

의사결정 트리

(decision tree)

2조 팀원 소개

김규진

김민주

장은조



목차

1

의사결정나무

2

기출문제로

개념잡기!

3

R코드

해석 및 실습

의사결정나무란?

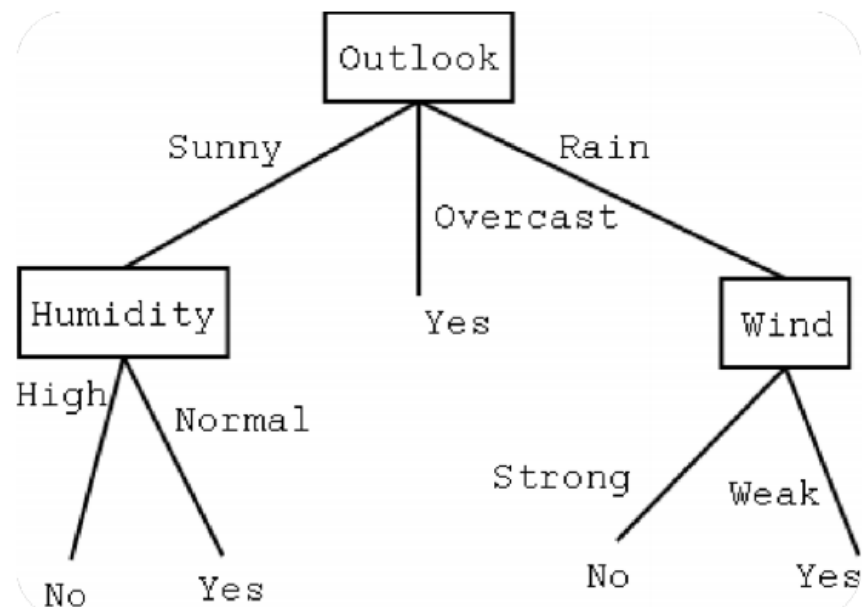
- 의사결정규칙을 도표화하여 관심대상이 되는 집단을 몇 개의 소집단으로 분류(classification)하거나 예측(prediction)을 수행하는 분석방법
- 의사결정나무는 분류 또는 예측을 목적으로 하는 어떤 경우에도 사용될 수 있으나 분석의 정확도보다는 분석과정의 설명이 필요한 경우에 더 유용하게 사용된다.

의사결정나무 형성 → 가지치기 → 타당성 평가 → 해석 및 예측

1. 의사결정나무



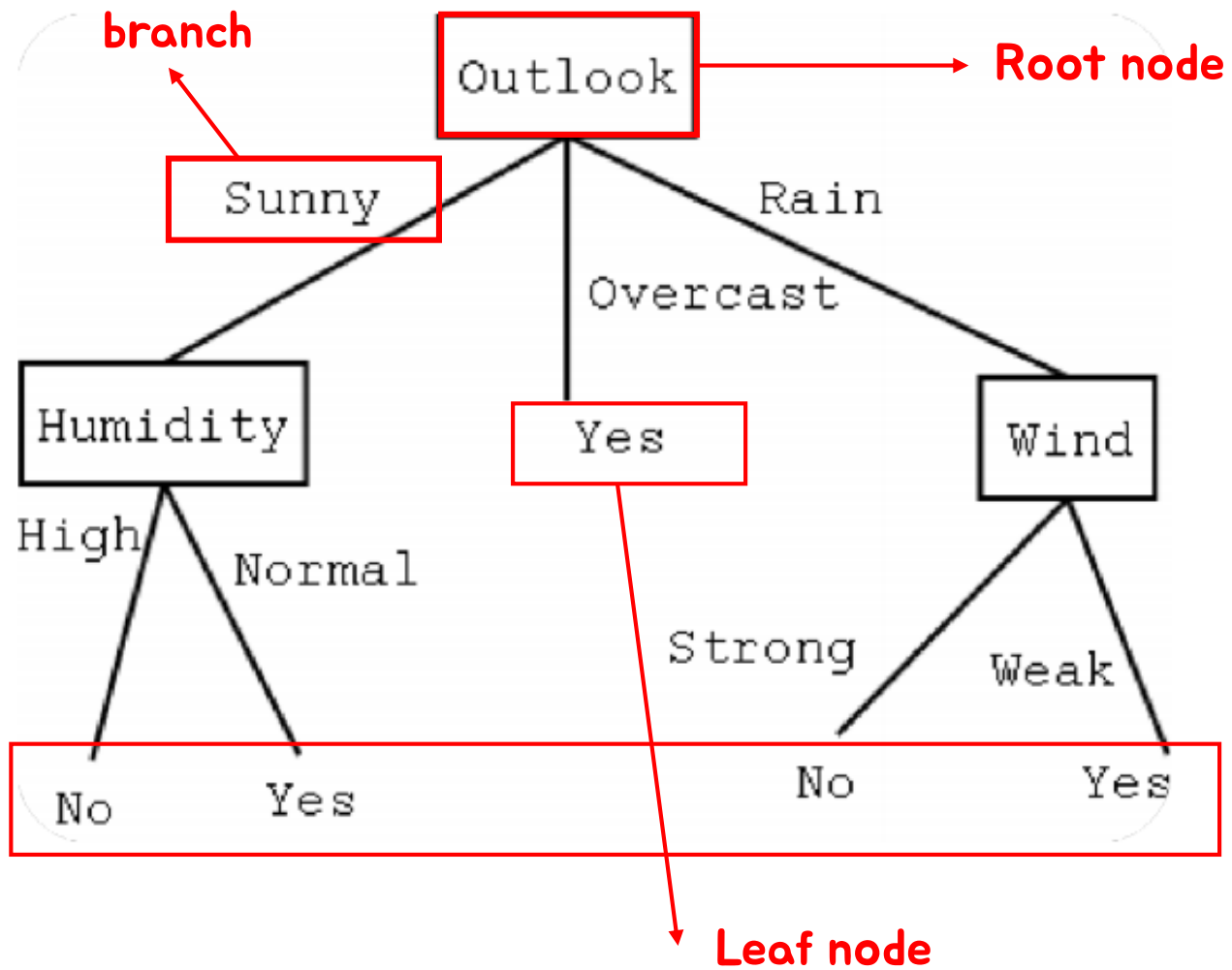
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



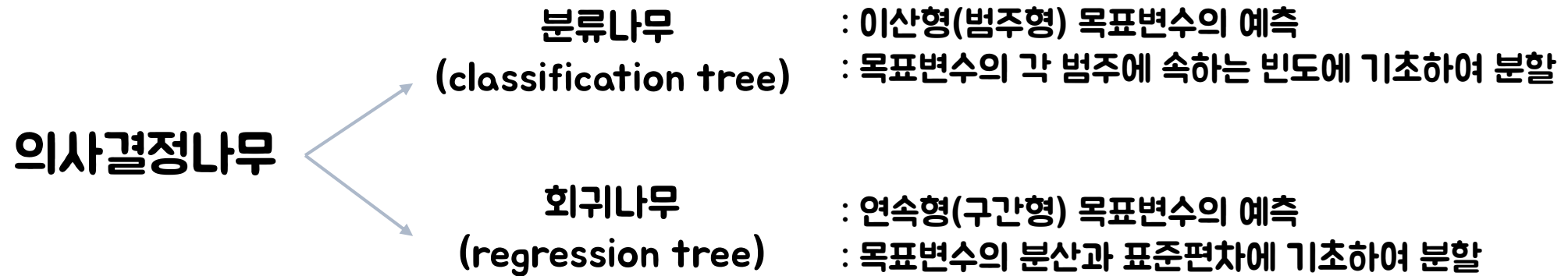
의사결정나무 용어

Node 분류의 기준이 되는 변수가 위치하는 곳

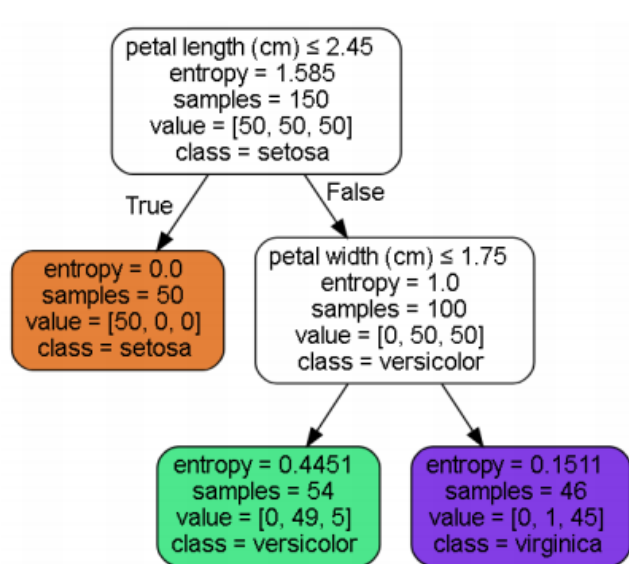
- Parent node 상대적인 개념. 상위 노드
- Child node 하위 노드
- Root node 상위노드가 없는 가장 위의 노드
- Leaf node 하위노드가 없는 가장 아래의 노드
- Internal node Leaf node가 아닌 노드
- Branch 샘플을 분류하는 조건이 위치하는 곳
- Depth Root node에서 특정 노드까지 도달하기 위해 거쳐야 하는 branch의 수



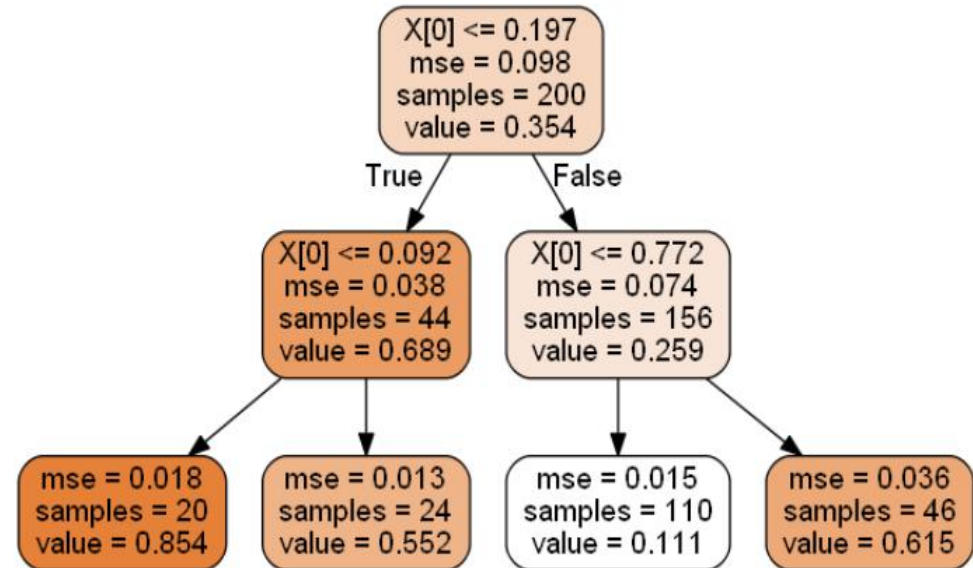
의사결정나무 종류



의사결정나무 종류



분류나무
(classification tree)

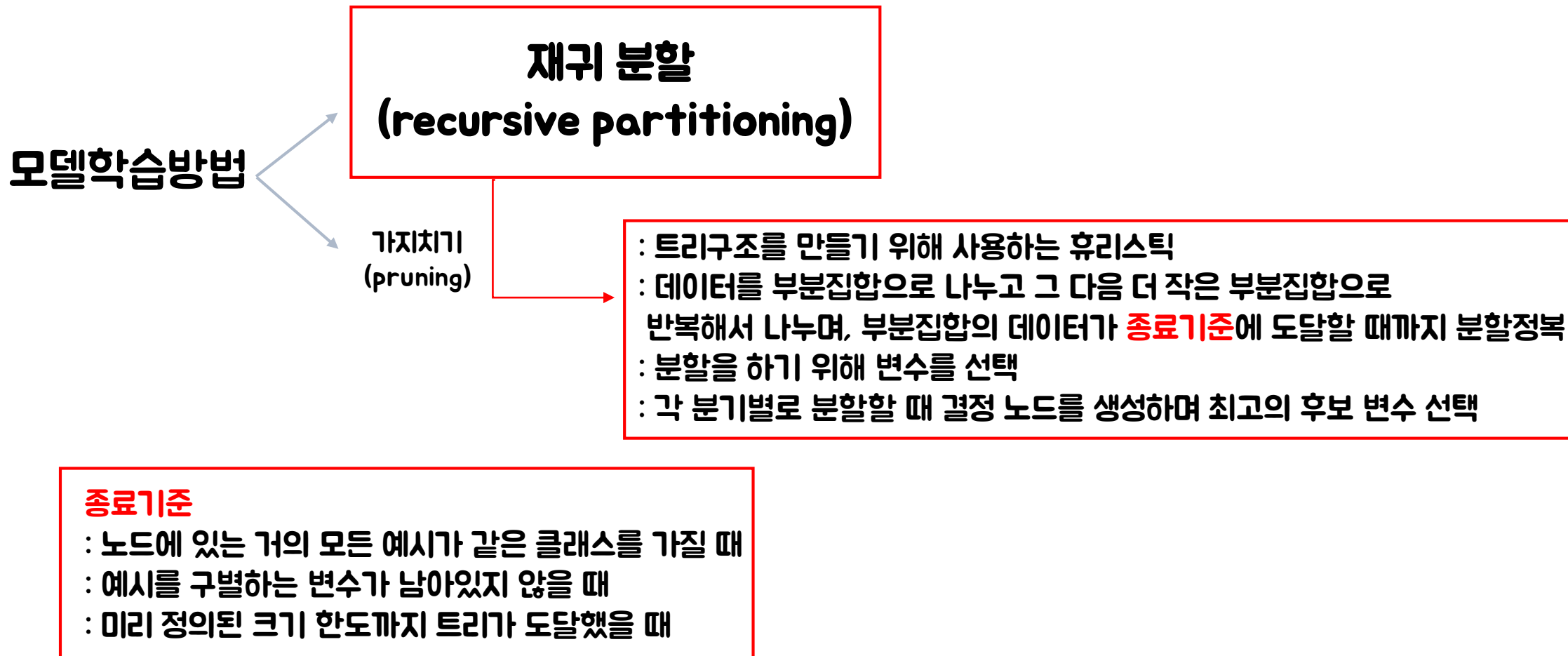


회귀나무
(regression tree)

의사결정나무 장단점

장점	단점
해석의 용이성 - 나무구조로 표현되어 모형을 사용자가 쉽게 이해할 수 있다	비연속성 - 연속형 변수를 비연속적 값으로 취급하기 때문에 분리의 경계점 부근에서 예측오류가 클 가능성이 있다
교호작용효과의 해석 - 두 개 이상의 변수가 결합하여 목표변수에 어떻게 영향을 주는지 쉽게 알 수 있다	비안정성 - 분석용자료(training data)에만 의존하므로 새로운 자료의 예측에서는 불안정할 가능성이 높다.
비모수적 모형 - 선형성, 정규성, 등분산성 등의 가정이 필요하지 않다.	

의사결정나무 방법



어떤 **기준 후보**에 대해 분할할 것인가?

변수에 대한 여러 개의 기준 후보 중에서 어떤 기준으로 분할 했을 때,
순도(purity)가 증가하고 불순도(impurity)가 감소하는 방향으로 재귀분할 진행

- 순도(purity) : 부분집합이 단일클래스를 포함하는 정도
- 순수(impurity) : 단일클래스로 이루어진 부분집합

순도 측정법에는 어떤것들이 있나요?

- 엔트로피 (Entropy) , 정보획득량
- 지니 지수 (Gini index)

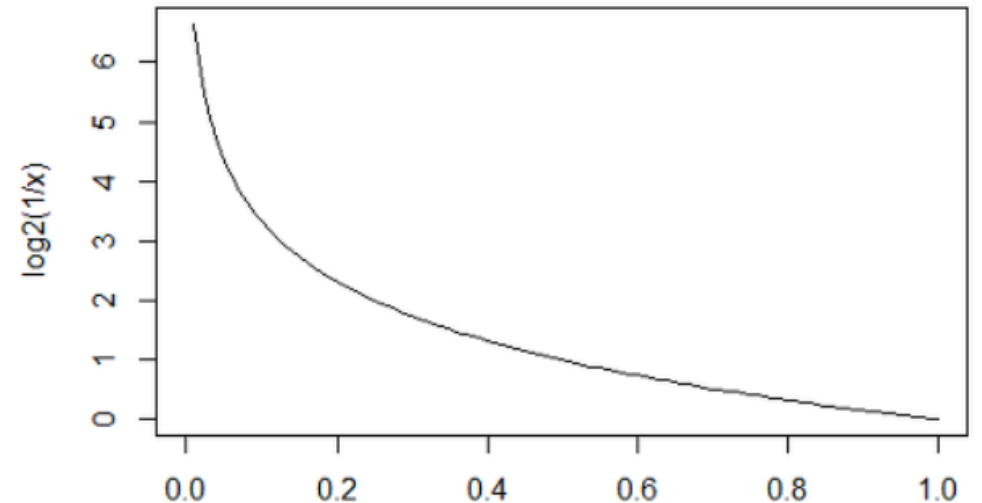
정보획득량

- : 어떤 사건이 얼마만큼의 정보를 줄 수 있는지를 수치화한 값
- : 정보획득량 계산하려면 정보함수와 엔트로피 개념을 알아야 한다!

정보함수

- : 정보의 가치를 반환하는데 발생할 확률이 작은 사건일수록 정보의 가치가 크고 반대로 확률이 큰 사건일수록 정보의 가치가 적다

$$I(x) = \log_2 \frac{1}{p(x)}$$



엔트로피 (Entropy)

- : 불순도를 측정하는 방법 중 하나
- : 엔트로피가 높으면 불순도가 높다는 것을 의미

쉽게 말해,
왼쪽식은 분할 전, 오른쪽식은 분할 후 엔트로피 식

$$Entropy(X) = - \sum_{j=1}^n P_j \log_2(P_j)$$

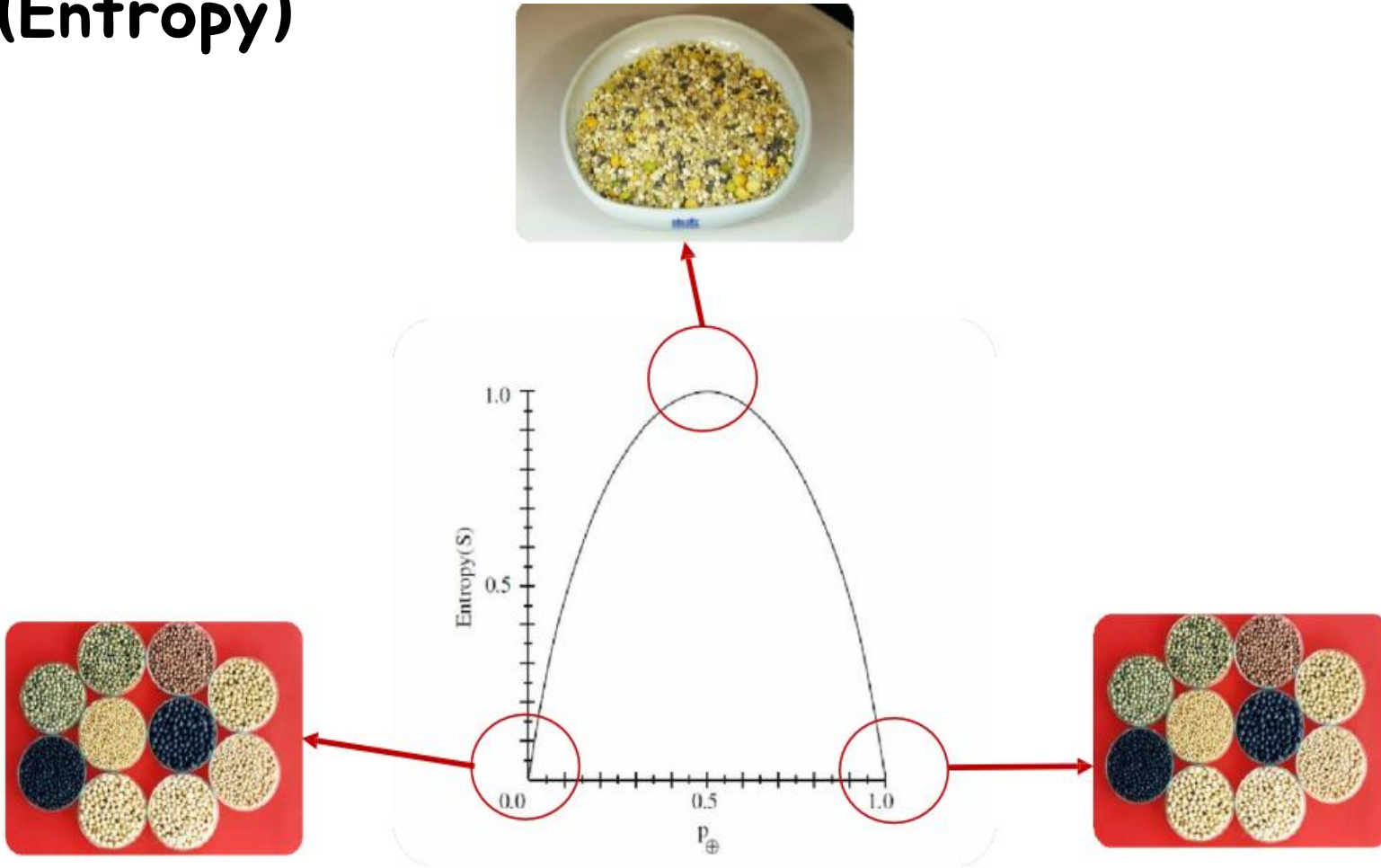
: 이때, P_j 는 X영역에 속하는 데이터 가운데
j범주에 속하는 데이터 비율

$$Entropy(X) = \sum_{i=1}^d R_i \left(- \sum_{j=1}^n P_j \log_2(P_j) \right)$$

: 이때, R_i 는 분할 전 데이터 가운데 분할 후
i범주에 속하는 데이터 비율

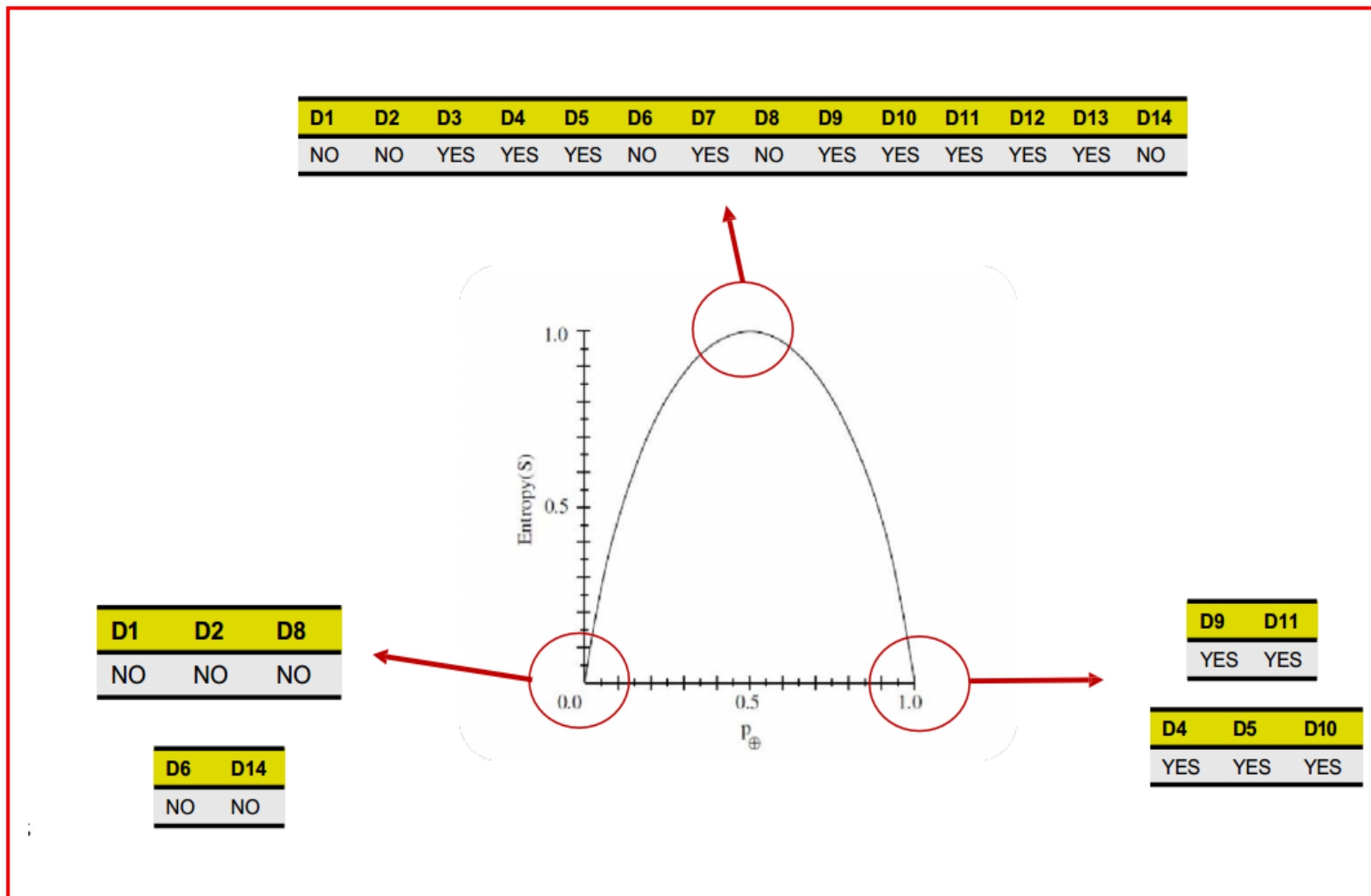
엔트로피가 낮게 만드는 것이 목적 !!!

엔트로피 (Entropy)

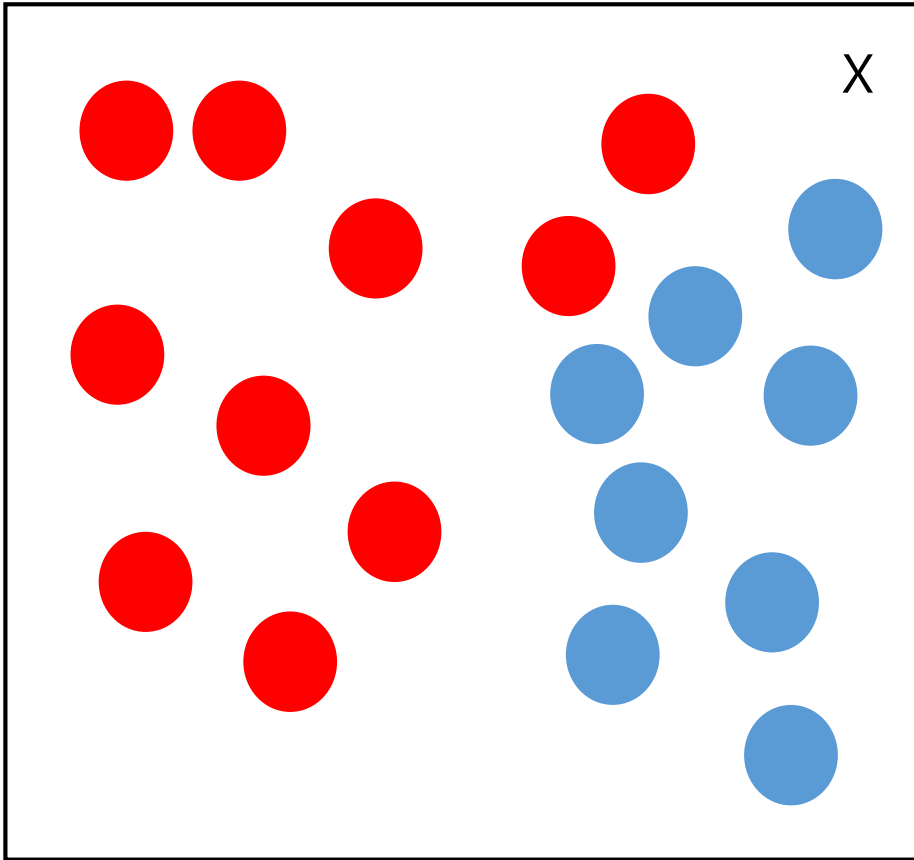


엔트로피 (Entropy)

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



엔트로피 (Entropy)



$$Entropy(X) = - \sum_{j=1}^n P_j \log_2(P_j)$$

; 이때, P_j 는 X영역에 속하는 데이터 가운데 j범주에 속하는 데이터 비율

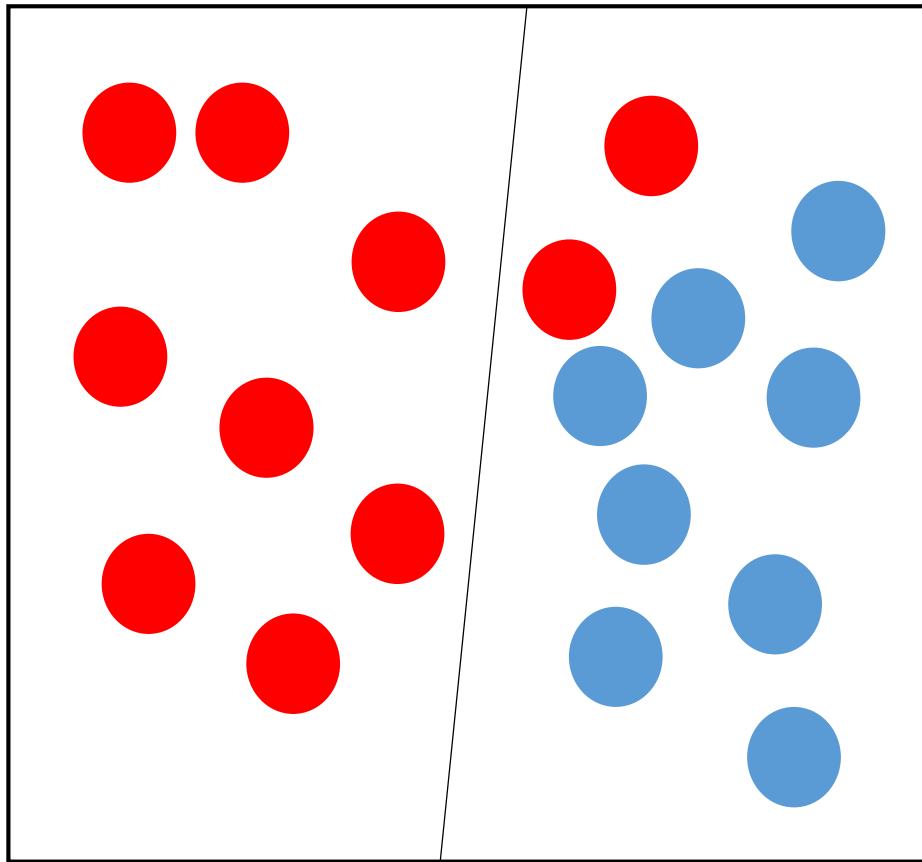
$$= - \frac{10}{18} \log_2\left(\frac{10}{18}\right) - \frac{8}{18} \log_2\left(\frac{8}{18}\right) = 0.991$$

P_1 : X영역에 속하는 전체 데이터 18개 가운데
첫번째 범주(빨간공)에 속하는 데이터의 비율

P_2 : X영역에 속하는 전체 데이터 18개 가운데
두번째 범주(파란공)에 속하는 데이터의 비율

엔트로피 (Entropy)

X



$$Entropy(X) = \sum_{i=1}^d R_i \left(- \sum_{j=1}^n P_j \log_2(P_j) \right)$$

; 이때, R_i 는 분할 전 데이터 가운데 분할 후 i 범주에 속하는 데이터 비율

$$= \frac{8}{18} (-1 \cdot \log_2(1)) + \frac{10}{18} \left(-\frac{2}{10} \cdot \log_2\left(\frac{2}{10}\right) - \frac{8}{10} \cdot \log_2\left(\frac{8}{10}\right) \right)$$

$$= 0.401$$

0.991 >> 0.401

- 엔트로피 감소
- 불순도 감소
- 잘 분류되었다

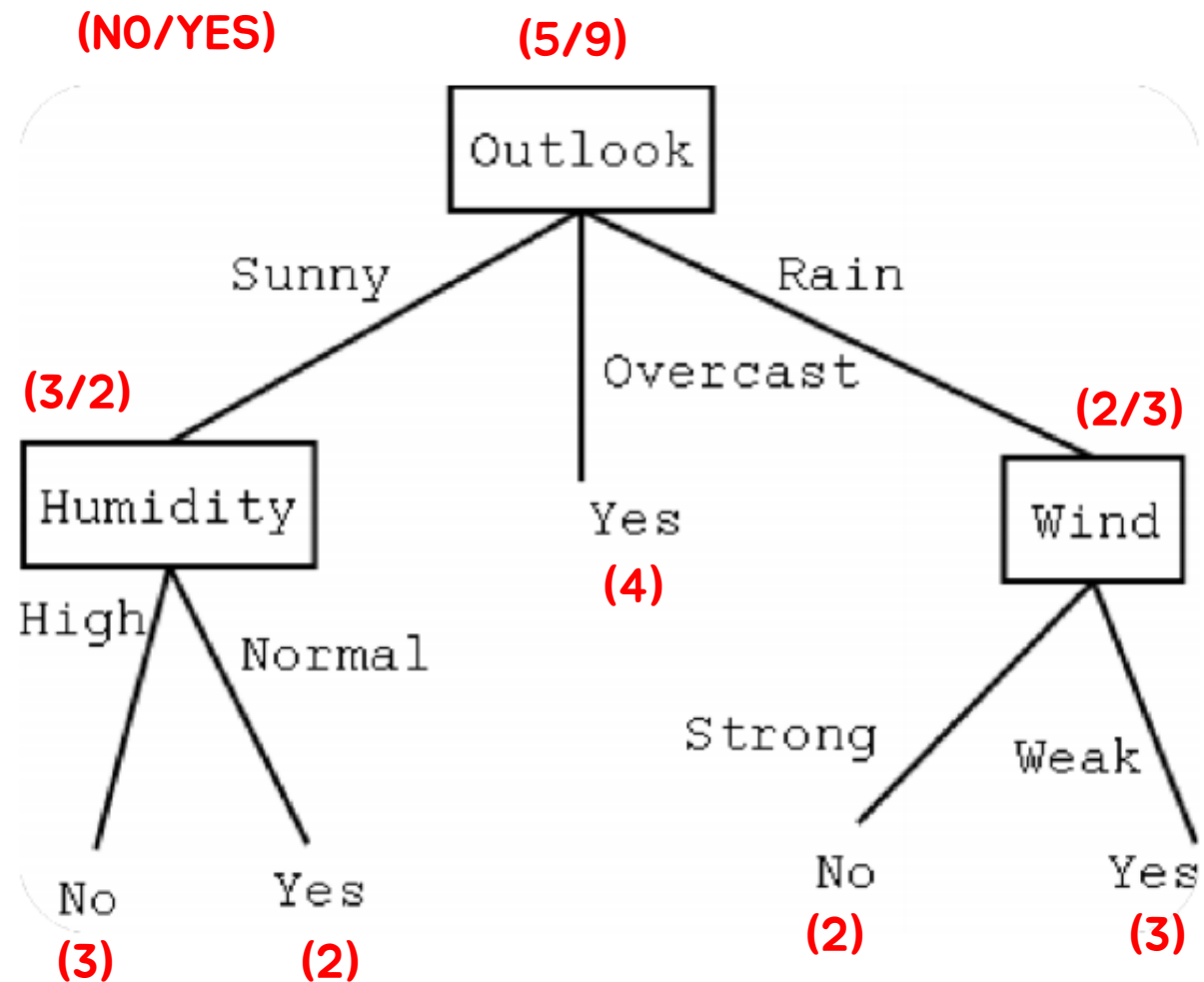
엔트로피 (Entropy)

Q1. 분류되기 전의 엔트로피를 구해보세요

$$Entropy(X) = - \sum_{j=1}^n P_j \log_2(P_j)$$

Q2. 분류된 후의 엔트로피를 구해보세요

$$Entropy(X) = \sum_{i=1}^d R_i \left(- \sum_{j=1}^n P_j \log_2(P_j) \right)$$



정보획득량 (information gain)

- : 어떤 사건이 얼마만큼의 정보를 줄 수 있는지를 수치화한 값
- : $\text{Entropy}(\text{before}) - \text{Entropy}(\text{after})$
- : 의사결정트리는 정보획득량을 최대로 하는 순서로 속성을 배치하는 것이 중요
- : 각각의 변수들 중 가장 큰 값을 가진 변수를 기준으로 분할을 진행한다.
- : 그리고 이러한 과정을 계속 반복하며, gain 값이 어느 정도 0에 가까워지면 계산을 멈춘다.

'그렇다면 **정보획득량**을 가지고 어떻게 **재귀분할** 하나요?'

1. 정보획득량이 가장 큰 방향으로 root node를 정한다

Ex). $Values(Wind) = Weak, Strong$

$$S = [9+, 5-]$$

$$S_{Weak} \leftarrow [6+, 2-]$$

$$S_{Strong} \leftarrow [3+, 3-]$$

$$\begin{aligned} Gain(S, Wind) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v) \\ &= Entropy(S) - (8/14) Entropy(S_{Weak}) \\ &\quad - (6/14) Entropy(S_{Strong}) \\ &= 0.940 - (8/14)0.811 - (6/14)1.00 \\ &= 0.048 \end{aligned}$$

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

$$Gain(S, Outlook) = 0.246$$

$$Gain(S, Humidity) = 0.151$$

$$Gain(S, Wind) = 0.048$$

$$Gain(S, Temperature) = 0.029$$

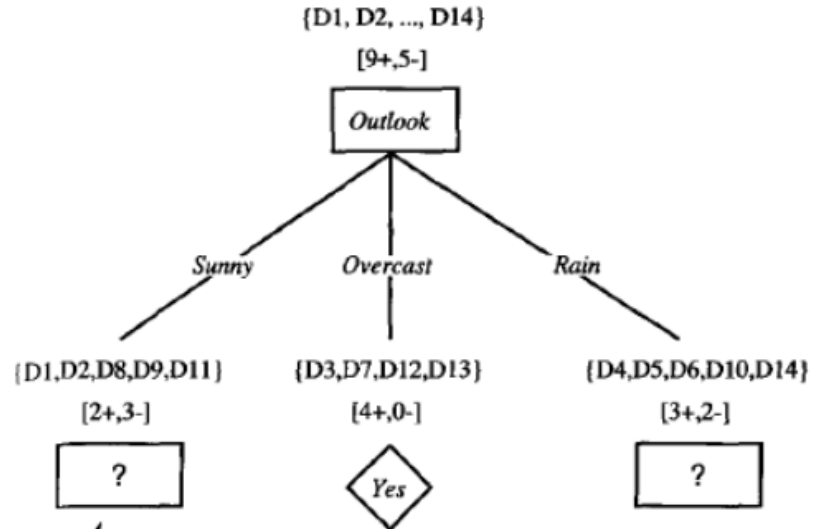
Outlook이 정보획득량이 가장 커서
Root node로 정함

1. 의사결정나무



2. Root node에서 분류된 데이터들의 entropy구하고
그 다음 노드의 attribute를 정보획득량이 큰 기준으로 정한다.

Ex).



Which attribute should be tested here?

$$S_{\text{sunny}} = \{D1, D2, D8, D9, D11\}$$

$$\text{Gain}(S_{\text{sunny}}, \text{Humidity}) = .970 - (3/5) 0.0 - (2/5) 0.0 = .970$$

$$\text{Gain}(S_{\text{sunny}}, \text{Temperature}) = .970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = .570$$

$$\text{Gain}(S_{\text{sunny}}, \text{Wind}) = .970 - (2/5) 1.0 - (3/5) .918 = .019$$

$$S_{\text{sunny}} = \{D1, D2, D8, D9, D11\}$$

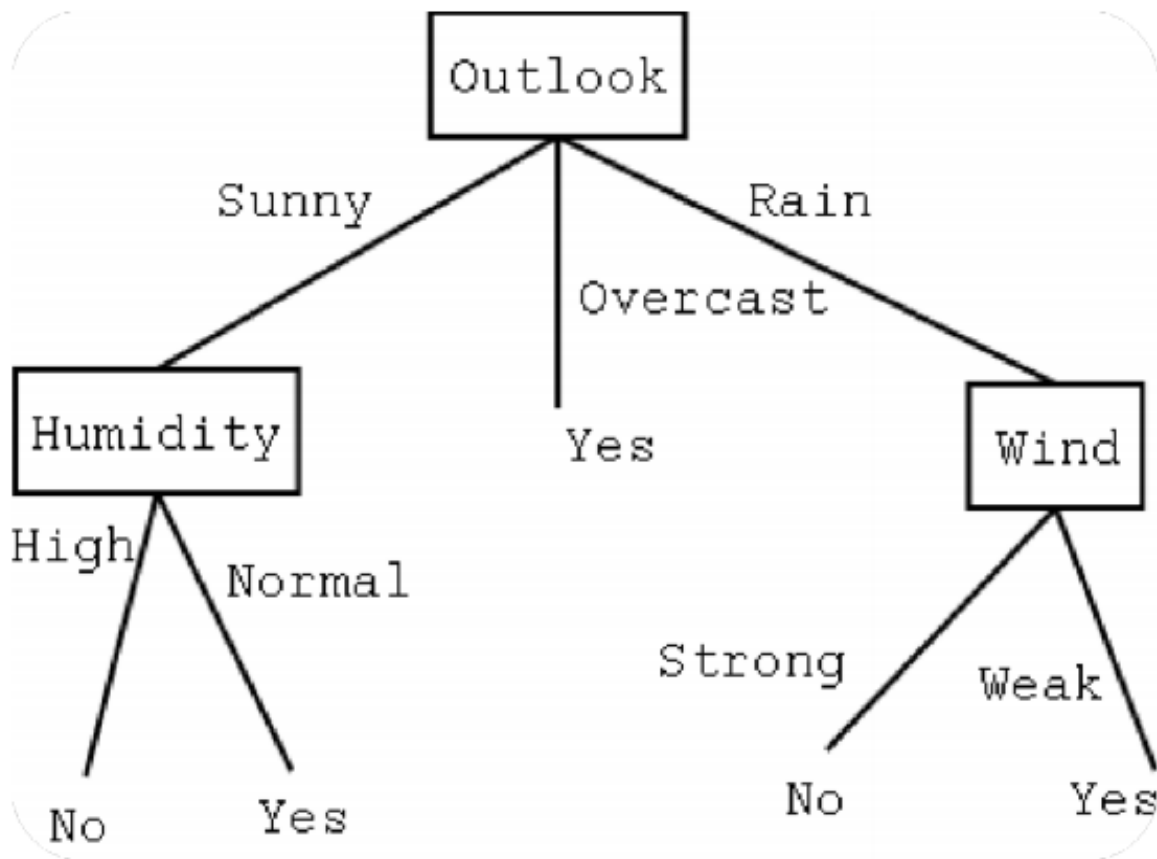
$$\text{Gain}(S_{\text{sunny}}, \text{Humidity}) = .970 - (3/5) 0.0 - (2/5) 0.0 = .970$$

$$\text{Gain}(S_{\text{sunny}}, \text{Temperature}) = .970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = .570$$

$$\text{Gain}(S_{\text{sunny}}, \text{Wind}) = .970 - (2/5) 1.0 - (3/5) .918 = .019$$

Humidity의 정보획득량이 가장 크므로
Sunny의 attribute는 humidity가 온다.

최종 트리



의사결정나무에 중요한 아이디어는 가장 큰 정보획득량을 가지는 attribute를 root에 가깝게 하는 것!

지니지수 (Gini index)

: 불순도를 측정하는 방법 중 하나

: 전체 노드 안에 특정 범주에 속하는 관측치의 비율을 모두 제외한 값

분할전

$$Gini(t) = 1 - \sum_i [p(i|t)]^2$$

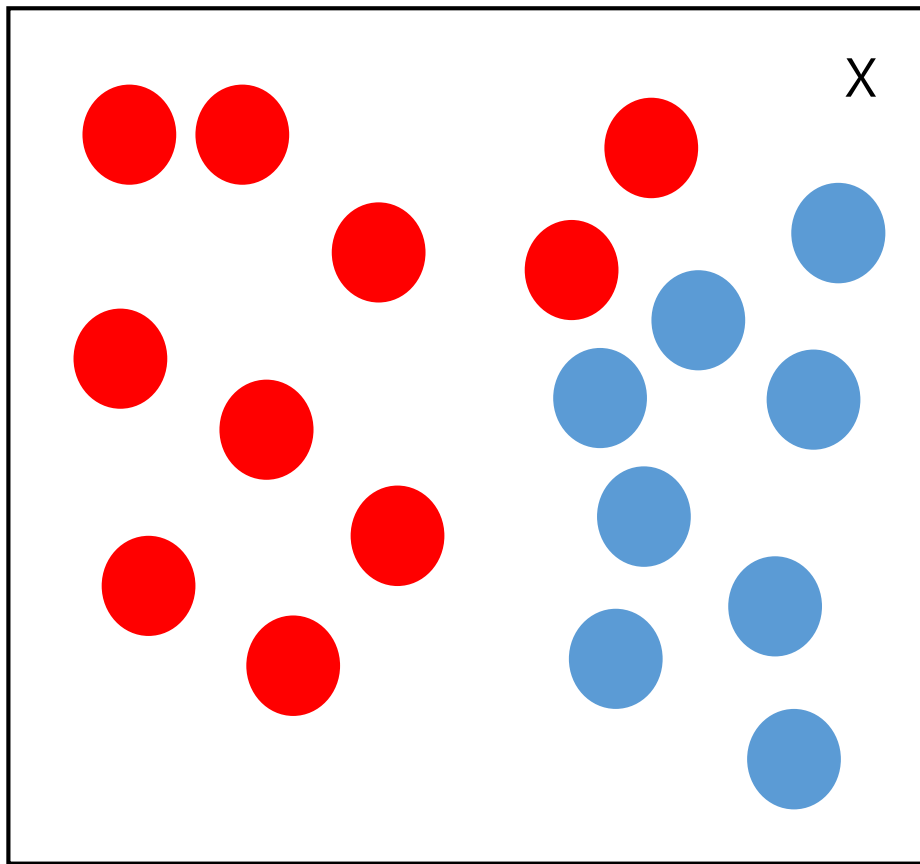
이때, $p(i|t)$: t노드에 속하는 관측치 가운데 i 번째 범주에 속하는 관측치의 비율

분할후

$$Gini_{split} = \sum_{i=1}^k \frac{n_i}{n} Gini(i)$$

이때, 노드 t는 k개의 부분집합으로 나뉘지며, n_i 는 i번째 부분집합의 관측치 개수

지니지수 (Gini index)



$$\text{Gini}(t) = 1 - \sum_i [p(i|t)]^2$$

이때, $p(i|t)$: t 노드에 속하는 관측치 가운데 i 번째 범주에 속하는 관측치의 비율

$$= 1 - \left(\frac{8}{18}\right)^2 - \left(\frac{10}{18}\right)^2$$

$$= 0.494$$

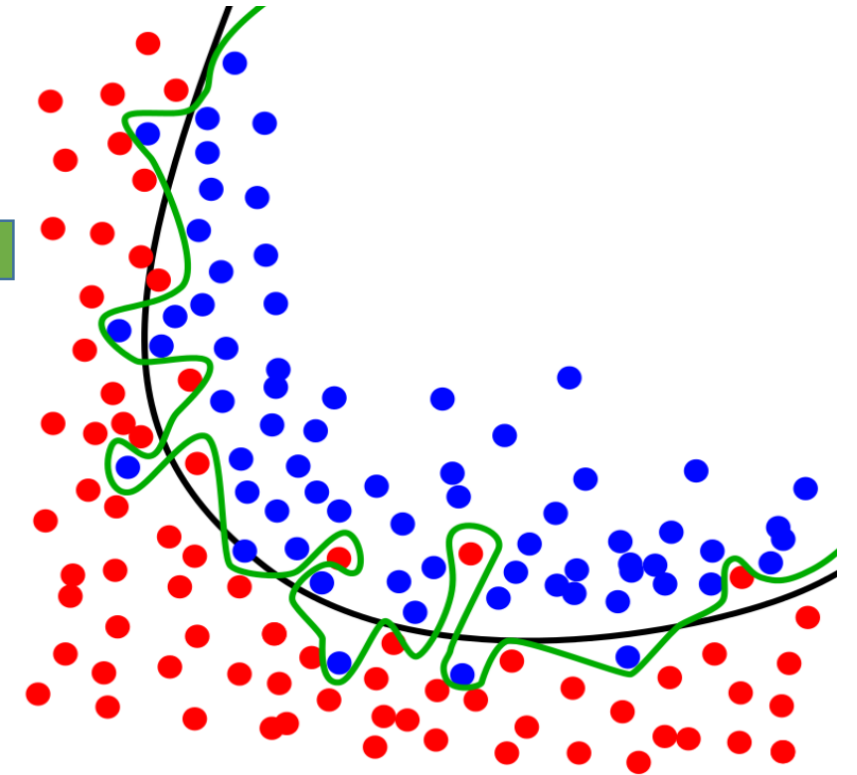
Pruning(가지치기)

<Machine Learning>은 사실 이미 output을 알고 있는 데이터에 맞추기 위해서가 아닌 새로운 데이터를 얼마나 잘 예측하는 모델을 세울 수 있을지가 중요하다!!

→ 결국, 단순히 train set의 정확도를 높일 것이 아니라 일반화 성능을 높여야 함!

각종 Splitting Criteria들로 나무를 끝까지 만들 경우 **overfitting!!** (그림의 **초록색** 선)

- 이렇게, 초록색 선처럼 모든 데이터 포인트마다 구분할 경우, outlier들에 영향도 상당히 많이 받고, 일반화 성능이 떨어지는 불안정한 모델이 설계가 될 것!!
- 따라서 사전에 검은색 선처럼 일반화 된 decision boundary를 설정해 주는 것이 좋다. 이를 위한 작업이 바로 "Pruning"이다



Pruning(가지치기)

자! 간단하게 **Pruning**을 다시 한 번 정리해보자면!!

What is pruning?

- Remove some leaf nodes that could harm future predictions.
- 마지막 leaf nodes들을 좀 제거해서 일반화 성능을 높이려는 것!

Why pruning?

- Escape from overfitting problem
- 모델이 오버피팅되는 문제를 해결하기 위해 진행하는 단계

Pruning(가지치기)

<pre-pruning> 또는 <post – pruning>의 두가지 방법

1. Pre – pruning

말 그대로, 사전에 가지를 치는 것

- max_depth
- min_sample_leaf
- max_leaf_nodes

2. Post – pruning

최대 크기로 키운 후, 가지를 치는 것

- Reduced Error pruning
- Pessimistic error pruning
- Error complexity pruning
- Minimum Error pruning
- Minimal Cost-Complexity Pruning

EX. 과적합 방지를 위한 Pre-Pruning Hyperparameter (rpart)

- maxdepth : Root node에서 Leaf node까지 최장 길이
- minsplit : splitting node에 남아있어야 하는 최소한의 records 개수
- minbucket : terminal node에 남아있어야 하는 최소한의 records 개수

그러나 rpart는 pre pruning보다는 post pruning을 하는 것이 일반적!

maxdepth: This parameter is used to set the maximum depth of a tree. Depth is the length of the longest path from a Root node to a Leaf node. Setting this parameter will stop growing the tree when the depth is equal the value set for **maxdepth**.

minsplit: It is the minimum number of records that must exist in a node for a split to happen or be attempted. For example, we set minimum records in a split to be 5; then, a node can be further split for achieving purity when the number of records in each split node is more than 5.

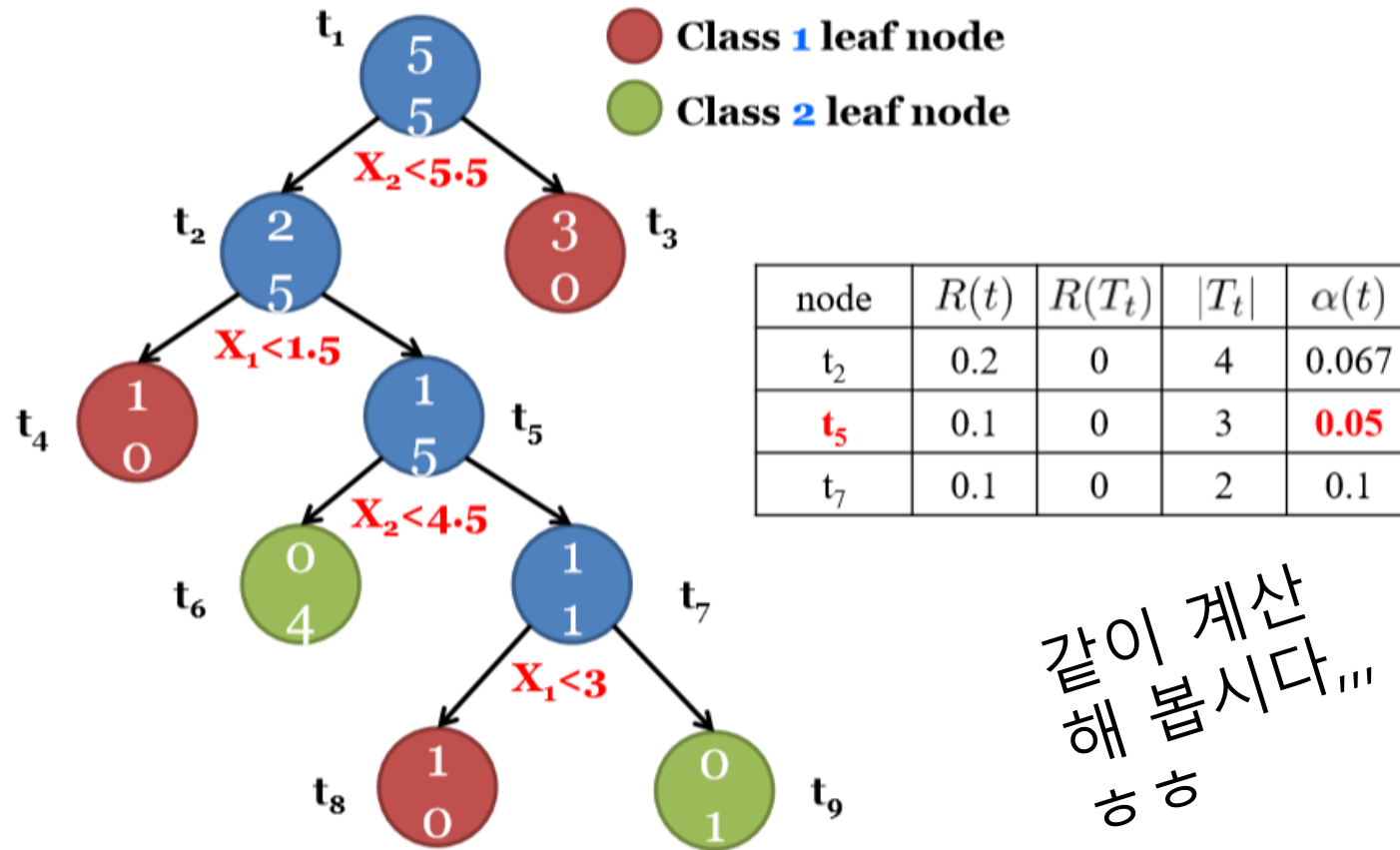
minbucket: It is the minimum number of records that can be present in a Terminal node. For example, we set the minimum records in a node to 5, meaning that every Terminal/Leaf node should have at least five records. We should also take care of not overfitting the model by specifying this parameter. If it is set to a too-small value, like 1, we may run the risk of overfitting our model.

<Minimal Cost-Complexity Pruning>의 알고리즘

- T : a final tree with leaf nodes
- T_t : a subtree which root node is the node t
- $|T|$: complexity of tree T = number of leaf nodes of T
- $|T_t|$: complexity of subtree T_t = number of leaf nodes of T_t
- $R(t)$: misclassification (error) rate of the tree T , when the node t is claimed as a leaf node.
- $R(T_t)$: misclassification (error) rate of the subtree T_t itself
- Prune the tree at the node t when $\alpha(t)$ is minimal.

$$\alpha(t) = \frac{R(t) - R(T_t)}{|T_t| - 1}$$

<Minimal Cost-Complexity Pruning>의 알고리즘



문제 1번.

Question) Consider the CART algorithm to build a decision tree for the data shown below.

X_1	X_2	X_3	Y
0	0	1	B
1	0	1	A
0	1	0	B
1	0	0	B
1	0	1	A
0	0	1	A
1	1	0	B
0	0	0	B
0	1	0	A
1	1	1	A

네. 맞아요. 어려워요.
풀어보세요 ㅎㅎ
같이 풀거예요 ㅎㅎ

- (1) Which variable would be chosen as the first splitting variable?
- (2) What is the misclassification rate of the best decision tree having two leaf nodes?

3. 과적합 방지를 위한 Post Pruning by CP value

```
> # Pruning (과적합 방지)
> printcp(credit_rpart)
```

Classification tree:

```
rpart(formula = default ~ ., data = credit_train)
```

Variables actually used in tree construction:

```
[1] age          amount      checking_balance  credit_history  job
[6] months_loan_duration percent_of_income  phone          savings_balance
```

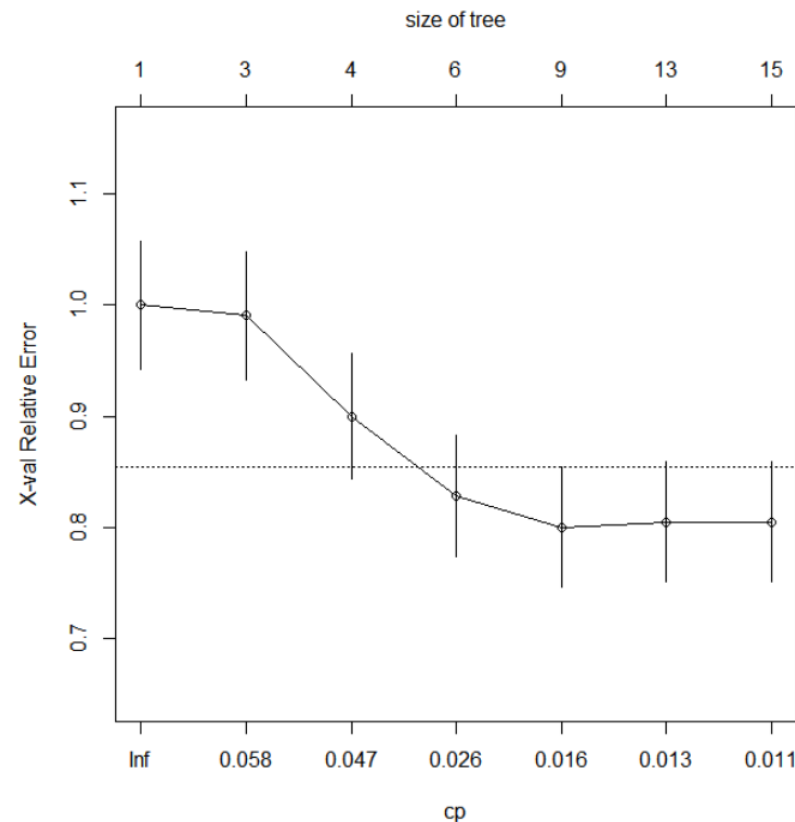
Root node error: 210/700 = 0.3

n= 700

	CP	nsplit	rel error	xerror	xstd
1	0.059524	0	1.00000	1.00000	0.057735
2	0.057143	2	0.88095	0.99048	0.057577
3	0.038095	3	0.82381	0.90000	0.055934
4	0.017460	5	0.74762	0.82857	0.054450
5	0.014286	8	0.69524	0.80000	0.053807
6	0.011905	12	0.62857	0.80476	0.053917
7	0.010000	14	0.60476	0.80476	0.053917

- CP(Complexity Parameter) : 값이 작아질수록 complexity 커짐
- nsplit = 가지의 분기수.(nsplit+1만큼의 leaf node생성)
- rel error=오류율 ($1-R^2$)
- xerror=교차검증 오류율 (rpart는 built-in cross validation있음)
- xstd = 교차검증오류의 표준오차.
- Horizontal line : the minimum average cross-validation error + SD of error

```
> plotcp(credit_rpart)
```



3. 과적합 방지를 위한 Post - Pruning

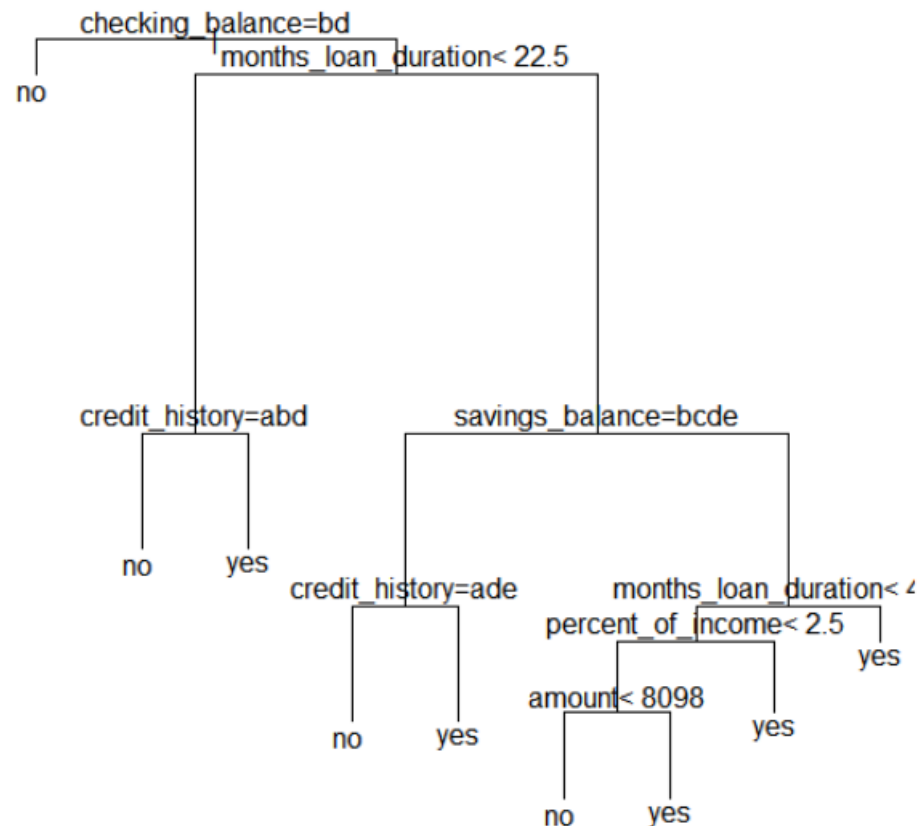
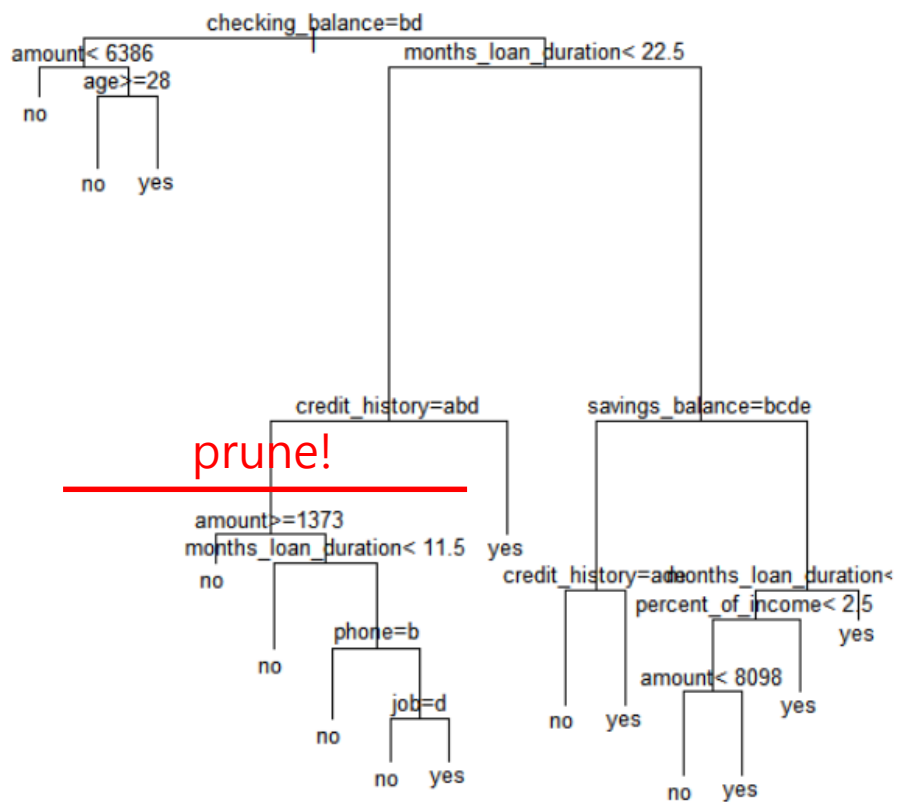
Pruning (과적합 방지)

printcp(credit_rpart)

plotcp(credit_rpart)

credit_prune <- prune(credit_rpart,cp=credit_rpart\$cpstable[which.min(credit_rpart\$cpstable[, "xerror"]), "CP"])

plot(credit_prune) ; text(credit_prune)



1. 'Classification tree" with C5.0 algorithm

- Decision Tree 알고리즘 중 가장 잘 알려진것 중 하나
- C4.5 알고리즘의 개선 (속도개선, boosting지원)
- **Classification만 가능**. 엔트로피사용. 다지분리(Multiple split)

강점

- 대부분의 문제에 적합
- NA데이터를 포함한 연속형, 이산형 데이터를 다룰 수 있음
- 중요하지 않은 특징 제거 (pruning)
- 훈련 데이터를 구성하는 변수들에 다른 가중치를 적용시킬 수 있음

약점

- Outlier에 취약함.

- **C5.0 알고리즘은 '합리적 기본값'을 이용하여 자동적으로 Post-pruning.**
- Overfitting 후 Classification error(분류오차)가 작은, 즉 분류에 큰 영향력이 없는 branch제거.
- Subtree Replacement , Subtree Raising
- <http://www.bowdoin.edu/~echown/courses/370/tutorial.html> : Informal Tutorial

< Data설명 >

- 목적 : C5.0 의사결정 트리를 이용한 간단한 대출승인모델 개발
- 데이터설명 : 독일의 한 신용기관에서 얻은 1000개의 대출예시와 대출 신청자의 특성을 나타내는 수치 특징과 명목 특징을 포함하고 있음.
- 16개의 Predictor variable (수치형, 명목형)
1개의 Response variable(default = 채무불이행여부. Yes/No의 two class)

< C5.0 모델링 순서 >

1. Data split into train and test
(Prop.table로 y variable 비율확인)
2. Building model
3. Test data에서 Confusion matrix로 성능평가
(C5.0은 Classification)
- 4-1. 정확도 향상 by Boosting
- 4-2. 정확도 향상 by Penalty matrix
5. Test data에서 Confusion matrix로 각 모델 성능평가

1. Data split into train and test set

```
### CREDIT (classification) - C5.0 (Set.seed(3))-----
```

```
library(C50)
```

```
credit <- read.csv("credit.csv")
```

```
str(credit)
```

```
summary(credit)
```

```
# Dataset split
```

```
set.seed(3)
```

```
train_sample <- sample(1000,900)
```

```
credit_train <- credit[train_sample,]
```

```
credit_test <- credit[-train_sample,]
```

```
# Train data와 Test data내의 default 비율 비슷한지 확인.
```

```
prop.table(table(credit_train$default))
```

```
prop.table(table(credit_test$default))
```

```
> str(credit)
```

```
'data.frame': 1000 obs. of 17 variables:
```

```
$ checking_balance : Factor w/ 4 levels "< 0 DM", "> 200 DM",...: 1
$ months_loan_duration: int 6 48 12 42 24 36 24 36 12 30 ...
$ credit_history : Factor w/ 5 levels "critical", "good",...: 1 2
$ purpose : Factor w/ 6 levels "business", "car",...: 5 5
$ amount : int 1169 5951 2096 7882 4870 9055 2835 6948
$ savings_balance : Factor w/ 5 levels "< 100 DM", "> 1000 DM",...
$ employment_duration : Factor w/ 5 levels "< 1 year", "> 7 years",...
$ percent_of_income : int 4 2 2 2 3 2 3 2 2 4 ...
$ years_at_residence : int 4 2 3 4 4 4 4 2 4 2 ...
$ age : int 67 22 49 45 53 35 53 35 61 28 ...
$ other_credit : Factor w/ 3 levels "bank", "none",...: 2 2 2 2
$ housing : Factor w/ 3 levels "other", "own",...: 2 2 2 1
$ existing_loans_count: int 2 1 1 1 2 1 1 1 1 2 ...
$ job : Factor w/ 4 levels "management", "skilled",...
$ dependents : int 1 1 2 2 2 2 1 1 1 1 ...
$ phone : Factor w/ 2 levels "no", "yes": 2 1 1 1 1 2 1
$ default : Factor w/ 2 levels "no", "yes": 1 2 1 1 2 1 1
```

2. Building Model

문법 : C5.0(X variables, Y variable(Categorical),
trials=, cost=error cost matrix)

```
## C5.0
# Building model
credit_model <- C5.0(credit_train[-17], credit_train$default)
credit_model #Tree size=69
summary(credit_model) #Error 10.4%
```

Summary 해석

- 수표 계좌잔고(Checking balance) 를 모르거나 200DM이상이고, 다른 신용카드가 없다면 채무불이행 가능성이 없음으로 분류
- 총 346개가 이 조건만족으로 NO라고 분리되었으나 그중 36개의 CASE는 잘못 분류된것.
- 수표계좌잔고가 0보다 작거나, 1~200DM이고, 대출이력이 완벽하거나 아주 좋고, 저금액이 500DM이하이면 채무불이행가능성 있음으로 분류
- 상식적으로 모순된 규칙이 가끔 발견됨 -> 실제 패턴을 반영한 것일 수도, 통계적 이상치 일 수도 있음. 이에 대한 관찰 필요!

```
> summary(credit_model) #Error 10.4%
```

```
Call:
C5.0.default(x = credit_train[-17], y = credit_train$default)
```

```
C5.0 [Release 2.07 GPL Edition]
```

```
Tue Mar 10 14:52:37 2020
```

```
Class specified by attribute `outcome`
```

```
Read 900 cases (17 attributes) from undefined.data
```

```
Decision tree:
```

```
checking_balance in {> 200 DM, unknown}:
...other_credit = none: no (346/36)
:   other_credit in {bank, store}:
:   ...purpose in {car0, furniture/appliances, renovations}: no (31/4)
:   purpose in {business, car, education}:
:   ...percent_of_income <= 1: no (7/1)
:   percent_of_income > 1:
:   ...existing_loans_count > 2: no (2)
:   existing_loans_count <= 2:
:   ...savings_balance in {< 100 DM, 500 - 1000 DM}: yes (17/4)
:   savings_balance in {> 1000 DM, 100 - 500 DM}: no (4/1)
:   savings_balance = unknown:
:   ...percent_of_income <= 3: no (4)
:   percent_of_income > 3: yes (4/1)
checking_balance in {< 0 DM, 1 - 200 DM}:
...credit_history in {perfect, very good}:
...savings_balance in {< 100 DM, 100 - 500 DM}: yes (54/12)
:   savings_balance = > 1000 DM: no (1)
:   savings_balance = 500 - 1000 DM:
:   ...housing = other: yes (1)
:   housing in {own, rent}: no (3)
:   savings_balance = unknown:
:   ...amount <= 4844: yes (2)
:   amount > 4844: no (3)
credit_history in {critical, good, poor}:
```

3. R코드 해석 및 실습

Evaluation on training data (900 cases):

Decision Tree	
Size	Errors
86	94(10.4%)

<<

- Tree size
= Number of leaves
- (Train) Error rate
= 10.4%

(a)	(b)	<-classified as
599	30	(a): class no
64	207	(b): class yes

Attribute usage:

100.00% checking_balance
60.11% other_credit
53.89% credit_history
46.78% months_loan_duration
44.78% savings_balance
24.00% purpose
21.44% years_at_residence
14.33% housing
14.11% employment_duration
12.67% job
9.11% age
6.67% percent_of_income
4.00% existing_loans_count
3.56% amount
1.67% phone

Variance
Importance

3. 기본모델 성능평가 : Test data에서 Confusion Matrix

```
# Confusion matrix  
credit_pred <- predict(credit_model, credit_test)  
caret::confusionMatrix(credit_pred, credit_test$default)
```

```
> caret::confusionMatrix(credit_pred, credit_test$default)  
Confusion Matrix and Statistics
```

	Reference	
Prediction	no	yes
no	59	19
yes	12	10

Accuracy : 0.69

95% CI : (0.5897, 0.7787)

No Information Rate : 0.71

P-Value [Acc > NIR] : 0.7127

Kappa : 0.1893

McNemar's Test P-Value : 0.2812

Sensitivity : 0.8310

Specificity : 0.3448

Pos Pred Value : 0.7564

Neg Pred Value : 0.4545

Prevalence : 0.7100

Detection Rate : 0.5900

Detection Prevalence : 0.7800

Balanced Accuracy : 0.5879

'Positive' Class : no

성능 그닥...
개선 필요해보임

4-1. 정확도 향상을 위한 Adaptive Boosting 모델세우기

앞서 보았듯이 기본 모델의 성능이 그다지 좋지 않으므로 모델을 향상시켜보려한다.

- C4.5에서 C5.0으로 개선된 사항 중 하나인 Adaptive boosting 옵션.
- Adaptive boosting : 여러 개의 의사결정트리(trial)를 만들어 최고 클래스에 투표하게 만드는 과정으로, 맞추기 어려운 문제를 더 잘 맞추게 하는 과정

```
## C5.0 의사결정 트리 정확도 향상
## Adaptive boosting (AdaBoosting)
# Building model
credit_boost10 <- C5.0(credit_train[-17], credit_train$default, trials=10)
credit_boost10 #Tree size = 56.6 (10회의 반복을 통해 트리의 크기가 줄어듦)
summary(credit_boost10) #Error 1.4%
```

```
> credit_boost10 #Tree size = 56.6 (10회의 반복을 통해 트리의 크기가 줄어듦)
```

```
Call:
C5.0.default(x = credit_train[-17], y = credit_train$default, trials = 10)
```

```
Classification Tree
Number of samples: 900
Number of predictors: 16
```

```
Number of boosting iterations: 10
Average tree size: 64
```

```
Non-standard options: attempt to group attributes
```

Evaluation on training data (900 cases):

Trial	Decision Tree	
	Size	Errors
0	86	94 (10.4%)
1	53	176 (19.6%)
2	52	173 (19.2%)
3	67	157 (17.4%)
4	53	177 (19.7%)
5	69	166 (18.4%)
6	64	175 (19.4%)
7	61	173 (19.2%)
8	61	185 (20.6%)
9	74	162 (18.0%)
boost		11 (1.2%)

(Train)Error rate
10.4%→1.4%

(a)	(b)	<-classified as
627	2	(a): class no
9	262	(b): class yes

4-2. 정확도 향상을 위한 Penalty 부여 모델 세우기

모델의 정확도를 향상시킬 수 있는 또다른 방법은 Penalty matrix로 manually 조작하는 것.

실제로 채무불이행인데 아니라고 예측한 경우에 대하여 패널티를 더 많이 주자
Why? 은행은 대출을 해주고 반드시 돈을 받아야하는 입장이므로!

```
> ## Penalty 부여
> # Cost matrix
> matrix_dimensions <- list(c("no","yes"),c("no","yes")) #먼저 차원구성
> names(matrix_dimensions) <- c("predicted","actual")
> matrix_dimensions
$predicted
[1] "no" "yes"
```

```
$actual
[1] "no" "yes"
```

```
> error_cost <- matrix(c(0,1,4,0),nrow=2,
+                       dimnames = matrix_dimensions)
> error_cost
```

```
      actual
predicted no yes
no       0  4
yes      1  0
```

실제로 yes(채무불이행) 인데 예측하기를
no(채무불이행 안함) 으로 한 경우 4의 패널티를,
반대의 경우에는 1의 패널티를 주자

```
# Building model
credit_cost <- C5.0(credit_train[-17],credit_train$default,
                    cost=error_cost)
summary(credit_cost)
```

Evaluation on training data (900 cases):

Decision Tree		
Size	Errors	Cost
43	194 (21.6%)	0.28

(a)	(b)	<-classified as
453	176	(a): class no
18	253	(b): class yes

(Train)Error rate
10.4%→1.4%→21.6%
기본 부스팅 패널티

Attribute usage:

```
100.00% checking_balance
65.44% credit_history
53.89% months_loan_duration
48.44% other_credit
41.11% savings_balance
36.44% purpose
24.33% existing_loans_count
18.56% age
17.00% job
10.56% employment_duration
8.44% amount
4.33% housing
4.22% percent_of_income
4.22% years_at_residence
```


5. 각 모델 성능평가 : Test data에서 Confusion Matrix

```
# Confusion matrix
credit_boost10_pred <- predict(credit_boost10, credit_test)
caret::confusionMatrix(credit_boost10_pred, credit_test$default)
```

```
> caret::confusionMatrix(credit_boost10_pred, credit_test$default)
Confusion Matrix and Statistics
```

```
      Reference
Prediction no yes
no      62  16
yes     9   13
```

Accuracy : 0.75

95% CI : (0.6534, 0.8312)

No Information Rate : 0.71

P-Value [Acc > NIR] : 0.2222

Kappa : 0.3462

Mcnemar's Test P-Value : 0.2301

Sensitivity : 0.8732

Specificity : 0.4483

Pos Pred Value : 0.7949

Neg Pred Value : 0.5909

Prevalence : 0.7100

Detection Rate : 0.6200

Detection Prevalence : 0.7800

Balanced Accuracy : 0.6608

'Positive' Class : no

```
# Confusion matrix
credit_cost_pred <- predict(credit_cost, credit_test)
caret::confusionMatrix(credit_cost_pred, credit_test$default)
```

```
> caret::confusionMatrix(credit_cost_pred, credit_test$default)
Confusion Matrix and Statistics
```

```
      Reference
Prediction no yes
no      41   7
yes     30  22
```

Accuracy : 0.63

95% CI : (0.5276, 0.7244)

No Information Rate : 0.71

P-Value [Acc > NIR] : 0.9671870

Kappa : 0.2722

Mcnemar's Test P-Value : 0.0002983

Sensitivity : 0.5775

Specificity : 0.7586

Pos Pred Value : 0.8542

Neg Pred Value : 0.4231

Prevalence : 0.7100

Detection Rate : 0.4100

Detection Prevalence : 0.4800

Balanced Accuracy : 0.6680

'Positive' Class : no

5. 부스팅 모델 성능평가 : Test data에서 Confusion Matrix (최종)

1) 기본

```
> caret::confusionMatrix(credit_pred, credit_test$default)
Confusion Matrix and Statistics
```

```

      Reference
Prediction no yes
no      59   19
yes     12   10

Accuracy : 0.69

```

2) Adaptive Boosting

```
> caret::confusionMatrix(credit_boost10_pred, credit_test$default)
Confusion Matrix and Statistics
```

```

      Reference
Prediction no yes
no      62   16
yes      9   13

Accuracy : 0.75

```

3) Penalty Matrix

```
> caret::confusionMatrix(credit_cost_pred, credit_test$default)
Confusion Matrix and Statistics
```

```

      Reference
Prediction no yes
no      41    7
yes     30   22

Accuracy : 0.63

```

• Test Accuracy

0.69 / 0.75 / 0.63
기본 부스팅 패널티

- 가장 오분류율이 적은 모델 : AdaBoosting 모델
- 그러나 패널티 모델이 은행의 입장에선 가장 좋다. 채무불이행 yes를 no로 분리할 경우 7%로 줄었기 때문!

❖ 부스팅은 보통 좋은 성능을 가지고 오지만,

- 1) 의사결정 트리를 여러 번 만드는데 드는 시간이 ↑
 - 2) 훈련데이터에 잡음이 많으면 부스팅을 통해 개선이 힘들 수 있음.
- 따라서 context에 따라 적절한 방법으로 모델성능개선이 필요함

1. "Classification tree" with CART (Classification and Regression tree)

- **Response variable** : 명목형, 수치형 변수 둘다 가능 <- > C5.0 : 명목형 변수만 가능
- Classification : 지니계수 / Regression : RSS
- Binary split
- **Package명: rpart**
- 장점 : 연산속도가 빠르다
- 단점 : 과적합화의 위험. 따라서 **가지치기 반드시 해야함**

< rpart package 모델링 순서 >

1. Data split into train and test
2. Building model
3. **Pruning**
4. Test data에서 Confusion matrix로 성능평가

1. Data split into train and test set

```
# Dataset split with caret package (Response variable 비율 유지)
library(caret)
set.seed(234)
train_index <- createDataPartition(credit$default, p=0.7)$Resample1
credit_train <- credit[train_index,]
credit_test <- credit[-train_index,]
```

```
> prop.table(table(credit_train$default))
```

```
no yes
0.7 0.3
```

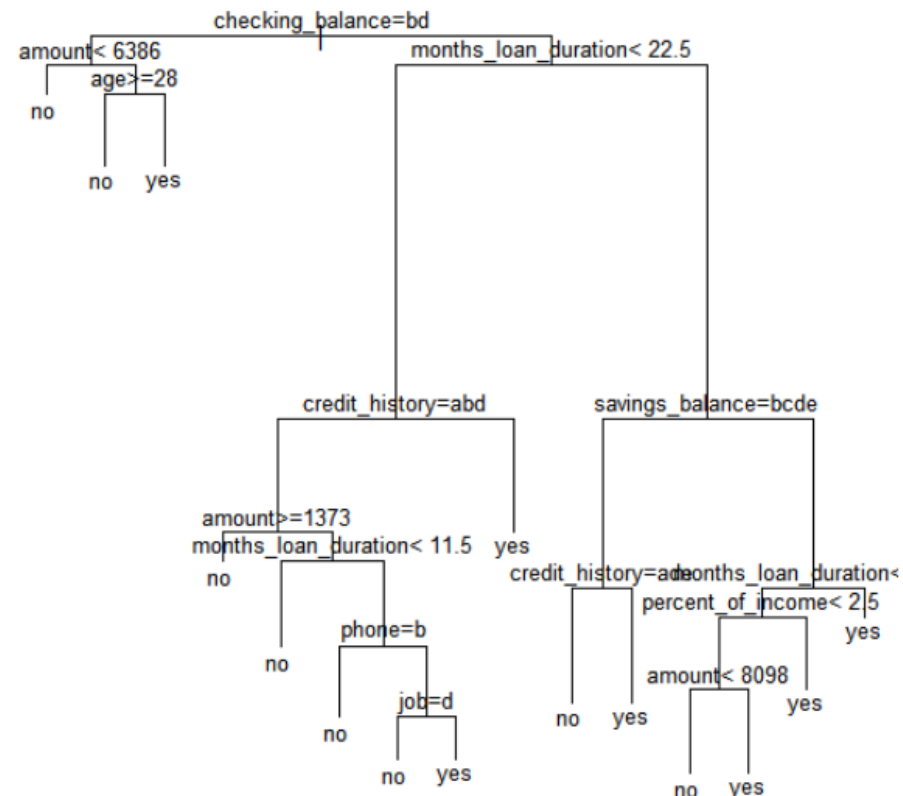
```
> prop.table(table(credit_test$default))
```

```
no yes
0.7 0.3
```

2. Building Model

문법 : `rpart(Response variable~., data)`

```
# Building model
library(rpart)
credit_rpart <- rpart(default~., data= credit_train )
plot(credit_rpart) ; text(credit_rpart,cex=0.8)
credit_rpart
```



2. Building Model

```
> credit_rpart
n= 700
```

```
node), split, n, loss, yval, (yprob)
* denotes terminal node
```

Regression tree였으면
Node) split, n, deviance, yval

```
1) root 700 210 no (0.70000000 0.30000000)
2) checking_balance=> 200 DM,unknown 311 40 no (0.87138264 0.12861736)
4) amount< 6386 282 29 no (0.89716312 0.10283688) *
5) amount>=6386 29 11 no (0.62068966 0.37931034)
10) age>=28 22 5 no (0.77272727 0.22727273) *
11) age< 28 7 1 yes (0.14285714 0.85714286) *
3) checking_balance=< 0 DM,1 - 200 DM 389 170 no (0.56298201 0.43701799)
6) months_loan_duration< 22.5 220 73 no (0.66818182 0.33181818)
12) credit_history=critical,good,poor 198 58 no (0.70707071 0.29292929)
24) amount>=1373 113 22 no (0.80530973 0.19469027) *
25) amount< 1373 85 36 no (0.57647059 0.42352941)
50) months_loan_duration< 11.5 31 6 no (0.80645161 0.19354839) *
51) months_loan_duration>=11.5 54 24 yes (0.44444444 0.55555556)
102) phone=yes 11 3 no (0.72727273 0.27272727) *
103) phone=no 43 16 yes (0.37209302 0.62790698)
206) job=unskilled 15 6 no (0.60000000 0.40000000) *
207) job=skilled,unemployed 28 7 yes (0.25000000 0.75000000) *
13) credit_history=perfect,very good 22 7 yes (0.31818182 0.68181818) *
7) months_loan_duration>=22.5 169 72 yes (0.42603550 0.57396450)
14) savings_balance=> 1000 DM,100 - 500 DM,500 - 1000 DM,unknown 52 20 no (0.61538462 0.38461538)
28) credit_history=critical,poor,very good 28 4 no (0.85714286 0.14285714) *
29) credit_history=good,perfect 24 8 yes (0.33333333 0.66666667) *
15) savings_balance=< 100 DM 117 40 yes (0.34188034 0.65811966)
30) months_loan_duration< 47.5 94 38 yes (0.40425532 0.59574468)
60) percent_of_income< 2.5 39 17 no (0.56410256 0.43589744)
120) amount< 8097.5 27 8 no (0.70370370 0.29629630) *
121) amount>=8097.5 12 3 yes (0.25000000 0.75000000) *
61) percent_of_income>=2.5 55 16 yes (0.29090909 0.70909091) *
31) months_loan_duration>=47.5 23 2 yes (0.08695652 0.91304348) *
```

Credit_model 해석

*가 총 15개 이므로 terminal node는 총 15개
Cf) Splitting node에서 해당변수가 NA이면 그대로 terminal node에 떨어진다.

- 1) Root node : 700건의 total data, 대표그룹=no, no:yes=7:3
- 2) checking_balance=>200DM,unknown
: 총 311건의 data, (여기서 끝나면)오분류 40건, 대표그룹=no
- 4) 그 중 amount<6386
: 총 282건의 data, 오분류 29건
- 10) 그 중 age=>28
: 총 22건, 오분류 5건, 대표그룹=no (Terminal node)

3. 과적합 방지를 위한 Post Pruning by CP value

```
> # Pruning (과적합 방지)
> printcp(credit_rpart)
```

```
Classification tree:
rpart(formula = default ~ ., data = credit_train)
```

```
Variables actually used in tree construction:
```

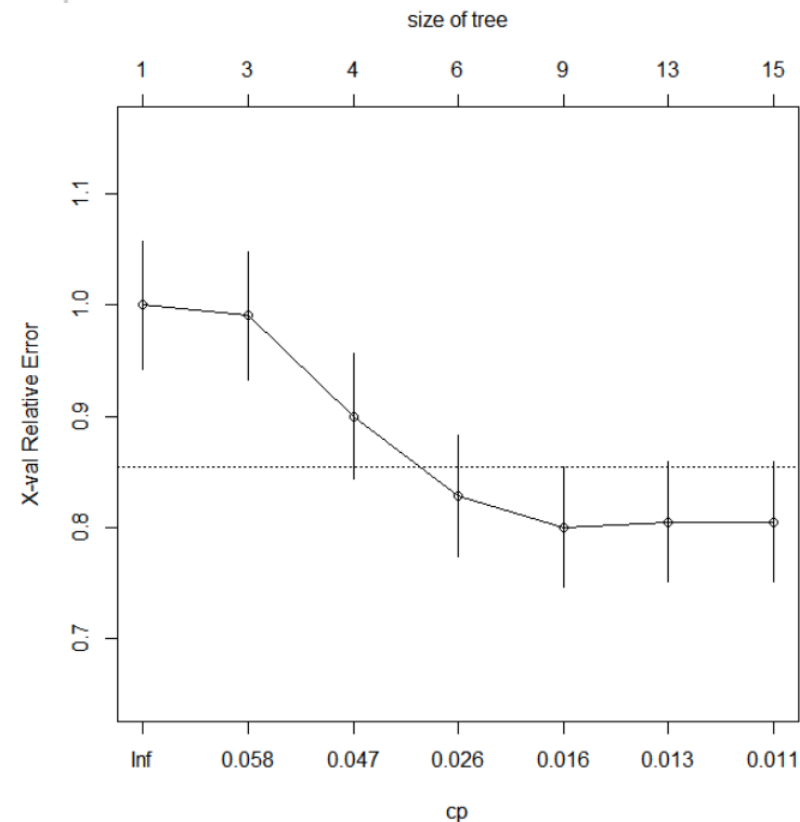
```
[1] age          amount          checking_balance  credit_history  job
[6] months_loan_duration percent_of_income  phone          savings_balance
```

```
Root node error: 210/700 = 0.3
```

```
n= 700
```

	CP	nsplit	rel error	xerror	xstd
1	0.059524	0	1.00000	1.00000	0.057735
2	0.057143	2	0.88095	0.99048	0.057577
3	0.038095	3	0.82381	0.90000	0.055934
4	0.017460	5	0.74762	0.82857	0.054450
5	0.014286	8	0.69524	0.80000	0.053807
6	0.011905	12	0.62857	0.80476	0.053917
7	0.010000	14	0.60476	0.80476	0.053917

```
> plotcp(credit_rpart)
```



3. 과적합 방지를 위한 Post - Pruning

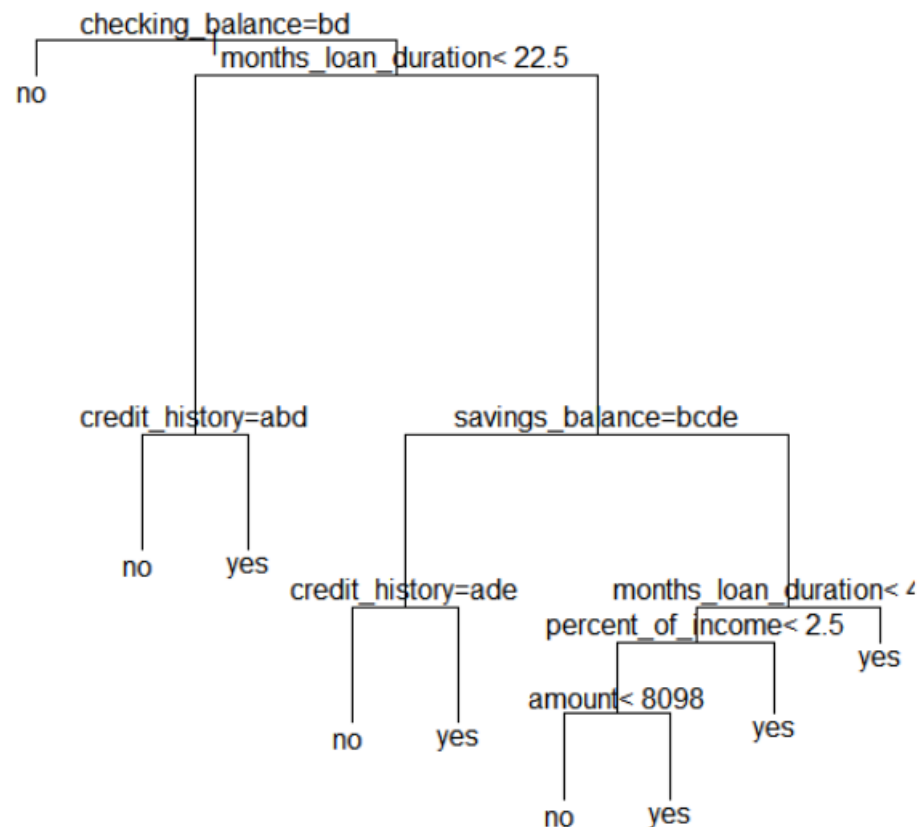
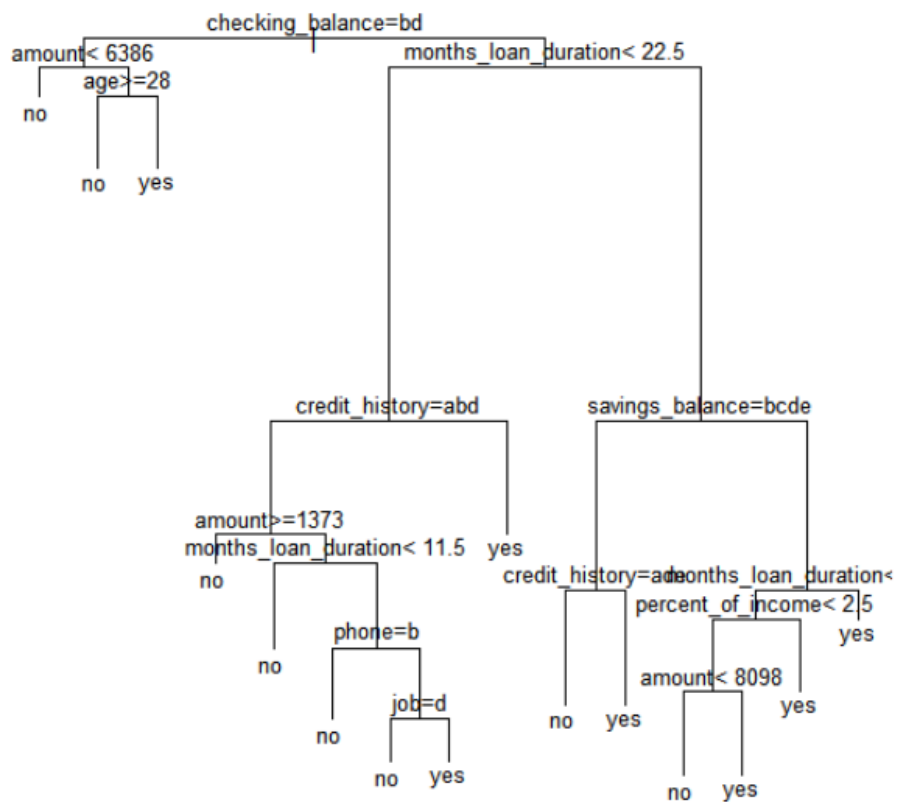
Pruning (과적합 방지)

printcp(credit_rpart)

plotcp(credit_rpart)

credit_prune <- prune(credit_rpart, cp=credit_rpart\$cpstable[which.min(credit_rpart\$cpstable[, "xerror"]), "CP"])

plot(credit_prune) ; text(credit_prune)



4. Test data에서 Confusion matrix로 각 모델 성능평가

```
> # Test data에서 검정 : Confusion matrix
> credit_pred <- predict(credit_rpart, credit_test, type = "class")
> confusionMatrix(credit_pred, credit_test$default)
Confusion Matrix and Statistics
```

	Reference	
Prediction	no	yes
no	181	53
yes	29	37

```

      Accuracy : 0.7267
      95% CI   : (0.6725, 0.7763)
No Information Rate : 0.7
P-Value [Acc > NIR] : 0.17257

      Kappa : 0.2955

McNemar's Test P-Value : 0.01109

      Sensitivity : 0.8619
      Specificity : 0.4111
      Pos Pred Value : 0.7735
      Neg Pred Value : 0.5606
      Prevalence : 0.7000
      Detection Rate : 0.6033
      Detection Prevalence : 0.7800
      Balanced Accuracy : 0.6365

      'Positive' Class : no
```

```
> credit_prune_pred <- predict(credit_prune, credit_test, type = "class")
> confusionMatrix(credit_prune_pred, credit_test$default)
Confusion Matrix and Statistics
```

	Reference	
Prediction	no	yes
no	188	56
yes	22	34

```

      Accuracy : 0.74
      95% CI   : (0.6865, 0.7887)
No Information Rate : 0.7
P-Value [Acc > NIR] : 0.0722789

      Kappa : 0.306

McNemar's Test P-Value : 0.0001866

      Sensitivity : 0.8952
      Specificity : 0.3778
      Pos Pred Value : 0.7705
      Neg Pred Value : 0.6071
      Prevalence : 0.7000
      Detection Rate : 0.6267
      Detection Prevalence : 0.8133
      Balanced Accuracy : 0.6365

      'Positive' Class : no
```


Thank You