

DATA MINING 1

Decision Tree Classifiers

Dino Pedreschi, Riccardo Guidotti

Revisited slides from Lecture Notes for Chapter 3 “Introduction to Data Mining”, 2nd Edition by Tan, Steinbach, Karpatne, Kumar



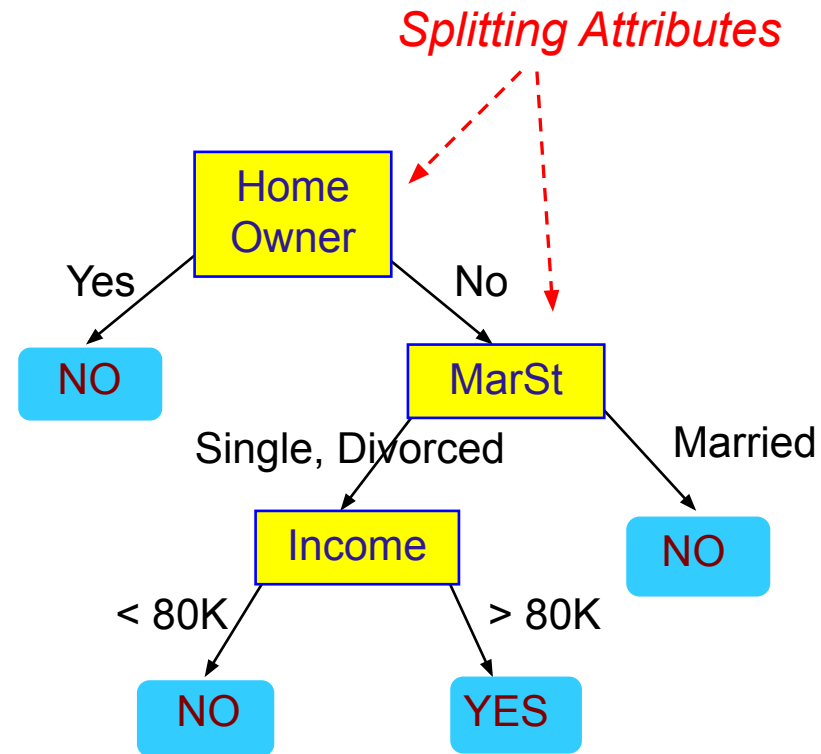
Example of a Decision Tree

Consider the problem of predicting whether a loan borrower will repay the loan or default on the loan payments.

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

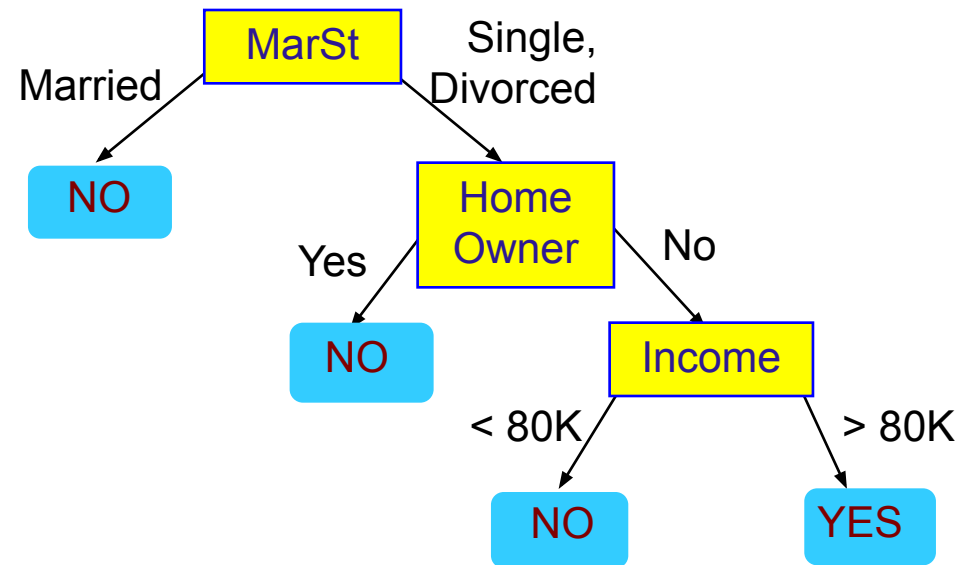
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

categorical

categorical

continuous

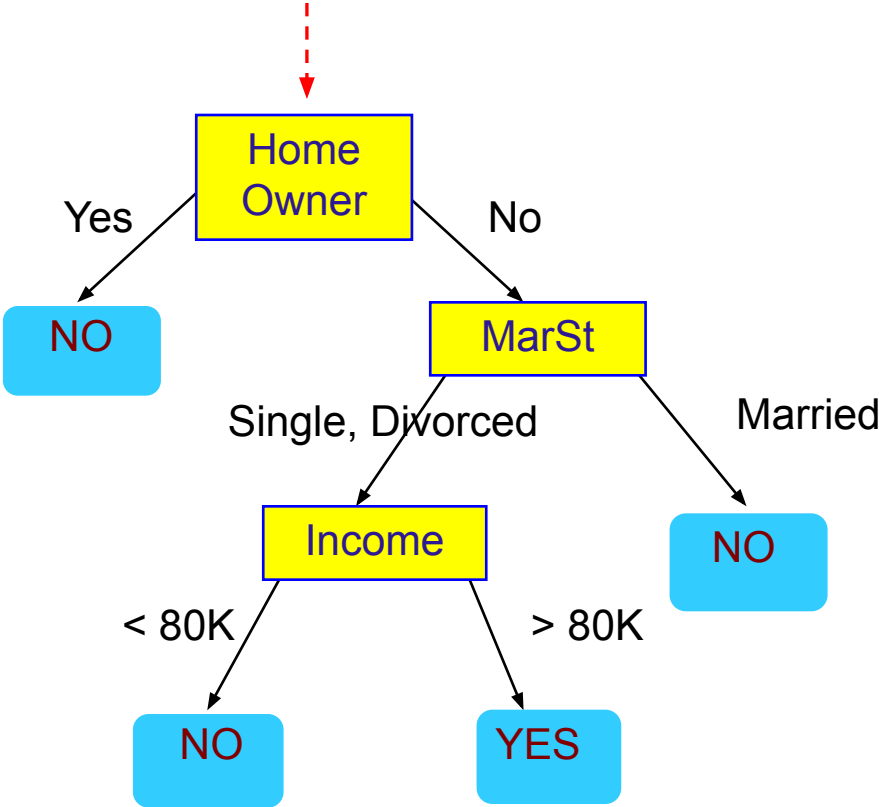
class



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

Apply Model to Test Data

Start from the root of tree.



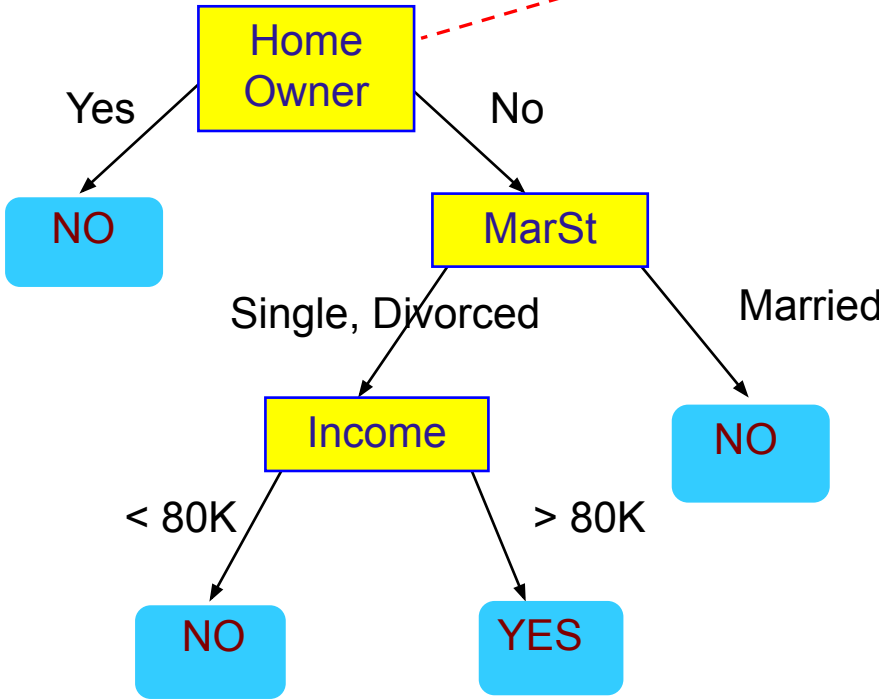
Test Data

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

Apply Model to Test Data

Test Data

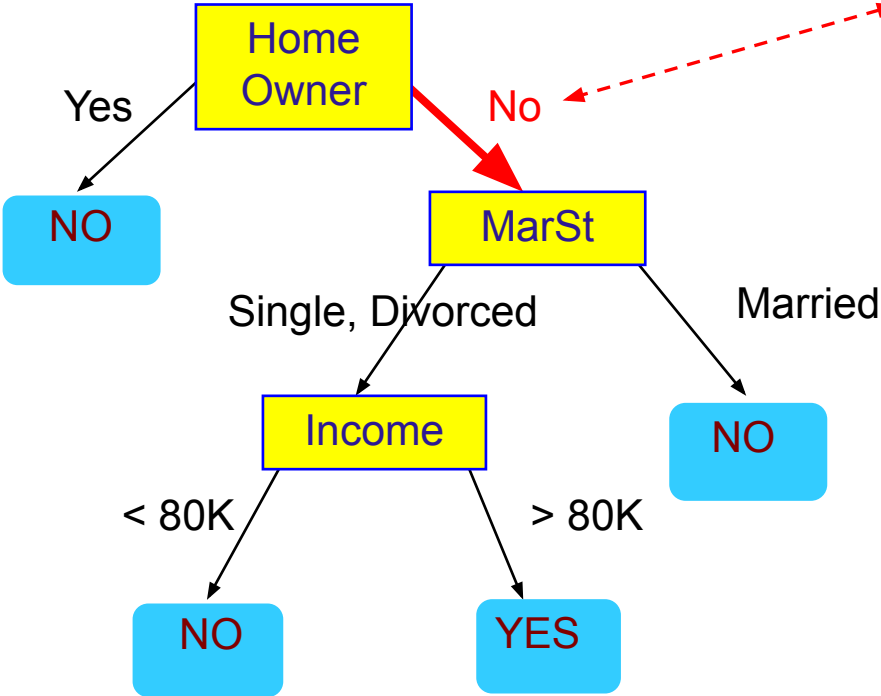
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply Model to Test Data

Test Data

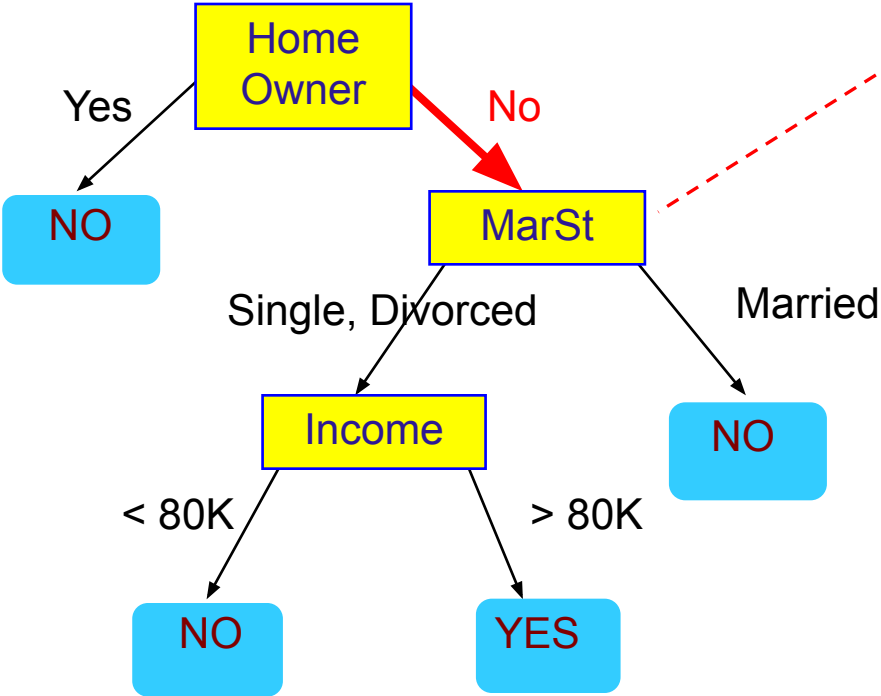
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply Model to Test Data

Test Data

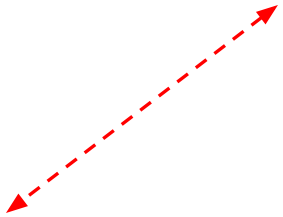
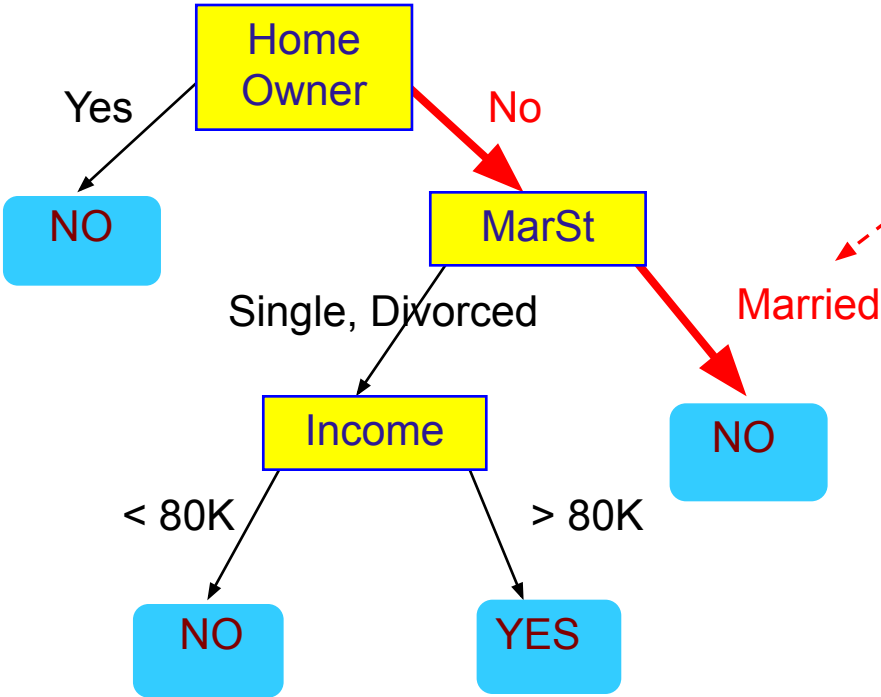
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply Model to Test Data

Test Data

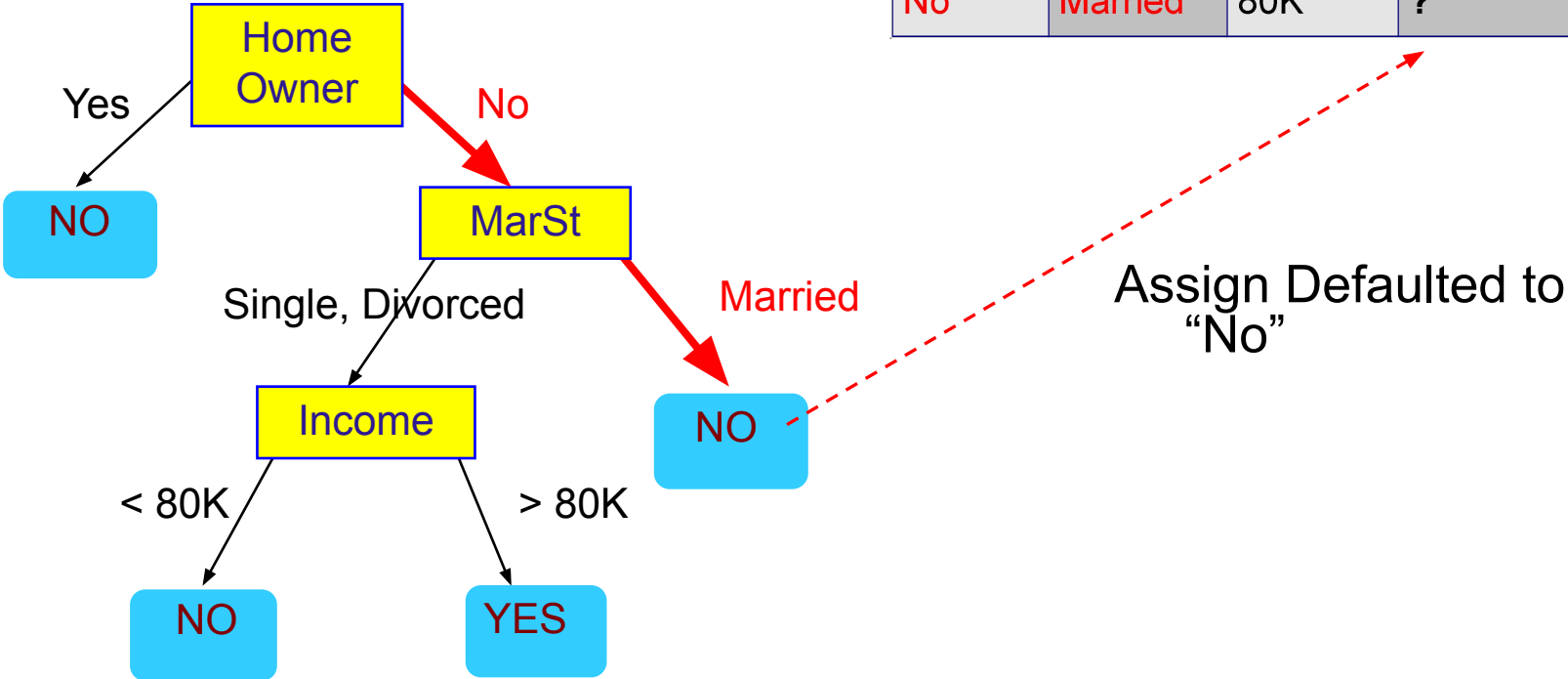
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply Model to Test Data

Test Data

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



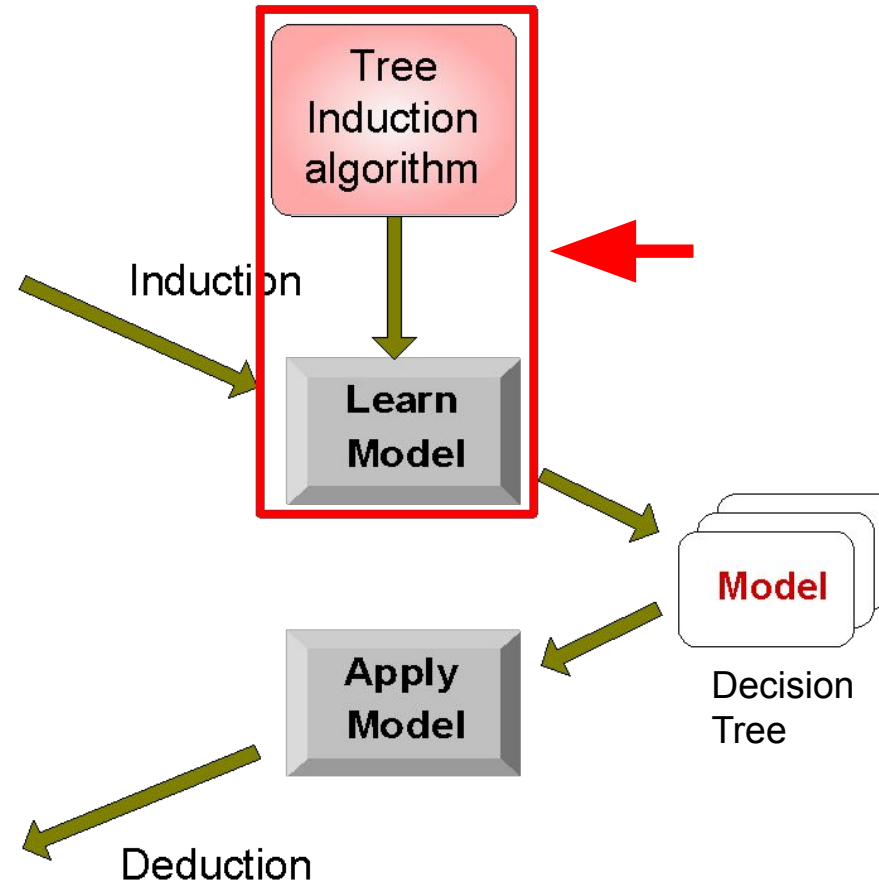
Decision Tree Classification Task

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



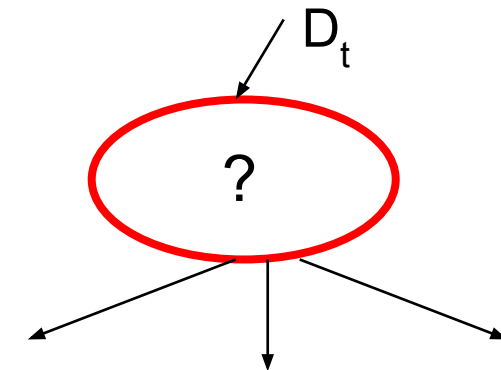
Decision Tree Induction

- 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)

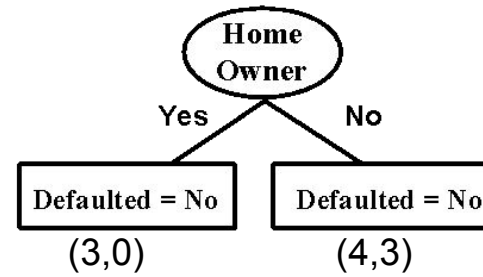
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Hunt's Algorithm

Defaulted = No

(7,3)

(a)



(b)

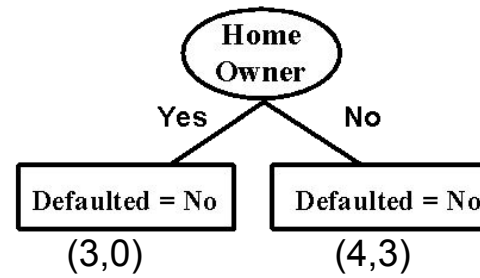
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Hunt's Algorithm

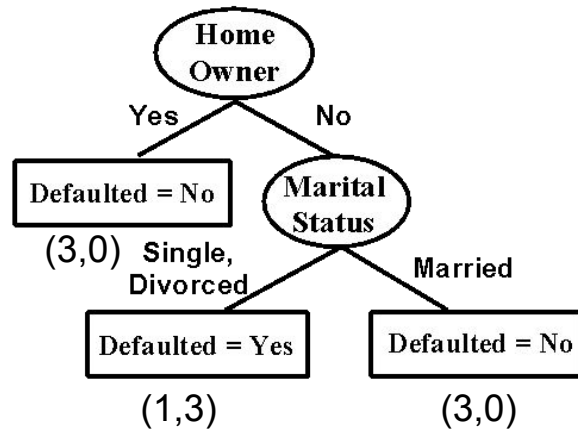
Defaulted = No

(7,3)

(a)



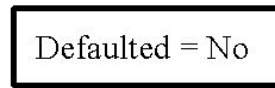
(b)



(c)

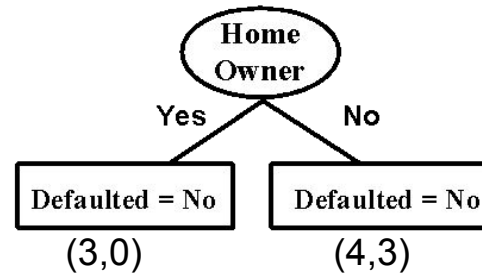
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Hunt's Algorithm

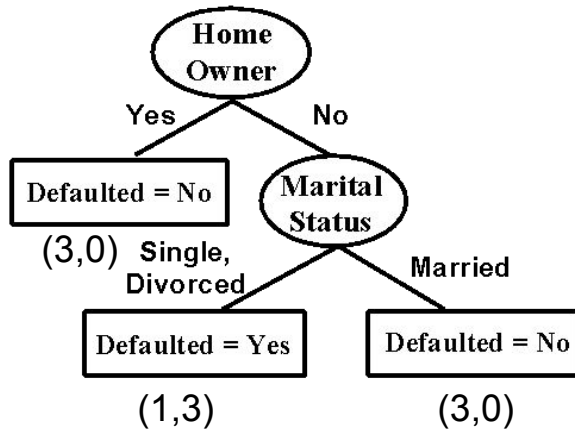


(7,3)

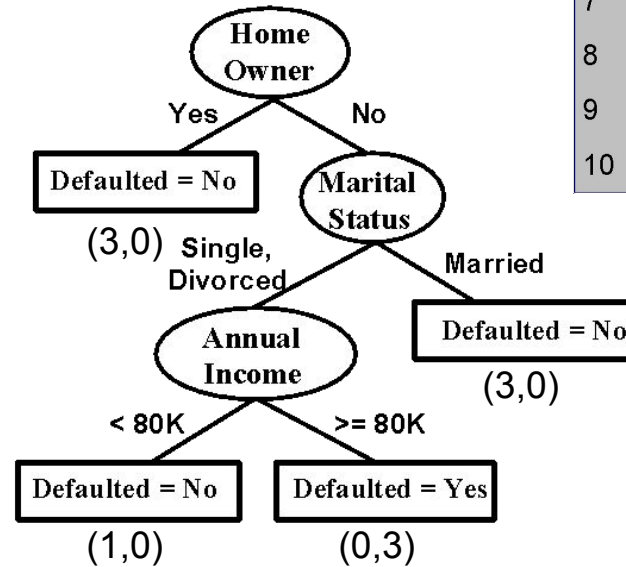
(a)



(b)



(c)



(d)

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

Design Issues of Decision Tree Induction

- **Greedy strategy:**

- the number of possible decision trees can be very large, many decision tree algorithms employ a heuristic-based approach to guide their search in the vast hypothesis space.
- **Split the records based on an attribute test that optimizes certain criterion.**

Tree Induction

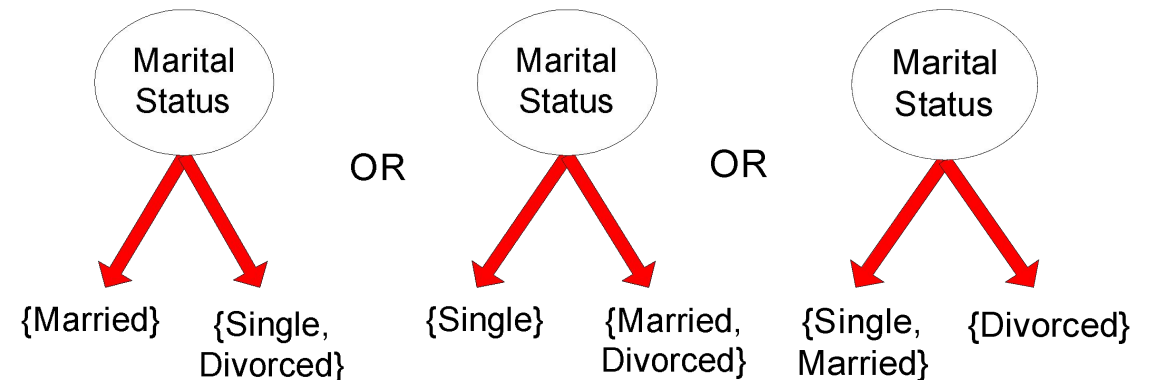
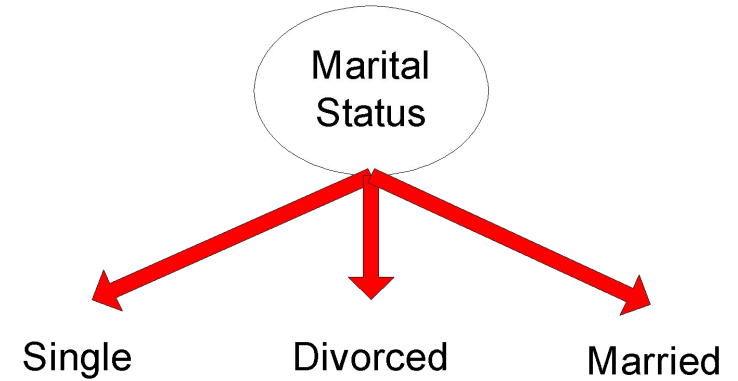
- How should training records be split?
 - Method for specifying test condition depending on attribute types
 - Measure for evaluating the goodness of a test condition
- How should the splitting procedure stop?
 - Stop splitting if all the records belong to the same class or have identical attribute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

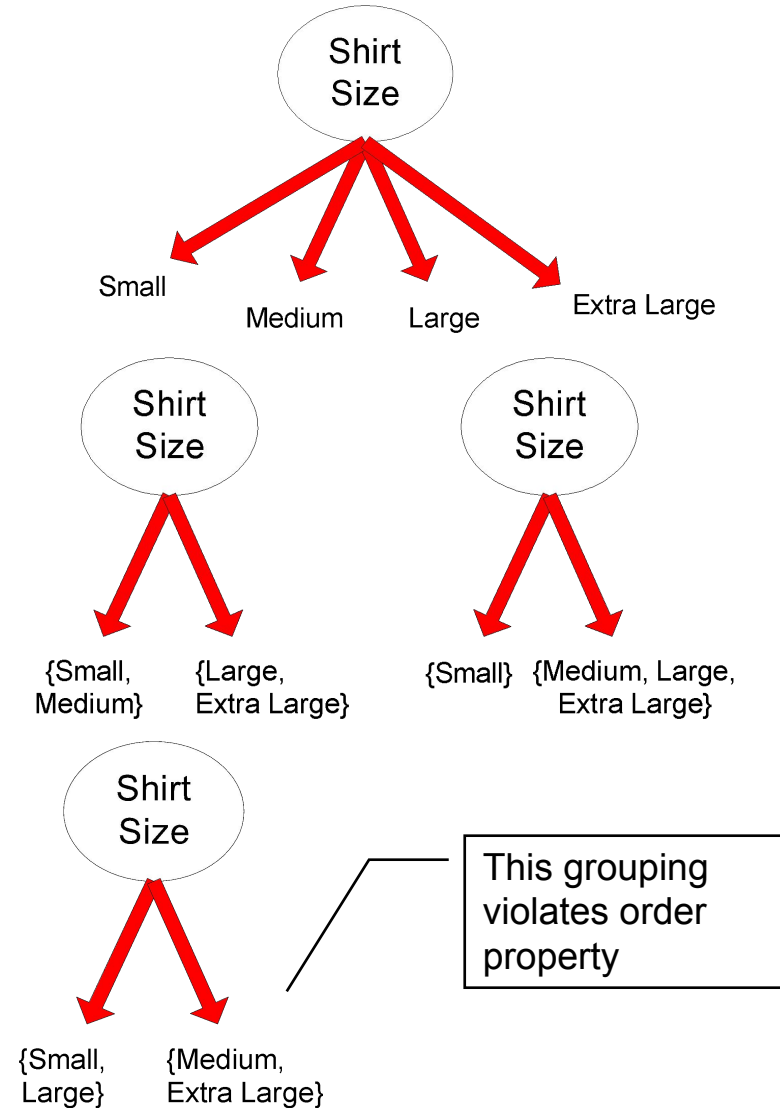
Test Condition for Nominal Attributes

- **Multi-way split:**
 - Use as many partitions as distinct values.
- **Binary split:**
 - Divides values into two subsets

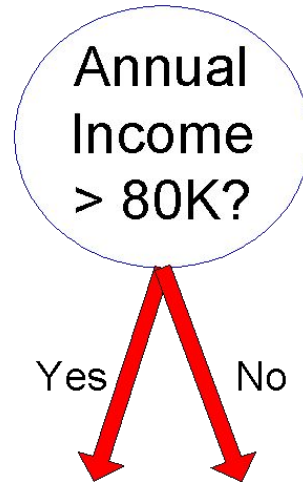


Test Condition for Ordinal Attributes

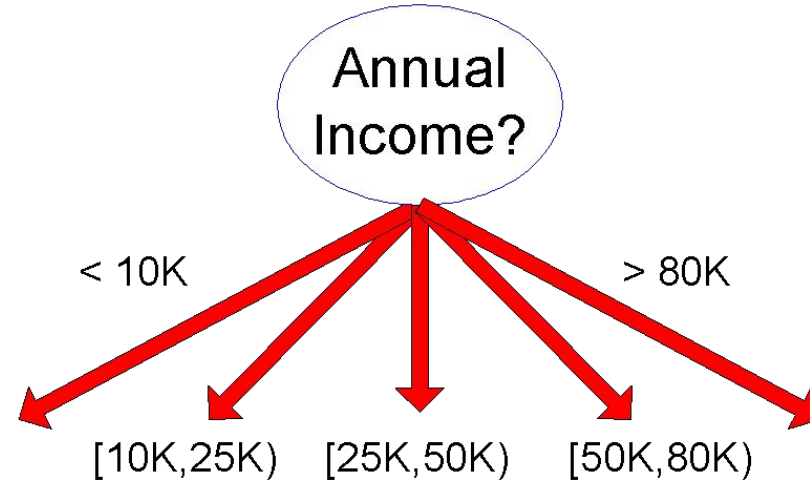
- **Multi-way split:**
 - Use as many partitions as distinct values
- **Binary split:**
 - Divides values into two subsets
 - Preserve order property among attribute values



Test Condition for Continuous Attributes



(i) Binary split



(ii) Multi-way split

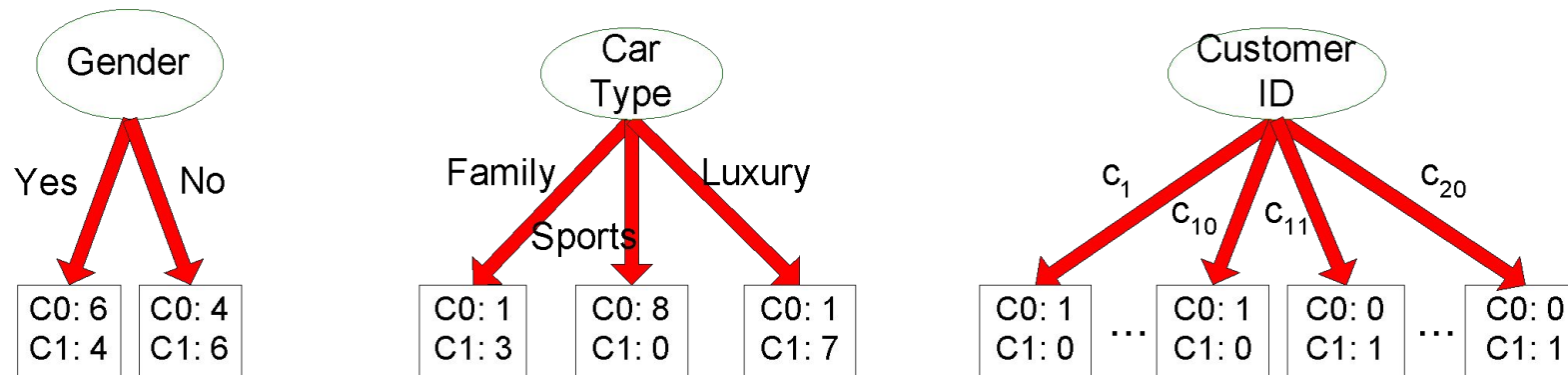
Splitting Based on Continuous Attributes

- 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

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

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



Which test condition is the best?

How to determine the Best Split

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

C0: 5
C1: 5

High degree of impurity,
Non-homogeneous

C0: 9
C1: 1

Low degree of impurity,
Homogeneous

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

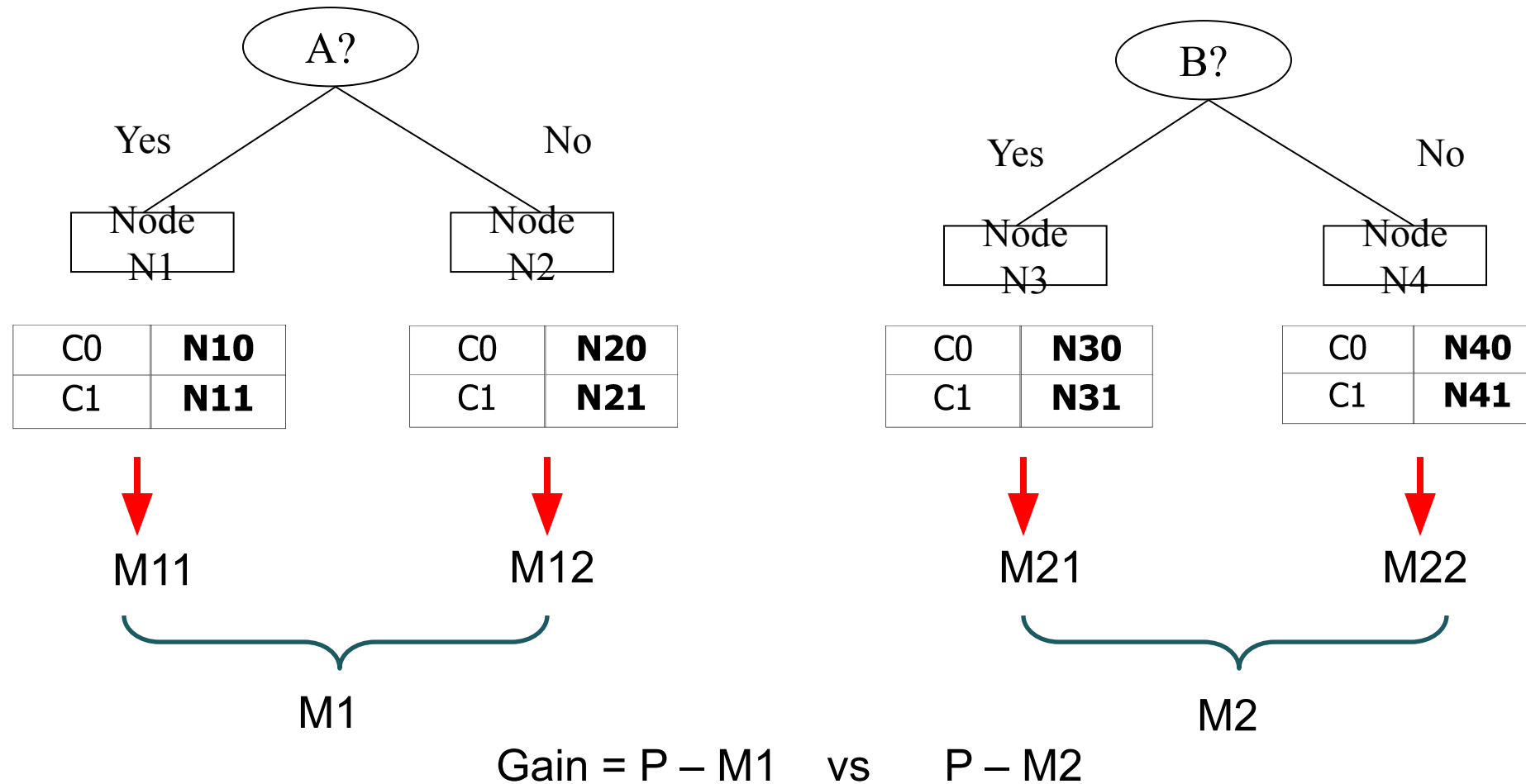
1. Compute impurity measure (P) before splitting
2. Compute impurity measure (M) after splitting
 - Compute impurity measure of each child node
 - M is the weighted impurity of children
3. Choose the attribute test condition that produces the highest gain (Gain = $P - M$) or equivalently, lowest impurity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

Finding the Best Split

Before Splitting:

C0	N00
C1	N01

→ P



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	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

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

C1	0
C2	6

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

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

C1	1
C2	5

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

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

C1	2
C2	4

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

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

Gini Index for a Collection of Nodes

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

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

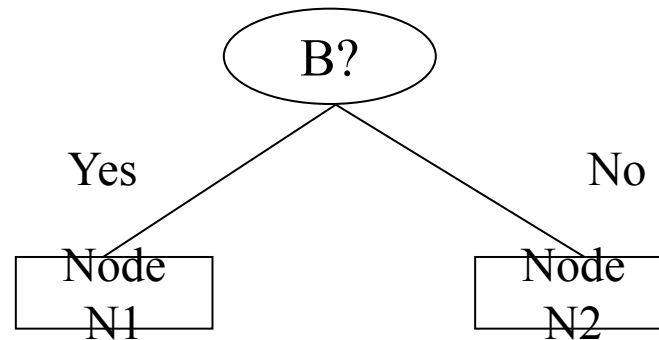
where, n_i = number of records at child i ,

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	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		

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

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

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	

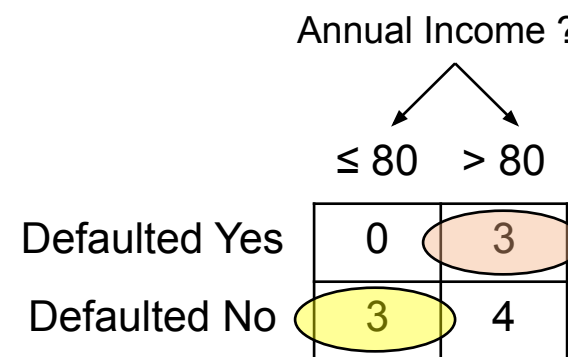
	CarType	
	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10
Gini	0.167	

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! ($O(N^2)$) 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 $O(N \log N)$: 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...

- 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		
		Annual 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							
	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

↓

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No												
Annual 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		
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											

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										
Annual 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	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	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	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

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

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

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

- Information Gain:

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

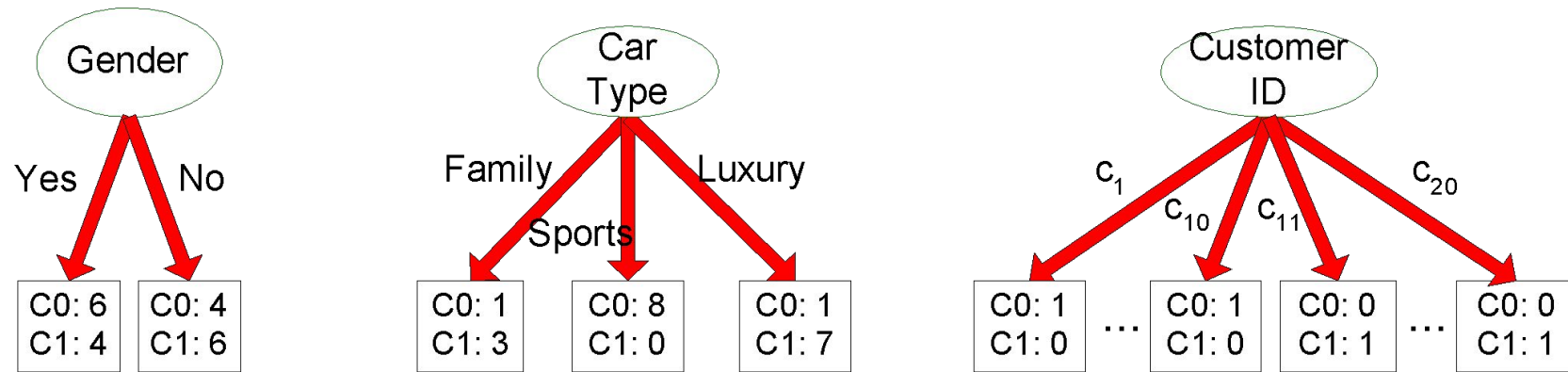
Parent Node, p is split into k partitions;

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 the ID3 and C4.5 decision tree algorithms
- **Disadvantage:** Tends to prefer splits that result in large number of partitions, each being small but pure.

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
- **Can we use such a test condition on new test instances?**

Solution

- A **low impurity value alone is insufficient** to find a good attribute test condition for a node
- **Solution:** Consider the **number of children** produced by the splitting attribute in the identification of the best split
- High number of child nodes implies more complexity

- **Method 1:** Generate only binary decision trees
 - This strategy is employed by decision tree classifiers such as CART
- **Method 2:** Modify the splitting criterion to take into account the number of partitions produced by the attribute

Gain Ratio

- Gain Ratio:

$$\textit{GainRATIO}_{split} = \frac{\textit{GAIN}_{Split}}{\textit{SplitINFO}}$$

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

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	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

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

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

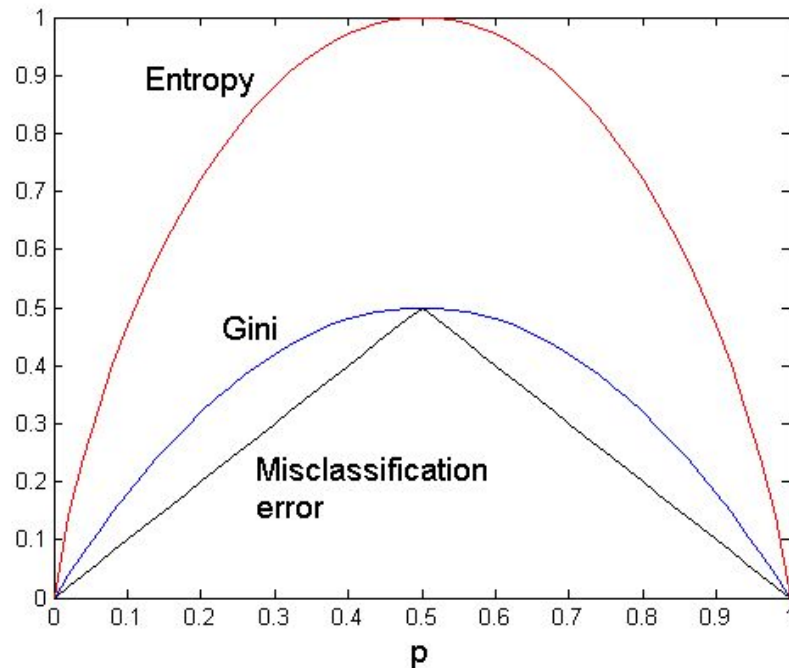
C1	2
C2	4

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

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

Comparison among Impurity Measures

For a 2-class problem:

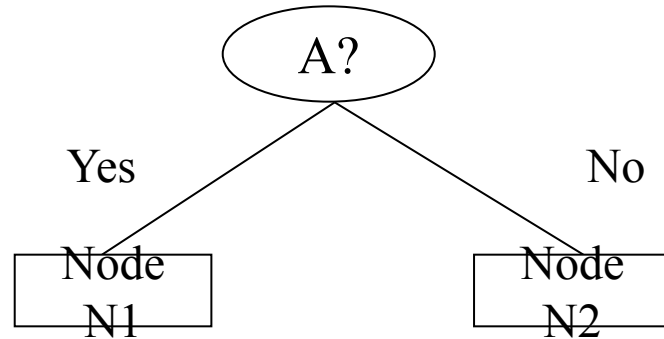


Consistency among the impurity measures

- if a node N1 has lower entropy than node N2, then the Gini index and error rate of N1 will also be lower than that of N2

The attribute chosen as splitting criterion by the impurity measures can still be different!

Misclassification Error vs Gini Index



	Parent
C1	7
C2	3
Gini = 0.42	

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

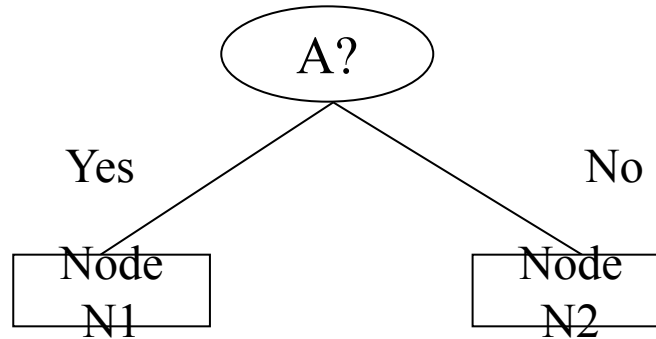
$$\begin{aligned} \text{Gini}(N2) &= 1 - (4/7)^2 - (3/7)^2 \\ &= 0.489 \end{aligned}$$

	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	

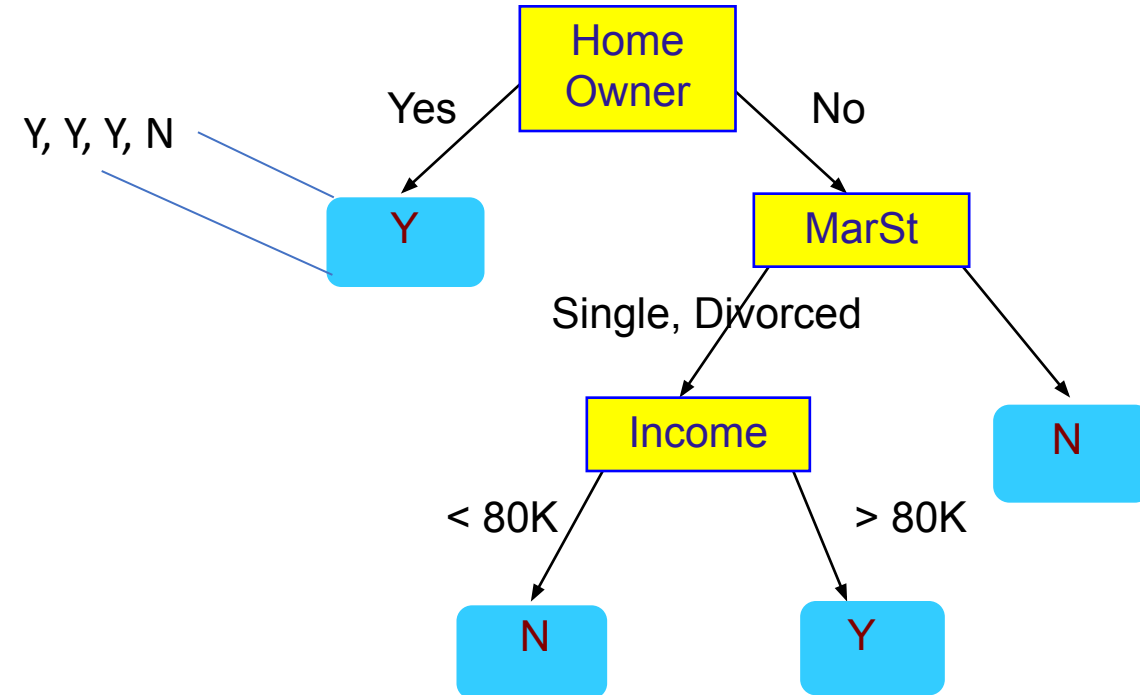
	N1	N2
C1	3	4
C2	0	3
Gini=0.342		

	N1	N2
C1	3	4
C2	1	2
Gini=0.416		

Misclassification error for all three cases = 0.3 !

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 (discussed later)



Algorithms: ID3, C4.5, C5.0, CART

- **ID3** uses the Hunt's algorithm with information gain criterion and gain ratio
- **C4.5** improves **ID3**
 - Needs entire data to fit in memory
 - Handles missing attributes and continuous attributes
 - Performs tree post-pruning
 - **C5.0** is the current commercial successor of **C4.5**
 - Unsuitable for Large Datasets
- **CART** builds multivariate decision (binary) trees

Advantages of Decision Tree

- Easy to interpret for small-sized trees
- Accuracy is comparable to other classification techniques for many simple data sets
- Robust to noise (especially when methods to avoid overfitting are employed)
- **Can easily handle redundant or irrelevant attributes**
- Inexpensive to construct
- Extremely fast at classifying unknown record
- Handle Missing Values

Irrelevant Attributes

- **Irrelevant** attributes are poorly associated with the target class labels, so they have little or no gain in purity
- In case of a **large number** of irrelevant attributes, some of them may be accidentally chosen during the tree-growing process
- Feature selection techniques can help to eliminate the irrelevant attributes during preprocessing

Redundant Attributes

- Decision trees can handle the presence of redundant attributes
- An attribute is **redundant** if it is strongly **correlated** with another attribute in the data
- Since redundant attributes show **similar gains** in purity if they are selected for splitting, **only one** of them will be selected as an attribute test condition in the decision tree algorithm.

Advantages of Decision Tree

- Easy to interpret for small-sized trees
- Accuracy is comparable to other classification techniques for many simple data sets
- Robust to noise (especially when methods to avoid overfitting are employed)
- Can easily handle redundant or irrelevant attributes
- **Inexpensive to construct**
- **Extremely fast at classifying unknown record**
- Handle Missing Values

Computational Complexity

- Finding an optimal decision tree is NP-hard
- Hunt's Algorithm uses a greedy, top-down, recursive partitioning strategy for growing a decision tree
- Such techniques quickly construct a reasonably good decision tree even when the training set size is very large.
- **Construction DT Complexity:** $O(M N \log N)$ where $M=n$. attributes, $N=n$. instances
- Once a decision tree has been built, **classifying** a test record is **extremely fast**, with a worst-case complexity of $O(w)$, where w is the **maximum depth of the tree**.

Handling Missing Attribute Values

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

Computing Impurity Measure

<i>Tid</i>	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	?	Single	90K	Yes

Missing value

Before Splitting:

$$\text{Entropy}(\text{Parent}) = -0.3 \log(0.3) - (0.7) \log(0.7) = 0.8813$$

	Class = Yes	Class = No
Refund=Yes	0	3
Refund=No	2	4
Refund=?	1	0

Split on Refund:

$$\text{Entropy}(\text{Refund=Yes}) = 0$$

$$\text{Entropy}(\text{Refund=No}) = -(2/6) \log(2/6) - (4/6) \log(4/6) = 0.9183$$

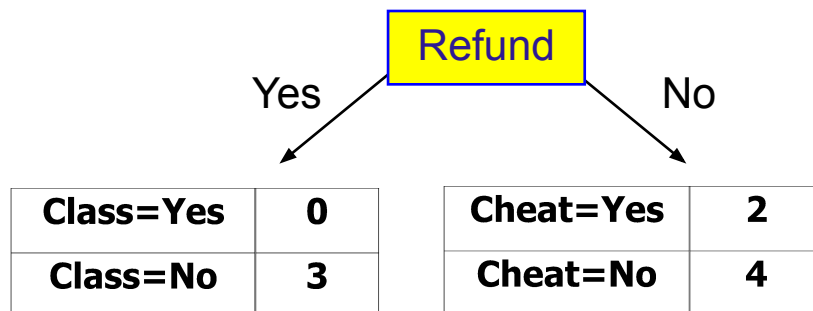
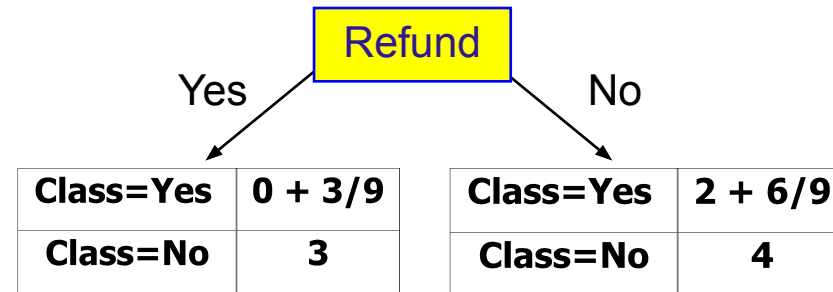
$$\text{Entropy}(\text{Children}) = 0.3(0) + 0.6(0.9183) = 0.551$$

$$\text{Gain} = 0.9 \times (0.8813 - 0.551) = 0.3303$$

Distribute Instances

<i>Tid</i>	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

<i>Tid</i>	Refund	Marital Status	Taxable Income	Class
10	?	Single	90K	Yes



Probability that Refund=Yes is $3/9$

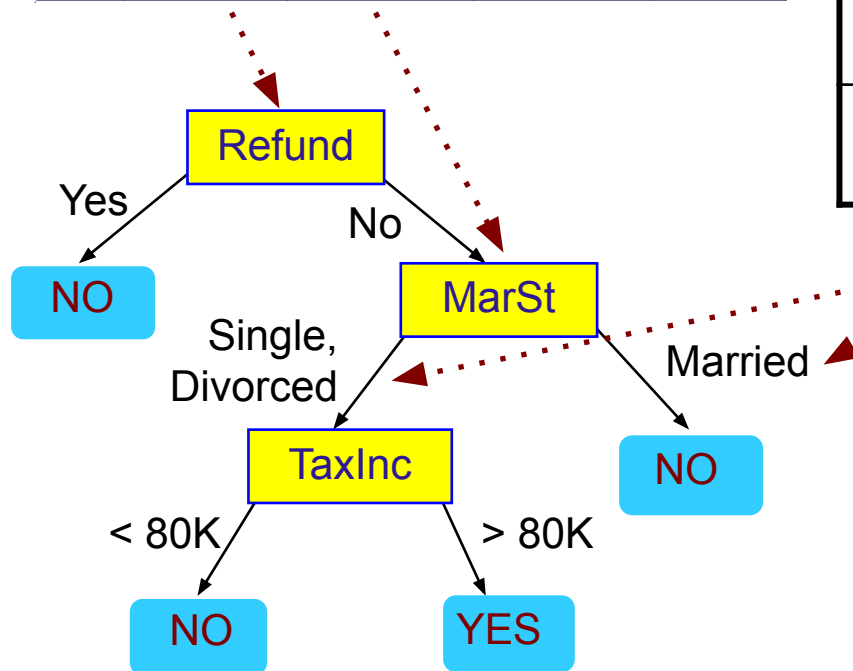
Probability that Refund=No is $6/9$

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

Classify Instances

New record:

<i>Tid</i>	Refund	Marital Status	Taxable Income	Class
11	No	?	85K	?



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

Probabilistic split method (C4.5)

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

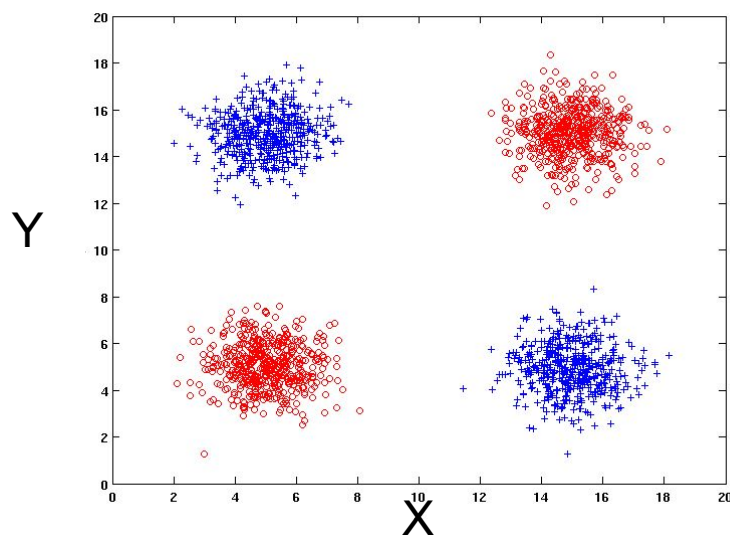
Probability that Marital Status = {Single, Divorced} is $3/6.67$

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

Interacting attributes: able to distinguish between classes when used together, but individually they provide little or no information.



+ : 1000 instances

o : 1000 instances

Test Condition:
 $X \leq 10$ and $Y \leq 10$

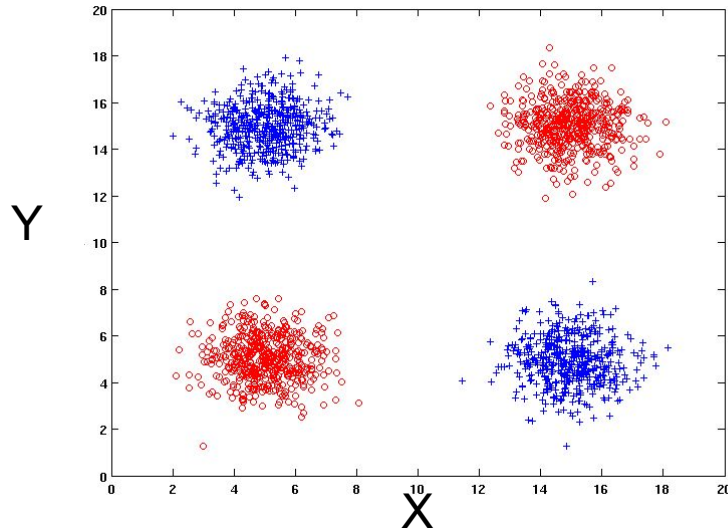
Entropy (X) : 0.99

Entropy (Y) : 0.99



No reduction in the
impurity measure when
used individually

Handling interactions



+ : 1000 instances

Entropy (X) : 0.99

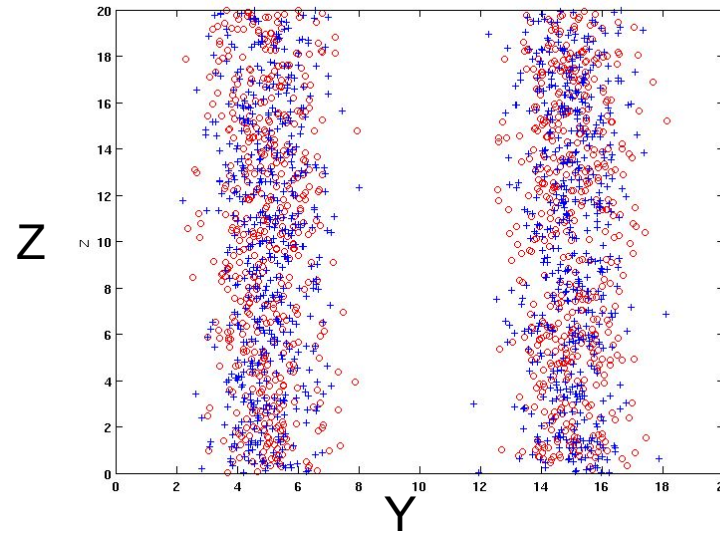
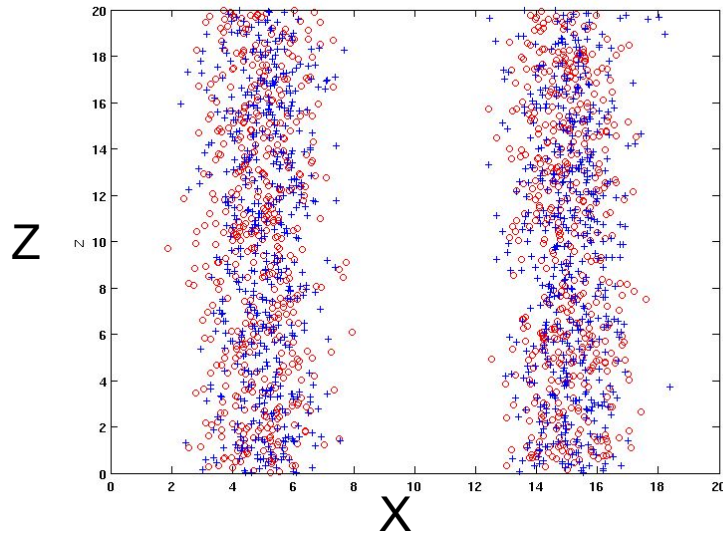
o : 1000 instances

Entropy (Y) : 0.99

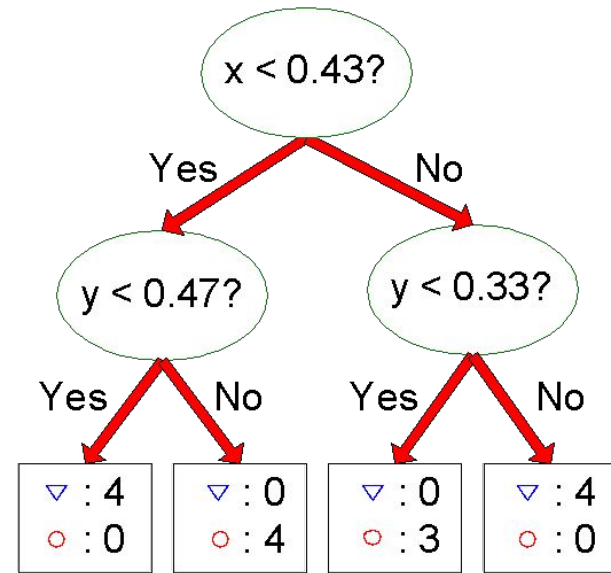
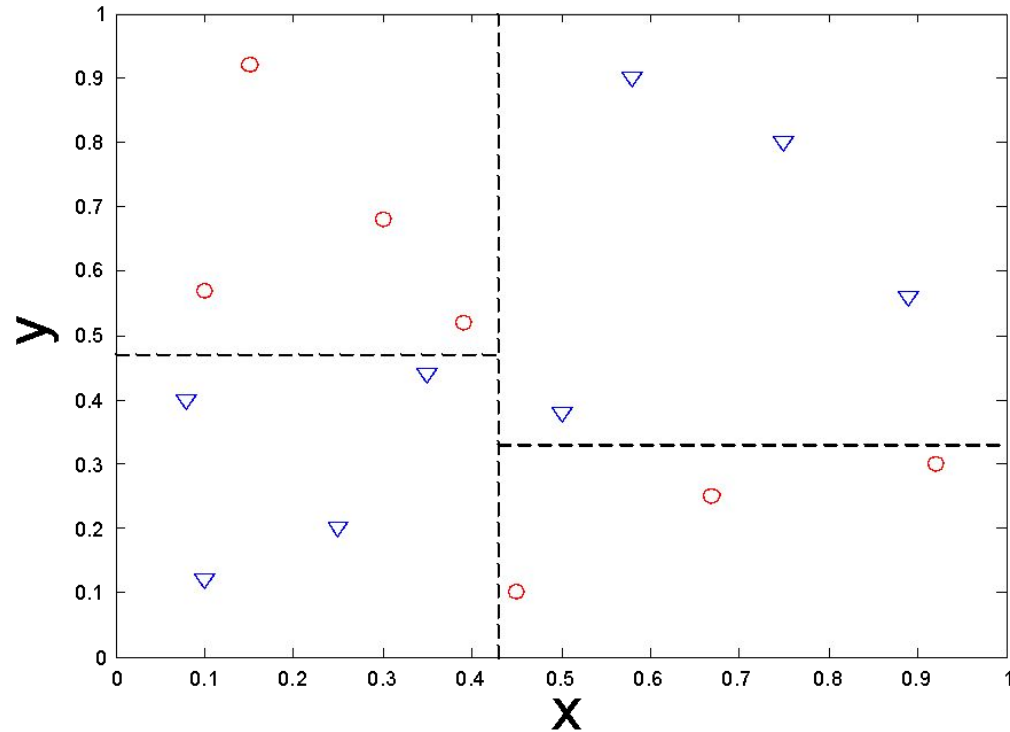
Entropy (Z) : 0.98

Adding Z as a noisy
attribute generated
from a uniform
distribution

Attribute Z will be
chosen for splitting!

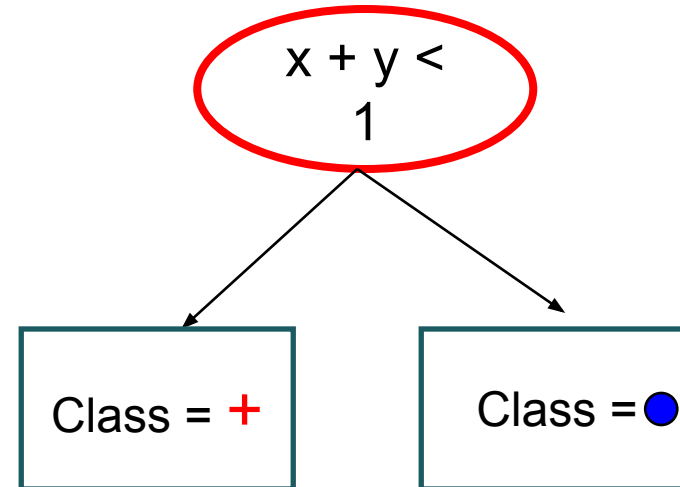
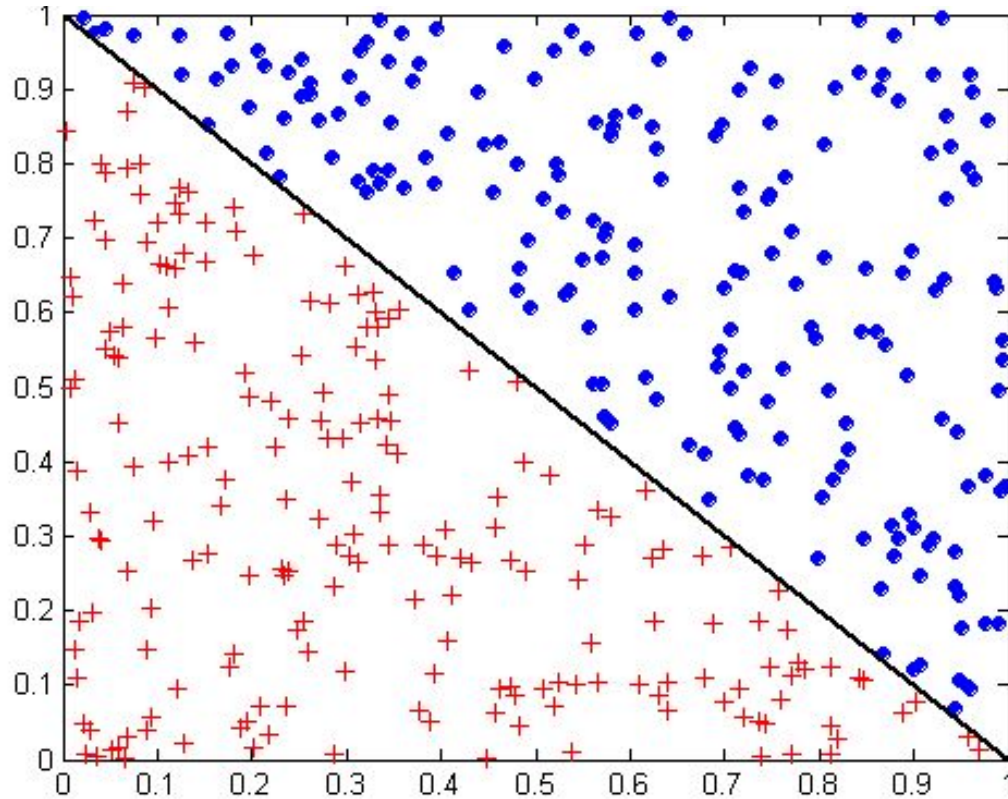


Decision Boundary



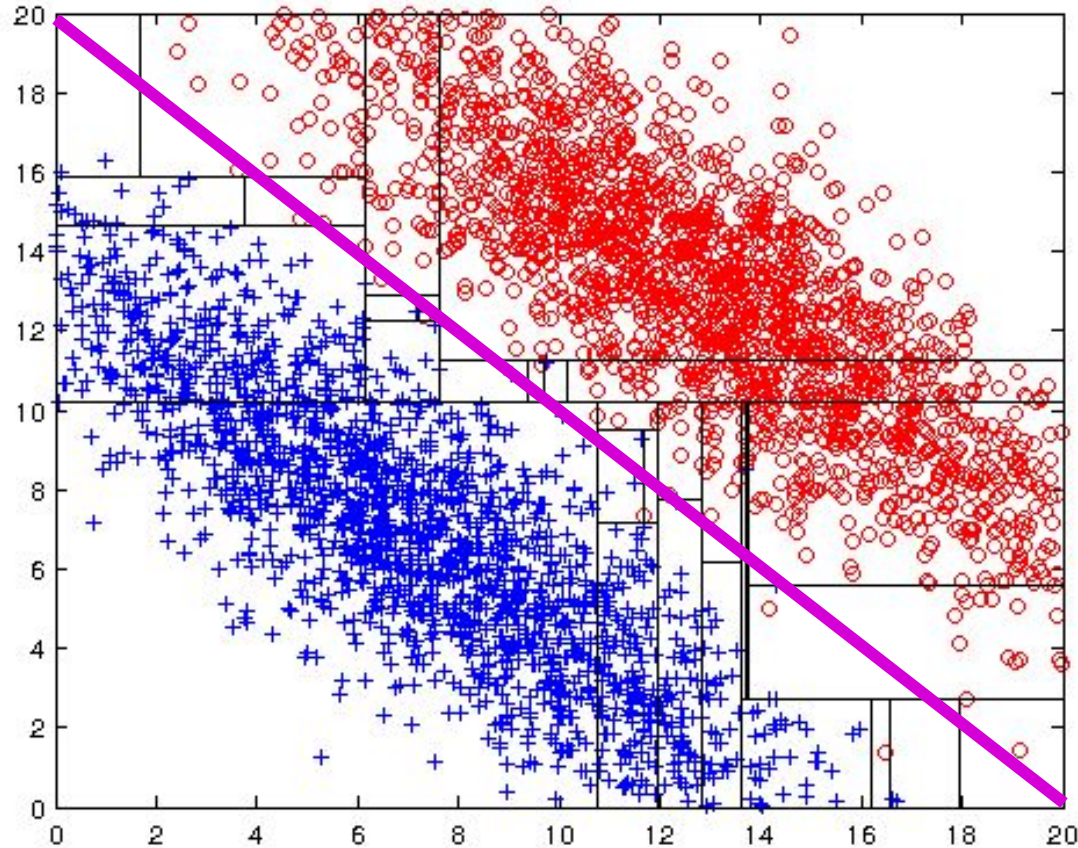
- Border line between two neighboring regions of different classes is known as decision boundary
- Decision boundary is parallel to axes because test condition involves a single attribute at-a-time

Oblique Decision Trees



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

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.

Test Condition
 $x + y < 20$

Other Issues

- Data Fragmentation
- Tree Replication

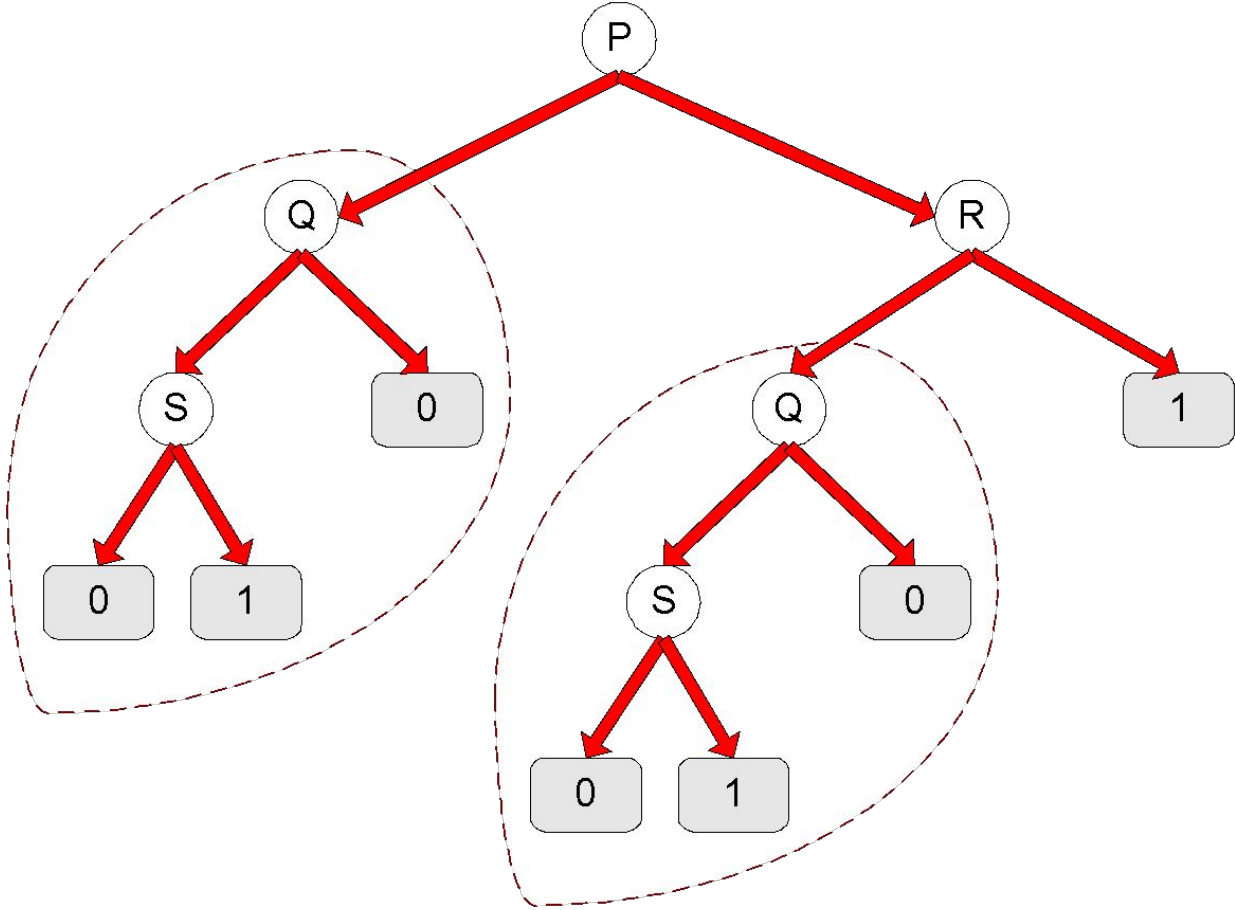
Data Fragmentation

- 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

Expressiveness

- Decision tree provides expressive representation for learning discrete-valued function
 - Every discrete-valued function can be represented as an assignment table, where every unique combination of discrete attributes is assigned a class label.
 - But they do not generalize well to certain types of Boolean functions
 - Example: parity function:
 - Class = 1 if there is an even number of Boolean attributes with truth value = True
 - Class = 0 if there is an odd number of Boolean attributes with truth value = True
 - For accurate modeling, must have a complete tree
- Not expressive enough for modeling continuous variables
 - Particularly when test condition involves only a single attribute at-a-time

Tree Replication



Same subtree appears in multiple branches

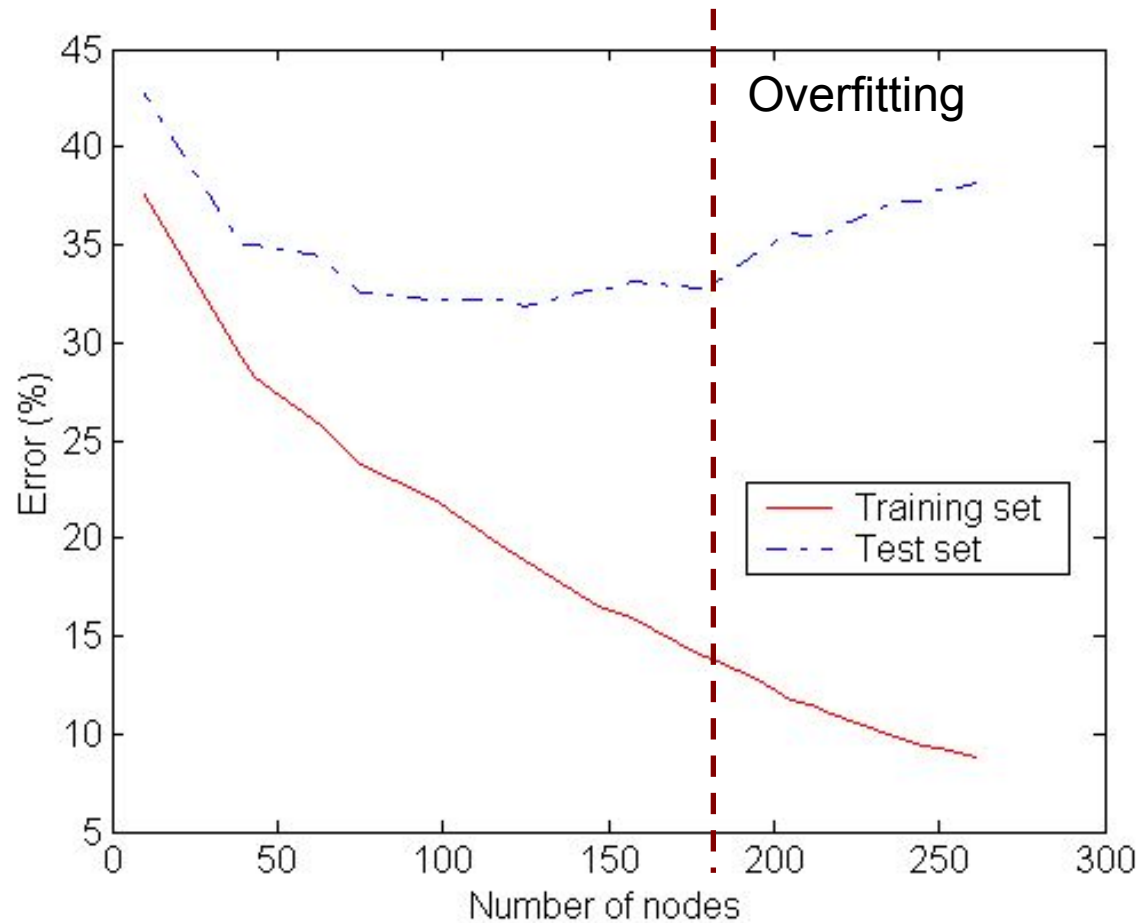
Practical Issues of Classification

- Underfitting and Overfitting
- Costs of Classification

Classification Errors

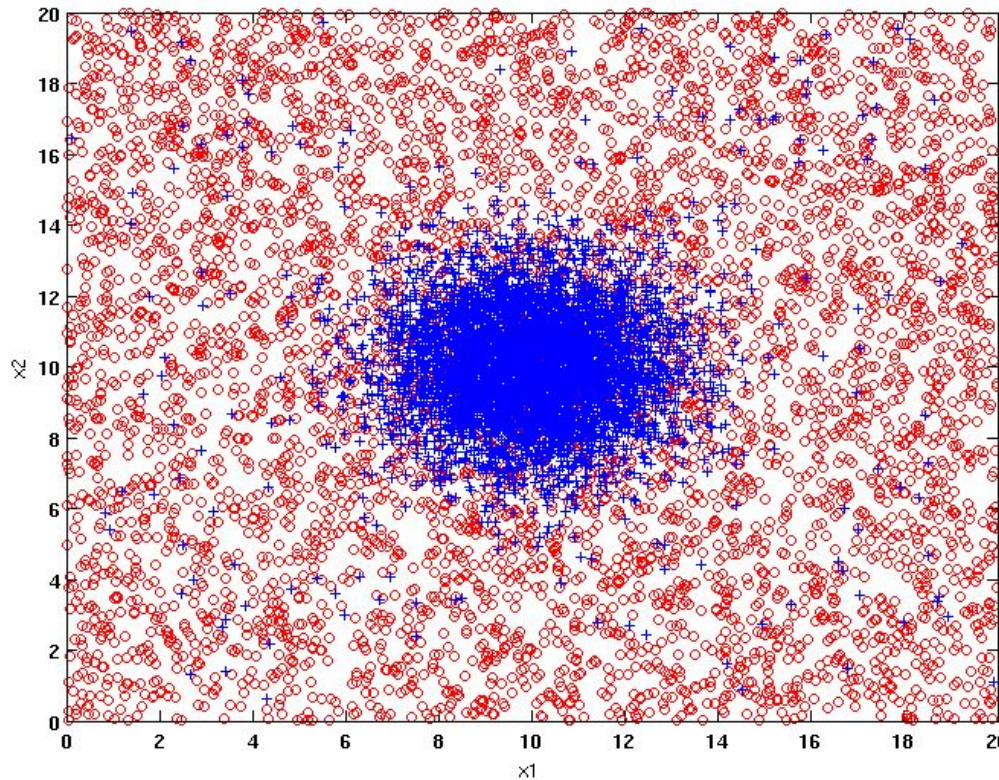
- Training errors (apparent errors)
 - Errors committed on the training set
- Test errors
 - Errors committed on the test set
- Generalization errors
 - Expected error of a model over random selection of records from same distribution

Underfitting and Overfitting



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

Example Data Set



Two class problem:

+ : 5200 instances

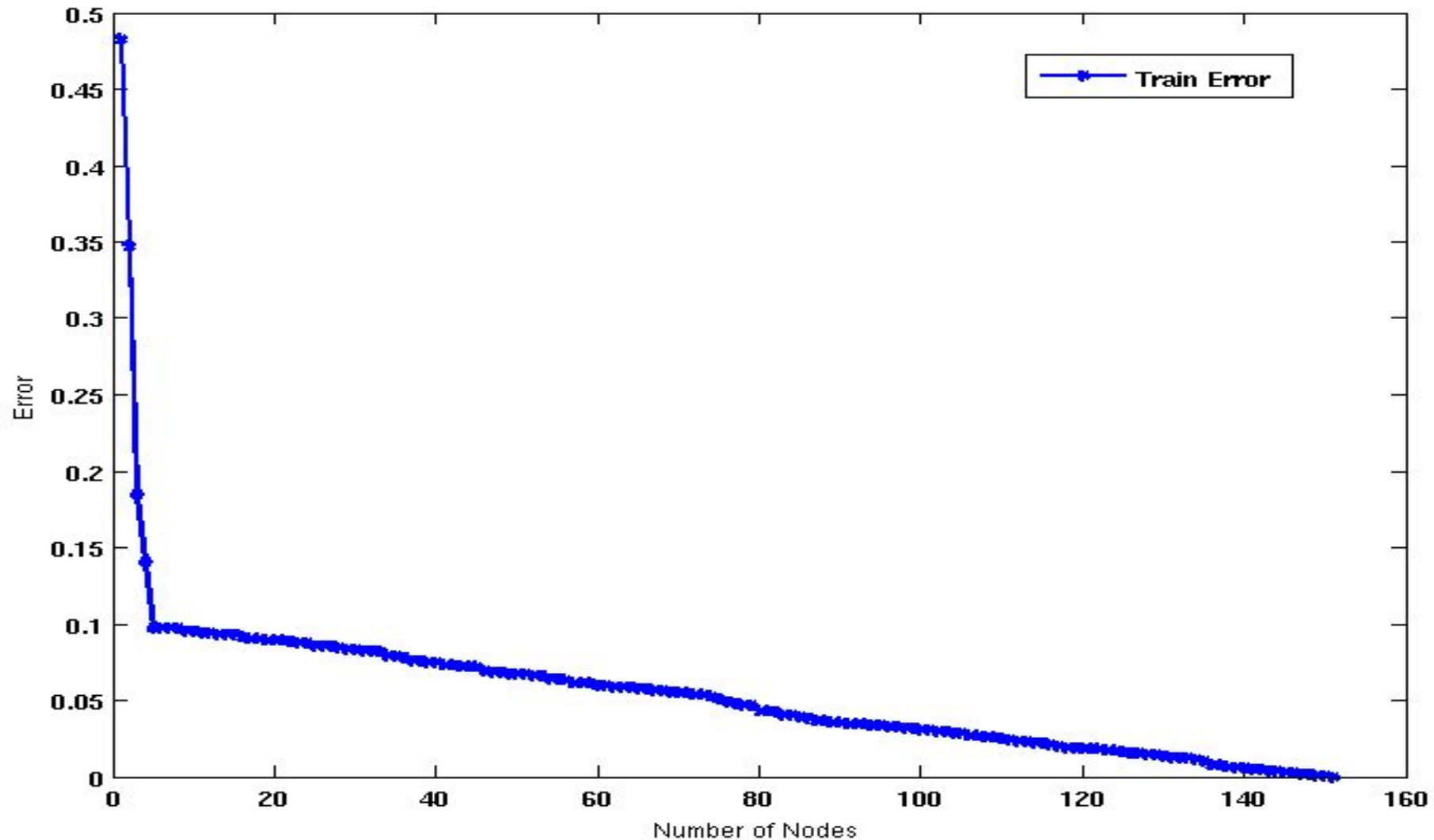
- 5000 instances generated from a Gaussian centered at (10,10)
- 200 noisy instances added

o : 5200 instances

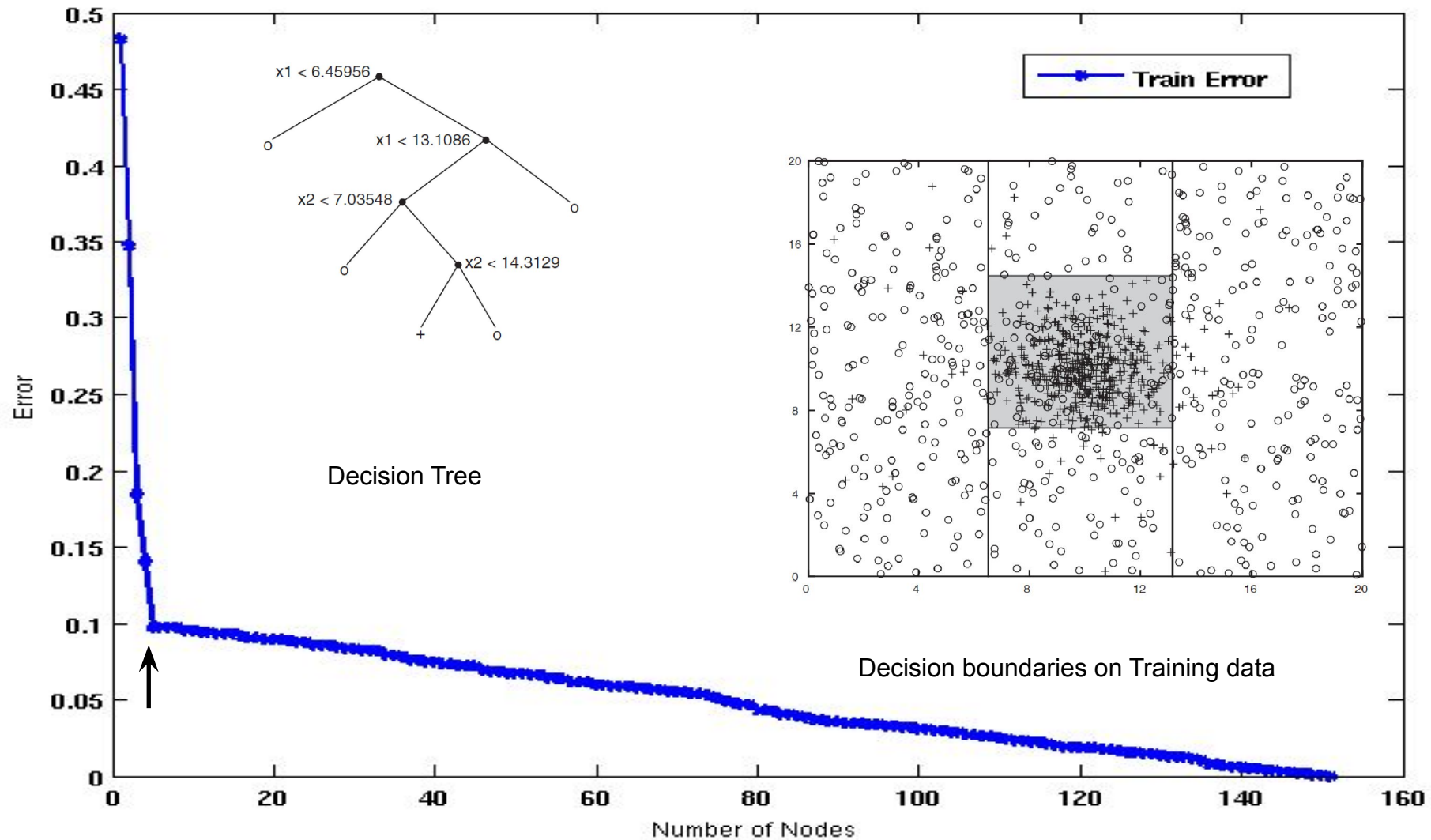
- Generated from a uniform distribution

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

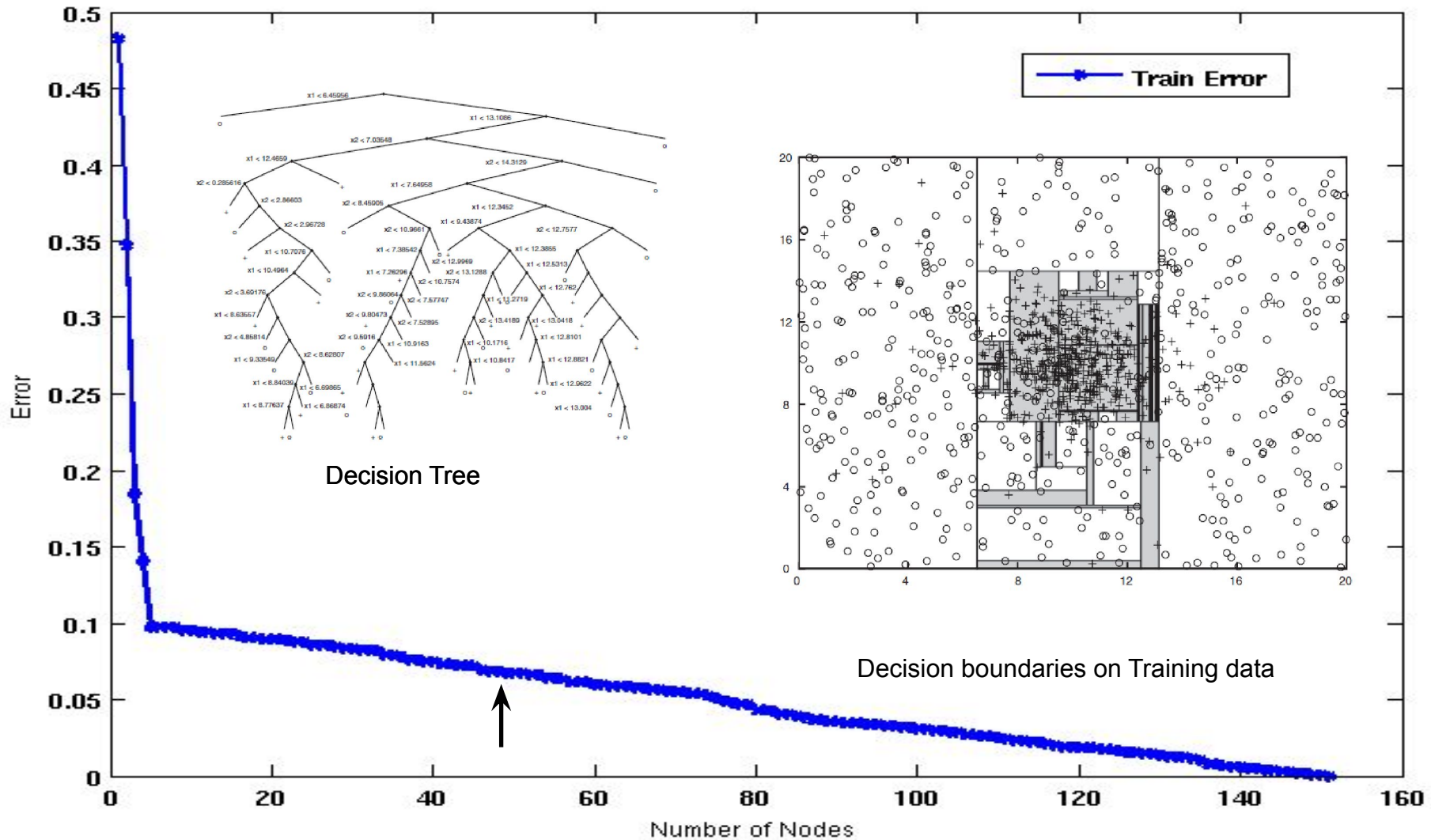
Increasing number of nodes in Decision Trees



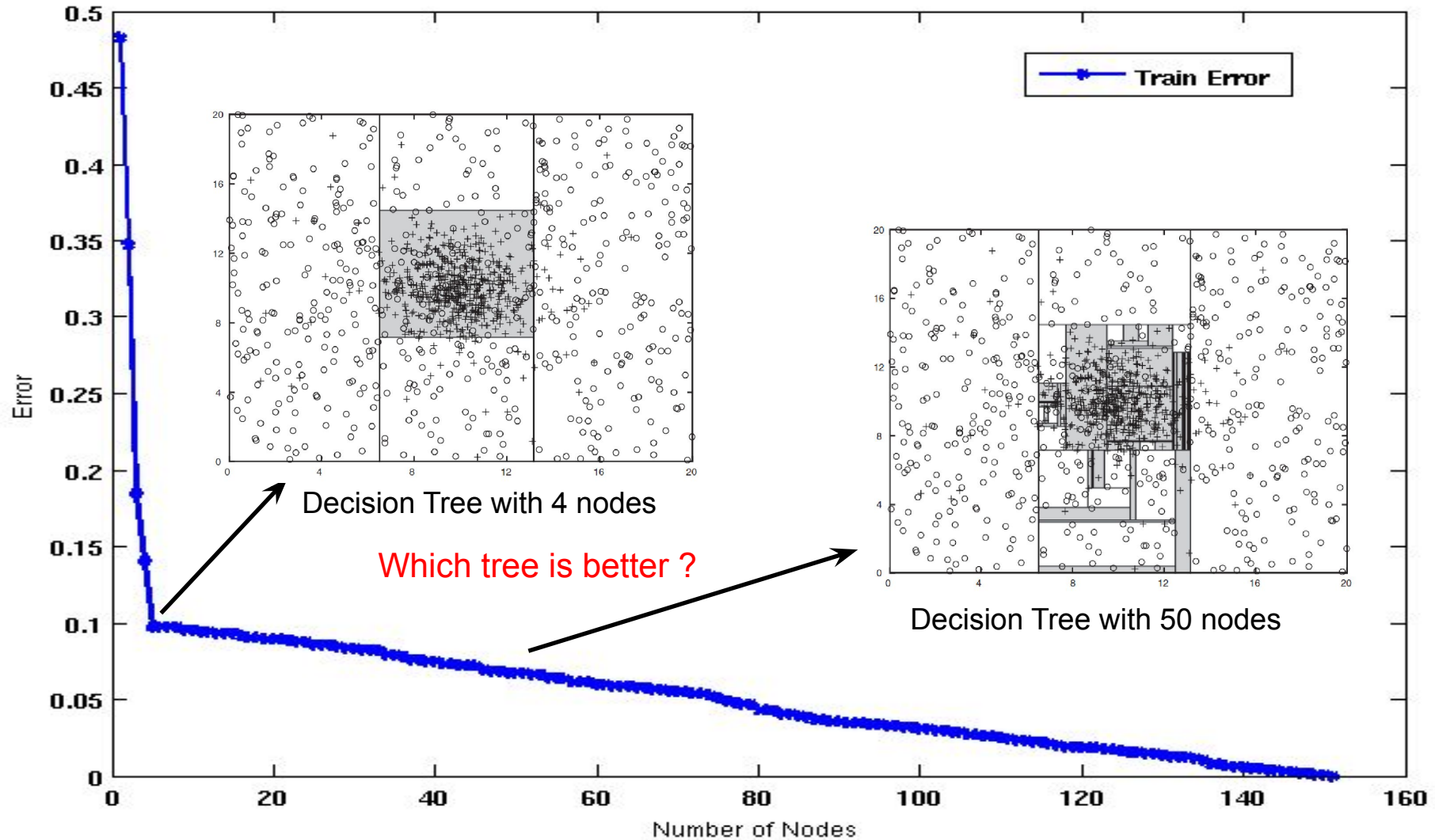
Decision Tree with 4 nodes



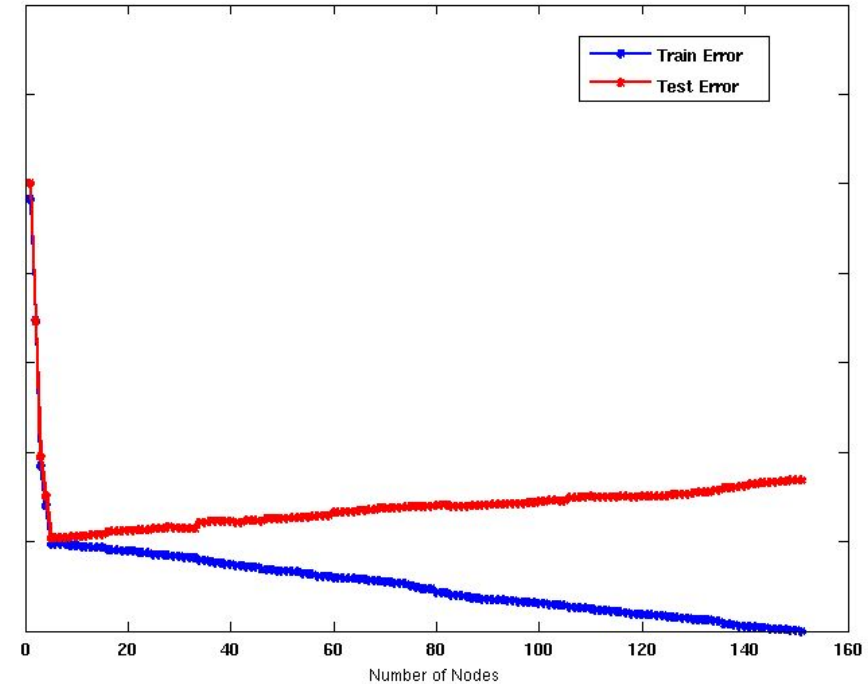
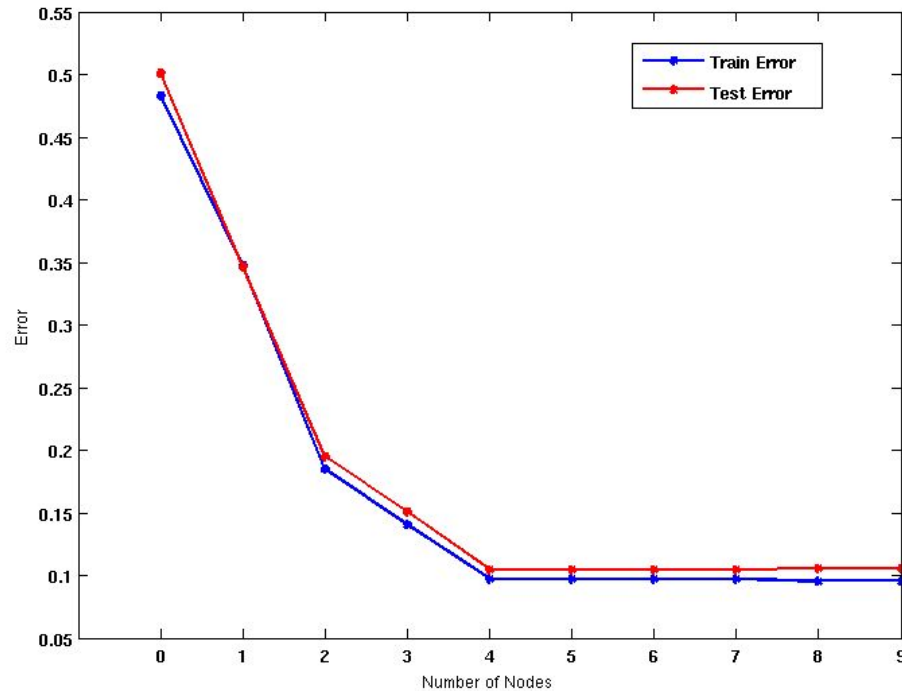
Decision Tree with 50 nodes



Which tree is better?



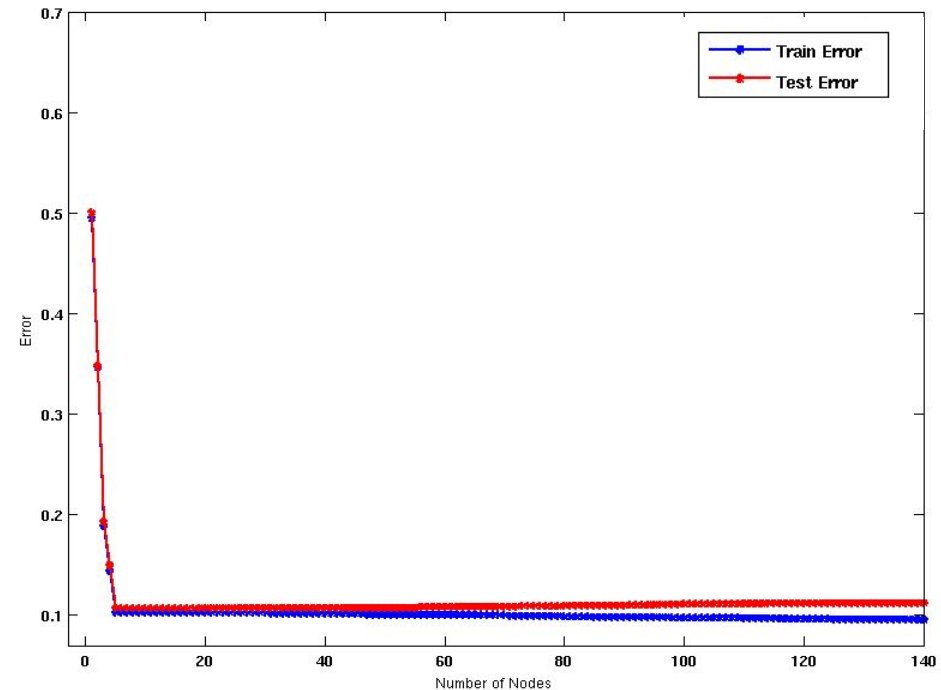
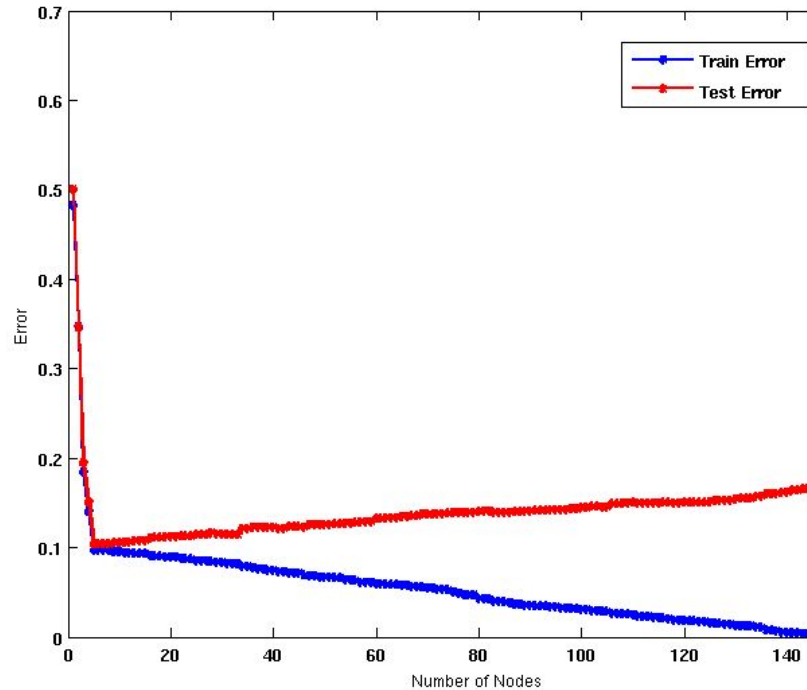
Model Overfitting



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

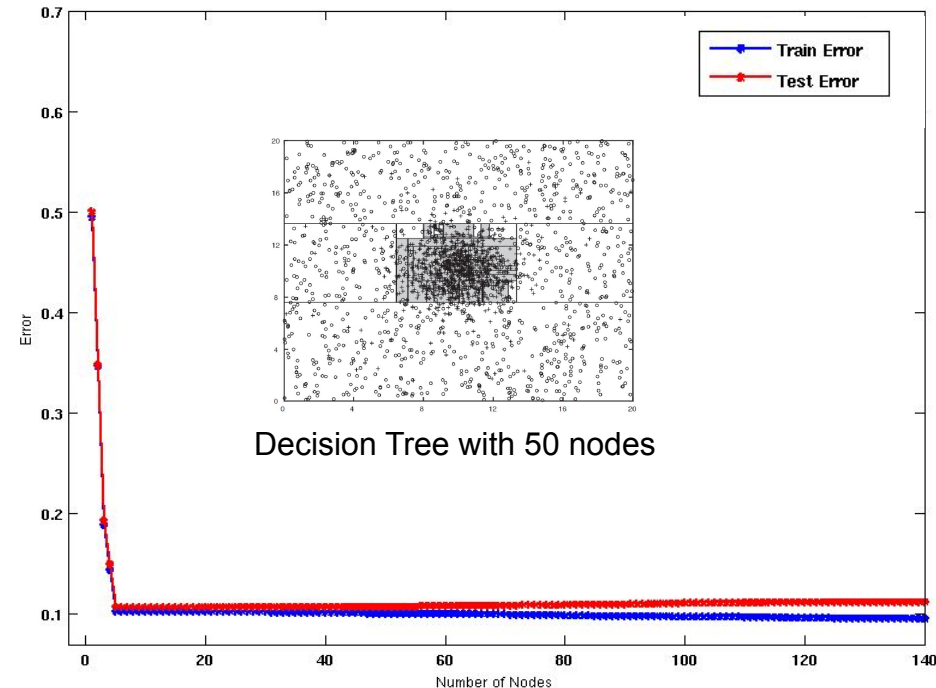
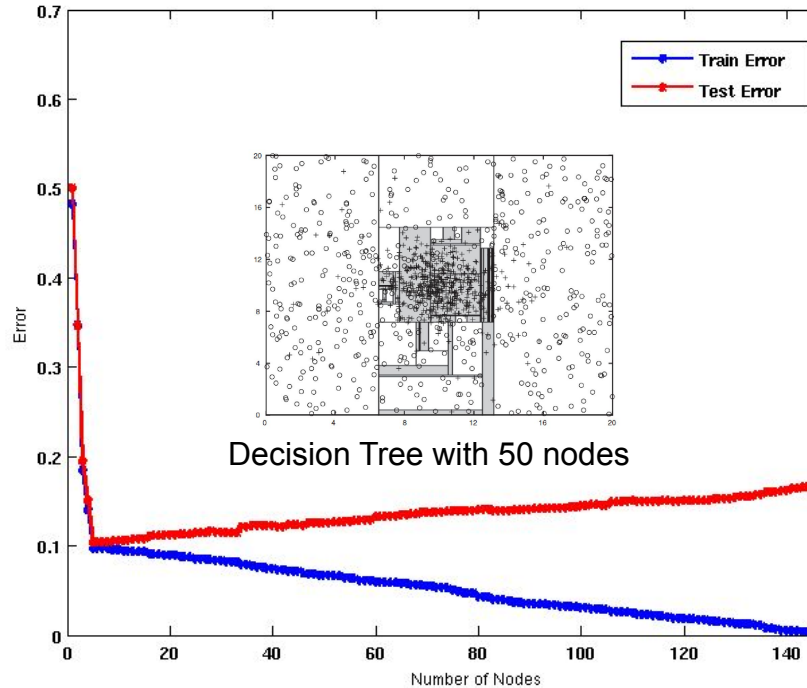
Model Overfitting



Using twice the number of data instances

- If training data is under-representative, testing errors increase and training errors decrease on increasing number of nodes
- Increasing the size of training data reduces the difference between training and testing errors at a given number of nodes

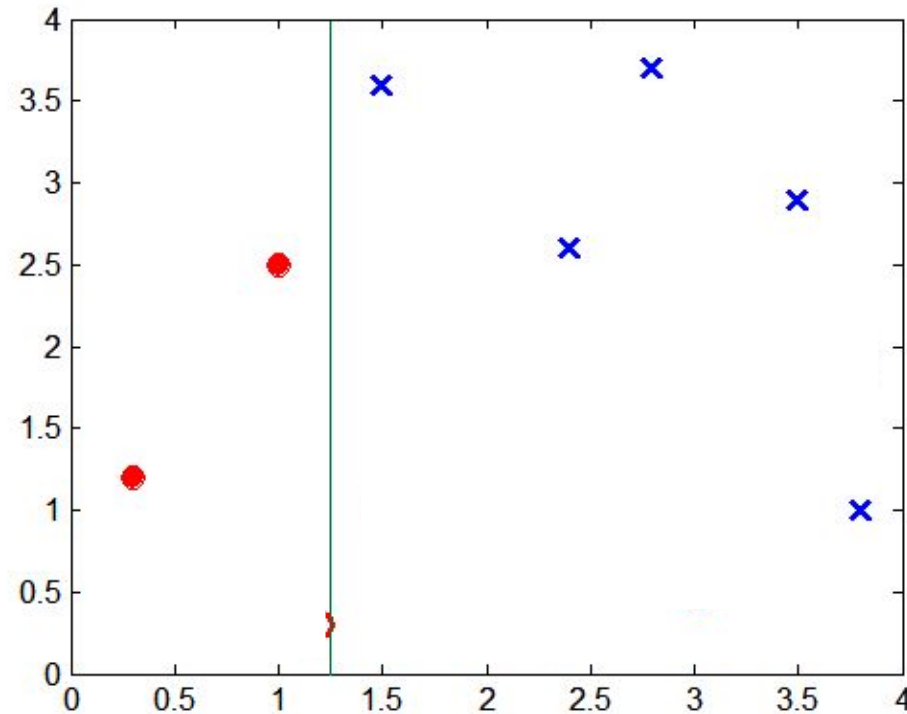
Model Overfitting



Using twice the number of data instances

- If training data is under-representative, testing errors increase and training errors decrease on increasing number of nodes
- Increasing the size of training data reduces the difference between training and testing errors at a given number of nodes

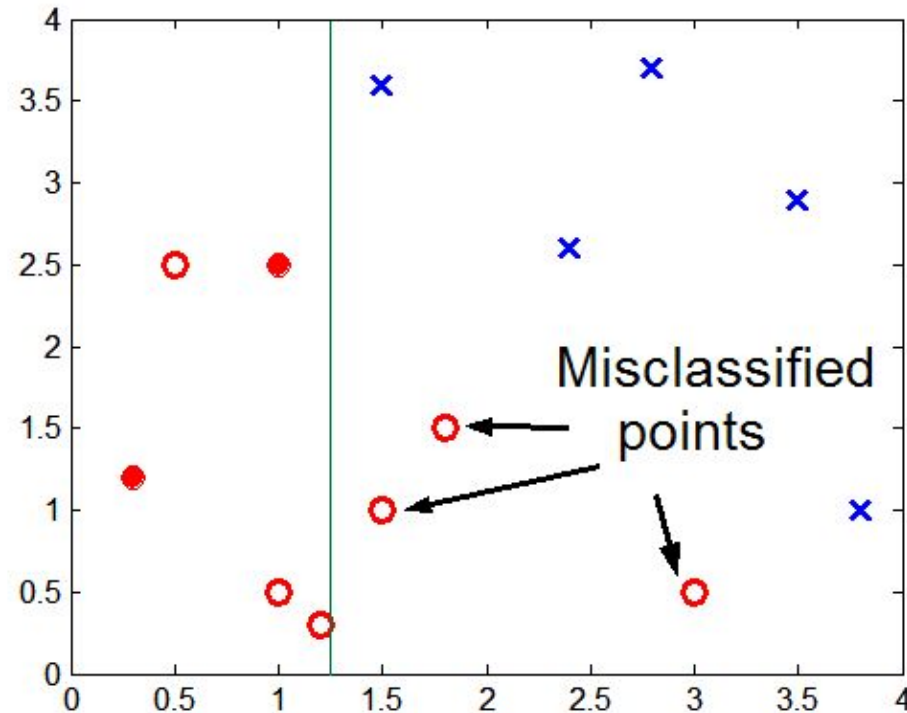
Overfitting due to Insufficient Examples



Lack of data points in the lower half of the diagram makes it difficult to predict correctly the class labels of that region

- Insufficient number of training records in the region causes the decision tree to predict the test examples using other training records that are irrelevant to the classification task

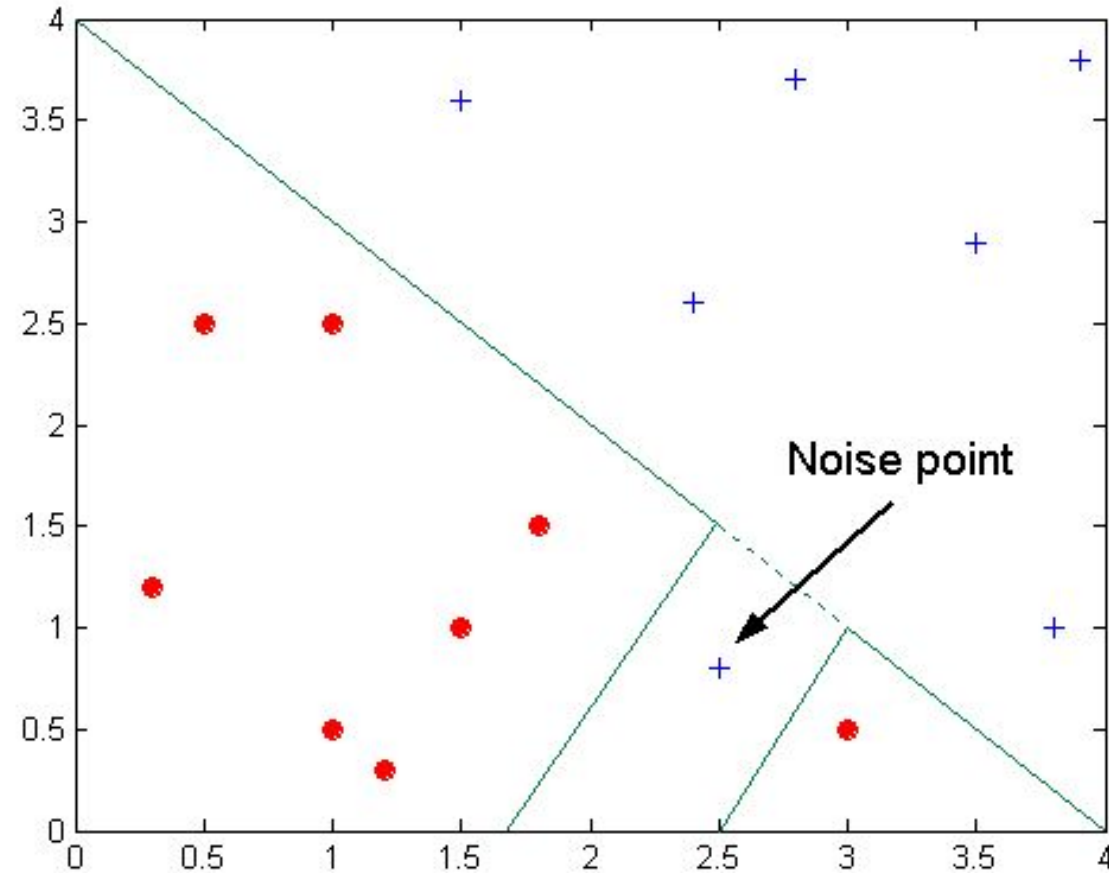
Overfitting due to Insufficient Examples



Lack of data points in the lower half of the diagram makes it difficult to predict correctly the class labels of that region

- Insufficient number of training records in the region causes the decision tree to predict the test examples using other training records that are irrelevant to the classification task

Overfitting due to Noise



Decision boundary is distorted by noise point

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

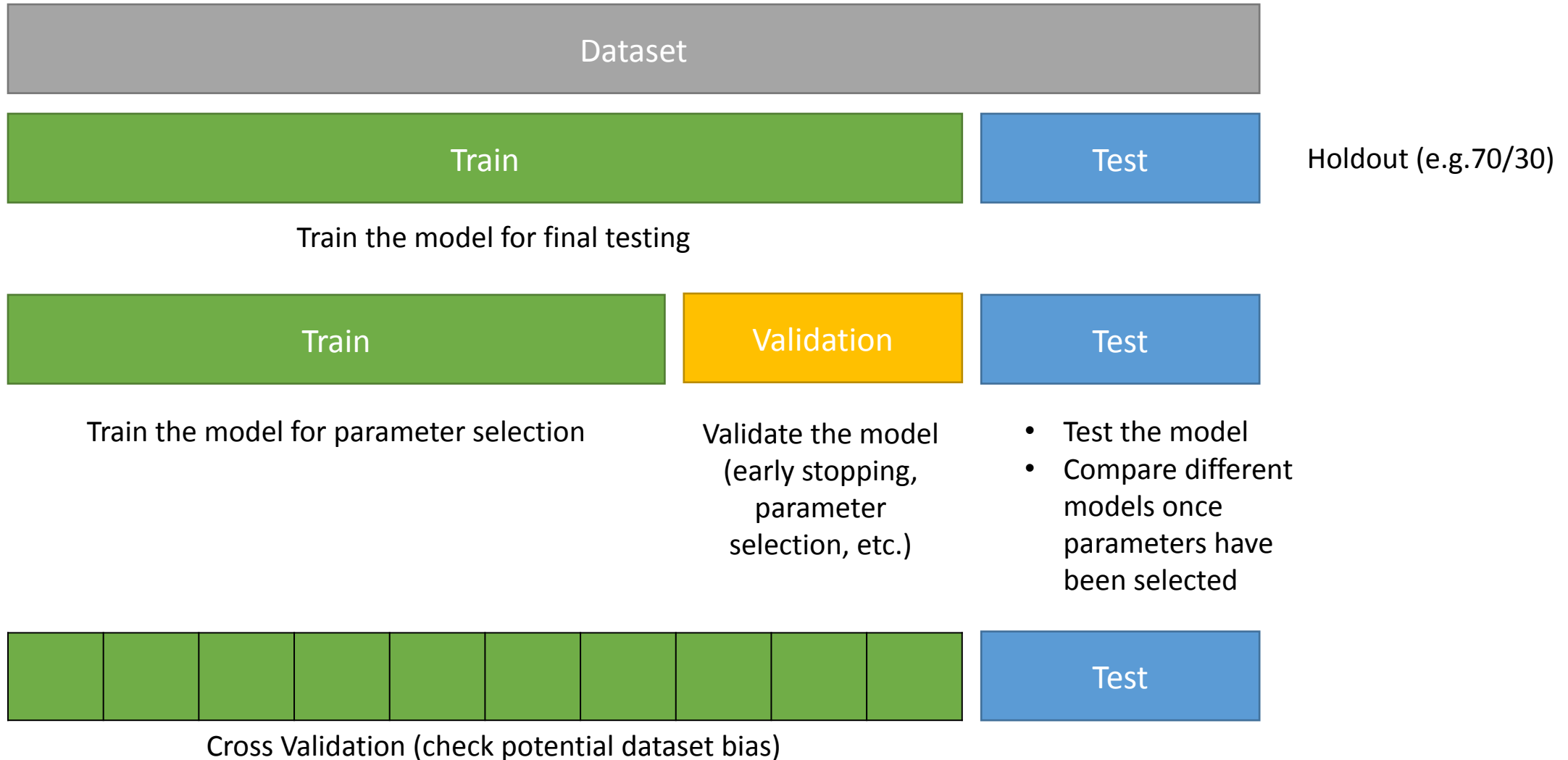
Model Selection

- 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
 - Estimating Statistical Bounds

Model Selection 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

Data Partitioning



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

Model Selection Incorporating Model Complexity

- 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

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

Estimating Generalization Errors

- **Re-substitution errors:** error on training ($\sum \text{err}(t)$)
- **Generalization errors:** error on testing ($\sum \text{err}'(t)$)
- Methods for estimating generalization errors:
 - **Pessimistic approach**
 - **Optimistic approach**
 - **Reduced error pruning (REP):**
 - uses validation data set to estimate generalization error

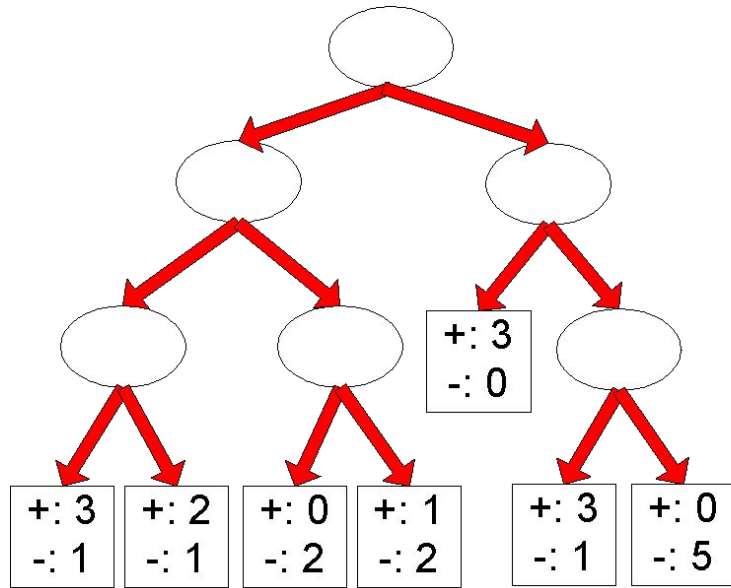
Estimating the Complexity of Decision Trees

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

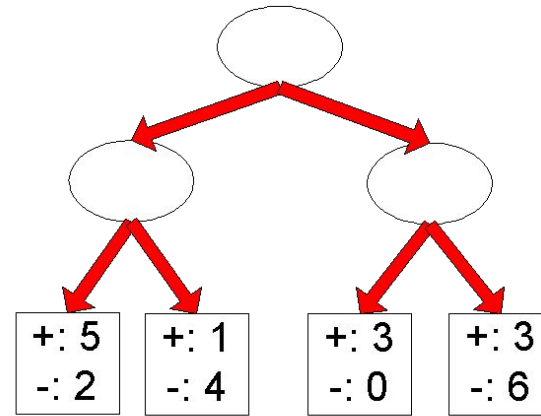
$$err_{gen}(T) = err(T) + \Omega \times \frac{k}{N_{train}},$$

- $err(T)$: error rate on all training records
- Ω : 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

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

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

$$\Omega = 1$$

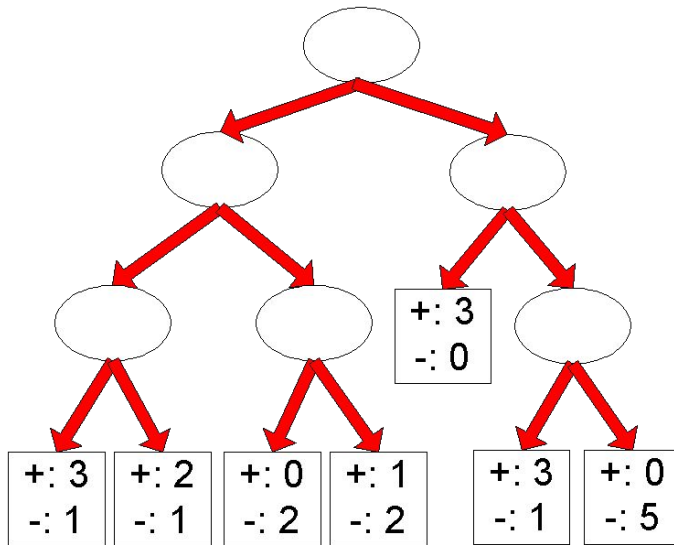
$$e_{\text{gen}}(T_L) = 4/24 + 1 \cdot 7/24 = 11/24 = 0.458$$

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

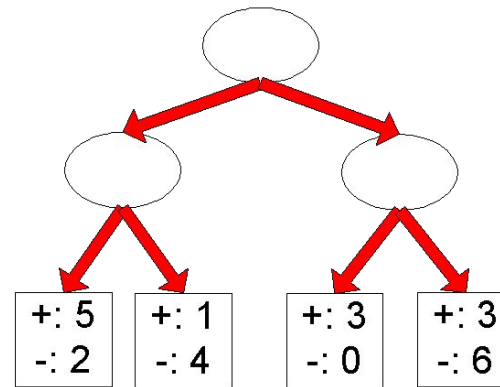
$$err_{\text{gen}}(T) = err(T) + \Omega \times \frac{k}{N_{\text{train}}},$$

Estimating the Complexity of Decision Trees

- Re-substitution Estimate:
 - Using training error as an **optimistic** estimate of generalization error
 - Referred to as optimistic error estimate



Decision Tree, T_L



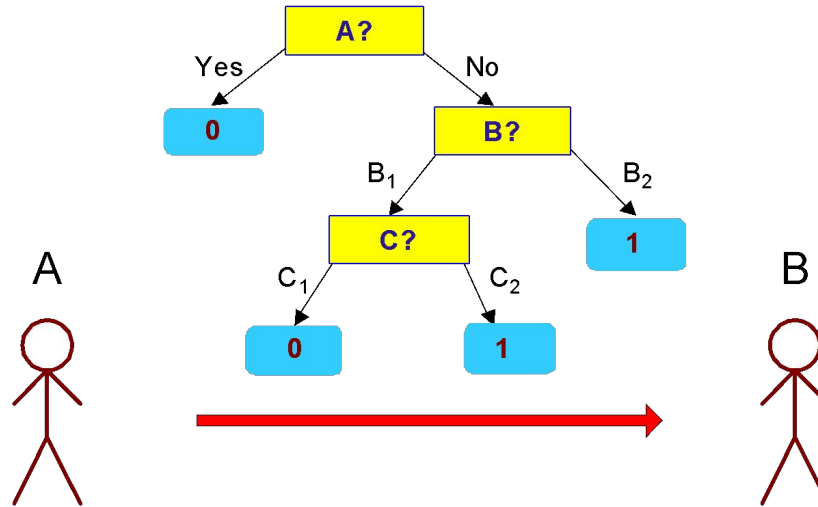
Decision Tree, T_R

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

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

Minimum Description Length (MDL)

X	y
X ₁	1
X ₂	0
X ₃	0
X ₄	1
...	...
X _n	1



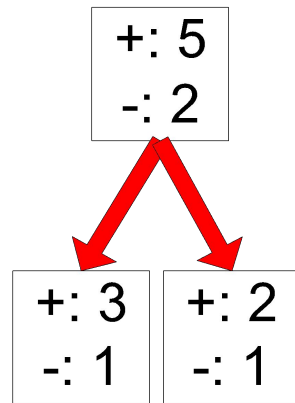
X	y
X ₁	?
X ₂	?
X ₃	?
X ₄	?
...	...
X _n	?

- $\text{Cost}(\text{Model}, \text{Data}) = \text{Cost}(\text{Data} | \text{Model}) + \text{Cost}(\text{Model})$
 - 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.

Estimating Statistical Bounds

Apply a statistical correction to the training error rate of the model that is indicative of its model complexity.

- Need probability distribution of training error: available or assumed.
- The number of errors committed by a leaf node in a decision tree can be assumed to follow a binomial distribution.



Before splitting: $e = 2/7$, $e'(7, 2/7, 0.25) = 0.503$
 $e'(T) = 7 \times 0.503 = 3.521$

After splitting:

$e(T_L) = 1/4$, $e'(4, 1/4, 0.25) = 0.537$
 $e(T_R) = 1/3$, $e'(3, 1/3, 0.25) = 0.650$
 $e'(T) = 4 \times 0.537 + 3 \times 0.650 = 4.098$

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

Therefore, do not split

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 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

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

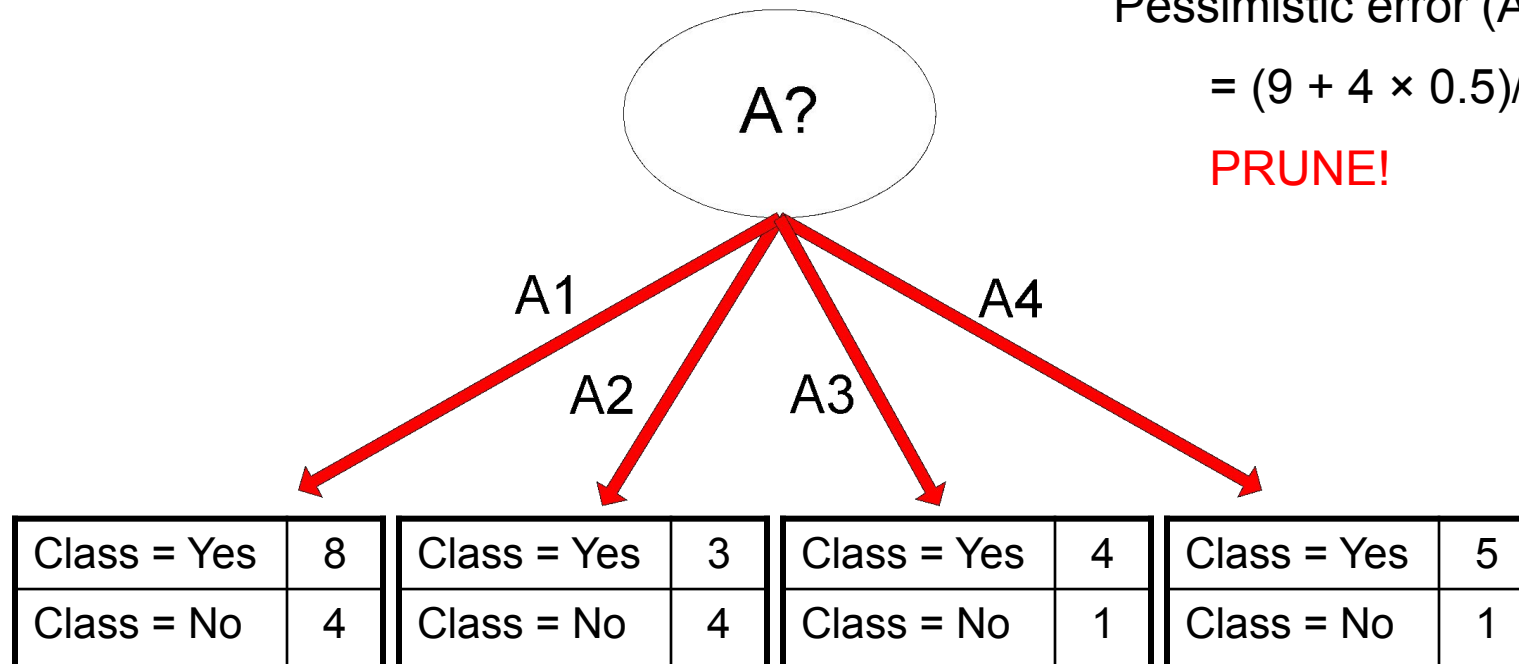
Pessimistic error = $(10 + 0.5)/30 = 10.5/30$

Training Error (After splitting) = 9/30

Pessimistic error (After splitting)

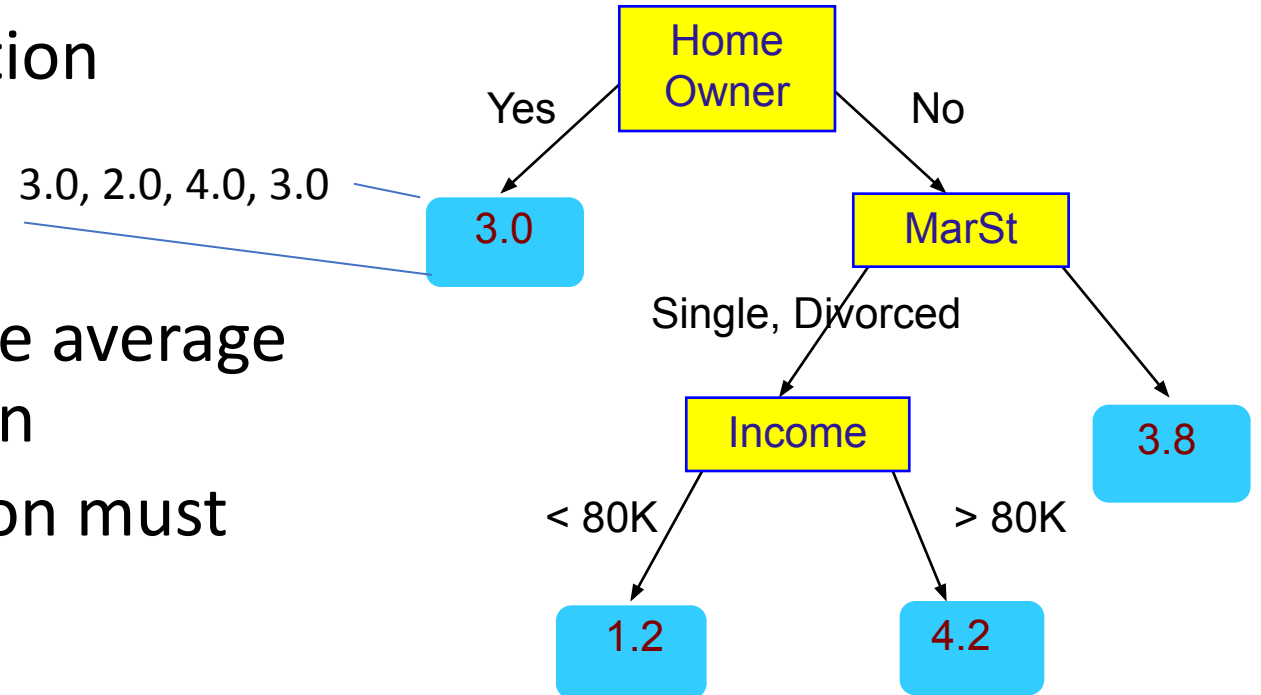
= $(9 + 4 \times 0.5)/30 = 11/30$

PRUNE!



Decision Trees for Regression

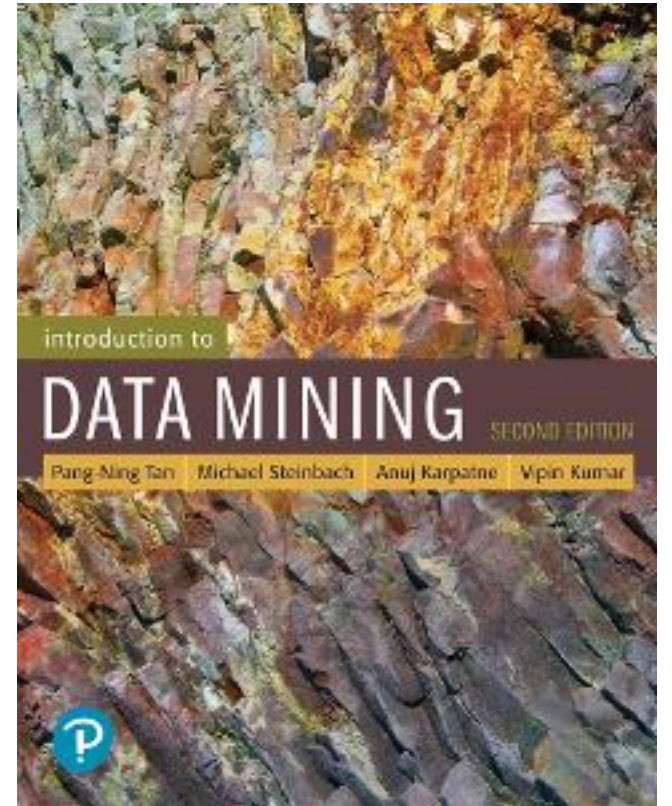
- The same induction and application procedures can be used.
- The only differences are:
 - When leaves are not pure, the average value is returned as prediction
 - Different optimization criterion must be used such as
 - MSE
 - MAE



$$\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2 \quad \text{MAE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} |y_i - \hat{y}_i|$$

References

- Classification: Basic Concepts and Techniques . Chapter 3. Introduction to Data Mining.



	Outlook	Temperature	Wind	Play Tennis
1	Sunny	25	Weak	No
2	Sunny	26	Strong	No
3	Overcast	27	Weak	Yes
4	Rain	19	Weak	Yes
5	Rain	8	Weak	Yes
6	Rain	7	Strong	No
7	Overcast	10	Strong	Yes
8	Sunny	15	Weak	No
9	Sunny	8	Weak	Yes
10	Rain	15	Weak	Yes

	Outlook	Temperature	Wind	Play Tennis
11	Sunny	14	Strong	Yes
12	Overcast	13	Strong	Yes
13	Overcast	26	Weak	Yes
14	Rain	16	Strong	No

Split Wind		
	Weak	Strong
Yes	5	1
No	2	2
	7	3
ME	0,28571429	0,33333333
ME	0,3	
Gain	0,1	

Split Outlook				
	Sunny	Outlook	Rain	
Yes	1	2	3	
No	3	0	1	
	4	2	4	
ME	0,25	0	0,25	
ME	0,2			
Gain	0,2			
	Sunny	Not Sunny		
Yes	1	5		
No	3	1		
	4	6		
ME	0,25	0,16666667		
ME	0,2			
Gain	0,2			

	Outlook	Not Outlook	
Yes	2	4	
No	0	4	
	2	8	
ME	0	0,5	
ME	0,4		
Gain	0		
	Rain	Not Rain	
Yes	3	5	
No	1	1	
	4	6	
ME	0,25	0,16666667	
ME	0,2		
Gain	0,2		

P

Yes	6		0,4
No	4		

Split Temperature															
	<= 7		<= 8		<= 10		<= 15		<= 19		<= 25		<= 26		
Yes	0	6	2	4	3	3	4	2	5	1	5	1	5	1	
No	1	3	1	3	1	3	2	2	2	2	3	1	4	0	
	1	9	3	7	4	6	6	4	7	3	8	2	9	1	
ME	0	0,3333	0,3333	0,4286	0,25	0,5	0,3333	0,5	0,2857	0,3333	0,375	0,5	0,4444	0	
ME	0,3		0,4		0,4		0,4		0,3		0,4		0,4		
Gain	0,1		0		0		0		0,1		0		0		

	Outlook	Temperature	Wind	Play Tennis
1	Sunny	25	Weak	No
2	Sunny	26	Strong	No
8	Sunny	15	Weak	No
9	Sunny	8	Weak	Yes

P	
Yes	1
No	3
ME	0,25

Split Wind		
	Weak	Strong
Yes	1	0
No	2	1
	3	1
ME	0,3333	0
ME	0,25	
Gain	0	

Split Temperature						
	<=8	>8	<=15	>15	<=25	>25
Yes	1	0	1	0	1	0
No	0	3	1	2	2	1
	1	3	2	2	3	1
ME	0	0	0,5	0	0,3333	0
ME	0		0,25		0,25	
Gain	0,25		0		0	

	Outlook	Temperature	Wind	Play Tennis
3	Overcast	27	Weak	Yes
7	Overcast	10	Strong	Yes
algorithm ends				

	Outlook	Temperature	Wind	Play Tennis
4	Rain	19	Weak	Yes
5	Rain	8	Weak	Yes
6	Rain	7	Strong	No
10	Rain	15	Weak	Yes

P	
Yes	3
No	1
ME	0,25

Split Wind		
	Weak	Strong
Yes	3	0
No	0	1
	3	1
ME	0	0
ME	0	
Gain	0,25	

Split Temperature						
	<=7	>7	<=8	>8	<=15	>15
Yes	0	3	1	2	2	1
No	1	0	1	0	1	0
	1	3	2	2	3	1
ME	0	0	0,5	0	0,3333	0
ME	0		0,25		0,25	
Gain	0,25		0		0	

if Outlook = Overcast		
	- Play Tennis yes	
elif Outlook = Sunny		
	if Temperature <=8	
		- Play Tennis yes
	else Temperature > 8	
		- Play Tennis no
else Outlook = Rain		
	if Wind = Weak	
		- Play Tennis yes
	else Wind = Strong	
		- Play Tennis no

	Outlook	Temp	Wind	Play Tennis	Predicted
11	Sunny	14	Strong	Yes	No
12	Overcast	13	Strong	Yes	Yes
13	Overcast	26	Weak	Yes	Yes
14	Rain	16	Strong	No	No

TP: 2

FP: 0

FN: 1

TN: 1

Accuracy = 3/4

Precision = $TP / (TP + FP) = 2/2$

Recall = $TP / (TP + FN) = 2/3$

Sex	Lies	Cookies	Present
F	10	Milk	Y
M	5	Dark	N
F	2	Milk	Y
F	3	Dark	Y
M	8	Milk	N
M	3	No	Y
M	10	Dark	N
F	2	No	N
M	1	No	Y