# Data Mining
# Classification: Basic Concepts, Decision Trees, and Model Evaluation

Lecture Notes for Chapter 4

# Classification: Definition

- Given a collection of records (*training set* )
  - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Find a *model* for class attribute as a function of the values of other attributes.
- 목표: **클래스가 결정되지 않은 레코드에 대해서, 가능한 정확하게 클래스를 부여하는 것**
  - A *test set* is used to determine the accuracy of the model
  - Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

# 척추동물 데이터 집합

| Name | Body Temperature | Skin Cover | Gives Birth | Aquatic Creature | Aerial Creature | Has Legs | Hiber-nates | Class Label |
|---|---|---|---|---|---|---|---|---|
| human | warm-blooded | hair | yes | no | no | yes | no | mammal |
| python | cold-blooded | scales | no | no | no | no | yes | reptile |
| salmon | cold-blooded | scales | no | yes | no | no | no | fish |
| whale | warm-blooded | hair | yes | yes | no | no | no | mammal |
| frog | cold-blooded | none | no | semi | no | yes | yes | amphibian |
| komodo dragon | cold-blooded | scales | no | no | no | yes | no | reptile |
| bat | warm-blooded | hair | yes | no | yes | yes | yes | mammal |
| pigeon | warm-blooded | feathers | no | no | yes | yes | no | bird |
| cat | warm-blooded | fur | yes | no | no | yes | no | mammal |
| leopard shark | cold-blooded | scales | yes | yes | no | no | no | fish |
| turtle | cold-blooded | scales | no | semi | no | yes | no | reptile |
| penguin | warm-blooded | feathers | no | semi | no | yes | no | bird |
| porcupine | warm-blooded | quills | yes | no | no | yes | yes | mammal |
| eel | cold-blooded | scales | no | yes | no | no | no | fish |
| salamander | cold-blooded | none | no | semi | no | yes | yes | amphibian |

입력 속성

클래스 속성
(타겟 속성)

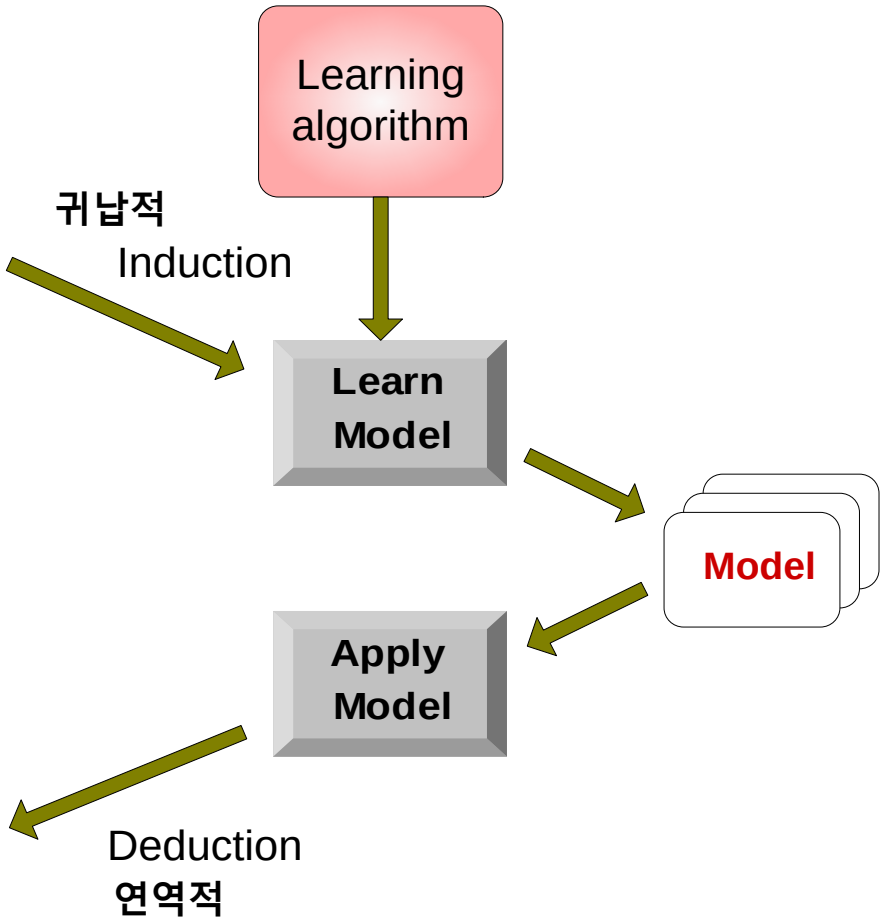
# Illustrating Classification Task

| *Tid* | Attrib1 | Attrib2 | Attrib3 | Class |
|---|---|---|---|---|
| 1 | Yes | Large | 125K | No |
| 2 | No | Medium | 100K | No |
| 3 | No | Small | 70K | No |
| 4 | Yes | Medium | 120K | No |
| 5 | No | Large | 95K | Yes |
| 6 | No | Medium | 60K | No |
| 7 | Yes | Large | 220K | No |
| 8 | No | Small | 85K | Yes |
| 9 | No | Medium | 75K | No |
| 10 | No | Small | 90K | Yes |

Training Set

| *Tid* | Attrib1 | Attrib2 | Attrib3 | Class |
|---|---|---|---|---|
| 11 | No | Small | 55K | ? |
| 12 | Yes | Medium | 80K | ? |
| 13 | Yes | Large | 110K | ? |
| 14 | No | Small | 95K | ? |
| 15 | No | Large | 67K | ? |

Test Set



# 분류 모델의 성능 평가

- Confusion matrix(혼동 행렬)
  - 분류 모델의 성능 평가는 해당 모델에 의해 정확하게 혹은 부정확하게 예측되는 시험 레코드들의 갯수를 기반으로 함
- 예: 이진 분류 문제의 혼동 행렬 사례
  - $f_{ij}$는 클래스 j로 예측된 클래스 i인 레코드들의 수

Class 1 을 0으로 잘못 예측

| | | Predicted Class | |
|---|---|---|---|
| | | Class = 1 | Class =0 |
| Actual Class | Class = 1 | $f_{11}$ | $f_{10}$ |
| | Class = 0 | $f_{01}$ | $f_{00}$ |

* 정확도 = $\frac{\text{정확한 예측 개수}}{\text{총 예측 개수}} = \frac{f11+f00}{f11+f10+f01+f00}$

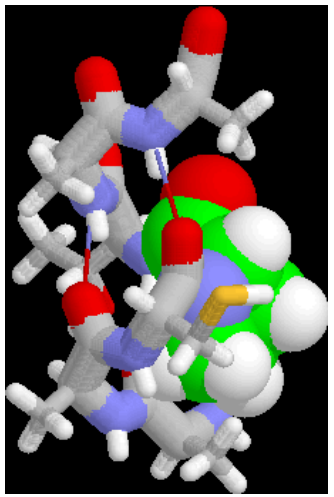
* 오류율 = $\frac{\text{부정확한 예측 개수}}{\text{총 예측 개수}} = \frac{f10+f01}{f11+f10+f01+f00}$

# Examples of Classification Task

- 종양세포(tumor cells)가 양성인지 음성(악성)인지 판별


- 신용카드 거래 트랜잭션이 정상인지 사기인지 구분한다.

- 단백질(protein)의 2차 구조가 alpha-helix인지, beta-sheet인지, random coil인지 분류한다.



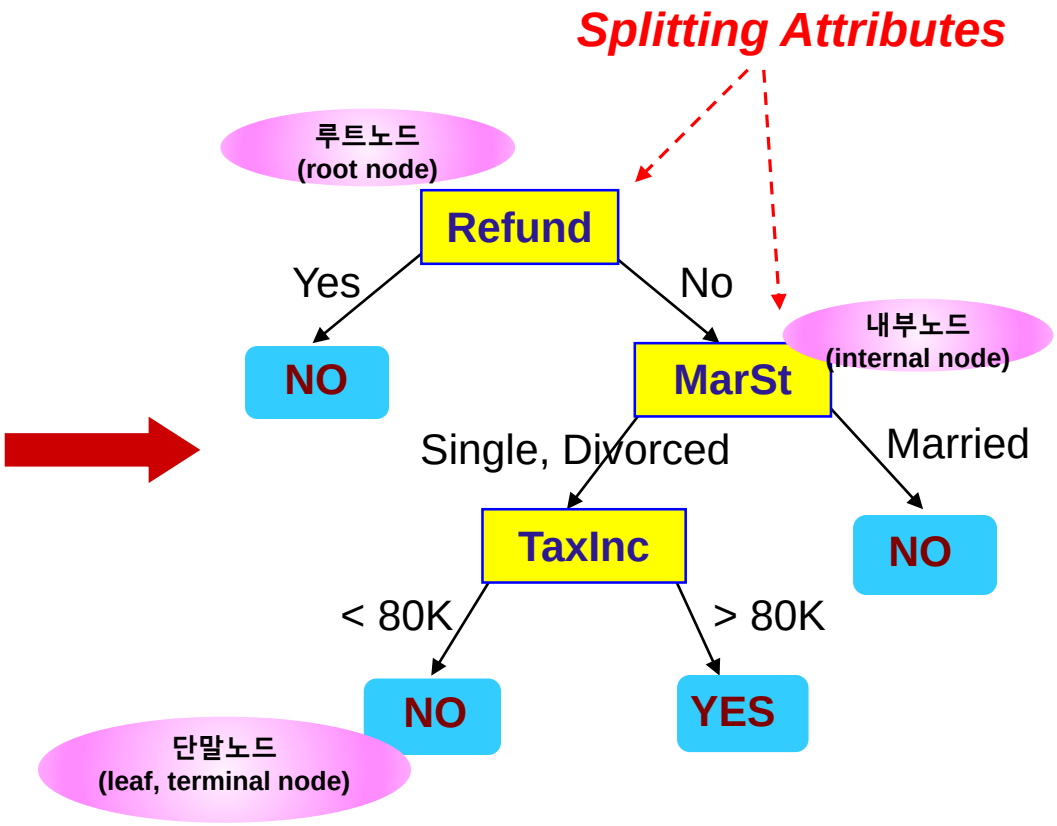
- 신문 기사를 경제, 날씨, 연예, 스포츠 등으로 구분한다.

# Classification Techniques

- Decision Tree based Methods(의사결정 트리)
- Rule-based Methods(규칙 기반 기법)
- Memory based reasoning
- Neural Networks
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines

# Example of a Decision Tree

| Tid | Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|---|
| | | categorical | continuous | class |
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | No | Single | 90K | Yes |



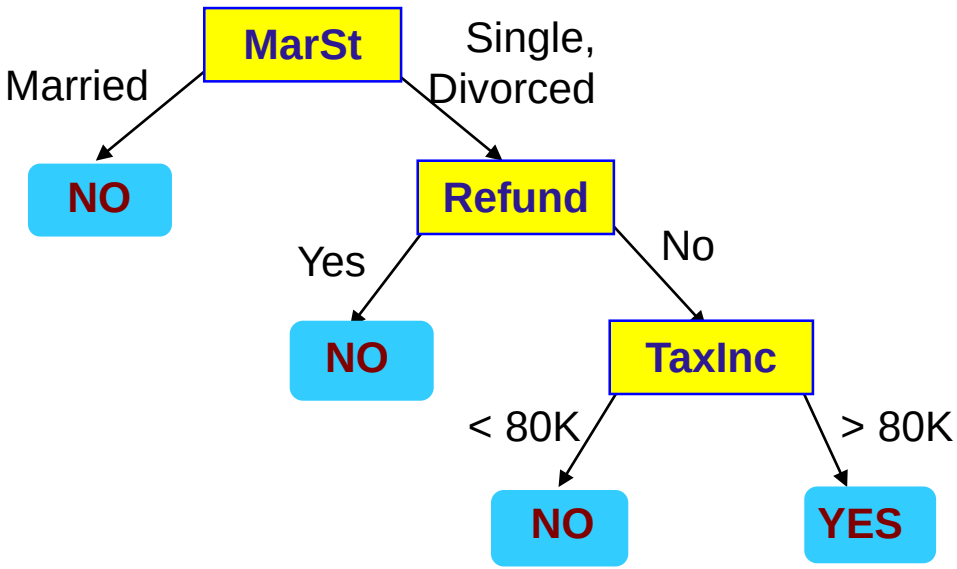

**Training Data**

**Model: Decision Tree**

**Cheat = Defaulted Borrower로 간주**

# Another Example of Decision Tree

| | categorical | categorical | continuous | class |
|---|---|---|---|---|
| *Tid* | **Refund** | **Marital Status** | **Taxable Income** | **Cheat** |
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | No | Single | 90K | Yes |




**There could be more than one tree that fits the same data!**

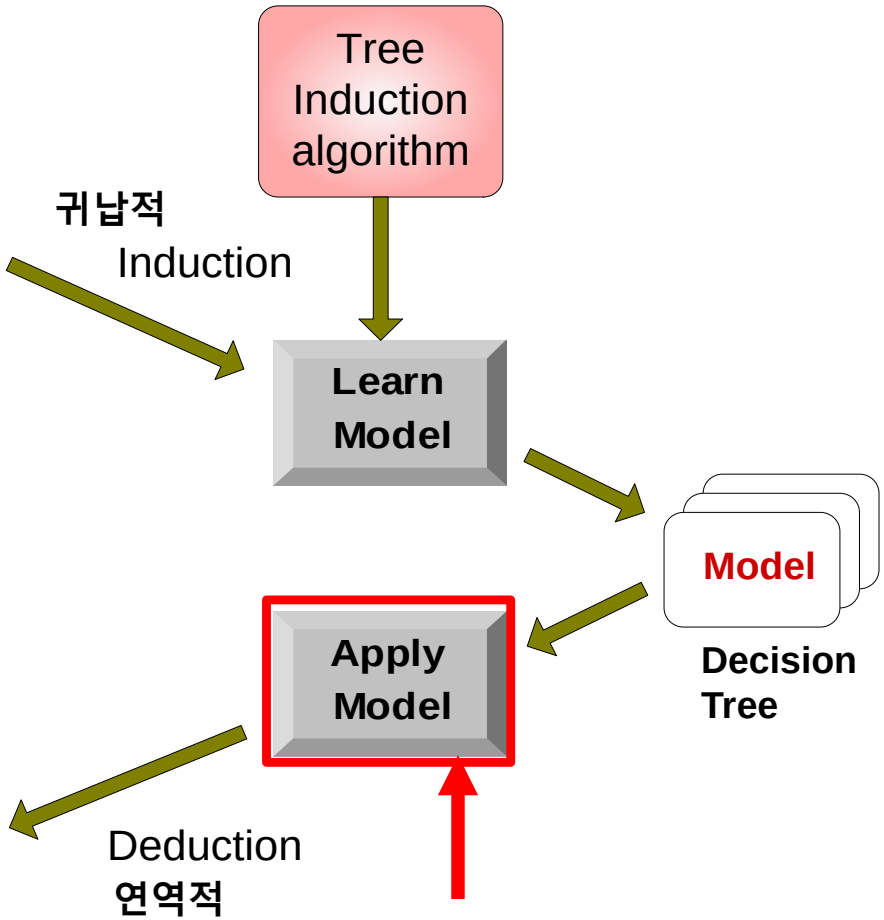
# Decision Tree Classification Task

| *Tid* | Attrib1 | Attrib2 | Attrib3 | Class |
|---|---|---|---|---|
| 1 | Yes | Large | 125K | No |
| 2 | No | Medium | 100K | No |
| 3 | No | Small | 70K | No |
| 4 | Yes | Medium | 120K | No |
| 5 | No | Large | 95K | Yes |
| 6 | No | Medium | 60K | No |
| 7 | Yes | Large | 220K | No |
| 8 | No | Small | 85K | Yes |
| 9 | No | Medium | 75K | No |
| 10 | No | Small | 90K | Yes |

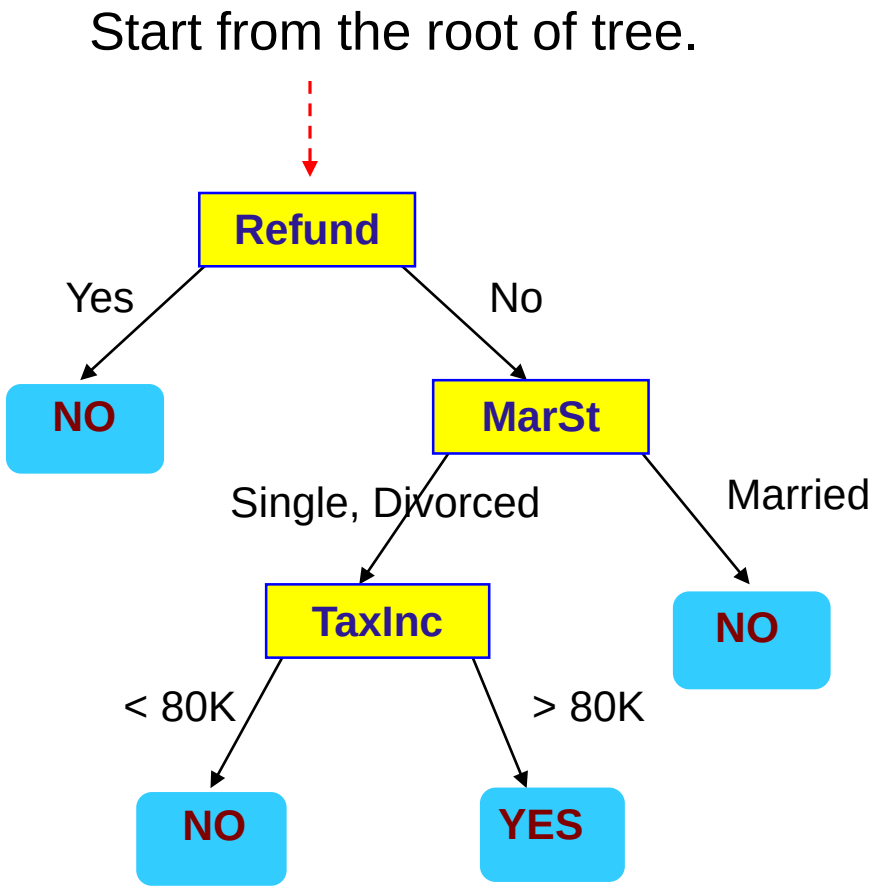
Training Set

| *Tid* | Attrib1 | Attrib2 | Attrib3 | Class |
|---|---|---|---|---|
| 11 | No | Small | 55K | ? |
| 12 | Yes | Medium | 80K | ? |
| 13 | Yes | Large | 110K | ? |
| 14 | No | Small | 95K | ? |
| 15 | No | Large | 67K | ? |

Test Set



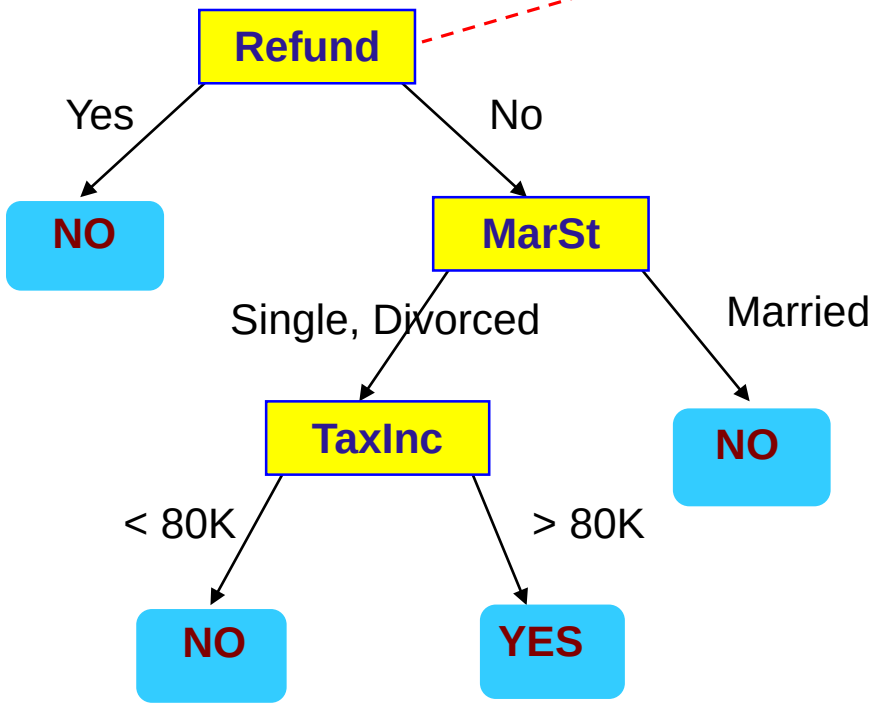
# Apply Model to Test Data




## Test Data

| Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|
| No | Married | 80K | ? |

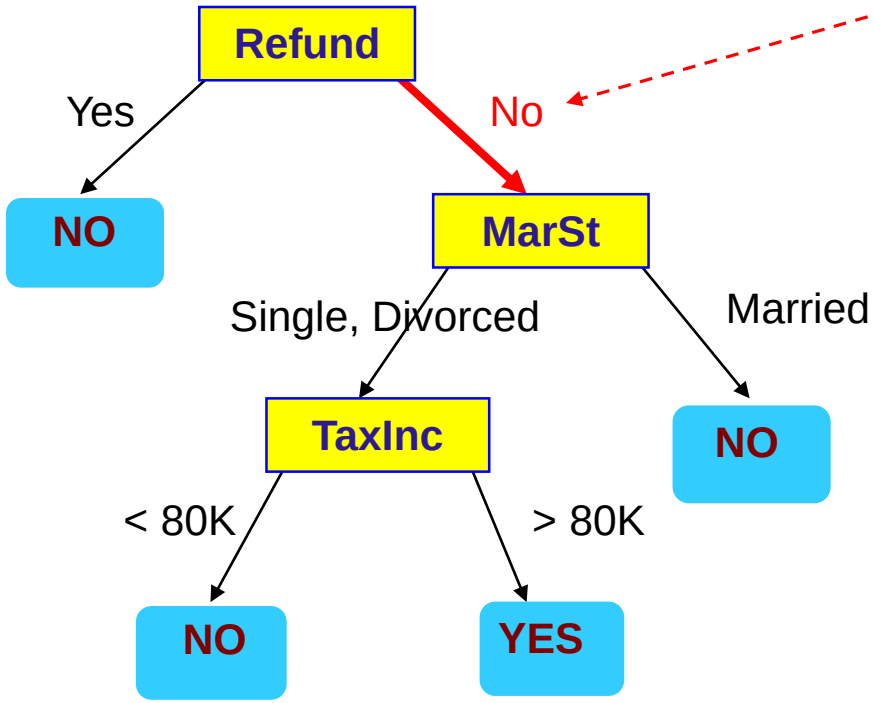
# Apply Model to Test Data




## Test Data

| Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|
| No | Married | 80K | ? |

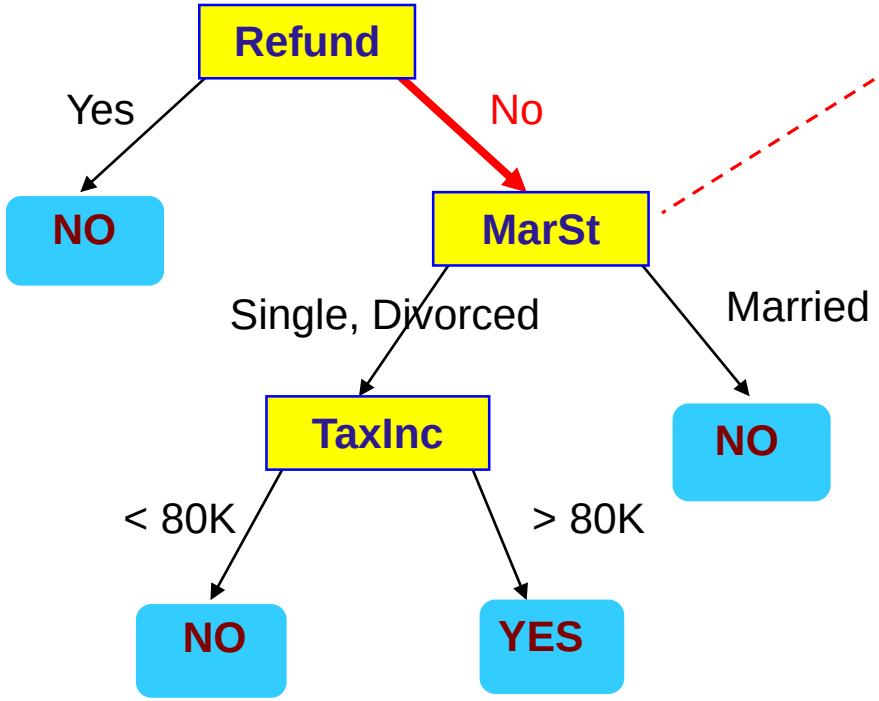
# Apply Model to Test Data




## Test Data

| Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|
| No | Married | 80K | ? |

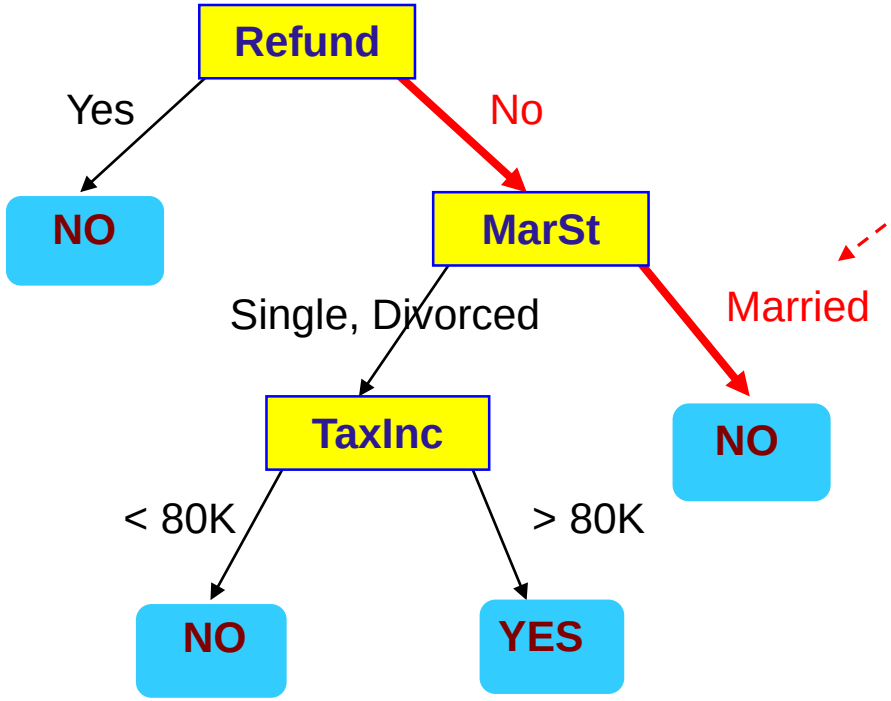
# Apply Model to Test Data




## Test Data

| Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|
| No | Married | 80K | ? |

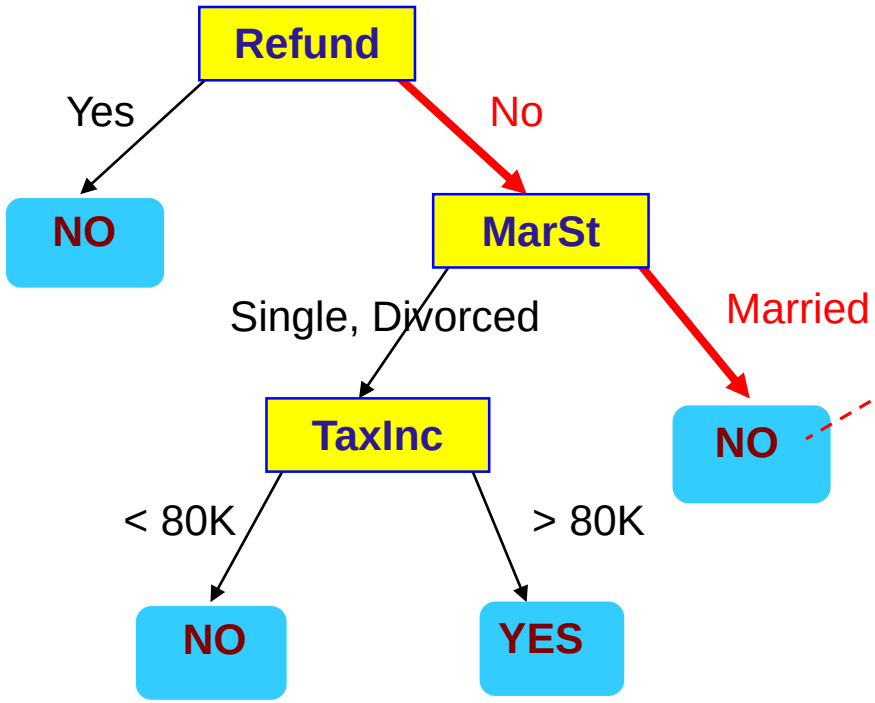
# Apply Model to Test Data




## Test Data

| Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|
| No | Married | 80K | ? |

# Apply Model to Test Data




## Test Data

| Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|
| No | Married | 80K | ? |

Assign Cheat to “No”

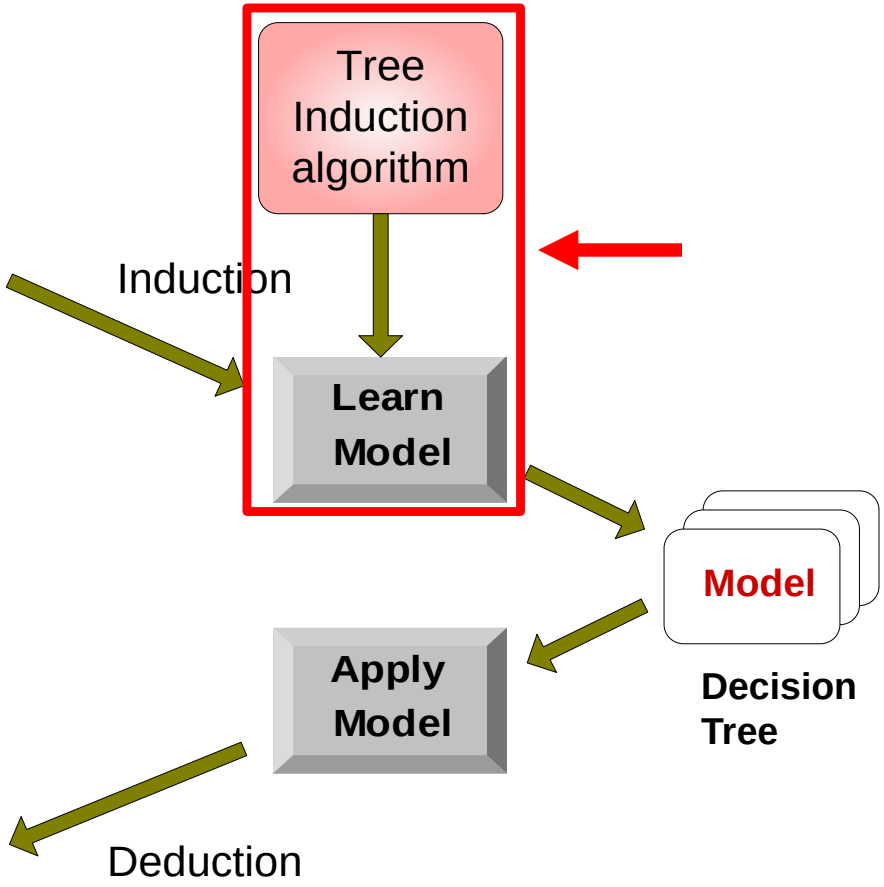
# Decision Tree Classification Task

| *Tid* | Attrib1 | Attrib2 | Attrib3 | Class |
|---|---|---|---|---|
| 1 | Yes | Large | 125K | No |
| 2 | No | Medium | 100K | No |
| 3 | No | Small | 70K | No |
| 4 | Yes | Medium | 120K | No |
| 5 | No | Large | 95K | Yes |
| 6 | No | Medium | 60K | No |
| 7 | Yes | Large | 220K | No |
| 8 | No | Small | 85K | Yes |
| 9 | No | Medium | 75K | No |
| 10 | No | Small | 90K | Yes |

Training Set

| *Tid* | Attrib1 | Attrib2 | Attrib3 | Class |
|---|---|---|---|---|
| 11 | No | Small | 55K | ? |
| 12 | Yes | Medium | 80K | ? |
| 13 | Yes | Large | 110K | ? |
| 14 | No | Small | 95K | ? |
| 15 | No | Large | 67K | ? |

Test Set



# Decision Tree Induction(의사결정트리 구축)

- Many Algorithms:
  - Hunt's Algorithm (one of the earliest)
  - CART
  - ID3, C4.5
  - SLIQ,SPRINT

# Tree Induction (트리 구축)

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.
  - 즉, **특정 기준에 가장 부합하는 속성을 분할 기준으로 선택함**

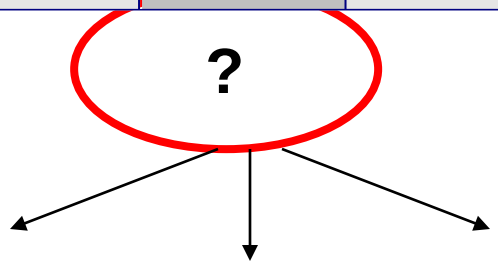
- Issues
  - Determine how to split the records
    - **속성 시험 조건**(attribute test condition)**을 어떻게 지정할 것인가**?
    - **최선의 분할**(best split)**은 어떻게 결정할 것인가**?
  - Determine when to stop splitting

# 헌트 알고리즘의 일반적 구조

- Let $D_t$ be the set of training records that reach a node t
- General Procedure:
  - ① If $D_t$ contains records that belong the same class $y_t$, then t is a leaf node labeled as $y_t$
  - ② If $D_t$ is an empty set, then t is a leaf node labeled by the default class, $y_d$
  - ③ If $D_t$ contains records that belong to more than one class, use an **attribute test** to split the data into smaller subsets. Recursively apply the procedure to each subset.

class

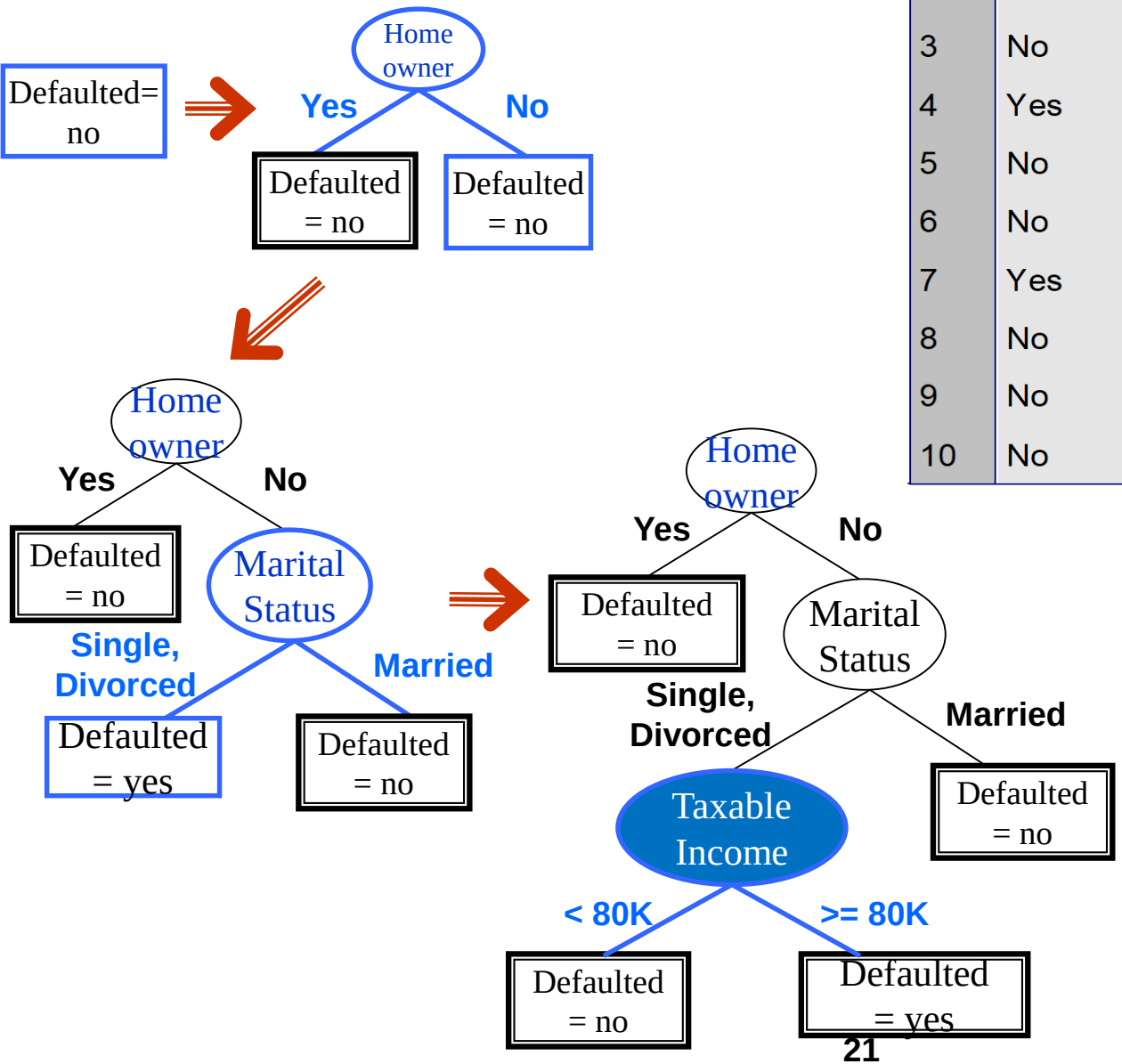
| *Tid* | Home owner | Marital Status | Taxable Income | Defaulted Borrower |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | No | Single | 90K | Yes |




**Cheat = Defaulted Borrower**

# Hunt's Algorithm

| Tid | Home owner | Marital Status | Taxable Income | Defaulted Borrower |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | No | Single | 90K | Yes |



# 속성 시험 조건을 어떻게 표현하나?

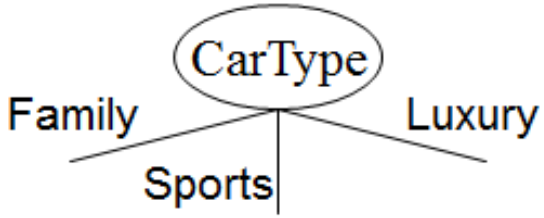
- 속성의 종류에 따라 다름
  - 명목형 (Nominal)
  - 서열형 (Ordinal)
  - 연속형 (Continuous)

- 분할 개수에 따라 다름
  - 이진 분할 (Binary split)
  - 다중 분할 (Multi-way split)

# 명목형 속성에 기반한 분할

- **다중 분할**(Multi-way split):

  **각기 다른 속성 값을 사용하여 가능한 많은 파티션으로 분할한다.**



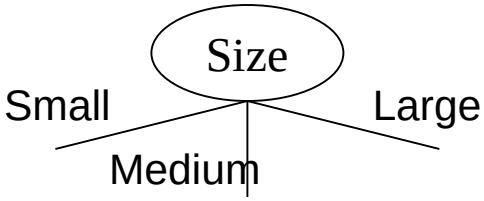

- **이진 분할**(Binary split):

  **속성 값을 두 개의 부분집합으로 분할한다. (최적 파티셔닝이 필요함)**

# 명목형 속성에 기반한 분할

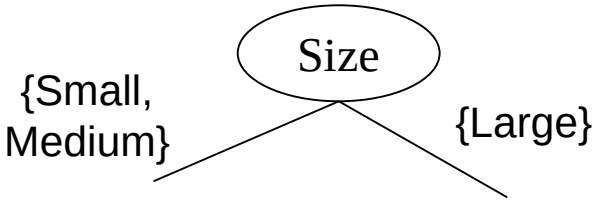
- **다중 분할**:

  각기 다른 속성 값을 사용하여 가능한 많은 파티션으로 분할한다.

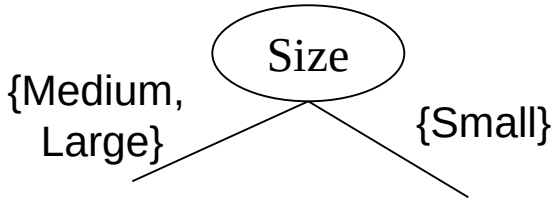



- **이진 분할**:

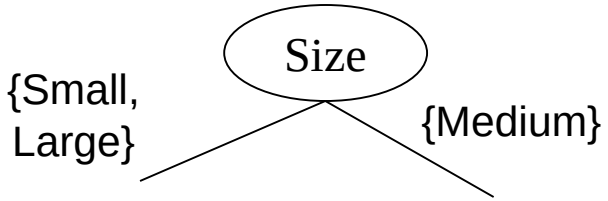
  속성 값을 두 개의 부분집합으로 분할한다. (최적 파티셔닝이 필요함)




OR



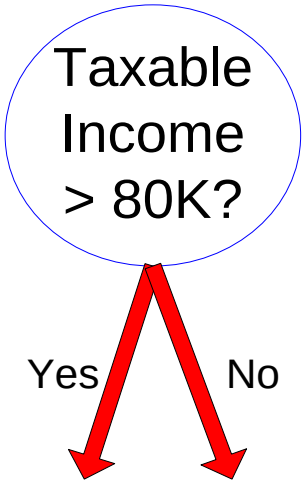

- **그런데, 오른쪽 분할은 어떤가?**



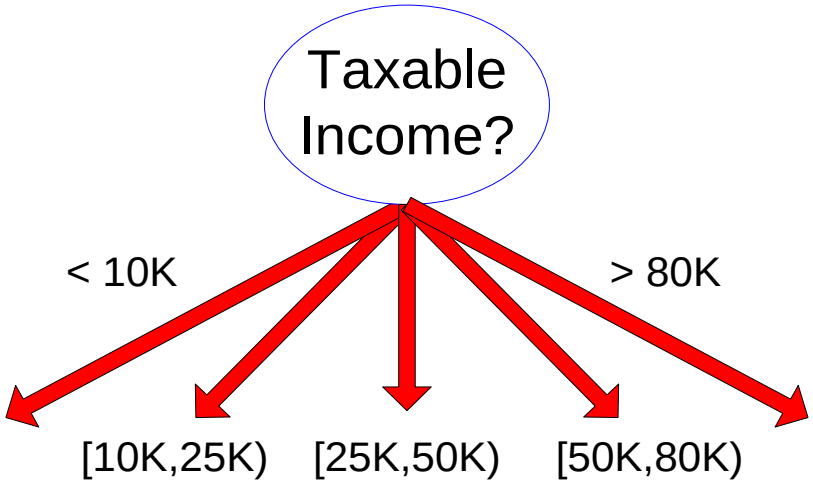
# 연속형 속성에 기반한 분할

- 연속형 속성을 처리하는 두 가지 방법
  - **서열형 속성이 되도록 이산화**(discretization)**를 적용함**
    - 정적 방법: 시작 시점에 이산화를 한번만 적용한다.
    - 동적 방법: 분할이 필요할 때 마다, 동일 너비, 동일 빈도, 클러스터링 등으로 이산화를 적용한다.
  - **이진 결정**(binary decision): $(A < v)$ or $(A \geq v)$
    - 모든 가능한 분할을 고려하고, 이중 최선의 분할을 찾는다.
    - 아주 많은 계산을 필요로 한다.

# 연속형 속성에 기반한 분할




(i) Binary split




(ii) Multi-way split

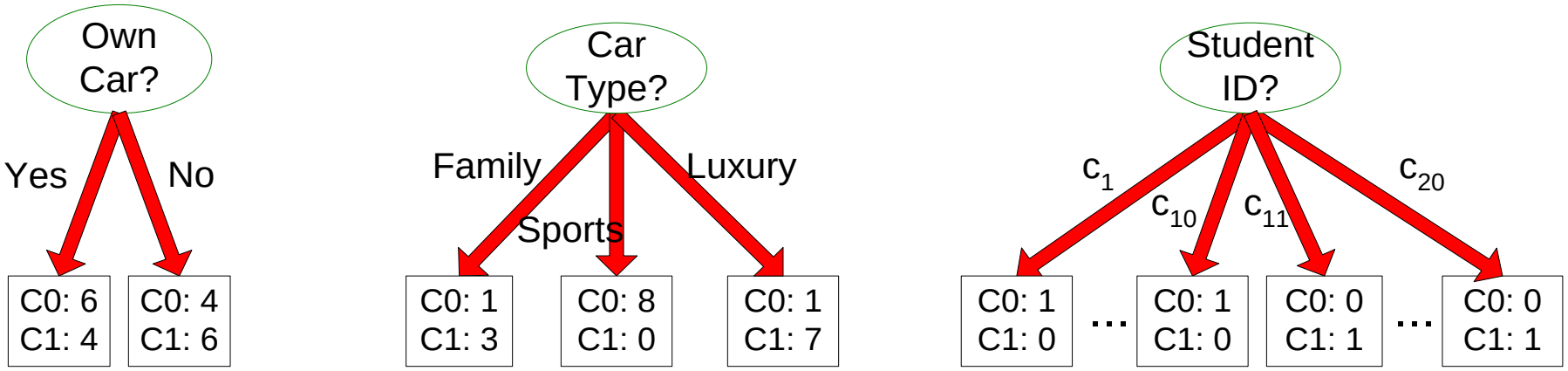
# Tree Induction (트리 구축)

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.
  - 즉, **특정 기준에 가장 부합하는 속성을 분할 기준으로 선택함**

- Issues
  - Determine how to split the records
    - **속성 시험 조건**(attribute test condition)**을 어떻게 지정할 것인가**?
    - **최선의 분할**(best split)**은 어떻게 결정할 것인가**?
  - Determine when to stop splitting

# 최선의 분할을 어떻게 할 것인가?

**Before Splitting: 10 records of class 0,**
**10 records of class 1**




**Which test condition is the best?**

- **첫 번째의** "Own Car"**를 사용하는 경우보다 두 번째의 "Car Type"을 사용하는 경우가 보다 순도(purity)가 높은 분할임**
- **분할을 위해서 불순도(impurity) 혹은 불순 척도(impurity measure) 개념을 도입하고, 이 불순도를 낮추는 방향으로 분할을 시도한다.**

# 최선의 분할을 어떻게 할 것인가?

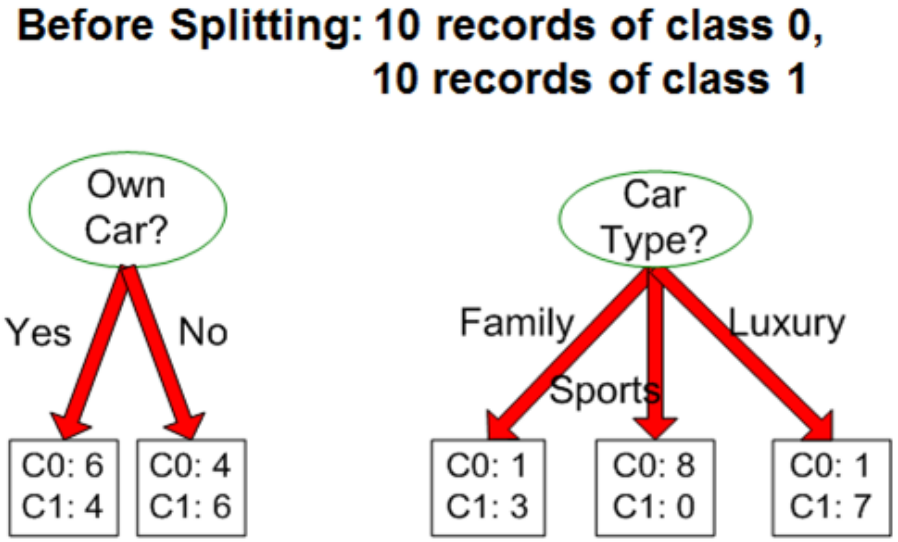
- Greedy approach:
  - 각 노드는 **동종 클래스(homogeneous class) 분포가 되도록 분할한다.**
  - 노드의 불순도를 측정할 필요가 있다

| C0: 5<br>C1: 5 | C0: 9<br>C1: 1 |
|---|---|
| **Non-homogeneous,**<br>**High degree of impurity** | **Homogeneous,**<br>**Low degree of impurity** |

# 최선의 분할을 어떻게 할 것인가?

- 노드에서 클래스 비율을 나타내는 척도 $p(i|t)$
  - 노드 $t$ 에서 클래스 $i$ 를 갖는 레코드들의 비율(분수)
  - 두 클래스 0, 1로 구성된 경우라면, $p(1|t) = 1 - p(0|t)$가 성립한다.
  - 간략히 $p_i$로 나타내기도 한다.

- 오른 예에서,
  - 분할 전 클래스 분포는 (0.5,0.5)이고
  - Own Car로 분할 시
    (0.6,0.4)와 (0.4,0.6)이며,
  - Car Type으로 분할 시
    (1/4,3/4), (1,0), (1/8,7/8)이다.




➔ 각 노드에서 클래스 분포가 편중(skewed)이 되도록 분할하는 것이 좋은 분할임

# 불순도 척도

- Gini Index $1 - \sum_{i=0}^{c-1} [p(i|t)]^2$
- Entropy $-\sum_{i=0}^{c-1} p(i|t) * log_2 p(i|t)$
- Misclassification error

$$1 - max_i [p(i|t)]$$

- 클래스 분류가 잘 되어, 한쪽으로 치우진 경우 불순도는 작으며,
- 클래스 분류가 잘 되지 않아서, 고르게 분포된 경우는 불순도는 큼

# 정보 이득

- **정보이득: 분할 전 부모 노드의 불순도와 분할 후 자식노드들의 불순도의 차이**

$$\Delta = I(parent) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)$$

  - N: 부모노드에서의 레코드 총 수
  - k : 속성값들의 수
  - N(vj) : 자식노드 vj와 관련된 레코드 수

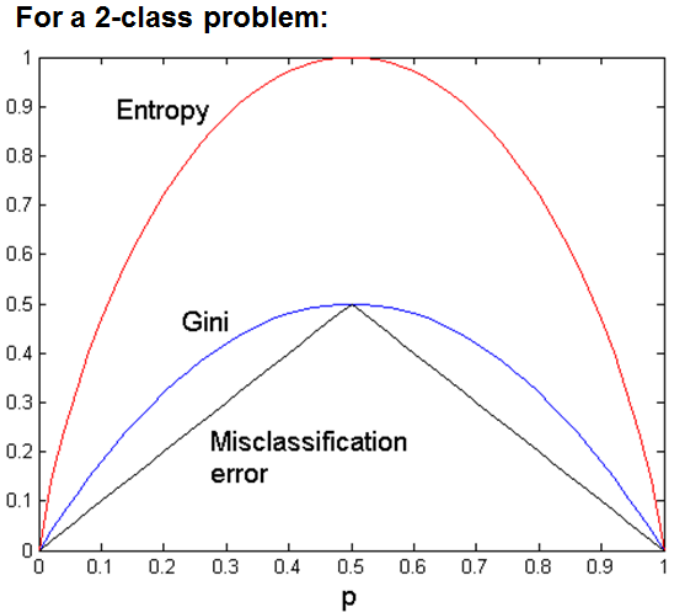
- Gain 값 최대화 = Children노드의 weighted 평균 불순도 값 최소화
- If **I() = 불순도 척도 (예: Entropy),** then $\Delta_{info}$ is called **information gain**

# Tree Induction (트리 구축)

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.
  - 즉, **특정 기준에 가장 부합하는 속성을 분할 기준으로 선택함**

- Issues
  - Determine how to split the records
    - **속성 시험 조건**(attribute test condition)**을 어떻게 지정할 것인가**?
    - **최선의 분할**(best split)**은 어떻게 결정할 것인가**? -- Gini **지수**
  - Determine when to stop splitting

# 불순도 척도: GINI




- Gini Index for a given node t :

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

( p( j | t) 혹은 pj는 노드 t에서 class j에 속하는 레코드의 수).

- **Maximum (1 - 1/$n_c$)** when records are equally distributed among all classes, implying least interesting information
- **Minimum (0)** when all records belong to one class, implying most interesting information

| C1 | **0** |
|---|---|
| C2 | **6** |
| **Gini=0.000** | |

| C1 | **1** |
|---|---|
| C2 | **5** |
| **Gini=0.278** | |

| C1 | **2** |
|---|---|
| C2 | **4** |
| **Gini=0.444** | |

| C1 | **3** |
|---|---|
| C2 | **3** |
| **Gini=0.500** | |

# 불순도 척도: GINI

- **Min value of the index:**
  - **A class with a relative frequency of 1, all the others 0**

$$1 - \sum_{j=1}^{n_c} p_j^2 = 1 - 1^2 = 0$$

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

- **Max value of the index:**
  - **$n_c$ classes with the same frequency:**

$$1 - \sum_{j=1}^{n_c} p_j^2 = 1 - \sum_{j=1}^{n_c} (\frac{1}{n_c})^2 = 1 - n_c(\frac{1}{n_c})^2 = 1 - \frac{1}{n_c}$$

**예: 2개의 클래스 유형을 가지며, 각각 반씩 분포하는 경우: 1- ( (1/2)^2 + (1/2)^2) = 1-1/2 = ½**

# GINI 계산 사례

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

| C1 | **0** |
|---|---|
| C2 | **6** |

**P(C1) = 0/6 = 0  P(C2) = 6/6 = 1**

**Gini = 1 – [ P(C1)$^2$ + P(C2)$^2$ ] = 1 – 0 – 1 = 0**

| C1 | **1** |
|---|---|
| C2 | **5** |

**P(C1) = 1/6  P(C2) = 5/6**

**Gini = 1 – [ (1/6)$^2$ + (5/6)$^2$ ] = 0.278**

| C1 | **2** |
|---|---|
| C2 | **4** |

**P(C1) = 2/6  P(C2) = 4/6**

**Gini = 1 – [ (2/6)$^2$ + (4/6)$^2$ ] = 0.444**

# GINI 기반 분할

- Used in CART, SLIQ, SPRINT.
- When a node p is split into k partitions (children), the quality of split is computed as,

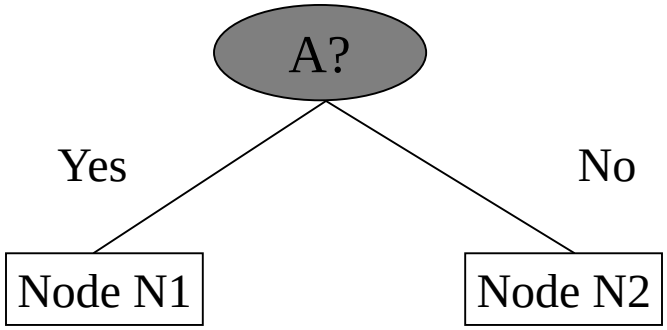
$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

where, $n_i$ = number of records at child i,

n = number of records at node p.

# 이진 속성의 분할: Computing GINI Index

- Splits into two partitions
- Effect of Weighing partitions:
  - Larger and Purer Partitions are sought for.




| | Parent |
|---|---|
| C0 | 6 |
| C1 | 6 |
| Gini = 0.500 | |

| | N1 | N2 |
|---|---|---|
| C0 | 4 | 2 |
| C1 | 3 | 3 |
| Gini=0.486 | | |

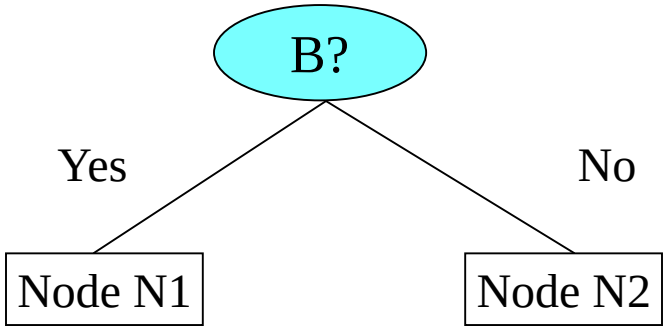
노드 N1의 지니지수
= 1 – [(4/7)² + (3/7)²]
= 0.4898

노드 N2의 지니 지수
= 1 – [(2/5)² + (3/5)²]
= 0.48

Children노드의 Gini 지수 → 가중 평균 필요

= (7/12)*0.4898 +
(5/12) * 0.48 = 0.486

# 이진 속성의 분할: Computing GINI Index




| | Parent |
|---|---|
| C1 | 6 |
| C2 | 6 |
| Gini = 0.500 | |

| | N1 | N2 |
|---|---|---|
| C0 | 1 | 5 |
| C1 | 4 | 2 |
| Gini=0.371 | | |

**Gini(N1)**
**= 1 – $(1/5)^2$ – $(4/5)^2$**
**= 0.32**

**Gini(N2)**
**= 1 – $(5/7)^2$ – $(2/7)^2$**
**= 0.408**

**Gini(Children)가중 평균**
**= 5/12 * 0.32 +**
**7/12 * 0.408**
**= 0.371**

속성 B에 대한 Gini 지수가 더 작으므로 속성 B를 사용한 분할이 속성 A를 사용한 분할보다 더 나은 분할임

# 명목형 속성 분할: Computing Gini Index

- 명목형 속성은 이진 분할 뿐만 아니라, 아래의 예처럼 다중분할(multi-way split) 가능함
  - 다중 분할이 이진분할보다 더 작은 Gini 지수 가짐 (이중 분할은 결국 다중 분할에서 일부 결과를 merge한 것이므로 순도는 낮게 됨)

Multi-way split

| | CarType | | |
|---|---|---|---|
| | Family | Sports | Luxury |
| C1 | 1 | 8 | 1 |
| C2 | 3 | 0 | 7 |
| Gini | 0.163 | | |

Two-way split
(find best partition of values)

| | CarType | |
|---|---|---|
| | {Sports, Luxury} | {Family} |
| C1 | 9 | 1 |
| C2 | 7 | 3 |
| Gini | 0.400 | |

| | CarType | |
|---|---|---|
| | {Sports} | {Family, Luxury} |
| C1 | 8 | 2 |
| C2 | 0 | 10 |
| Gini | 0.167 | |

Gini(Family) = $1 - (1/4)^2 - (3/4)^2$ = 0.375

Gini(Sports) = $1 - (8/8)^2 - (0)^2$ = 0

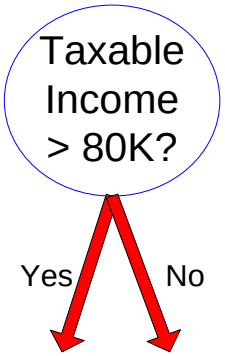
Gini(Luxury) = 1 – (1/8)^2 – (7/8)^2= 0.218

* Weighted Gini = 4/20 * 0.375 + 8/20 * 0.218 = 0.163

# 연속형 속성 분할: Computing Gini Index

- Use Binary Decisions based on one value
- Several Choices for the splitting value
  - Number of possible splitting values = Number of distinct values
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions, A < v and A ≥ v
- Simple method to choose best v
  - For each v, scan the database to gather count matrix and compute its Gini index
  - Computationally Inefficient! Repetition of work.

| *Tid* | Home owner | Marital Status | Taxable Income | Defaulted Borrower |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | No | Single | 90K | Yes |



# 연속형 속성 분할: Computing Gini Index

- 아래의 예는 Taxable Income 속성의 모든 값을 분할 위치 후보로 사용하여 적절한 사이값을 기준으로 분할함 (예, 100과 120 속성에 대해 중간값인 110을 기준으로 분할함)
  - 이 후, 각각의 구간에 대해 Gini 지수 계산
  - 가장 낮은 Gini 값을 가지는 경우는 v=97인 경우임

| Defaulted | No | No | No | Yes | Yes | Yes | No | No | No | No |
|---|---|---|---|---|---|---|---|---|---|---|
| | Taxable Income | | | | | | | | | |
| Sorted Values → | 60 | 70 | 75 | 85 | 90 | 95 | 100 | 120 | 125 | 220 |
| Split Positions → | 55 | 65 | 72 | 80 | 87 | 92 | 97 | 110 | 122 | 172 | 230 |
| | <= > | <= > | <= > | <= > | <= > | <= > | <= > | <= > | <= > | <= > | <= > |
| Yes | 0 3 | 0 3 | 0 3 | 0 3 | 1 2 | 2 1 | 3 0 | 3 0 | 3 0 | 3 0 | 3 0 |
| No | 0 7 | 1 6 | 2 5 | 3 4 | 3 4 | 3 4 | 3 4 | 4 3 | 5 2 | 6 1 | 7 0 |
| Gini | 0.420 | 0.400 | 0.375 | 0.343 | 0.417 | 0.400 | *0.300* | 0.343 | 0.375 | 0.400 | 0.420 |

Gini(값 55 중심) = $1 - (3/10)^2 - (7/10)^2$ = 0.42

Gini(값 65 중심) = $1 - (0)^2 - (0)^2$ = 0

Gini(값 65 중심) = 1 – (3/9)^2 – (6/9)^2= 0.4444

* Weighted Gini(값 65 중심) = 1/10 * 0 + 9/10 * 0.4444 = 0.4

# Tree Induction (트리 구축)

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.
  - 즉, **특정 기준에 가장 부합하는 속성을 분할 기준으로 선택함**

- Issues
  - Determine how to split the records
    - **속성 시험 조건**(attribute test condition)**을 어떻게 지정할 것인가**?
    - **최선의 분할**(best split)**은 어떻게 결정할 것인가**? -- Entropy
  - Determine when to stop splitting

# Alternative Splitting Criteria based on INFO

- Entropy at a given node t:

$$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

  - Measures homogeneity of a node.
    - Maximum (log $n_c$) when records are equally distributed among all classes implying least information
    - Minimum (0.0) when all records belong to one class, implying most information
  - Entropy based computations are similar to the GINI index computations

# Alternative Splitting Criteria based on INFO

- **Min value of the index:**
  - **A class with a relative frequency of 1, all the others 0**

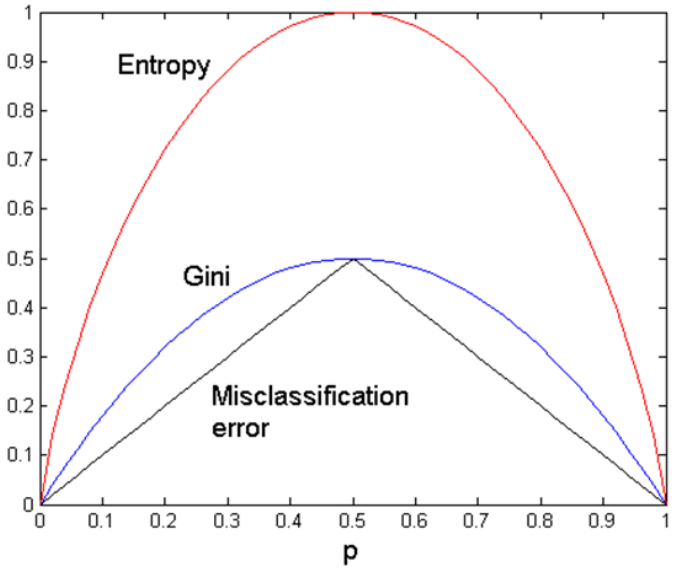
$$-\sum_{j=1}^{n_c}(p_j \log p_j) = -1 \log 1 = 0$$

- **Max value of the index:**
  - **$n_c$ classes with the same frequency:**

$$-\sum_{j=1}^{n_c}(p_j \log p_j) = -\sum_{j=1}^{n_c}(\frac{1}{n_c} \log \frac{1}{n_c}) =$$

$$= -n_c \frac{1}{n_c} \log \frac{1}{n_c} = -(\log 1 - \log n_c) = \log n_c$$

**For a 2-class problem:**



# Examples for computing Entropy

$$Entropy(t) = -\sum_j p(j \mid t) \log_2 p(j \mid t)$$

| C1 | **0** |
|---|---|
| C2 | **6** |

**P(C1) = 0/6 = 0  P(C2) = 6/6 = 1**

**Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0**

| C1 | **1** |
|---|---|
| C2 | **5** |

**P(C1) = 1/6  P(C2) = 5/6**

**Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (5/6) = 0.65**

| C1 | **2** |
|---|---|
| C2 | **4** |

**P(C1) = 2/6  P(C2) = 4/6**

**Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92**

# Splitting Based on INFO...

- Information Gain:

$$GAIN_{split} = Entropy(p) - \left( \sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

Parent Node, p is split into k partitions;

$n_i$ is number of records in partition i

  - Measures Reduction in Entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)
  - Used in ID3 and C4.5
  - Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.

# Splitting Based on INFO...

- Gain Ratio:

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO}$$

$$SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

Parent Node, p is split into k partitions

$n_i$ is the number of records in partition i

  - Adjusts Information Gain by the entropy of the partitioning (SplitINFO). Higher entropy partitioning (large number of small partitions) is penalized!
  - Used in C4.5
  - Designed to overcome the disadvantage of Information Gain

# Tree Induction (트리 구축)

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.
  - 즉, **특정 기준에 가장 부합하는 속성을 분할 기준으로 선택함**

- Issues
  - Determine how to split the records
    - **속성 시험 조건**(attribute test condition)**을 어떻게 지정할 것인가**?
    - **최선의 분할**(best split)**은 어떻게 결정할 것인가**? – Classification Error
  - Determine when to stop splitting

# Examples for Classification Error

$$Error(t) = 1 - \max_i P(i \mid t)$$

| C1 | 0 |
|---|---|
| C2 | 6 |

P(C1) = 0/6 = 0  P(C2) = 6/6 = 1

Error = 1 – max (0, 1) = 1 – 1 = 0

| C1 | 1 |
|---|---|
| C2 | 5 |

P(C1) = 1/6  P(C2) = 5/6

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

| C1 | 2 |
|---|---|
| C2 | 4 |

P(C1) = 2/6  P(C2) = 4/6

Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

# Splitting Criteria based on Classification Error

- Classification error at a node t :

$$Error(t) = 1 - \max_i P(i \mid t)$$

Let $p(i|t)$ denote the fraction of records belonging to class $i$ at a given node $t$. We sometimes omit the reference to node $t$ and express the fraction as $p_i$.

- Measures misclassification error made by a node.
  - Maximum (1 - $1/n_c$) when records are equally distributed among all classes, implying least interesting information
  - Minimum (0.0) when all records belong to one class, implying most interesting information

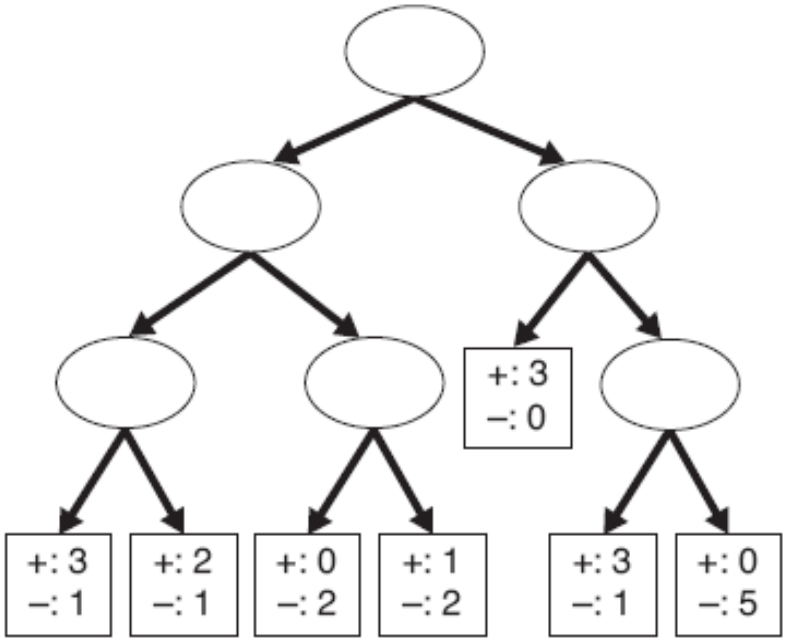



**Figure 4.27.** Example of two decision trees generated from the same training data.

**훈련오류율 e(TL)**

$$1 - \left( \frac{\max(1,3)}{24} + \frac{\max(2,1)}{24} + \frac{\max(2,0)}{24} + \frac{\max(1,2)}{24} + \frac{\max(3,0)}{24} + \frac{\max(3,1)}{24} + \frac{\max(0,5)}{24} \right)$$

$$\frac{1}{6}$$

**훈련오류율 e(TR)**

$$1 - \left( \frac{\max(5,2)}{24} + \frac{\max(1,4)}{24} + \frac{\max(3,0)}{24} + \frac{\max(3,6)}{24} \right)$$

$$\frac{1}{4}$$

# Splitting Criteria based on Classification Error

- **Min value of the index:**
  - **A class with a relative frequency of 1, all the others 0**

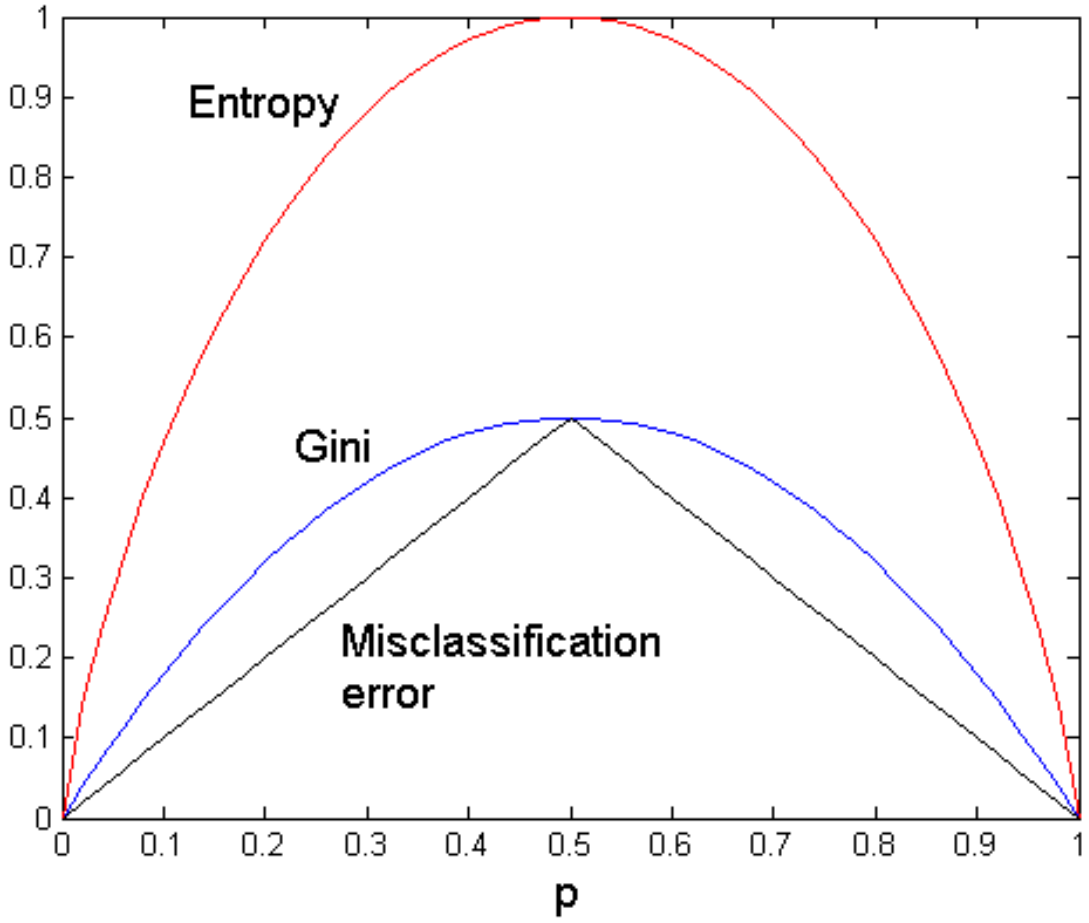
$$1 - \max_{j=1\ldots n_c} p_j = 1 - 1 = 0$$

- **Max value of the index:**
  - **$n_c$ classes with the same frequency:**

$$1 - \max_{j=1\ldots n_c} p_j = 1 - \frac{1}{n_c}$$

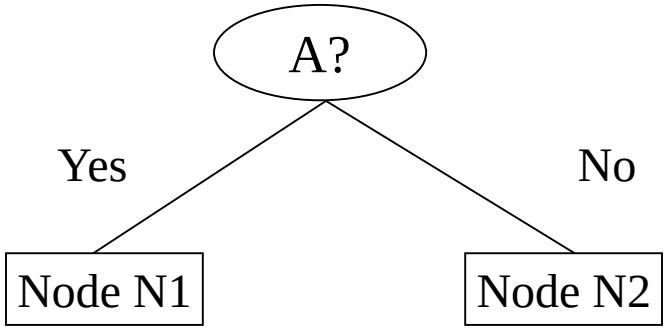
# Comparison among Splitting Criteria

**For a 2-class problem:**




- 특정 클래스의 비율(p)이 아주 작거나 높을 경우엔 불순도가 낮아지나,
- (이진 분류에서) 0.5에 가까운 경우에는 불순도가 높아진다.

# Misclassification Error vs Gini




| | Parent |
|---|---|
| C1 | 7 |
| C2 | 3 |
| Gini = 0.42 | |

ME(Parent) = 1- max(7,3) /10 = 1- 7/10 = 0.3

Gini(N1)
= 1 – $(3/3)^2$ – $(0/3)^2$
= 0

Gini(N2)
= 1 – $(4/7)^2$ – $(3/7)^2$
= 0.489

| | N1 | N2 |
|---|---|---|
| C1 | 3 | 4 |
| C2 | 0 | 3 |
| Gini=0.342 | | |

Gini(Children)
= 3/10 * 0
+ 7/10 * 0.489
= 0.342

Gini improves !!

Misclassification Error
ME(N1)= 1 – max(3,0)/3 = 0

ME(N2)= 1 – max(4,3)/7 = 1-4/7 = 0.428

ME(Children) = 3/10 * 0 + 7/10 * 0.428 = 0.299

# Tree Induction (트리 구축)

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.
  - 즉, **특정 기준에 가장 부합하는 속성을 분할 기준으로 선택함**

- Issues
  - Determine how to split the records
    - **속성 시험 조건**(attribute test condition)**을 어떻게 지정할 것인가**?
    - **최선의 분할**(best split)**은 어떻게 결정할 것인가**? – Classification Error
  - Determine when to stop splitting (분할 멈추는 시점)

# 트리 구축 중단 시점

- 노드에 속하는 모든 레코드들이 동일한 클래스를 갖는 경우,더 이상 분할하지 않고 멈춤
- 노드에 속하는 모든 레코드들이 동일한(유사한) 속성 값을 갖는 경우,더 이상 분할하지 않고 멈춤
- 노드의 레코드 수가 임계치 이하로 떨어지는 경우, 분할을 멈춤
- 그 외, 미리 멈춤(early termination)도 존재함

# 의사결정 트리 기반 분류 장점

- 장점
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Accuracy is comparable to other classification techniques for many simple data sets

# 의사결정 트리 사례: C4.5

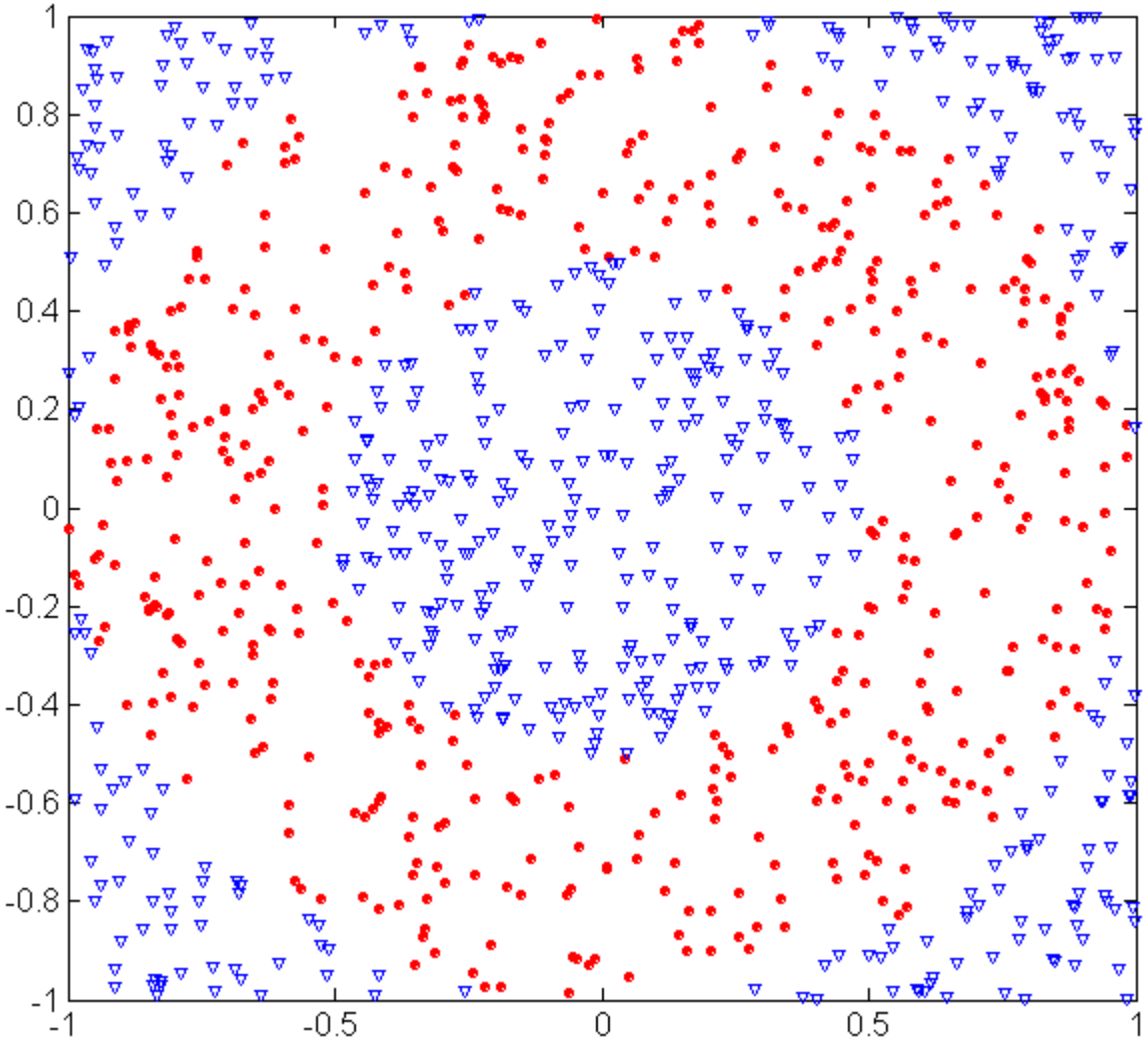
- Simple depth-first construction.
- Uses Information Gain
- Sorts Continuous Attributes at each node.
- Needs entire data to fit in memory.
- Unsuitable for Large Datasets.
  - Needs out-of-core sorting.

- You can download the software from: http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz

# 의사결정 트리의 주요 이슈

- Overfitting (과잉적합)
  - (일반적으로) 트리가 너무 크고 자세하게 형성되어, 오히려 정확도가 떨어지는 문제
  - 훈련집합에 과잉적합 생성되어, 테스트 집합이나 실제 분류되지 않은 데이터에 대해서는 오류가 오히려 더 커지는 문제
  - Cf. Underfitting(부족적합)
- Missing Values (누락 값)
  - 누락 값이 있는 경우, 부정확한 트리가 구축됨
- Costs of Classification (분류 비용)
  - 대용량 데이터 집합, 고차원 데이터, 복잡한 데이터의 경우, 분류에 많은 시간이 걸림
  - 정확도 높은 의사결정 트리를 **생성하기 위해 많은 비용이 요구됨**

# Underfitting and Overfitting (Example)




**500 circular and 500 triangular data points.**

**Circular points:**

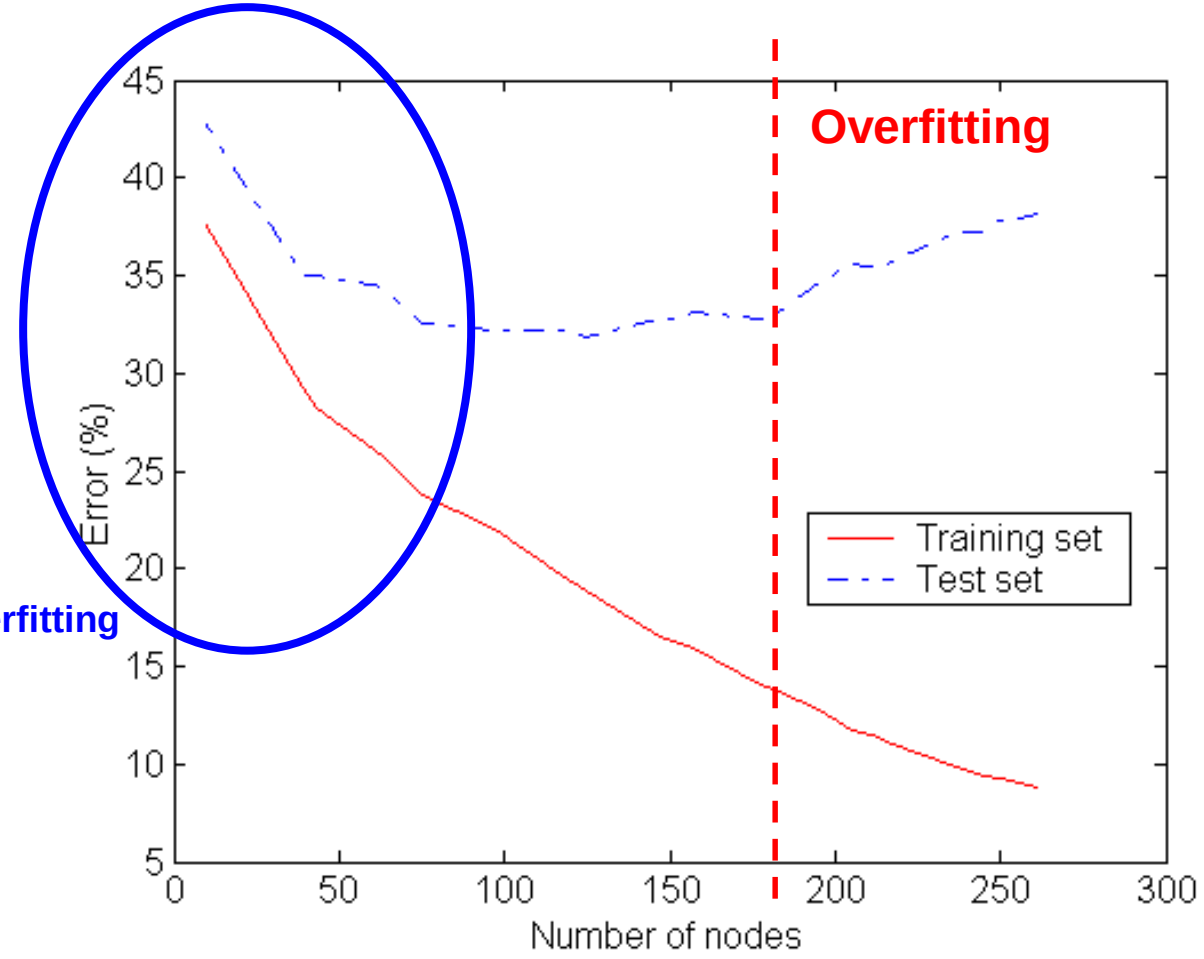
$0.5 \leq \text{sqrt}(x_1^2+x_2^2) \leq 1$

**Triangular points:**

$\text{sqrt}(x_1^2+x_2^2) > 0.5$ **or**

$\text{sqrt}(x_1^2+x_2^2) < 1$

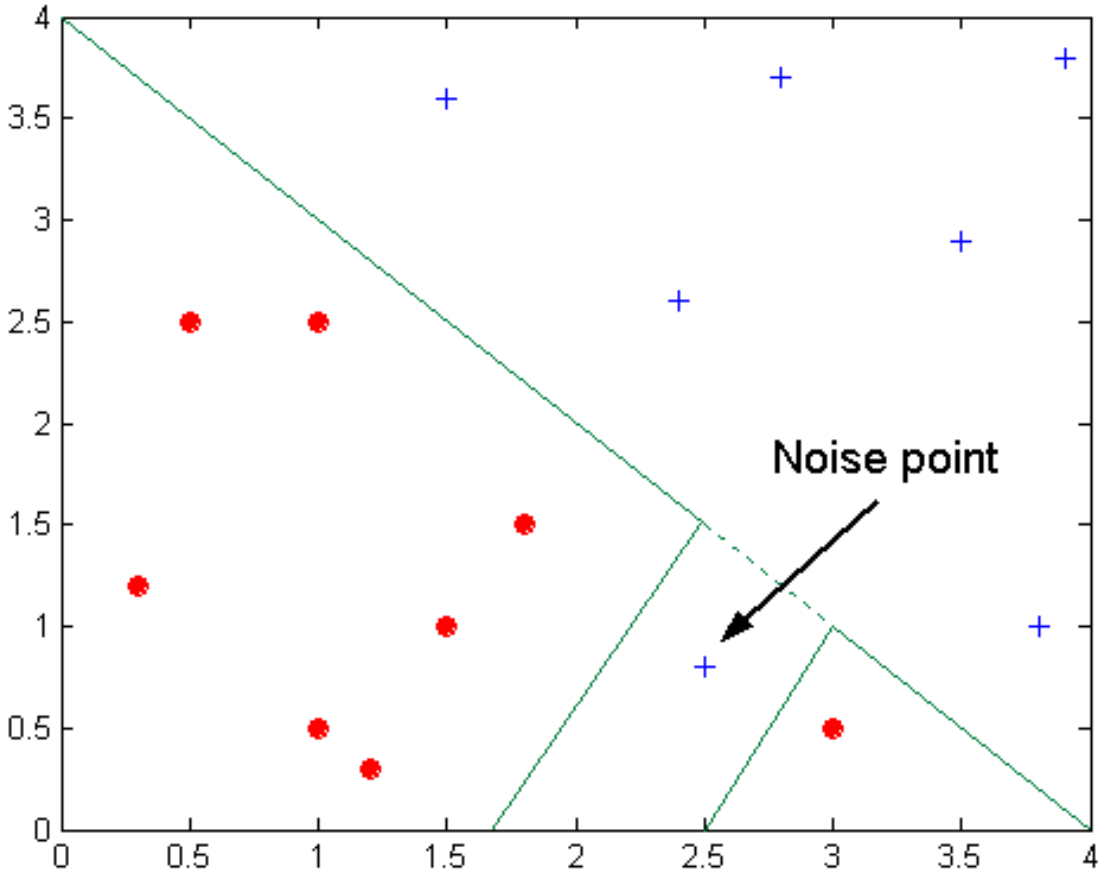
# Underfitting and Overfitting




- **Overfitting:**
- 트리에서 속성 값을 계속 분할할 수록 훈련집합에 대한 분류 에러는 줄어듦
- 하지만, 트리가 너무 세분화 되어 분할되었기 때문에, 시험 집합에 대해서는 어떤 시점 부터는 분류 에러가 증가함

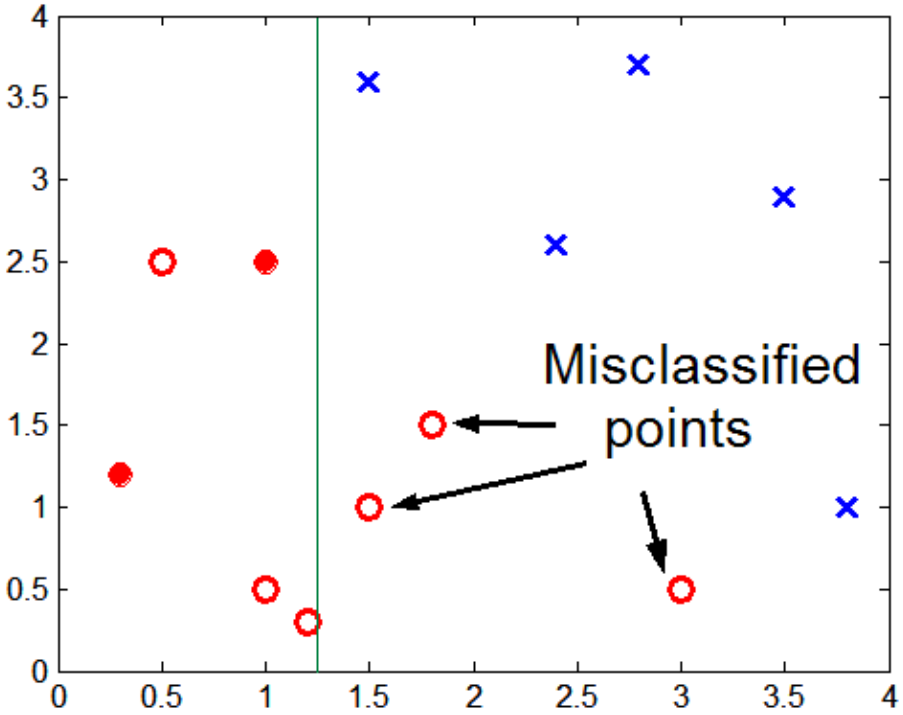
**Underfitting**: when model is too simple, both training and test errors are large

# 노이즈에 의한 과잉 적합




**Decision boundary is distorted by noise point**

# 데이터 부족으로 인한 과잉 적합




- **적은 수의 training data에 의해 과잉 적합되는 경우가 많음**
- Insufficient number of training records in the region causes the decision tree to predict the test examples using other training records that are irrelevant to the classification task (붉은색으로 채워진 데이터가 너무 부족(2개)하여, 붉은색 채워지지 않은 예측 데이터로 학습을 할 경우, 원래의 분류 선(녹색)과는 많이 다르게 됨)

# 다중 비교 절차가 필요한 경우의 과잉 적합

- Many algorithms employ the following greedy strategy:
  - Initial model: M
  - Alternative model: M' = M $\cup \gamma$, where $\gamma$ is a component to be added to the model (e.g., a test condition of a decision tree)
  - Keep M' if improvement, $\Delta(M,M') > \alpha$

초기 모델 M에서, 추가고려( $\gamma$)를 통해 이득이 있으면, 추가 고려사항이 포함된 대안모델을 사용할 수 있음

- Often times, $\gamma$ is chosen from a set of alternative components, $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_k\}$

이 때 다양한 추가고려 $\gamma$ 사항이 존재할 수 있음

- If many alternatives are available, one may inadvertently add irrelevant components to the model, resulting in model overfitting

즉, 많은 수의 대안이 있는 경우, 의도치않게 잘못된 선택을 할 수 있으며, 이는 overfitting을 유발 할 수도 있음

# 일반화 오류(Generalization error)에 대한 추정

- 일반화 오류를 최대한 줄여야 함. 하지만, 모델을 만들때는 training set만 사용할 수 있으므로, 일반화 오류를 간접적으로 추정해서 이를 줄여야 함

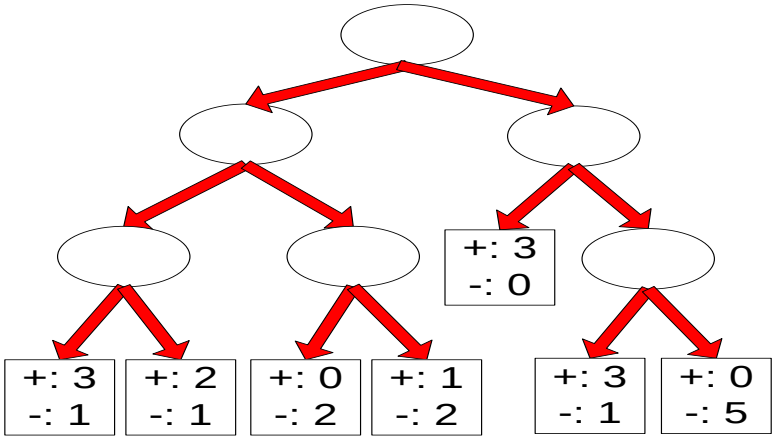
1. Re-substitution estimate (재치환 추정) 이용하기
   - 훈련집합이 전체 데이터를 잘 대표한다고 가정하여, 훈련 오류(즉, 재치환 오류)는 일반화 오류에 대한 추정치를 제공하는데 이용할 수 있다고 가정함
   - 이에, 의사결정 트리 귀납 알고리즘은 단순히 가장 낮은 훈련 오류율을 보이는 모델을 그 최종모델로 선택함 → 당연, 훈련 오류는 좋은 일반화 오류에 대한 추정이 아님 !
2. Model Complexity(모델 복잡도) 고려하기
   - 모델이 복잡해질 수록 overfitting 발생 가능성 높아짐. 이에, 모델의 복잡도를 고려하여 일반화 오류를 줄여야 함
3. Pessimistic Estimate(비관적 오류 추정)
   - 일반화 오류를 훈련오류와 모델 복잡도에 대한 penalty값의 합으로 봄(각 leaf node에 penalty 값 추가함)

# 일반화 오류(Generalization error)에 대한 추정

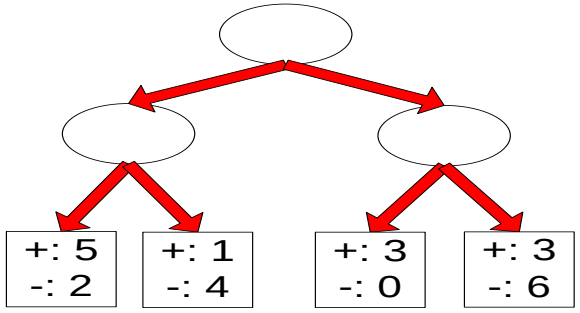
4. 최소 서술길이 원리(Minimum Description Length Principle)
   - :
5. Statistical Bounds(통계적 한계) 추정하기
   - 일반화 오류는 훈련 오류에 대한 통계적 보정(statistical correction)으로 추정될 수 있음
   - 일반화 오류는 훈련 오류보다 일반적으로 크므로, 훈련 오류의 상한으로 계산 추정하는 경우도 있음
6. Using Validation Set(검증 집합) 이용하기
   - 원래의 훈련 집합을 두 개의 작은 부분집합으로 나눔
   - 하나는 훈련, 다른 하나는 검증 집합으로 사용

# Resubstitution Estimate

- Using training error as an optimistic estimate of generalization error




Decision Tree, $T_L$




Decision Tree, $T_R$

**$e(T_L) = 4/24$**

**$e(T_R) = 6/24$**

**훈련오류율 e(TL)**

$$1 - \left( \frac{\max(1,3)}{24} + \frac{\max(2,1)}{24} + \frac{\max(2,0)}{24} + \frac{\max(1,2)}{24} + \frac{\max(3,0)}{24} + \frac{\max(3,1)}{24} + \frac{\max(0,5)}{24} \right)$$

$\frac{1}{6}$

**훈련오류율 e(TR)**

$$1 - \left( \frac{\max(5,2)}{24} + \frac{\max(1,4)}{24} + \frac{\max(3,0)}{24} + \frac{\max(3,6)}{24} \right)$$

$\frac{1}{4}$

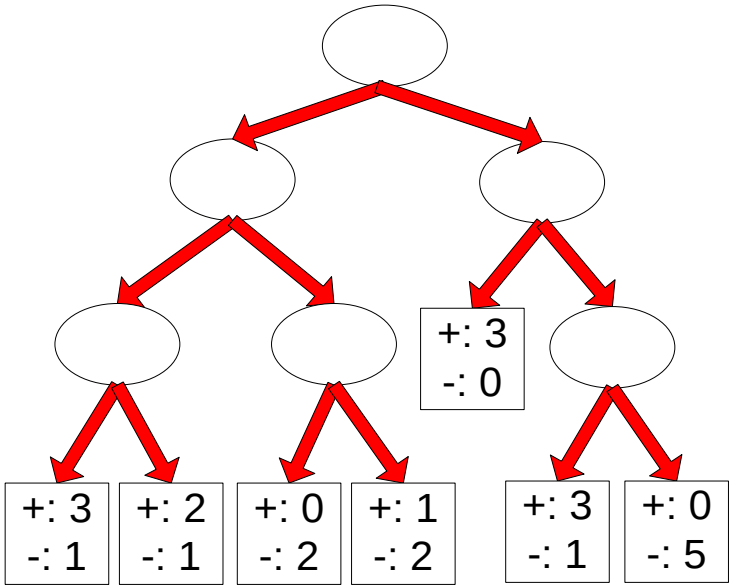
# Incorporating Model Complexity(모델 복잡성 고려)

- Rationale: Occam's Razor
  - Given two models of similar generalization errors, one should prefer the simpler model over the more complex model
  - A complex model has a greater chance of being fitted accidentally by errors in data
  - Therefore, one should include model complexity when evaluating a model

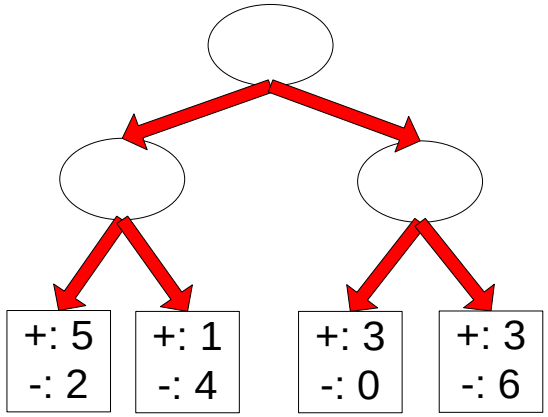
# Occam's Razor (오컴의 면도날)

- 정의: 같은 일반화 오류를 갖는 두개의 모델이 있을때, 더 단순한 모델이 복잡한 모델보다 선호된다.
  - (Given two models of similar generalization errors, one should prefer the simpler model over the more complex model)
  - 복잡한 모델에서는 데이터에존재하는 오류에 의해 적합해질 가능성이 더 커지기 때문

# Pessimistic Estimate




Decision Tree, $T_L$




Decision Tree, $T_R$

**$e(T_L) = 4/24$**

**$e(T_R) = 6/24$**

**$\Omega = 1$**

**$e'(T_L) = (4 + 7 \times 1)/24 = 0.458$ (7개 leaf node에 penalty값 각각 "1")**

**$e'(T_R) = (6 + 4 \times 1)/24 = 0.417$ (4개 leaf node에 penalty값 각각 "1")**

# Pessimistic Estimate

- 일반화 오류를 훈련오류와 모델 복잡도에 대한 penalty값의 합으로 봄(각 leaf node에 penalty 값 추가함)
- Given a decision tree node $t$
  - $n(t)$: number of training records classified by $t$
  - $e(t)$: misclassification error of node $t$
  - Training error of tree $T$:

$$e'(T) = \frac{\sum_i [e(t_i) + \Omega(t_i)]}{\sum_i n(t_i)} = \frac{e(T) + \Omega(T)}{N}$$

  - $\Omega$: is the cost of adding a node
  - N: total number of training records

# Minimum Description Length (최소 서술 길이)

- halting growth of the tree when the encoding is minimized

- C.f) Occam's razor
  - Most data mining tasks can be described as creating a model for the data
  - E.g.) the K-means models the data as a set of centroids.
  - Occam's razor: All other things being equal, the simplest model is the best.

# Minimum Description Length (최소 서술 길이)

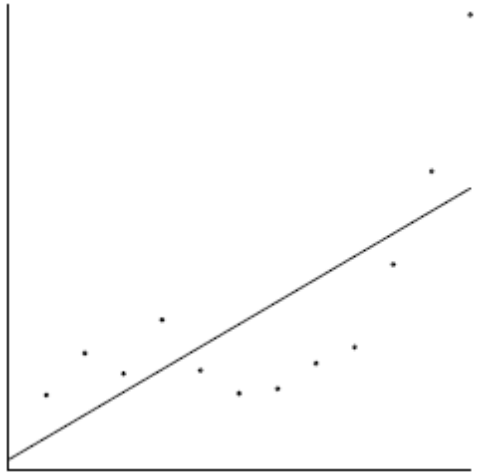
- Then, what is a simple model?
- Minimum Description Length Principle: Every model provides a encoding of our data. The model that gives the shortest encoding (best compression) of the data is the best.
  - MDL restricts the family of models considered
  - Encoding cost: cost of party A to transmit to party B the data.

# Minimum Description Length (최소 서술 길이)

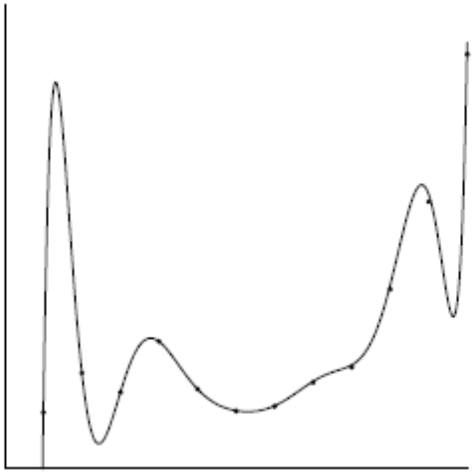
- The description length consists of two terms
  - The cost of describing the model (model cost)
  - The cost of describing the data given the model (data cost).
  - $L(D) = L(M) + L(D|M)$
- There is a tradeoff between the two costs
  - Very complex models describe the data in a lot of detail but are expensive to describe
  - Very simple models are cheap to describe but require a lot of work to describe the data given the model

# Minimum Description Length (최소 서술 길이)

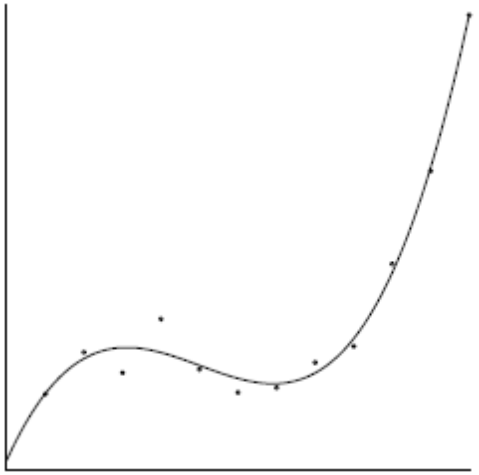
- Regression: find the polynomial for describing the data
  - Complexity of the model vs. Goodness of fit



**Low model cost**
**High data cost**



**High model cost**
**Low data cost**



**Low model cost**
**Low data cost**

**MDL avoids overfitting automatically!**

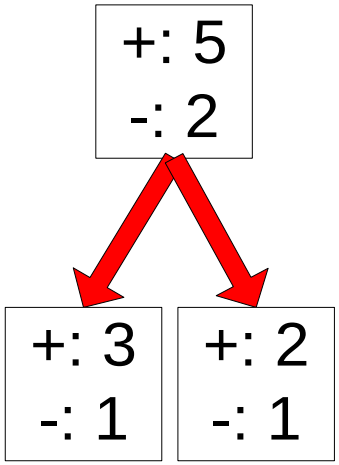
Source: Grnwald et al. (2005) *Advances in Minimum Description Length: Theory and Applications.*

# Estimating Statistical Bounds

- 일반화 오류는 훈련 오류에 대한 통계적 보정(statistical correction)으로 추정될 수 있음

$$e'(N,e,\alpha) = \frac{e + \frac{z^2_{\alpha/2}}{2N} + z_{\alpha/2}\sqrt{\frac{e(1-e)}{N} + \frac{z^2_{\alpha/2}}{4N^2}}}{1 + \frac{z^2_{\alpha/2}}{N}}$$

$\alpha$ : 신뢰 수준(confidence level)
$z_{\alpha/2}$ : 표준 정규 분포로부터 표준화된 값
$N$: e를 구하기 위해 사용되는 훈련 레코드의 총 수




**Before splitting:** e = 2/7, e'(7, 2/7, 0.25) = 0.503

e'(T) = 7 × 0.503 = 3.521

**After splitting:**

$e(T_L)$ = 1/4, e'(4, 1/4, 0.25) = 0.537

$e(T_R)$ = 1/3, e'(3, 1/3, 0.25) = 0.650

e'(T) = 4 × 0.537 + 3 × 0.650 = 4.098

# Using Validation Set (검증 집합 사용)

- 원래의 훈련 집합을 두 개의 작은 부분집합으로 나눔
- 하나는 훈련, 다른 하나는 검증 집합으로 사용
- Divide training data into two parts:
  - Training set:
    - use for model building
  - Validation set:
    - use for estimating generalization error
    - Note: validation set is not the same as test set

- Drawback:
  - Less data available for training

# Notes on Overfitting

- Overfitting은 의사결정 트리를 불필요하게 복잡하게 만드는 결과를 초래함
  - 훈련 오류(training error)의 최소화가 반드시 가장 좋은 의사결정 트리를 생성하는 것을 의미하지는 않음

- 그렇다면, overfitting을 줄이는 방법은?

# How to Address Overfitting

- Pre-Pruning,Early Stopping Rule (사전가지치기,조기정지규칙)
  - 전체 훈련 데이터에 완벽하게 맞는 완전히 성장한 트리가 만들어지기 전에 트리 성장 알고리즘을 정지함
  - 일반적인 정지 조건:
    - Stop if all instances belong to the same class
    - Stop if all the attribute values are the same
  - **더욱 엄격한 정지조건 사용:**
    - Stop if number of instances is less than some user-specified threshold
    - Stop if class distribution of instances are independent of the available features (e.g., using $\chi^2$ test)
    - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).
  - 이 방식(More restrictive condition)의 장점은 훈련 데이터에 지나치게 **과잉 적합하는 복잡한 subtree가 생성되지 않도록 한다는 것**

# How to Address Overfitting…

- Post-pruning(사후 가지치기)
  - 의사결정 트리는 처음에는 최대 크기로 성장함
  - 그 다음, 완전히 자란 트리를 상향식(bottom-up)으로 다듬어가는 가지치기 절차를 수행함
    - (1) 만약, trimming 후에 일반화 오류(generalization error)가 개선된다면, 해당 sub-tree를 leaf node로 교체함
    - (2) sub-tree의 다수 클래스(majority class)가 해당 class를 갖는 단일 leaf node로 대체됨
  - 사후 가지치기는 트리 성장 과정이 너무 이르게 종료될 수 있는 사전 가지치기와 달리, 완전히 성장한 트리를 기반으로 가지치기를 하므로, 사전 가지치기보다 더 나은 결과를 가지는 경향있음.
  - 하지만, 트리를 완전히 성장시키기 위해 요구되는 계산이 낭비적일 수 있음

# Example of Post-Pruning

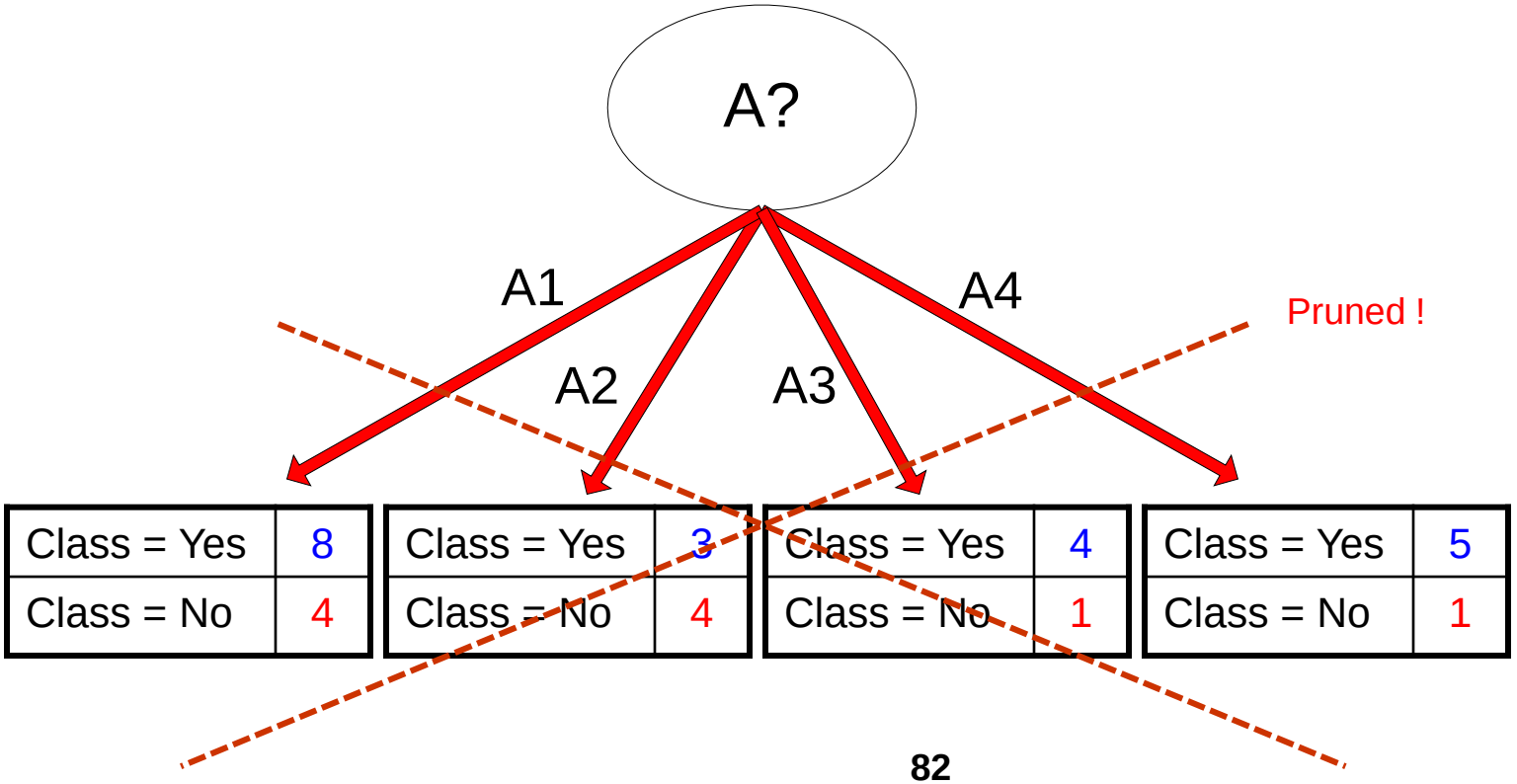
| | |
|---|---|
| Class = Yes | 20 |
| Class = No | 10 |
| Error = 10/30 | |

- **단일 노드인 경우:**

Training error: 1-max(20,10)/30 = 1-20/30 = 10/30
Pessimistic error = (10 + 노드 1개 * 0.5)/30 = 10.5/30

- **4개 노드 subset인 경우:**

1-max(8,4)/30 – max(3,4)/30 – max(4,1)/30 – max(5,1)/30 = 9/30
Pessimistic error = (9 + 노드 4개 * 0.5)/30 = 11/30
→Subset의 error가 하나의 노드의 error보다 큼 → Pruning 됨



# Handling Missing Attribute Values

- Missing values affect decision tree construction in three different ways:
  - Affects how impurity measures are computed
  - Affects how to distribute instance with missing value to child nodes
  - Affects how a test instance with missing value is classified

# Computing Impurity Measure

| Tid | Home owner | Marital Status | Taxable Income | Defaulted Borrower |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | ? | Single | 90K | Yes |

Missing value

**Before Splitting:**

**Entropy(Parent)**
**= -0.3 log(0.3)-(0.7)log(0.7) = 0.8813**

| | Class = Yes | Class = No |
|---|---|---|
| HomeOwner=Yes | 0 | 3 |
| HomeOwner =No | 2 | 4 |
| HomeOwner =? | 1 | 0 |

**Split on Home owner:**

**Entropy(Home=Yes) = 0**

**Entropy(Home=No)**
**= -(2/6)log(2/6) – (4/6)log(4/6) = 0.9183**

**Entropy(Children)**
**= 0.3 (0) + 0.6 (0.9183) = 0.551**

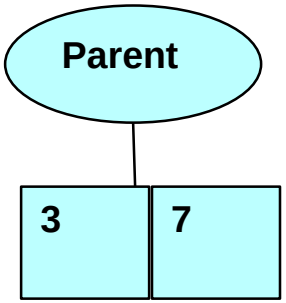
**Gain = (0.8813 – 0.551) = 0.3303**

# Computing Impurity Measure

**Before Splitting:**

**Entropy(Parent)**
**= -0.3 log(0.3)-(0.7)log(0.7) = 0.8813**

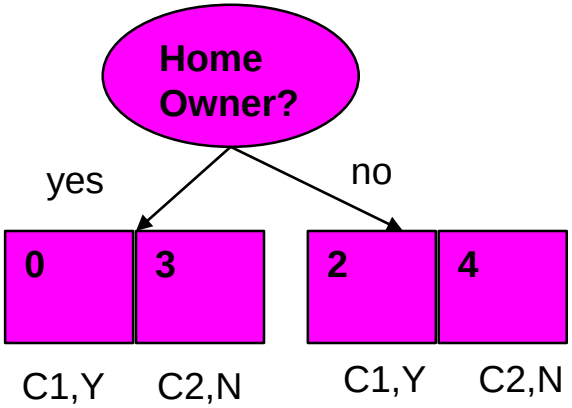
| | Class = Yes | Class = No |
|---|---|---|
| HomeOwner=Yes | 0 | 3 |
| HomeOwner =No | 2 | 4 |
| HomeOwner =? | 1 | 0 |




**Split on Home owner:**

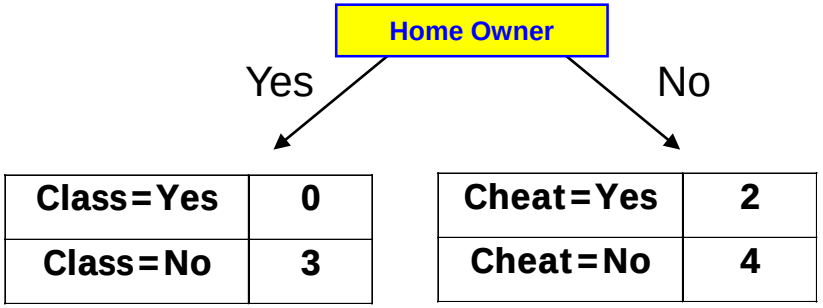
- **Entropy(Home=Yes)**
  P(C1)=0/3, P(C2)=3/3
  Entropy=-(0)log2(0) – (1)log2(1) = 0
- **Entropy(Home=No)**
  P(C1)=2/6, P(C2)=4/6
  Entropy=-(2/6)log2(2/6) – (4/6)log2(4/6)= 0.918
- **Entropy(Children)**
  = 0.3 (0) + 0.6 (0.9183) = 0.551

**Gain = (0.8813 – 0.551) = 0.3303**



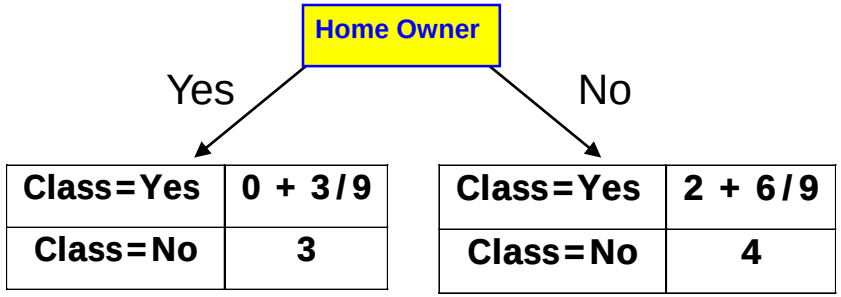
# Distribute Instances

| Tid | Home owner | Marital Status | Taxable Income | Defaulted Borrower |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | ? | Single | 90K | Yes |




| Tid | Home owner | Marital Status | Taxable Income | Defaulted Borrower |
|---|---|---|---|---|
| r10 | ? | Single | 90K | Yes |

Yes 혹은 No로 될 수 있음




**Probability that Home=Yes is 3/9**

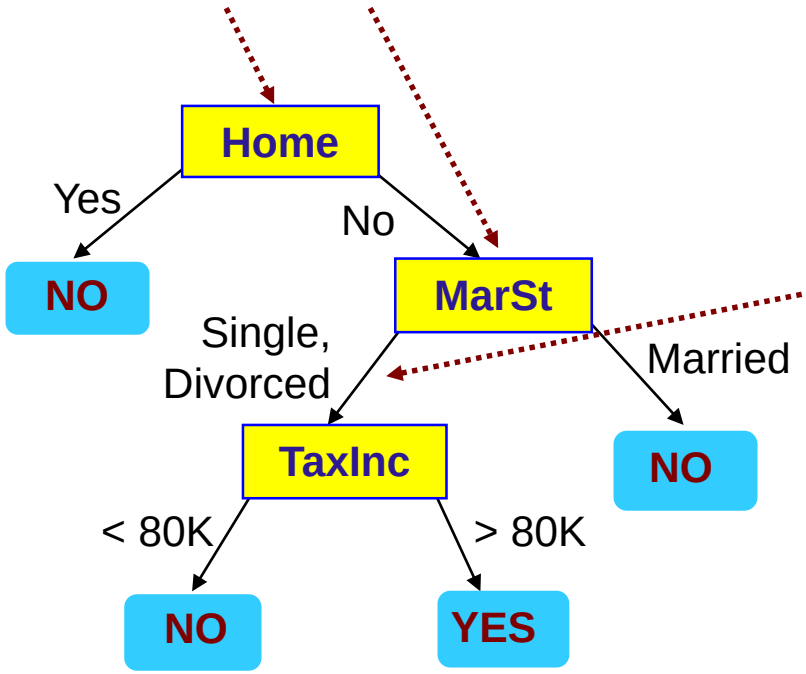
**Probability that Home=No is 6/9**

**Assign record to the left child with weight = 3/9 and to the right child with weight = 6/9**

# Classify Instances

**New record:**

| Tid | Home Owner | Marital Status | Taxable Income | Class |
|---|---|---|---|---|
| 11 | No | ? | 85K | ? |




| | Married | Single | Divorced | Total |
|---|---|---|---|---|
| Class=No | 3 | 1 | 0 | 4 |
| Class=Yes | 6/9 | 1 | 1 | 2.67 |
| Total | 3.67 | 2 | 1 | 6.67 |

**Probability that Marital Status = Married is 3.67/6.67**

**Probability that Marital Status ={Single,Divorced} is 3/6.67**

# Other Issues

- Data Fragmentation
- Search Strategy
- Expressiveness
- Tree Replication

# Data Fragmentation

- Data fragmentation problem:
  - As tree is developed, the questions are selected on the basis of less and less data
  - Number of records(data) gets smaller as you traverse down the tree

- Number of records(data) at the leaf nodes could be too small to make any statistically significant decision

- You can introduce a lower bound on the number of items per leaf node in stopping criterion

# Search Strategy

- Finding an optimal decision tree is NP-hard
- The algorithm presented so far uses a greedy, top-down, recursive partitioning strategy to induce a reasonable solution
- Other strategies?
  - Bottom-up
  - Bi-directional

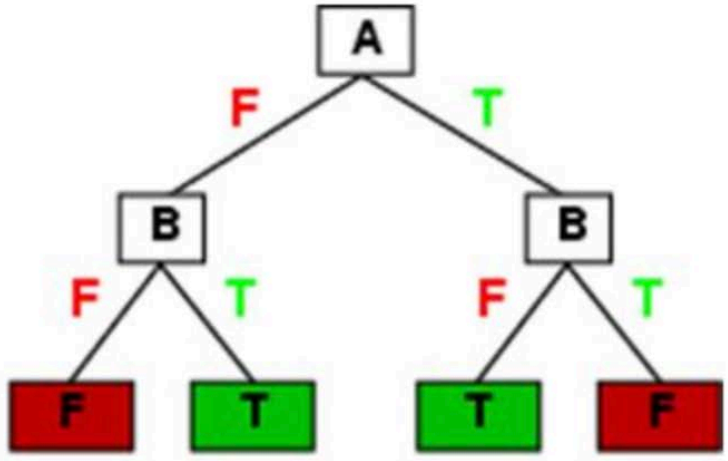
# Expressiveness (1/2)

- Decision tree provides expressive representation for learning discrete-valued function
  - But they do not generalize well to certain types of Boolean functions (ex) XOR or parity functions) → 정확한 모델링을 위해서 complete tree까지 만들어야 함
    - Example: parity function:
      - Class = 1 if there is an even number of Boolean attributes with truth value = True
      - Class = 0 if there is an odd number of Boolean attributes with truth value = True
    - For accurate modeling, must have a complete tree
- Not expressive enough for modeling continuous variables
  - Particularly when test condition involves only a single attribute at-a-time

# Expressiveness (2/2)

- Decision trees can express any function of the input attributes (eg., for Boolean functions, truth tables)

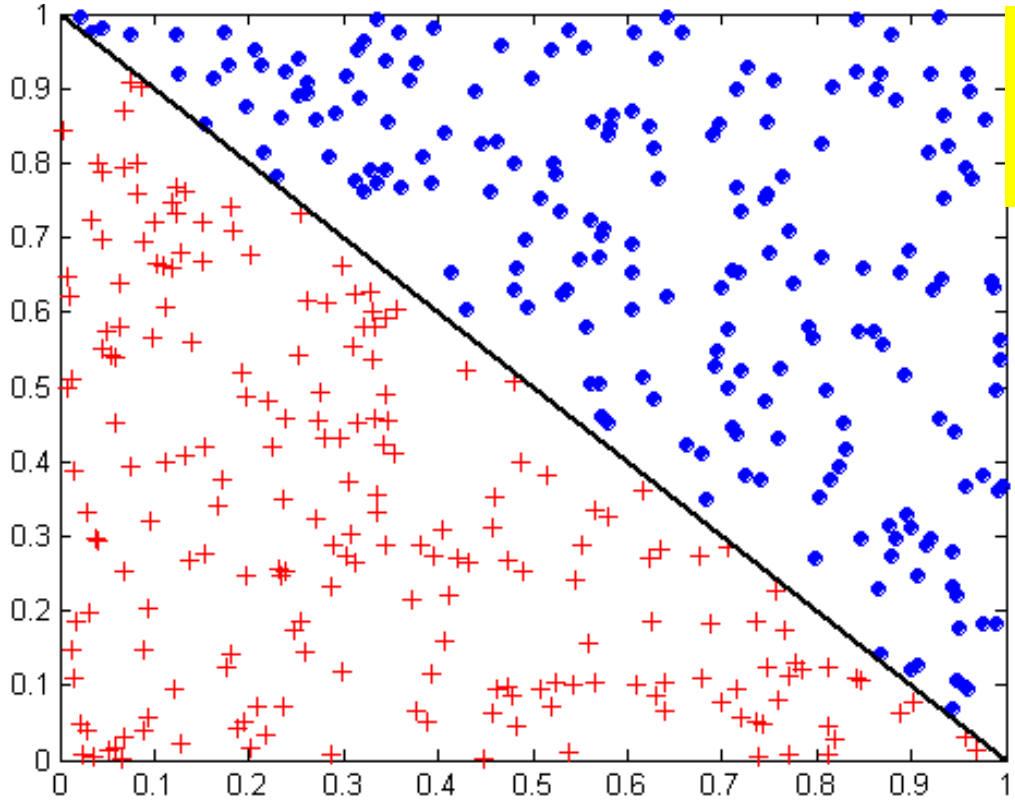
| A | B | A xor B |
|---|---|---|
| F | F | F |
| F | T | T |
| T | F | T |
| T | T | F |



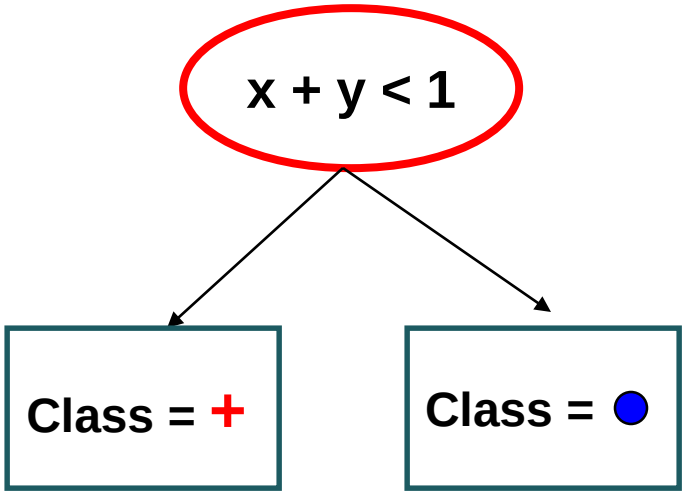

- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example but it probably won't generalize to new examples

즉, 어떤 training set이라도 여기에 일치하는 DT는 존재함 →
하지만, 우리는 DT가 complete DT(끝까지 성장하는 것보다)보다 compact DT를 선호하는 경우 많음

# Oblique Decision Trees (Oblique splits)



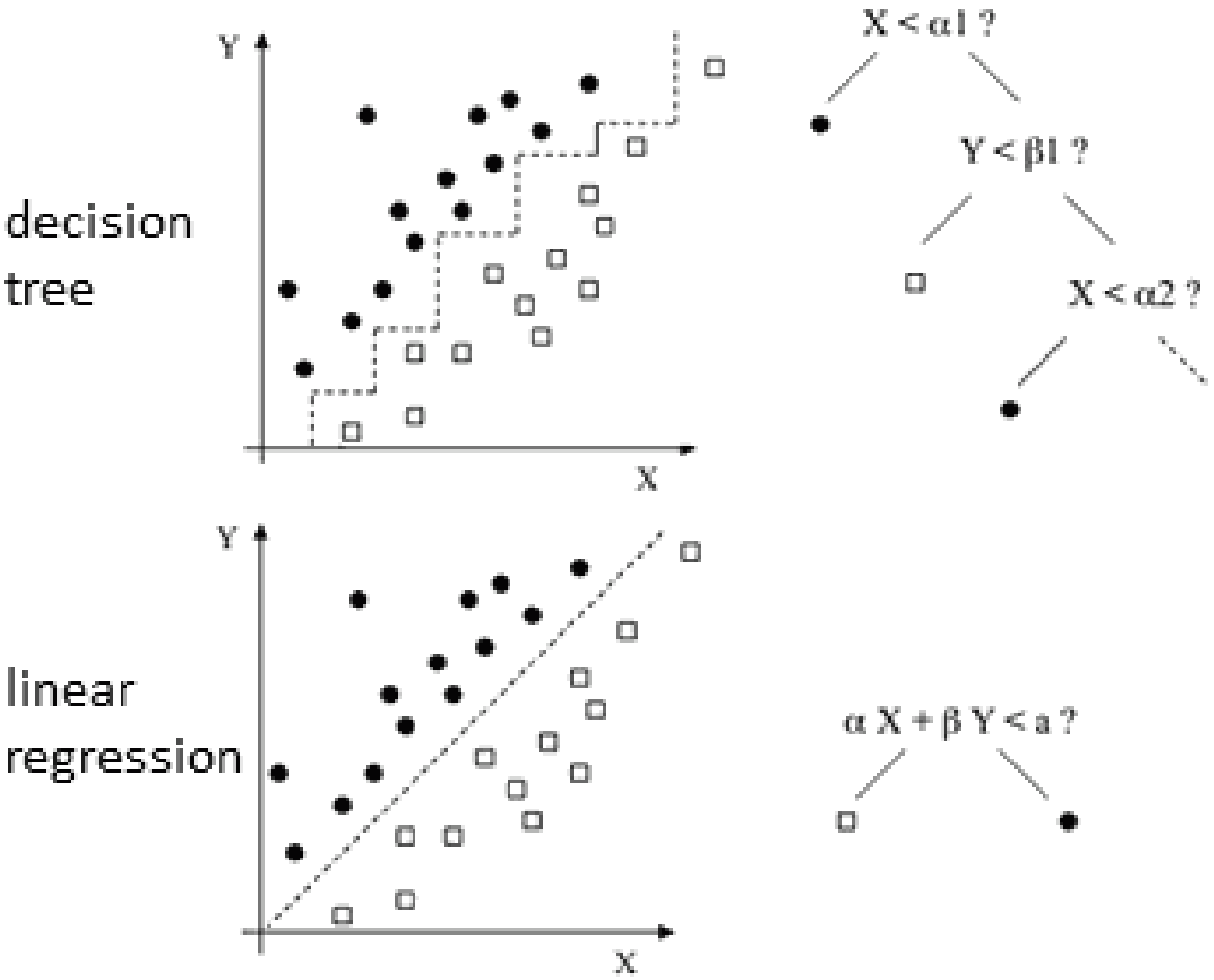

- 속성 축 값과 orthogonal하지 않은 "oblique(기울어진) split"도 가능함
- More expressive하고 compact한 DT도 가능함




- Test condition may involve multiple attributes
- More expressive representation
- Finding optimal test condition is computationally expensive

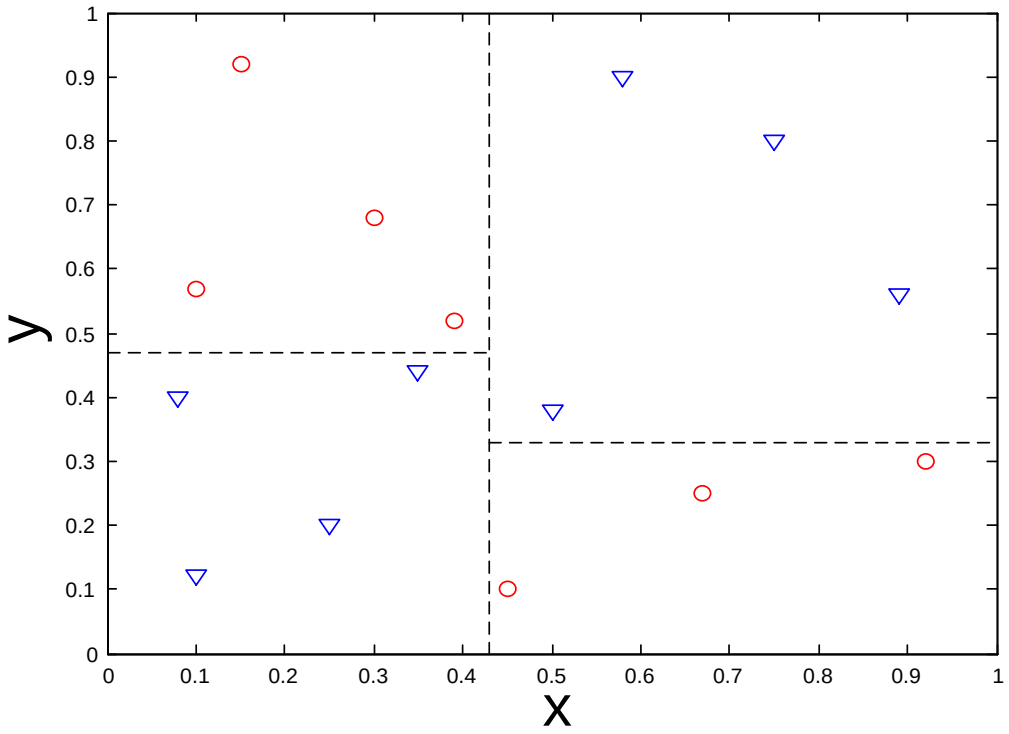
# Oblique Decision Trees (Oblique splits)

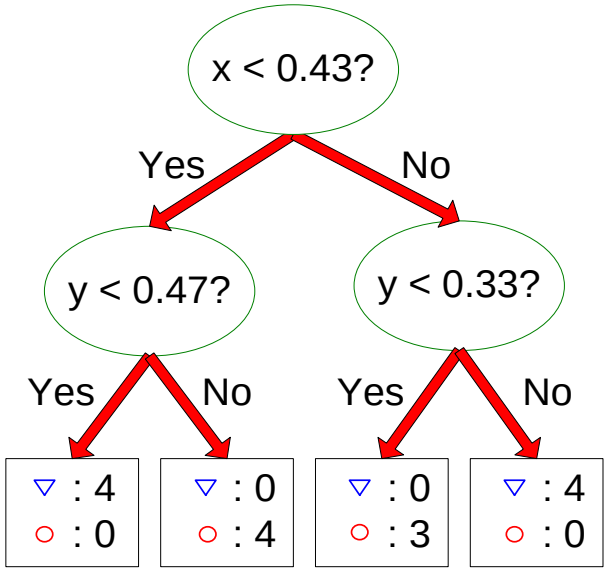
- DT에서도 oblique split이 가능하지만 제한이 많음
- 아래 예의 경우는 DT보다 Linear Regression 사용이 더 용이한 것을 보이고 있음



# Decision Boundary (1/2)

- Test 조건에 따라 Decision Boundary 정해짐

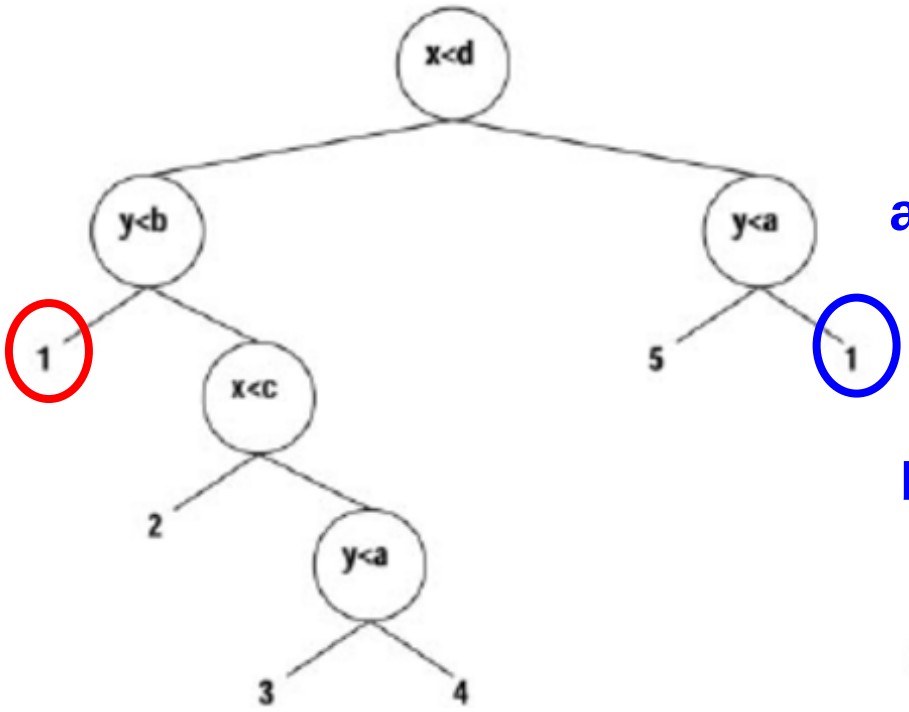


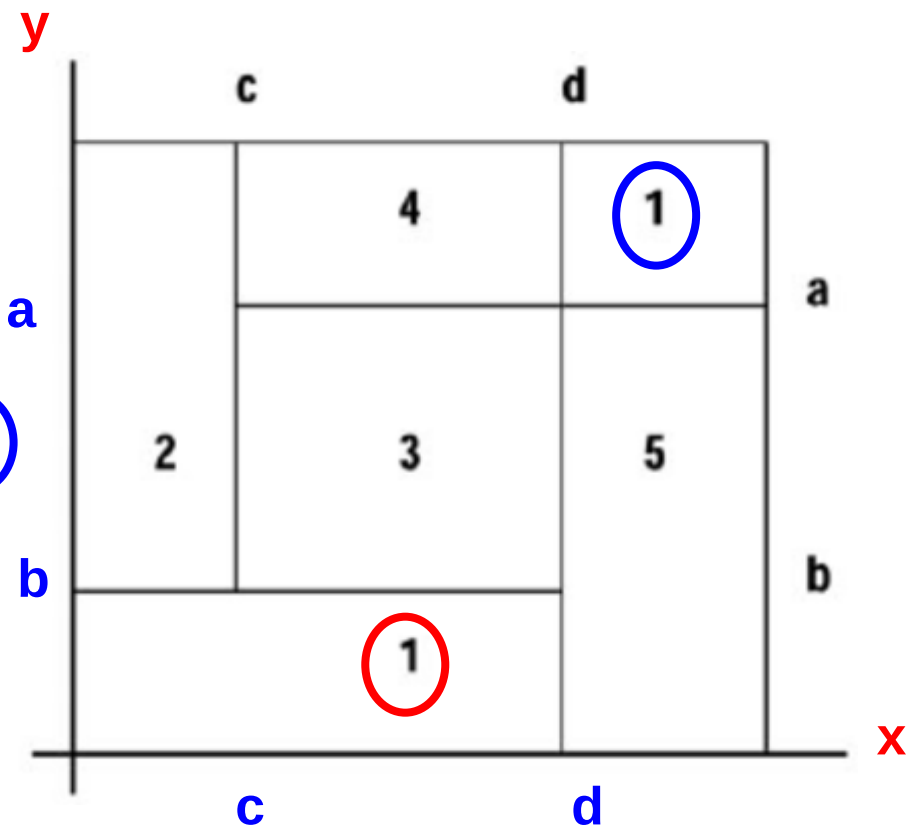




- **Border line between two neighboring regions of different classes is known as decision boundary**
- **Decision boundary is parallel to axes because test condition involves a single attribute at-a-time**

# Decision Boundary (2/2)

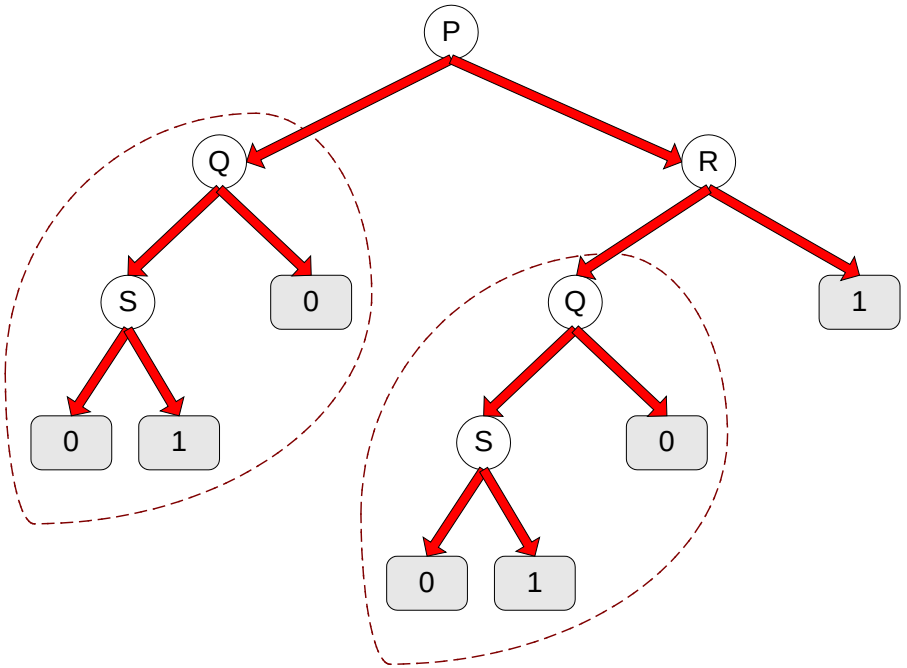
- Test 조건에 따라 Decision Boundary 정해짐
- Depth 4를 가지는 DT와 이에 의해 만들어진 Decision Boundary 사례



# Tree Replication

- 아래 그림처럼 subtree 가 DT에서 여러번 복제된 상태로 나올 수 있음
  - 이는 DT를 필요이상으로 복잡하게 만들고 해석도 어렵게 함
  - DT는 node의 single 속성 테스트 조건으로 만들어지므로 이러한 상황 발생 가능




- **Same subtree appears in multiple branches**

# Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

# Metrics for Performance Evaluation

- Focus on the predictive capability of a model
  - Rather than how fast it takes to classify or build models, scalability, etc.
- Confusion Matrix:

| | | PREDICTED CLASS | |
|---|---|---|---|
| | | Class=Yes | Class=No |
| ACTUAL CLASS | Class=Yes | **a** | **b** |
| | Class=No | **c** | **d** |

**a: TP (true positive)**

**b: FN (false negative)**

**c: FP (false positive)**

**d: TN (true negative)**

# Metrics for Performance Evaluation…

| | | PREDICTED CLASS | |
|---|---|---|---|
| | | Class=Yes | Class=No |
| ACTUAL CLASS | Class=Yes | a (TP) | b (FN) |
| | Class=No | c (FP) | d (TN) |

- Most widely-used metric:

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

# Limitation of Accuracy

- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10

- If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %
  - Accuracy is misleading because model does not detect any class 1 example

# Cost Matrix

| | PREDICTED CLASS | | |
|---|---|---|---|
| ACTUAL CLASS | C(**i**\|**j**) | **Class=Yes** | **Class=No** |
| | **Class=Yes** | C(Yes\|Yes) | C(No\|Yes) |
| | **Class=No** | C(Yes\|No) | C(No\|No) |

C(**i**|**j**) : **Class j를 class i로 잘못 예측한 코스트**

C(i|j): Cost of misclassifying class j example as class i

# Cost vs Accuracy

| Count | PREDICTED CLASS | | |
|---|---|---|---|
| ACTUAL CLASS | | Class=Yes | Class=No |
| | Class=Yes | a | b |
| | Class=No | c | d |

| Cost | PREDICTED CLASS | | |
|---|---|---|---|
| ACTUAL CLASS | | Class=Yes | Class=No |
| | Class=Yes | p | q |
| | Class=No | q | p |

Accuracy is proportional to cost if
1. C(Yes|No)=C(No|Yes) = q
2. C(Yes|Yes)=C(No|No) = p

$N = a + b + c + d$

$\text{Accuracy} = (a + d)/N$

$\text{Cost} = p\,(a + d) + q\,(b + c)$

$= p\,(a + d) + q\,(N - a - d)$

$= q\,N - (q - p)(a + d)$

$= N\,[q - (q-p) \times \text{Accuracy}]$

# Computing Cost of Classification

| Cost Matrix | PREDICTED CLASS | | |
|---|---|---|---|
| | C(i\|j) | + | - |
| ACTUAL CLASS | + | -1 | 100 |
| | - | 1 | 0 |

| Model $M_1$ | PREDICTED CLASS | | |
|---|---|---|---|
| | | + | - |
| ACTUAL CLASS | + | 150 | 40 |
| | - | 60 | 250 |

Accuracy = 80%

Cost = 3910

| Model $M_2$ | PREDICTED CLASS | | |
|---|---|---|---|
| | | + | - |
| ACTUAL CLASS | + | 250 | 45 |
| | - | 5 | 200 |

Accuracy = 90%

Cost = 4255

# Cost-Sensitive Measures

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

| | | PREDICTED CLASS | |
|---|---|---|---|
| | | Class=Yes | Class=No |
| ACTUAL CLASS | Class=Yes | **True Positive** | **False Negative** |
| | Class=No | **False Positive** | **True Negative** |

- **Precision(정밀도) 혹은 positive predictive value:**
  - **탐지했다고 주장하는 것(positive)이 그게 맞는 경우임**
- **Recall(재현율) 혹은 민감도(sensitivity)**
  - **실제 정답 중에서 제대로 이를 찾아내는 경우임**
- **F-Measure**
  - **정밀도와 재현율의 조화 평균**

- 예) 만약 10,000개의 패킷 중에서 실제 악성 패킷의 빈도가 매우 적은 경우:

- Precision(정밀도)는 탐지했다고 주장한 것(positive) 중에서 그게 정확한 경우임.
- Recall(재현율)은 실제 정답중에서 이를 제대로 찾아내는 경우임.
- 이에 만약, 실제 악성 공격 패킷이 매우 적은 경우에는 (실제 공격을 찾아내는 것이 중요한데, 즉 높은 재현율), 무조건 찾았다..라고 남발할 경우(즉, fp기 많아질 경우), 재현율은 좋아지지만, 정밀도는 떨어짐. 이에, 이 둘간의 조화평균이 중요해짐(F-Measure)

# Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?

- Methods for Performance Evaluation
  - 시험 데이터를 사용한 성능 평가
  - How to obtain reliable estimates?

- Methods for Model Comparison
  - How to compare the relative performance among competing models?

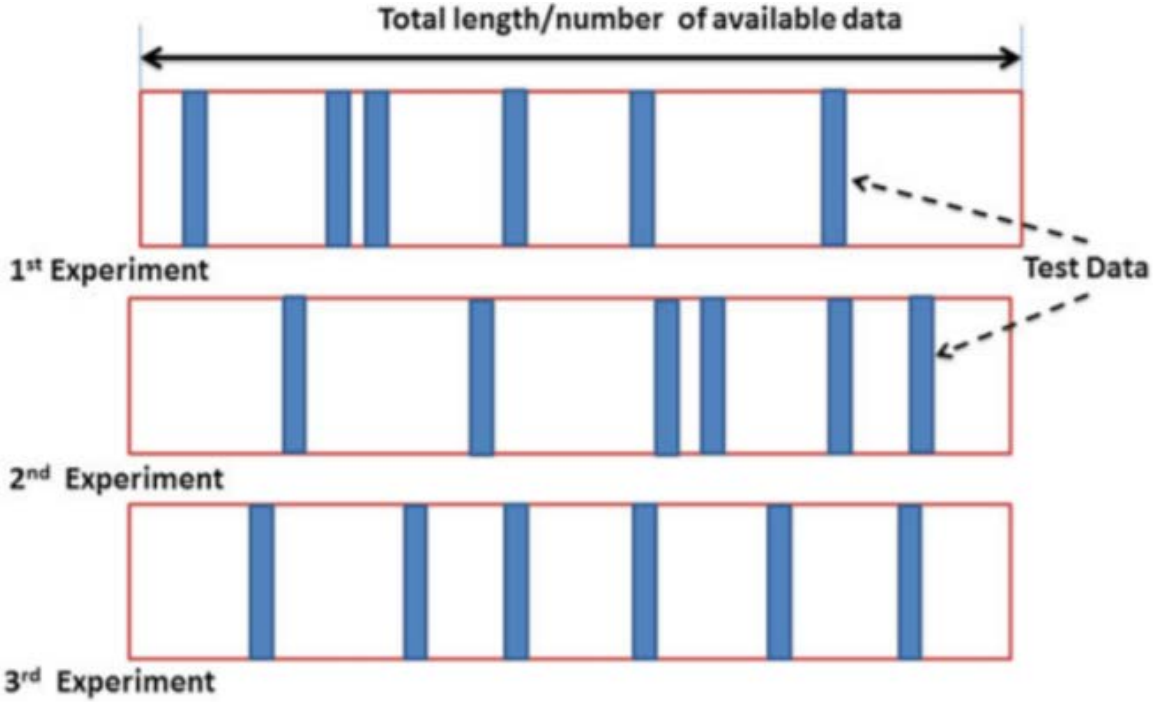
# Methods for Performance Evaluation

- 시험 데이터를 사용한 성능 평가
- How to obtain a reliable estimate of performance?

- Performance of a model may depend on other factors besides the learning algorithm:
  - Class distribution
  - Cost of misclassification
  - Size of training and test sets

# Methods of Estimation (1/4: 분류기 성능 평가 방법)

- Holdout (예비기법)
  - 예) 훈련 데이터(2/3), 시험 데이터(1/3)로 구성
  - 훈련 데이터 부족시 model이 부실해짐
  - 훈련 데이터 집합과 시험 데이터 집합 구성에 의존적임
- Random subsampling (랜덤 서브샘플링)
  - Holdout(예비기법)을 모델의 성능 향상을 위해 여러 번 반복하는 것
  - 이때, 시험 데이터가 무작위로 선택됨



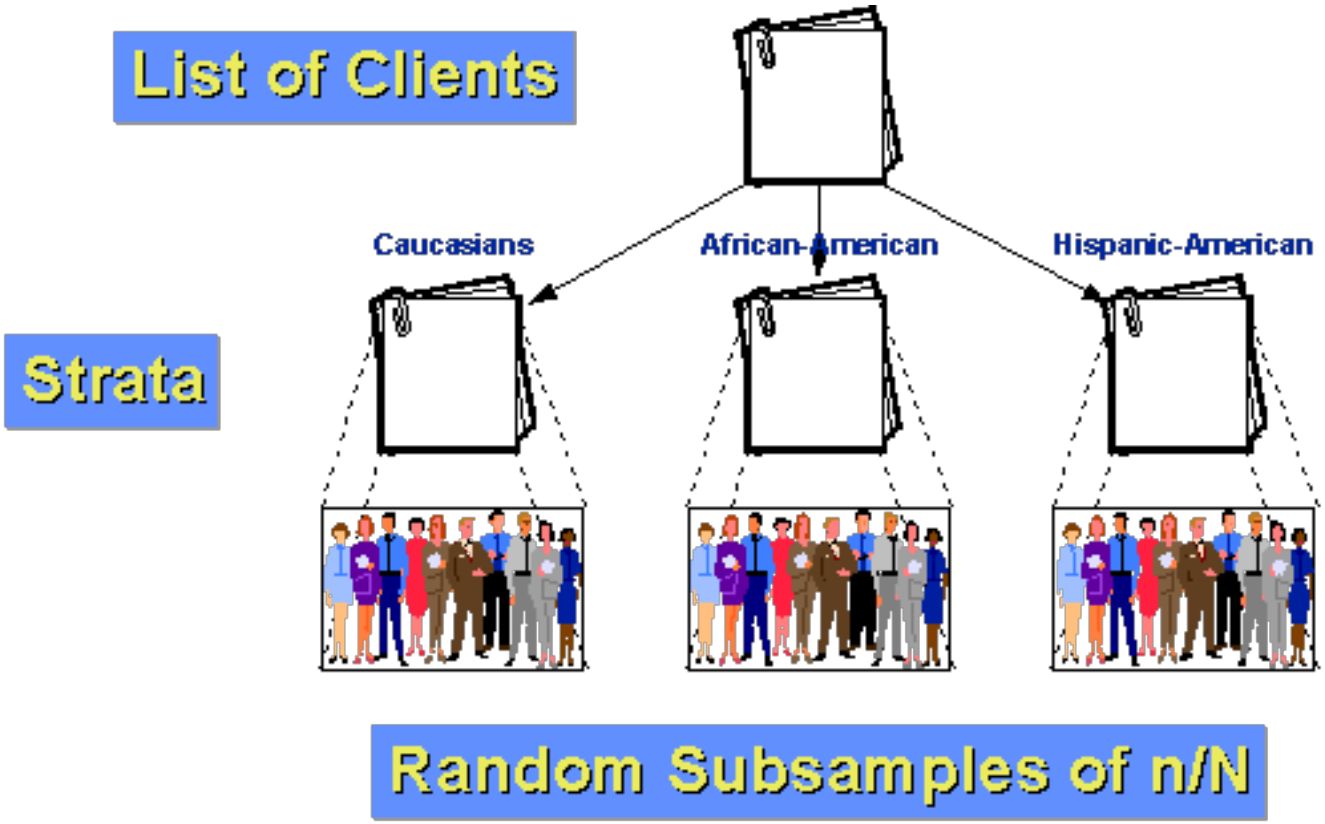
# Methods of Estimation (2/4: 분류기 성능 평가 방법)

- Cross validation (교차 검증)
  - Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Leave-one-out: k=n
  - 예) 데이터를 동일한 크기로 5개로 나눈 후, 한번에 정확히 하나의 데이터만 사용하여, 시험함. 총 5번 시험가능

만약 데이터를 훈련용 시험용 2개로 나누는 경우, 이중교차검증(two-fold cross validation)이라고 함

# Methods of Estimation (3/4: 분류기 성능 평가 방법)

- Stratified sampling (층화 추출)
  - 데이터를 어떤 기준으로 그룹을 만듦(Strata)
  - 각 Strata에서 subsample이 선택되어 시험에 사용됨



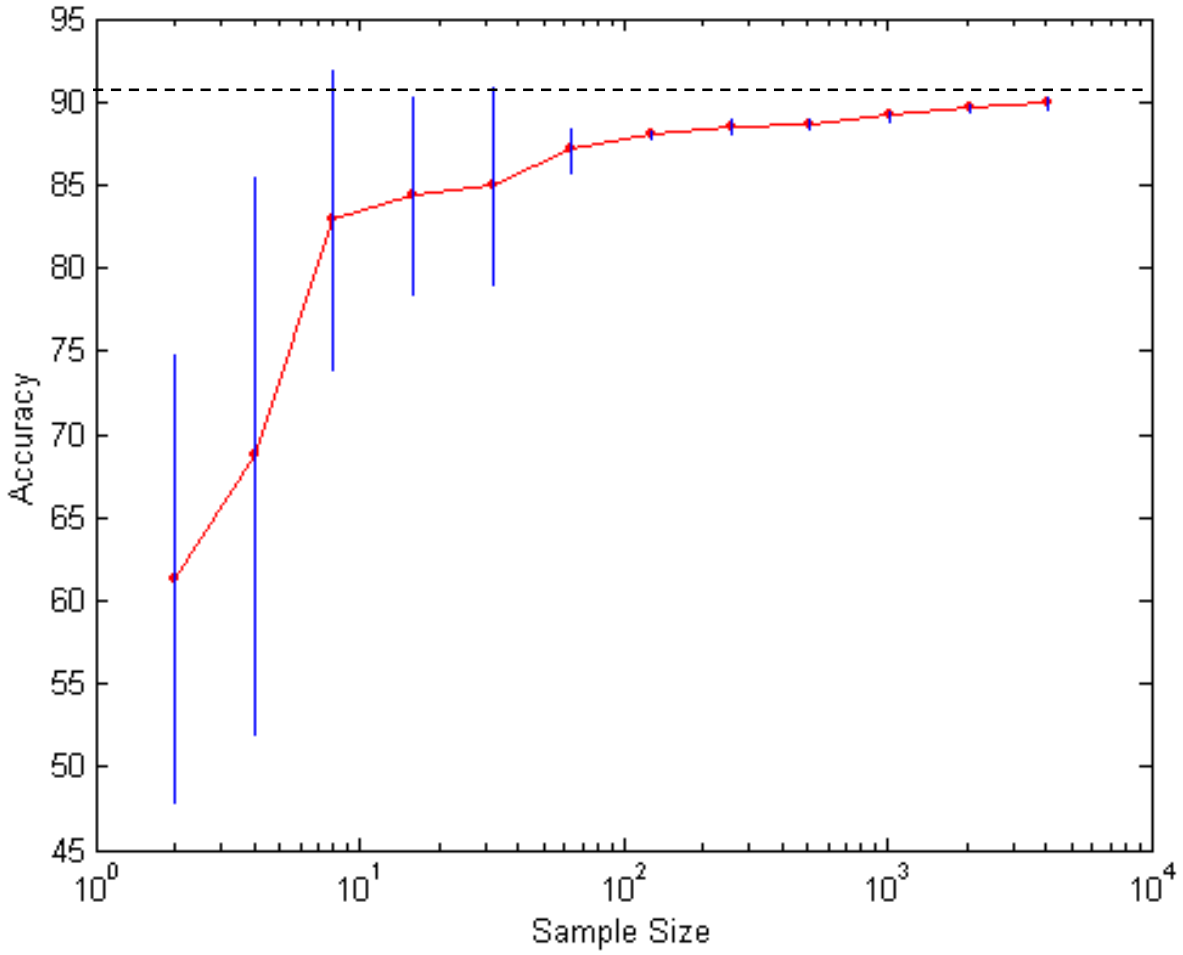
# Methods of Estimation (4/4: 분류기 성능 평가 방법)

- Bootstrap (부트 스트랩)
  - Sampling with replacement (이전에 사용된 데이터가 다시 반환되므로 다시 샘플링 되어 사용될 수 있음)

# Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

# Learning Curve (1/2)




- Learning curve shows how accuracy changes with varying sample size
- Requires a sampling schedule for creating learning curve:
  - Arithmetic sampling (Langley, et al)
  - Geometric sampling (Provost et al)

Effect of small sample size:

- Bias in the estimate
- Variance of estimate

# Learning Curve (2/2)

- Arithmetic sampling
  - 샘플 크기가 단순한 산술 수식으로 표시됨
  - 예: arithmetic sampling with the following equation: Si = S0 + I × C
  - 여기서, S0 is the initial sample size and C is a constant.
  - S0, S0 + C, S0 + 2C, S0 + 3C…, 만약S0 = 1,000 이고 C = 100 이라면, S1 = 1,100, S2 = 1,200, … 이 됨
- Geometric sampling
  - Sample size is increased geometrically so that sample sizes are in geometrical progression
  - 예: Si = S0 × C^I
  - 예: S0, S0·C, S0·C^2 , S0·C^3
  - S0 = 1,000이고 C = 2 이라면, S1 = 2,000, S2 = 4,000, S3=8,000, ...

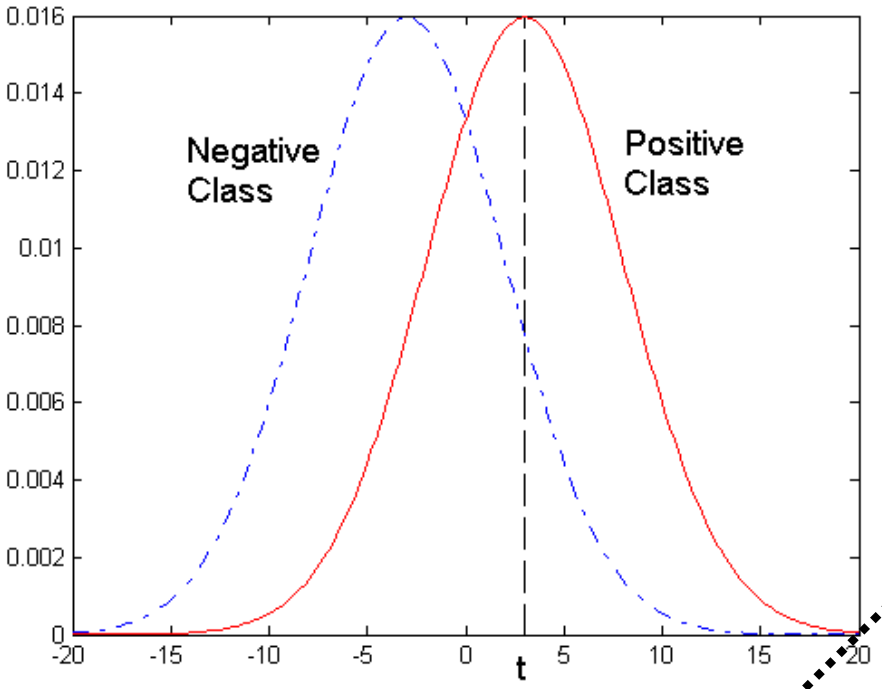
# ROC (Receiver Operating Characteristic)

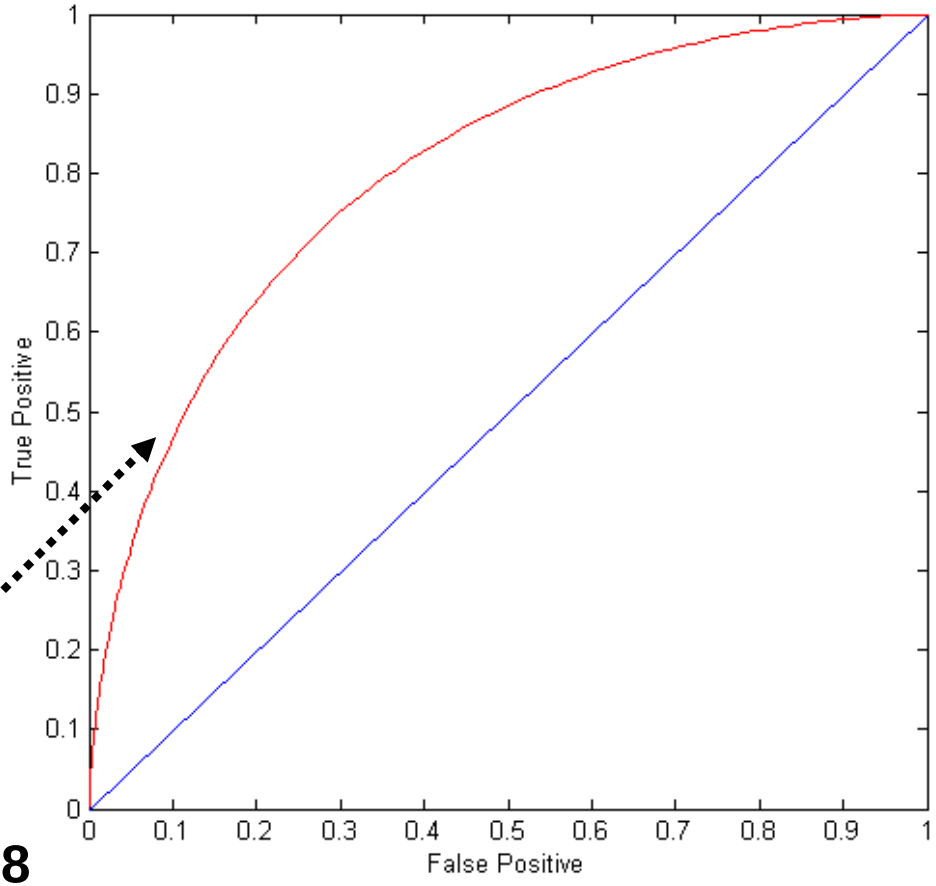
- Developed in 1950s for signal detection theory to analyze noisy signals
  - Characterize the trade-off between positive hits and false alarms
- ROC curve plots TP (on the y-axis) against FP (on the x-axis)
- Performance of each classifier represented as a point on the ROC curve
  - changing the threshold of algorithm, sample distribution or cost matrix changes the location of the point

# ROC Curve

- 1-dimensional data set containing 2 classes (positive and negative)

- any points located at x > t is classified as positive

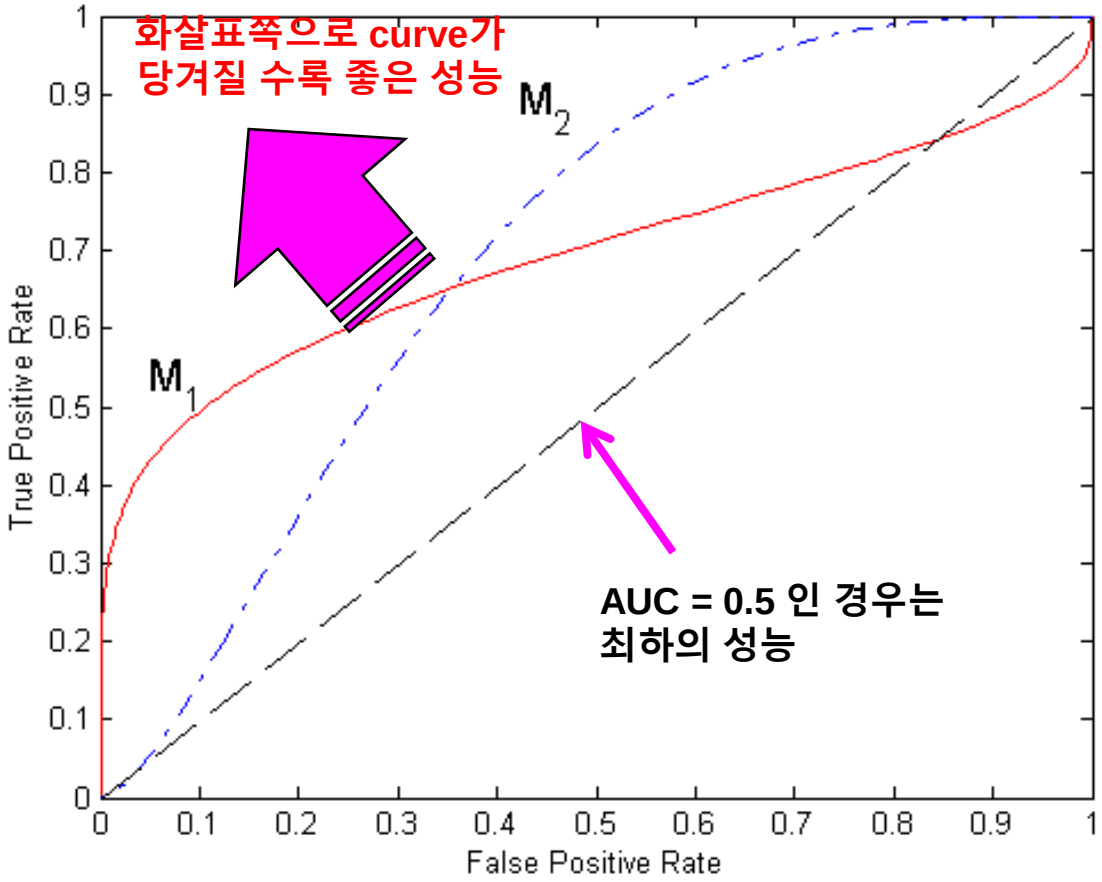



**At threshold t:**

**TP=0.5, FN=0.5, FP=0.12, FN=0.88**

# Using ROC for Model Comparison



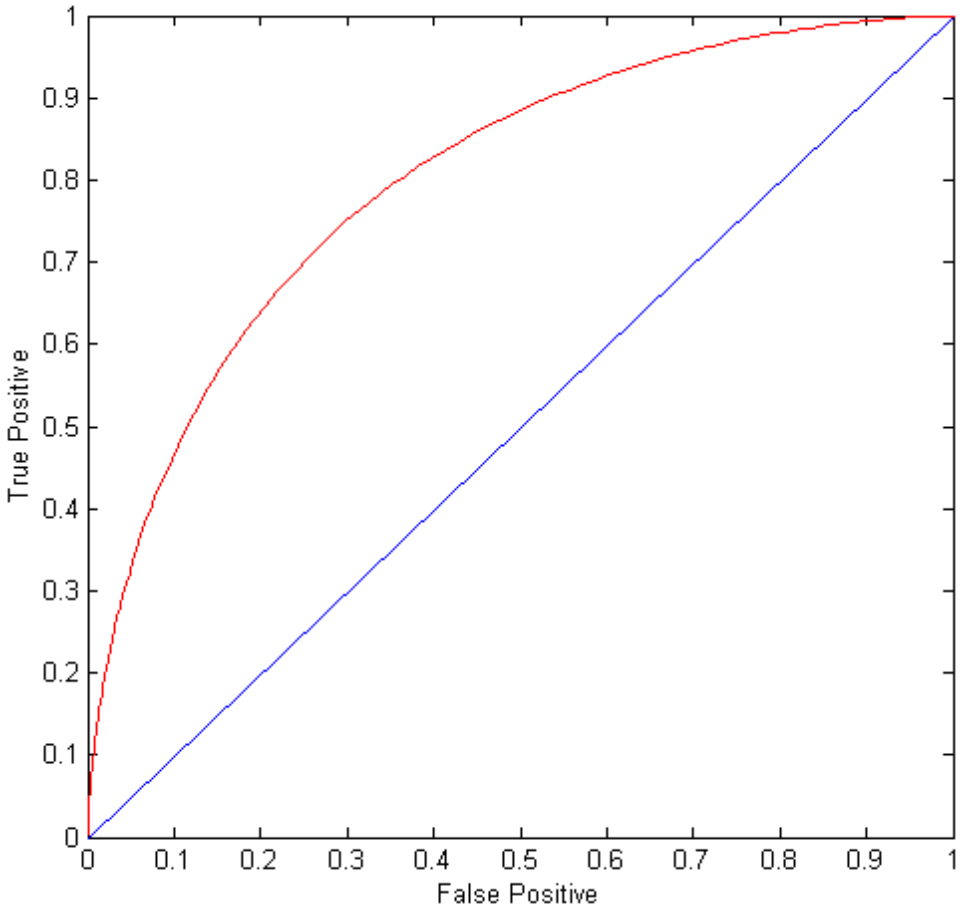

- No model consistently outperform the other
  - $M_1$ is better for small FPR
  - $M_2$ is better for large FPR
- Area Under the ROC curve (AUC)
  - Ideal:
    - Area = 1
  - Random guess:
    - Area = 0.5

# ROC Curve

(TP,FP):

- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (1,0): ideal

- Diagonal line:
  - Random guessing
  - Below diagonal line:
    - prediction is opposite of the true class



# How to Construct an ROC curve

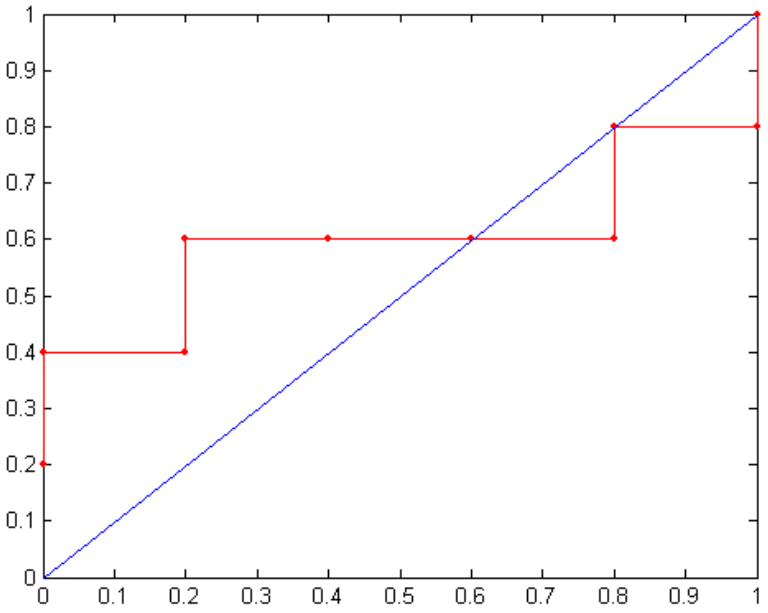
| Instance | P(+\|A) | True Class |
|---|---|---|
| 1 | 0.95 | + |
| 2 | 0.93 | + |
| 3 | 0.87 | - |
| 4 | 0.85 | - |
| 5 | 0.85 | - |
| 6 | 0.85 | + |
| 7 | 0.76 | - |
| 8 | 0.53 | + |
| 9 | 0.43 | - |
| 10 | 0.25 | + |

- Use classifier that produces posterior probability for each test instance P(+|A)
- Sort the instances according to P(+|A) in decreasing order
- Apply threshold at each unique value of P(+|A)
- Count the number of TP, FP, TN, FN at each threshold
- TP rate, TPR = TP/(TP+FN)
- FP rate, FPR = FP/(FP + TN)

# How to construct an ROC curve

| | Class | + | - | + | - | - | - | + | - | + | + | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Threshold >= | | 0.25 | 0.43 | 0.53 | 0.76 | 0.85 | 0.85 | 0.85 | 0.87 | 0.93 | 0.95 | 1.00 |
| | TP | 5 | 4 | 4 | 3 | 3 | 3 | 3 | 2 | 2 | 1 | 0 |
| | FP | 5 | 5 | 4 | 4 | 3 | 2 | 1 | 1 | 0 | 0 | 0 |
| | TN | 0 | 0 | 1 | 1 | 2 | 3 | 4 | 4 | 5 | 5 | 5 |
| | FN | 0 | 1 | 1 | 2 | 2 | 2 | 2 | 3 | 3 | 4 | 5 |
| → | TPR | 1 | 0.8 | 0.8 | 0.6 | 0.6 | 0.6 | 0.6 | 0.4 | 0.4 | 0.2 | 0 |
| → | FPR | 1 | 1 | 0.8 | 0.8 | 0.6 | 0.4 | 0.2 | 0.2 | 0 | 0 | 0 |

**ROC Curve:**



# Test of Significance (유의성 테스트)

- 데이터 크기에 따라 두개의 분류기 모델에서 관찰된 정확도는 의미가 없을 수도 있음
- Given two models:
  - Model M1: accuracy = 85%, tested on 30 instances
  - Model M2: accuracy = 75%, tested on 5000 instances
- Can we say M1 is better than M2?
  - M1이 M2보다 더 높은 정확도를 갖지만, 더 작은 시험 집합에 대해 시험됨. → M1의 정확도를 얼마나 신뢰할 수 있나?

# Confidence Interval for Accuracy

- Prediction can be regarded as a Bernoulli trial
  - A Bernoulli trial has 2 possible outcomes
  - Possible outcomes for prediction: correct or wrong
  - Collection of Bernoulli trials has a Binomial distribution:
    - $x \sim Bin(N, p)$ x: number of correct predictions
    - e.g: Toss a fair coin 50 times, how many heads would turn up?
      Expected number of heads = $N \times p = 50 \times 0.5 = 25$

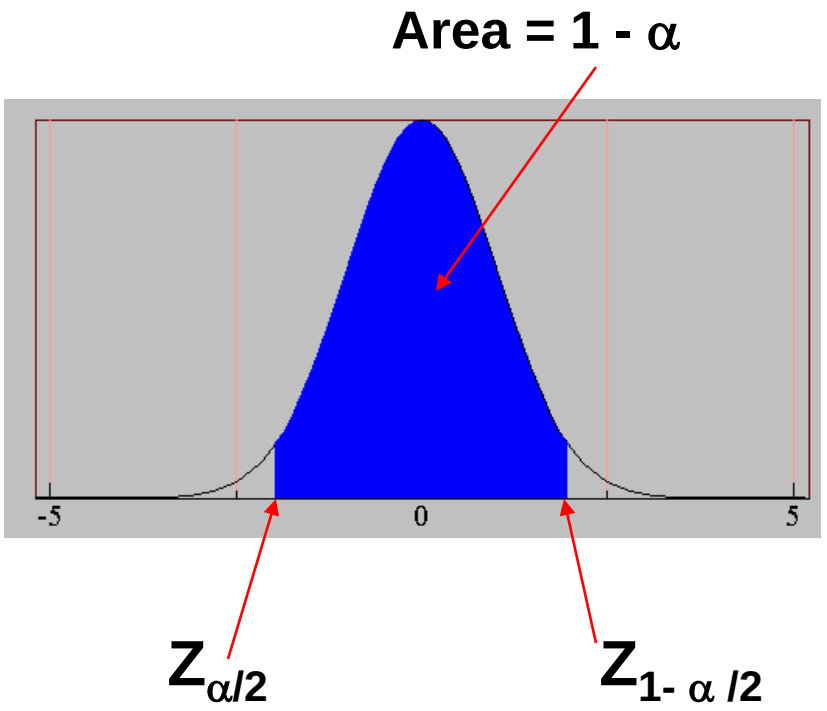
- Given x (# of correct predictions) and N (total # of test instances) → 실험적 정확도 **acc=x/N**

Can we predict p (true accuracy of model)?

# Confidence Interval for Accuracy

- For large test sets (N > 30),
  - **acc** has a normal distribution with mean p and variance p(1-p)/N

$$P(Z_{\alpha/2} < \frac{acc - p}{\sqrt{p(1-p)/N}} < Z_{1-\alpha/2}) = 1-\alpha$$




- Confidence Interval for p:

$$p = \frac{2 \times N \times acc + Z^2_{\alpha/2} \pm \sqrt{Z^2_{\alpha/2} + 4 \times N \times acc - 4 \times N \times acc^2}}{2(N + Z^2_{\alpha/2})}$$

# Confidence Interval for Accuracy

- Consider a model that produces an accuracy of 80% when evaluated on 100 test instances:
  - N=100, acc = 0.8
  - Let 1-$\alpha$ = 0.95 (95% confidence)
  - From probability table, $Z_{\alpha/2}$=1.96

| 1-$\alpha$ | Z |
|---|---|
| 0.99 | 2.58 |
| 0.98 | 2.33 |
| 0.95 | 1.96 |
| 0.90 | 1.65 |

| N | 50 | 100 | 500 | 1000 | 5000 |
|---|---|---|---|---|---|
| p(lower) | 0.670 | 0.711 | 0.763 | 0.774 | 0.789 |
| p(upper) | 0.888 | 0.866 | 0.833 | 0.824 | 0.811 |

# Comparing Performance of 2 Models

- Given two models, say M1 and M2, which is better?
  - M1 is tested on D1 (size=n1), found error rate = $e_1$
  - M2 is tested on D2 (size=n2), found error rate = $e_2$
  - Assume D1 and D2 are independent
  - If n1 and n2 are sufficiently large, then

$$e_1 \sim N(\mu_1, \sigma_1)$$
$$e_2 \sim N(\mu_2, \sigma_2)$$

  - Approximate: $\hat{\sigma}_i = \frac{e_i(1-e_i)}{n_i}$

# Comparing Performance of 2 Models

- To test if performance difference is statistically significant: d = e1 – e2
  - d ~ $N(d_t,\sigma_t)$ where $d_t$ is the true difference
  - Since D1 and D2 are independent, their variance adds up:

$$\sigma_t^2 = \sigma_1^2 + \sigma_2^2 \cong \hat{\sigma}_1^2 + \hat{\sigma}_2^2$$

$$= \frac{e1(1-e1)}{n1} + \frac{e2(1-e2)}{n2}$$

  - At (1-$\alpha$) confidence level, $d_t = d \pm Z_{\alpha/2}\hat{\sigma}_t$

# An Illustrative Example

- Given: M1: n1 = 30, e1 = 0.15
  M2: n2 = 5000, e2 = 0.25
- d = |e2 – e1| = 0.1  (2-sided test)

$$\hat{\sigma}_d = \frac{0.15(1-0.15)}{30} + \frac{0.25(1-0.25)}{5000} = 0.0043$$

- At 95% confidence level, $Z_{\alpha/2}$=1.96

$$d_t = 0.100 \pm 1.96 \times \sqrt{0.0043} = 0.100 \pm 0.128$$

=> Interval contains 0 => difference may not be statistically significant

# Comparing Performance of 2 Algorithms

- Each learning algorithm may produce k models:
  - L1 may produce M11 , M12, …, M1k
  - L2 may produce M21 , M22, …, M2k
- If models are generated on the same test sets D1,D2, …, Dk (e.g., via cross-validation)
  - For each set: compute $d_j = e_{1j} – e_{2j}$
  - $d_j$ has mean $d_t$ and variance $\sigma_t$
  - Estimate:

$$\hat{\sigma}_t^2 = \frac{\sum_{j=1}^{k} (d_j - \overline{d})^2}{k(k-1)}$$

$$d_t = d \pm t_{1-\alpha,k-1} \hat{\sigma}_t$$