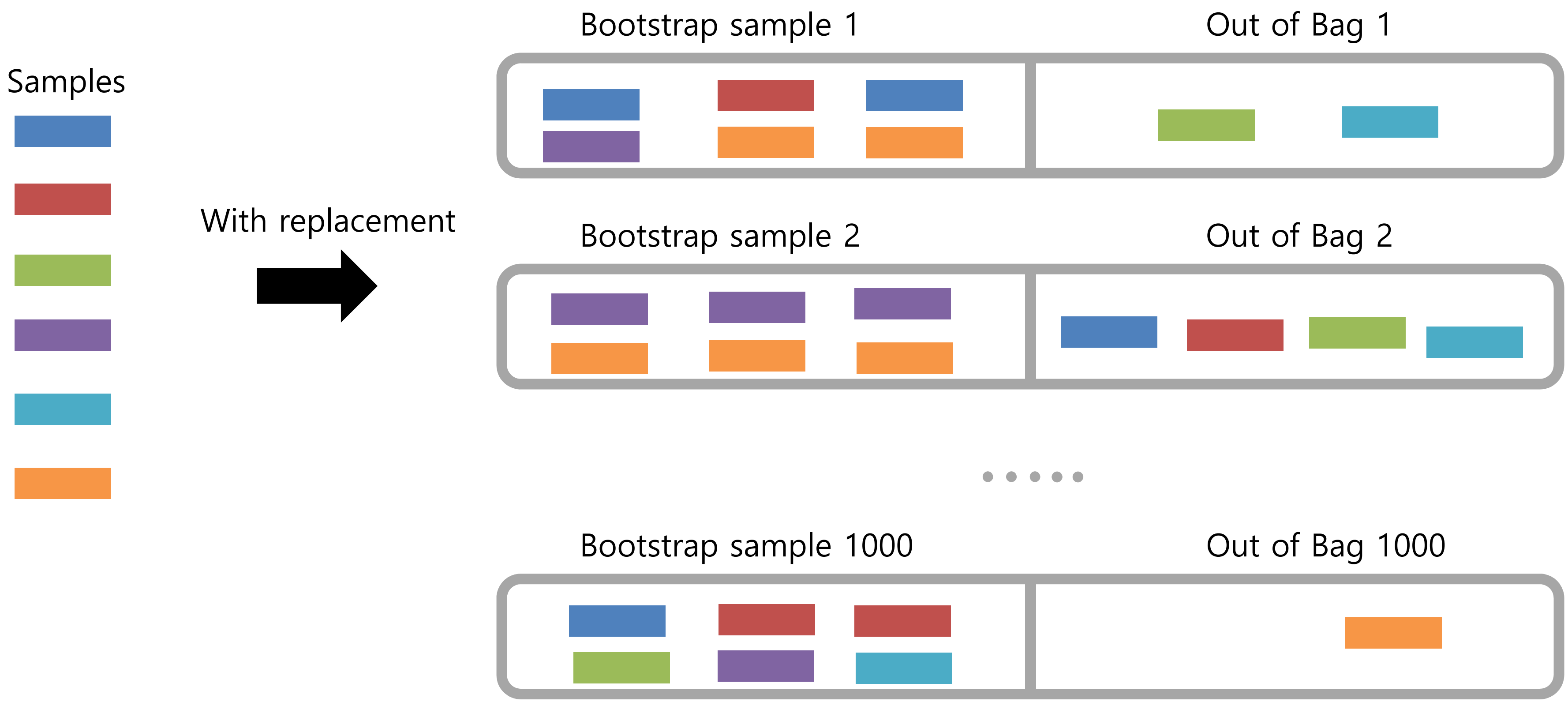
Ensemble Method

- Decision Tree가 충분히 deep하게 성장하면, Low Bias, High Variance한 모델
- 여러 개의 모형을 만들고 이에 대한 평균을 구함으로써 분산을 줄일 수 있다.
- Bagging, Boosting ...

Bagging / Boosting

- Bagging : Bootstrap을 통해 생긴 개별적인 data set에 각 개별적인 tree 생성
  - 따라서 각 tree가 만들어지는 과정은 다른 tree형성에 영향을 주지 않음
  - Parallel Ensemble
- Boosting : 이전 tree의 정보를 이용하여 tree를 순차적으로 생성 (Bootstrap X)
  - 각 tree 원래의 training data에서 이전 tree를 토대로 수정된 new training data set에 적합
  - Sequential Ensemble

Bootstrapping



0.632 Bootstrap

- N개의 전체 데이터에서 N번 데이터를 복원 추출

- 각 데이터가 나타날 확률은 0.632

$$\Pr\{observation\ i\ \in bootstrap\ sample\ b\}$$
$$= 1 - \prod_{i=1}^{N}\left(1 - \frac{1}{N}\right) = 1 - \left(1 - \frac{1}{N}\right)^{N} \simeq 1 - e^{-1} = 0.632$$

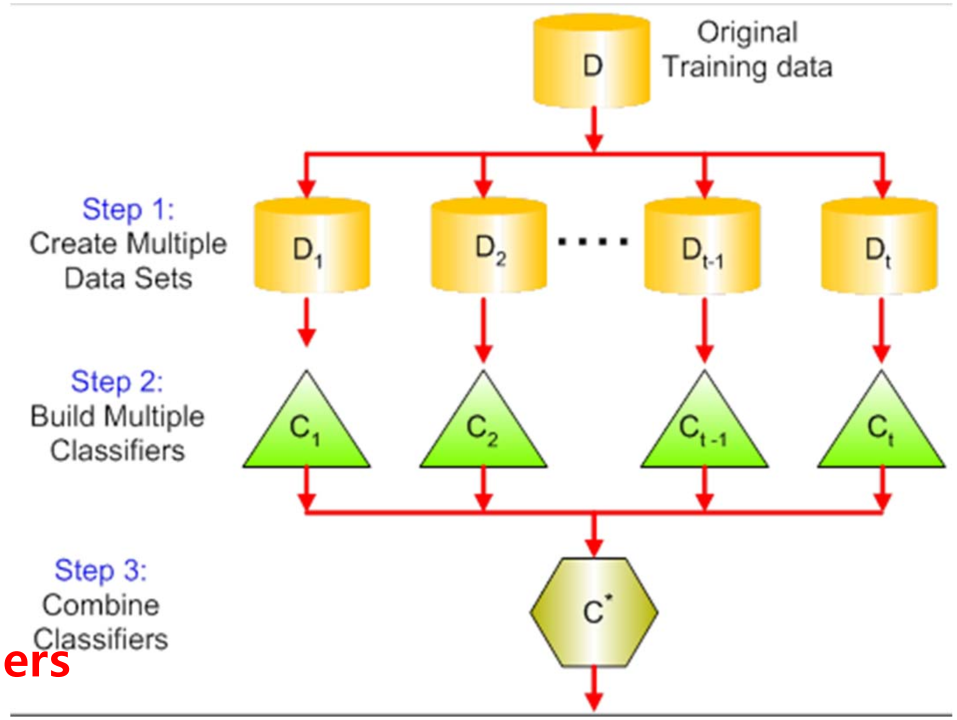
Bagging

Bootstrap 하고 합친다(Aggregating) →Bagging

Bootstrapping

Build Classifiers

Combine Classifiers



Why Bagging works?

- Recall that given a set of n independent observations $Z_1, Z_2, \ldots Z_n$ each with variance σ^2

$$Var(\bar{Z}) = \frac{\sigma^2}{\mathrm{n}}$$

- 따라서 우리는 a set of observations의 averaging 이 variance 를 감소시킨다는 걸 알 수 있음

Why Bagging works?

- 통계적 학습 방법론의 variance를 감소시키는 방법은 다음과 같을 것
- Take many training sets of data from the population
- Build a separate prediction model using each training data
- Vote(Average) every prediction

Bagging

- Training set에서 Bootstrap을 이용하여, B개의 서로 다른 bootstrapped training data sets을 만든다.

- b번째 bootstrapped training set에서 만든 모델을 $\hat{f}^{*b}(x)$ 라 한다면 bagging은 다음과 같이 표현

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$$

Bagging

- 이렇게 Bagging을 통해 만들어진 tree는 서로 비슷한 모델일 것 (split 기준이 비슷)
- 따라서 생성된 모델들 간의 Correlation 값 ↑
- Bagging이 의도만큼 Variance를 줄이지 못할 수 있다. Why?

Bagging

$$\hat{f_{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} f_b(x)$$

for given x

$$\begin{aligned}
Var(\hat{f_{bag}}(x)) &= Cov\left(\frac{1}{B}\sum_{i=1}^{B} f_i(x), \frac{1}{B}\sum_{j=1}^{B} f_j(x)\right) \\
&= \left(\frac{1}{B}\right)^2 \sum_{i=j} Var(f_i(x)) + \left(\frac{1}{B}\right)^2 \sum_{i\neq j} Cov(f_i(x), f_j(x))
\end{aligned}$$

Suppose $Var(f_1(x)) =, ..., = Var(f_B(x))$ and $Cov(f_i(x), f_j(x))$ are equal for $\forall$ pair of (i,j) with i $\neq$ j

$$\begin{aligned}
&= \left(\frac{1}{B}\right) Var(f_1(x)) + \left(\frac{1}{B}\right)^2 \times \binom{B}{2} Cov((f_1(x), f_2(x)) \\
&= \left(\frac{1}{B}\right) Var(f_1(x)) + \left(\frac{1}{B}\right)^2 \times \frac{B(B-1)}{2} Cov((f_1(x), f_2(x))
\end{aligned}$$

Let $Var(f_1(x)) = \sigma^2$ and $Cov((f_1(x), f_2(x)) = \rho\sigma^2$

$$= \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

Random Forest

- 분산이 σ^2인 B개의 i.i.d random variables의 평균값의 분산 : $\frac{1}{B}\sigma^2$

- Variables이 i.d(identically distributed, not independent)하고 correlation 값이 ρ일 때, 평균값의 분산

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

- B ↑, second term ↓

- Bagged tree의 correlation값이 양수라면, 평균 내는 것의 효과를 감소

Random Forest's Idea

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

- Tree들간의 correlation을 줄임으로써 bagging의 variance reduction을 향상시키는 것.

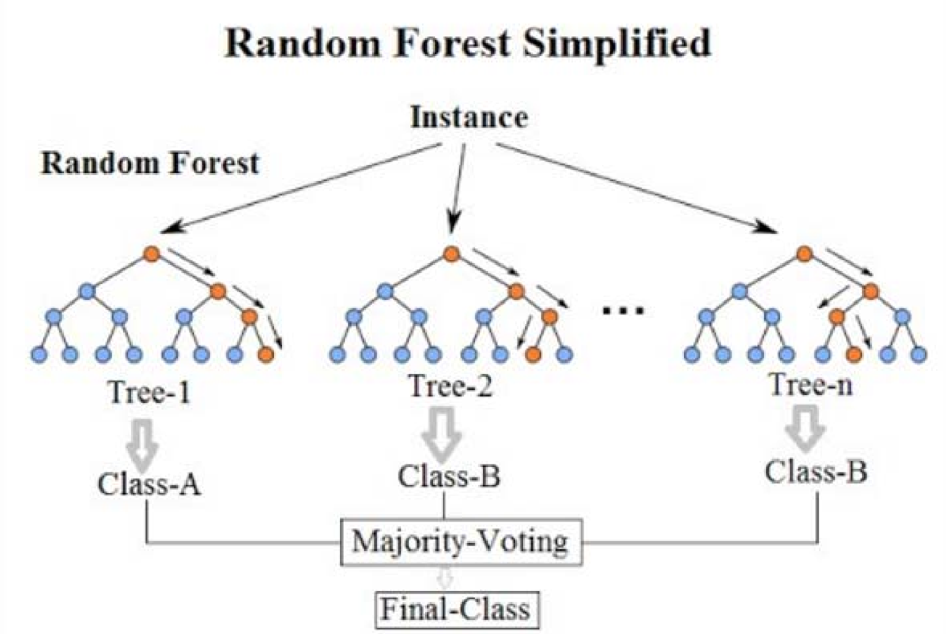
- 이는 변수를 random select하며 tree를 성장시키는 과정을 거치면서 달성

Random Forest

- Random Forest는 tree간의 decorrelate
- 똑같이 Bootstrap된 데이터를 통해 tree를 생성하지만, 전체 p개의 변수들 중 Random하게 m개의 변수만을 선택, 이를 고려하여 split 기준을 설정.
- 또 다음 split을 만들 때, 다시 m개의 변수를 랜덤하게 선택하여 split
- $m = \sqrt{p}\ (Classification) \quad m = \frac{p}{3}(Regression)$
- Dimension Reduction 효과

Random Forest

다양한(Correlation이 낮은) Tree를 생성



Classification : Majority-Voting / Regression : Average 값

Variable Importance

- OOB 샘플을 이용하여 각 Tree별 오류율을 계산
- OOB 샘플 데이터를 각 변수 별로 permutation한 후, 각 Tree별 오류율 계산
- 오류율 차이가 클수록 해당 변수가 Tree에서 중요한 역할
- 모든 Tree에서 오류율 차이를 평균내어 Variable Importance 구함.

Variable Importance

Y	X1	X2	X3	X4
1	5	1	1	2
0	5	4	5	7
0	3	1	1	2
1	6	8	1	3
0	4	1	3	2

$\hat{Y}$

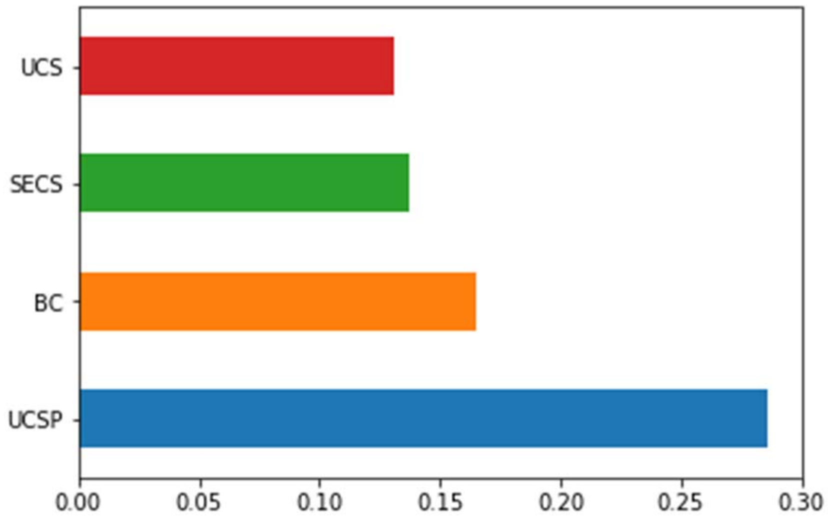
X1 변수를 permutation

Y	X1	X2	X3	X4
1	3	1	1	2
0	5	4	5	7
0	5	1	1	2
1	4	8	1	3
0	6	1	3	2

$\hat{Y}$

두 값의 차이가 크다면 X1의 중요도 ↑

Variable Importance plot



- Ensemble Method는 Black Box
- Variable Importance plot은 부분적 해석을 가능하게 함
- 그러나 random select 변수 수 결정에 따라 그림이 달라질 수 있음
- Python Code에서 random_state 조건에 따라서도 달라짐