InfoLab

IC612: Data Warehousing and Data Mining

# Lecture 8:
# Data Classification

Min-Soo Kim

DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology

# Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by **labels** indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

# Prediction Problems: Classification vs. Numeric Prediction

- Classification
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Numeric Prediction
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical applications
  - Credit/loan approval:
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is

# 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.**
- **Goal: previously unseen records should be assigned a class as accurately as possible.**
  - 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.

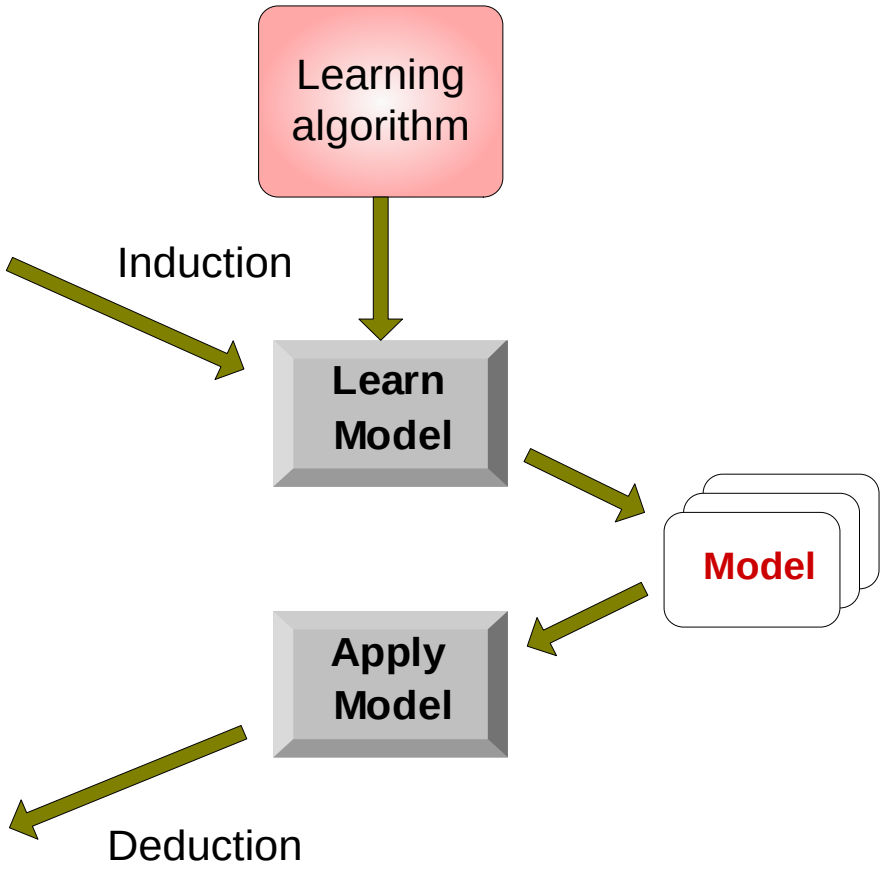
# Illustrating Classification Task

| 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





InfoLab DGIST 대구경북과학기술원

# Classification: A Two-Step Process

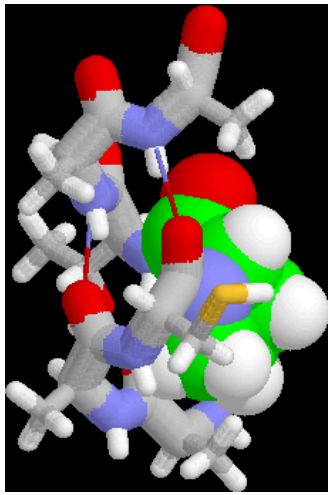
- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set (otherwise overfitting)
  - If the accuracy is acceptable, use the model to classify new data
- Note: If *the test set* is used to select models, it is called validation (test) set

# Examples of Classification Task

- Predicting tumor cells as benign or malignant

- Classifying credit card transactions as legitimate or fraudulent

- Classifying secondary structures of protein
  as alpha-helix, beta-sheet, or random coil



- Categorizing news stories as finance, weather, entertainment, sports, etc

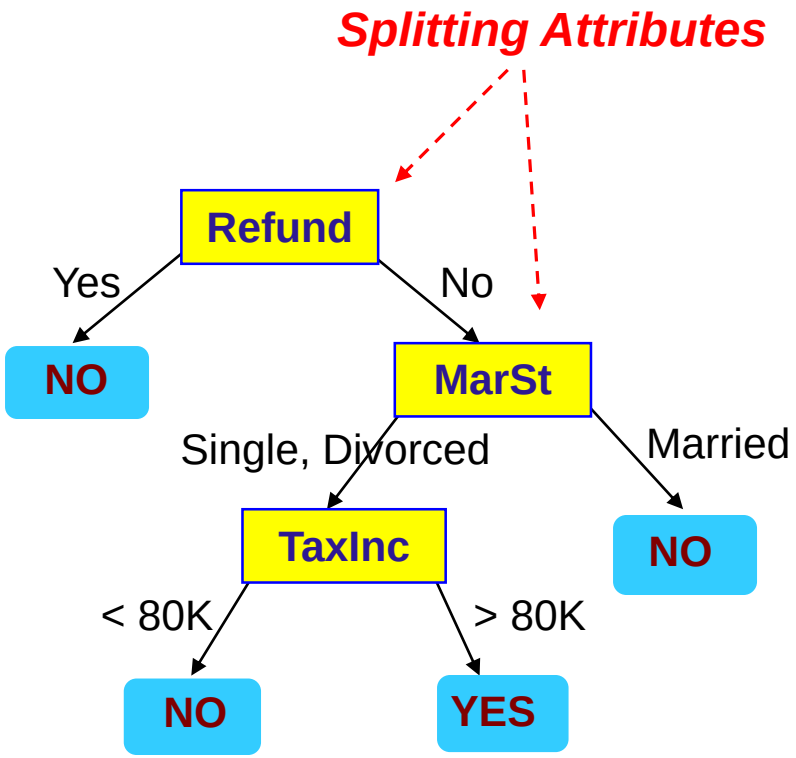
# Classification Techniques

- Decision Tree based Methods
- Rule-based Methods
- 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 | 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

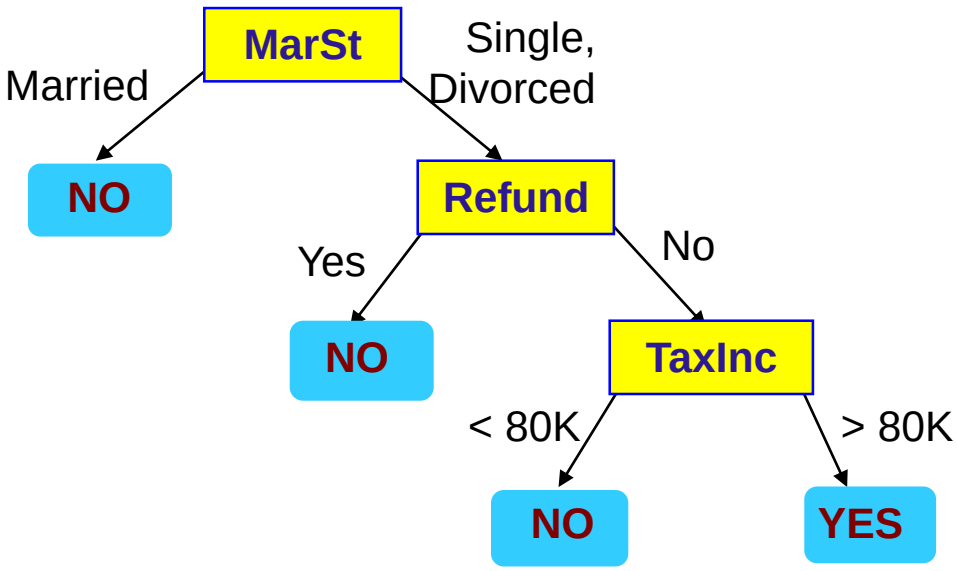



Model: Decision Tree


InfoLab DGIST 대구경북과학기술원

# Another Example of Decision Tree

| Tid | Refund (categorical) | Marital Status (categorical) | Taxable Income (continuous) | Cheat (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 |




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


InfoLab DGIST 대구경북과학기술원

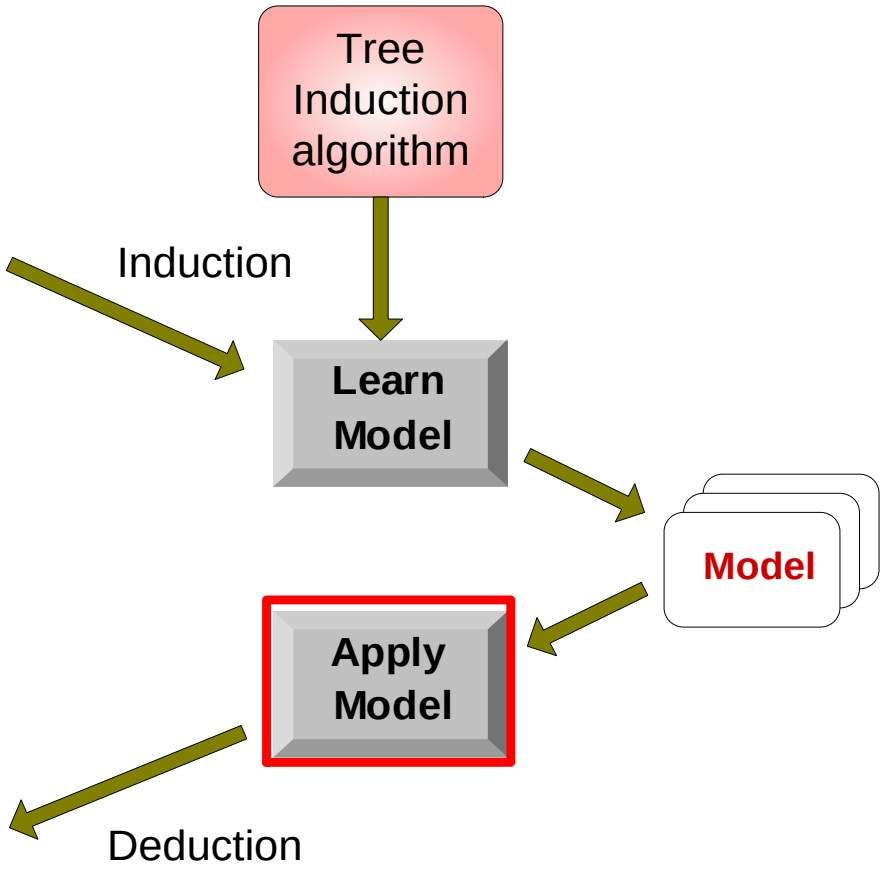
# Decision Tree Classification Task

| 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

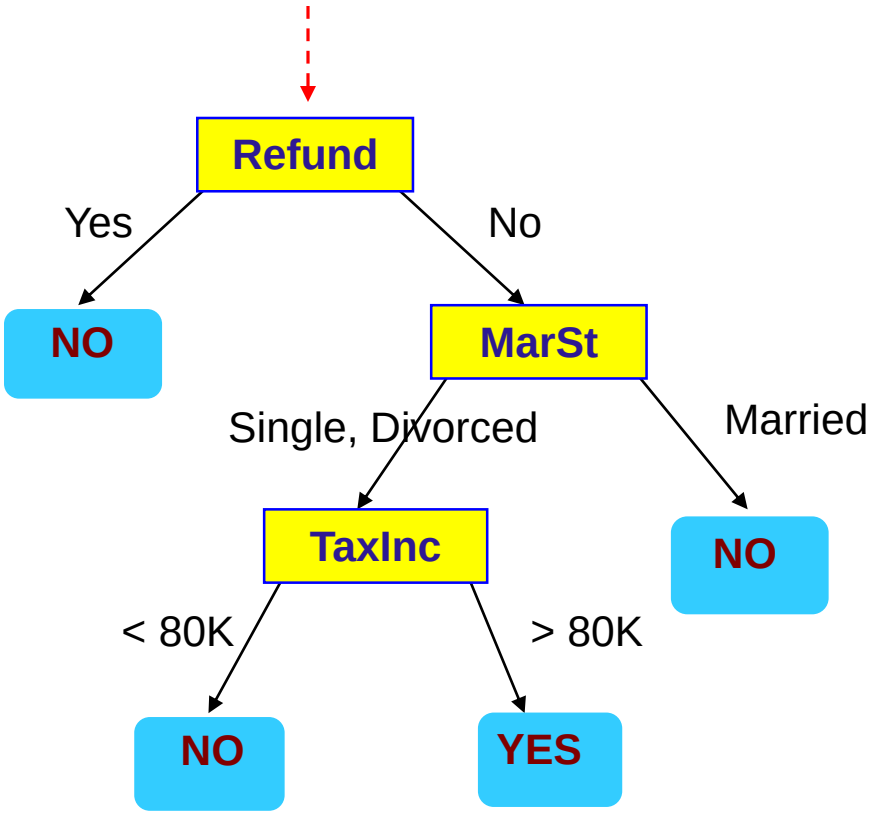




InfoLab DGIST 대구경북과학기술원

# Apply Model to Test Data

Start from the root of tree.

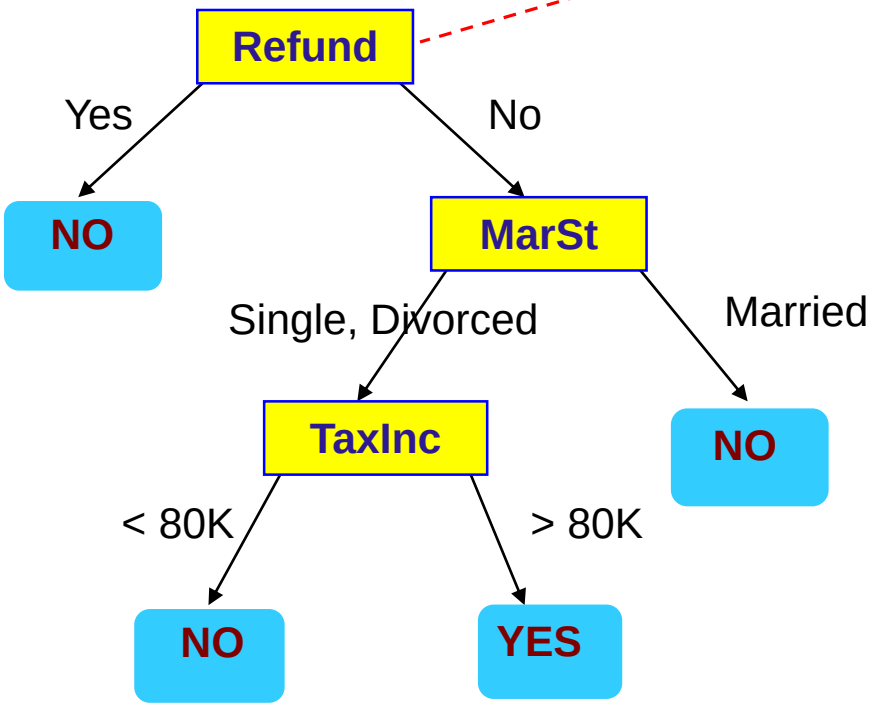



## Test Data

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


InfoLab DGIST 대구경북과학기술원

# Apply Model to Test Data

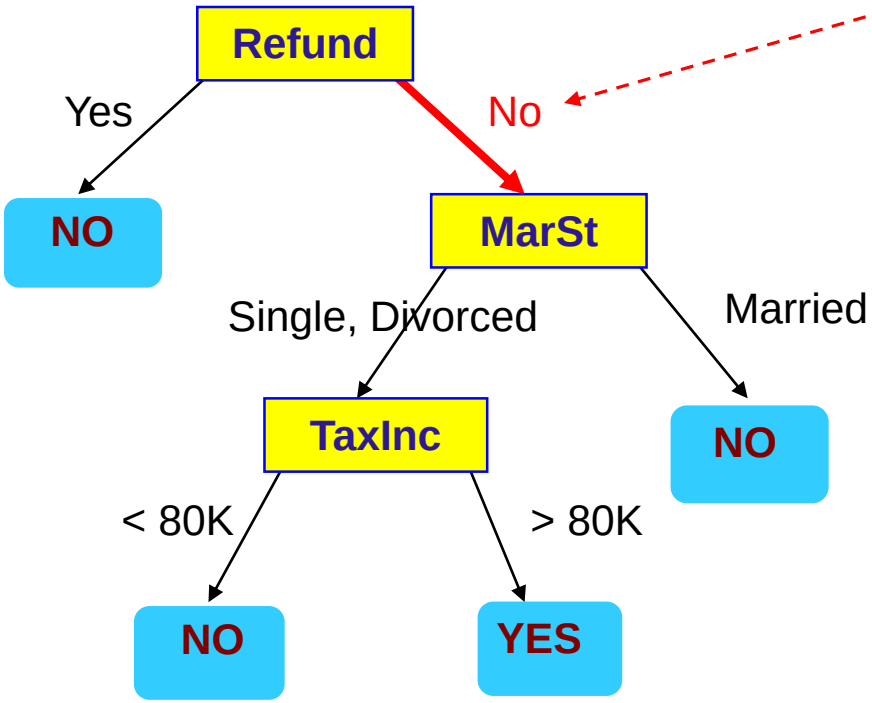



## Test Data

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


InfoLab DGIST 대구경북과학기술원

# Apply Model to Test Data

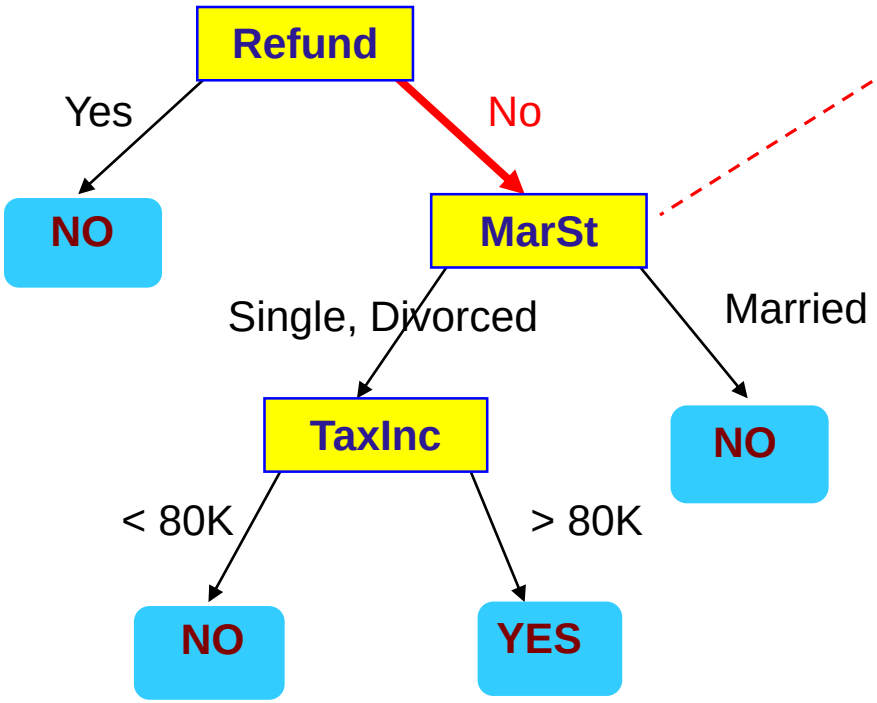



## Test Data

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


InfoLab DGIST 대구경북과학기술원

# Apply Model to Test Data

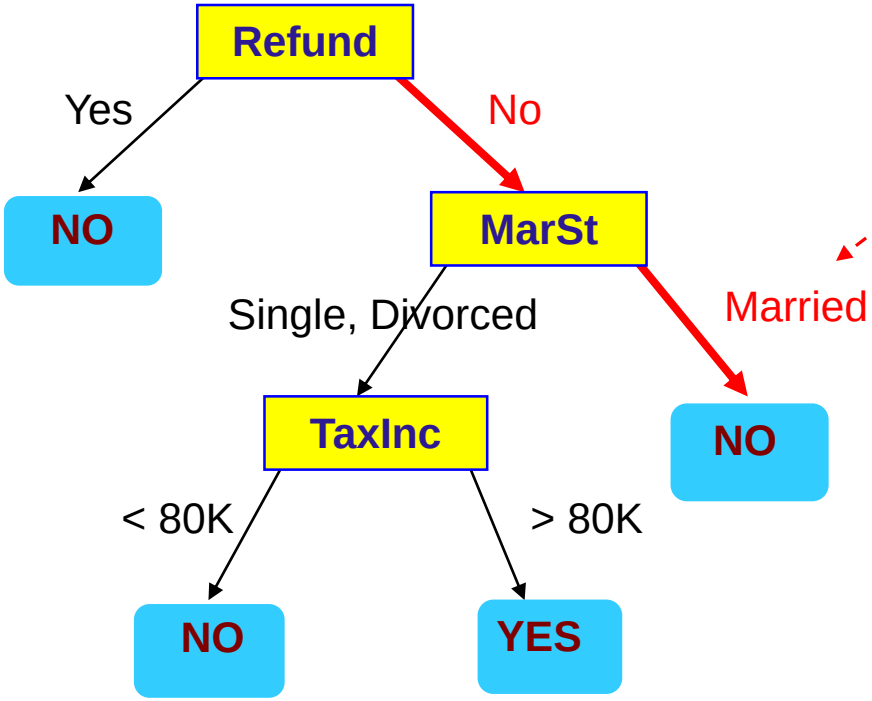



## Test Data

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


InfoLab DGIST 대구경북과학기술원

# Apply Model to Test Data

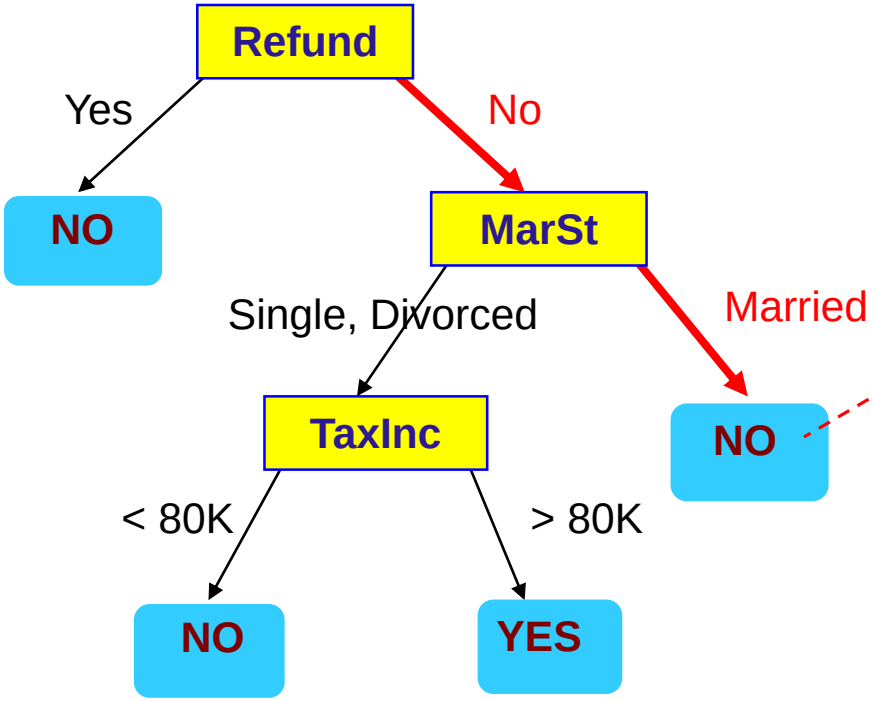



## Test Data

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


InfoLab DGIST 대구경북과학기술원

# Apply Model to Test Data




## Test Data

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

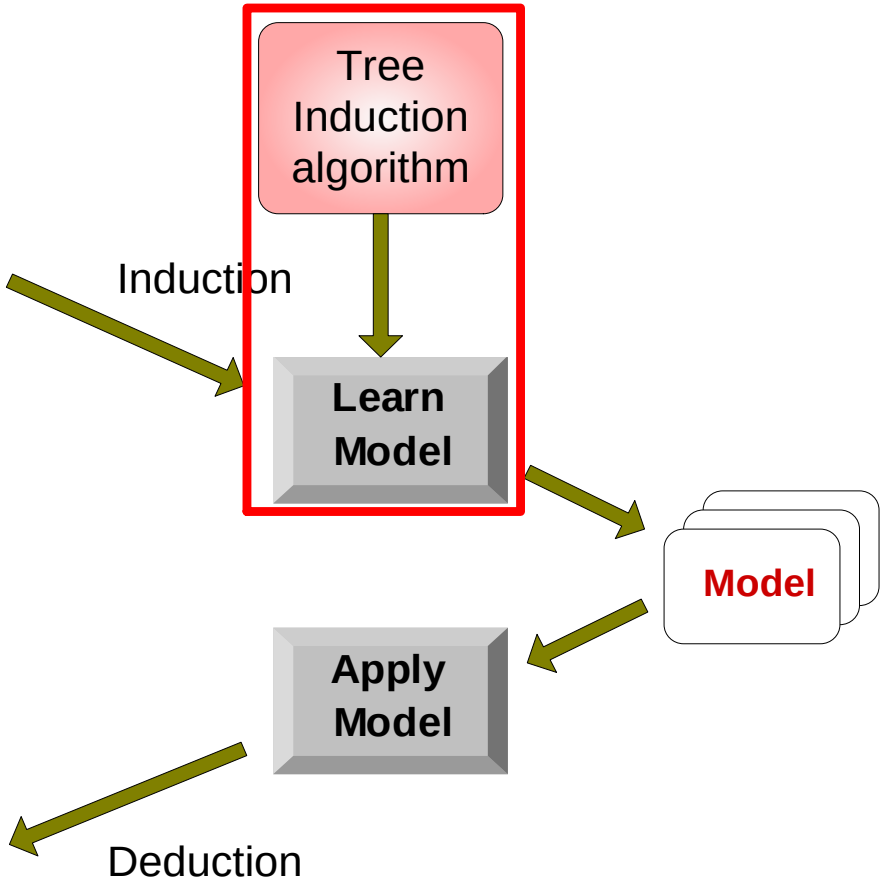
Assign Cheat to “No”


InfoLab DGIST 대구경북과학기술원

# 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


InfoLab DGIST 대구경북과학기술원

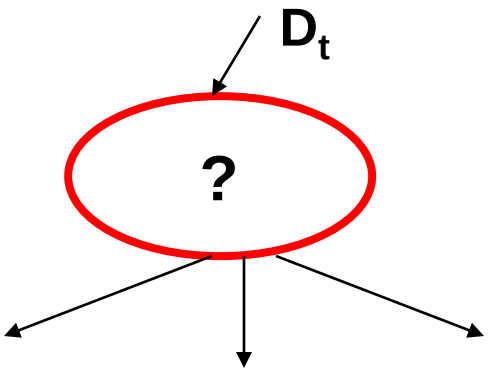
# Decision Tree Induction

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

# General Structure of Hunt's Algorithm

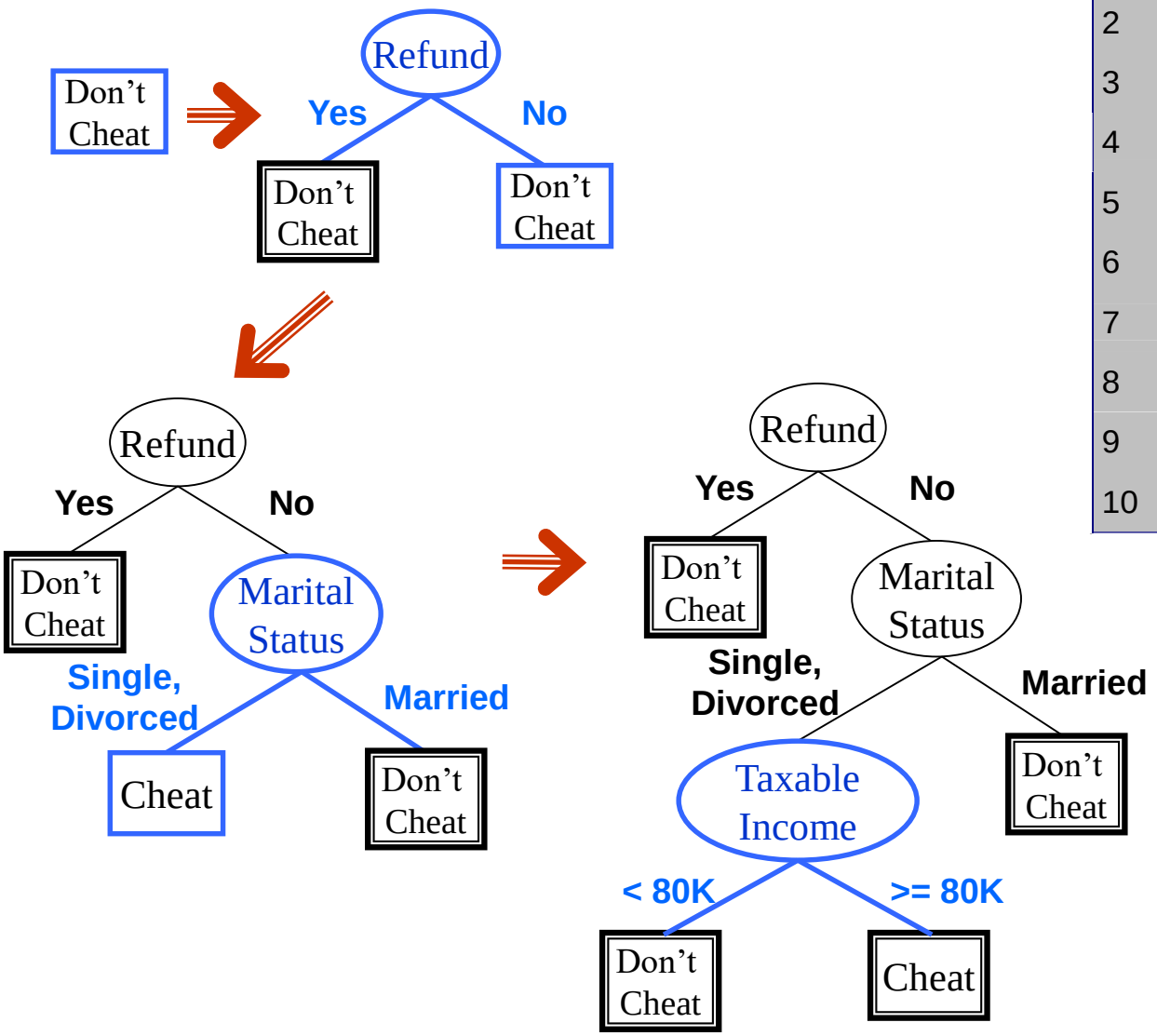
- **Let $D_t$ be the set of training records that reach a node t**
- **General Procedure:**
  - If $D_t$ contains records that belong the same class $y_t$, then t is a leaf node labeled as $y_t$
  - If $D_t$ is an empty set, then t is a leaf node labeled by the default class, $y_d$
  - If $D_t$ contains 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.

| *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 |



# Hunt's Algorithm

| 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 |





InfoLab DGIST 대구경북과학기술원

# Tree Induction

- **Greedy strategy.**
  - Split the records based on an attribute test that optimizes certain criterion.

- **Issues**
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

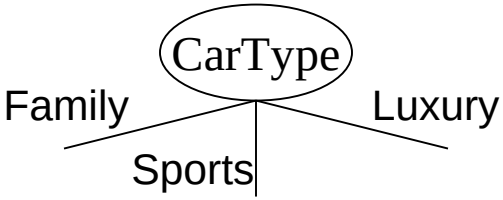

InfoLab DGIST 대구경북과학기술원

# How to Specify Test Condition?

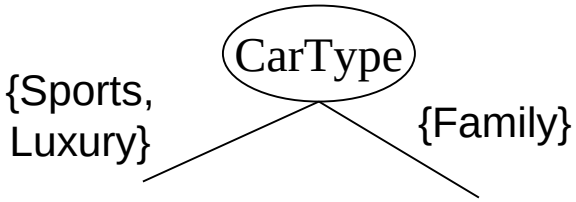
- **Depends on attribute types**
  - Nominal
  - Ordinal
  - Continuous
- **Depends on number of ways to split**
  - 2-way split
  - Multi-way split

# Splitting Based on Nominal Attributes

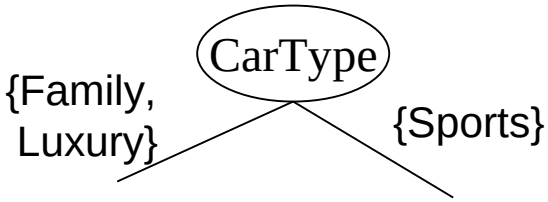
- **Multi-way split:** **Use as many partitions as distinct values.**




- **Binary split:** **Divides values into two subsets. Need to find optimal partitioning.**




OR

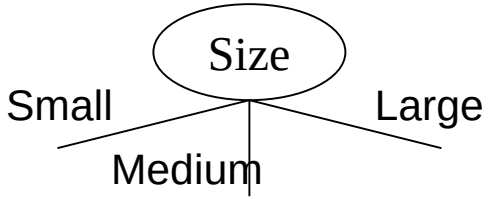




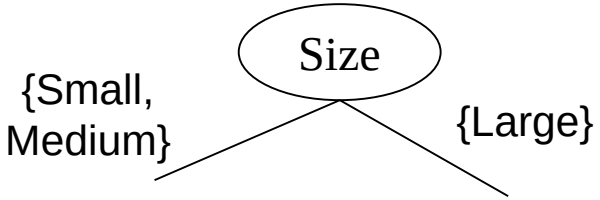
InfoLab DGIST 대구경북과학기술원

# Splitting Based on Ordinal Attributes

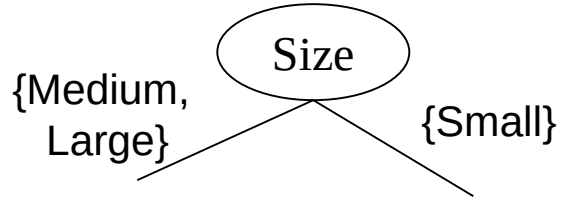
- **Multi-way split:** **Use as many partitions as distinct values.**



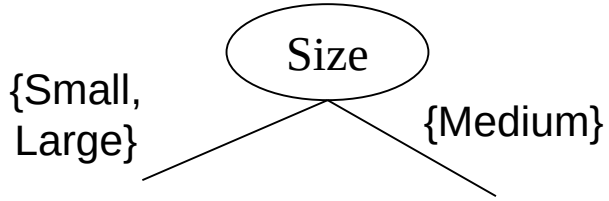

- **Binary split:** **Divides values into two subsets. Need to find optimal partitioning.**




OR




- **What about this split?**



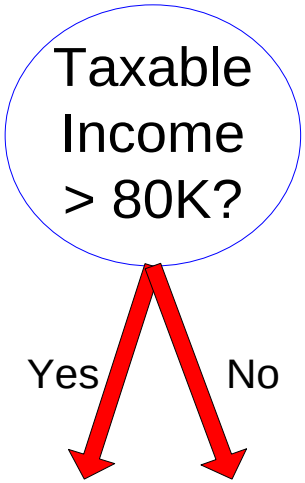


InfoLab DGIST 대구경북과학기술원

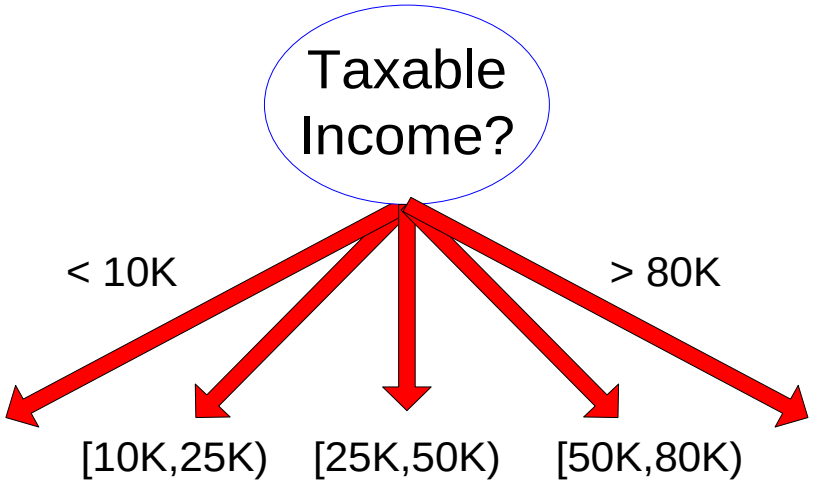
# Splitting Based on Continuous Attributes

- **Different ways of handling**
  - Discretization to form an ordinal categorical attribute
    - Static – discretize once at the beginning
    - Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
  - Binary Decision: $(A < v)$ or $(A \geq v)$
    - consider all possible splits and finds the best cut
    - can be more compute intensive

# Splitting Based on Continuous Attributes




(i) Binary split

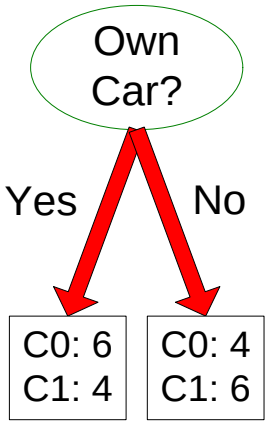


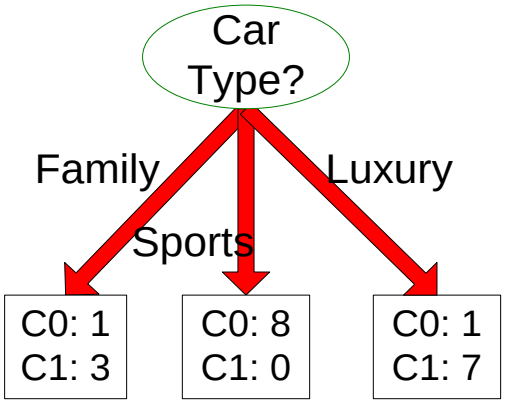

(ii) Multi-way split

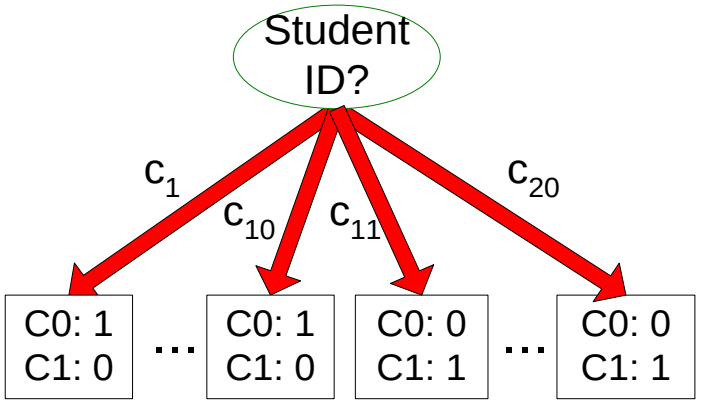

InfoLab DGIST 대구경북과학기술원

# How to determine the Best Split

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




**Which test condition is the best?**


InfoLab DGIST 대구경북과학기술원

# How to determine the Best Split

- **Greedy approach:**
  - Nodes with homogeneous class distribution are preferred
- **Need a measure of node impurity:**

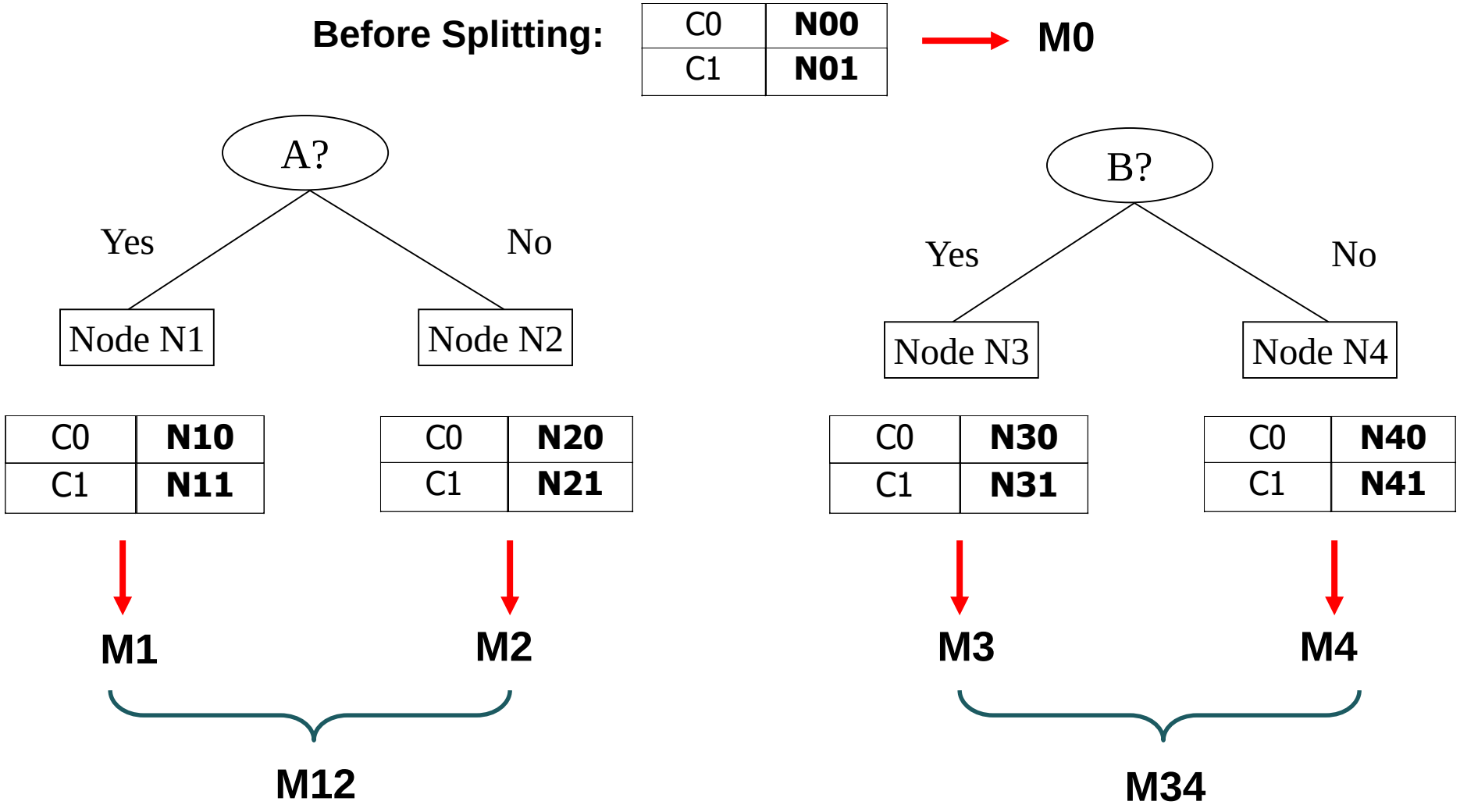
| C0: 5<br>C1: 5 | C0: 9<br>C1: 1 |
|---|---|
| **Non-homogeneous,**<br>**High degree of impurity** | **Homogeneous,**<br>**Low degree of impurity** |


InfoLab DGIST 대구경북과학기술원

# Measures of Node Impurity

- Gini Index
- Entropy
- Misclassification error

# How to Find the Best Split



# Measure of Impurity: GINI

- **Gini Index for a given node t :**

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

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

- 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

| 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 | |

# Examples for computing GINI

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

| 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**


InfoLab DGIST 대구경북과학기술원

# Splitting Based on GINI

- **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,**

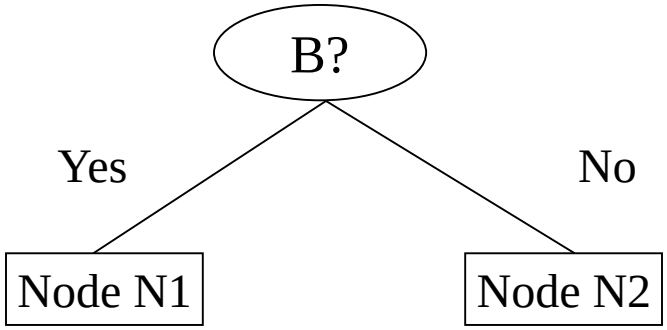
**n = number of records at node p.**



InfoLab DGIST 대구경북과학기술원

# Binary Attributes: Computing GINI Index

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




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

**Gini(N1)**
**= 1 – (5/7)² – (2/7)²**
**= 0.408**

**Gini(N2)**
**= 1 – (1/5)² – (4/5)²**
**= 0.320**

| | N1 | N2 |
|---|---|---|
| C1 | **5** | **1** |
| C2 | **2** | **4** |
| **Gini=0.371** | | |

**Gini(Children)**
**= 7/12 * 0.408 +**
**5/12 * 0.320**
**= 0.371**


InfoLab DGIST 대구경북과학기술원

# Categorical Attributes: Computing Gini Index

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

Multi-way split

| | CarType | | |
|---|---|---|---|
| | Family | Sports | Luxury |
| C1 | 1 | 2 | 1 |
| C2 | 4 | 1 | 1 |
| Gini | 0.393 | | |

Two-way split
(find best partition of values)

| | CarType | |
|---|---|---|
| | {Sports, Luxury} | {Family} |
| C1 | 3 | 1 |
| C2 | 2 | 4 |
| Gini | 0.400 | |

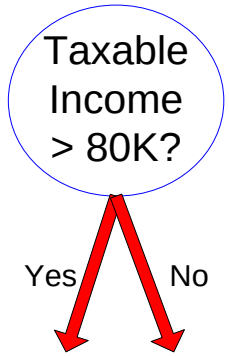
| | CarType | |
|---|---|---|
| | {Sports} | {Family, Luxury} |
| C1 | 2 | 2 |
| C2 | 1 | 5 |
| Gini | 0.419 | |


InfoLab DGIST 대구경북과학기술원

# Continuous Attributes: 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* | 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 |



# Continuous Attributes: Computing Gini Index...

- **For efficient computation: for each attribute,**
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing gini index
  - Choose the split position that has the least gini index

| Cheat | 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 | |


InfoLab DGIST 대구경북과학기술원

# 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

# 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) $\log_2$ (1/6) = 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**


InfoLab DGIST 대구경북과학기술원

# 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

# Splitting Criteria based on Classification Error

- **Classification error at a node t :**

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

- **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

# Examples for Computing 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** |

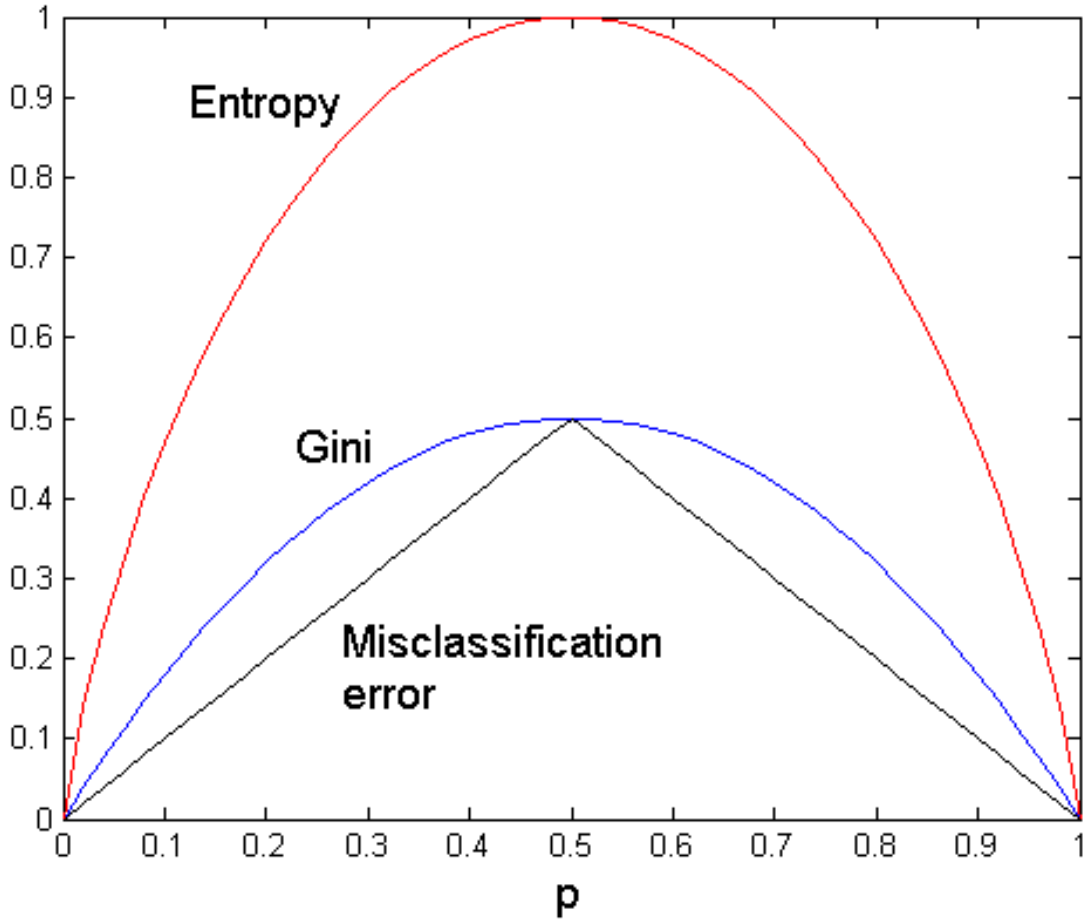
**P(C1) = 1/6** **P(C2) = 5/6**

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

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

# Comparison among Splitting Criteria

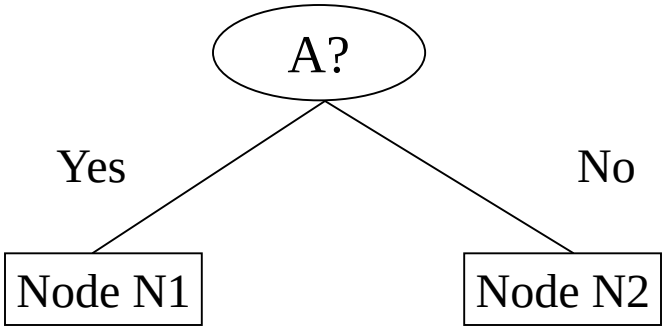
**For a 2-class problem:**





InfoLab DGIST 대구경북과학기술원

# Misclassification Error vs Gini




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

**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=0.361 | | |

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

**Gini improves !!**


InfoLab DGIST 대구경북과학기술원

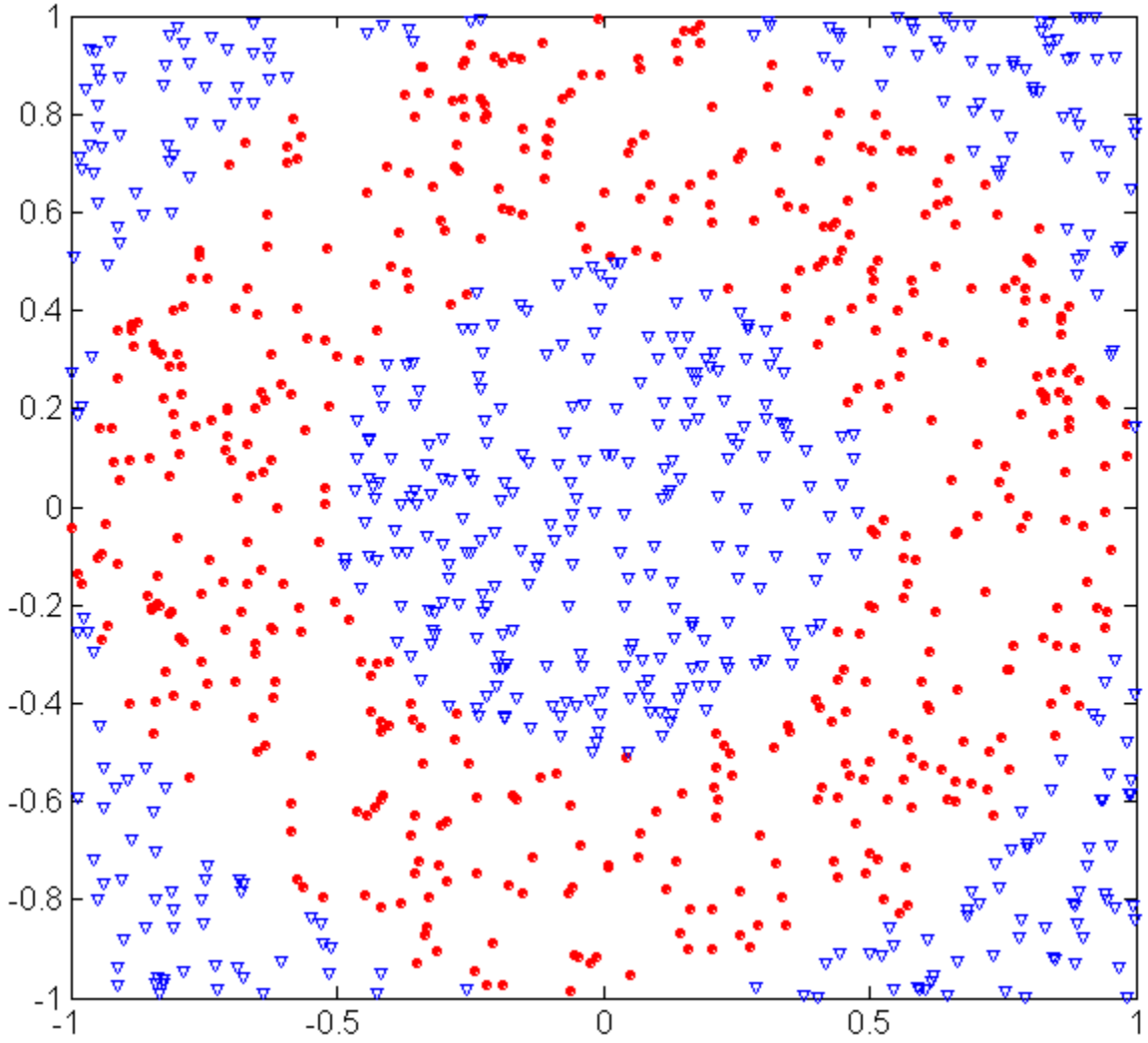
# Stopping Criteria for Tree Induction

- Stop expanding a node when all the records belong to the same class
- Stop expanding a node when all the records have similar attribute values
- Early termination (to be discussed later)

# Decision Tree Based Classification

- **Advantages:**
  - 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

# Underfitting and Overfitting (Example)




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

**Circular points:**

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

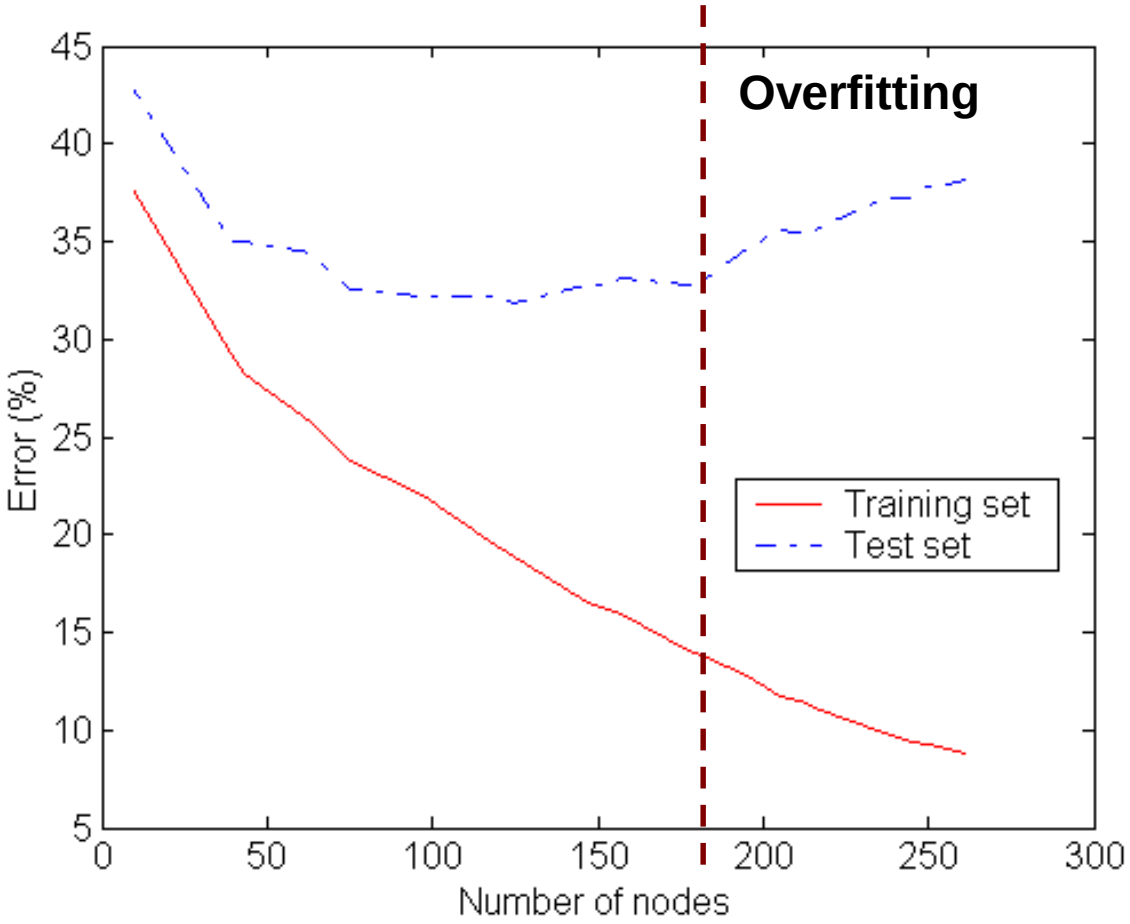
**Triangular points:**

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

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

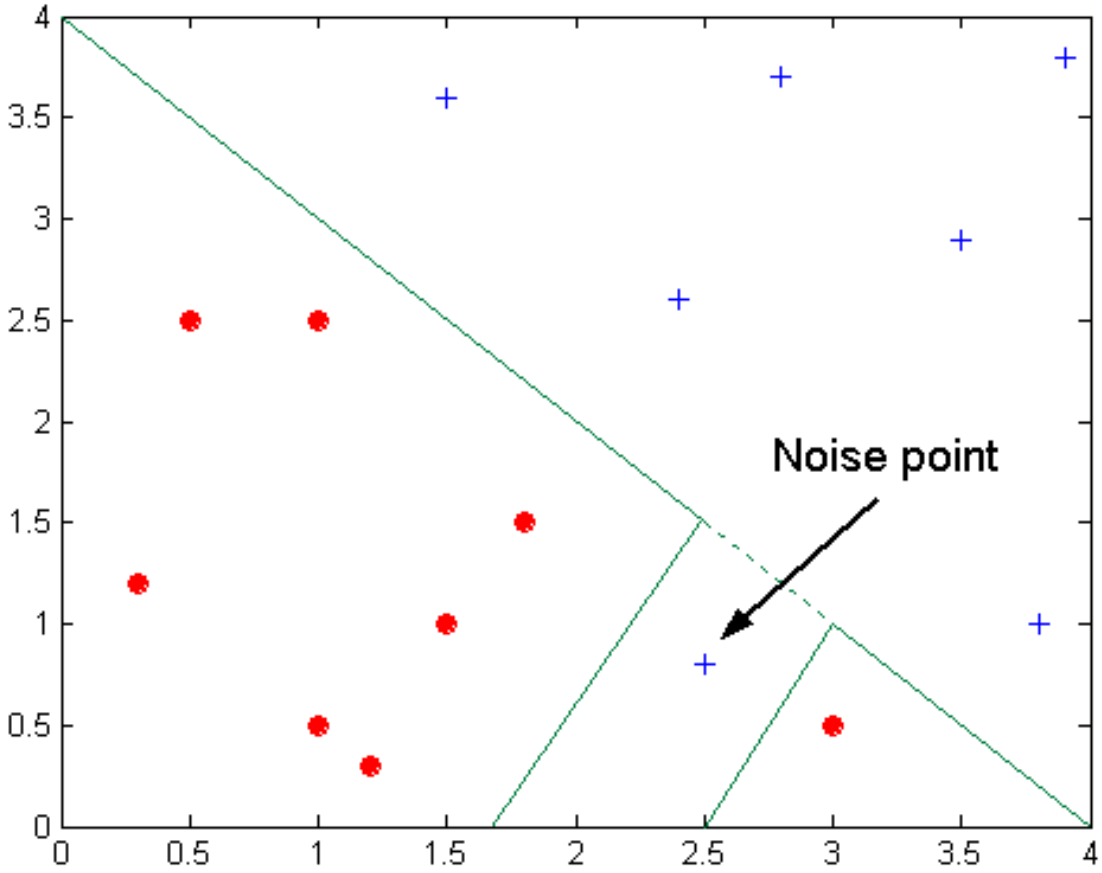

InfoLab DGIST 대구경북과학기술원

# Underfitting and Overfitting




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

# Overfitting due to Noise




**Decision boundary is distorted by noise point**

# Overfitting due to Insufficient Examples

**Table 4.5.** An example training set for classifying mammals.

| Name | Body Temperature | Gives Birth | Four-legged | Hibernates | Class Label |
|---|---|---|---|---|---|
| salamander | cold-blooded | no | yes | yes | no |
| guppy | cold-blooded | yes | no | no | no |
| eagle | warm-blooded | no | no | no | no |
| poorwill | warm-blooded | no | no | yes | no |
| platypus | warm-blooded | no | yes | yes | yes |

human

Non-mammals

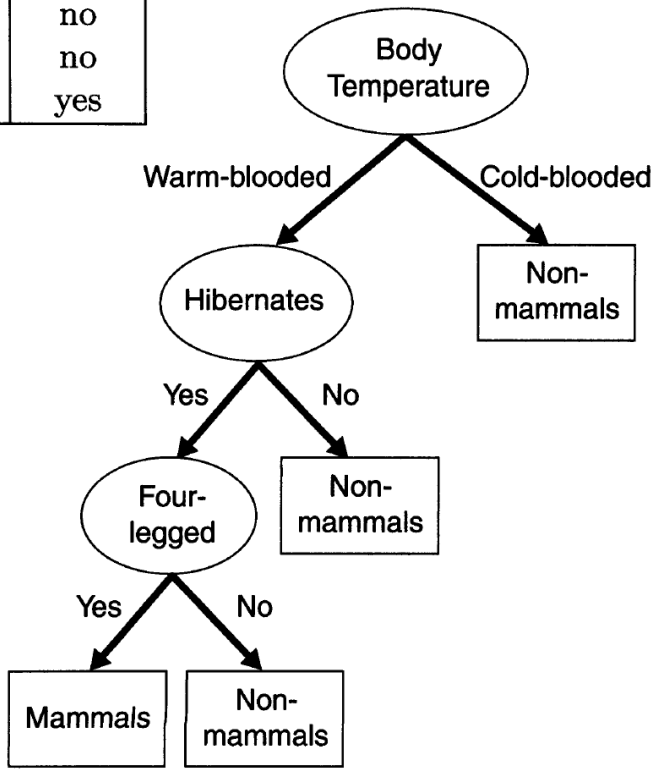



**Figure 4.26.** Decision tree induced from the data set shown in Table 4.5.


InfoLab DGIST 대구경북과학기술원

# Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary
- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records
- Need new ways for estimating errors

# Estimating Generalization Errors

- **Re-substitution errors: error on training ($\Sigma$ e(t) )**
- **Generalization errors: error on testing ($\Sigma$ e'(t))**
- **Methods for estimating generalization errors:**
  - Optimistic approach: e'(t) = e(t)
  - Pessimistic approach:
    - For each leaf node: e'(t) = (e(t)+0.5)
    - Total errors: e'(T) = e(T) + N $\times$ 0.5 (N: number of leaf nodes)
    - For a tree with 30 leaf nodes and 10 errors on training (out of 1000 instances):
      Training error = 10/1000 = 1%
      Generalization error = (10 + 30$\times$0.5)/1000 = 2.5%
  - Reduced error pruning (REP):
    - uses validation data set to estimate generalization error

# Occam's Razor

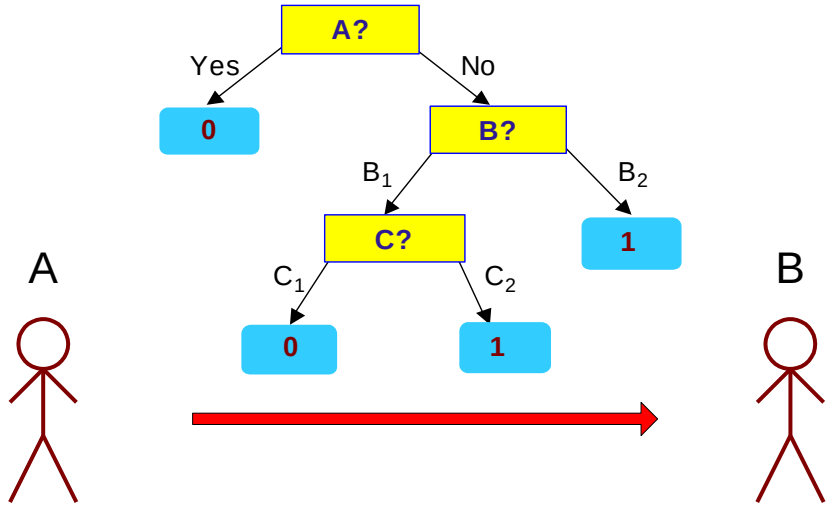
- Given two models of similar generalization errors, one should prefer the simpler model over the more complex model

- For complex models, there is a greater chance that it was fitted accidentally by errors in data

- Therefore, one should include model complexity when evaluating a model

# Minimum Description Length (MDL)

| X | y |
|---|---|
| $X_1$ | 1 |
| $X_2$ | 0 |
| $X_3$ | 0 |
| $X_4$ | 1 |
| … | … |
| $X_n$ | 1 |




| X | y |
|---|---|
| $X_1$ | ? |
| $X_2$ | ? |
| $X_3$ | ? |
| $X_4$ | ? |
| … | … |
| $X_n$ | ? |

- **Cost(Model,Data) = Cost(Data|Model) + Cost(Model)**
  - Cost is the number of bits needed for encoding.
  - Search for the least costly model.
- **Cost(Data|Model) encodes the misclassification errors.**
- **Cost(Model) uses node encoding (number of children) plus splitting condition encoding.**

# How to Address Overfitting

- **Pre-Pruning (Early Stopping Rule)**
  - Stop the algorithm before it becomes a fully-grown tree
  - Typical stopping conditions for a node:
    - Stop if all instances belong to the same class
    - Stop if all the attribute values are the same
  - More restrictive conditions:
    - 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).

# How to Address Overfitting…

- **Post-pruning**
  - Grow decision tree to its entirety
  - Trim the nodes of the decision tree in a bottom-up fashion
  - If generalization error improves after trimming, replace sub-tree by a leaf node.
  - Class label of leaf node is determined from majority class of instances in the sub-tree
  - Can use MDL for post-pruning

# Example of Post-Pruning

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

**Training Error (Before splitting) = 10/30**

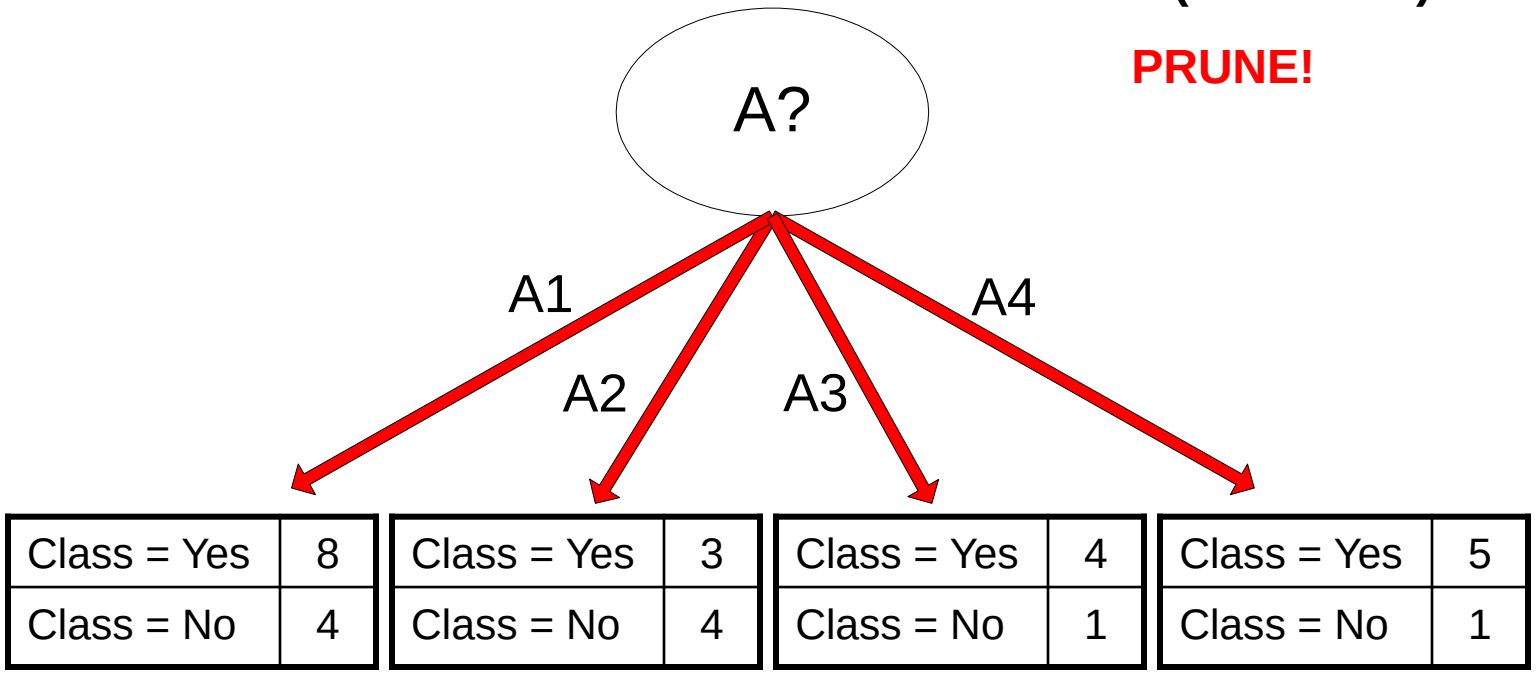
**Pessimistic error = (10 + 0.5)/30 = 10.5/30**

**Training Error (After splitting) = 9/30**

**Pessimistic error (After splitting)**

**= (9 + 4 × 0.5)/30 = 11/30**

**PRUNE!**

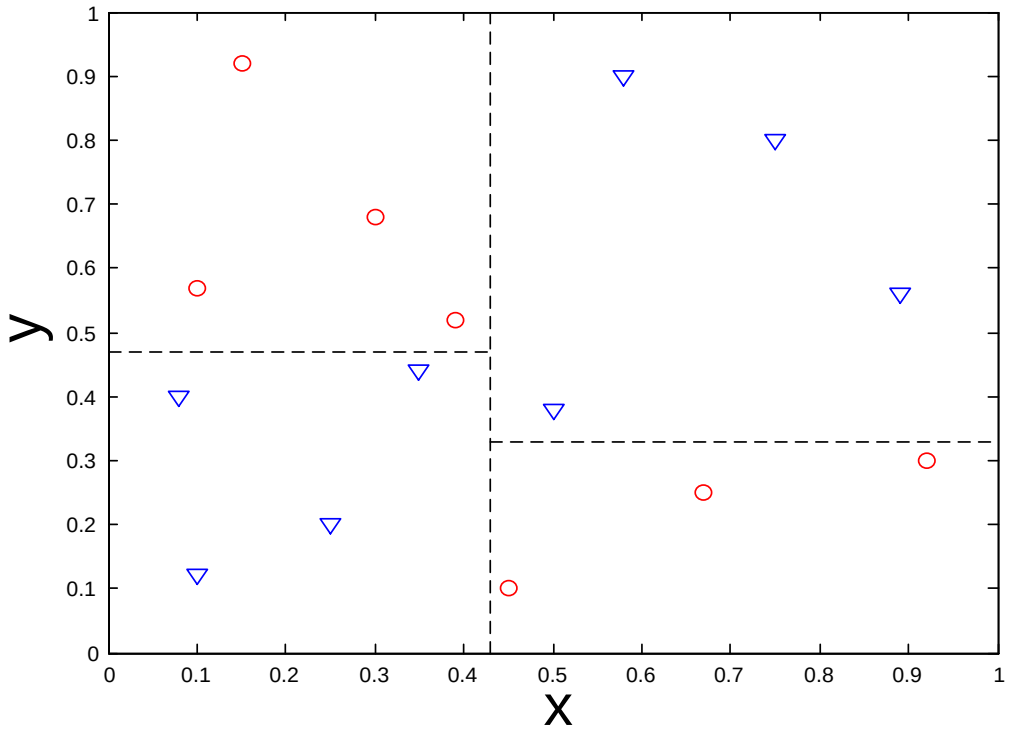


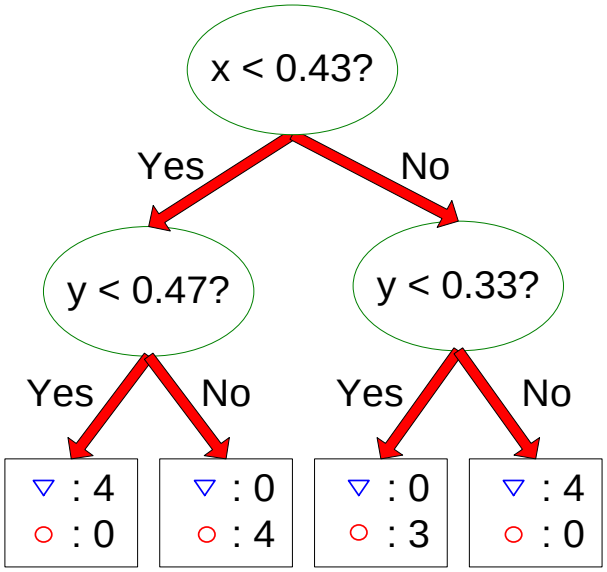
# Data Fragmentation

- Number of instances gets smaller as you traverse down the tree

- Number of instances at the leaf nodes could be too small to make any statistically significant decision

# Decision Boundary

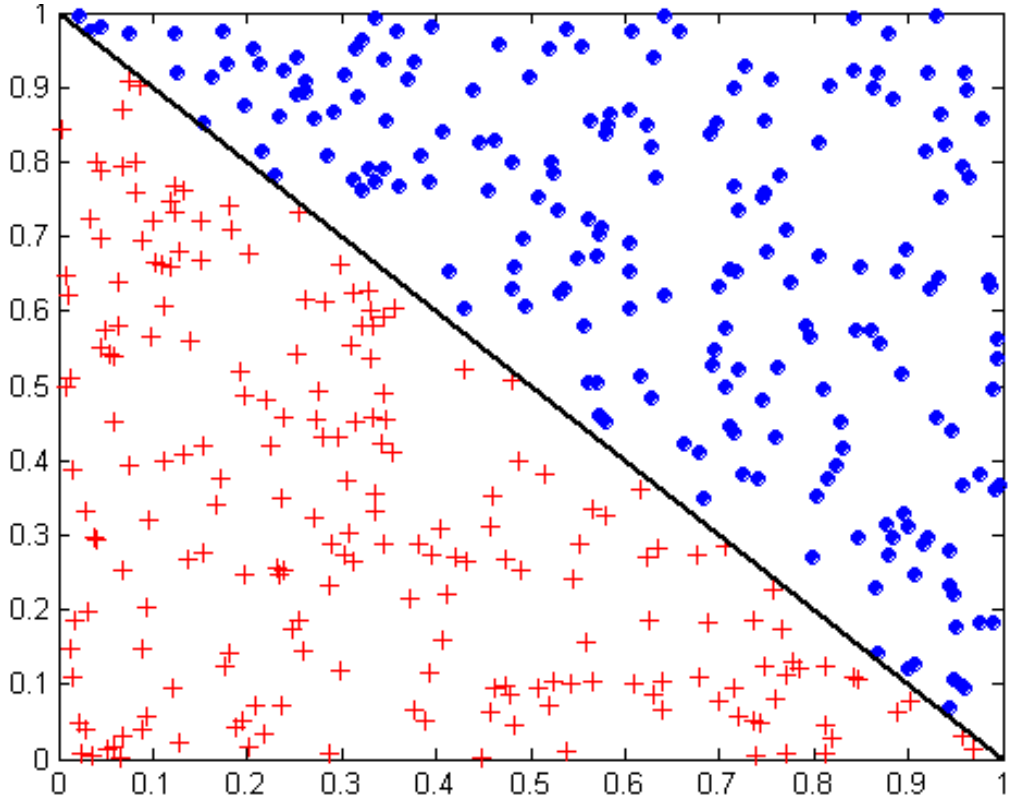


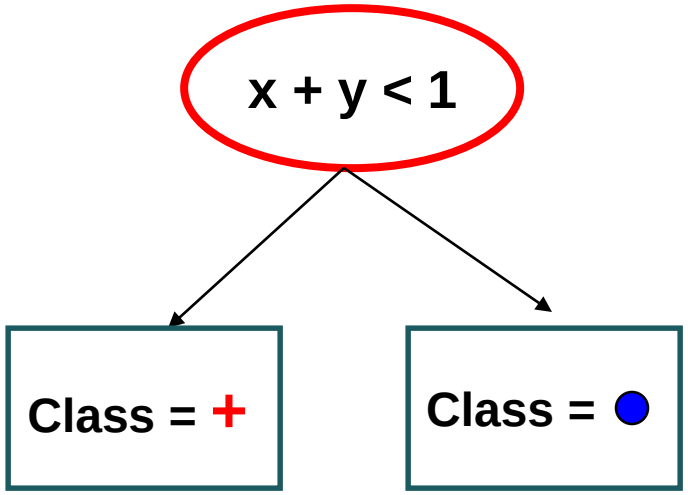




- **Border line between two neighboring regions of different classes is known as decision boundary**
- **Decision boundary is parallel to axes because test condition involves a single attribute at-a-time**


InfoLab DGIST 대구경북과학기술원

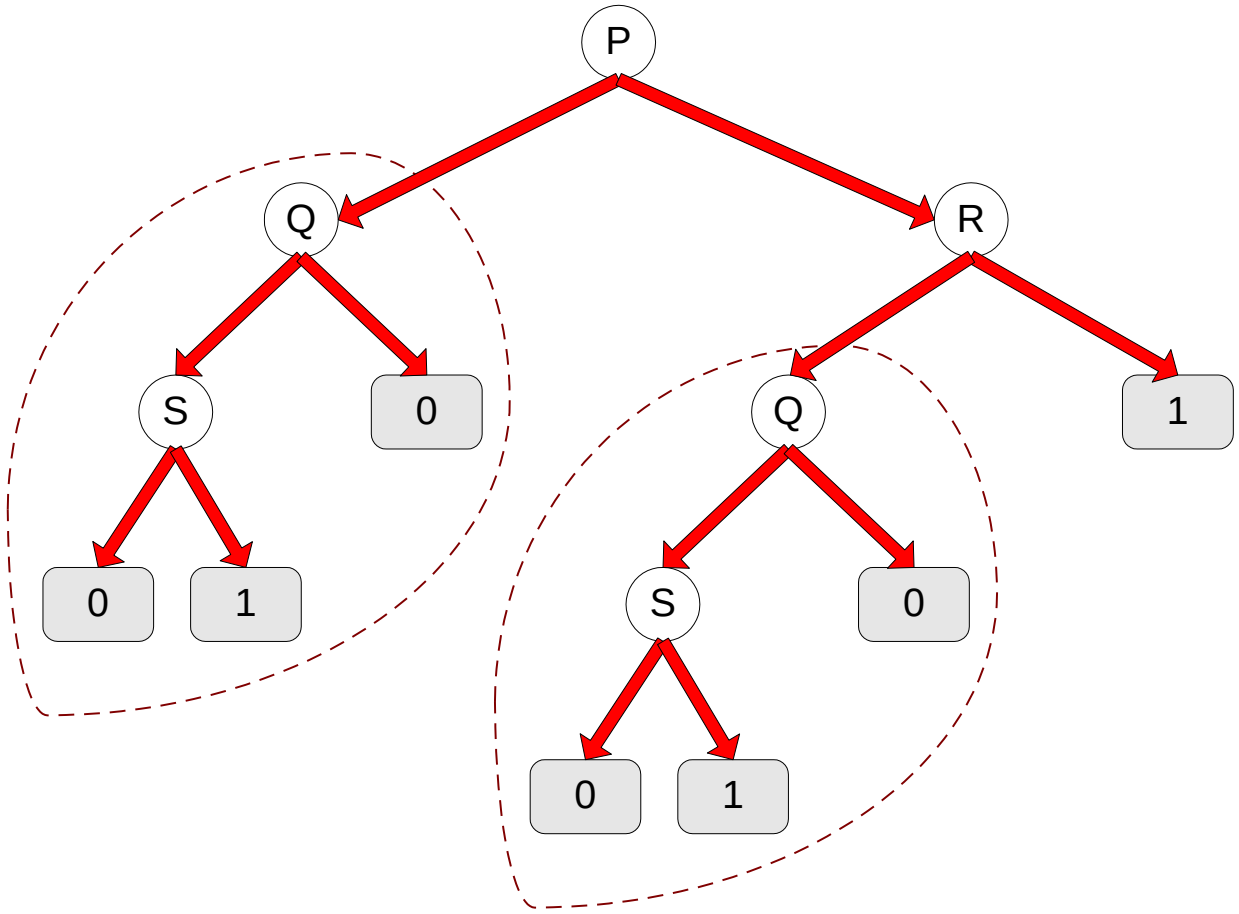
# Oblique Decision Trees




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

# Tree Replication




- **Same subtree appears in multiple branches**

# Rule-Based Classifier

- **Classify records by using a collection of "if…then…" rules**
- **Rule: (*Condition*) → *y***
  - where
    - *Condition* is a conjunctions of attributes
    - *y* is the class label
  - *LHS*: rule antecedent or condition
  - *RHS*: rule consequent
  - Examples of classification rules:
    - (Blood Type=Warm) ∧ (Lay Eggs=Yes) → Birds
    - (Taxable Income < 50K) ∧ (Refund=Yes) → Evade=No

# Rule-based Classifier (Example)

| Name | Blood Type | Give Birth | Can Fly | Live in Water | Class |
|---|---|---|---|---|---|
| human | warm | yes | no | no | mammals |
| python | cold | no | no | no | reptiles |
| salmon | cold | no | no | yes | fishes |
| whale | warm | yes | no | yes | mammals |
| frog | cold | no | no | sometimes | amphibians |
| komodo | cold | no | no | no | reptiles |
| bat | warm | yes | yes | no | mammals |
| pigeon | warm | no | yes | no | birds |
| cat | warm | yes | no | no | mammals |
| leopard shark | cold | yes | no | yes | fishes |
| turtle | cold | no | no | sometimes | reptiles |
| penguin | warm | no | no | sometimes | birds |
| porcupine | warm | yes | no | no | mammals |
| eel | cold | no | no | yes | fishes |
| salamander | cold | no | no | sometimes | amphibians |
| gila monster | cold | no | no | no | reptiles |
| platypus | warm | no | no | no | mammals |
| owl | warm | no | yes | no | birds |
| dolphin | warm | yes | no | yes | mammals |
| eagle | warm | no | yes | no | birds |

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

# Application of Rule-Based Classifier

- **A rule *r* covers an instance x if the attributes of the instance satisfy the condition of the rule**

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

| Name | Blood Type | Give Birth | Can Fly | Live in Water | Class |
|---|---|---|---|---|---|
| hawk | warm | no | yes | no | ? |
| grizzly bear | warm | yes | no | no | ? |

The rule R1 covers a hawk => Bird

The rule R3 covers the grizzly bear => Mammal

# Rule Coverage and Accuracy

- **Coverage of a rule:**
  - Fraction of records that satisfy the antecedent of a rule
- **Accuracy of a rule:**
  - Fraction of records that satisfy both the antecedent and consequent of a rule

| *Tid* | Refund | Marital Status | Taxable Income | 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 |

**(Status=Single) → No**

**Coverage = 40%, Accuracy = 50%**


InfoLab DGIST 대구경북과학기술원

# How does Rule-based Classifier Work?

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

| Name | Blood Type | Give Birth | Can Fly | Live in Water | Class |
|---|---|---|---|---|---|
| lemur | warm | yes | no | no | ? |
| turtle | cold | no | no | sometimes | ? |
| dogfish shark | cold | yes | no | yes | ? |

A lemur triggers rule R3, so it is classified as a mammal

A turtle triggers both R4 and R5

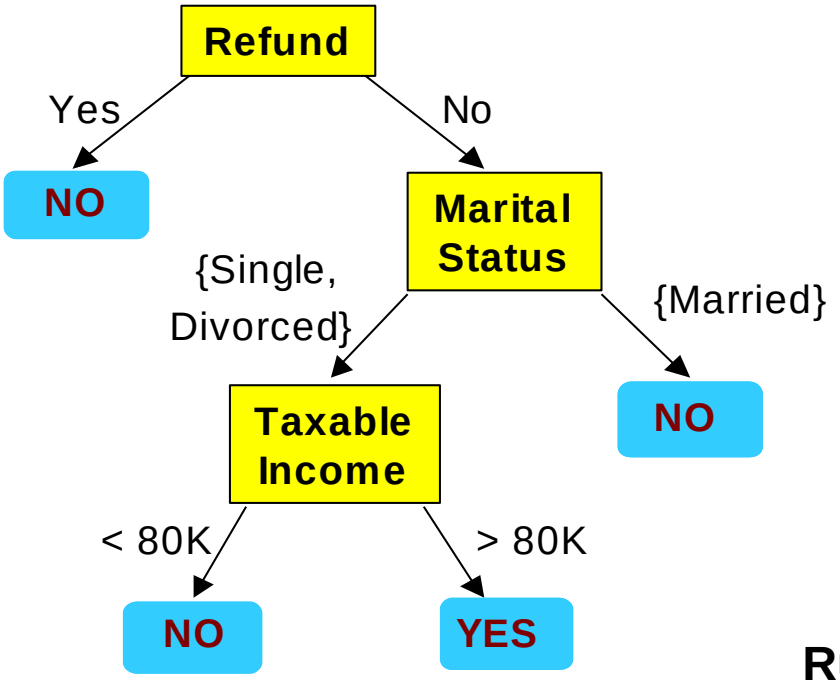
A dogfish shark triggers none of the rules

# Characteristics of Rule-Based Classifier

- **Mutually exclusive rules**
  - Classifier contains mutually exclusive rules if the rules are independent of each other
  - Every record is covered by at most one rule

- **Exhaustive rules**
  - Classifier has exhaustive coverage if it accounts for every possible combination of attribute values
  - Each record is covered by at least one rule

# From Decision Trees To Rules




**Classification Rules**

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income<80K) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income>80K) ==> Yes

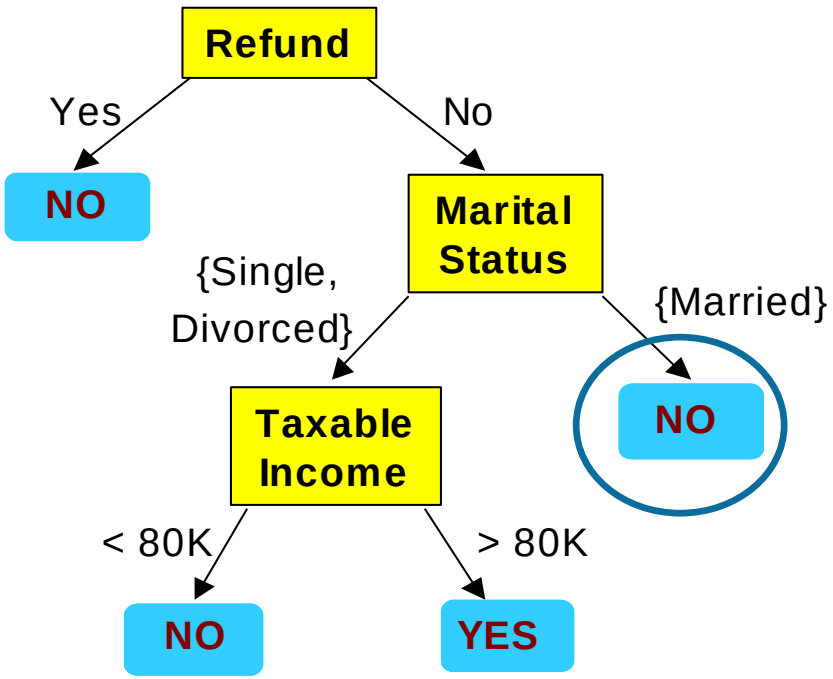
(Refund=No, Marital Status={Married}) ==> No

**Rules are mutually exclusive and exhaustive**

**Rule set contains as much information as the tree**


InfoLab DGIST 대구경북과학기술원

# Rules Can Be Simplified




| 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 |

**Initial Rule:** $(\text{Refund=No}) \wedge (\text{Status=Married}) \rightarrow \text{No}$

**Simplified Rule:** $(\text{Status=Married}) \rightarrow \text{No}$


InfoLab DGIST 대구경북과학기술원

# Effect of Rule Simplification

- **Rules are no longer mutually exclusive**
  - A record may trigger more than one rule
  - Solution?
    - Ordered rule set
    - Unordered rule set – use voting schemes

- **Rules are no longer exhaustive**
  - A record may not trigger any rules
  - Solution?
    - Use a default class

# Ordered Rule Set

- **Rules are rank ordered according to their priority**
  - An ordered rule set is known as a decision list
- **When a test record is presented to the classifier**
  - It is assigned to the class label of the highest ranked rule it has triggered
  - If none of the rules fired, it is assigned to the default class

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

| Name | Blood Type | Give Birth | Can Fly | Live in Water | Class |
|---|---|---|---|---|---|
| turtle | cold | no | no | sometimes | ? |


InfoLab DGIST 대구경북과학기술원

# Rule Ordering Schemes

- **Rule-based ordering**
  - Individual rules are ranked based on their quality
- **Class-based ordering**
  - Rules that belong to the same class appear together

**Rule-based Ordering**

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income<80K) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income>80K) ==> Yes

(Refund=No, Marital Status={Married}) ==> No

**Class-based Ordering**

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income<80K) ==> No

(Refund=No, Marital Status={Married}) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income>80K) ==> Yes

# Building Classification Rules

- **Direct Method:**
  - Extract rules directly from data
  - e.g.: RIPPER, CN2, Holte's 1R

- **Indirect Method:**
  - Extract rules from other classification models (e.g. decision trees, neural networks, etc).
  - e.g: C4.5 rules

# Direct Method: Sequential Covering

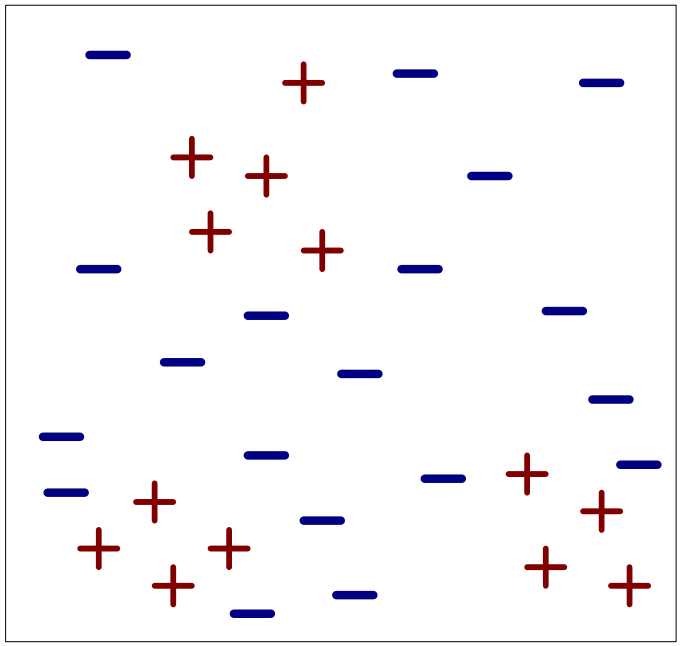
1. Start from an empty rule

2. Grow a rule using the Learn-One-Rule function

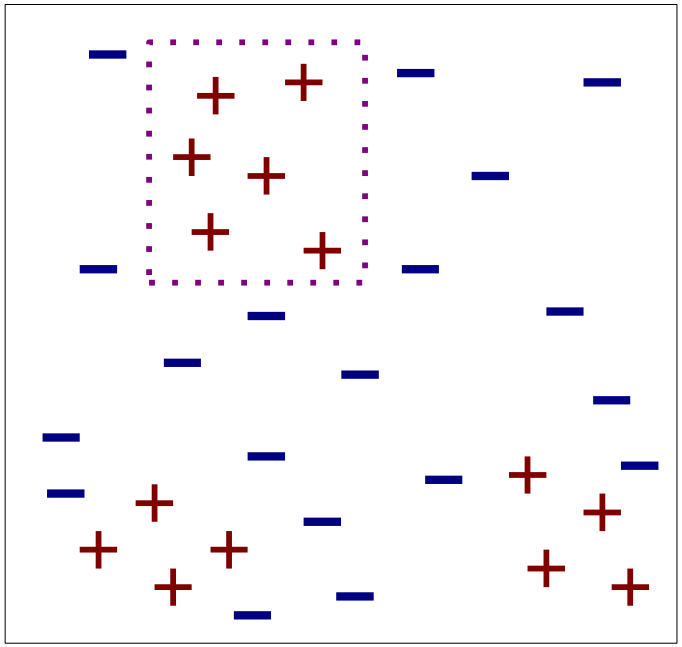
3. Remove training records covered by the rule

4. Repeat Step (2) and (3) until stopping criterion is met

# Example of Sequential Covering

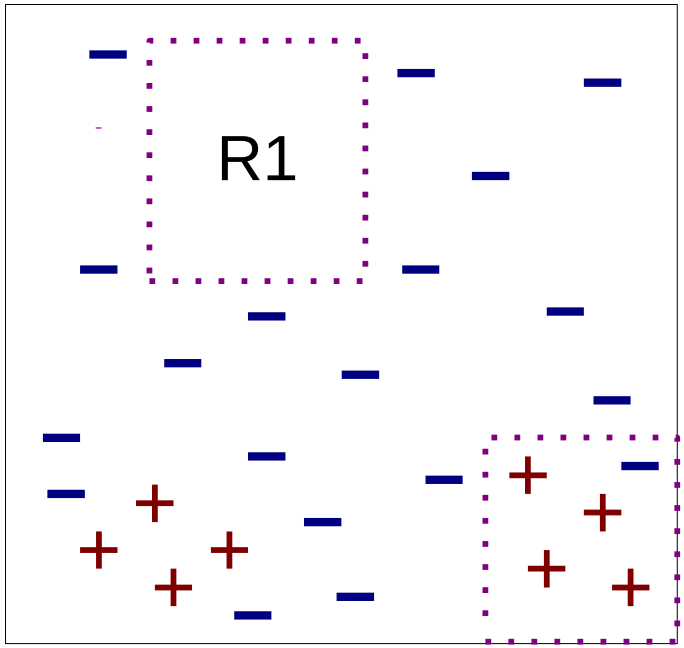


(i) Original Data

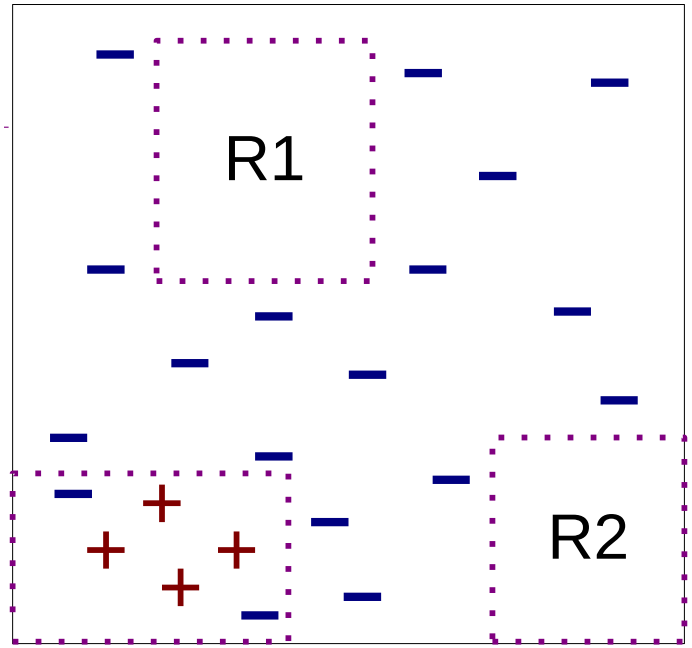


(ii) Step 1

# Example of Sequential Covering…



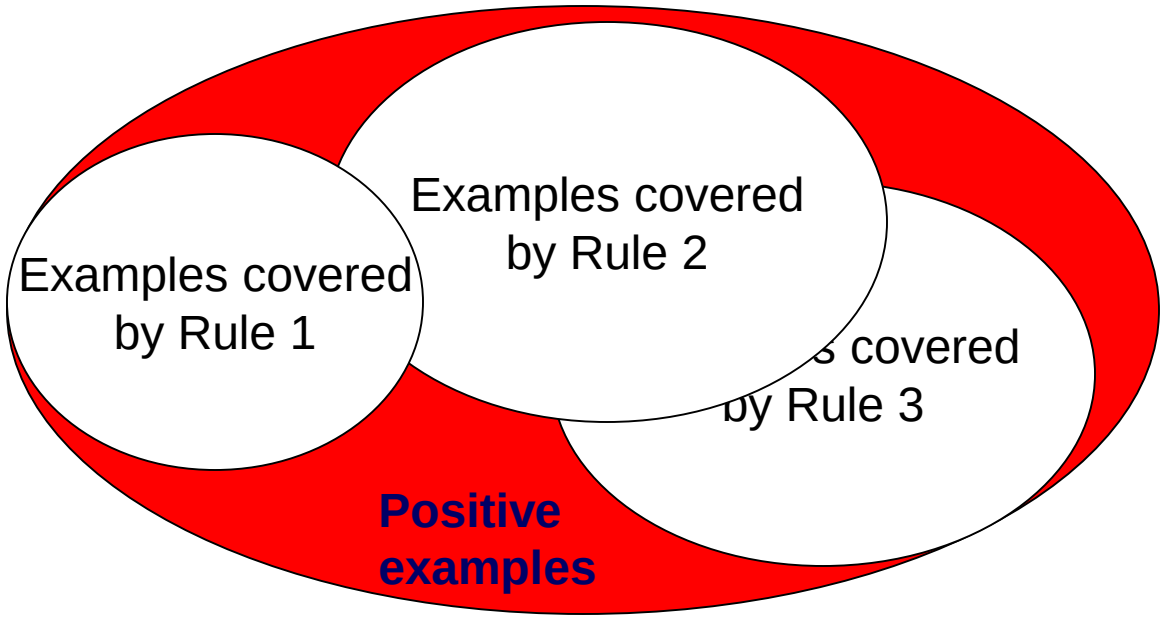

(iii) Step 2




(iv) Step 3

# Sequential Covering Algorithm

**while** (enough target tuples left)
    generate a rule
    remove positive target tuples satisfying this rule





InfoLab DGIST 대구경북과학기술원

# Rule Generation

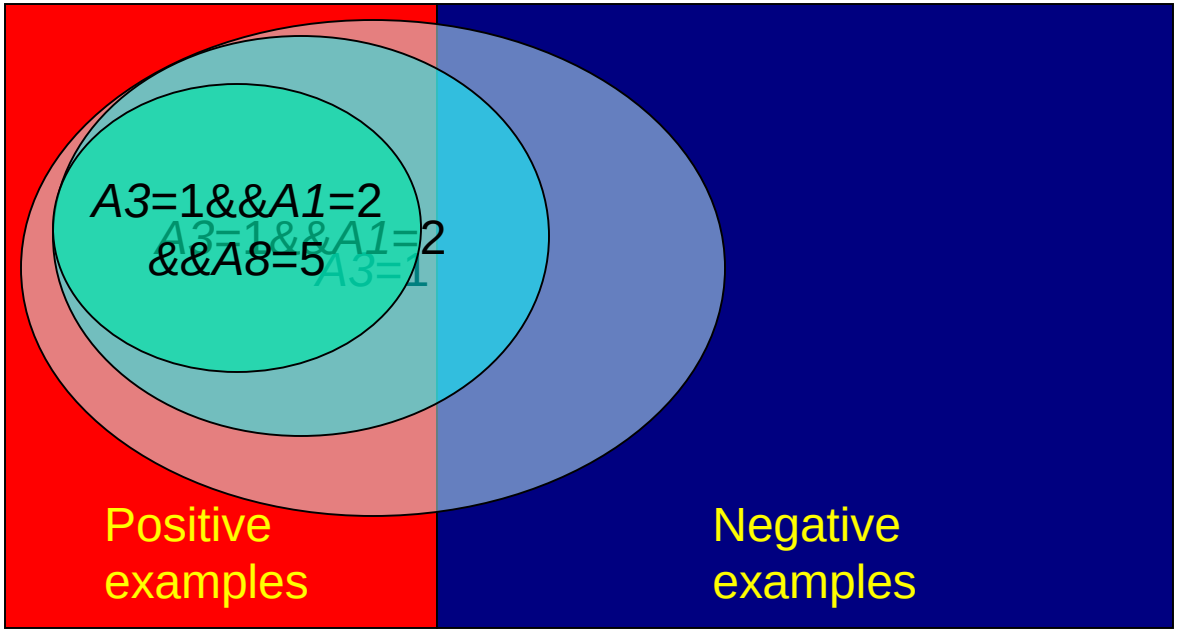
- To generate a rule

  **while**(true)

    find the best predicate *p*

    **if** foil-gain(*p*) > threshold **then** add *p* to current rule

    **else** break



# How to Learn-One-Rule?

- Start with the *most general rule* possible: condition = empty
- *Adding new attributes* by adopting a greedy depth-first strategy
  - Picks the one that most improves the rule quality
- Rule-Quality measures: consider both coverage and accuracy
  - Foil-gain (in FOIL & RIPPER): assesses info_gain by extending condition

$$FOIL_Gain = pos' \times (\log_2 \frac{pos'}{pos'+neg'} - \log_2 \frac{pos}{pos+neg})$$

    - favors rules that have high accuracy and cover many positive tuples
- Rule pruning based on an independent set of test tuples

$$FOIL_Prune(R) = \frac{pos - neg}{pos + neg}$$

Pos/neg are # of positive/negative tuples covered by R.

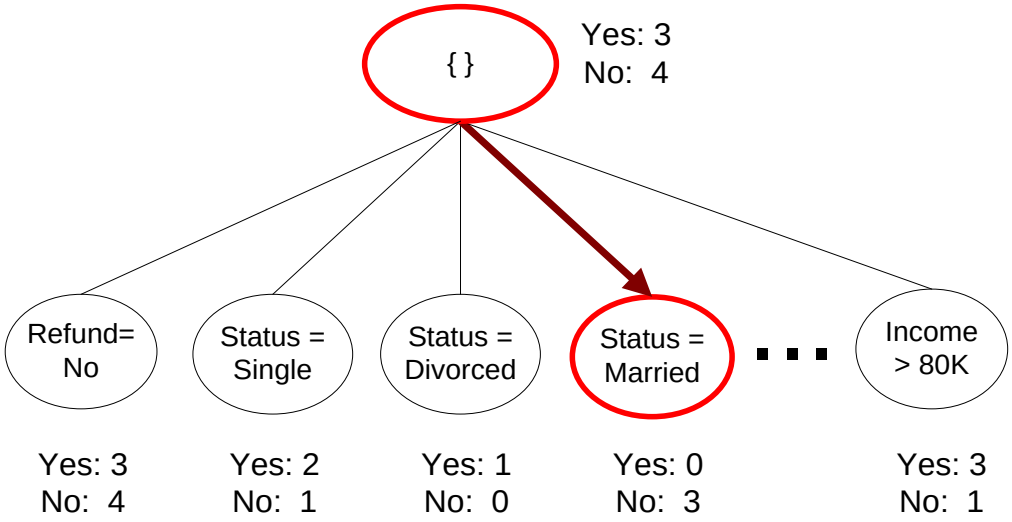
If *FOIL_Prune* is higher for the pruned version of R, prune R

# Aspects of Sequential Covering

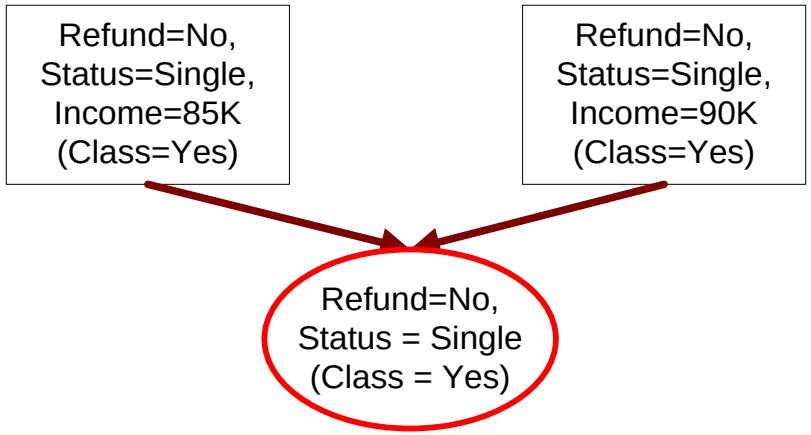
- Rule Growing
- Instance Elimination
- Rule Evaluation
- Stopping Criterion
- Rule Pruning

# Rule Growing

- **Two common strategies**




(a) General-to-specific




(b) Specific-to-general


InfoLab DGIST 대구경북과학기술원

# Rule Growing (Examples)

- **CN2 Algorithm:**
  - Start from an empty conjunct: {}
  - Add conjuncts that minimizes the entropy measure: {A}, {A,B}, …
  - Determine the rule consequent by taking majority class of instances covered by the rule

- **RIPPER Algorithm:**
  - Start from an empty rule: {} => class
  - Add conjuncts that maximizes FOIL's information gain measure:
    - R0: {} => class (initial rule)
    - R1: {A} => class (rule after adding conjunct)
    - Gain(R0, R1) = t [ log (p1/(p1+n1)) – log (p0/(p0 + n0)) ]
    - where t: number of positive instances covered by both R0 and R1

      p0: number of positive instances covered by R0

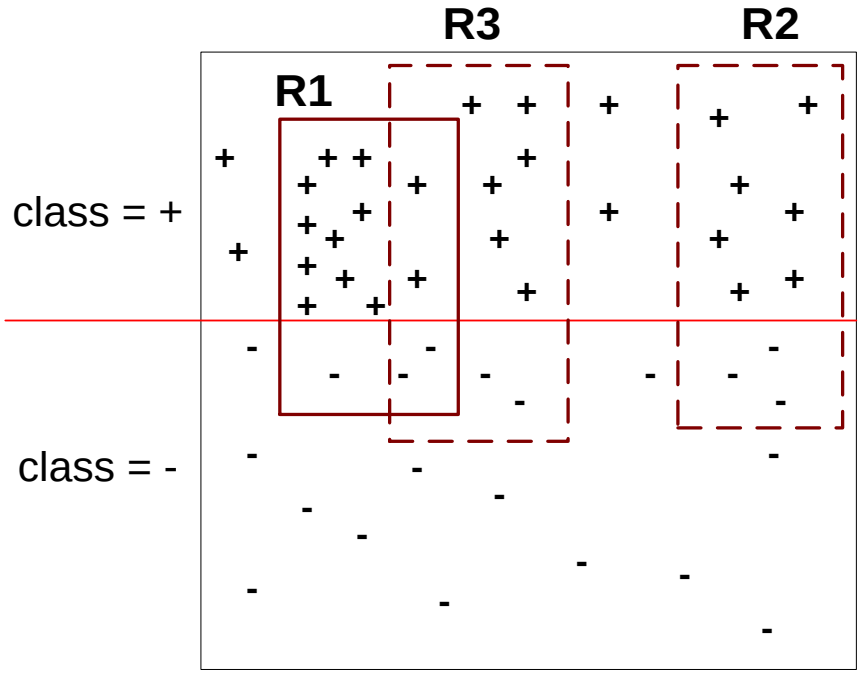
      n0: number of negative instances covered by R0

      p1: number of positive instances covered by R1

      n1: number of negative instances covered by R1

# Instance Elimination

- **Why do we need to eliminate instances?**
  - Otherwise, the next rule is identical to previous rule
- **Why do we remove positive instances?**
  - Ensure that the next rule is different
- **Why do we remove negative instances?**
  - Prevent underestimating accuracy of rule
  - Compare rules R2 and R3 in the diagram



# Rule Evaluation

- **Metrics:**
  - Accuracy $= \frac{n_c}{n}$
  - Laplace $= \frac{n_c + 1}{n + k}$
  - M-estimate $= \frac{n_c + kp}{n + k}$

$n$ : Number of instances covered by rule

$n_c$ : Number of +instances covered by rule

$k$ : Number of classes

$p$ : Prior probability

# Stopping Criterion and Rule Pruning

- **Stopping criterion**
  - Compute the gain
  - If gain is not significant, discard the new rule

- **Rule Pruning**
  - Similar to post-pruning of decision trees
  - Reduced Error Pruning:
    - Remove one of the conjuncts in the rule
    - Compare error rate on validation set before and after pruning
    - If error improves, prune the conjunct

# Summary of Direct Method

- Grow a single rule
- Remove Instances from rule
- Prune the rule (if necessary)
- Add rule to Current Rule Set
- Repeat

# Direct Method: RIPPER

- **For 2-class problem, choose one of the classes as positive class, and the other as negative class**
  - Learn rules for positive class
  - Negative class will be default class

- **For multi-class problem**
  - Order the classes according to increasing class prevalence (fraction of instances that belong to a particular class)
  - Learn the rule set for smallest class first, treat the rest as negative class
  - Repeat with next smallest class as positive class

# Direct Method: RIPPER

- **Growing a rule:**
  - Start from empty rule
  - Add conjuncts as long as they improve FOIL's information gain
  - Stop when rule no longer covers negative examples
  - Prune the rule immediately using incremental reduced error pruning
  - Measure for pruning: $v = (p-n)/(p+n)$
    - p: number of positive examples covered by the rule in the validation set
    - n: number of negative examples covered by the rule in the validation set
  - Pruning method: delete any final sequence of conditions that maximizes v


InfoLab DGIST 대구경북과학기술원

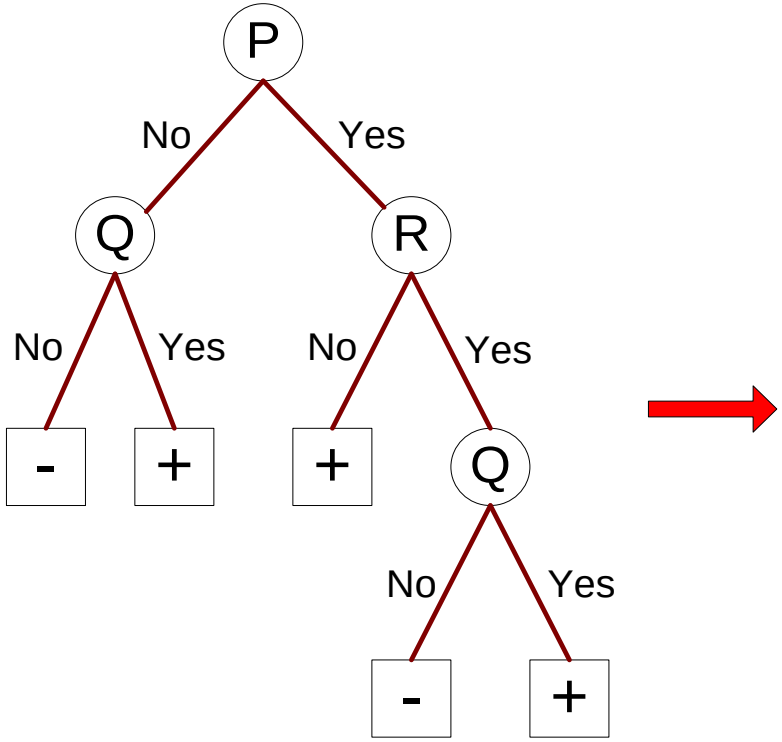
# Direct Method: RIPPER

- **Building a Rule Set:**
  - Use sequential covering algorithm
    - Finds the best rule that covers the current set of positive examples
    - Eliminate both positive and negative examples covered by the rule
  - Each time a rule is added to the rule set, compute the new description length
    - stop adding new rules when the new description length is d bits longer than the smallest description length obtained so far

# Direct Method: RIPPER

- **Optimize the rule set:**
  - For each rule *r* in the rule set ***R***
    - Consider 2 alternative rules:
      - Replacement rule (r*): grow new rule from scratch
      - Revised rule(r'): add conjuncts to extend the rule *r*
    - Compare the rule set for *r* against the rule set for r* and r'
    - Choose rule set that minimizes MDL principle
  - Repeat rule generation and rule optimization for the remaining positive examples

# Indirect Methods




**Rule Set**

r1: (P=No,Q=No) ==> -
r2: (P=No,Q=Yes) ==> +
r3: (P=Yes,R=No) ==> +
r4: (P=Yes,R=Yes,Q=No) ==> -
r5: (P=Yes,R=Yes,Q=Yes) ==> +


InfoLab DGIST 대구경북과학기술원

# Indirect Method: C4.5rules

- **Extract rules from an unpruned decision tree**

- **For each rule, r: A → y,**
  - consider an alternative rule r': A' → y where A' is obtained by removing one of the conjuncts in A
  - Compare the pessimistic error rate for r against all r's
  - Prune if one of the r's has lower pessimistic error rate
  - Repeat until we can no longer improve generalization error

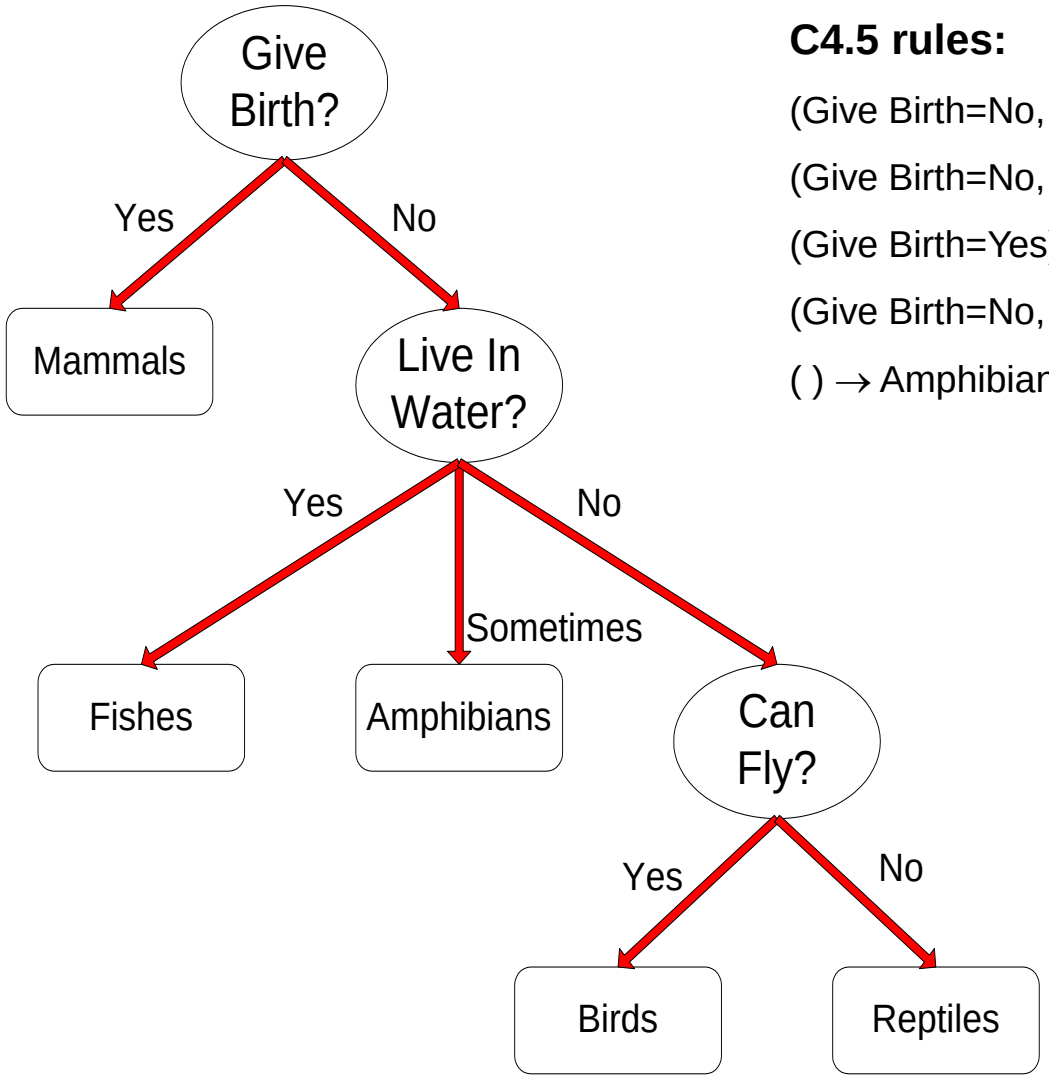
# Indirect Method: C4.5rules

- **Instead of ordering the rules, order subsets of rules (class ordering)**
  - Each subset is a collection of rules with the same rule consequent (class)
  - Compute description length of each subset
    - Description length = L(error) + g L(model)
    - g is a parameter that takes into account the presence of redundant attributes in a rule set (default value = 0.5)

# Example

| Name | Give Birth | Lay Eggs | Can Fly | Live in Water | Have Legs | Class |
|---|---|---|---|---|---|---|
| human | yes | no | no | no | yes | mammals |
| python | no | yes | no | no | no | reptiles |
| salmon | no | yes | no | yes | no | fishes |
| whale | yes | no | no | yes | no | mammals |
| frog | no | yes | no | sometimes | yes | amphibians |
| komodo | no | yes | no | no | yes | reptiles |
| bat | yes | no | yes | no | yes | mammals |
| pigeon | no | yes | yes | no | yes | birds |
| cat | yes | no | no | no | yes | mammals |
| leopard shark | yes | no | no | yes | no | fishes |
| turtle | no | yes | no | sometimes | yes | reptiles |
| penguin | no | yes | no | sometimes | yes | birds |
| porcupine | yes | no | no | no | yes | mammals |
| eel | no | yes | no | yes | no | fishes |
| salamander | no | yes | no | sometimes | yes | amphibians |
| gila monster | no | yes | no | no | yes | reptiles |
| platypus | no | yes | no | no | yes | mammals |
| owl | no | yes | yes | no | yes | birds |
| dolphin | yes | no | no | yes | no | mammals |
| eagle | no | yes | yes | no | yes | birds |

# C4.5 versus C4.5 rules versus RIPPER




**C4.5 rules:**

(Give Birth=No, Can Fly=Yes) → Birds

(Give Birth=No, Live in Water=Yes) → Fishes

(Give Birth=Yes) → Mammals

(Give Birth=No, Can Fly=No, Live in Water=No) → Reptiles

( ) → Amphibians

**RIPPER:**

(Live in Water=Yes) → Fishes

(Have Legs=No) → Reptiles

(Give Birth=No, Can Fly=No, Live In Water=No) → Reptiles

(Can Fly=Yes,Give Birth=No) → Birds

() → Mammals


InfoLab DGIST 대구경북과학기술원

# C4.5 versus C4.5 rules versus RIPPER

## C4.5 and C4.5 rules:

| | | | PREDICTED CLASS | | | |
|---|---|---|---|---|---|---|
| | | Amphibians | Fishes | Reptiles | Birds | Mammals |
| ACTUAL | Amphibians | 2 | 0 | 0 | 0 | 0 |
| CLASS | Fishes | 0 | 2 | 0 | 0 | 1 |
| | Reptiles | 1 | 0 | 3 | 0 | 0 |
| | Birds | 1 | 0 | 0 | 3 | 0 |
| | Mammals | 0 | 0 | 1 | 0 | 6 |

## RIPPER:

| | | | PREDICTED CLASS | | | |
|---|---|---|---|---|---|---|
| | | Amphibians | Fishes | Reptiles | Birds | Mammals |
| ACTUAL | Amphibians | 0 | 0 | 0 | 0 | 2 |
| CLASS | Fishes | 0 | 3 | 0 | 0 | 0 |
| | Reptiles | 0 | 0 | 3 | 0 | 1 |
| | Birds | 0 | 0 | 1 | 2 | 1 |
| | Mammals | 0 | 2 | 1 | 0 | 4 |

# Advantages of Rule-Based Classifiers

- As highly expressive as decision trees
- Easy to interpret
- Easy to generate
- Can classify new instances rapidly
- Performance comparable to decision trees

# Instance-Based Classifiers

Set of Stored Cases

| Atr1 | ……. | AtrN | Class |
|---|---|---|---|
| | | | A |
| | | | B |
| | | | B |
| | | | C |
| | | | A |
| | | | C |
| | | | B |

- **Store the training records**
- **Use training records to predict the class label of unseen cases**

Unseen Case

| Atr1 | ……. | AtrN |
|---|---|---|
| | | |

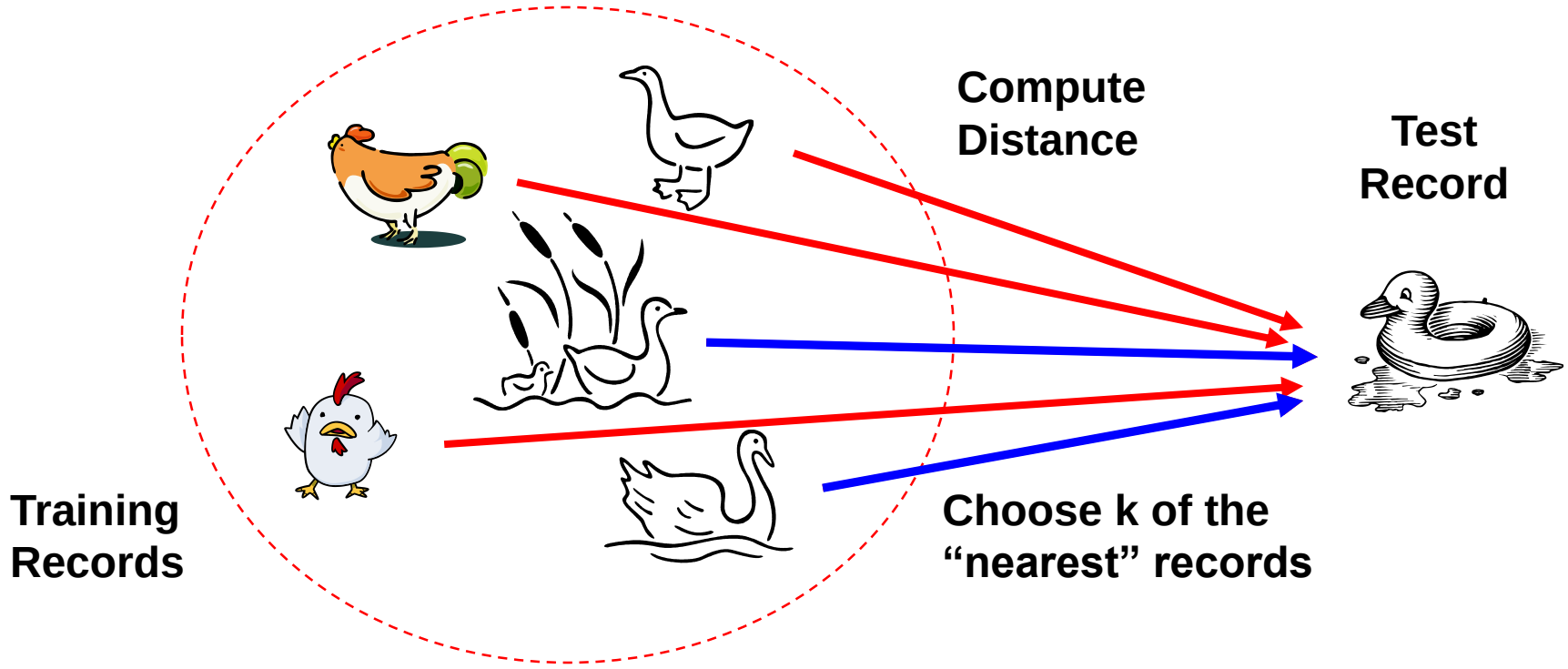

InfoLab DGIST 대구경북과학기술원

# Instance Based Classifiers

- **Examples:**
  - Rote-learner
    - Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
  - Nearest neighbor
    - Uses k “closest” points (nearest neighbors) for performing classification

# Nearest Neighbor Classifiers

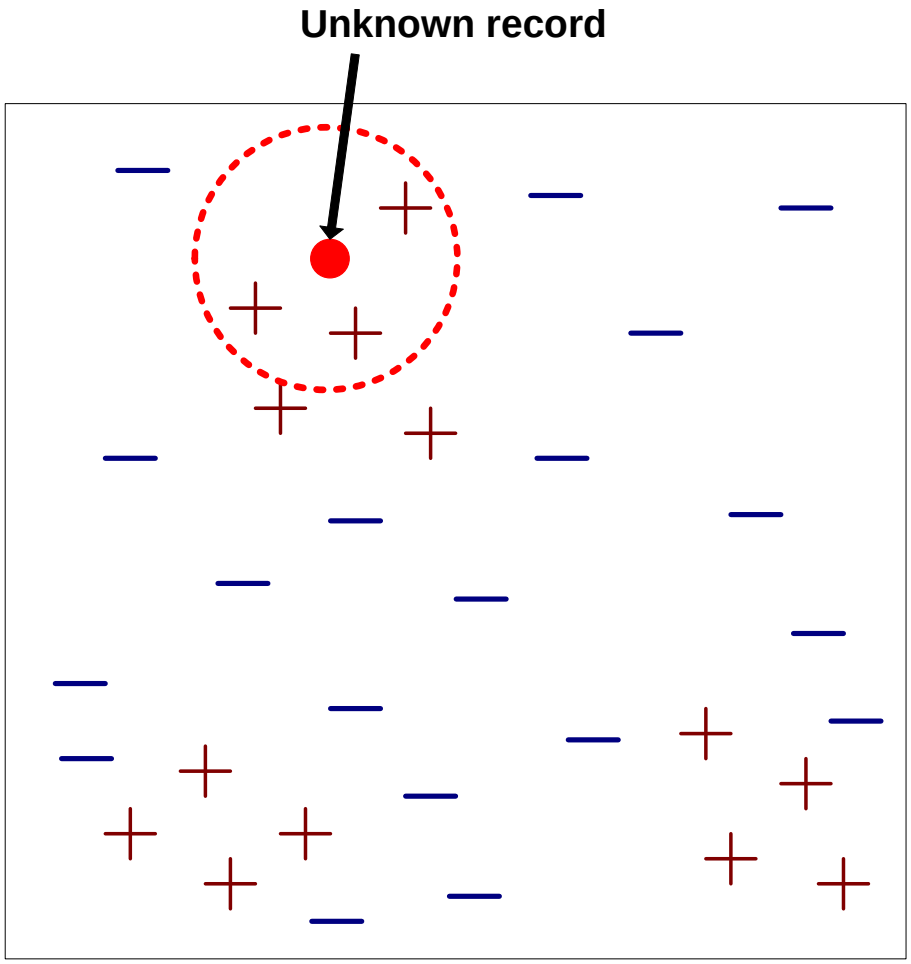
- **Basic idea:**
  - If it walks like a duck, quacks like a duck, then it's probably a duck





InfoLab DGIST 대구경북과학기술원

# Nearest-Neighbor Classifiers

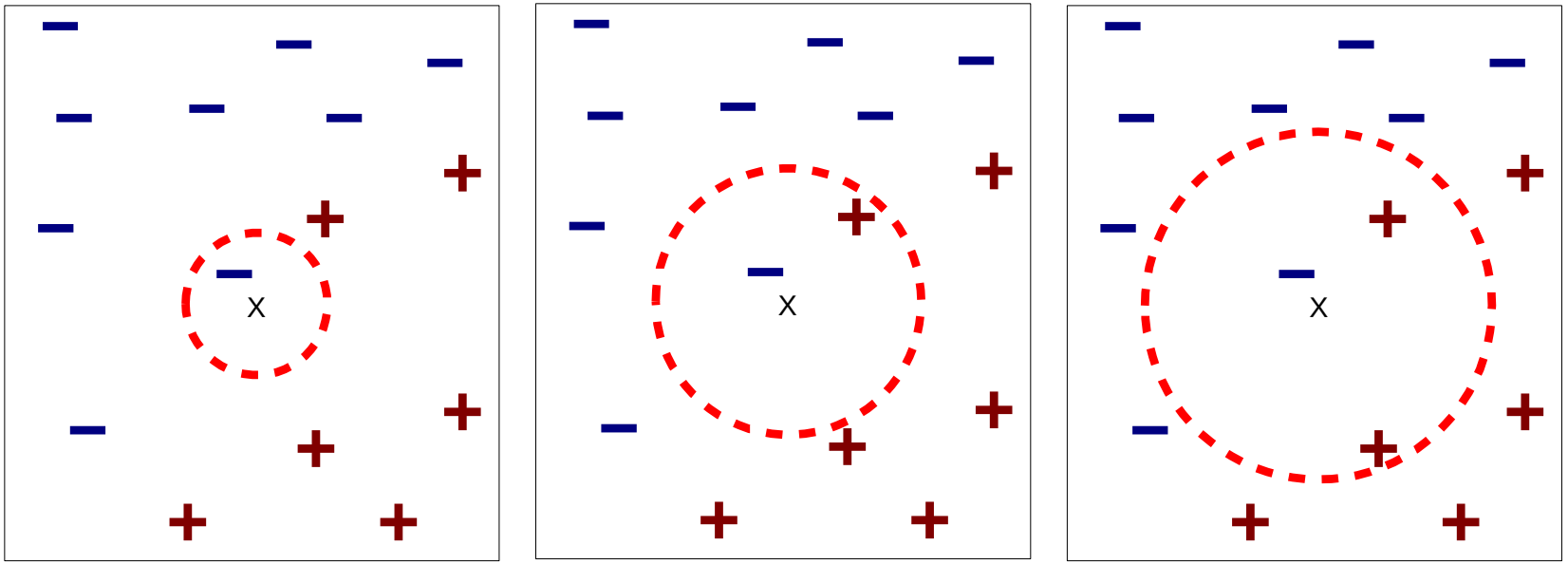



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve

- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)


InfoLab DGIST 대구경북과학기술원

# Definition of Nearest Neighbor

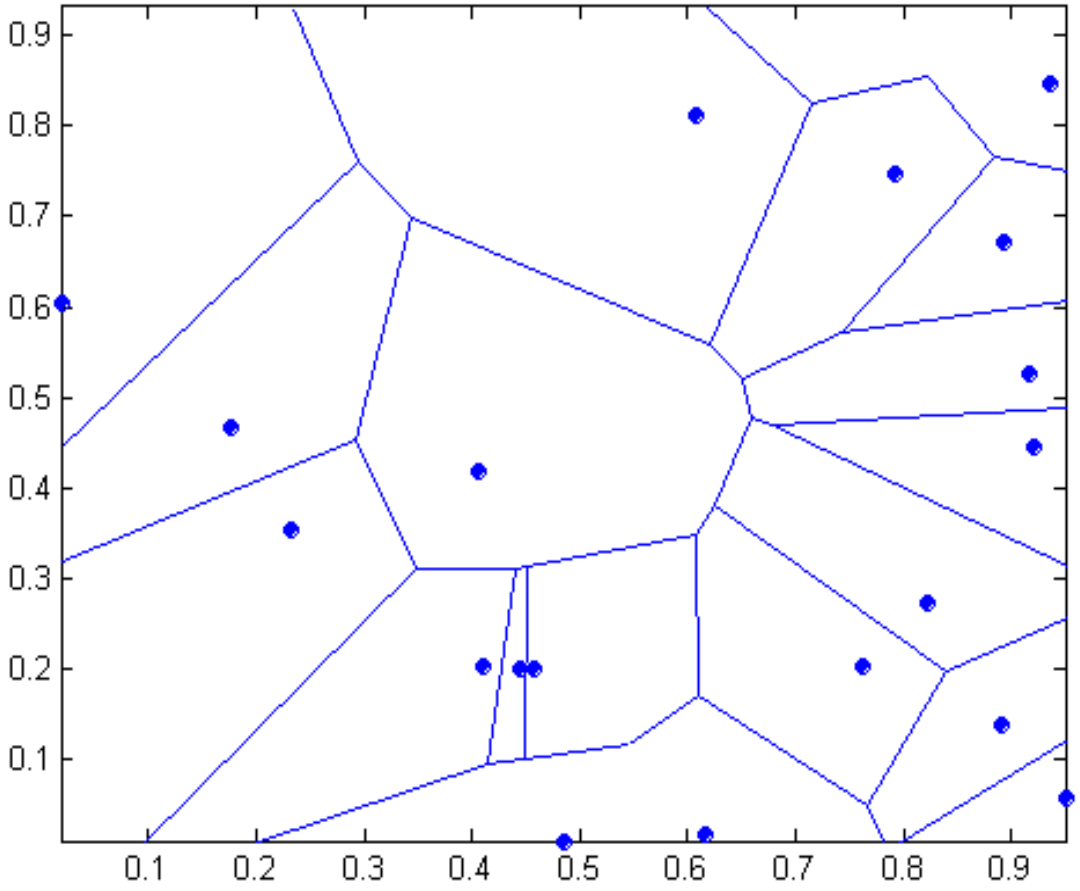



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

# 1 nearest-neighbor

Voronoi Diagram





InfoLab DGIST 대구경북과학기술원

# Nearest Neighbor Classification

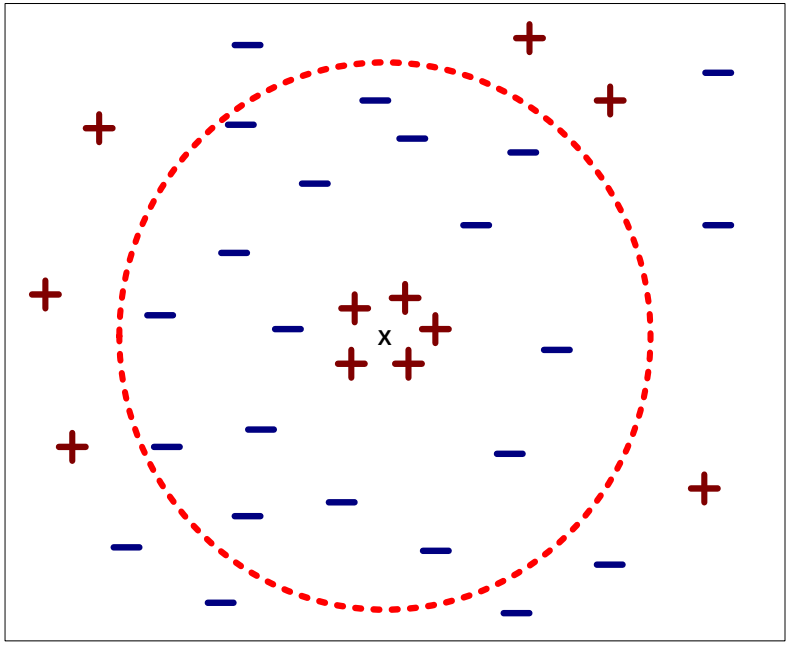
- **Compute distance between two points:**
  - Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- **Determine the class from nearest neighbor list**
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor, $w = 1/d^2$

# Nearest Neighbor Classification…

- **Choosing the value of k:**
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes



# Nearest Neighbor Classification…

- **Scaling issues**
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
  - Example:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 90lb to 300lb
    - income of a person may vary from $10K to $1M

# Nearest Neighbor Classification…

- **Problem with Euclidean measure:**
  - High dimensional data
    - curse of dimensionality
  - Can produce counter-intuitive results

| | | | |
|---|---|---|---|
| 1 1 1 1 1 1 1 1 1 1 1 0 | | 1 0 0 0 0 0 0 0 0 0 0 0 |
| 0 1 1 1 1 1 1 1 1 1 1 1 | vs | 0 0 0 0 0 0 0 0 0 0 0 1 |
| d = 1.4142 | | d = 1.4142 |

- **Solution**: Normalize the vectors to unit length

# Nearest neighbor Classification…

- **k-NN classifiers are lazy learners**
  - It does not build models explicitly
  - Unlike eager learners such as decision tree induction and rule-based systems
  - Classifying unknown records are relatively expensive

# Example: PEBLS

- **PEBLS: Parallel Examplar-Based Learning System (Cost & Salzberg)**
  - Works with both continuous and nominal features
    - For nominal features, distance between two nominal values is computed using modified value difference metric (MVDM)
  - Each record is assigned a weight factor
  - Number of nearest neighbor, k = 1

# Example: PEBLS

| 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 |

Distance between nominal attribute values:

d(Single,Married)

= | 2/4 – 0/4 | + | 2/4 – 4/4 | = 1

d(Single,Divorced)

= | 2/4 – 1/2 | + | 2/4 – 1/2 | = 0

d(Married,Divorced)

= | 0/4 – 1/2 | + | 4/4 – 1/2 | = 1

d(Refund=Yes,Refund=No)

= | 0/3 – 3/7 | + | 3/3 – 4/7 | = 6/7

| Class | Marital Status | | |
|---|---|---|---|
| | Single | Married | Divorced |
| Yes | 2 | 0 | 1 |
| No | 2 | 4 | 1 |

| Class | Refund | |
|---|---|---|
| | Yes | No |
| Yes | 0 | 3 |
| No | 3 | 4 |

$$d(V_1, V_2) = \sum_i \left| \frac{n_{1i}}{n_1} - \frac{n_{2i}}{n_2} \right|$$

# Example: PEBLS

| *Tid* | Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|---|
| X | Yes | Single | 125K | **No** |
| Y | No | Married | 100K | **No** |

Distance between record X and record Y:

$$\Delta(X,Y) = w_X w_Y \sum_{i=1}^{d} d(X_i, Y_i)^2$$

where:

$$w_X = \frac{\text{Number of times X is used for prediction}}{\text{Number of times X predicts correctly}}$$

$w_X \cong 1$ if X makes accurate prediction most of the time

$w_X > 1$ if X is not reliable for making predictions


InfoLab DGIST 대구경북과학기술원

# 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)**


InfoLab DGIST 대구경북과학기술원

# 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): Cost of misclassifying class j example as class i

# 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


InfoLab DGIST 대구경북과학기술원

# Cost vs Accuracy

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

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

$N = a + b + c + d$

$Accuracy = (a + d)/N$

$$\begin{aligned} Cost &= p\,(a + d) + q\,(b + c) \\ &= p\,(a + d) + q\,(N - a - d) \\ &= q\,N - (q - p)(a + d) \\ &= N\,[q - (q-p) \times Accuracy] \end{aligned}$$

# Cost-Sensitive Measures

$$\text{Precision (p)} = \frac{a}{a+c}$$

$$\text{Recall (r)} = \frac{a}{a+b}$$

$$\text{F-measure (F)} = \frac{2rp}{r+p} = \frac{2a}{2a+b+c}$$

- Precision is biased towards C(Yes|Yes) & C(Yes|No)
- Recall is biased towards C(Yes|Yes) & C(No|Yes)
- F-measure is biased towards all except C(No|No)

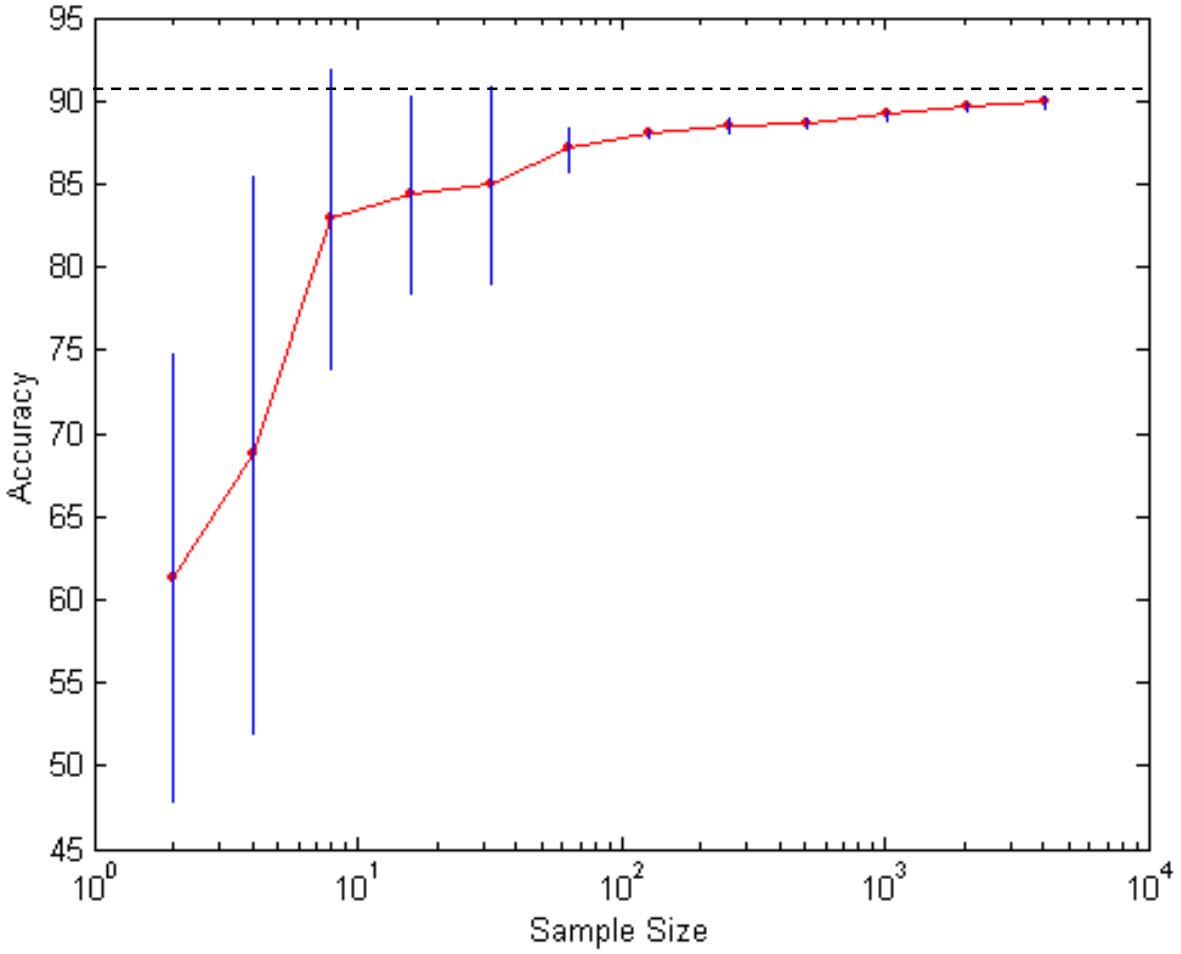
$$\text{Weighted Accuracy} = \frac{w_1 a + w_4 d}{w_1 a + w_2 b + w_3 c + w_4 d}$$

# Methods for Performance Evaluation

- **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

# Learning Curve




- 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


InfoLab DGIST 대구경북과학기술원

# Methods of Estimation

- **Holdout**
  - Reserve 2/3 for training and 1/3 for testing
- **Random subsampling**
  - Repeated holdout
- **Cross validation**
  - Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Leave-one-out: k=n
- **Stratified sampling**
  - oversampling vs undersampling
- **Bootstrap**
  - Sampling with replacement

# Holdout & Cross-Validation Methods

- **Holdout method**
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - Random sampling: a variation of holdout
    - Repeat holdout k times, accuracy = avg. of the accuracies obtained
- **Cross-validation** (*k*-fold, where k = 10 is most popular)
  - Randomly partition the data into *k mutually exclusive* subsets, each approximately equal size
  - At *i*-th iteration, use $D_i$ as test set and others as training set
  - Leave-one-out: *k* folds where *k* = # of tuples, for small sized data
  - ***Stratified cross-validation***: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data

# Bootstrap

- **Bootstrap**
  - Works well with small data sets
  - Samples the given training tuples uniformly *with replacement*
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
- Several bootstrap methods, and a common one is **.632 boostrap**
  - A data set with *d* tuples is sampled *d* times, with replacement, resulting in a training set of *d* samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data end up in the bootstrap, and the remaining 36.8% form the test set (since $(1 - 1/d)^d \approx e^{-1} = 0.368$)
  - Repeat the sampling procedure *k* times, overall accuracy of the model:

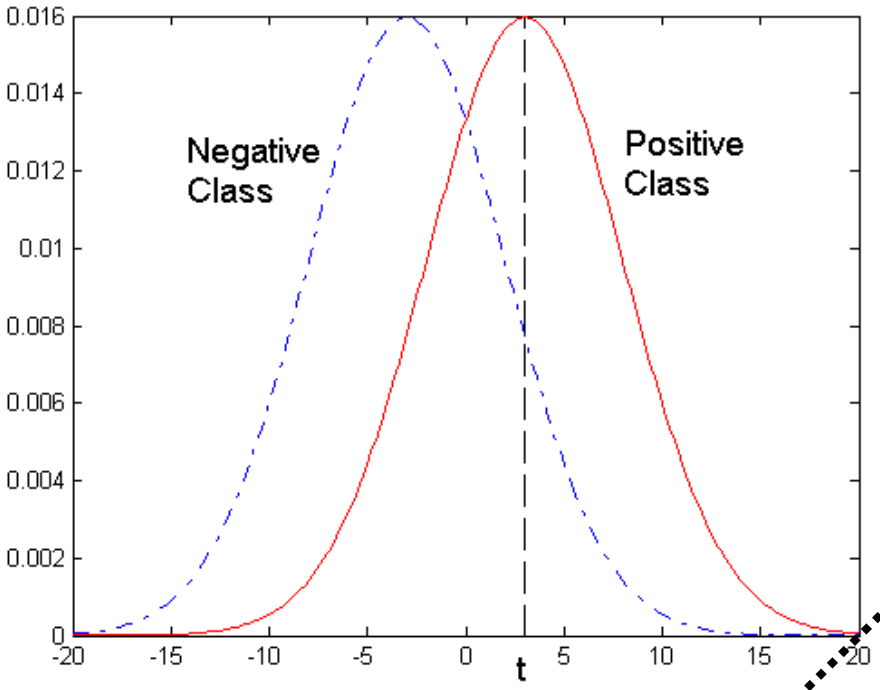
$$Acc(M) = \frac{1}{k}\sum_{i=1}^{k}(0.632 \times Acc(M_i)_{test_set} + 0.368 \times Acc(M_i)_{train_set})$$

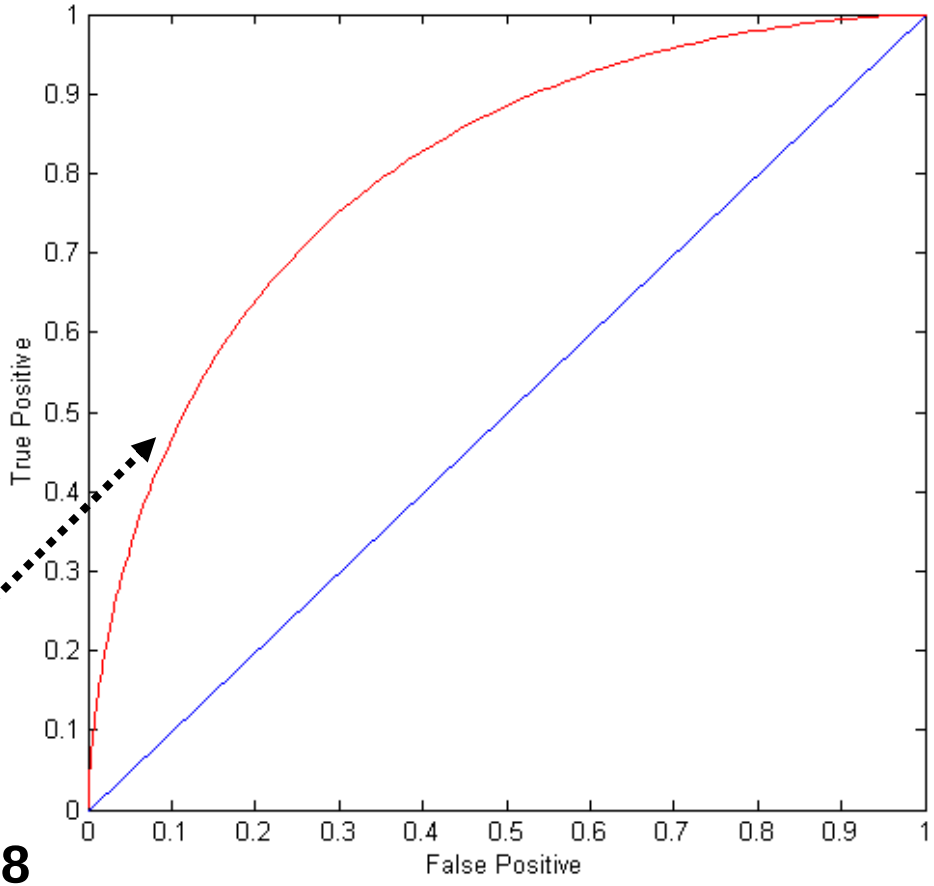
# ROC (Receiver Operating Characteristic)

- **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, TN=0.88**

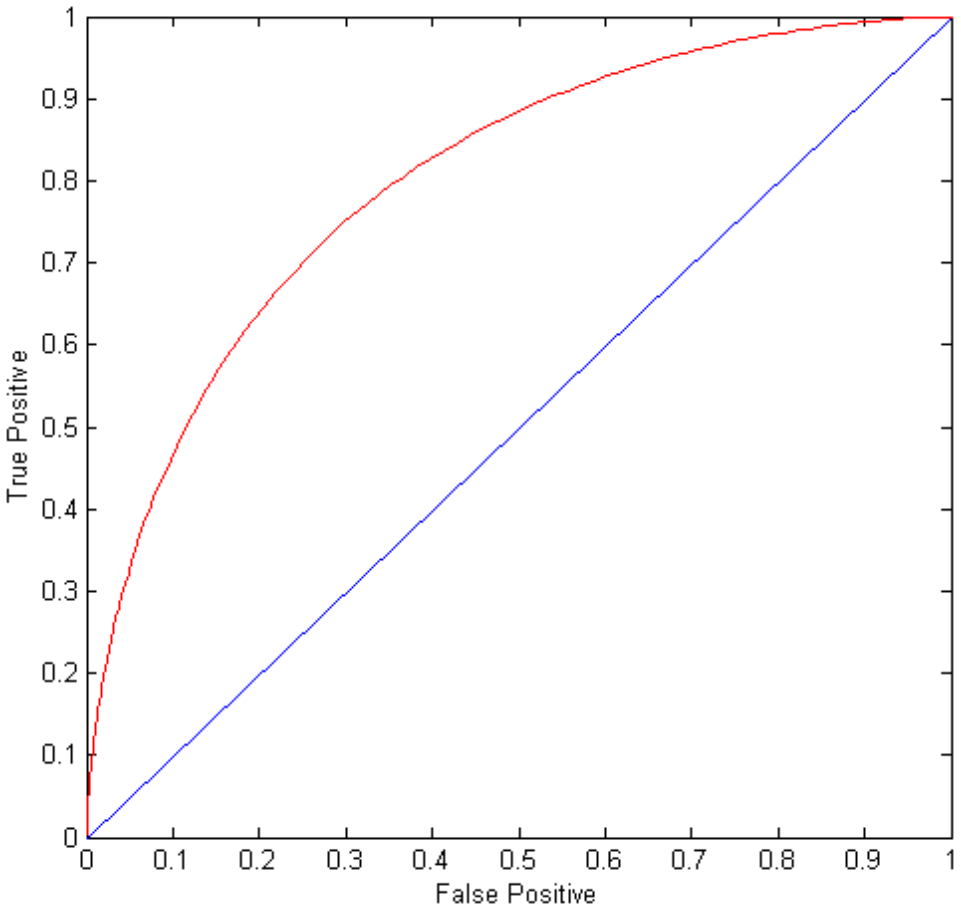

InfoLab DGIST 대구경북과학기술원

# ROC Curve

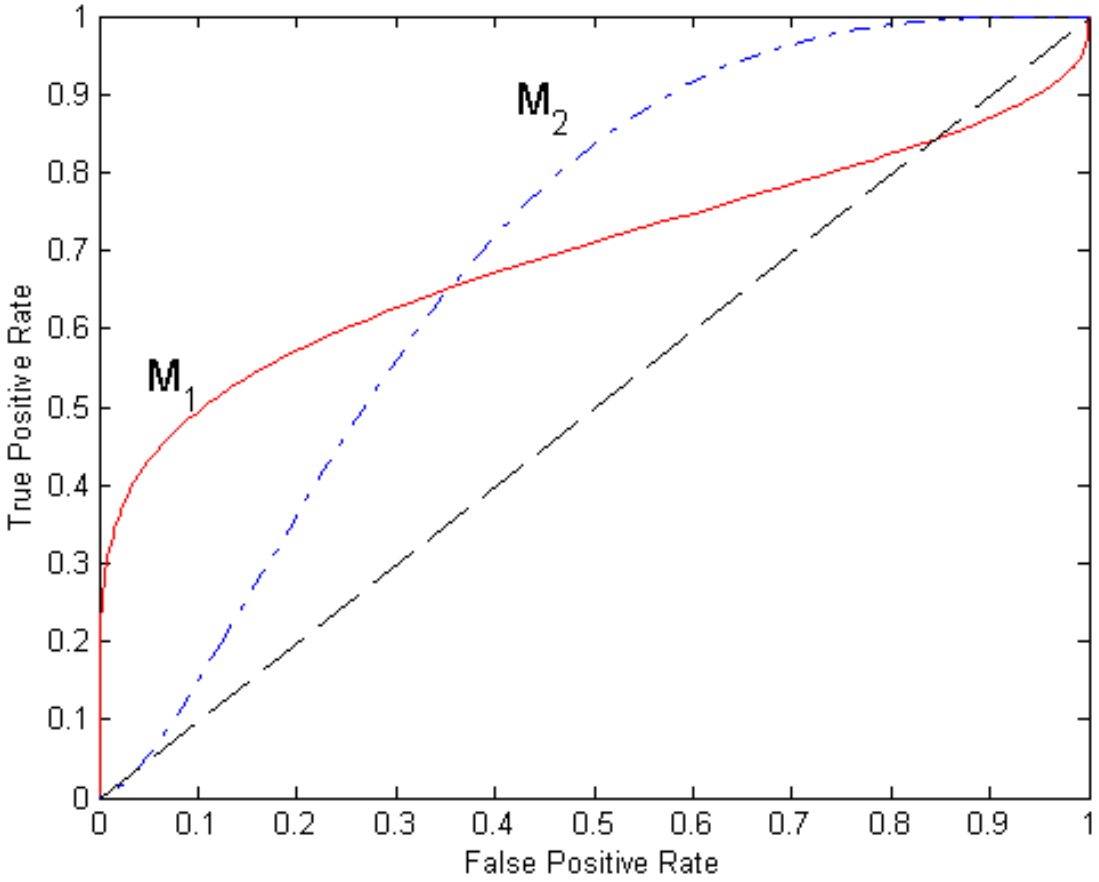
**(TP,FP):**

- **(0,0): declare everything to be negative clas**
- **(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



# 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
  - Ideal:
    - Area = 1
  - Random guess:
    - Area = 0.5


InfoLab DGIST 대구경북과학기술원

# How to Construct an ROC curve

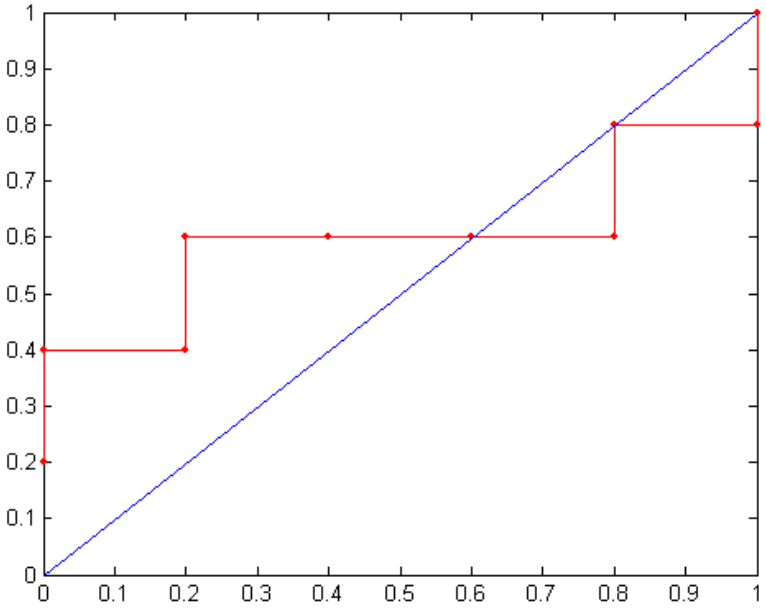
| Instance | P(+\|A) | True Class |
|---|---|---|
| 1 | 0.95 | + |
| 2 | 0.93 | + |
| 3 | 0.87 | - |
| 4 | 0.85 | - |
| 5 | 0.85 | - |
| 6 | 0.85 | + |
| 7 | 0.76 | - |
| 8 | 0.53 | + |
| 9 | 0.43 | - |
| 10 | 0.25 | + |

- 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:**





InfoLab DGIST 대구경북과학기술원

# Test of Significance

- **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?**
  - How much confidence can we place on accuracy of M1 and M2?
  - Can the difference in performance measure be explained as a result of random fluctuations in the test set?

# Confidence Interval for Accuracy

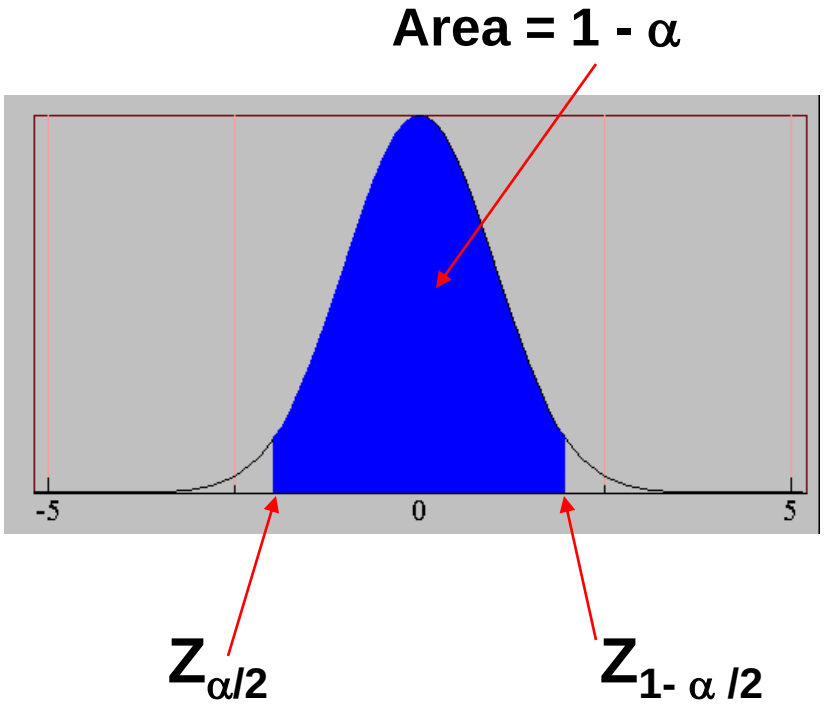
- **Prediction can be regarded as a Bernoulli trial**
  - A Bernoulli trial has 2 possible outcomes
  - Possible outcomes for prediction: correct or wrong
  - Collection of Bernoulli trials has a Binomial distribution:
    - $x \sim Bin(N, p)$ x: number of correct predictions
    - e.g: Toss a fair coin 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) or equivalently, acc=x/N, and N (# of test instances),**

  **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:**

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


InfoLab DGIST 대구경북과학기술원

# 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, $d_t = d \pm Z_{\alpha/2}\hat{\sigma}_t$


InfoLab DGIST 대구경북과학기술원

# 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**

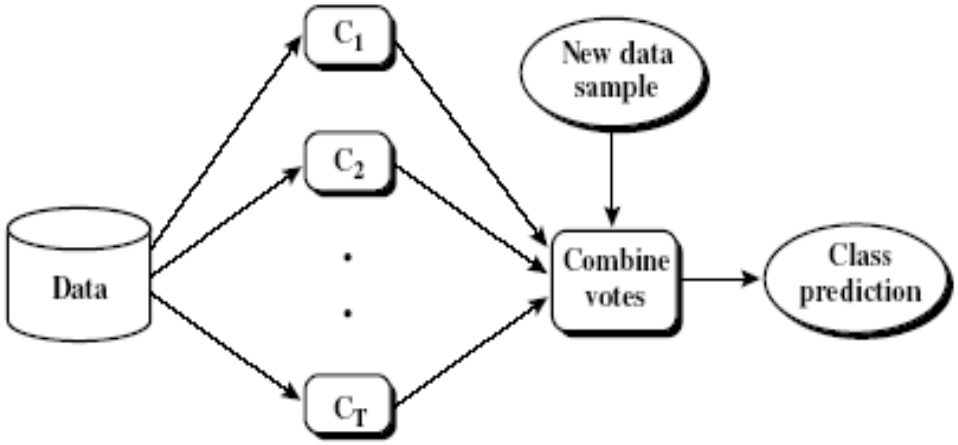
$$d_t = 0.100 \pm 1.96 \times \sqrt{0.0043} = 0.100 \pm 0.128$$

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

# Issues Affecting Model Selection

- **Accuracy**
  - classifier accuracy: predicting class label
- **Speed**
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- **Robustness**: handling noise and missing values
- **Scalability**: efficiency in disk-resident databases
- **Interpretability**
  - understanding and insight provided by the model
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules

# Ensemble Methods: Increasing the Accuracy




- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of k learned models, $M_1$, $M_2$, …, $M_k$, with the aim of creating an improved model M*
- Popular ensemble methods
  - Bagging: averaging the prediction over a collection of classifiers
  - Boosting: weighted vote with a collection of classifiers
  - Ensemble: combining a set of heterogeneous classifiers

# Bagging: Boostrap Aggregation

- Analogy: Diagnosis based on multiple doctors' majority vote
- Training
  - Given a set D of *d* tuples, at each iteration *i*, a training set $D_i$ of *d* tuples is sampled with replacement from D (i.e., bootstrap)
  - A classifier model $M_i$ is learned for each training set $D_i$
- Classification: classify an unknown sample **X**
  - Each classifier $M_i$ returns its class prediction
  - The bagged classifier M* counts the votes and assigns the class with the most votes to **X**
- Prediction: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- Accuracy
  - Often significantly better than a single classifier derived from D
  - For noise data: not considerably worse, more robust
  - Proved improved accuracy in prediction

# Boosting

- Analogy: Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy
- How boosting works?
  - **Weights** are assigned to each training tuple
  - A series of k classifiers is iteratively learned
  - After a classifier $M_i$ is learned, the weights are updated to allow the subsequent classifier, $M_{i+1}$, to **pay more attention to the training tuples that were misclassified** by $M_i$
  - The final **M* combines the votes** of each individual classifier, where the weight of each classifier's vote is a function of its accuracy
- Boosting algorithm can be extended for numeric prediction
- Comparing with bagging: Boosting tends to have greater accuracy, but it also risks overfitting the model to misclassified data

# Adaboost (Freund and Schapire, 1997)

- Given a set of *d* class-labeled tuples, ($\mathbf{X_1}$, $y_1$), …, ($\mathbf{X_d}$, $y_d$)
- Initially, all the weights of tuples are set the same (1/d)
- Generate k classifiers in k rounds. At round i,
  - Tuples from D are sampled (with replacement) to form a training set $D_i$ of the same size
  - Each tuple's chance of being selected is based on its weight
  - A classification model $M_i$ is derived from $D_i$
  - Its error rate is calculated using $D_i$ as a test set
  - If a tuple is misclassified, its weight is increased, o.w. it is decreased
- Error rate: err($\mathbf{X_j}$) is the misclassification error of tuple $\mathbf{X_j}$. Classifier $M_i$ error rate is the sum of the weights of the misclassified tuples:

$$error(M_i) = \sum_{j}^{d} w_j \times err(\mathbf{X_j})$$

- The weight of classifier $M_i$'s vote is $\log \frac{1-error(M_i)}{error(M_i)}$

# Random Forest (Breiman 2001)

- Random Forest:
  - Each classifier in the ensemble is a *decision tree* classifier and is generated using a random selection of attributes at each node to determine the split
  - During classification, each tree votes and the most popular class is returned
- Two Methods to construct Random Forest:
  - Forest-RI (*random input selection*): Randomly select, at each node, F attributes as candidates for the split at the node. The CART methodology is used to grow the trees to maximum size
  - Forest-RC (*random linear combinations*): Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)
- Comparable in accuracy to Adaboost, but more robust to errors and outliers
- Insensitive to the number of attributes selected for consideration at each split, and faster than bagging or boosting

# Classification of Class-Imbalanced Data Sets

- Class-imbalance problem: Rare positive example but numerous negative ones, e.g., medical diagnosis, fraud, oil-spill, fault, etc.
- Traditional methods assume a balanced distribution of classes and equal error costs: not suitable for class-imbalanced data
- Typical methods for imbalance data in 2-class classification:
  - **Oversampling**: re-sampling of data from positive class
  - **Under-sampling**: randomly eliminate tuples from negative class
  - **Threshold-moving**: moves the decision threshold, t, so that the rare class tuples are easier to classify, and hence, less chance of costly false negative errors
  - Ensemble techniques: Ensemble multiple classifiers introduced above
- Still difficult for class imbalance problem on multiclass tasks

# Summary (I)

- Classification is a form of data analysis that extracts models describing important data classes.
- Effective and scalable methods have been developed for decision tree induction, Naive Bayesian classification, rule-based classification, and many other classification methods.
- Evaluation metrics include: accuracy, sensitivity, specificity, precision, recall, *F* measure, and $F_\beta$ measure.
- Stratified k-fold cross-validation is recommended for accuracy estimation. Bagging and boosting can be used to increase overall accuracy by learning and combining a series of individual models.

# Summary (II)

- Significance tests and ROC curves are useful for model selection.
- There have been numerous comparisons of the different classification methods; the matter remains a research topic
- No single method has been found to be superior over all others for all data sets
- Issues such as accuracy, training time, robustness, scalability, and interpretability must be considered and can involve trade-offs, further complicating the quest for an overall superior method

InfoLab

# Thank you!

DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology