

# **Data Mining Classification: Basic Concepts and Techniques**

---

## Lecture Notes for Chapter 3

Introduction to Data Mining, 2<sup>nd</sup> Edition

by

Tan, Steinbach, Karpatne, Kumar

# Classification: Definition

---

- Given a collection of records (training set )
  - Each record is by characterized by a tuple  $(x,y)$ , where  $x$  is the attribute set and  $y$  is the class label
    - ◆  $x$ : attribute, predictor, independent variable, input
    - ◆  $y$ : class, response, dependent variable, output
- Task:
  - Learn a model that maps each attribute set  $x$  into one of the predefined class labels  $y$

# Examples of Classification Task

Task	Attribute set, $x$	Class label, $y$
Categorizing email messages	Features extracted from email message header and content	spam or non-spam
Identifying tumor cells	Features extracted from MRI scans	malignant or benign cells
Cataloging galaxies	Features extracted from telescope images	Elliptical, spiral, or irregular-shaped galaxies

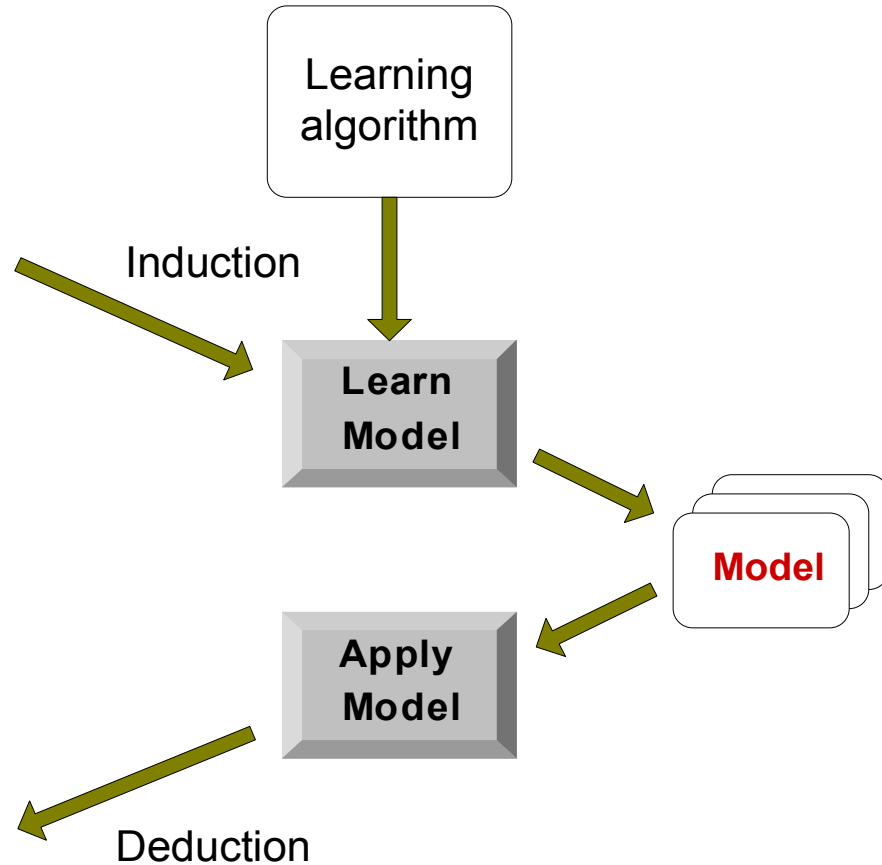
# General Approach for Building Classification Model

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



# Classification Techniques

---

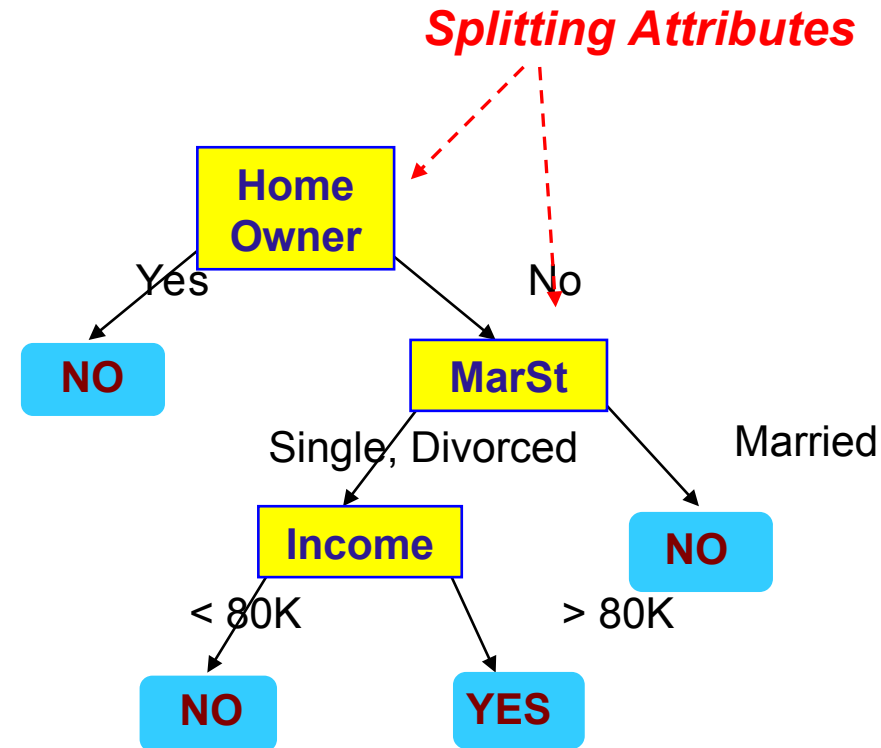
- Base Classifiers
  - Decision Tree based Methods
  - Rule-based Methods
  - Nearest-neighbor
  - Neural Networks
  - Deep Learning
  - Naïve Bayes and Bayesian Belief Networks
  - Support Vector Machines
- Ensemble Classifiers
  - Boosting, Bagging, Random Forests

# Example of a Decision Tree

categorical  
categorical  
continuous  
class

ID	Home Owner	Marital Status	Annual 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

Training Data

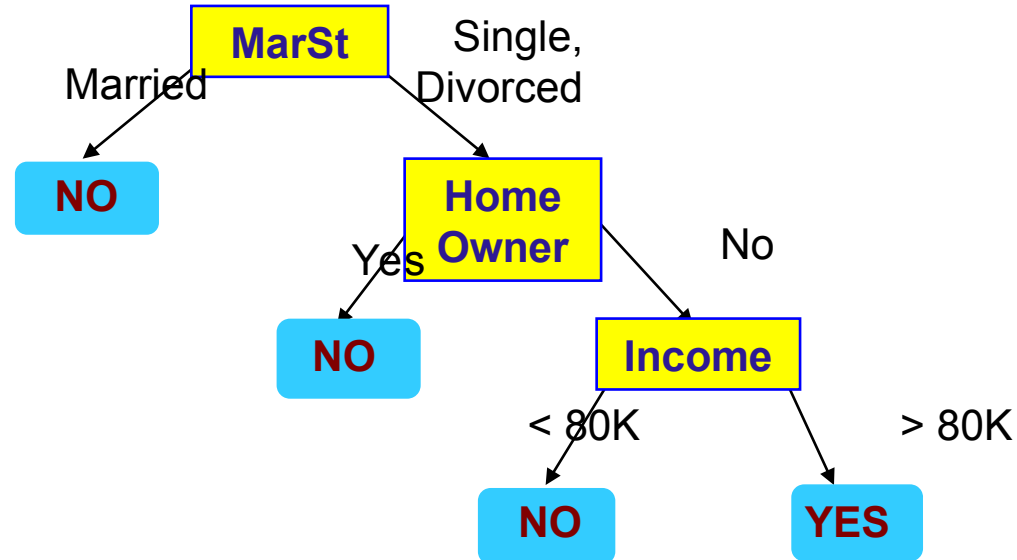


Model: Decision Tree

# Another Example of Decision Tree

*categorical*  
*categorical*  
*continuous*  
*class*

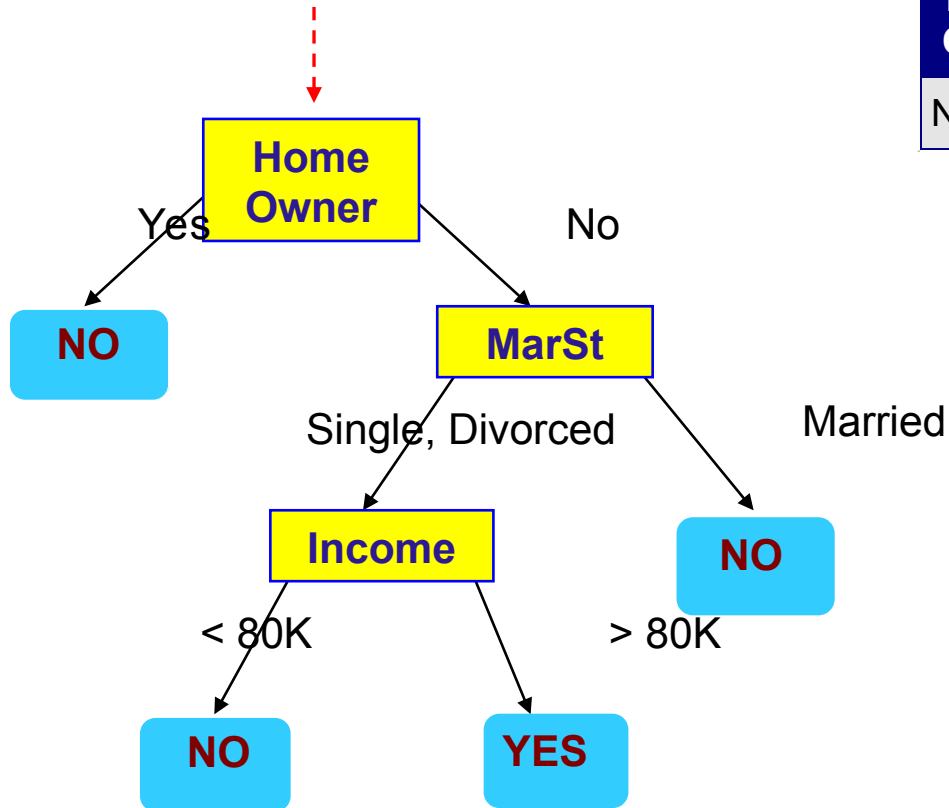
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



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

# Apply Model to Test Data

Start from the root of tree.



## Test Data

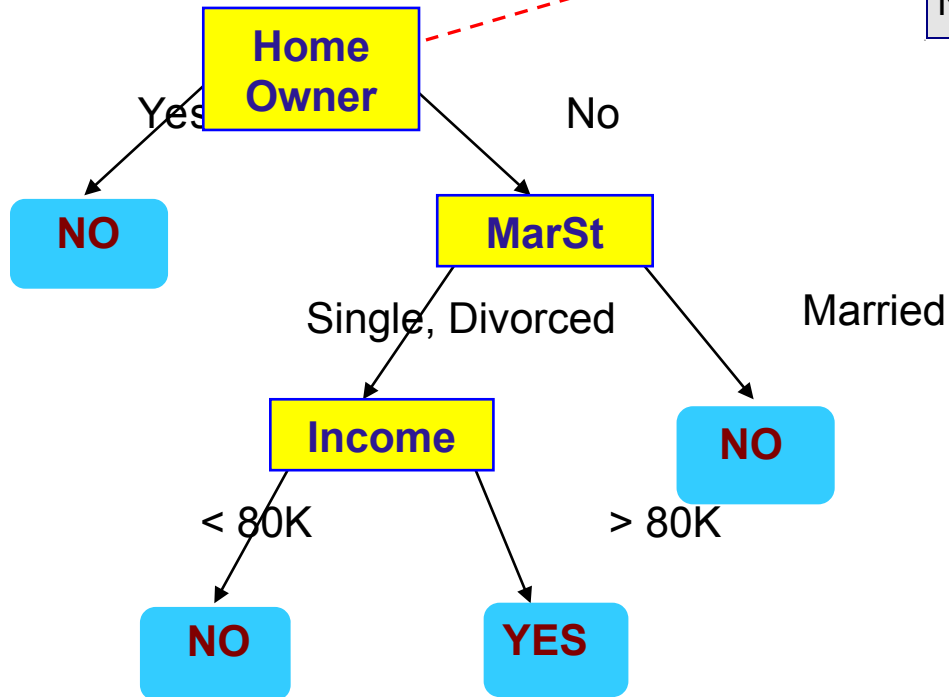
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



# Apply Model to Test Data

## Test Data

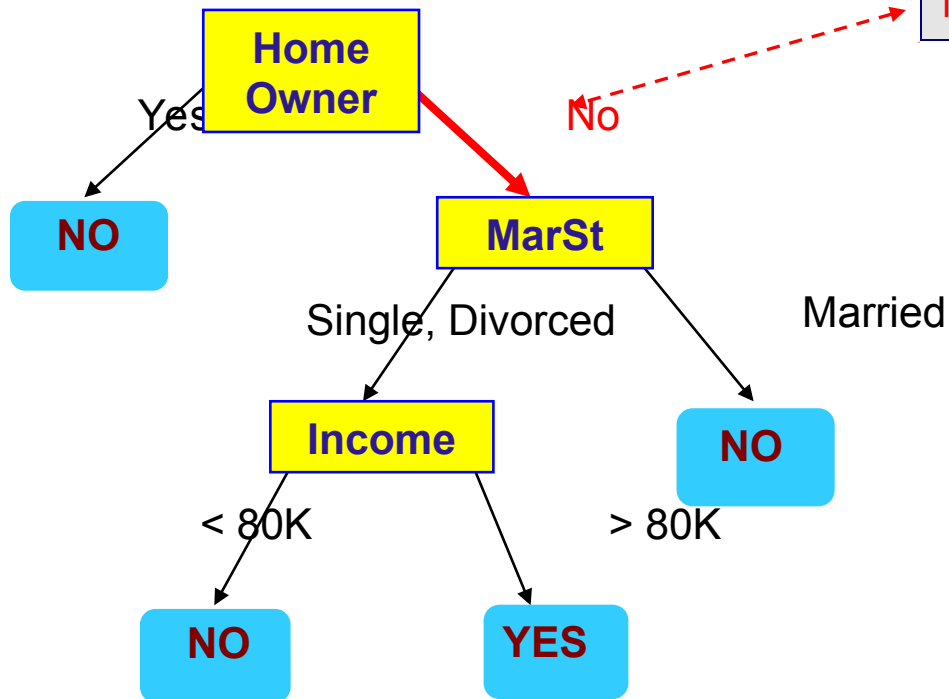
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



# Apply Model to Test Data

## Test Data

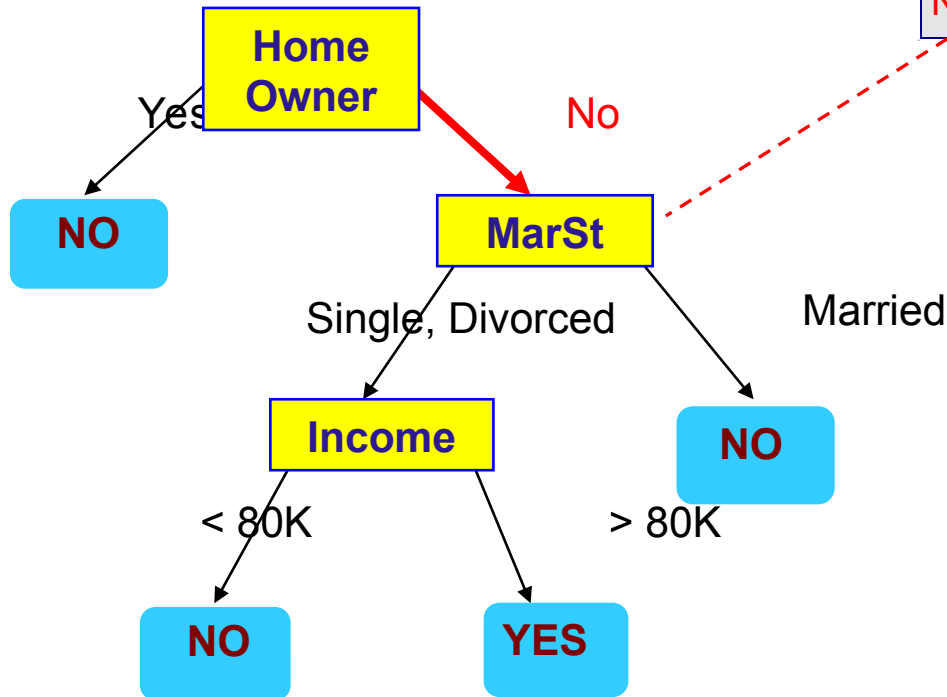
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



# Apply Model to Test Data

## Test Data

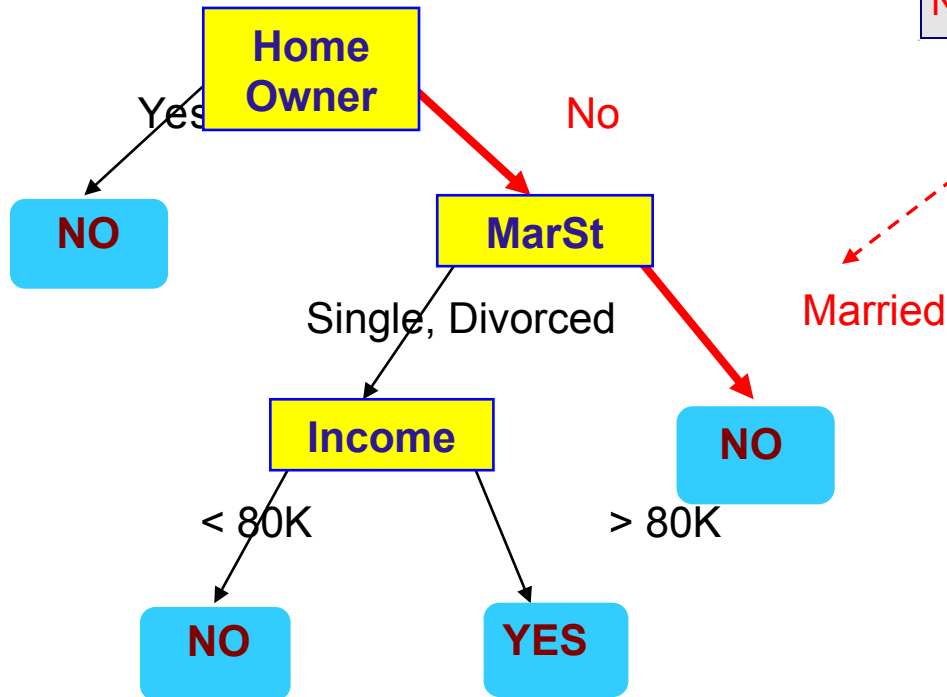
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



# Apply Model to Test Data

## Test Data

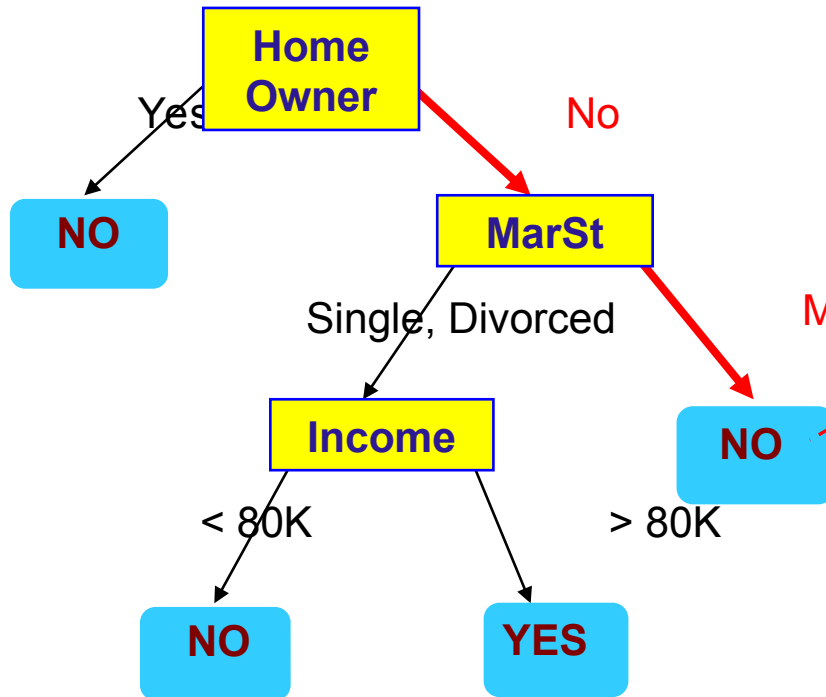
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



# Apply Model to Test Data

## Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Married

Assign Defaulted to  
"No"

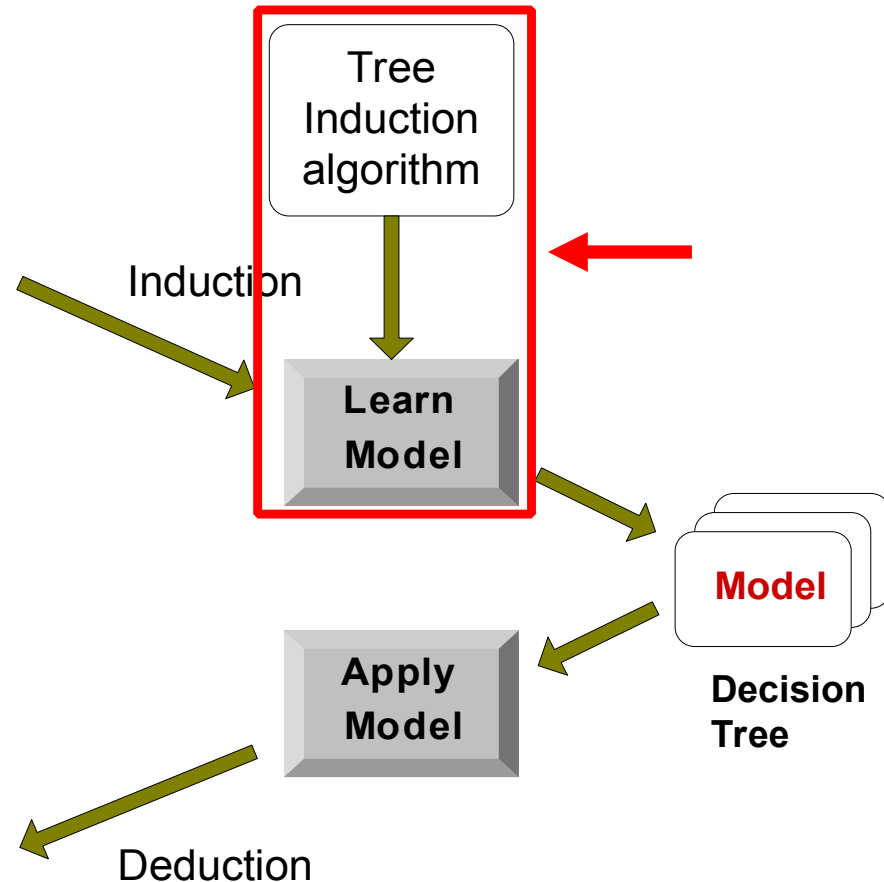
# Decision Tree Classification Task

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

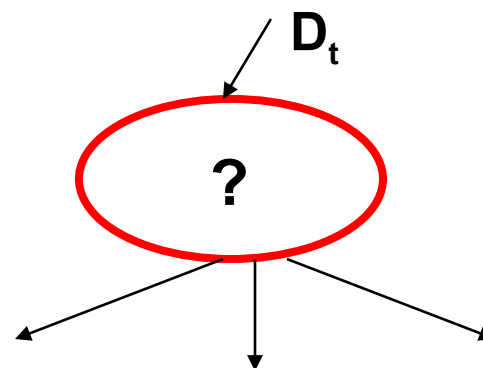
---

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

# General Structure of Decision Tree 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  $D_t$  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.

ID	Home Owner	Marital Status	Annual 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





# Hunt's Algorithm

Defaulted = No

(7,3)

(a)

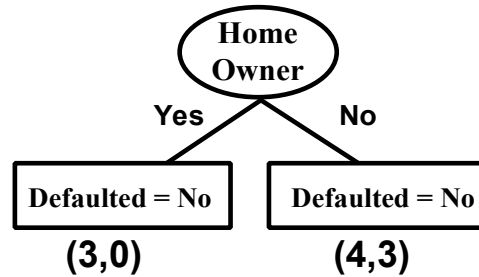
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

# Hunt's Algorithm

Defaulted = No

(7,3)

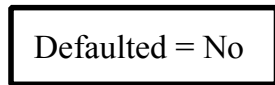
(a)



(b)

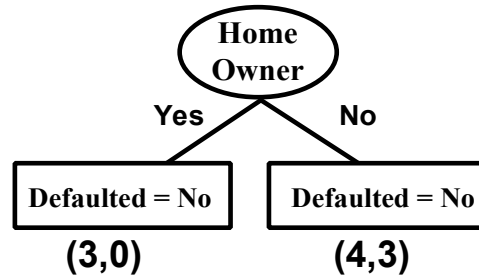
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

# Hunt's Algorithm



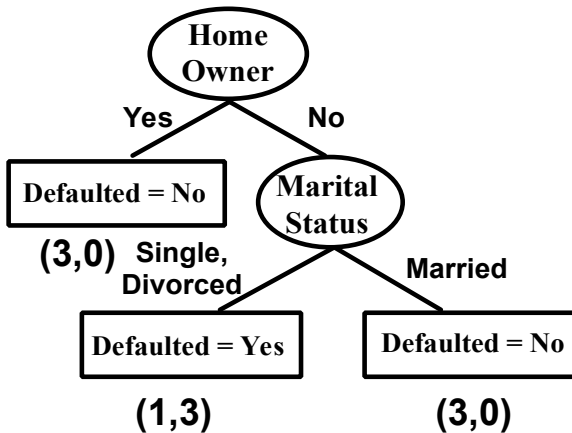
(7,3)

(a)



(b)

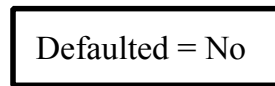
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



(c)

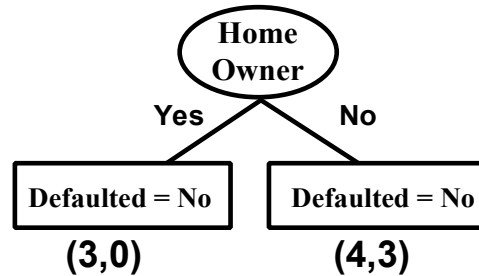
# Hunt's Algorithm

ID	Home Owner	Marital Status	Annual 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

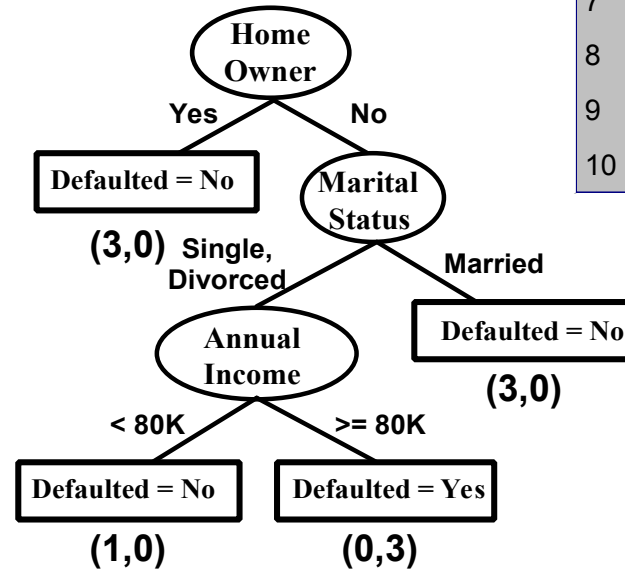


(7,3)

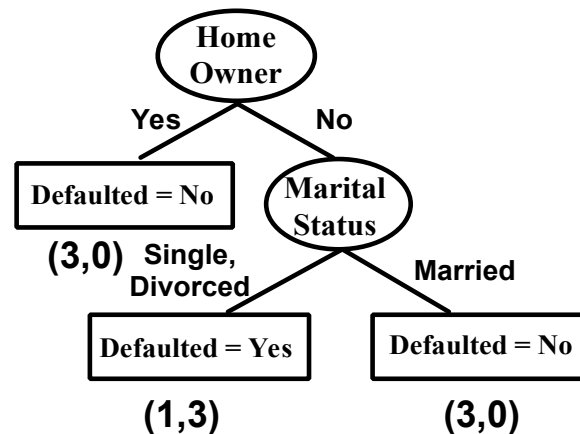
(a)



(b)



(d)



(c)

# Design Issues of Decision Tree Induction

---

- How should training records be split?
  - Method for specifying test condition
    - ◆ depending on attribute types
  - Measure for evaluating the goodness of a test condition
- How should the splitting procedure stop?
  - Stop splitting if all the records belong to the same class or have identical attribute values
  - Early termination

# Methods for Expressing Test Conditions

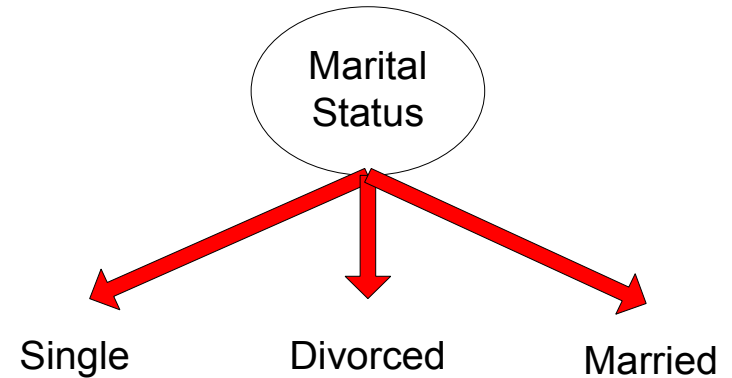
---

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

# Test Condition for Nominal Attributes

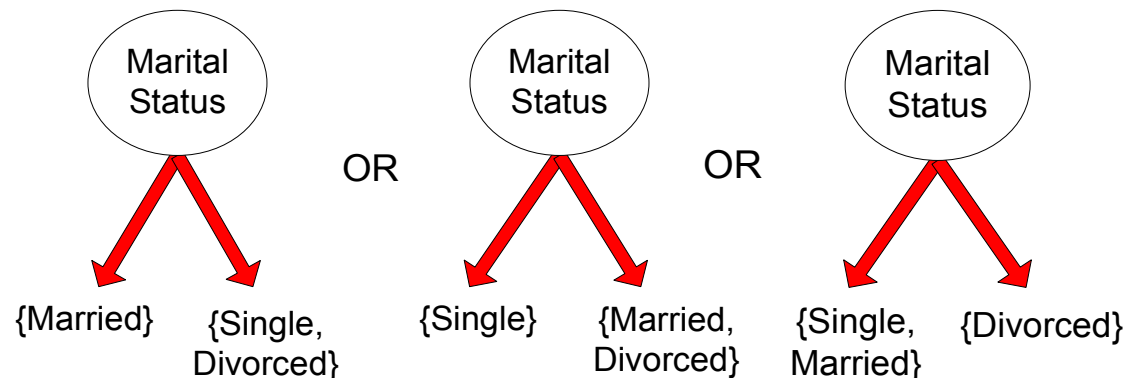
- **Multi-way split:**

- Use as many partitions as distinct values.



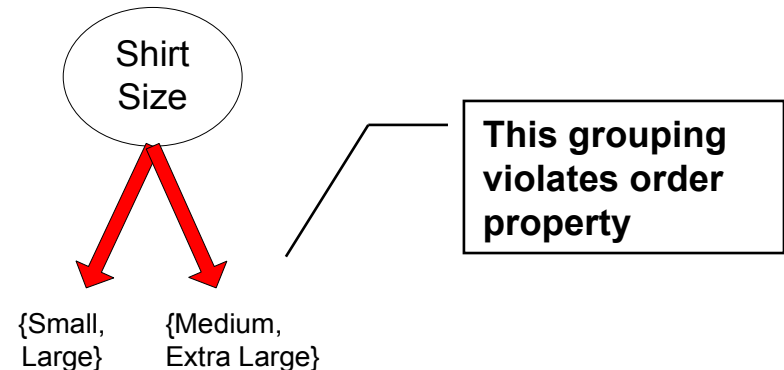
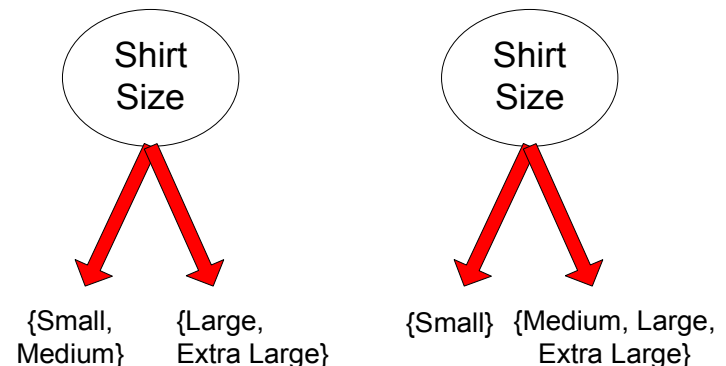
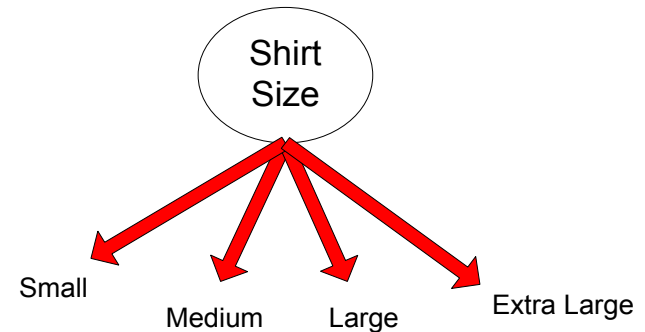
- **Binary split:**

- Divides values into two subsets



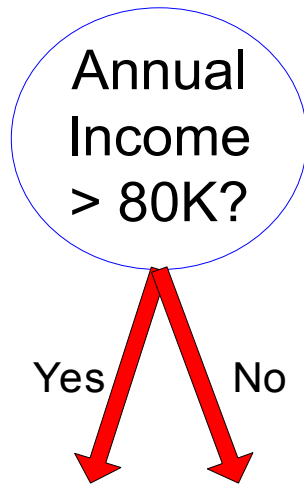
# Test Condition for Ordinal Attributes

- **Multi-way split:**
  - Use as many partitions as distinct values
- **Binary split:**
  - Divides values into two subsets
  - Preserve order property among attribute values

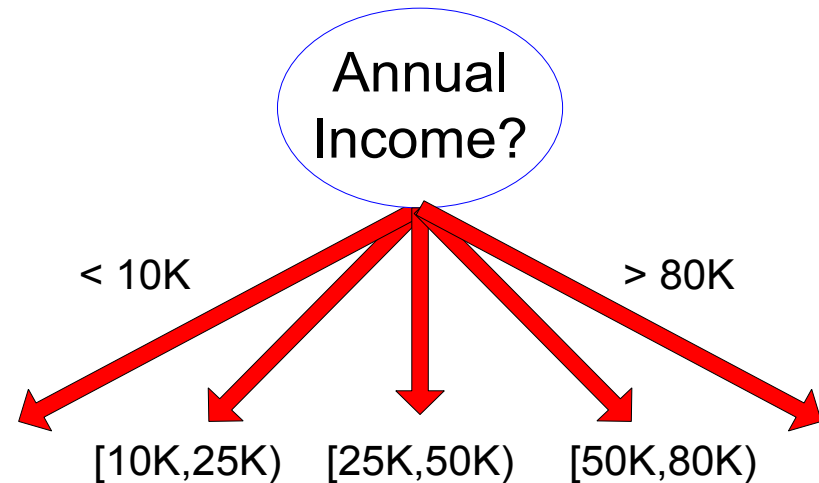




# Test Condition for Continuous Attributes



(i) Binary split



(ii) Multi-way split

# Splitting Based on Continuous Attributes

---

- Different ways of handling
  - **Discretization** to form an ordinal categorical attribute

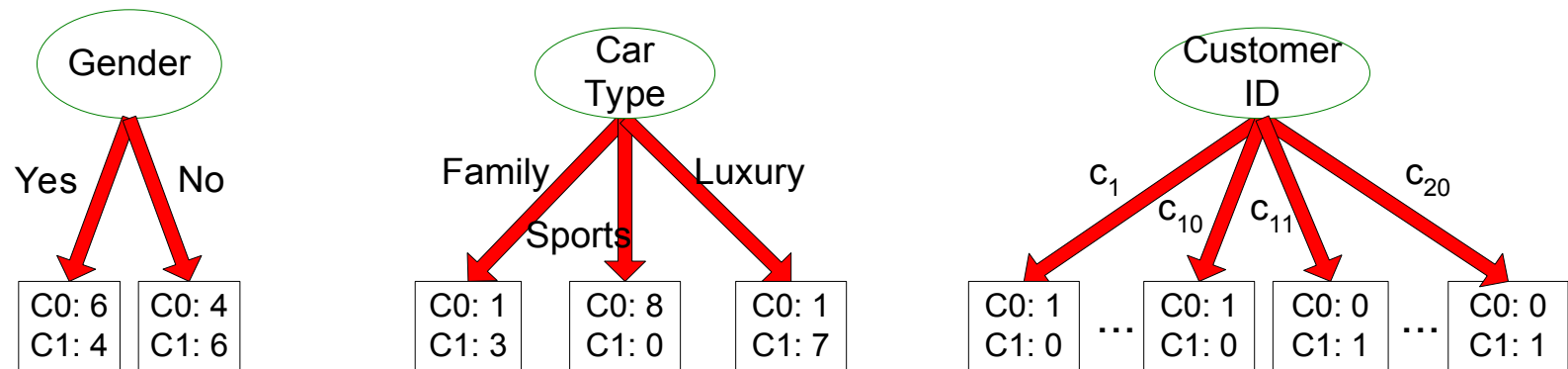
Ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.

    - ◆ Static – discretize once at the beginning
    - ◆ Dynamic – repeat at each node
  - **Binary Decision**:  $(A < v)$  or  $(A \geq v)$ 
    - ◆ consider all possible splits and finds the best cut
    - ◆ can be more compute intensive

# How to determine the Best Split

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

Customer Id	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1



**Which test condition is the best?**

# How to determine the Best Split

---

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

C0: 5
C1: 5

High degree of impurity

C0: 9
C1: 1

Low degree of impurity

# Measures of Node Impurity

- Gini Index

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

- Entropy

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

- Misclassification error

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

# Finding the Best Split

---

1. Compute impurity measure (P) before splitting
2. Compute impurity measure (M) after splitting
  - Compute impurity measure of each child node
  - M is the weighted impurity of children
3. Choose the attribute test condition that produces the highest gain

$$\text{Gain} = P - M$$

or equivalently, lowest impurity measure after splitting (M)

# Finding the Best Split

Before Splitting:

C0	<b>N00</b>
C1	<b>N01</b>

→ P

A?

Yes

No

Node N1

Node N2

C0	<b>N10</b>
C1	<b>N11</b>

C0	<b>N20</b>
C1	<b>N21</b>

↓  
**M11**

↓  
**M12**

**M1**

B?

Yes

No

Node N3

Node N4

C0	<b>N30</b>
C1	<b>N31</b>

C0	<b>N40</b>
C1	<b>N41</b>

↓  
**M21**

↓  
**M22**

**M2**

**Gain = P – M1    vs    P – M2**

# Measure of Impurity: GINI

- Gini Index for a given node  $t$  :

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

(NOTE:  $p(j | t)$  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



# Measure of Impurity: GINI

- Gini Index for a given node  $t$  :

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

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

- For 2-class problem ( $p, 1 - p$ ):
  - ◆  $GINI = 1 - p^2 - (1 - p)^2 = 2p(1-p)$

C1	<b>0</b>
C2	<b>6</b>
<b>Gini=0.000</b>	

C1	<b>1</b>
C2	<b>5</b>
<b>Gini=0.278</b>	

C1	<b>2</b>
C2	<b>4</b>
<b>Gini=0.444</b>	

C1	<b>3</b>
C2	<b>3</b>
<b>Gini=0.500</b>	

# Computing Gini Index of a Single Node

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

C1	<b>0</b>
C2	<b>6</b>

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$Gini = 1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = 0$$

C1	<b>1</b>
C2	<b>5</b>

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$Gini = 1 - (1/6)^2 - (5/6)^2 = 0.278$$

C1	<b>2</b>
C2	<b>4</b>

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$Gini = 1 - (2/6)^2 - (4/6)^2 = 0.444$$

# Computing Gini Index for a Collection of Nodes

- When a node  $p$  is split into  $k$  partitions (children)

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

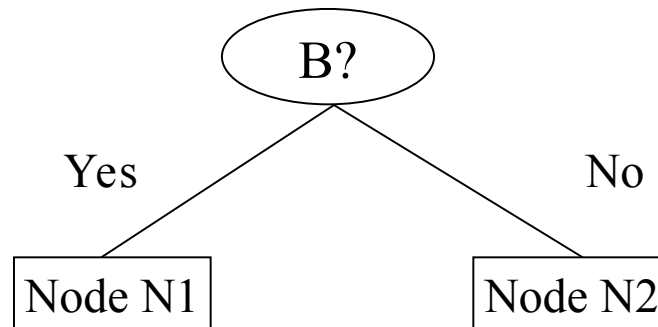
where,  $n_i$  = number of records at child  $i$ ,

$n$  = number of records at parent node  $p$ .

- Choose the attribute that minimizes weighted average Gini index of the children
- Gini index is used in decision tree algorithms such as CART, SLIQ, SPRINT

# Binary Attributes: Computing GINI Index

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



	Parent
C1	<b>7</b>
C2	<b>5</b>
<b>Gini = 0.486</b>	

$$\begin{aligned}
 \text{Gini}(N1) &= 1 - (5/6)^2 - (1/6)^2 \\
 &= 0.278
 \end{aligned}$$

$$\begin{aligned}
 \text{Gini}(N2) &= 1 - (2/6)^2 - (4/6)^2 \\
 &= 0.444
 \end{aligned}$$

	N1	N2
C1	<b>5</b>	<b>2</b>
C2	<b>1</b>	<b>4</b>
<b>Gini=0.361</b>		

$$\begin{aligned}
 \text{Weighted Gini of N1 N2} &= 6/12 * 0.278 + \\
 &\quad 6/12 * 0.444 \\
 &= 0.361
 \end{aligned}$$

$$\text{Gain} = 0.486 - 0.361 = 0.125$$

# 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	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	0.468	

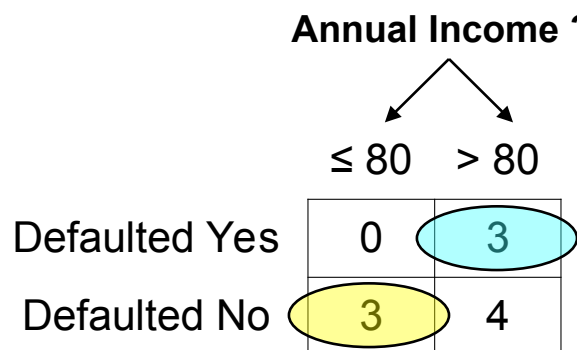
	CarType	
	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10
Gini	0.167	

Which of these is the best?

# 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  $A \geq v$
- 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.

ID	Home Owner	Marital Status	Annual Income	Defaulted
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

Sorted Values		Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
		Annual Income										
	→		60	70	75	85	90	95	100	120	125	220

---

- Sorted Values** →
- Split Positions** →

[illegible]



\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

- ↓

Cheat	No		No		No		Yes		Yes		Yes		No		No		No		No	
S	Annual Income																			
	60		70		75		85		90		95		100		120		125		220	
	55	65	72	80	87	92	97	110	122	172	230									
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes				0	3															
No				3	4															
Gini				0.343																

# 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

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

# 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			
Sorted Values Split Positions		Annual Income																					
		60		70		75		85		90		95		100		120		125		220			
		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		<u>0.300</u>		0.343		0.375		0.400		0.420	

# Measure of Impurity: Entropy

- Entropy at a given node t:

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

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

- ◆ 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 quite similar to the GINI index computations

# Computing Entropy of a Single Node

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

C1	<b>0</b>
C2	<b>6</b>

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$Entropy = - 0 \log 0 - 1 \log 1 = - 0 - 0 = 0$$

C1	<b>1</b>
C2	<b>5</b>

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

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

C1	<b>2</b>
C2	<b>4</b>

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

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

# Computing Information Gain After Splitting

- Information Gain:

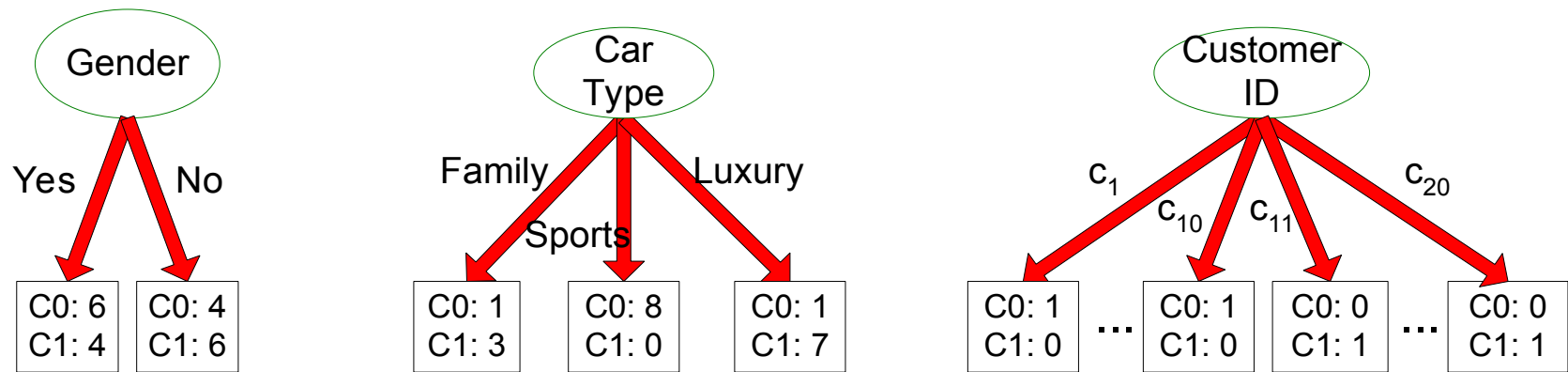
$$GAIN_{split} = Entropy(p) - \left( \sum_{i=1}^k \frac{n_i}{n} Entropy(i) \right)$$

Parent Node, p is split into k partitions;  
 $n_i$  is number of records in partition i

- Choose the split that achieves most reduction (maximizes GAIN)
- Used in the ID3 and C4.5 decision tree algorithms

# Problem with large number of partitions

- Node impurity measures tend to prefer splits that result in large number of partitions, each being small but pure



- Customer ID has highest information gain because entropy for all the children is zero

# Gain Ratio

- Gain Ratio:

$$GainRatio_{split} = \frac{GAIN_{Split}}{SplitINFO} \quad SplitINFO = - \sum_{i=1}^k \frac{n_i}{n} \log \frac{n_i}{n}$$

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 algorithm
- Designed to overcome the disadvantage of Information Gain



# Gain Ratio

- Gain Ratio:

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO} \quad SplitINFO = - \sum_{i=1}^k \frac{n_i}{n} \log \frac{n_i}{n}$$

Parent Node, p is split into k partitions

$n_i$  is the number of records in partition i

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

SplitINFO = 1.52

	CarType	
	{Sports, Luxury}	{Family}
C1	9	1
C2	7	3
Gini	0.468	

SplitINFO = 0.72

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

SplitINFO = 0.97

# Measure of Impurity: Classification Error

- Classification error at a node  $t$  :

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

- 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

# Computing Error of a Single Node

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

C1	<b>0</b>
C2	<b>6</b>

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$Error = 1 - \max(0, 1) = 1 - 1 = 0$$

C1	<b>1</b>
C2	<b>5</b>

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$Error = 1 - \max(1/6, 5/6) = 1 - 5/6 = 1/6$$

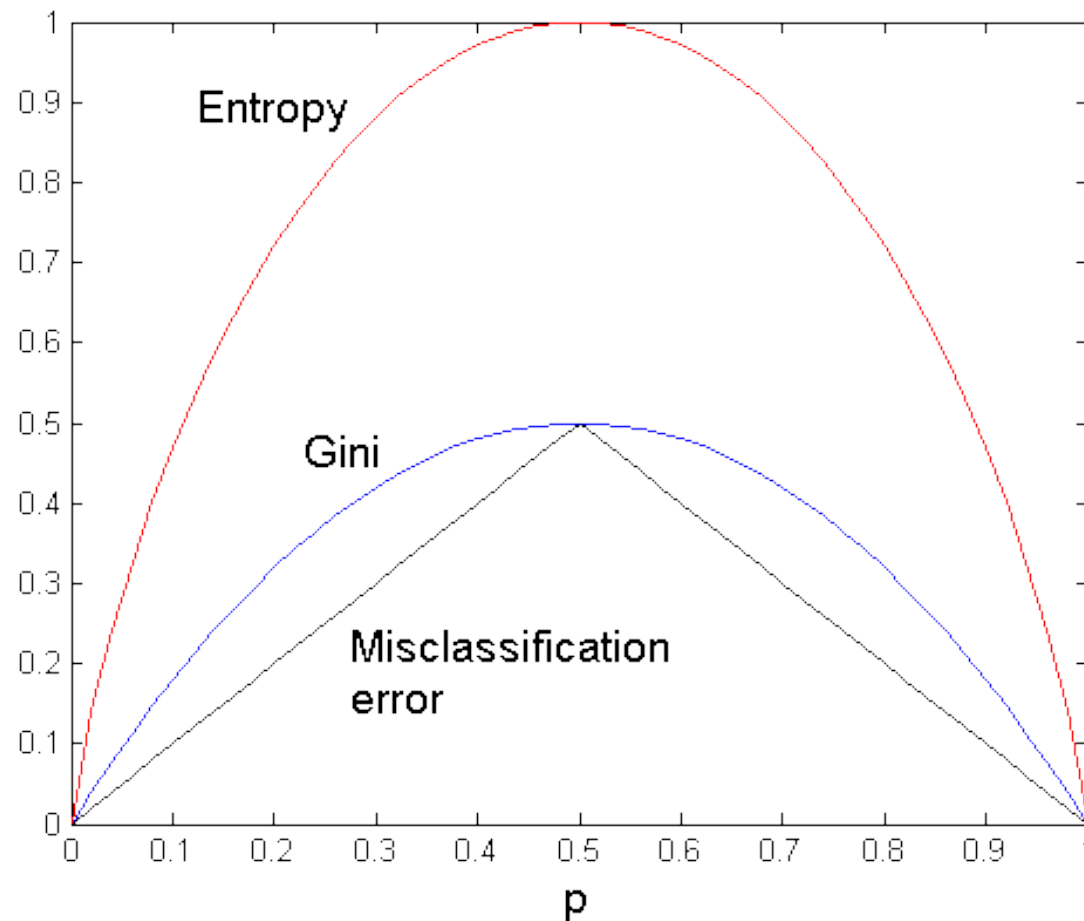
C1	<b>2</b>
C2	<b>4</b>

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

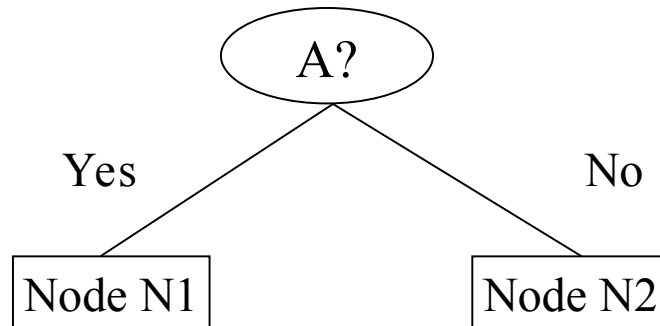
$$Error = 1 - \max(2/6, 4/6) = 1 - 4/6 = 1/3$$

# Comparison among Impurity Measures

For a 2-class problem:



# Misclassification Error vs Gini Index



	Parent
C1	7
C2	3
<b>Gini = 0.42</b>	

$$\begin{aligned}\text{Gini}(N1) &= 1 - (3/3)^2 - (0/3)^2 \\ &= 0\end{aligned}$$

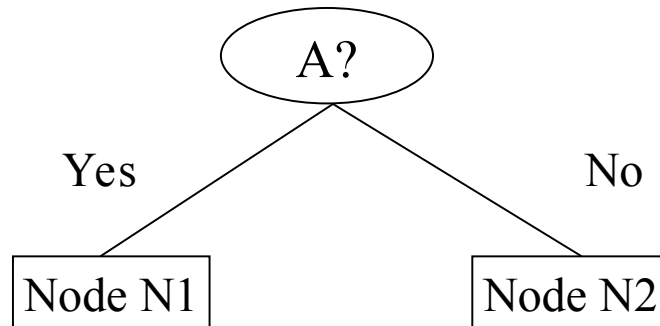
$$\begin{aligned}\text{Gini}(N2) &= 1 - (4/7)^2 - (3/7)^2 \\ &= 0.489\end{aligned}$$

	N1	N2
C1	3	4
C2	0	3
<b>Gini=0.342</b>		

$$\begin{aligned}\text{Gini(Children)} &= 3/10 * 0 \\ &+ 7/10 * 0.489 \\ &= 0.342\end{aligned}$$

**Gini improves but  
error remains the  
same!!**

# Misclassification Error vs Gini Index



	Parent
C1	7
C2	3
<b>Gini = 0.42</b>	

	N1	N2
C1	3	4
C2	0	3
<b>Gini=0.342</b>		

	N1	N2
C1	3	4
C2	1	2
<b>Gini=0.416</b>		

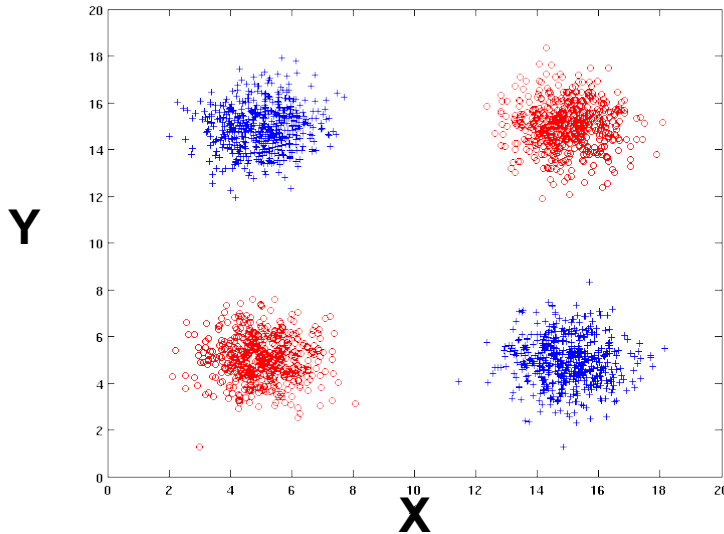
**Misclassification error for all three cases = 0.3 !**

# Decision Tree Based Classification

---

- Advantages:
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Robust to noise (especially when methods to avoid overfitting are employed)
  - Can easily handle redundant or irrelevant attributes (unless the attributes are interacting)
- Disadvantages:
  - Space of possible decision trees is exponentially large. Greedy approaches are often unable to find the best tree.
  - Does not take into account interactions between attributes
  - Each decision boundary involves only a single attribute

# Handling interactions



**+** : 1000 instances

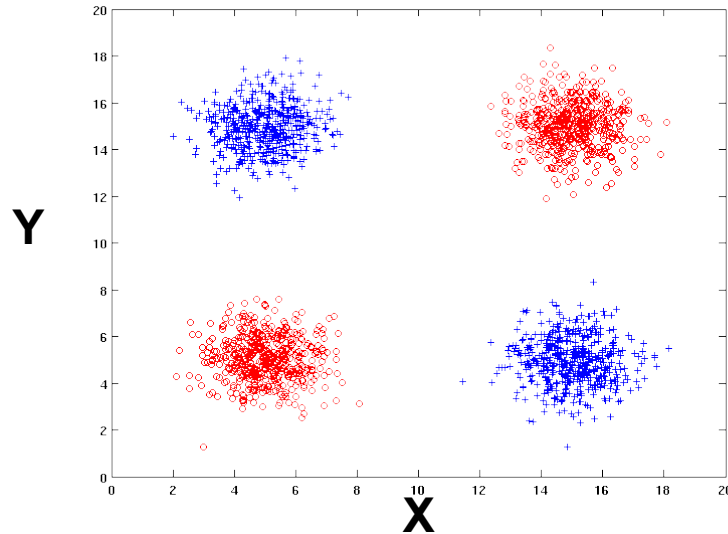
**o** : 1000 instances

Entropy (X) : 0.99

Entropy (Y) : 0.99



# Handling interactions



**+** : 1000 instances

**o** : 1000 instances

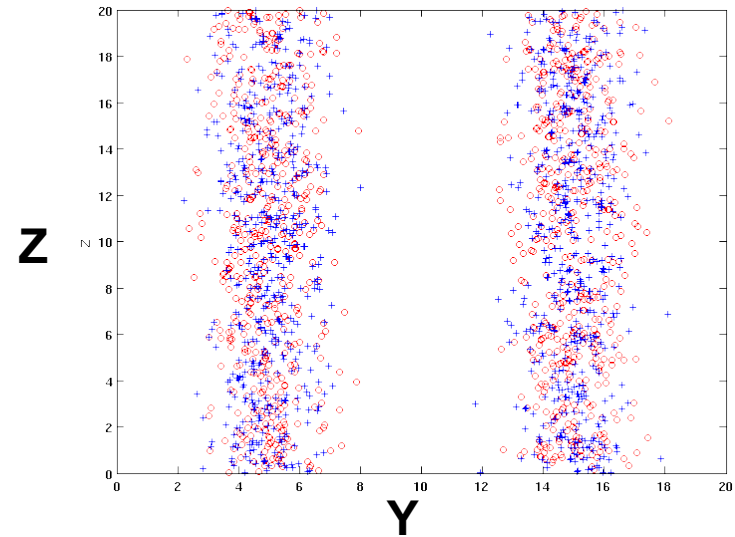
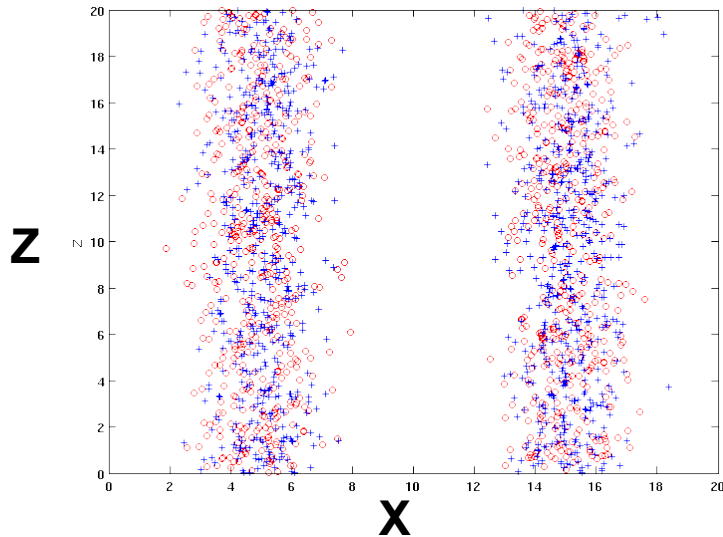
Entropy (X) : 0.99

Entropy (Y) : 0.99

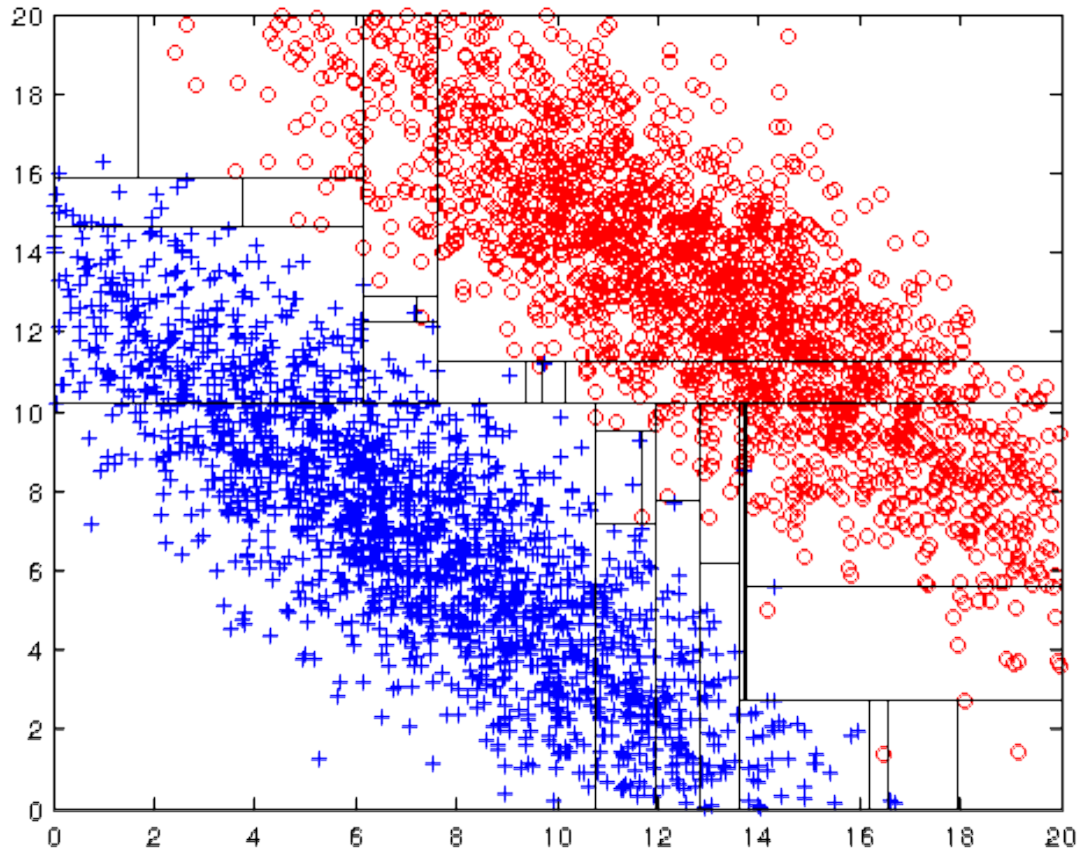
Entropy (Z) : 0.98

Adding Z as a noisy  
attribute generated  
from a uniform  
distribution

Attribute Z will be  
chosen for splitting!



# Limitations of single attribute-based decision boundaries



Both **positive (+)** and **negative (o)** classes generated from skewed Gaussians with centers at (8,8) and (12,12) respectively.