

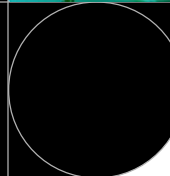
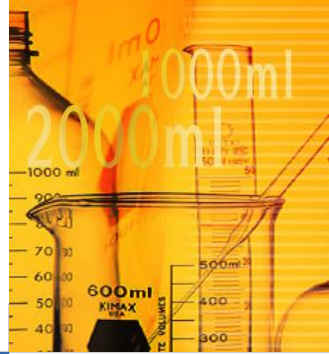
# Machine learning

## Chapter 5

# Decision Tree

Sejong Oh

Bio Information Technology Lab.

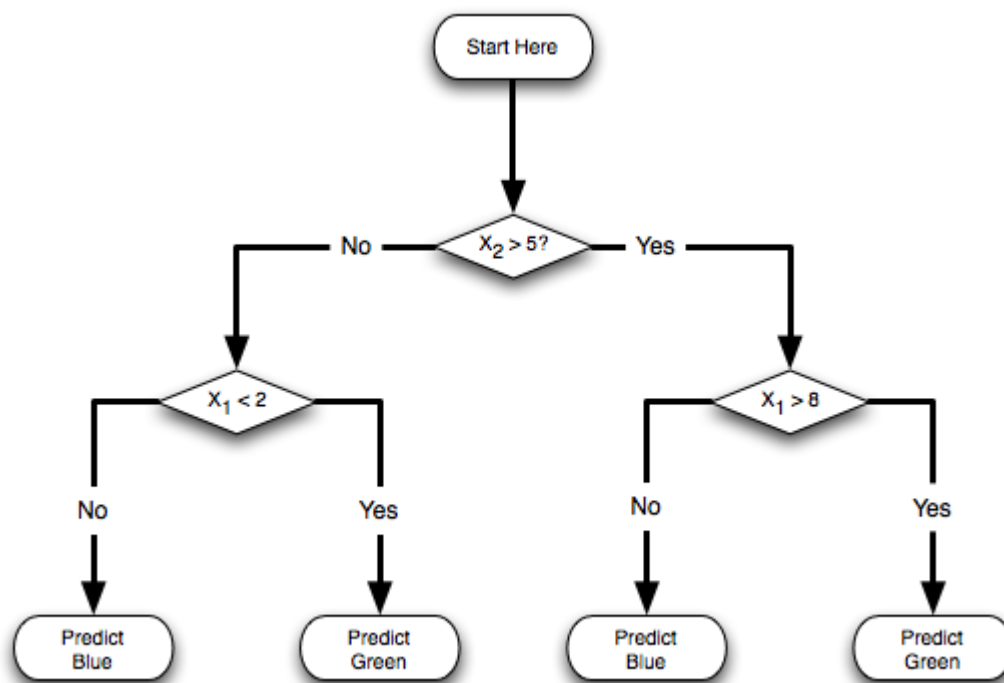


# Contents

- Summary
- Decision tree 생성 과정
- Issues of decision tree
- C5.0 기초
- C5.0 Adaptive boosting
- C5.0 Reduce false negative
- C5.0 Modify decision rule
- Random forest

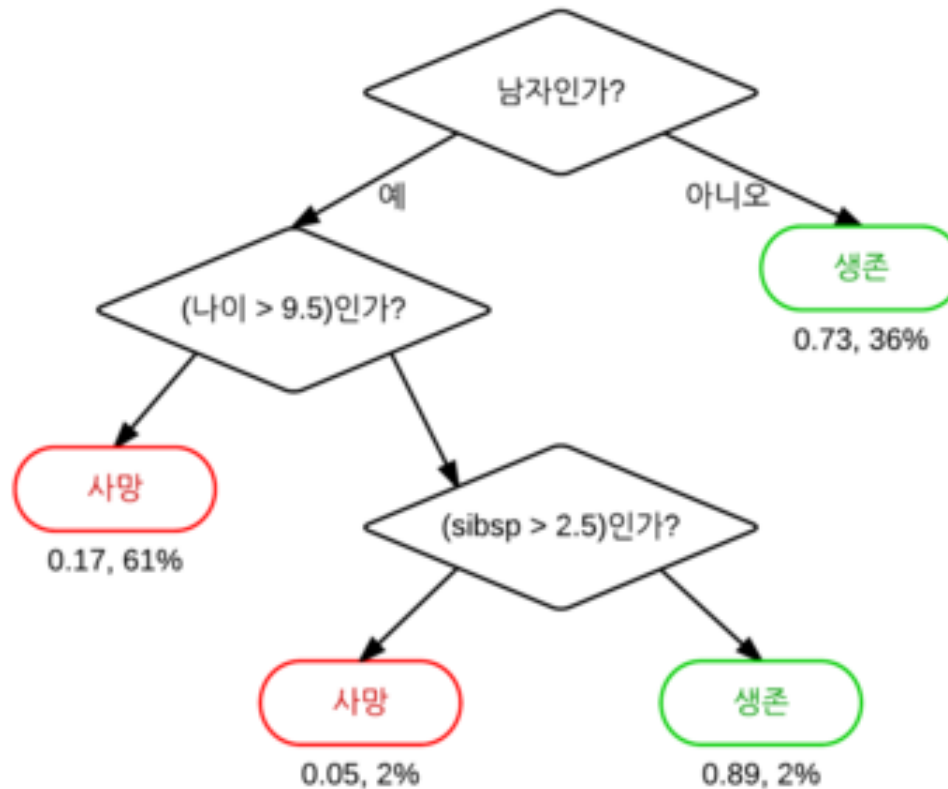
# Summary

- 결정 트리(Decision tree) 이해
  - 큰 문제를 작은 문제들의 조각으로 나누어 해결하는 기법
  - 인간의 의사결정 과정과 유사
  - 의사결정 또는 예측 결과의 의 근거가 명확히 제시되어야 하는 경우 많이 이용되는 학습 모델



# Summary

- 타이타닉호 탑승객의 생존 여부를 나타내는 결정 트리



("sibsp"는 탑승한 배우자와 자녀의 수를 의미한다.) 옆 아래의 숫자는 각각 생존 확률과 탑승객이 그 옆에 해당될 확률을 의미한다.

# Summary

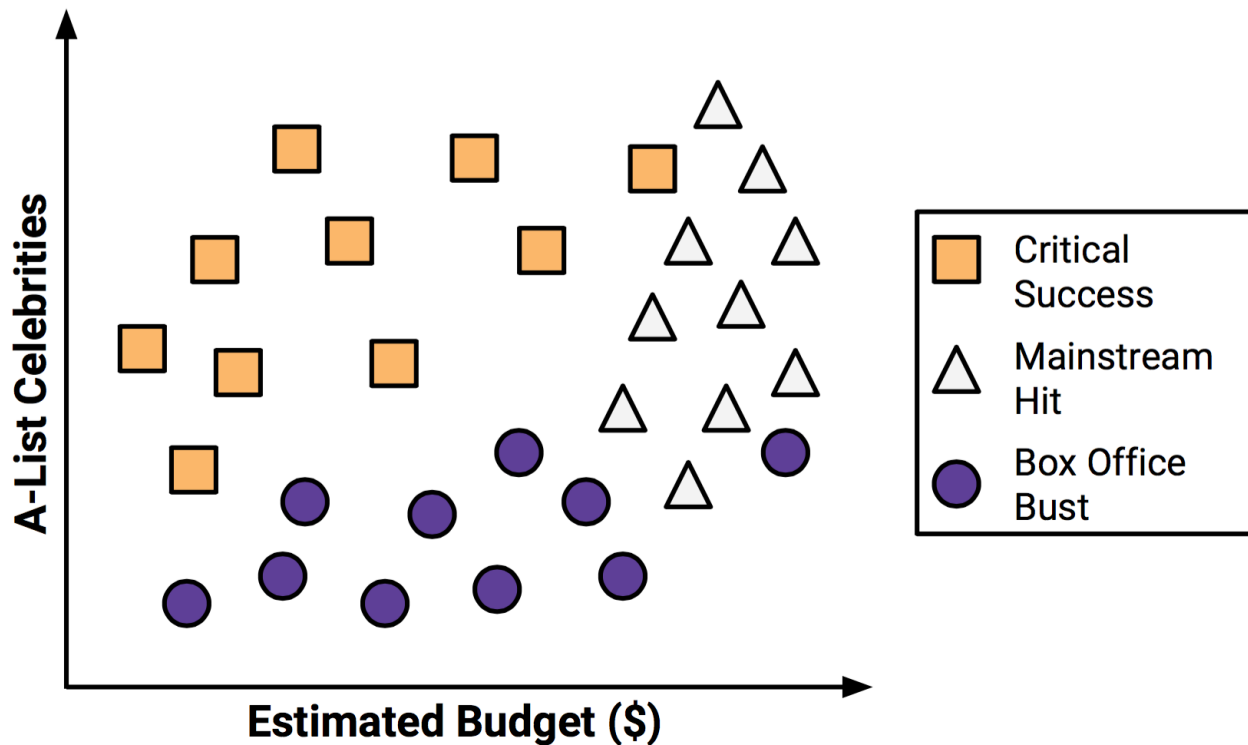
- 알고리즘이 생성한 결정 트리 예제

Decision tree:

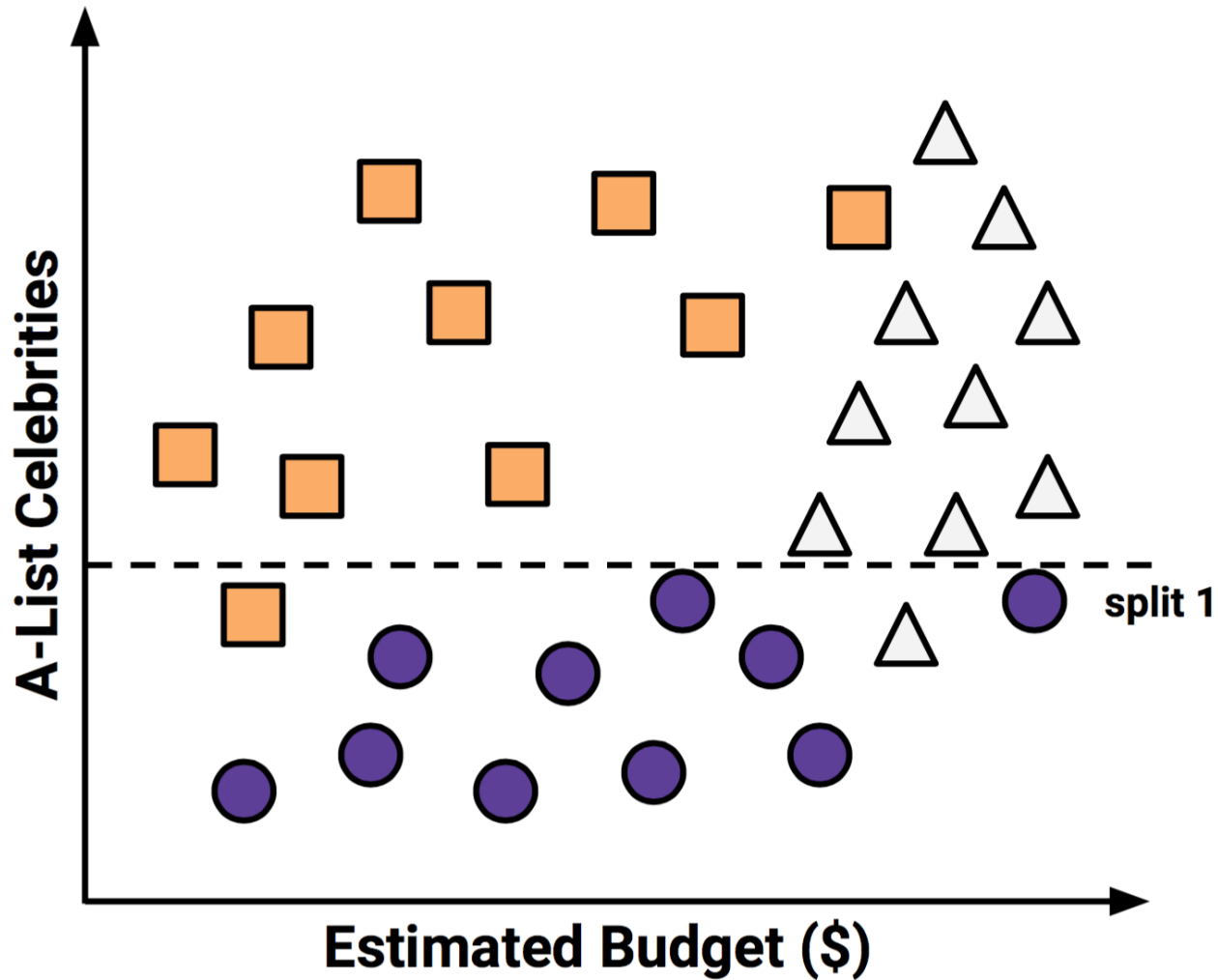
```
total_day_minutes > 264.4:
...voice_mail_plan = yes:
:   ...international_plan = no: no (45/1)
:   :   international_plan = yes: yes (8/3)
:   voice_mail_plan = no:
:   :   ...total_eve_minutes > 187.7:
:   :   :   ...total_night_minutes > 126.9: yes (94/1)
:   :   :   :   total_night_minutes <= 126.9:
:   :   :   :   :   ...total_day_minutes <= 277: no (4)
:   :   :   :   :   :   total_day_minutes > 277: yes (3)
:   :   total_eve_minutes <= 187.7:
:   :   :   ...total_eve_charge <= 12.26: no (15/1)
:   :   :   :   total_eve_charge > 12.26:
:   :   :   :   :   ...total_day_minutes <= 277:
:   :   :   :   :   :   ...total_night_minutes <= 224.8: no (13)
:   :   :   :   :   :   :   total_night_minutes > 224.8: yes (5/1)
:   :   :   total_day_minutes > 277:
:   :   :   :   ...total_night_minutes > 151.9: yes (18)
:   :   :   :   :   total_night_minutes <= 151.9:
:   :   :   :   :   :   ...account_length <= 123: no (4)
:   :   :   :   :   :   :   account_length > 123: yes (2)
total_day_minutes <= 264.4:
...number_customer_service_calls > 3:
:   ...total_day_minutes <= 160.2:
```

# 결정 트리 생성 과정

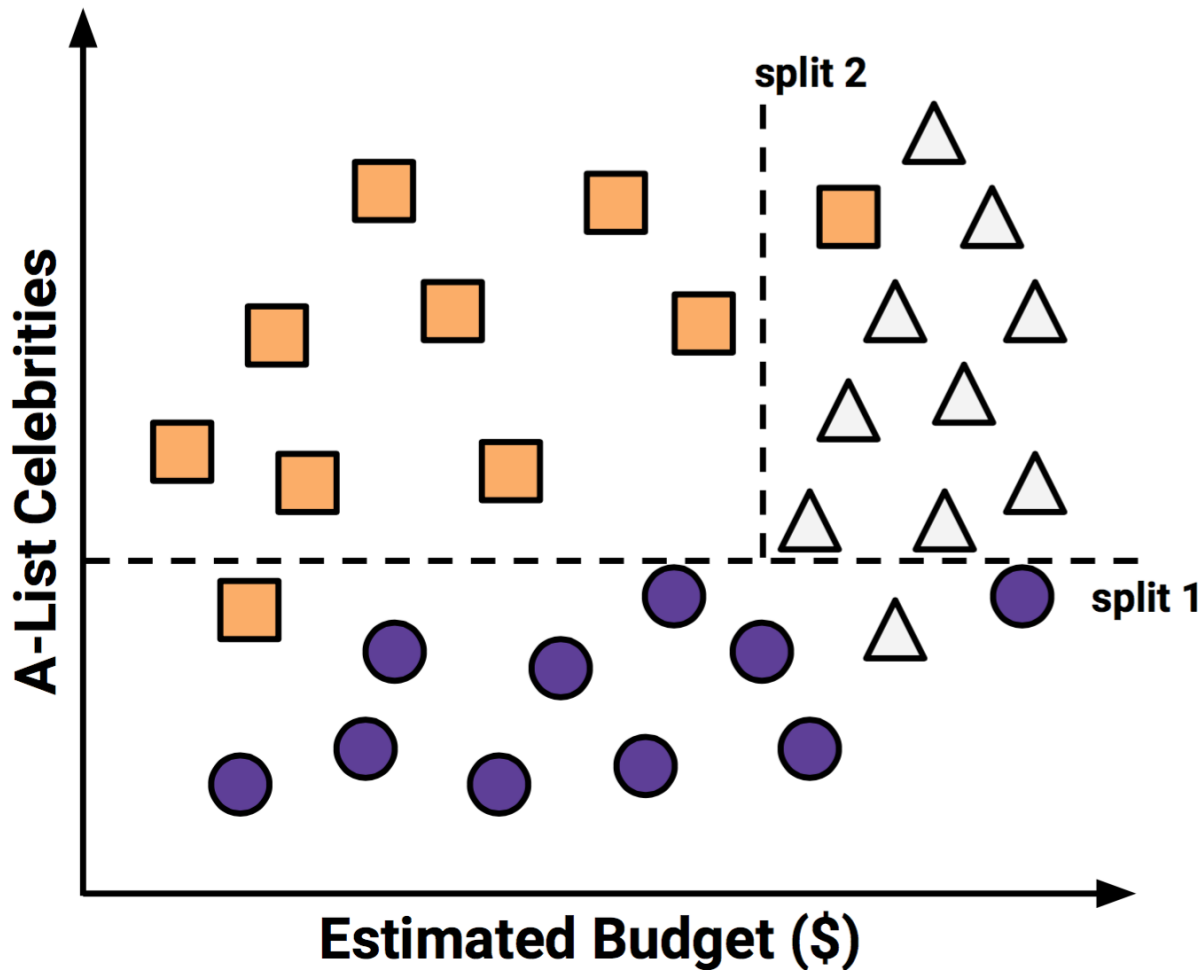
- 영화 대본 및 기본 정보를 이용하여 영화가 흥행할지를 예측하는 모델을 만들고자 한다.
- 사용하는 정보는 유명 배우의 수와 추정 제작비
- 예측 결과는 “매우 흥행”, “어느정도 흥행”, “흥행 실패”



- Step 1

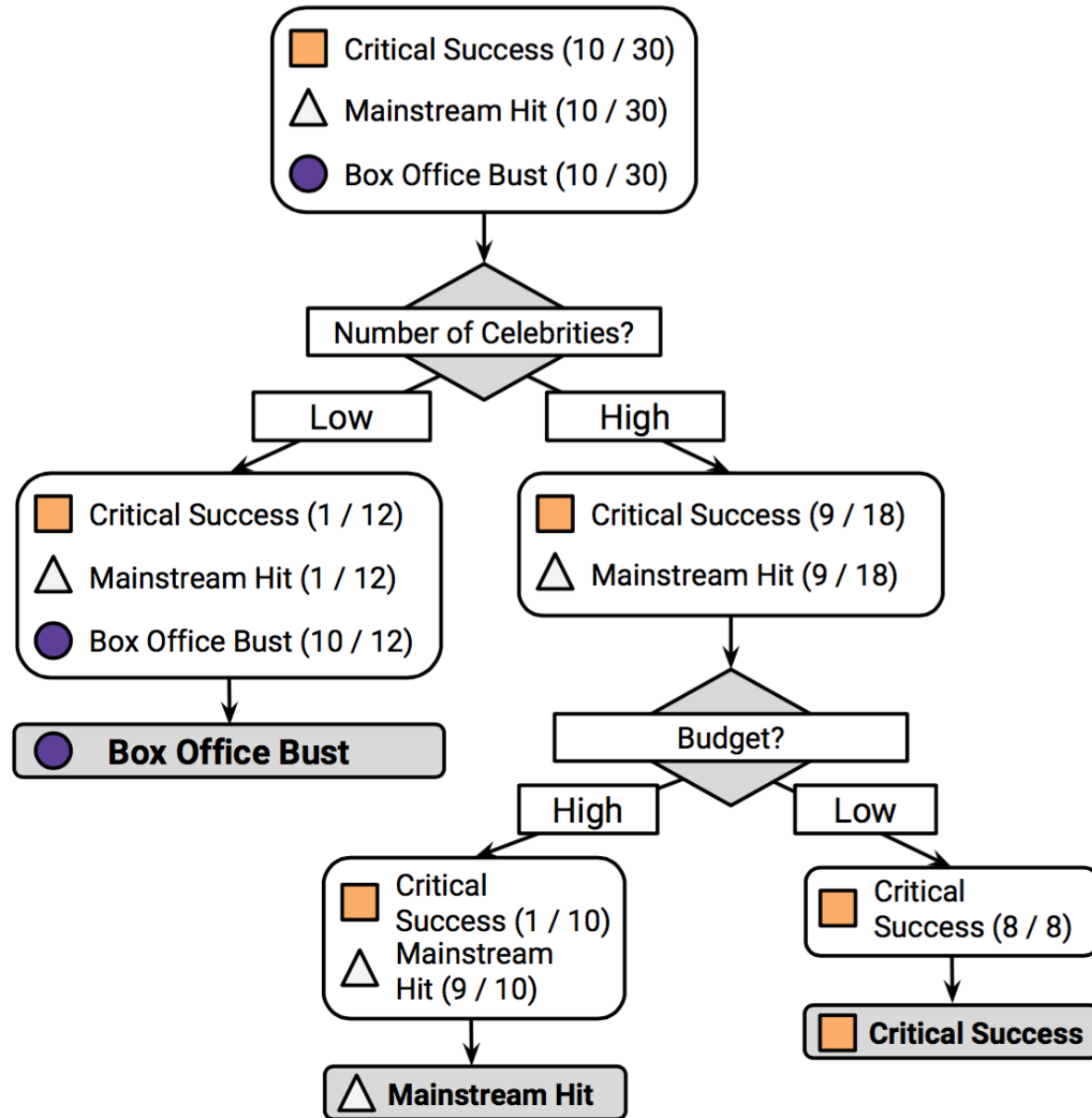


- Step 2



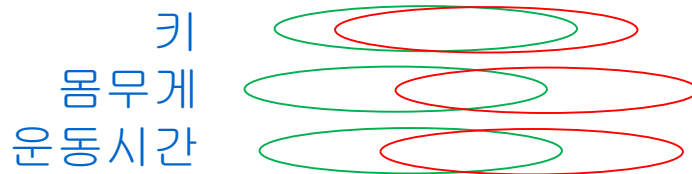


# 결정 트리 생성 과정

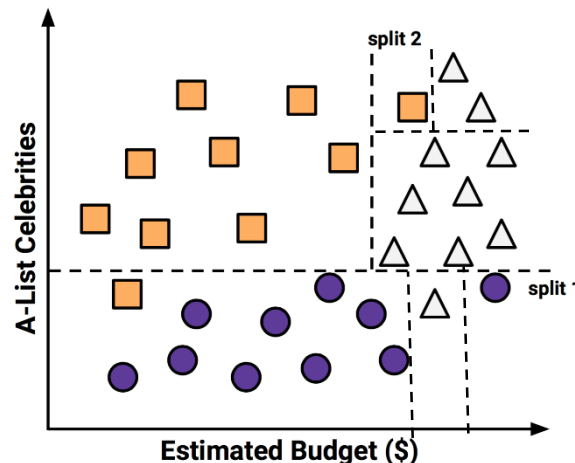


# Issues of decision tree

- 트리의 node 를 선택 할 때 데이터셋에서 어떤 속성부터 선택할 것인가
  - 키, 몸무게, 운동시간 데이터를 가지고 고혈압 여부를 예측하는 모델을 만들고자 했을 때 root node 에는 어떤 속성이 있어야 하는가?
  - Feature evaluation 이 필요



- 트리를 split 할 때 언제 중단할 것인가
  - 트리의 가지를 계속 뺀어 나가면 모든 instance를 100% 식별할 수 있는 tree 를 만들 수 있다. => overfitting 발생
  - 따라서 적당할 때 트리 생성을 중단해야 한다 => 가지치기(pruning)



- 장점

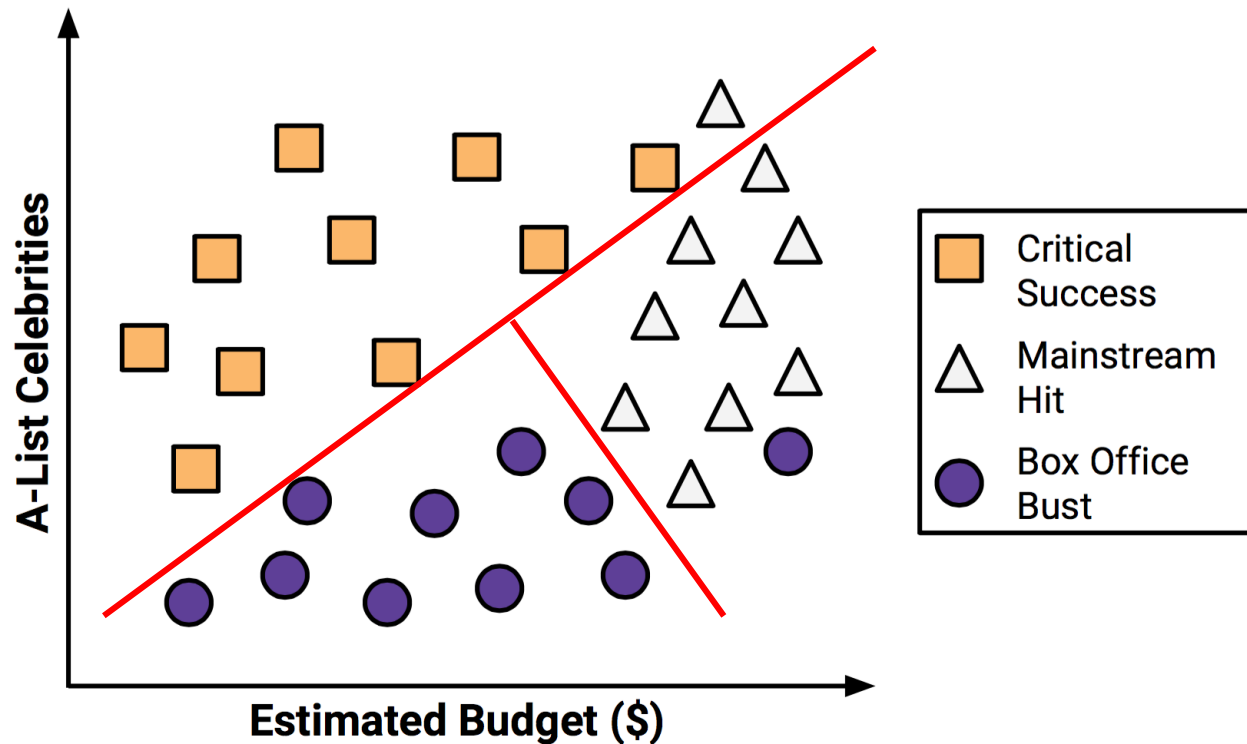
- 모든 문제에 적합하다
- 결측치, 명목속성, 수치속성을 처리하기에 용이
- 여러 속성중 중요한 속성들만을 사용하여 예측
- 매우 많은 수 또는 상대적으 적은 훈련 데이터로도 모델 구축 가능
- 수학적 배경이 없이도 해석 가능한 모델
- 단순한 이론적 근거에 비해 높은 효율성

- 단점

- 결정 트리는 다수의 레벨을 가진 속성쪽 으로 구분하는 경향이 있음
- 모델이 쉽게 과적합하거나 과소적합화 됨
- 축에 평행한 구분선을 사용하기 때문에 일부 관계를 모델화 하는데 문제가 있음
- 훈련 데이터에 대한 약간의 변경이 결정 논리에 큰 변화를 줌
- 큰 트리는 이해하기 어렵고 직관적이지 않음

# Decision tree 모델의 장단점

- Decision tree 는 아래와 같은 구분선은 만들 수 없다 (아래 구분선은 의사결정 규칙으로 표현이 안된다)



# Decision tree algorithms

- ID3 (Iterative Dichotomiser 3)
- C4.5 (successor of ID3)
- C5.0 (successor of C4.5)
- CART (Classification And Regression Tree)
- CHAID (CHi-squared Automatic Interaction Detector)
  - 이 알고리즘은 분류 트리를 계산할 때 다단계 분할을 수행
- MARS (Multivariate adaptive regression splines)
  - 더 많은 수치 데이터를 처리하기 위해 결정 트리를 사용
- 조건부 추론 트리 (Conditional Inference Trees)
  - 과적합을 피하기 위해 여러 테스트에 대해 보정 분할 기준으로 비 - 파라미터 테스트를 사용하는 통계 기반의 방법.
  - 이 방법은 편견 예측 선택 결과와 가지 치기가 필요하지 않다.

## C5.0 기초

- algorithm used to generate a decision tree developed by Ross Quinlan.
- C5.0 is an extension of Quinlan's earlier ID3 and C4.5 algorithms.

- 
- Dataset : credit.csv
  - R package : {C50}

- credit.csv

- 신용정보를 보고 대출을 해줄지 말지를 결정하고자함
- credit.csv 는 과거 대출자의 신용정보 및 채무 불이행 여부 정보를 포함
- 1000 개의 관측값, 16 개의 변수

```
"checking_balance"      "months_loan_duration"  
"credit_history"        "purpose"  
"amount"                "savings_balance"  
"employment_duration"  "percent_of_income"  
"years_at_residence"   "age"  
"other_credit"         "housing"  
"existing_loans_count"  "job"  
"dependents"           "phone"  
"default"
```

"default" : 클래스 정보. Yes-채무불이행, No-채무이행

```
library(C50)          # for C5.0
library(gmodels)      # for CrossTable
# load dataset
credit <- read.csv("credit.csv")
str(credit)

# make train/test data
set.seed(12345)
credit_rand <- credit[order(runif(1000)), ]
credit_train <- credit_rand[1:900, ]
credit_test  <- credit_rand[901:1000, ]

# generate model (decision tree)
credit_model <- C5.0(credit_train[-17], credit_train$default)
```



## `credit_model`

Call:

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

Classification Tree

Number of samples: 900

Number of predictors: 16

Tree size: 67

Non-standard options: attempt to group attributes

## summary(credit\_model)

Decision tree:

```
checking_balance = unknown: no (358/44)
checking_balance in {< 0 DM,> 200 DM,1 - 200 DM}:
:...credit_history in {perfect,very good}:
:   ...dependents > 1: yes (10/1)
:   :   dependents <= 1:
:   :       ...savings_balance = < 100 DM: yes (39/11)
:   :       savings_balance in {> 1000 DM,500 - 1000 DM,unknown}: no (8/1)
:   :       savings_balance = 100 - 500 DM:
:   :           ...checking_balance = < 0 DM: no (1)
:   :           checking_balance in {> 200 DM,1 - 200 DM}: yes (5/1)
credit_history in {critical,good,poor}:
:...months_loan_duration <= 11: no (87/14)
:   months_loan_duration > 11:
:       ...savings_balance = > 1000 DM: no (13)
:       savings_balance in {< 100 DM,100 - 500 DM,500 - 1000 DM,unknown}:
:           ...checking_balance = > 200 DM:
:               ...dependents > 1: yes (3)
:               :   dependents <= 1:
:               :       ...credit_history in {good,poor}: no (23/3)
:               :       credit_history = critical:
:               :           ...amount <= 2337: yes (3)
:               :           amount > 2337: no (6)
:       checking_balance = 1 - 200 DM:
:           ...savings_balance = unknown: no (34/6)
:           savings_balance in {< 100 DM,100 - 500 DM,500 - 1000 DM}:
```


```
checking_balance = unknown: no (358/44)
checking_balance in {< 0 DM,> 200 DM,1 - 200 DM}:
:...credit_history in {perfect,very good}:
: ...dependents > 1: yes (10/1)
:   dependents <= 1:
:     :...savings_balance = < 100 DM: yes (39/11)
:       savings_balance in {> 1000 DM,500 - 1000 DM,unknown}: no (8/1)
:       savings_balance = 100 - 500 DM:
:         :...checking_balance = < 0 DM: no (1)
:           checking_balance in {> 200 DM,1 - 200 DM}: yes (5/1)
```

(Line 1 해석)

체크계좌 잔액을 알 수 없으면 분류결과: no (채무이행)

총402명이 이경우에 해당하는데 358명은 정확히 분류가 되고  
44명은 오분류 되었다

Evaluation on training data (900 cases):

Decision Tree			
-----			
Size		Errors	
66		125(13.9%)	<< 
-----			
Predict			
	(a)	(b)	<-classified as
-----			
Fact	609	23	(a): class no
	102	166	(b): class yes

생성된 decision rule 의 개수 : 66  
900 case 중 오분류 개수 : 125 (13.9%)

Attribute usage:

100.00% checking\_balance  
60.22% credit\_history  
53.22% months\_loan\_duration  
49.44% savings\_balance  
30.89% job  
25.89% other\_credit  
17.78% dependents  
9.67% existing\_loans\_count  
7.22% percent\_of\_income

전체 decision rule 중 60.22% 에 사용됨

```
# perform predict
credit_pred <- predict(credit_model, credit_test)
credit_pred      # prediction result for test data
mean(credit_pred == credit_test$default) # print accuracy

# Analysis of prediction result
CrossTable(credit_test$default, credit_pred,
            prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
            dnn = c('actual default', 'predicted default'))
```

Total Observations in Table: 100

		predicted default		
actual default		no	yes	Row Total
채무이행 no		57	11	68
		0.570	0.110	
채무불이행 yes		16	16	32
		0.160	0.160	
Column Total		73	27	100

## C5.0 Adaptive boosting

- C5.0 의 분류 정확도를 높일수 있는 방법
  - Boosting : 성능이 낮은 학습기 여러 개를 합쳐서 성능을 높이는 machine learning 기법
  - C5.0의 adaptive boosting : decision tree 를 다수 만들어서 각각 분류작업을 하게 한 뒤 그 결과를 모아 최종 결정을 하게함

# C5.0 Adaptive boosting

```
# apply adaptive boosting
credit_boost10 <- C5.0(credit_train[-17], credit_train$default,
                        trials = 10)

summary(credit_boost10)
```

Tree 를 10개 생성

생성된 Tree 10개를 볼수 있음

Evaluation on training data (900 cases):

Trial	Decision Tree	
	Size	Errors
0	66	125 (13.9%)
1	40	205 (22.8%)
2	46	196 (21.8%)
3	45	193 (21.4%)
4	68	163 (18.1%)
5	62	175 (19.4%)
6	56	186 (20.7%)
7	62	188 (20.9%)
8	66	156 (17.3%)
9	49	200 (22.2%)
boost		31 ( 3.4%) <<

(a)	(b)	<-classified as
626	6	(a): class no
25	243	(b): class yes

Error 가 많이 줄었다

## C5.0 Reduce false Positive

- 대출의 경우 잘못 대출했다가 회수를 못하게 되었을 때 은행이 입는 피해가 크다. (판단을 no (채무이행) 으로 내린 경우 – false positive)
- 이를 해결하는 방법
  - Yes 로 해야할지 No 로 해야할지 분명하지 않으면 yes (채무불이행)로 판단
  - 즉, Yes 와 No 의 경계지대에 있으면 Yes 로 판단 한다
- C5.0 에서는 고비용 실수에 대해 트리를 활성화 하지 못하도록 하기 위해서 오류에 대해 벌점 부여 가능
  - 어느정도 벌점을 부여할지를 비용 매트릭스 (cost matrix) 로 표현

(실제)

(예측)

	no	yes
no	0	4
yes	1	0

해석 : TP, TN 에는 벌점을 부여하지 않음  
FP 에는 벌점 4, FN 에는 벌점 1 부여

결국 조절하려는 에러에 가중치를 두는 것이다



## C5.0 Reduce false Positive

```
# create cost table
error_cost <- matrix(c(0, 1, 4, 0), nrow = 2)
error_cost

# apply cost table
credit_cost <- C5.0(credit_train[-17], credit_train$default,
                    costs = error_cost)

# prediction
credit_cost_pred <- predict(credit_cost, credit_test)

# analysis of prediction
CrossTable(credit_test$default, credit_cost_pred,
            prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
            dnn = c('actual default', 'predicted default'))
```

# C5.0 Reduce false Positive

Total Observations in Table: 100

별점  
부여전

actual default	predicted default		Row Total
	no	yes	
no	57 0.570	11 0.110	68
yes	16 0.160	16 0.160	32
Column Total	73	27	100



Total Observations in Table: 100

별점  
부여후

actual default	predicted default		Row Total
	no	yes	
no	42 0.420	26 0.260	68
yes	6 0.060	26 0.260	32
Column Total	48	52	100

# C5.0 Modify decision rule

- C5.0 은 생성된 model (decision rule) 을 수정할 수 있는 기능 제공

```
# create decision rule
credit_rule <- C5.0(credit_train[-17], credit_train$default,
                    rules = TRUE)
summary(credit_rule)
```

Rules:

Rule 1: (15, lift 1.3)  
checking\_balance in {< 0 DM, > 200 DM, 1 - 200 DM}  
months\_loan\_duration > 11  
savings\_balance = > 1000 DM  
-> class no [0.941]

Rule 2: (18/1, lift 1.3)  
checking\_balance = > 200 DM  
amount > 2337  
dependents <= 1  
-> class no [0.900]

## C5.0 Modify decision rule

```
# save decision rules as text file
ruleText = credit_rule$rules
write(ruleText, file= "ruleText.txt")

# after modify ruleText.txt
ruleText = paste(readLines("ruleText.txt"), collapse= "\n")
credit_rule$rules = ruleText
```

# C5.0 Modify decision rule

Rules:

```
Rule 1: (15, lift 1.3)
  checking_balance in {< 0 DM, > 200 DM, 1 - 200 DM}
  months_loan_duration > 11
  savings_balance = > 1000 DM
  -> class no [0.941]
```

```
Rule 2: (18/1, lift 1.3)
  checking_balance = > 200 DM
  amount > 2337
  dependents <= 1
  -> class no [0.900]
```

ruleText.txt

```
id="See5/C5.0 2.07 GPL Edition 2016-08-10"
entries="1"
rules="21" default="no"
```

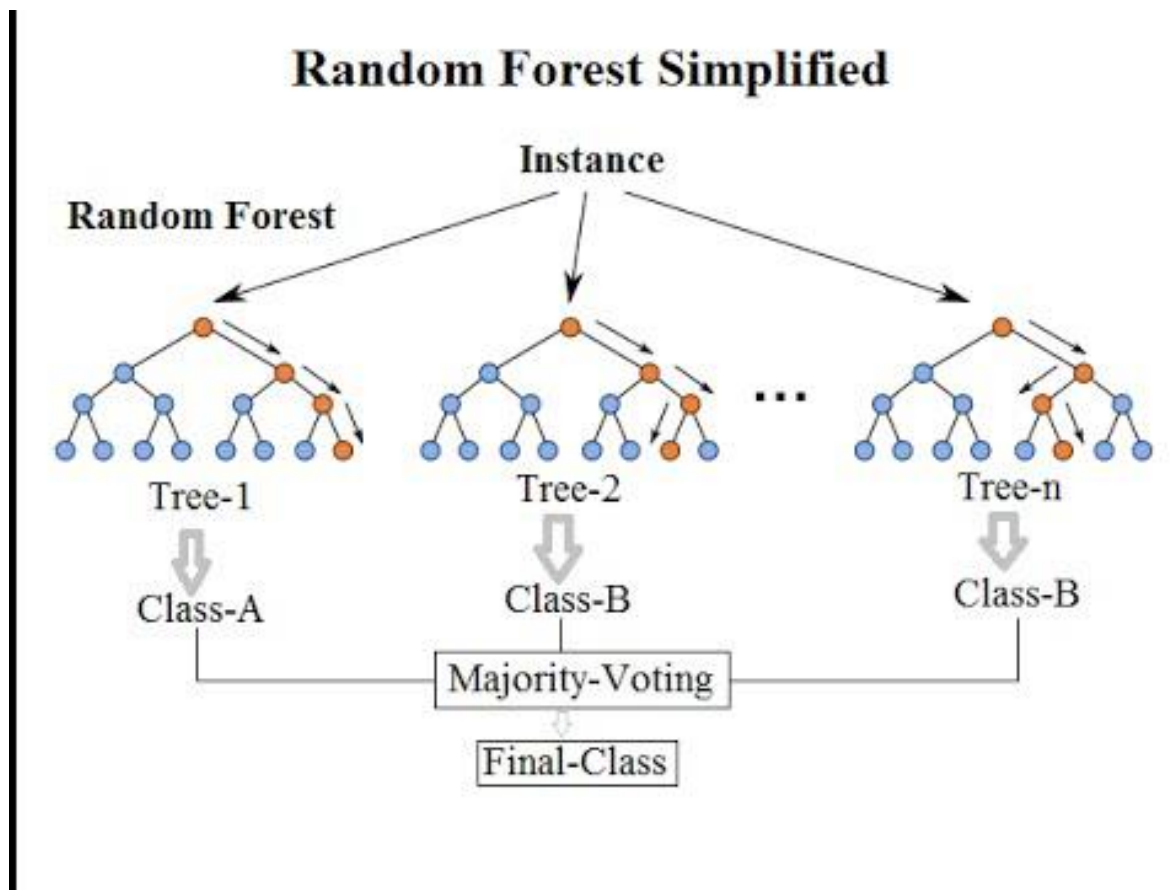
```
conds="3" cover="15" ok="15" lift="1.34028" class="no"
type="3" att="checking_balance" elts="< 0 DM", "> 200 DM", "1 - 200 DM"
type="2" att="months_loan_duration" cut="11" result=">"
type="1" att="savings_balance" val="> 1000 DM"
```

```
|conds="3" cover="18" ok="17" lift="1.28165" class="no"
type="1" att="checking_balance" val="> 200 DM"
type="2" att="amount" cut="2337" result=">"
type="2" att="dependents" cut="1" result="<"
conds="1" cover="358" ok="314" lift="1.24604" class="no"
type="1" att="checking_balance" val="unknown"
conds="2" cover="42" ok="36" lift="1.1975" class="no"
type="1" att="checking_balance" val="unknown"
```

- R 에서 제공 하는 다른 decision tree package 비교
  - R에는 의사결정나무 분석을 할 수 있는 패키지가 여러개 존재. 그 중 대표적인 3개의 패키지 : tree, rpart, party
  - tree 패키지는 binary recursive partitioning을, rpart 패키지는 CART(classification and regression trees) 방법론을 사용
    - 이 패키지들은 엔트로피, 지니계수를 기준으로 가지치기를 할 변수를 결정하기 때문에 상대적으로 연산 속도는 빠르지만 과적합화의 위험성이 존재. Pruning 필요
  - party 패키지는 Unbiased recursive partitioning based on permutation tests 방법론을 사용
    - p-test를 거친 Significance를 기준으로 가지치기를 할 변수를 결정하기 때문에 biased 될 위험이 없어 별도로 Pruning할 필요가 없다는 장점
    - 입력 변수의 레벨이 31개 까지로 제한
  - 예제 : <http://www.dodomira.com/2016/05/29/564/>

# Random Forest

- N개의 Decision Tree가 투표를 통해 결정하는 방식



# Random Forest

- " Bagging은 Bootstrap Aggregation의 약자인데 주어진 데이터(training set)에서 랜덤하게 subset을 N번 sampling해서 (좀 더 정확하게는 observations과 features들을 random하게 sampling) N개의 예측모형을 만들어 개별 예측모형이 voting하는 방식으로 예측결과를 결정하여 Low Bias는 유지하고 High Variance는 줄이는 방법이다.
- Random Forest는 이런 Bagging 계열의 가장 대표적이고 예측력 좋은 알고리즘이다. 예측결과와 정확성(Low Bias)은 개별 예측모형에 쓰이는 알고리즘(decision tree)의 평균값으로 유지되는 반면 낮은 안정성(High Variance)은 Central Limit Theorem에 의해 낮아진다.



# Random Forest

- R package : randomForest

```
library(randomForest)
data(iris)
set.seed(71)
ind <- sample(2, nrow(iris), replace = TRUE, prob=c(0.8, 0.2))
iris.rf <- randomForest(Species ~ ., data=iris[ind == 1,])
iris.pred <- predict(iris.rf, iris[ind == 2,])

table(observed = iris[ind==2, "Species"], predicted =
iris.pred)

acc <- mean(iris.pred==iris[ind == 2,5])
cat("accuracy:", acc, "\n")
```

## [Exercise]

- Using mushrooms.csv,
    - test C5.0
    - test RandomForest
- 
- train:test = 60:40
  - Show CrossTable as a result

type	cap_shape	cap_surface	cap_color	bruises	odor	gill_attach	gill_spacin	gill_size	gill_color	stalk_shape	stalk_size
poisonous	convex	smooth	brown	yes	pungent	free	close	narrow	black	enlarging	equ
edible	convex	smooth	yellow	yes	almond	free	close	broad	black	enlarging	club
edible	bell	smooth	white	yes	anise	free	close	broad	brown	enlarging	club
poisonous	convex	scaly	white	yes	pungent	free	close	narrow	brown	enlarging	equ
edible	convex	smooth	gray	no	none	free	crowded	broad	black	tapering	equ
edible	convex	scaly	yellow	yes	almond	free	close	broad	brown	enlarging	club
edible	bell	smooth	white	yes	almond	free	close	broad	gray	enlarging	club
edible	bell	scaly	white	yes	anise	free	close	broad	brown	enlarging	club
poisonous	convex	scaly	white	yes	pungent	free	close	narrow	pink	enlarging	equ
edible	bell	smooth	yellow	yes	almond	free	close	broad	gray	enlarging	club
edible	convex	scaly	yellow	yes	anise	free	close	broad	gray	enlarging	club