

Data Mining Classification: Basic Concepts and Techniques

Lecture Notes for Chapter 3

Introduction to Data Mining, 2nd 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 x-rays or 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

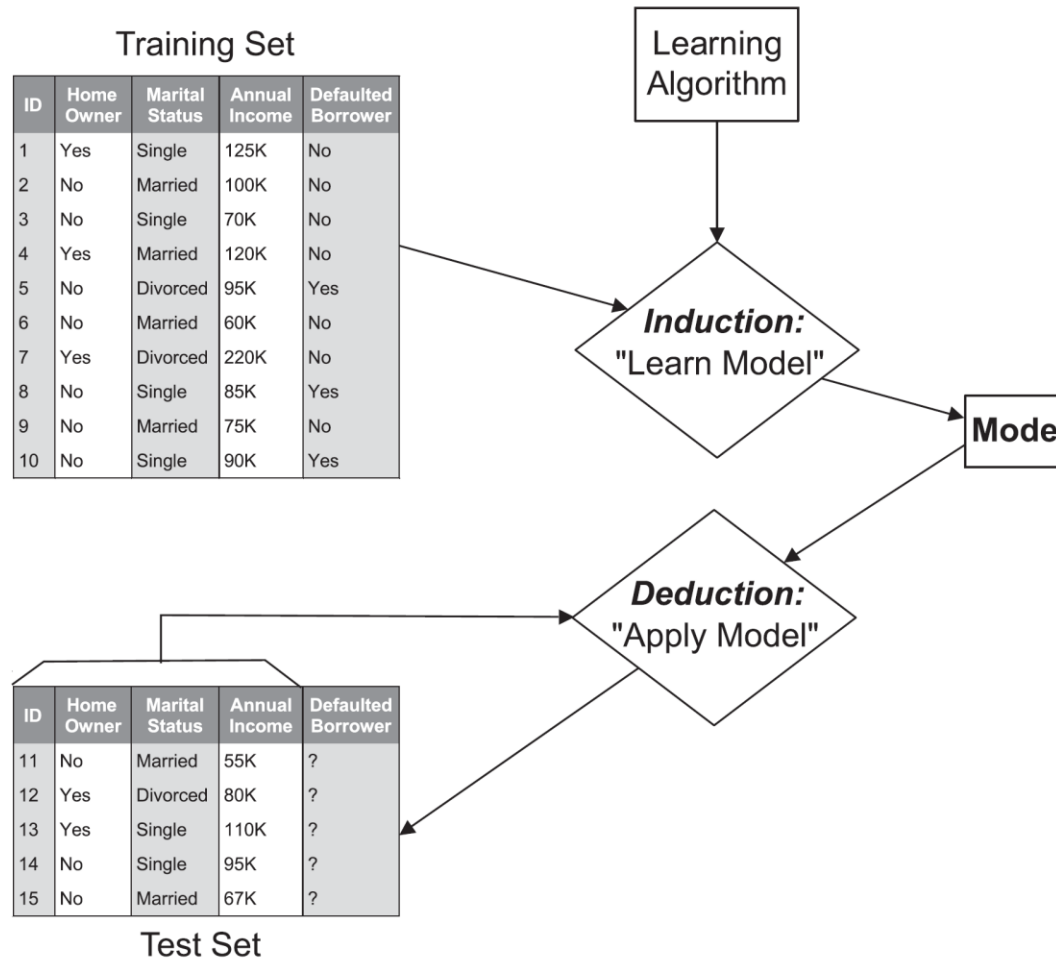


Figure 3.3. General framework for building a classification model.

Classification Techniques

□ Base Classifiers

- Decision Tree based Methods
- Rule-based Methods
- Nearest-neighbor
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines
- Neural Networks, Deep Neural Nets

□ Ensemble Classifiers

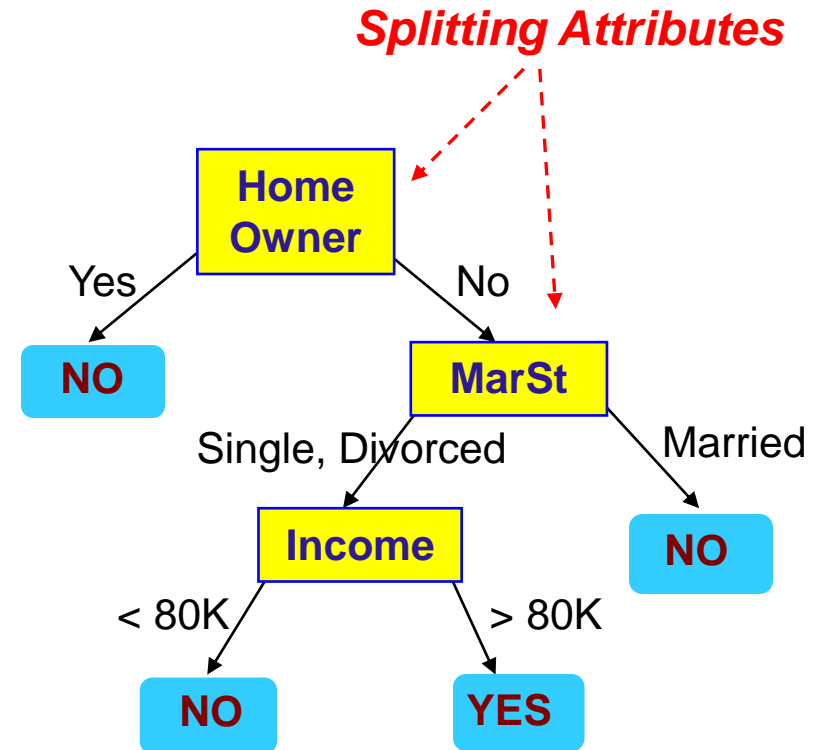
- Boosting, Bagging, Random Forests

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

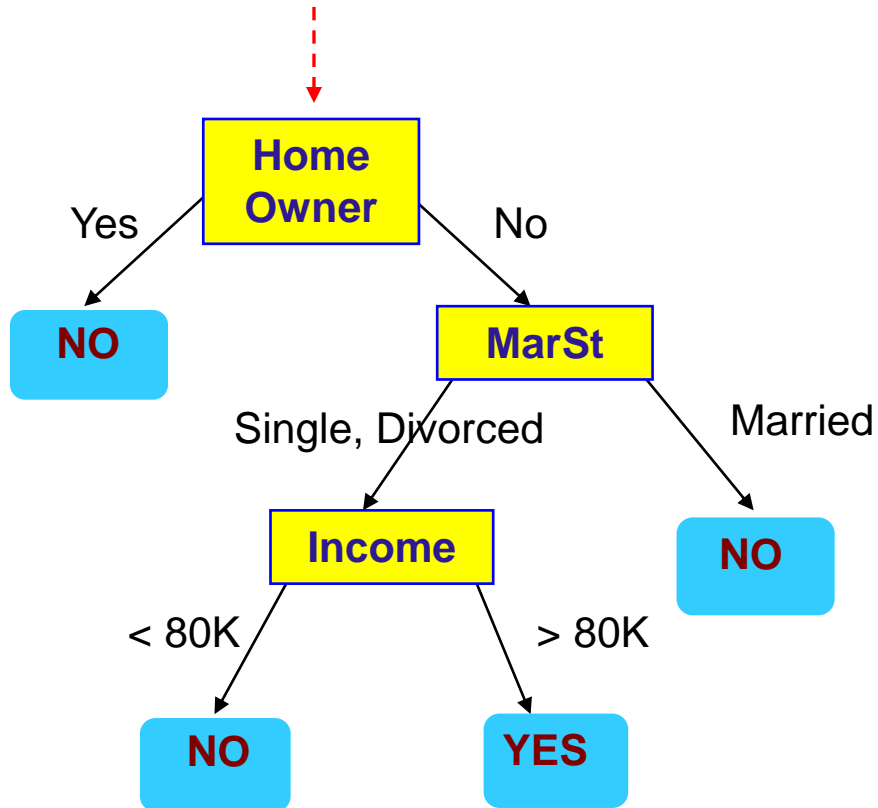
Training Data



Model: Decision Tree

Apply Model to Test Data

Start from the root of tree.



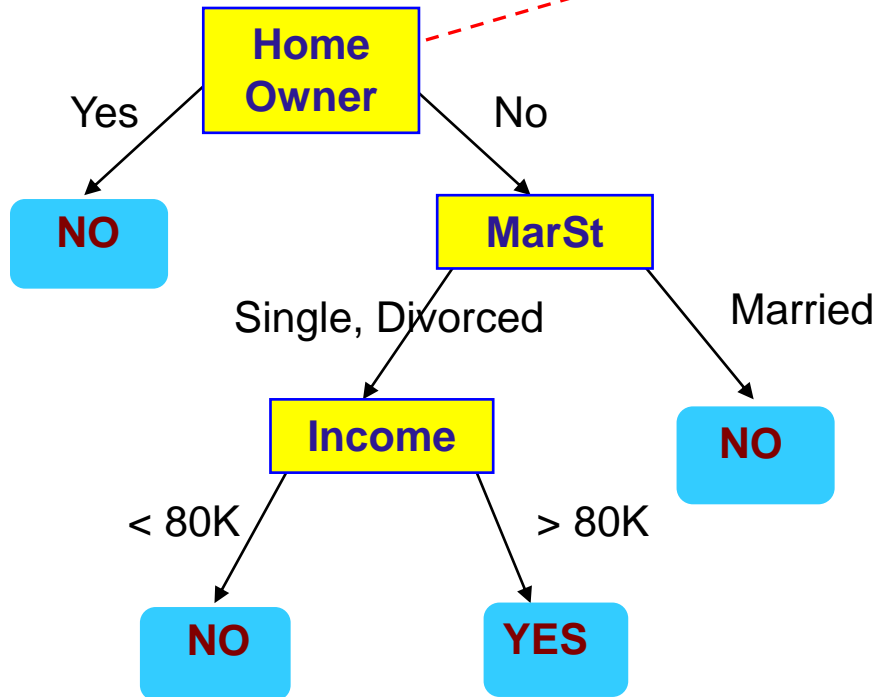
Test Data

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

Apply Model to Test Data

Test Data

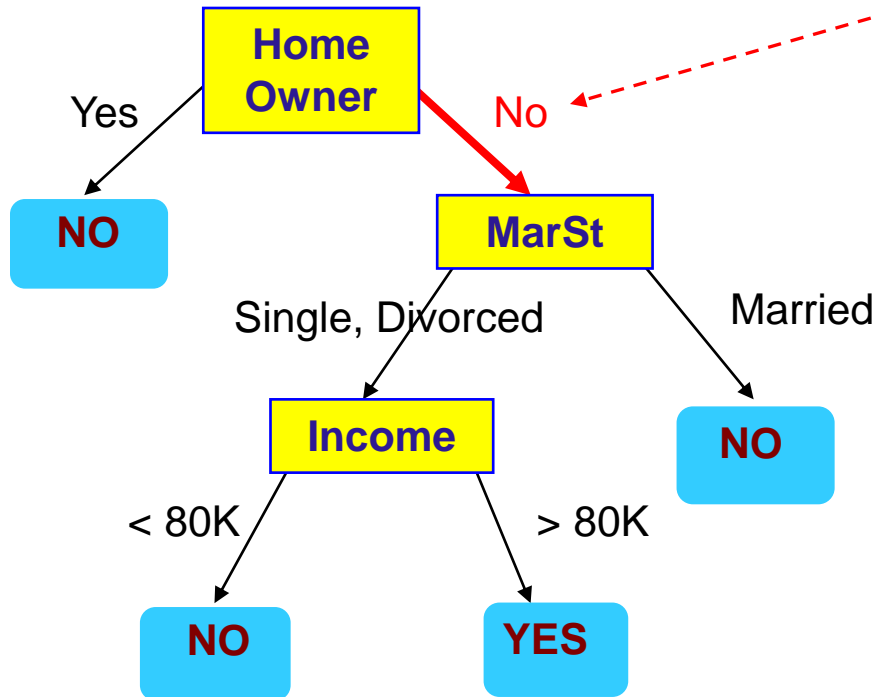
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply Model to Test Data

Test Data

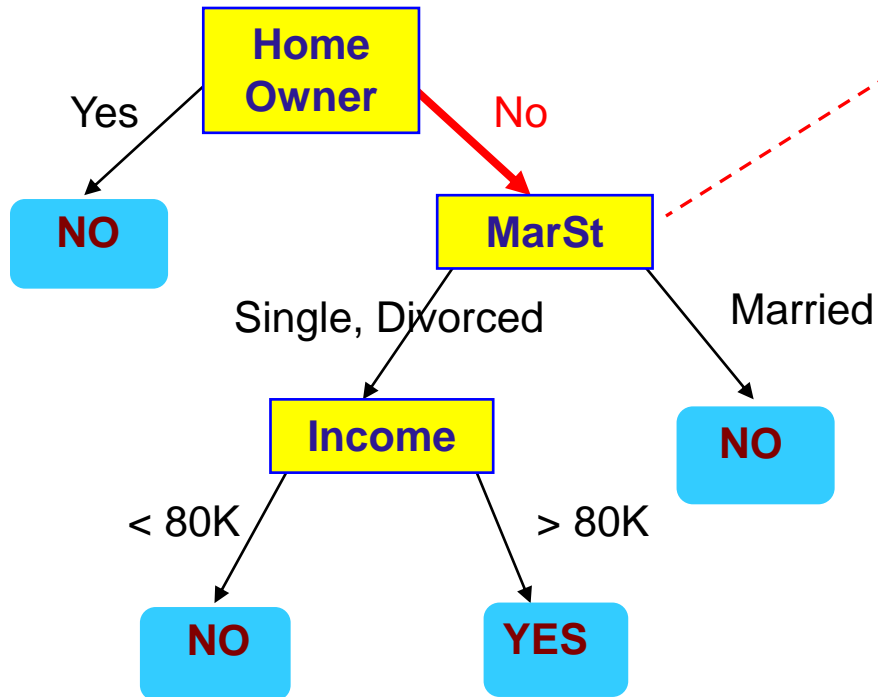
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply Model to Test Data

Test Data

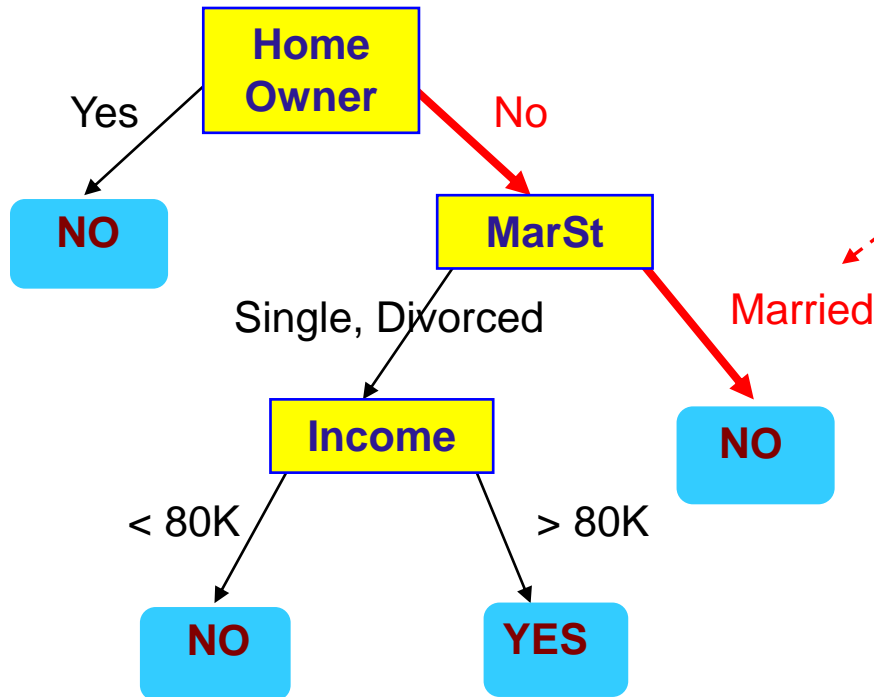
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply Model to Test Data

Test Data

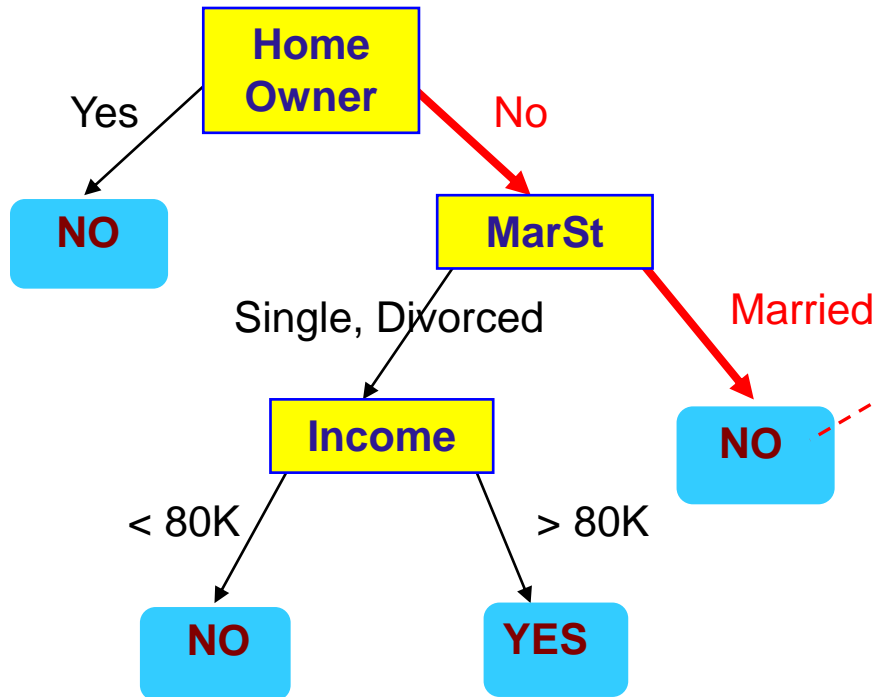
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply Model to Test Data

Test Data

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

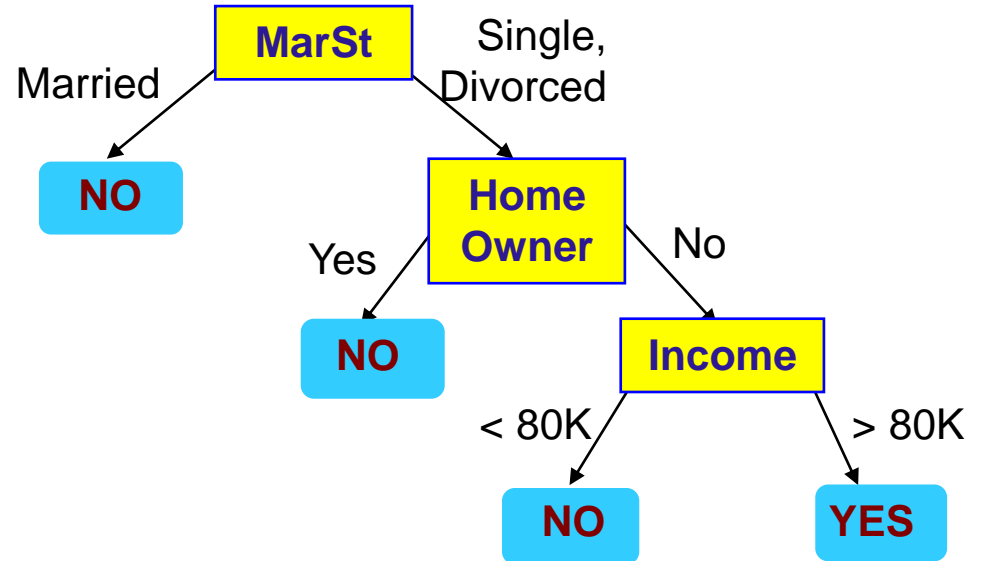


Assign Defaulted to
"No"

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!

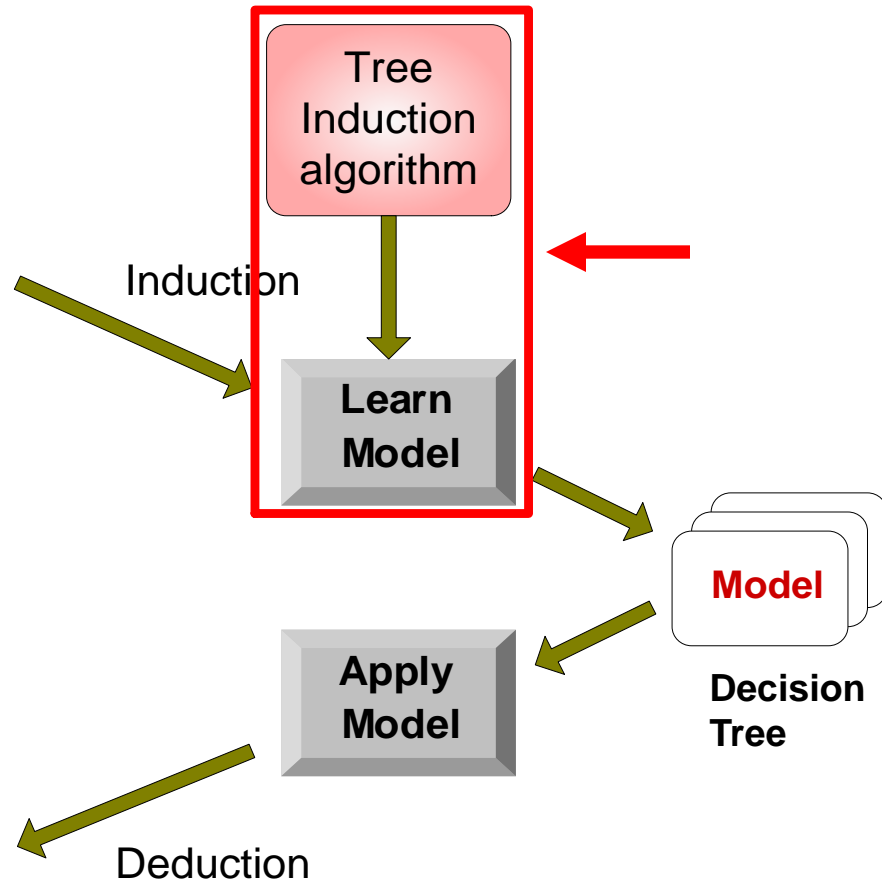
Decision Tree Classification Task

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



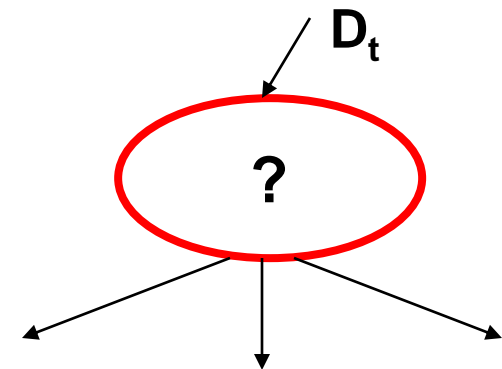
Decision Tree Induction

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

General Structure of 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 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)

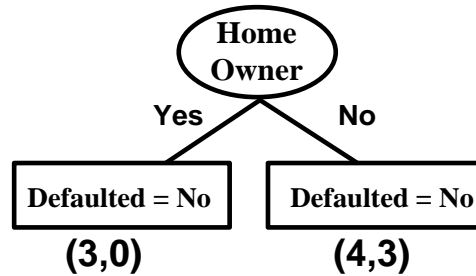
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Hunt's Algorithm

Defaulted = No

(7,3)

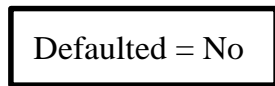
(a)



(b)

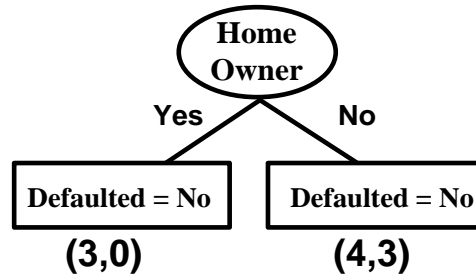
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Hunt's Algorithm



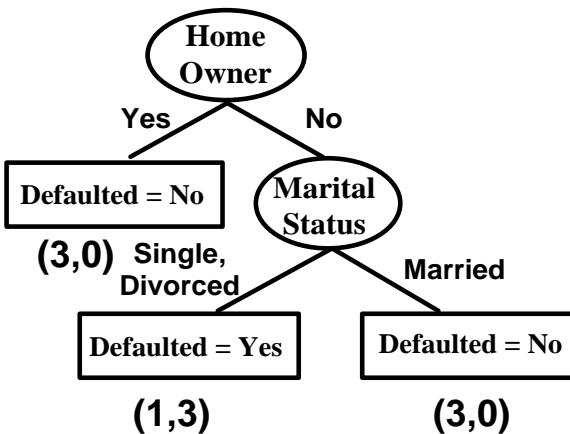
(7,3)

(a)



(b)

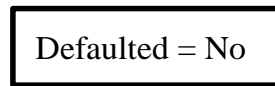
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



(c)

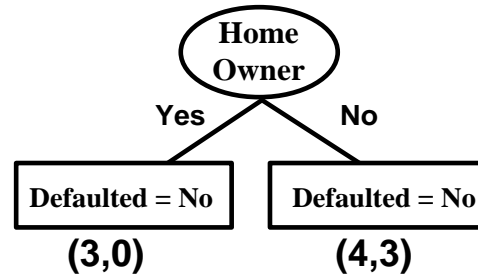
Hunt's Algorithm

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

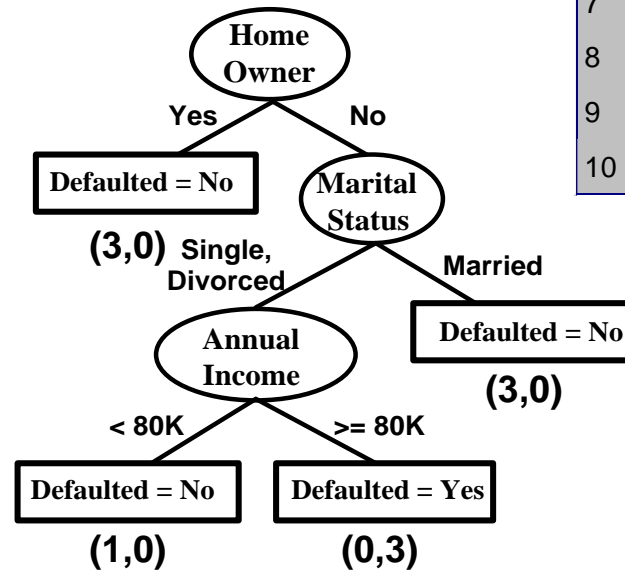


(7,3)

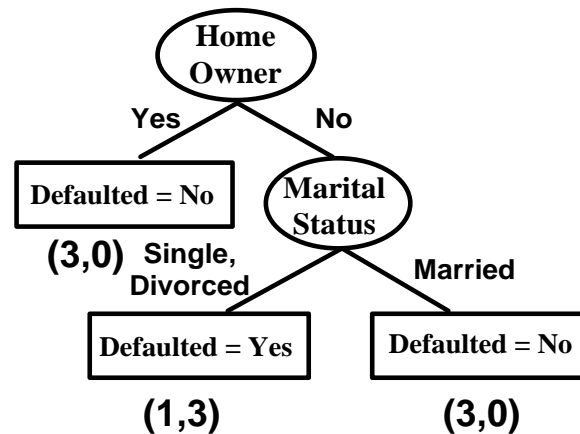
(a)



(b)



(d)



(c)

Design Issues of Decision Tree Induction

- | How should training records be split?
 - Method for expressing 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

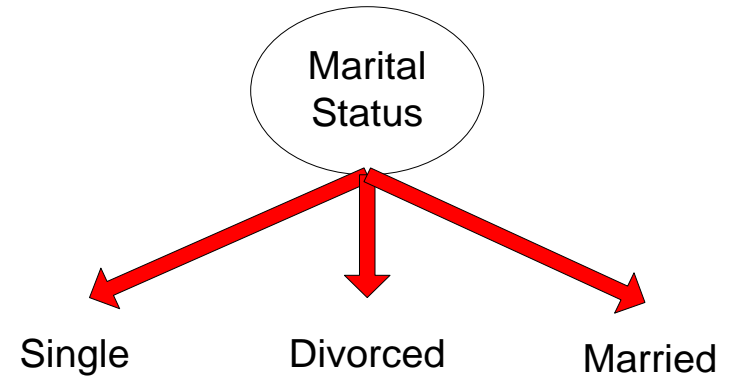
Methods for Expressing Test Conditions

- | Depends on attribute types
 - Binary
 - Nominal
 - Ordinal
 - Continuous

Test Condition for Nominal Attributes

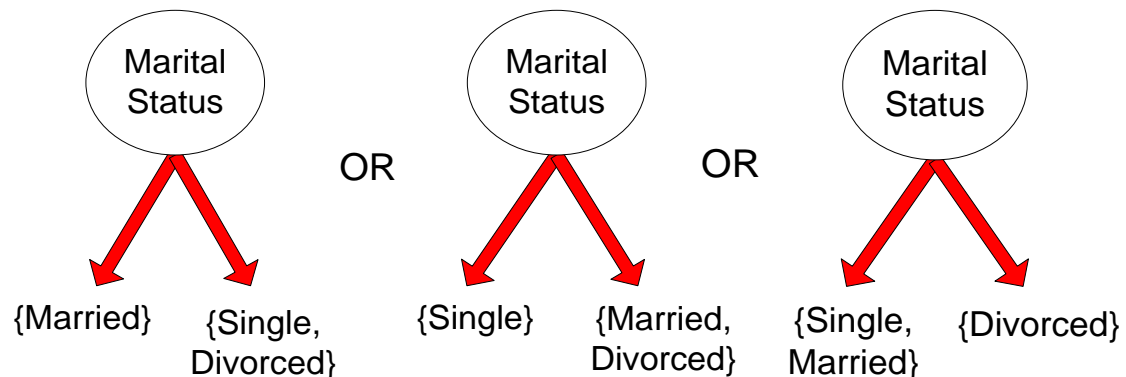
□ Multi-way split:

- Use as many partitions as distinct values.



□ Binary split:

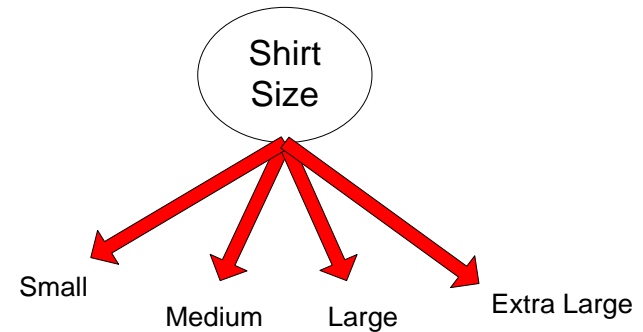
- Divides values into two subsets



Test Condition for Ordinal Attributes

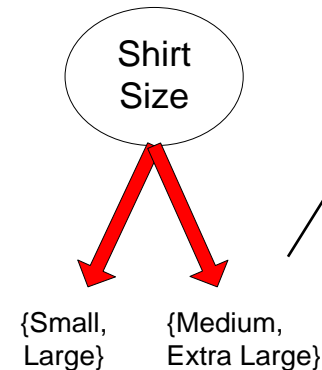
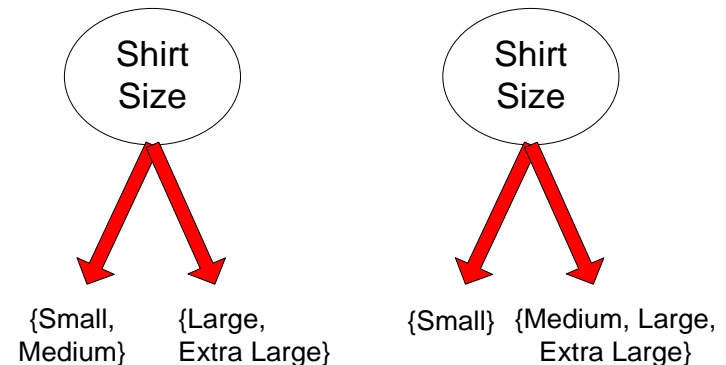
| Multi-way split:

- Use as many partitions as distinct values



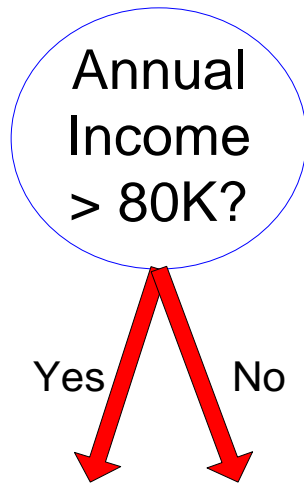
| Binary split:

- Divides values into two subsets
- Preserve order property among attribute values

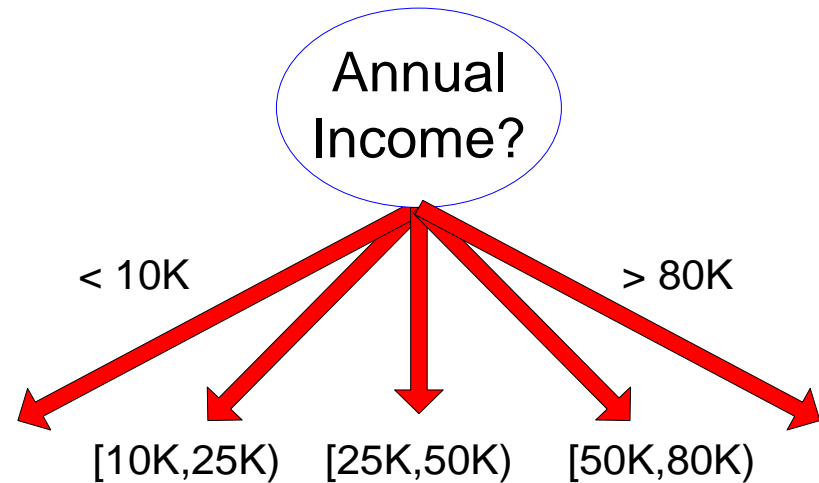


**This grouping
violates order
property**

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

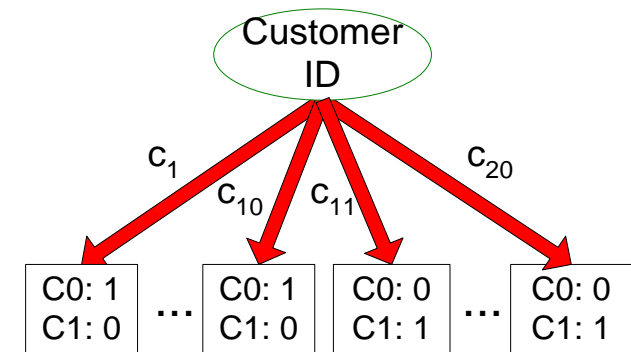
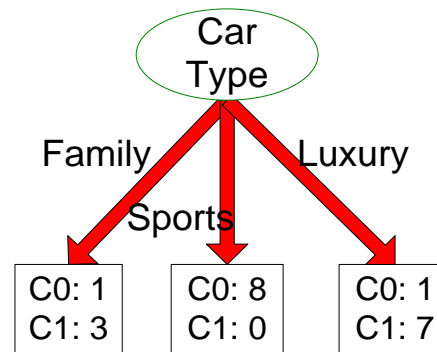
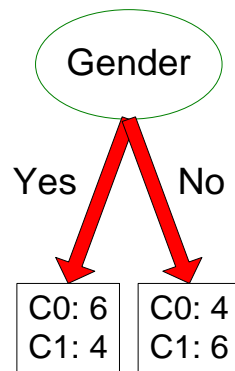
- **Binary Decision**: $(A < v)$ or $(A \geq v)$

- ◆ 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

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

Where $p_i(t)$ is the frequency of class i at node t , and c is the total number of classes

| Entropy

$$Entropy = - \sum_{i=0}^{c-1} p_i(t) \log_2 p_i(t)$$

| Misclassification error

$$Classification\ error = 1 - \max[p_i(t)]$$

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 child nodes
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	N00
C1	N01

→ **P**

A?

Yes

No

Node N1

Node N2

C0

N10

C1

N11

C0

N20

C1

N21



M11



M12

M1

B?

Yes

No

Node N3

Node N4

C0

N30

C1

N31

C0

N40

C1

N41



M21



M22

M2

Gain = P – M1 vs P – M2

Measure of Impurity: GINI

- Gini Index for a given node t

$$\text{Gini Index} = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

Where $p_i(t)$ is the frequency of class i at node t , and c is the total number of classes

- Maximum of $1 - 1/c$ when records are equally distributed among all classes, implying the least beneficial situation for classification
- Minimum of 0 when all records belong to one class, implying the most beneficial situation for classification
- Gini index is used in decision tree algorithms such as CART, SLIQ, SPRINT

Measure of Impurity: GINI

- Gini Index for a given node t :

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

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

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	

Computing Gini Index of a Single Node

$$\text{Gini Index} = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

C1	0
C2	6

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

$$\text{Gini} = 1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = 0$$

C1	1
C2	5

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

$$\text{Gini} = 1 - (1/6)^2 - (5/6)^2 = 0.278$$

C1	2
C2	4

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

$$\text{Gini} = 1 - (2/6)^2 - (4/6)^2 = 0.444$$

Computing Gini Index for a Collection of Nodes

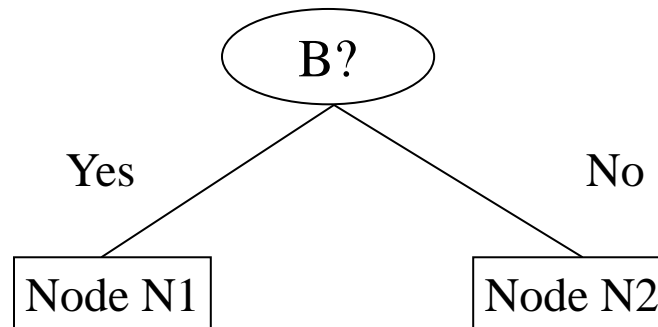
- | When a node p is split into k partitions (children)

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

where, n_i = number of records at child i ,
 n = number of records at parent node p .

Binary Attributes: Computing GINI Index

- Splits into two partitions (child nodes)
- Effect of Weighing partitions:
 - Larger and purer partitions are sought



	Parent
C1	7
C2	5
Gini = 0.486	

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

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

	N1	N2
C1	5	2
C2	1	4
Gini=0.361		

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

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

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	

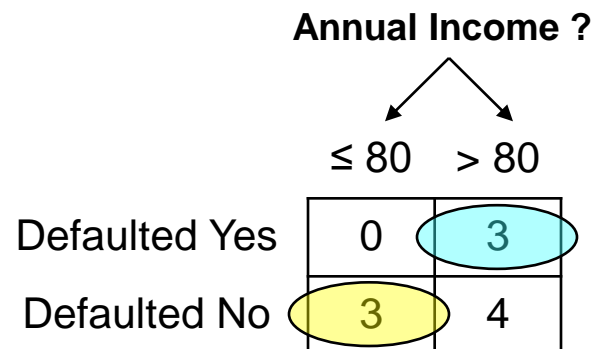
	CarType	
	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10
Gini	0.167	

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, $A \leq v$ 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.

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

Continuous Attributes: Computing Gini Index...

- I 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 Split Positions	Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No											
	Annual Income																					
	60		70		75		85		90		95		100		120		125		220			
	55		65		72		80		87		92		97		110		122		172		230	
	<= >		<= >		<= >		<= >		<= >		<= >		<= >		<= >		<= >		<= >		<= >	

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

Sorted Values

Split Positions

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No		
	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	1	2					
No				3	4	3	4					
Gini				0.343	0.417							

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 Split Positions		Cheat	No		No		No		Yes		Yes		Yes		No		No		No		No		
		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		
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

$$Entropy = - \sum_{i=0}^{c-1} p_i(t) \log_2 p_i(t)$$

Where $p_i(t)$ is the frequency of class i at node t , and c is the total number of classes

- ◆ Maximum of $\log_2 c$ when records are equally distributed among all classes, implying the least beneficial situation for classification
- ◆ Minimum of 0 when all records belong to one class, implying most beneficial situation for classification
- Entropy based computations are quite similar to the GINI index computations

Computing Entropy of a Single Node

$$Entropy = - \sum_{i=0}^{c-1} p_i(t) \log_2 p_i(t)$$

C1	0
C2	6

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

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

C1	1
C2	5

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

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

C1	2
C2	4

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

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

Computing Information Gain After Splitting

I Information Gain:

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

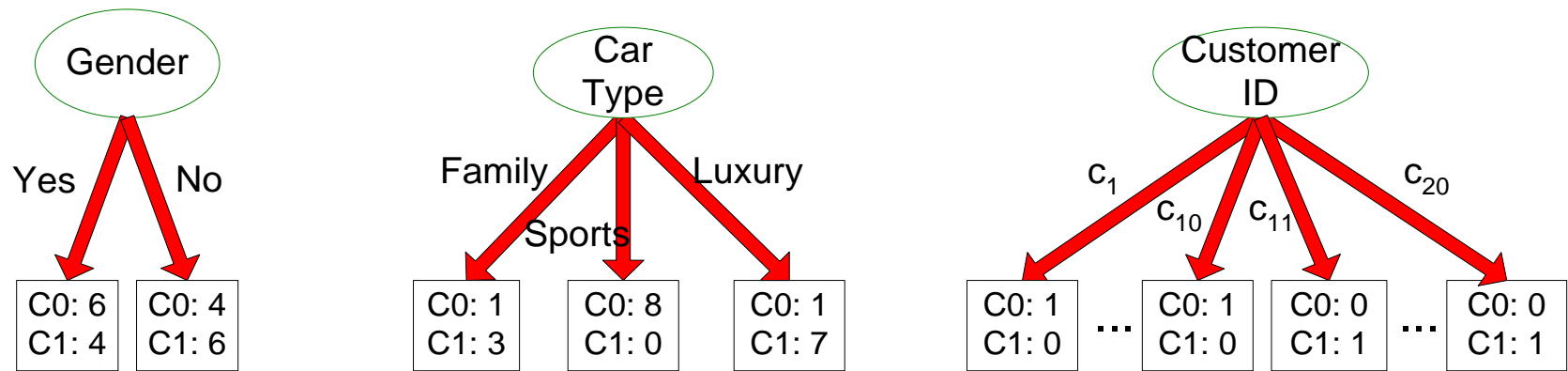
Parent Node, p is split into k partitions (children)

n_i is number of records in child node i

- Choose the split that achieves most reduction (maximizes GAIN)
- Used in the ID3 and C4.5 decision tree algorithms
- Information gain is the mutual information between the class variable and the splitting variable

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:

$$\text{Gain Ratio} = \frac{\text{Gain}_{\text{split}}}{\text{Split Info}} \qquad \text{Split Info} = - \sum_{i=1}^k \frac{n_i}{n} \log_2 \frac{n_i}{n}$$

Parent Node, p is split into k partitions (children)

n_i is number of records in child node i

- Adjusts Information Gain by the entropy of the partitioning (*Split Info*).
 - ◆ 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:

$$\text{Gain Ratio} = \frac{\text{Gain}_{\text{split}}}{\text{Split Info}} \quad \text{Split Info} = \sum_{i=1}^k \frac{n_i}{n} \log_2 \frac{n_i}{n}$$

Parent Node, p is split into k partitions (children)

n_i is number of records in child node 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 of $1 - 1/c$ when records are equally distributed among all classes, implying the least interesting situation
- Minimum of 0 when all records belong to one class, implying the most interesting situation

Computing Error of a Single Node

$$Error(t) = 1 - \max_i [p_i(t)]$$

C1	0
C2	6

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

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

C1	1
C2	5

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

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

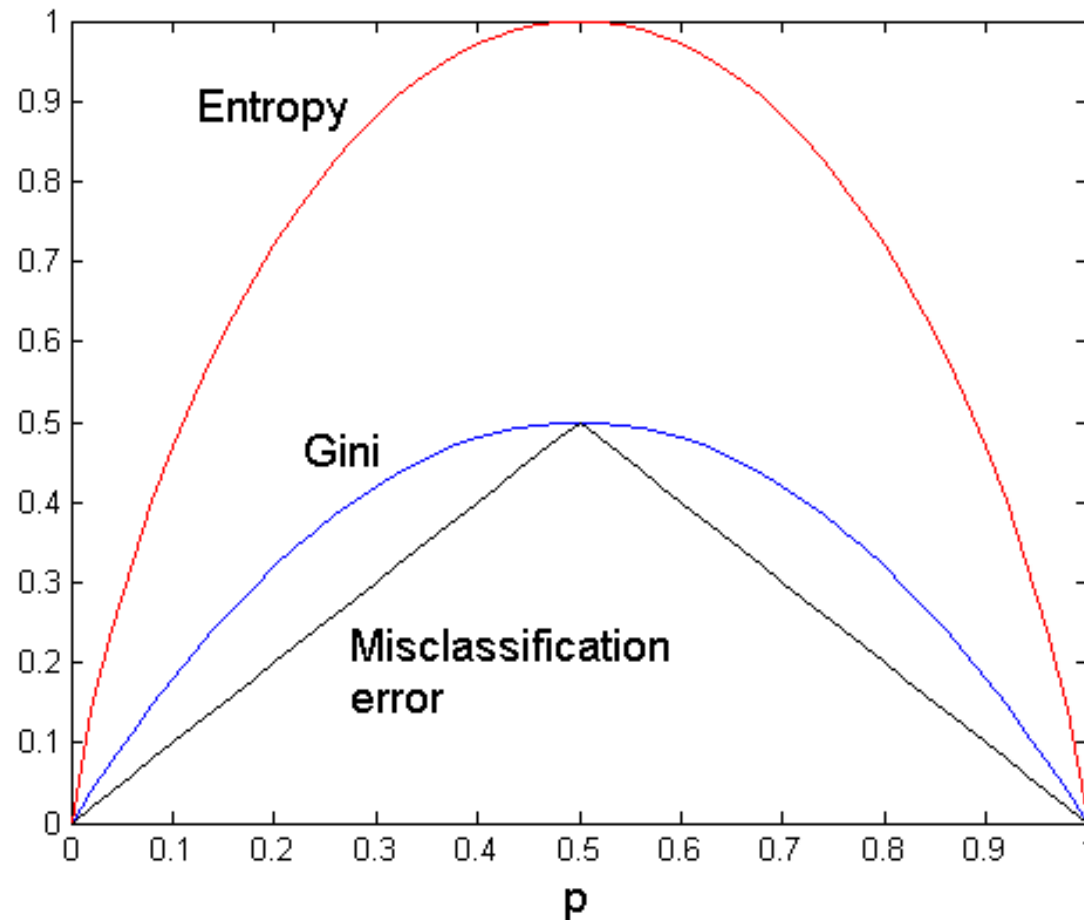
C1	2
C2	4

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

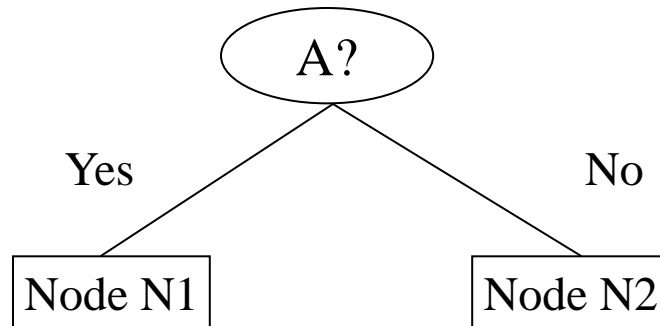
$$Error = 1 - \max(2/6, 4/6) = 1 - 4/6 = 1/3$$

Comparison among Impurity Measures

For a 2-class problem:



Misclassification Error vs Gini Index



	Parent
C1	7
C2	3
Gini = 0.42	

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

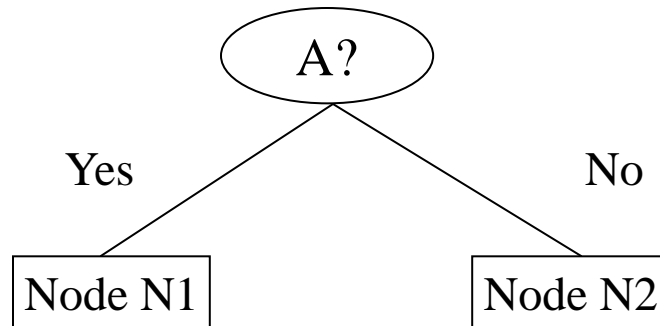
$$\begin{aligned}\text{Gini}(N2) &= 1 - (4/7)^2 - (3/7)^2 \\ &= 0.489\end{aligned}$$

	N1	N2
C1	3	4
C2	0	3
Gini=0.342		

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

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

Misclassification Error vs Gini Index



	Parent
C1	7
C2	3
Gini = 0.42	

	N1	N2
C1	3	4
C2	0	3
Gini=0.342		

	N1	N2
C1	3	4
C2	1	2
Gini=0.416		

Misclassification error for all three cases = 0.3 !

Decision Tree Based Classification

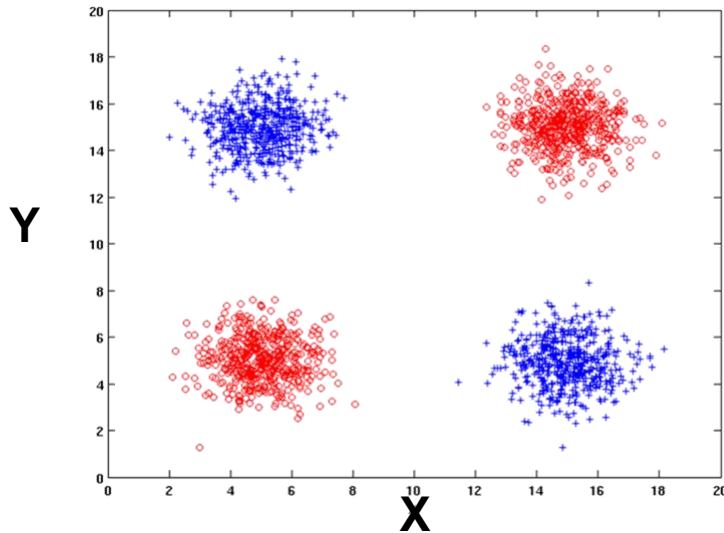
| Advantages:

- Relatively 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 attributes
- Can easily handle irrelevant attributes (unless the attributes are **interacting**)

| Disadvantages: .

- Due to the greedy nature of splitting criterion, **interacting** attributes (that can distinguish between classes together but not individually) may be passed over in favor of other attributed that are less discriminating.
- Each decision boundary involves only a single attribute

Handling interactions



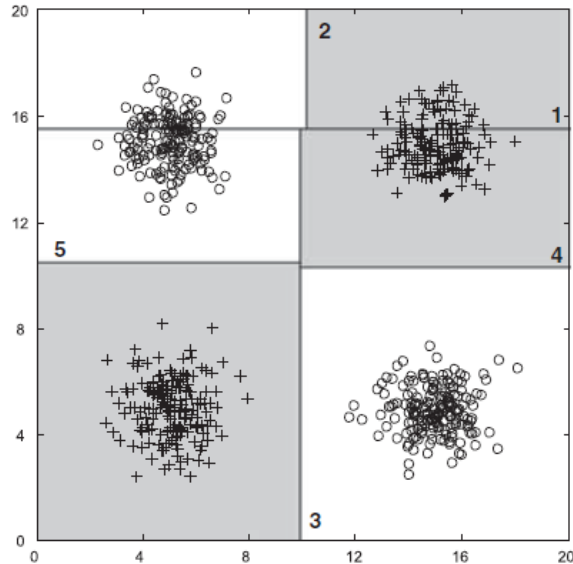
+ : 1000 instances

o : 1000 instances

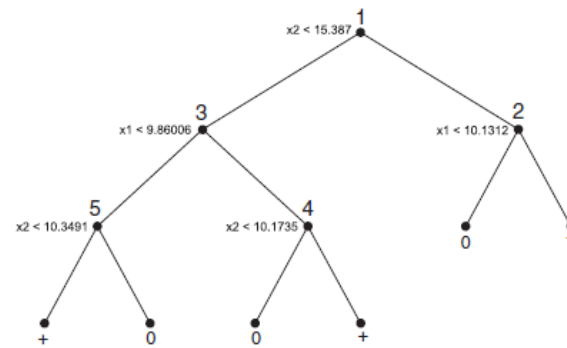
Entropy (X) : 0.99

Entropy (Y) : 0.99

Handling interactions



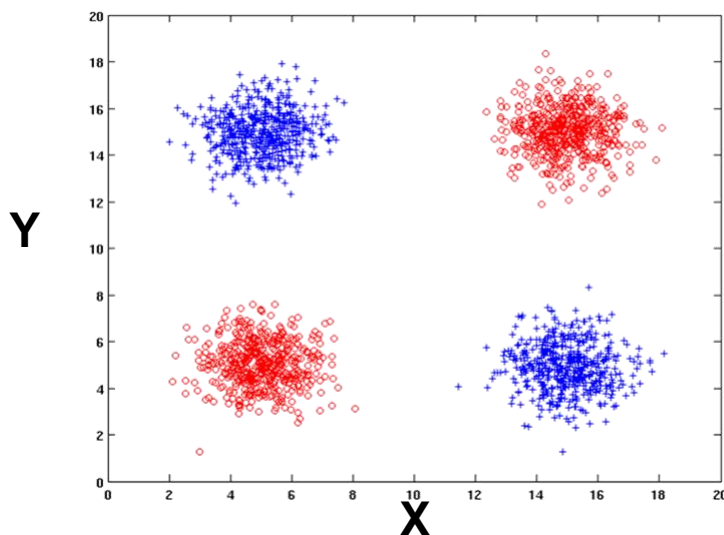
(a) Decision boundary for tree with 6 leaf nodes.



(b) Decision tree with 6 leaf nodes.

Figure 3.28. Decision tree with 6 leaf nodes using X and Y as attributes. Splits have been numbered from 1 to 5 in order of their occurrence in the tree.

Handling interactions given irrelevant attributes



+ : 1000 instances

o : 1000 instances

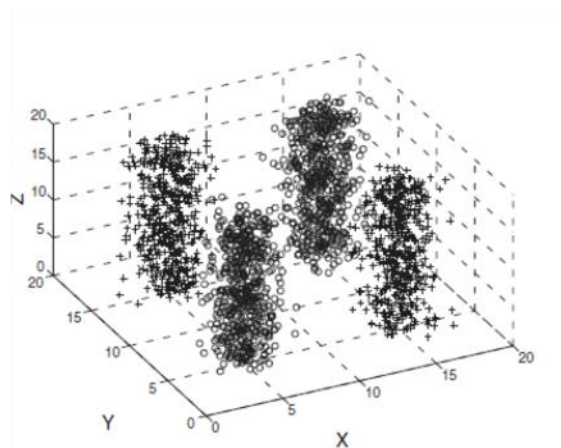
Adding Z as a noisy attribute generated from a uniform distribution

Entropy (X) : 0.99

Entropy (Y) : 0.99

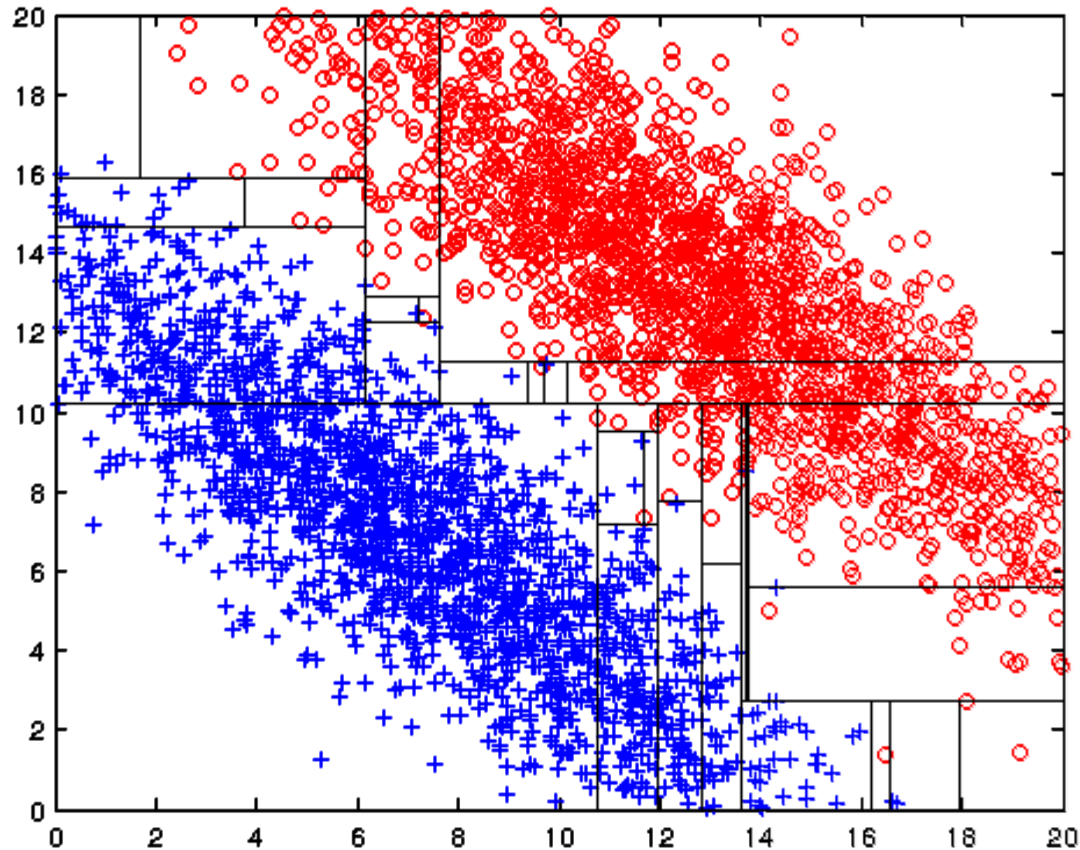
Entropy (Z) : 0.98

Attribute Z will be chosen for splitting!



(a) Three-dimensional data with attributes X, Y, and Z.

Limitations of single attribute-based decision boundaries



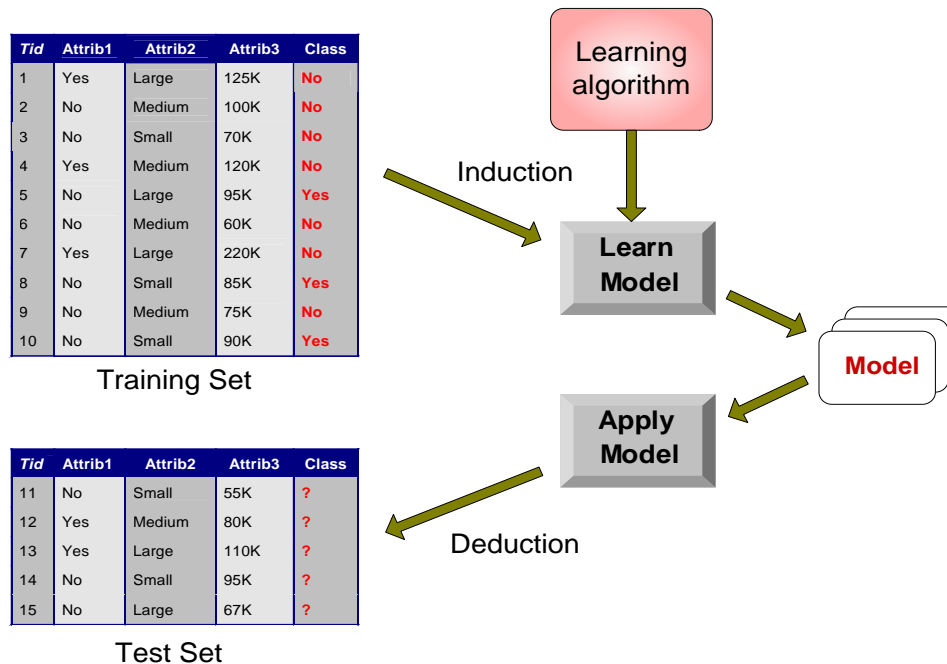
Both **positive (+)** and **negative (o)** classes generated from skewed Gaussians with centers at (8,8) and (12,12) respectively.

Model Overfitting

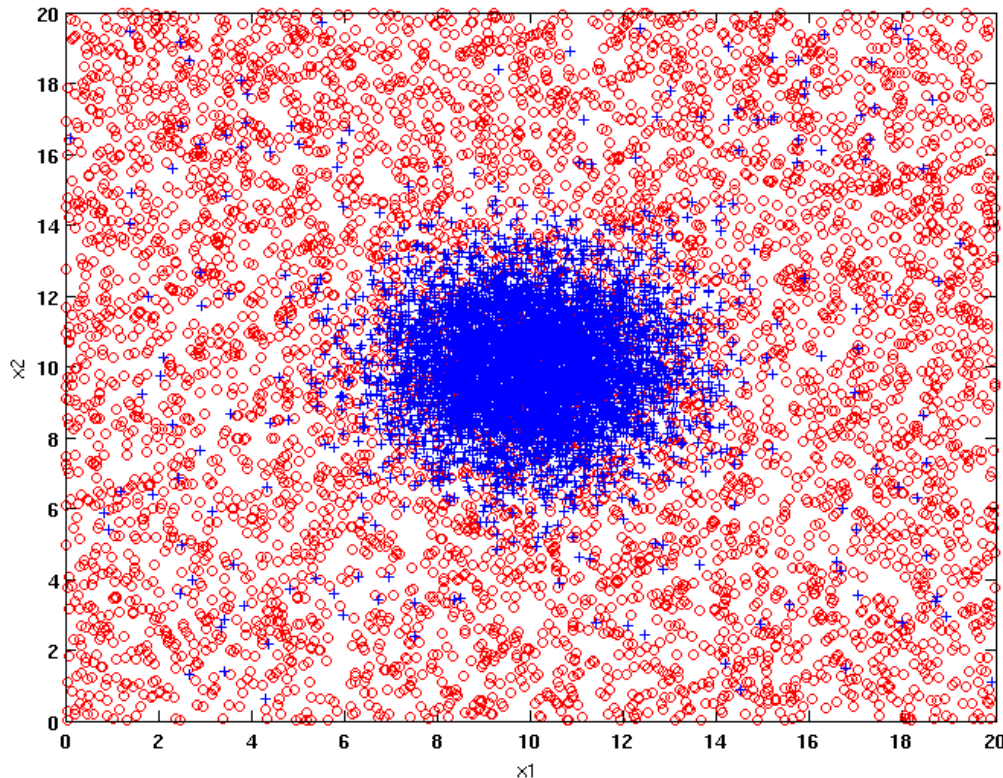
Introduction to Data Mining, 2nd Edition
by
Tan, Steinbach, Karpatne, Kumar

Classification Errors

- **Training errors:** Errors committed on the training set
- **Test errors:** Errors committed on the test set
- **Generalization errors:** Expected error of a model over random selection of records from same distribution



Example Data Set



Two class problem:

+ : 5400 instances

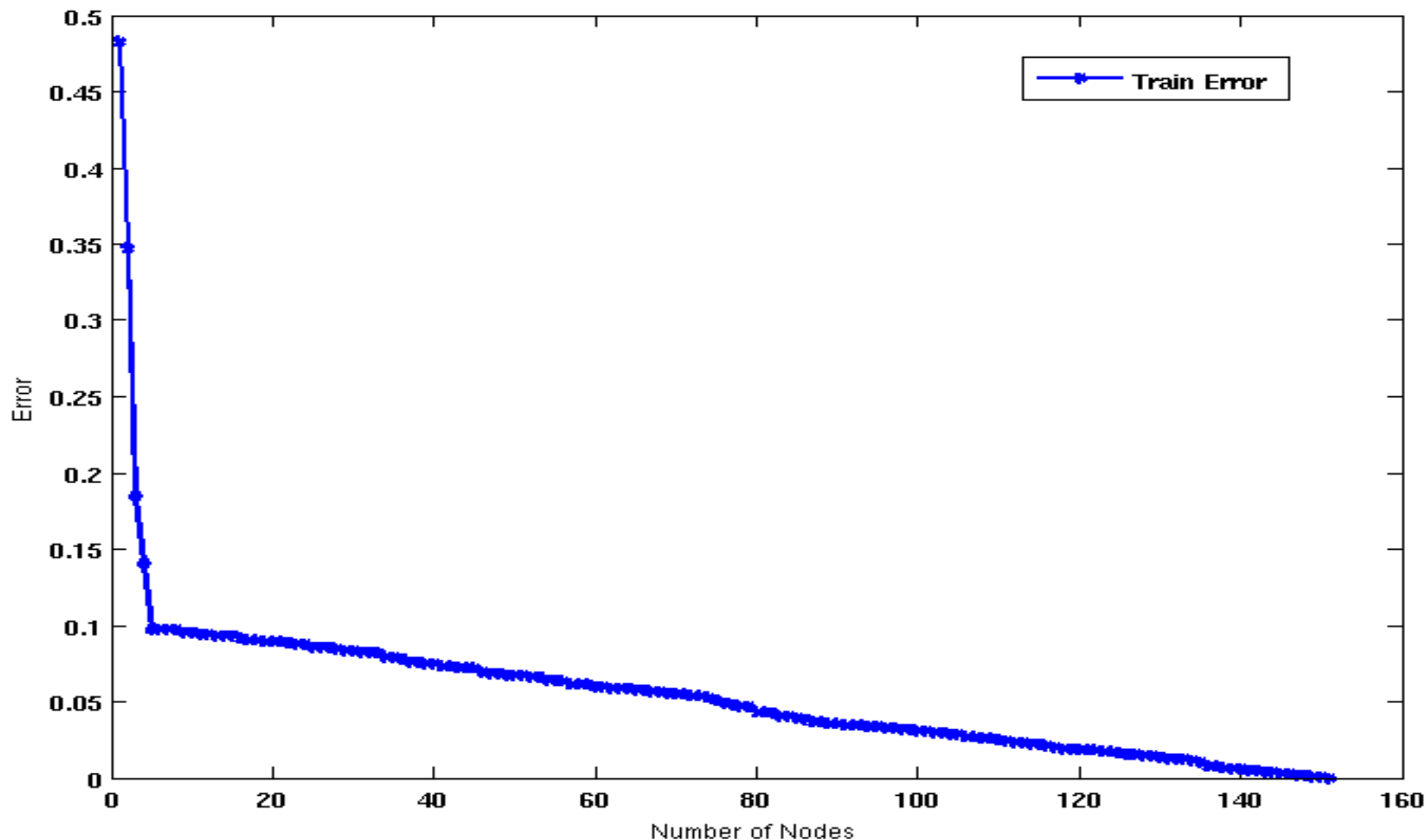
- 5000 instances generated from a Gaussian centered at (10,10)
- 400 noisy instances added

o : 5400 instances

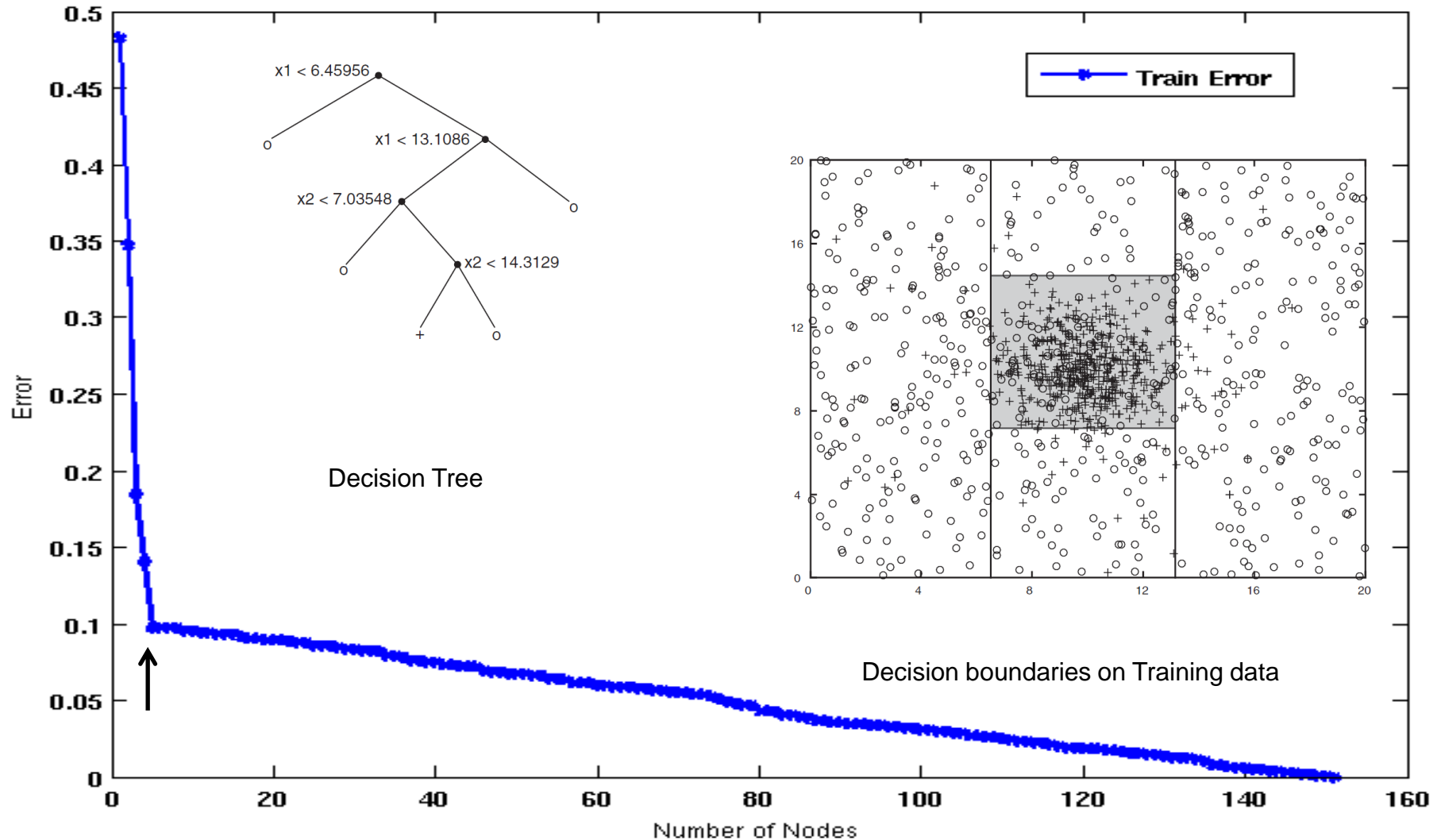
- Generated from a uniform distribution

10 % of the data used for training and 90% of the data used for testing

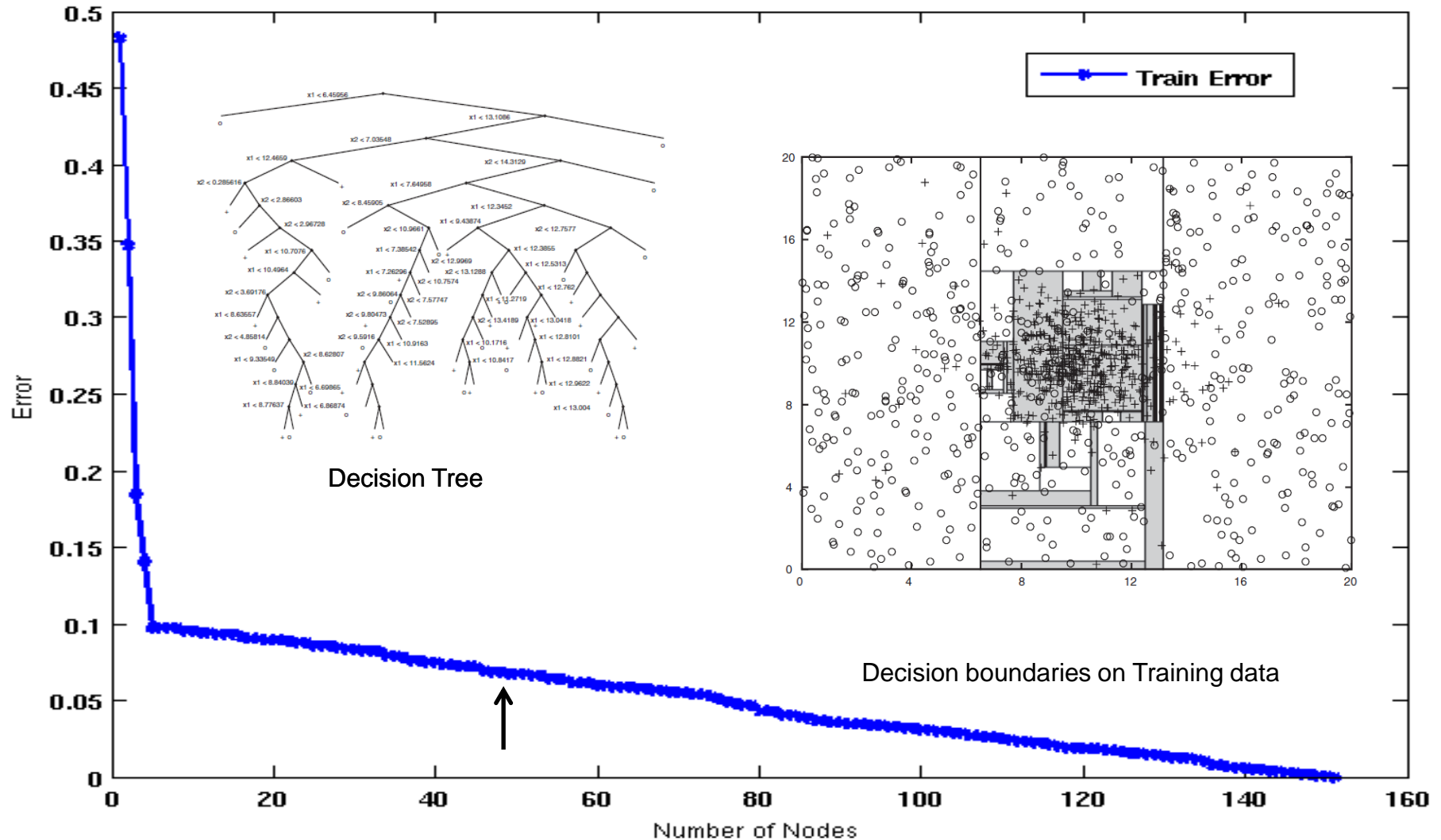
Increasing number of nodes in Decision Trees



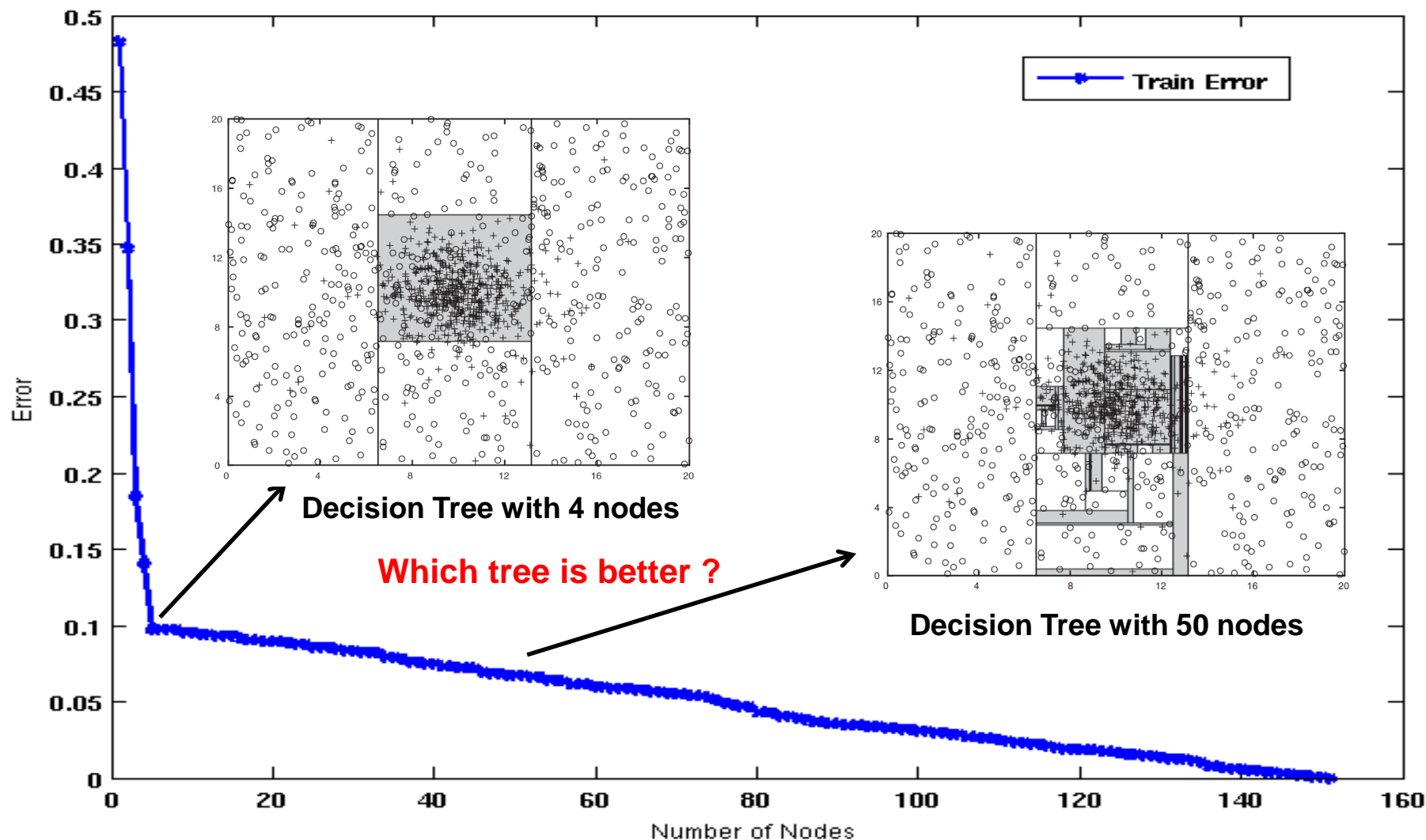
Decision Tree with 4 nodes



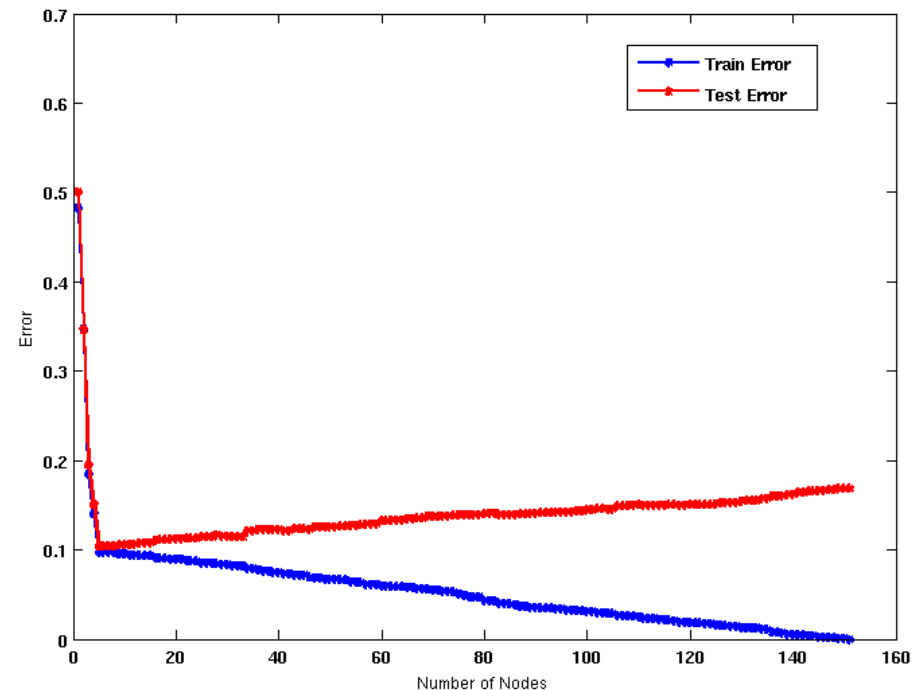
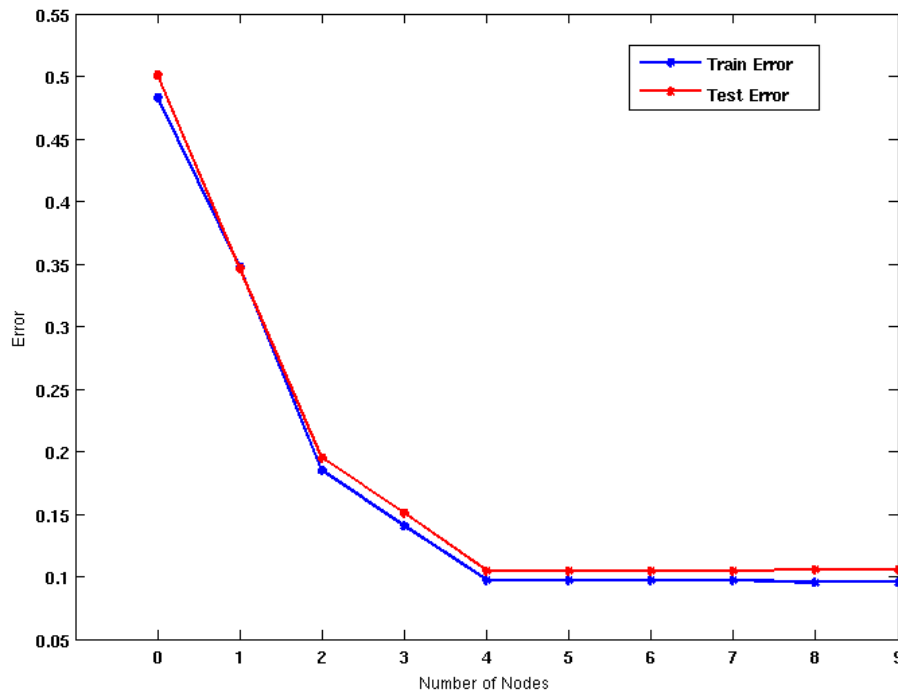
Decision Tree with 50 nodes



Which tree is better?



Model Underfitting and Overfitting

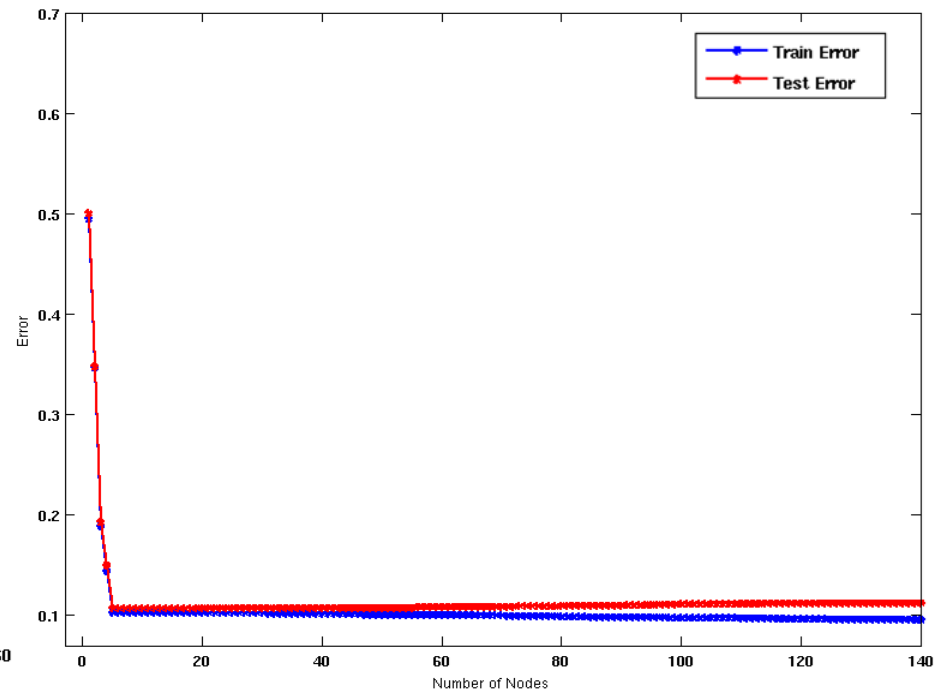
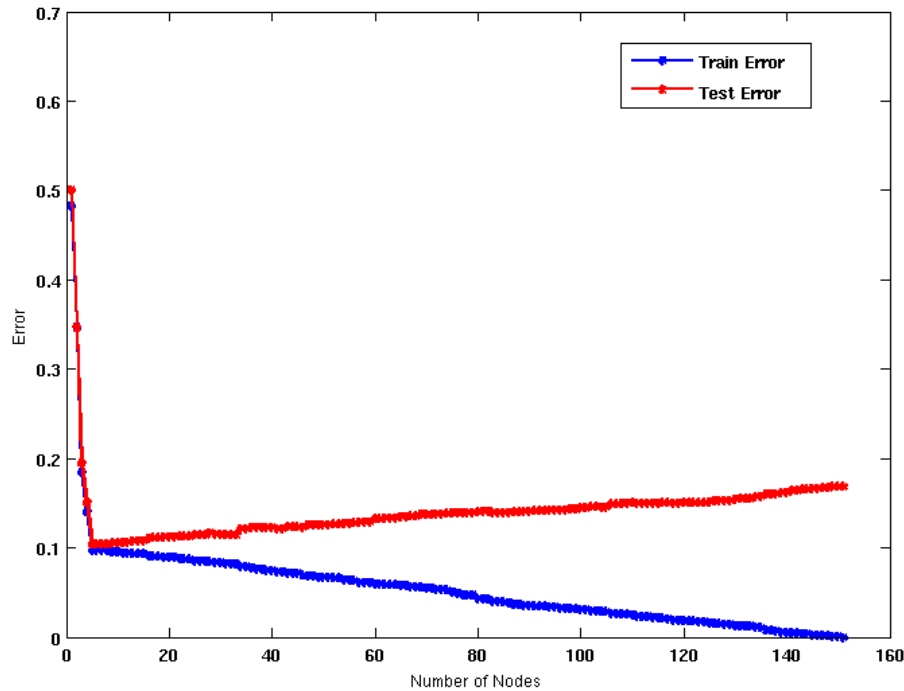


- As the model becomes more and more complex, test errors can start increasing even though training error may be decreasing

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

Overfitting: when model is too complex, training error is small but test error is large

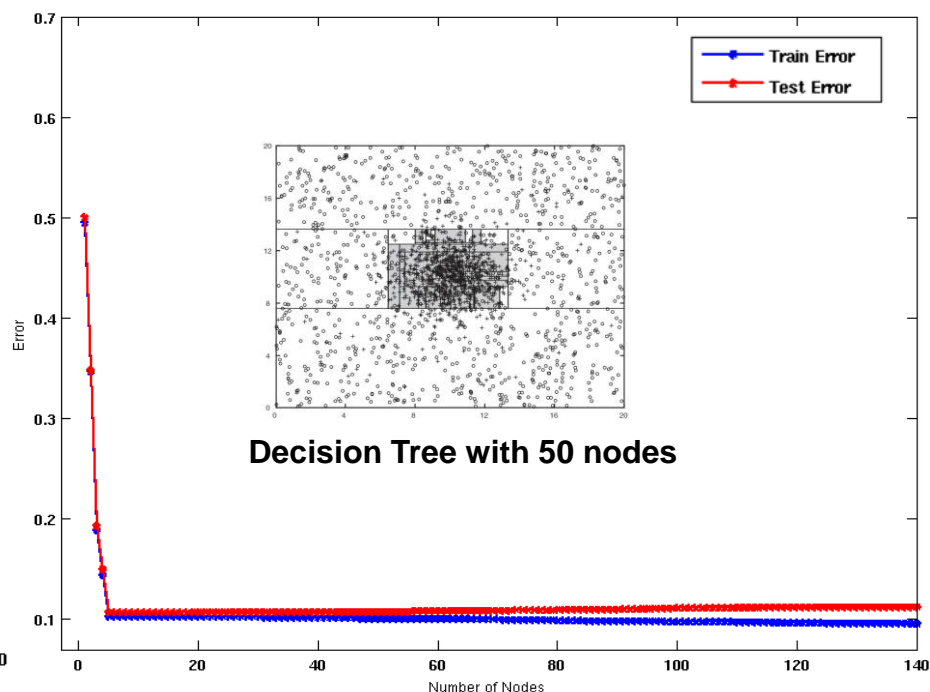
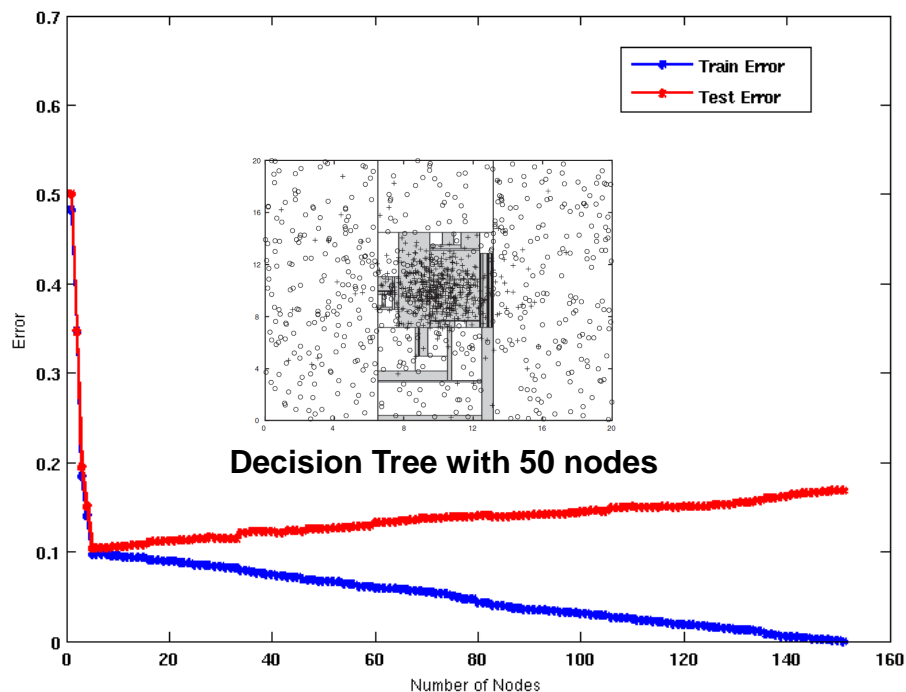
Model Overfitting – Impact of Training Data Size



Using twice the number of data instances

- Increasing the size of training data reduces the difference between training and testing errors at a given size of model

Model Overfitting – Impact of Training Data Size



Using twice the number of data instances

- Increasing the size of training data reduces the difference between training and testing errors at a given size of model

Reasons for Model Overfitting

- Not enough training data
- High model complexity
 - Multiple Comparison Procedure

Effect of Multiple Comparison Procedure

- Consider the task of predicting whether stock market will rise/fall in the next 10 trading days

- Random guessing:

$$P(\text{correct}) = 0.5$$

- Make 10 random guesses in a row:

$$P(\# \text{correct} \geq 8) = \frac{\binom{10}{8} + \binom{10}{9} + \binom{10}{10}}{2^{10}} = 0.0547$$

Day 1	Up
Day 2	Down
Day 3	Down
Day 4	Up
Day 5	Down
Day 6	Down
Day 7	Up
Day 8	Up
Day 9	Up
Day 10	Down

Effect of Multiple Comparison Procedure

- Approach:
 - Get 50 analysts
 - Each analyst makes 10 random guesses
 - Choose the analyst that makes the most number of correct predictions

- Probability that at least one analyst makes at least 8 correct predictions

$$P(\#correct \geq 8) = 1 - (1 - 0.0547)^{50} = 0.9399$$

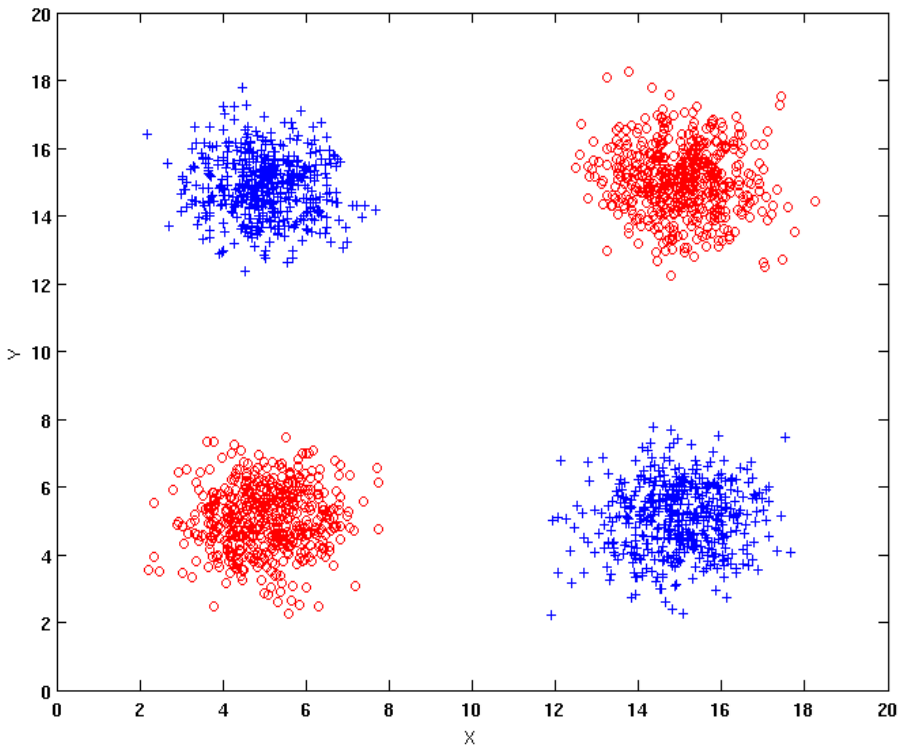
Effect of Multiple Comparison Procedure

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

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

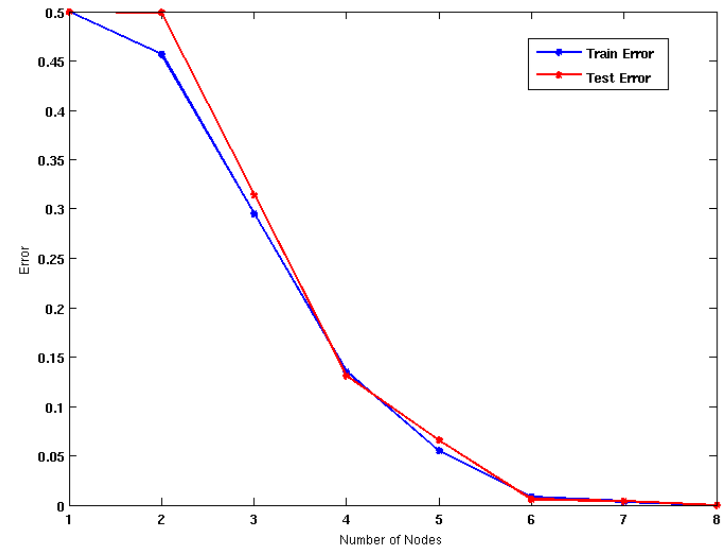
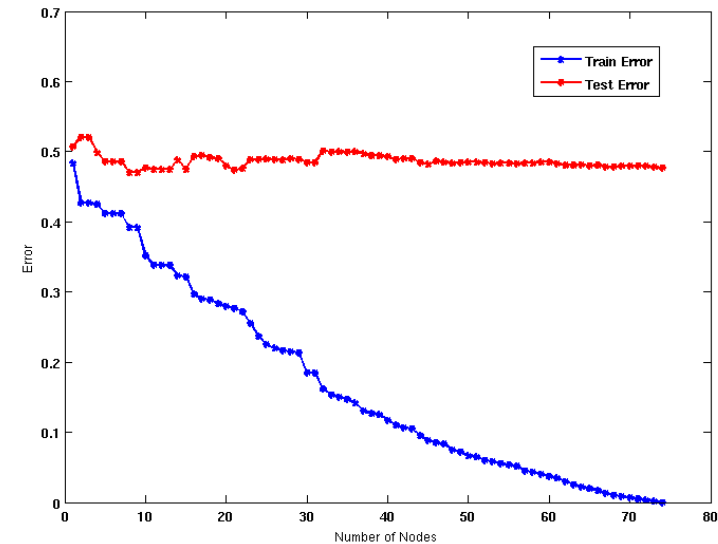
- If many alternatives are available, one may inadvertently add irrelevant components to the model, resulting in model overfitting

Effect of Multiple Comparison - Example



Use additional 100 noisy variables generated from a uniform distribution along with X and Y as attributes.

Use 30% of the data for training and 70% of the data for testing



Using only X and Y as attributes

Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary
- Training error does not provide a good estimate of how well the tree will perform on previously unseen records
- Need ways for estimating generalization errors

Model Selection

- Performed during model building
- Purpose is to ensure that model is not overly complex (to avoid overfitting)
- Need to estimate generalization error
 - Using Validation Set
 - Incorporating Model Complexity

Using Validation Set

- Divide training data into two parts:
 - Training set:
 - ◆ use for model building
 - Validation set:
 - ◆ use for estimating generalization error
 - ◆ Note: validation set is not the same as test set

- Drawback:
 - Less data available for training

Incorporating Model Complexity

□ Rationale: Occam's Razor

- Given two models of similar generalization errors, one should prefer the simpler model over the more complex model
- A complex model has a greater chance of being fitted accidentally
- Therefore, one should include model complexity when evaluating a model

$$\text{Gen. Error}(\text{Model}) = \text{Train. Error}(\text{Model}, \text{Train. Data}) + \alpha \times \text{Complexity}(\text{Model})$$

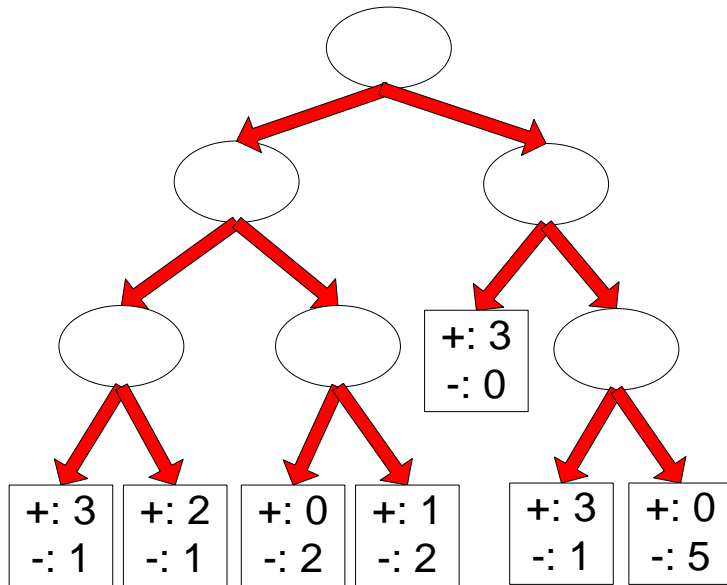
Estimating the Complexity of Decision Trees

- **Pessimistic Error Estimate** of decision tree T with k leaf nodes:

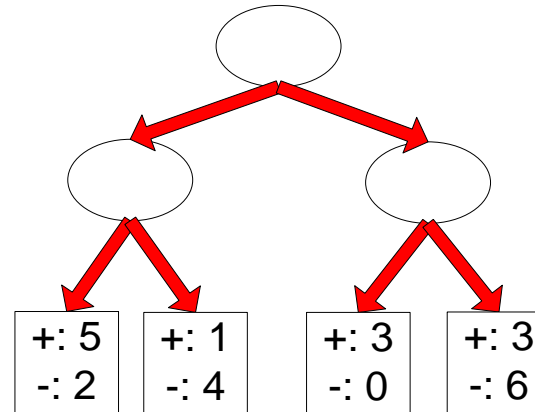
$$err_{gen}(T) = err(T) + \Omega \times \frac{k}{N_{train}}$$

- $err(T)$: error rate on all training records
- Ω : trade-off hyper-parameter (similar to α)
 - ◆ Relative cost of adding a leaf node
- k : number of leaf nodes
- N_{train} : total number of training records

Estimating the Complexity of Decision Trees: Example



Decision Tree, T_L



Decision Tree, T_R

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

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

$$\Omega = 1$$

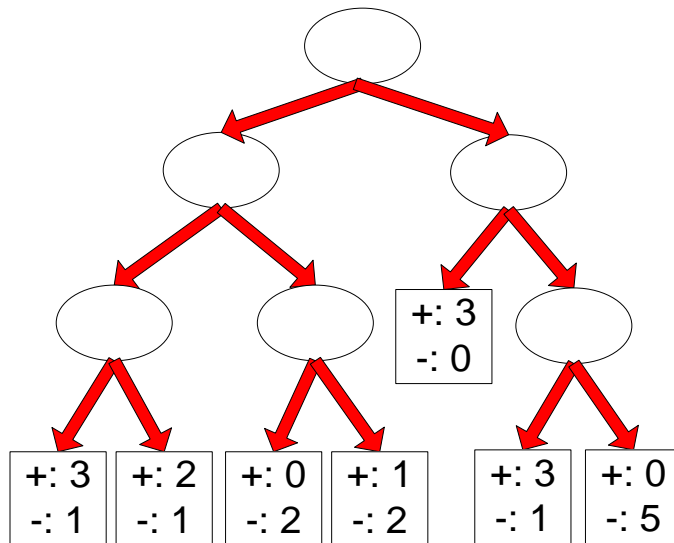
$$e_{\text{gen}}(T_L) = 4/24 + 1 \cdot 7/24 = 11/24 = 0.458$$

$$e_{\text{gen}}(T_R) = 6/24 + 1 \cdot 4/24 = 10/24 = 0.417$$

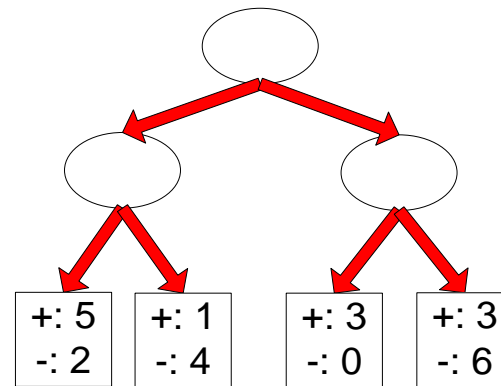
Estimating the Complexity of Decision Trees

□ Resubstitution Estimate:

- Using training error as an **optimistic** estimate of generalization error
- Referred to as **optimistic error** estimate



Decision Tree, T_L



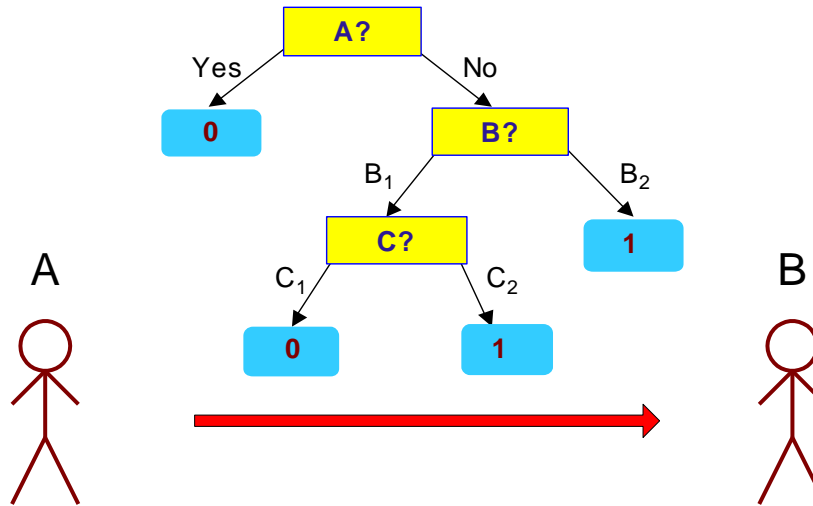
Decision Tree, T_R

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

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

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	?

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

Model Selection for Decision Trees

□ 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 χ^2 test)
 - ◆ Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).
 - ◆ Stop if estimated generalization error falls below certain threshold

Model Selection for Decision Trees

□ Post-pruning

- Grow decision tree to its entirety
- Subtree replacement
 - ◆ 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

Example of Post-Pruning

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

Training Error (Before splitting) = 10/30

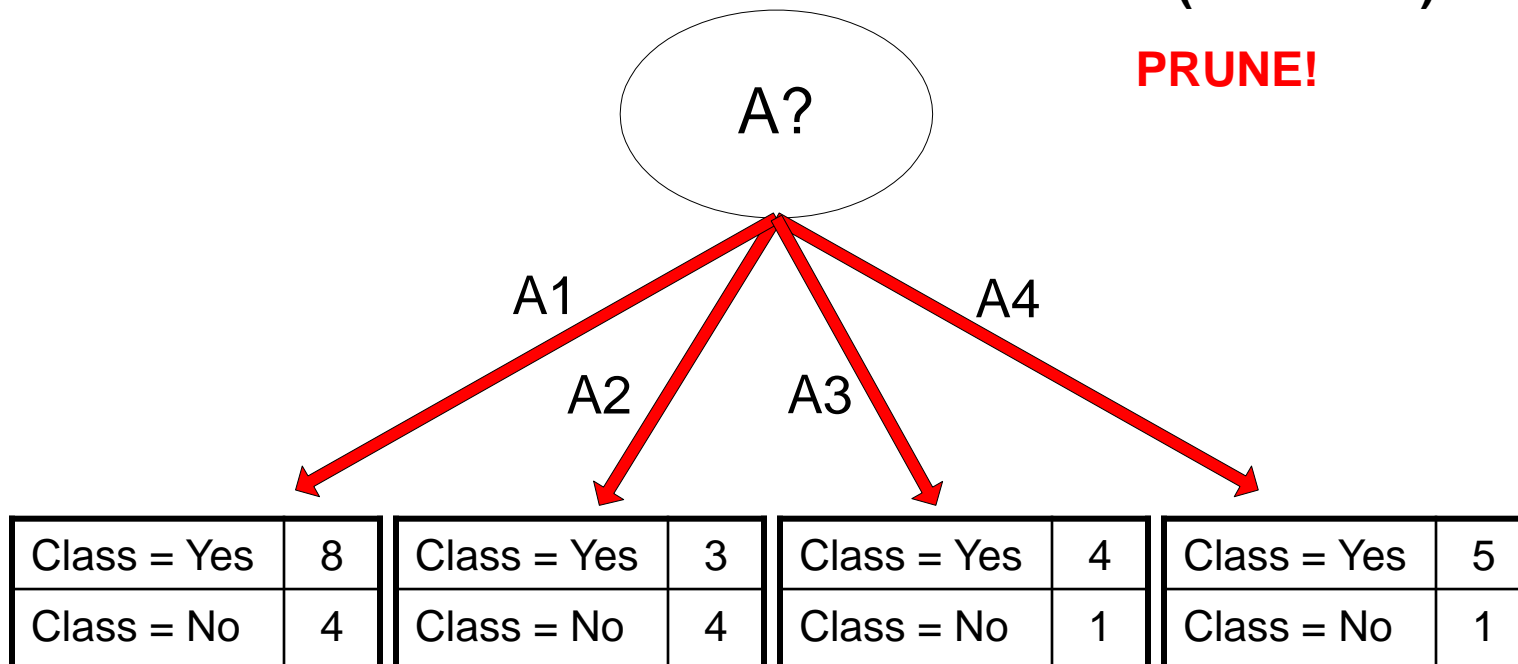
Pessimistic error = $(10 + 0.5)/30 = 10.5/30$

Training Error (After splitting) = 9/30

Pessimistic error (After splitting)

$= (9 + 4 \times 0.5)/30 = 11/30$

PRUNE!



Examples of Post-pruning

Decision Tree:

```

depth = 1 :
| breadth > 7 : class 1
| breadth <= 7 :
| | breadth <= 3 :
| | | ImagePages > 0.375 : class 0
| | | ImagePages <= 0.375 :
| | | | totalPages <= 6 : class 1
| | | | totalPages > 6 :
| | | | | breadth <= 1 : class 1
| | | | | breadth > 1 : class 0
| | width > 3 :
| | | MultiIP = 0:
| | | | ImagePages <= 0.1333 : class 1
| | | | ImagePages > 0.1333 :
| | | | | breadth <= 6 : class 0
| | | | | breadth > 6 : class 1
| | | MultiIP = 1:
| | | | TotalTime <= 361 : class 0
| | | | TotalTime > 361 : class 1
| depth > 1 :
| | MultiAgent = 0:
| | | depth > 2 : class 0
| | | depth <= 2 :
| | | | MultiIP = 1: class 0
| | | | MultiIP = 0:
| | | | | breadth <= 6 : class 0
| | | | | breadth > 6 :
| | | | | | RepeatedAccess <= 0.0322 : class 0
| | | | | | RepeatedAccess > 0.0322 : class 1
| | | MultiAgent = 1:
| | | | totalPages <= 81 : class 0
| | | | totalPages > 81 : class 1

```

Subtree
Raising

Simplified Decision Tree:

```

depth = 1 :
| ImagePages <= 0.1333 : class 1
| ImagePages > 0.1333 :
| | breadth <= 6 : class 0
| | breadth > 6 : class 1
depth > 1 :
| MultiAgent = 0: class 0
| MultiAgent = 1:
| | totalPages <= 81 : class 0
| | totalPages > 81 : class 1

```

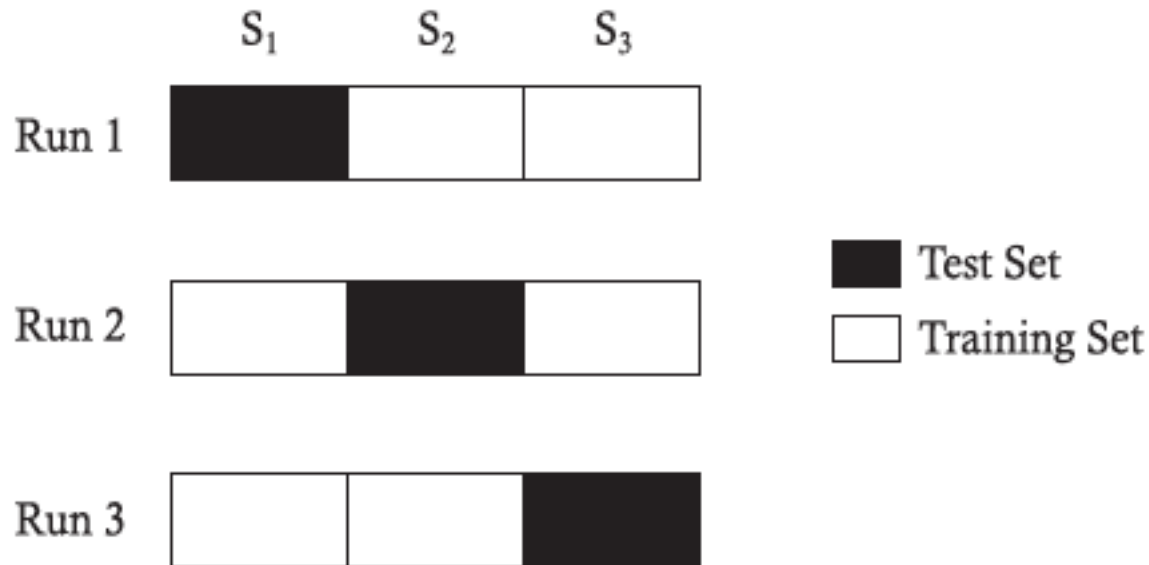
Subtree
Replacement

Model Evaluation

- Purpose:
 - To estimate performance of classifier on previously unseen data (test set)
- Holdout
 - Reserve $k\%$ for training and $(100-k)\%$ 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: $k=n$

Cross-validation Example

□ 3-fold cross-validation



Variations on Cross-validation

- Repeated cross-validation
 - Perform cross-validation a number of times
 - Gives an estimate of the variance of the generalization error
- Stratified cross-validation
 - Guarantee the same percentage of class labels in training and test
 - Important when classes are imbalanced and the sample is small
- Use nested cross-validation approach for model selection and evaluation