Introduction to Data Mining

Chapter 3

Classification:

Basic Concepts and Techniques

Source: revised from slides provided by Tan, Steinbach, Karpatne, and Kumar

Classification: Definition

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

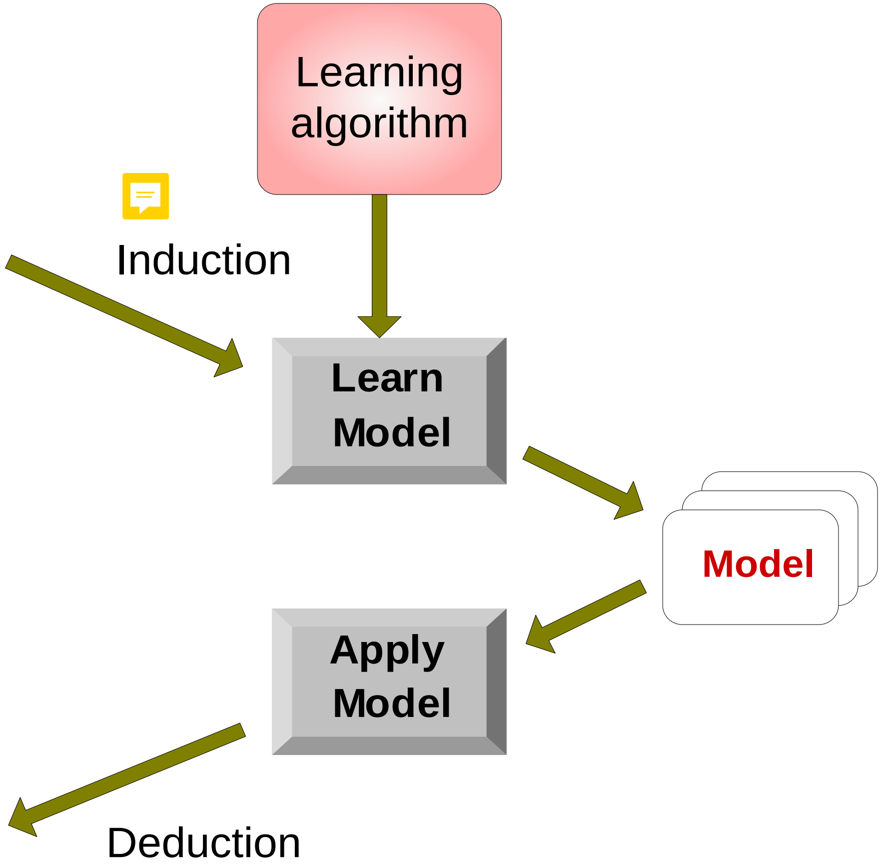
Examples of Classification Task

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

General Approach for Building Classification Model

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

Training Set



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

Test Set

Classification Techniques

- Base Classifiers
  - Decision Tree based Methods
  - Rule-based Methods
  - Nearest-neighbor
  - Neural Networks
  - Deep Learning
  - Naïve Bayes and Bayesian Belief Networks
  - Support Vector Machines

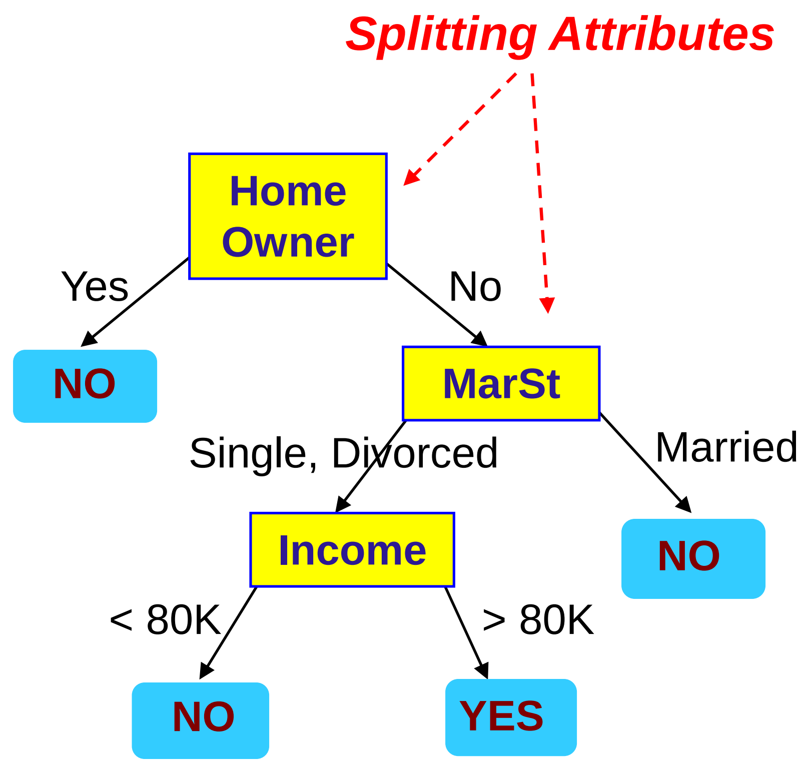
- Ensemble Classifiers
  - Boosting, Bagging, Random Forests

Example of a Decision Tree

categorical categorical continuous class

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

Training Data

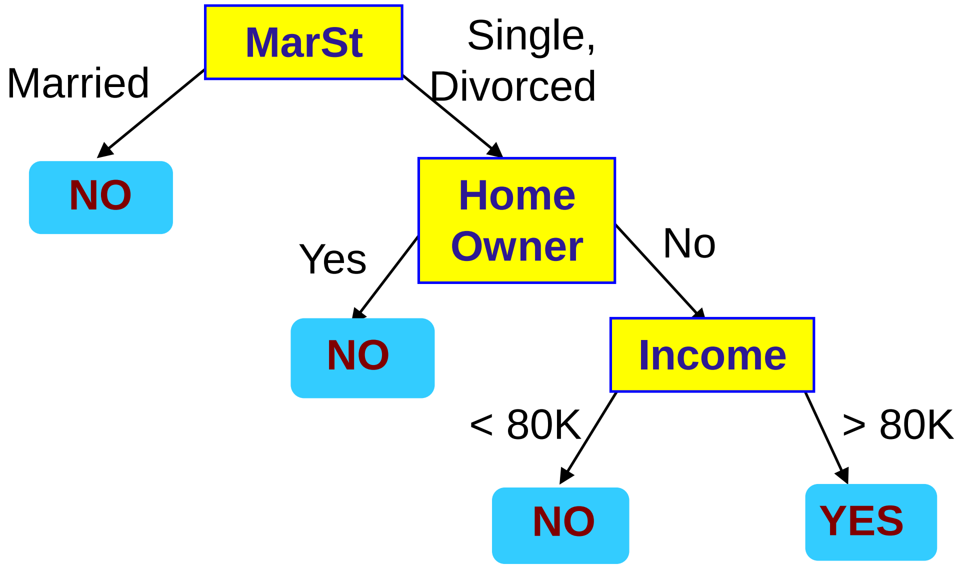


Model: Decision Tree

Another Example of Decision Tree

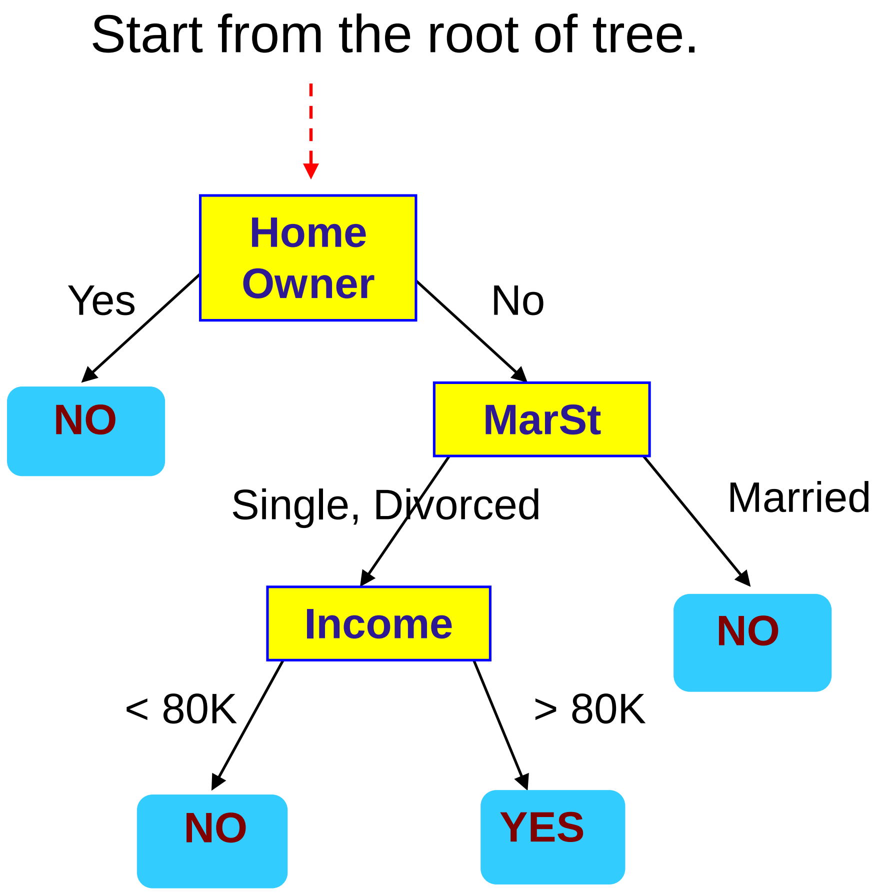
categorical categorical continuous class

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



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

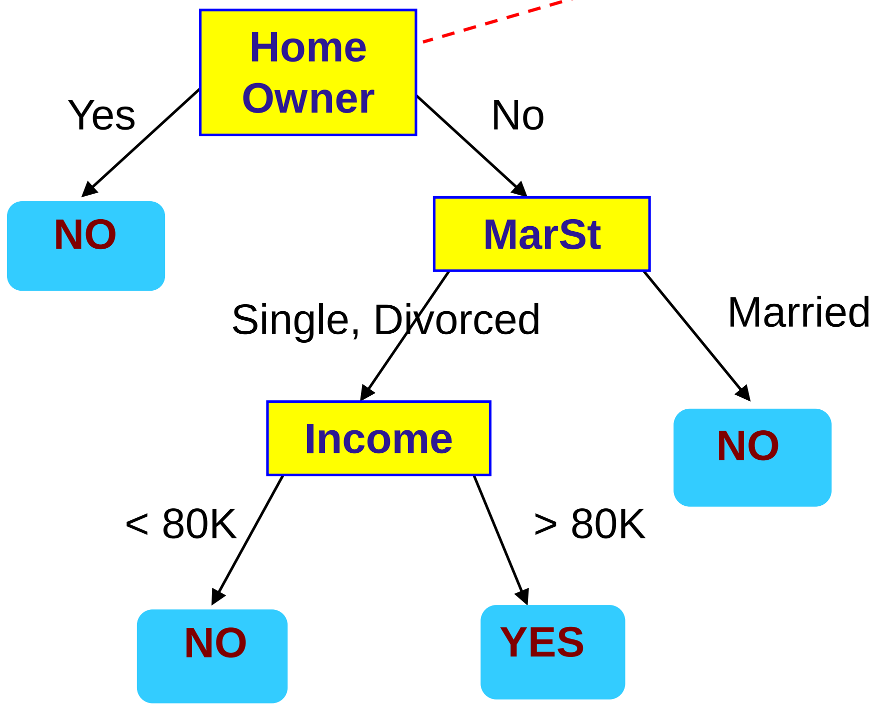
Apply Model to Test Data



Test Data

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

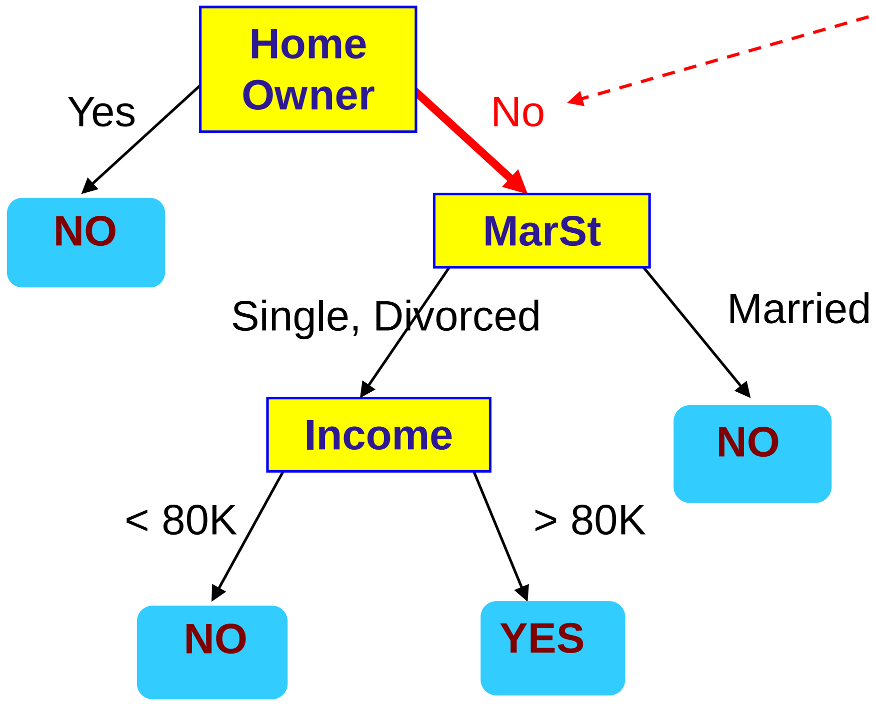
Apply Model to Test Data



Test Data

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

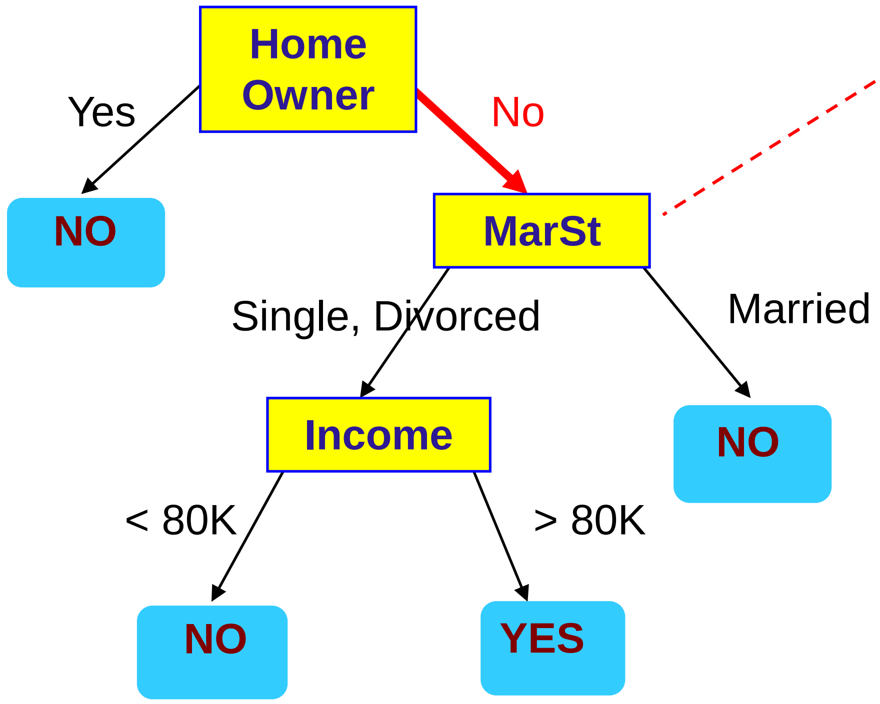
Apply Model to Test Data



Test Data

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

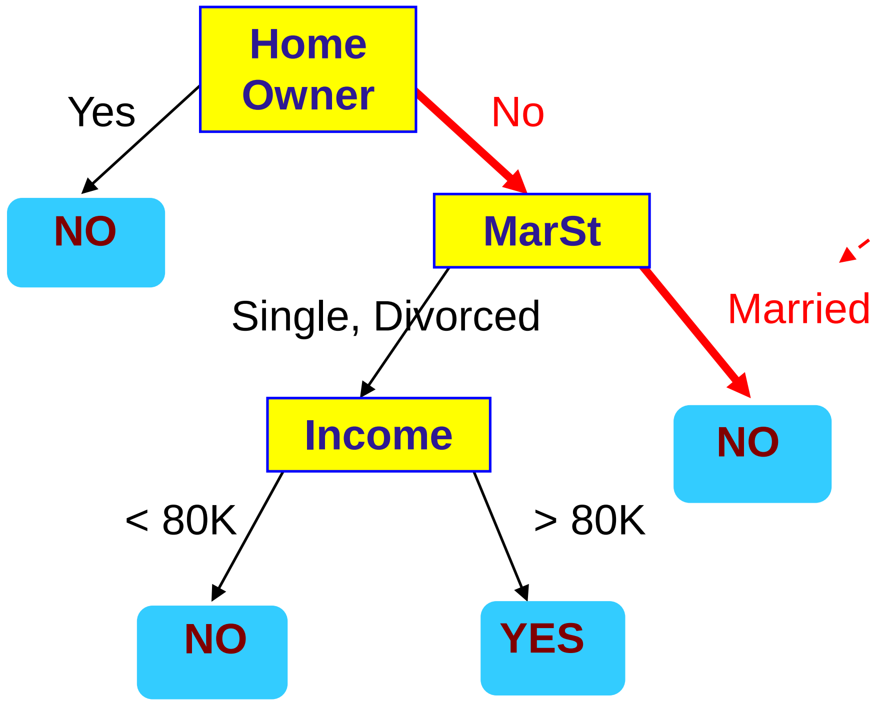
Apply Model to Test Data



Test Data

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

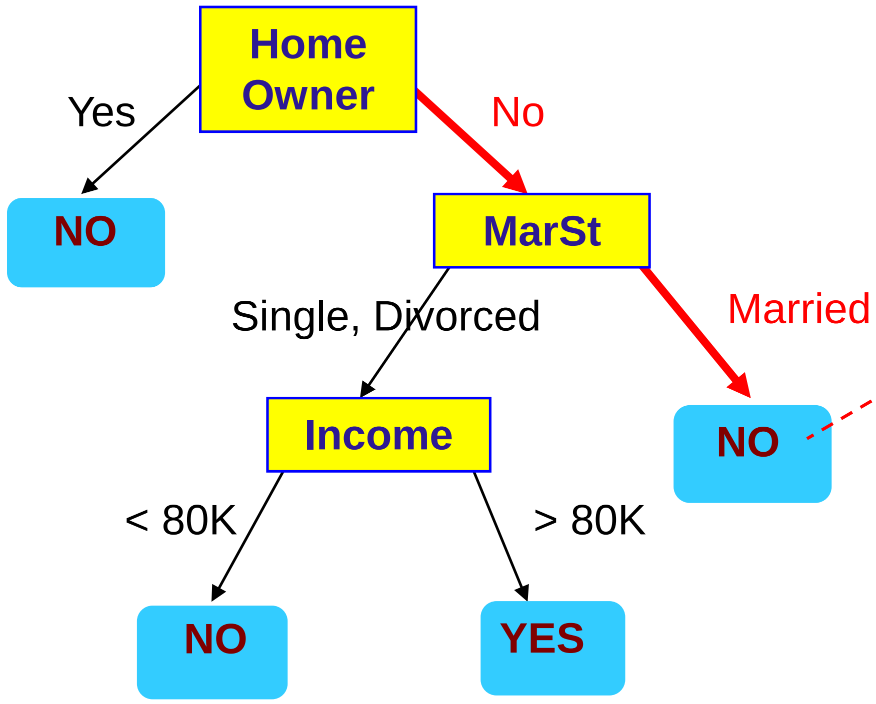
Apply Model to Test Data



Test Data

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

Apply Model to Test Data



Test Data

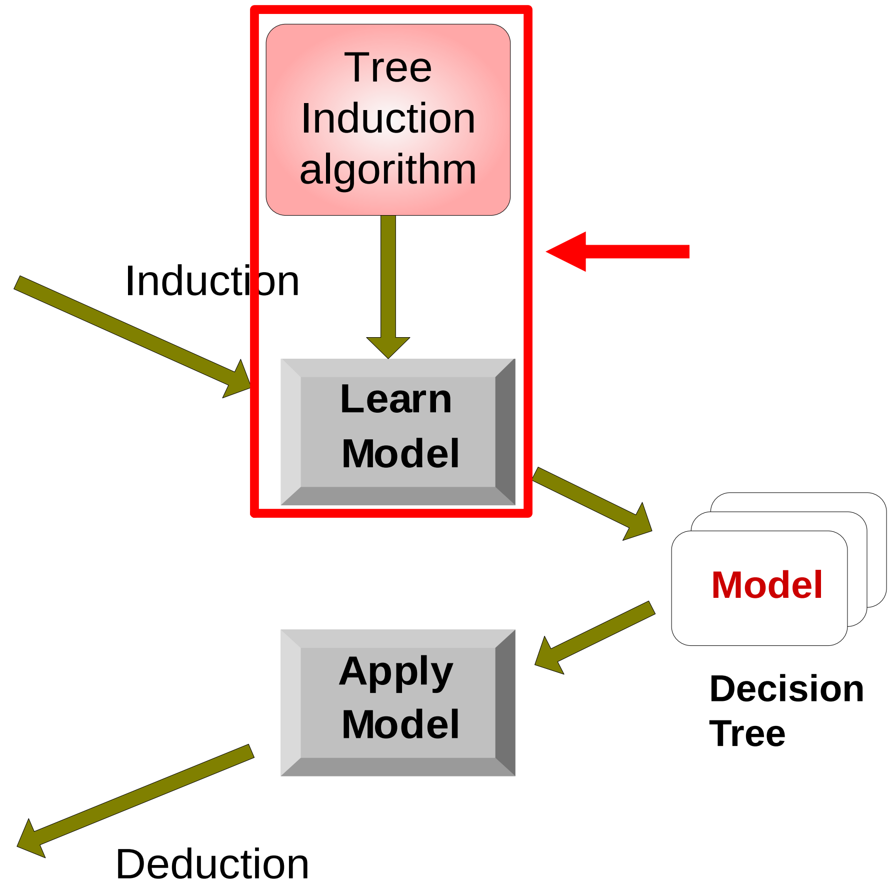
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

Assign Defaulted to "No"

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

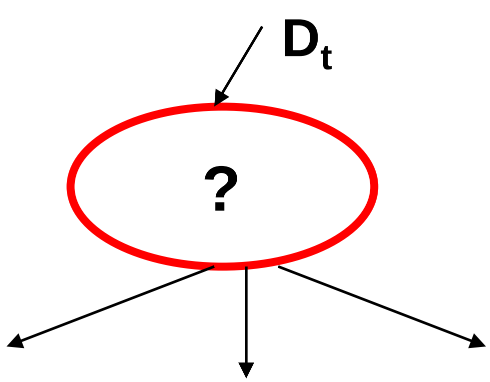
- Many Algorithms:
  - Hunt's Algorithm (one of the earliest)
  - CART
  - ID3, C4.5
  - SLIQ,SPRINT

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

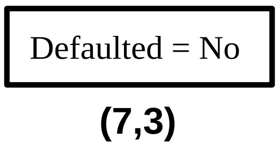
Defaulted = No

(7,3)

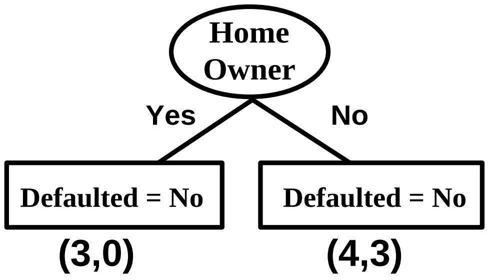
(a)

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



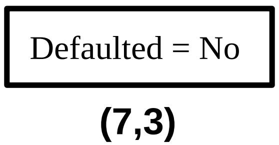
(a)



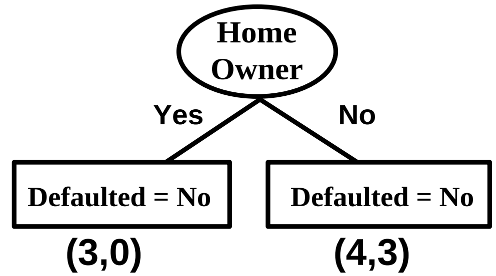
(b)

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

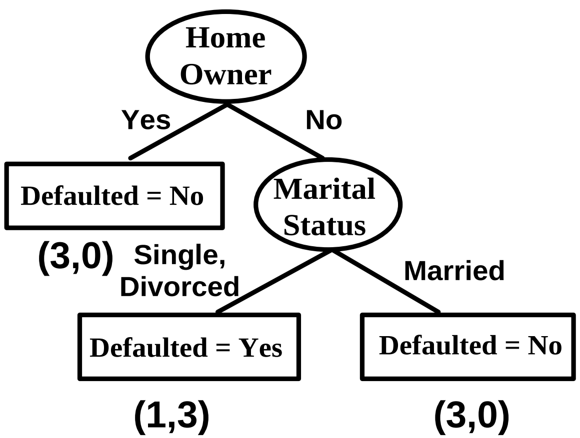
Hunt’s Algorithm



(a)



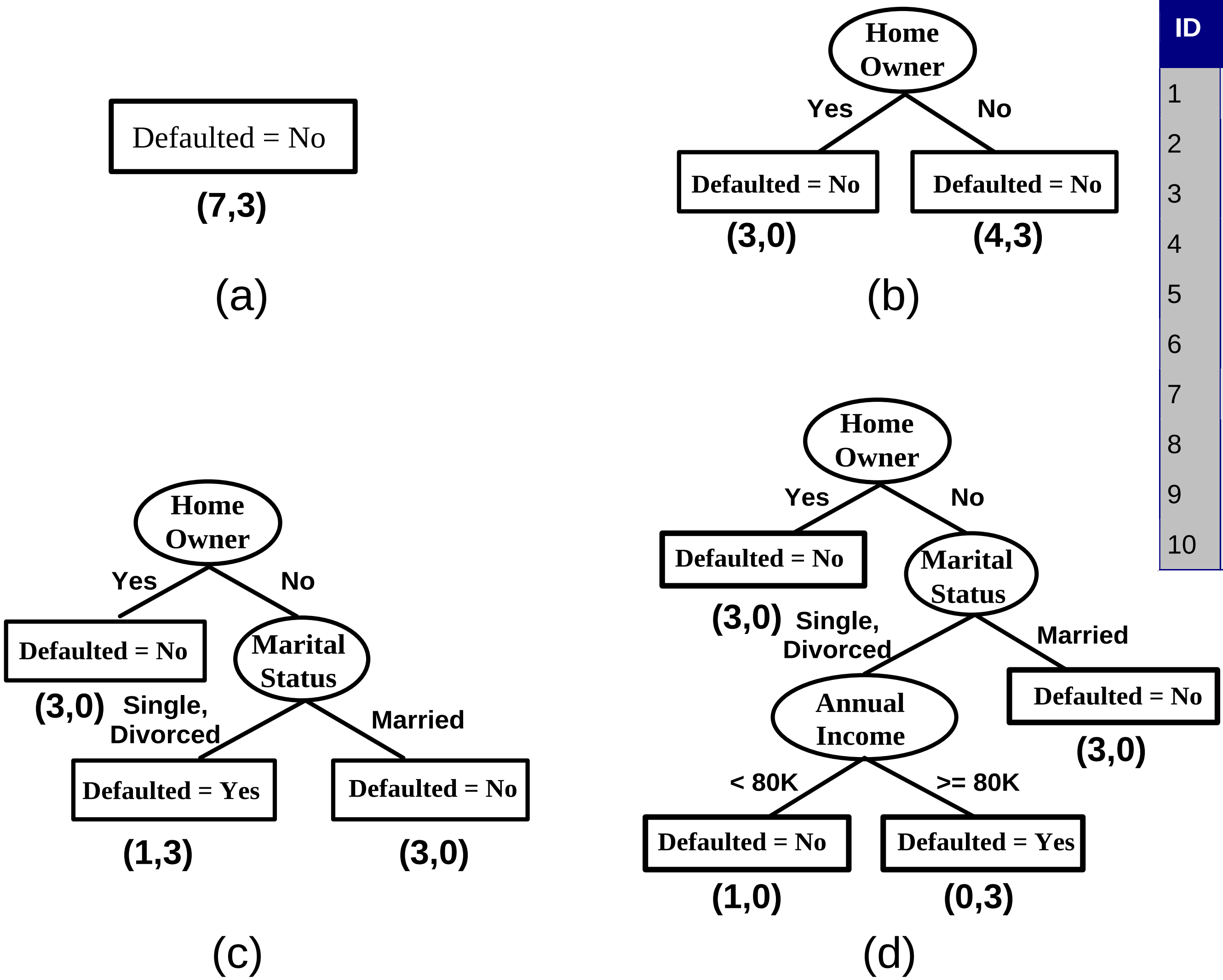
(b)



(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



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

- 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

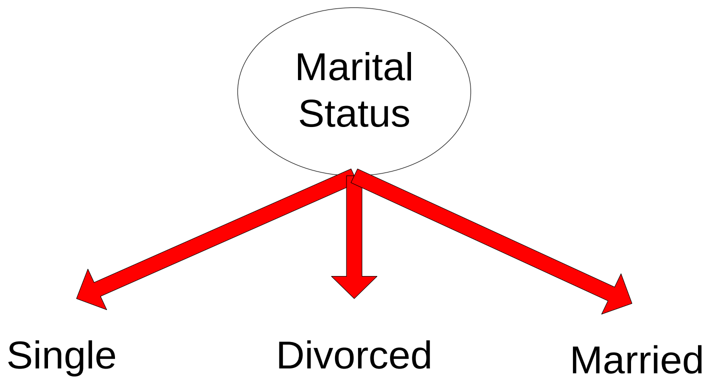
Methods for Expressing Test Conditions

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

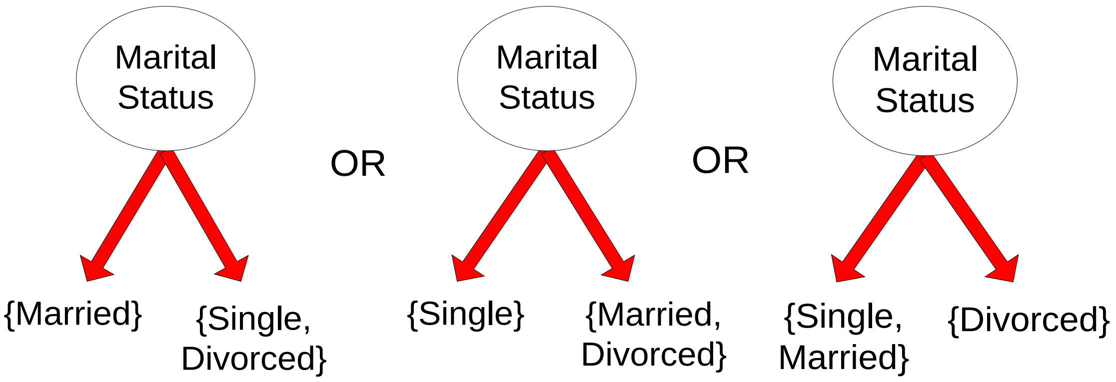
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Test Condition for Nominal Attributes

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

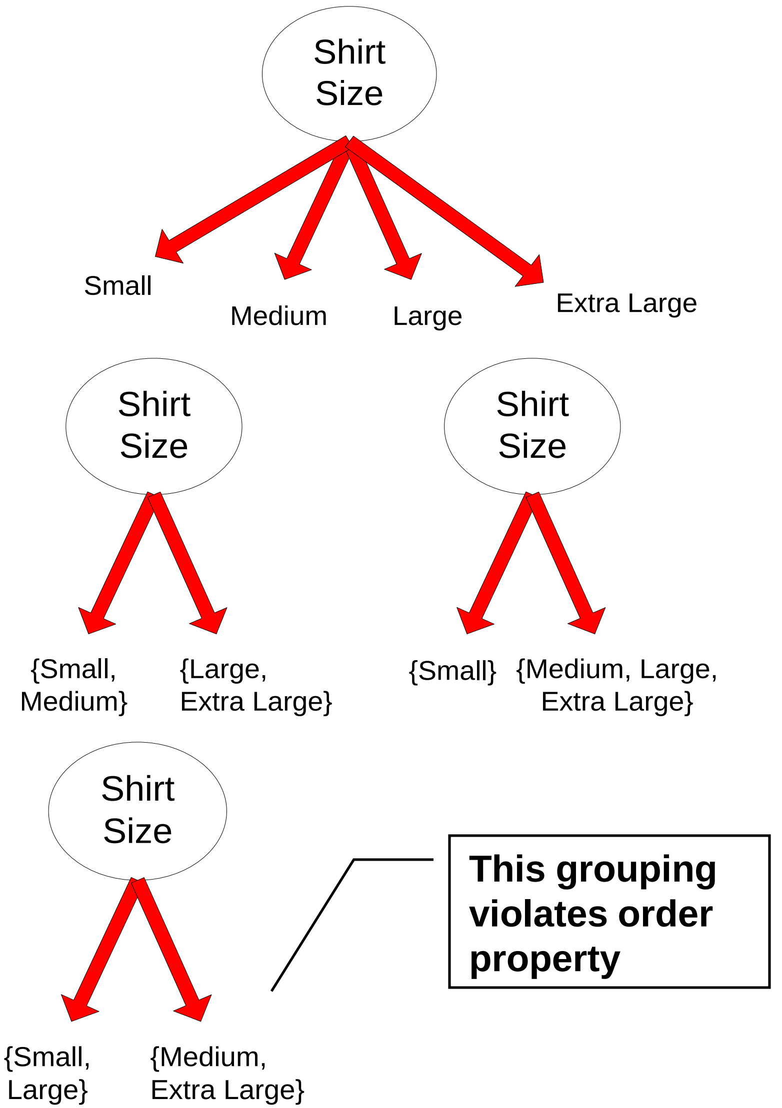


- Binary split:
  - Divides values into two subsets

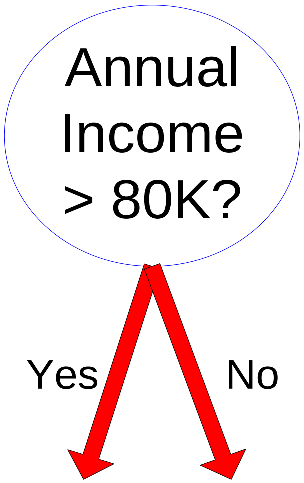


Test Condition for Ordinal Attributes

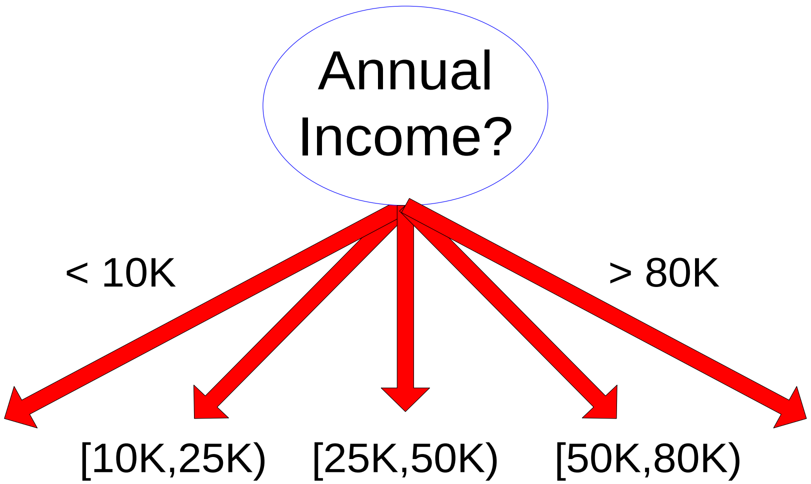
- **Multi-way split:**
  - Use as many partitions as distinct values
- **Binary split:**
  - Divides values into two subsets
  - Preserve order property among attribute values



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.

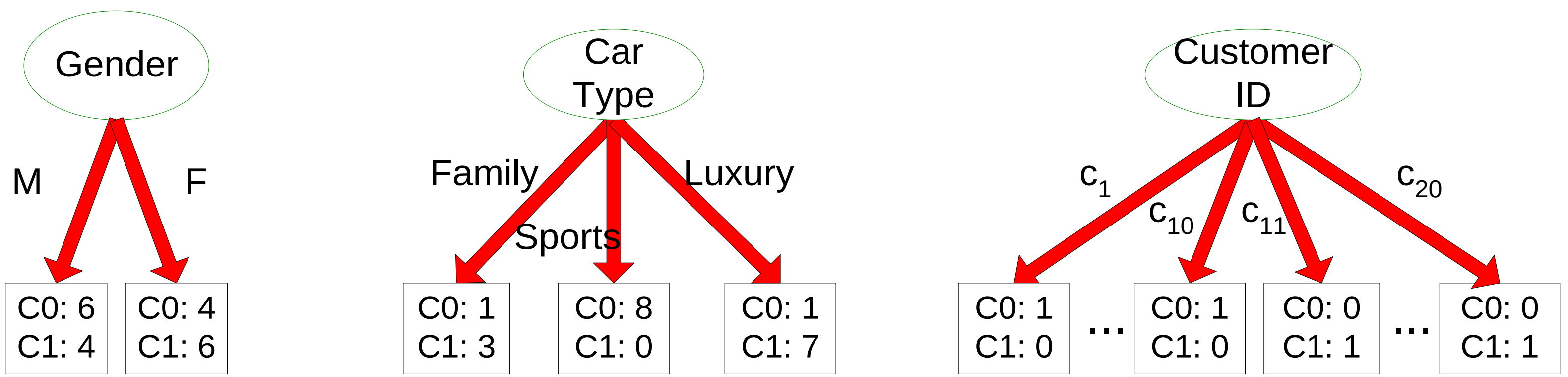
    - Static – discretize once at the beginning
    - Dynamic – repeat at each node

  - 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 purer class distribution are preferred

- Need a measure of node impurity:

C0: 5	C0: 9
C1: 5	C1: 1
High degree of impurity	**Low degree of impurity**

Measures of Node Impurity

- Gini Index

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

- Entropy

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

- Misclassification error

$$Error(t) = 1 - \max_j p(j \mid t)$$

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

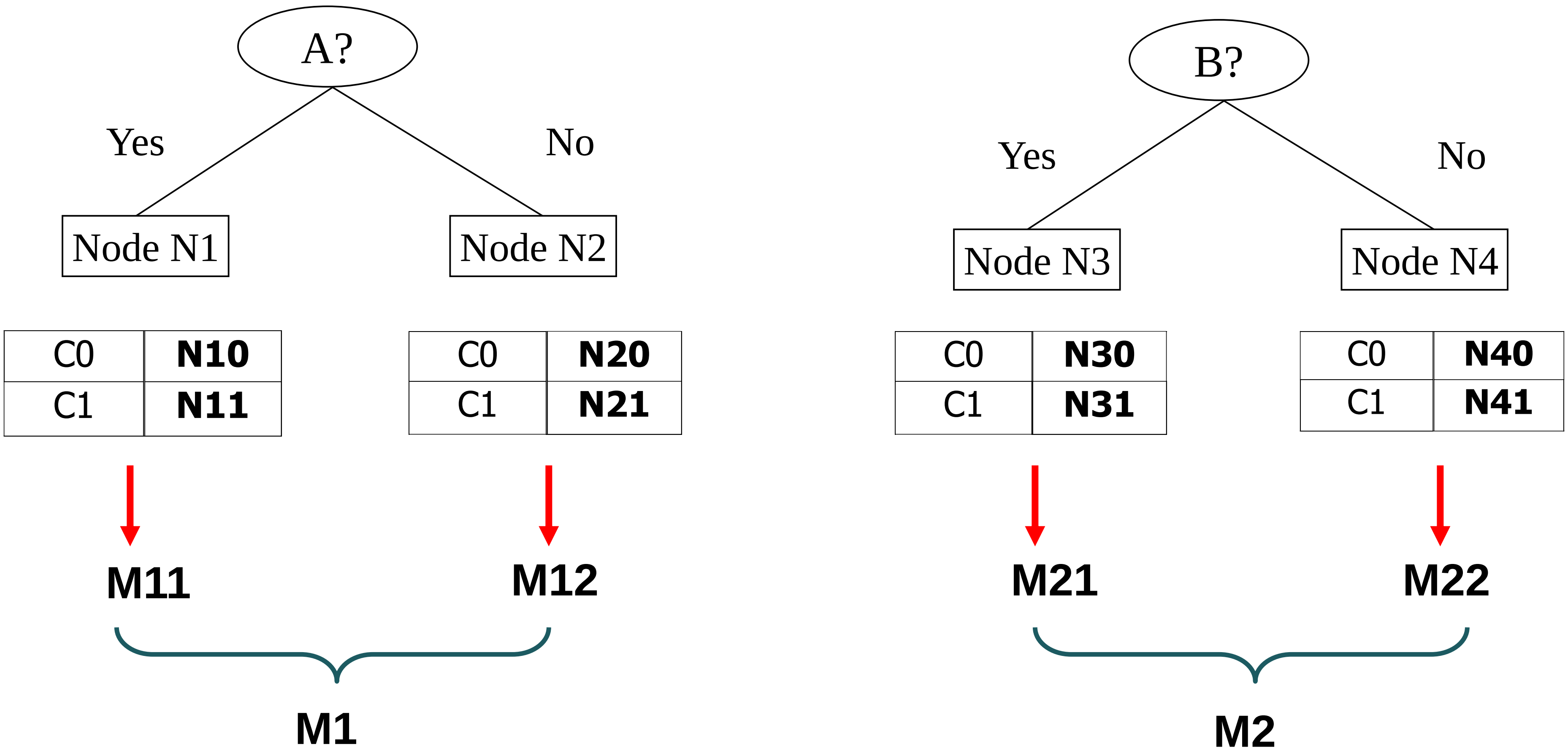
   Gain = P – M

   or equivalently, lowest impurity measure after splitting (M)

Finding the Best Split

Before Splitting:

C0	**N00**	→ **P**
C1	**N01**	



Gain = P – M1 vs P – M2

Measure of Impurity: GINI

- Gini Index for a given node t :

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

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

  – <u>Maximum (1 - 1/n_c)</u> when records are <u>equally distributed</u> among all classes, implying least interesting information
  – <u>Minimum (0.0)</u> when <u>all records belong to one class</u>, implying most interesting information

Measure of Impurity: GINI

- Gini Index for a given node t :

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

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

- 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

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

C1	**0**
C2	**6**

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

Gini = 1 – p(C1)^2 – p(C2)^2 = 1 – 0 – 1 = 0

C1	**1**
C2	**5**

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

Gini = 1 – (1/6)^2 – (5/6)^2 = 0.278

C1	**2**
C2	**4**

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

Gini = 1 – (2/6)^2 – (4/6)^2 = 0.444

Computing Gini Index for a Collection of Nodes

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

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

where, n_i = number of records at child i,

n = number of records at parent node p.

- Choose the attribute that minimizes weighted average Gini index of the children

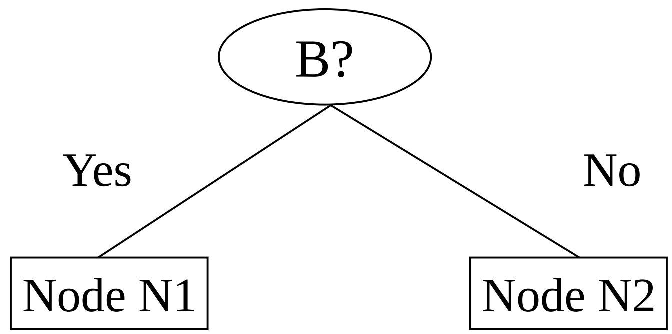
- Gini index is used in decision tree algorithms such as CART, SLIQ, SPRINT

Binary Attributes: Computing GINI Index

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

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

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



	Parent
C1	**7**
C2	**5**
Gini = 0.486	

Gini(N1)
$= 1 – (5/6)^2 – (1/6)^2$
$= 0.278$

Gini(N2)
$= 1 – (2/6)^2 – (4/6)^2$
$= 0.444$

	N1	**N2**
C1	**5**	**2**
C2	**1**	**4**
Gini=0.361		

Weighted Gini of N1 N2
= 6/12 * 0.278 +
6/12 * 0.444
= 0.361

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 (±1)
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions, A ≤ v and A > v
- Simple method to choose best v
  - For each v, scan the database to gather count matrix and compute its Gini index
  - Computationally Inefficient! Repetition of work.

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

Annual Income ?

	≤ 80	> 80
Defaulted Yes	0	3
Defaulted No	3	4

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

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	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>

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	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

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	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		***0.300***		0.343		0.375		0.400		0.420	

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

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

C1	**0**
C2	**6**

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

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

C1	**1**
C2	**5**

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

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

C1	**2**
C2	**4**

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

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

Computing Information Gain After Splitting

- Information Gain:

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

Parent Node, p is split into k partitions;

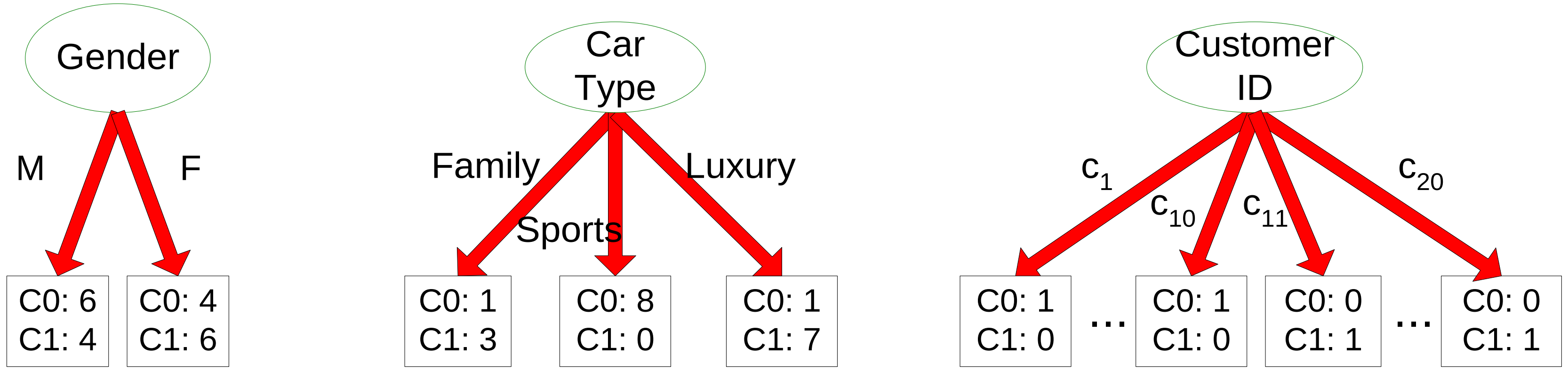
n_i is number of records in partition i

  – Choose the split that achieves most reduction (maximizes GAIN)

  – Used in the ID3 and C4.5 decision tree algorithms

Problem with large number of partitions

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



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

Prevent large number of partitions

- Generate only binary decision trees
  - Avoid the difficulty of handling attributes with varying number of partitions
  - CART

- Modify the splitting criterion
  - Take into account the number of partitions produced by the attribute
  - C4.5 (Gain ratio)

Gain Ratio

- Gain Ratio:

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

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

Parent Node, p is split into k partitions

n_i is the number of records in partition i

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

Example

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

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

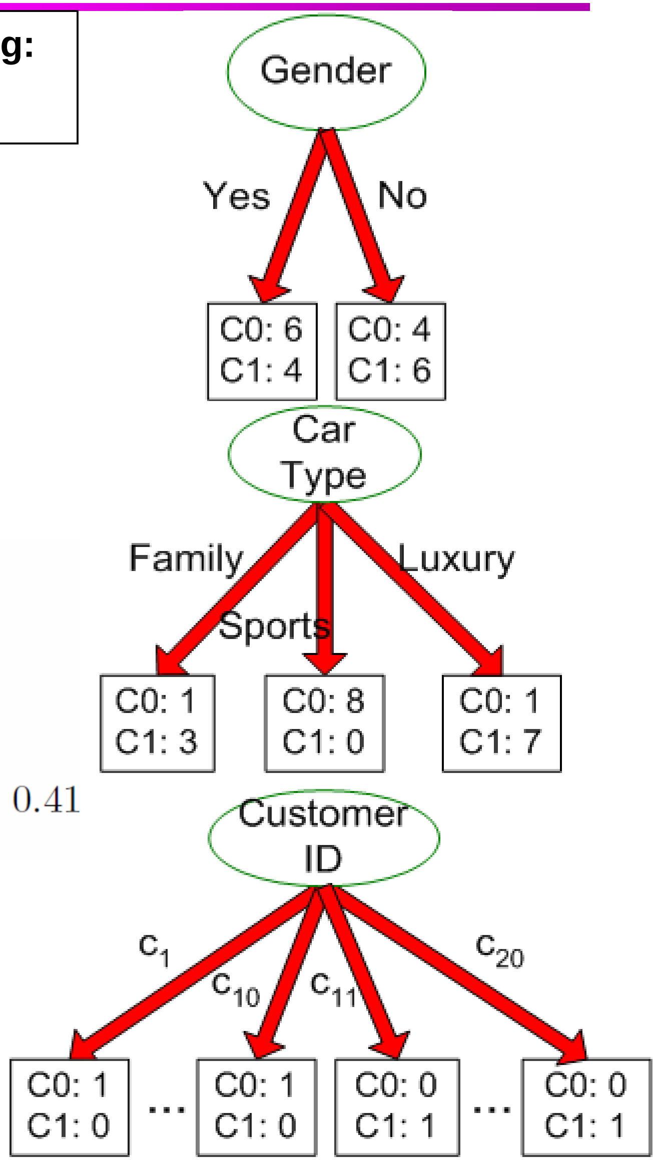
$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO}$$

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

$$\mathrm{Entropy(parent)} = -\frac{10}{20}\log_2\frac{10}{20} - \frac{10}{20}\log_2\frac{10}{20} = 1$$

Before Splitting:

C0: 10, C1: 10



Gender

$$\begin{aligned}\mathrm{Entropy(children)} &= \frac{10}{20}\left[-\frac{6}{10}\log_2\frac{6}{10} - \frac{4}{10}\log_2\frac{4}{10}\right] \times 2 = 0.971 \\ \mathrm{Gain\ Ratio} &= \frac{1-0.971}{-\frac{10}{20}\log_2\frac{10}{20} - \frac{10}{20}\log_2\frac{10}{20}} = \frac{0.029}{1} = 0.029\end{aligned}$$

Car Type

$$\begin{aligned}\mathrm{Entropy(children)} &= \frac{4}{20}\left[-\frac{1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4}\right] + \frac{8}{20} \times 0 \\ &\quad + \frac{8}{20}\left[-\frac{1}{8}\log_2\frac{1}{8} - \frac{7}{8}\log_2\frac{7}{8}\right] = 0.380 \\ \mathrm{Gain\ Ratio} &= \frac{1-0.380}{-\frac{4}{20}\log_2\frac{4}{20} - \frac{8}{20}\log_2\frac{8}{20} - \frac{8}{20}\log_2\frac{8}{20}} = \frac{0.620}{1.52} = 0.41\end{aligned}$$

Customer ID

$$\begin{aligned}\mathrm{Entropy(children)} &= \frac{1}{20}\left[-\frac{1}{1}\log_2\frac{1}{1} - \frac{0}{1}\log_2\frac{0}{1}\right] \times 20 = 0 \\ \mathrm{Gain\ Ratio} &= \frac{1-0}{-\frac{1}{20}\log_2\frac{1}{20} \times 20} = \frac{1}{4.32} = 0.23\end{aligned}$$

Measure of Impurity: Classification Error

- Classification error at a node t :

$$Error(t) = 1 - \max_j p(j \mid t)$$

  – 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

$$Error(t) = 1 - \max_j p(j \mid t)$$

C1	**0**
C2	**6**

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

Error = 1 – max (0, 1) = 1 – 1 = 0

C1	**1**
C2	**5**

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

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

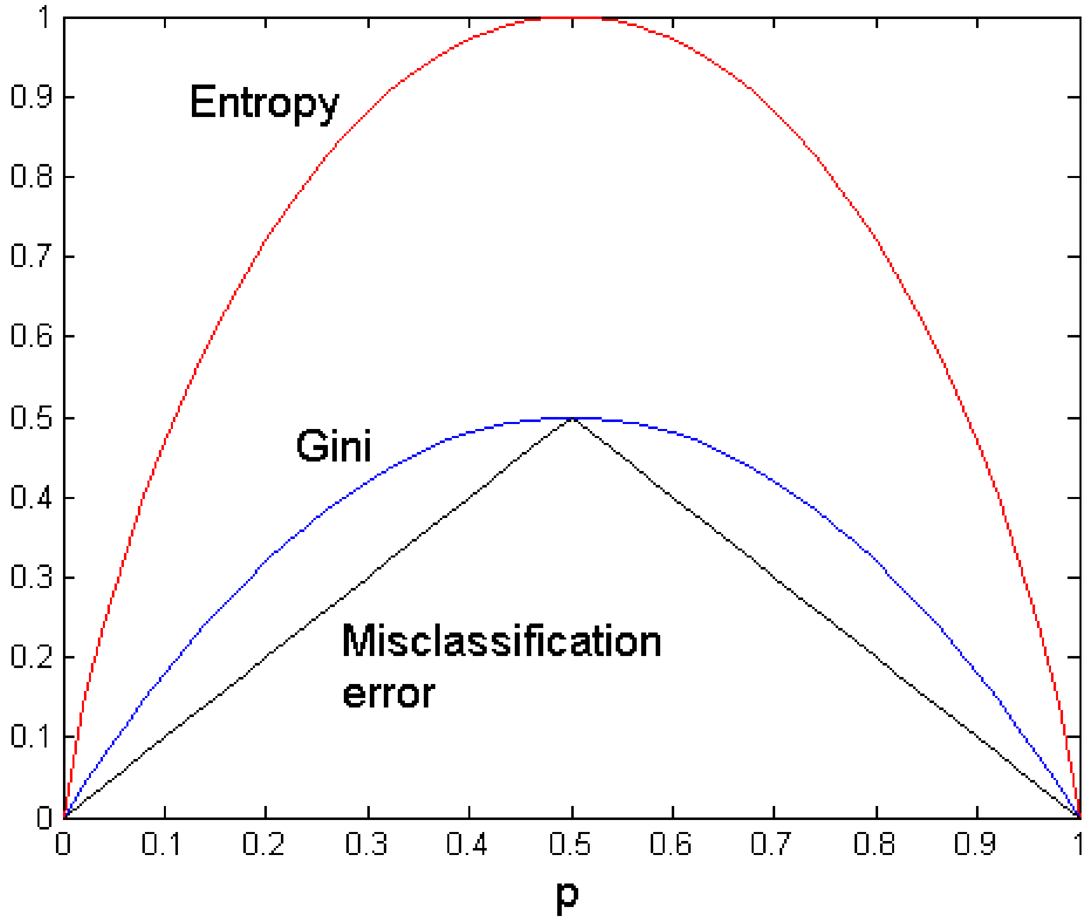
C1	**2**
C2	**4**

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

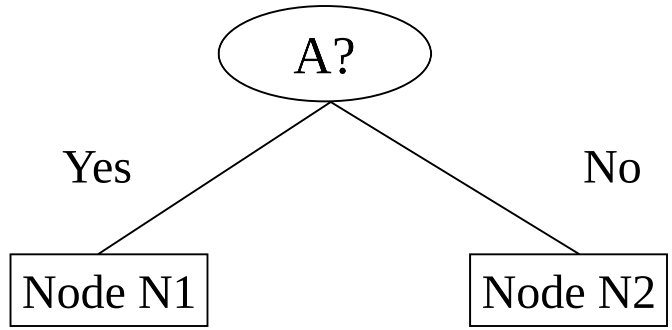
Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Comparison among Impurity Measures

For a 2-class problem:



Misclassification Error vs Gini Index



	Parent
C1	**7**
C2	**3**
Gini = 0.42	

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

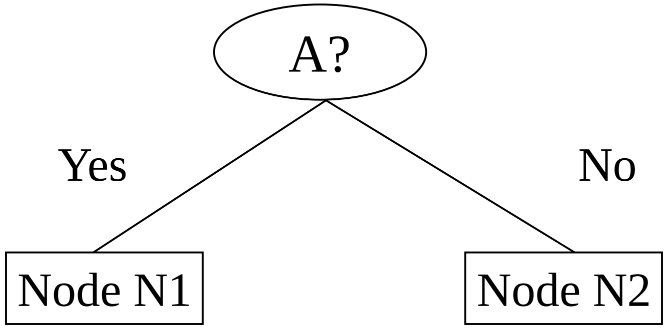
Gini(N2)
= 1 – $(4/7)^2$ – $(3/7)^2$
= 0.489

	N1	N2
C1	**3**	**4**
C2	**0**	**3**
Gini=0.342		

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

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		

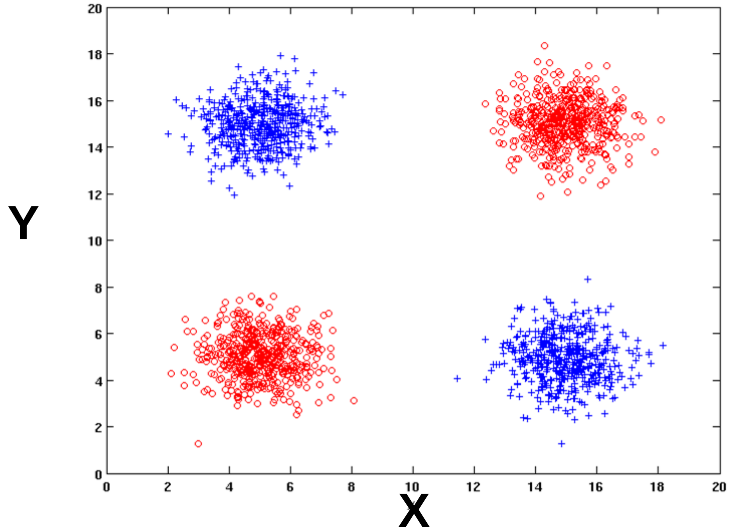
	N1	N2
C1	**3**	**4**
C2	**1**	**2**
Gini=0.416		

Misclassification error for all three cases = 0.3 !

Decision Tree Based Classification

- Advantages:
  - Inexpensive to construct
  - No parameter is required
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Robust to noise (especially when methods to avoid overfitting are employed)
  - Can easily handle redundant or irrelevant attributes (unless the attributes are interacting)
  - Can deal with multiclass problems (vs. binary classifiers)
- 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

Interactions between Attributes

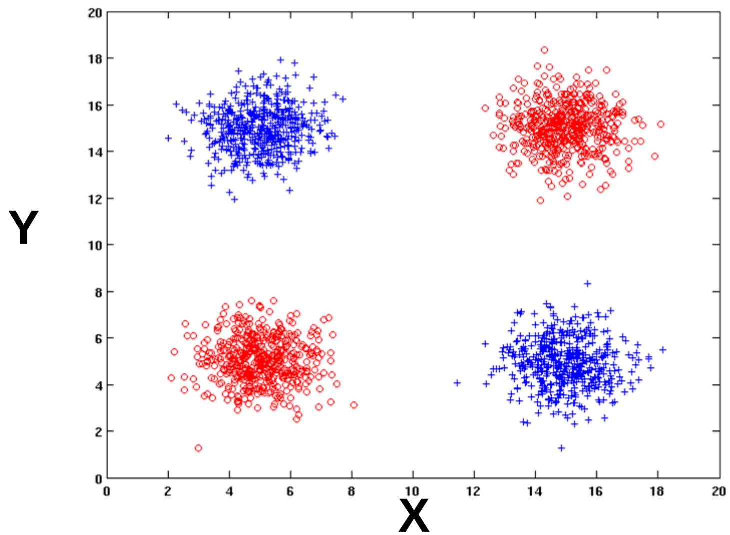


+ : 1000 instances

o : 1000 instances

Entropy (X) : 0.99
Entropy (Y) : 0.99

Interactions between Attributes



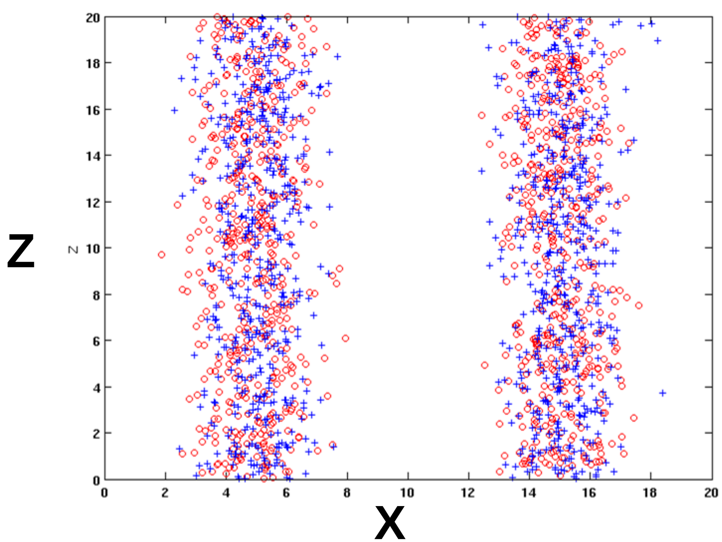
+ : 1000 instances

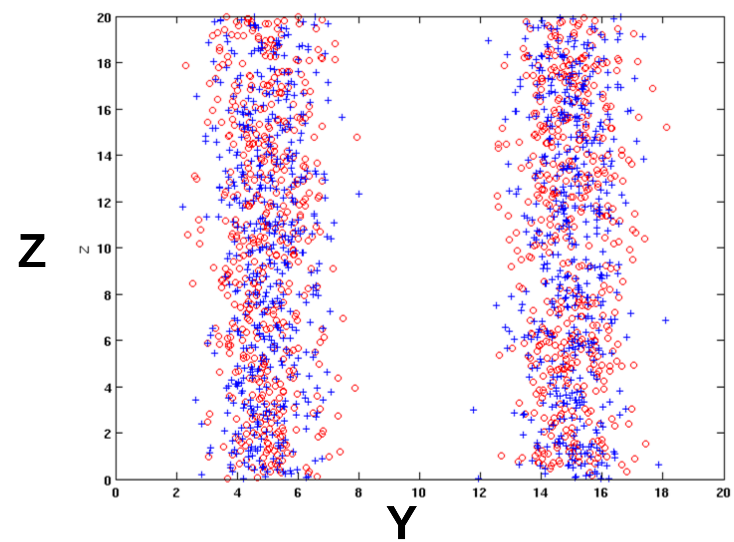
o : 1000 instances

Adding Z as a noisy attribute generated from a uniform distribution

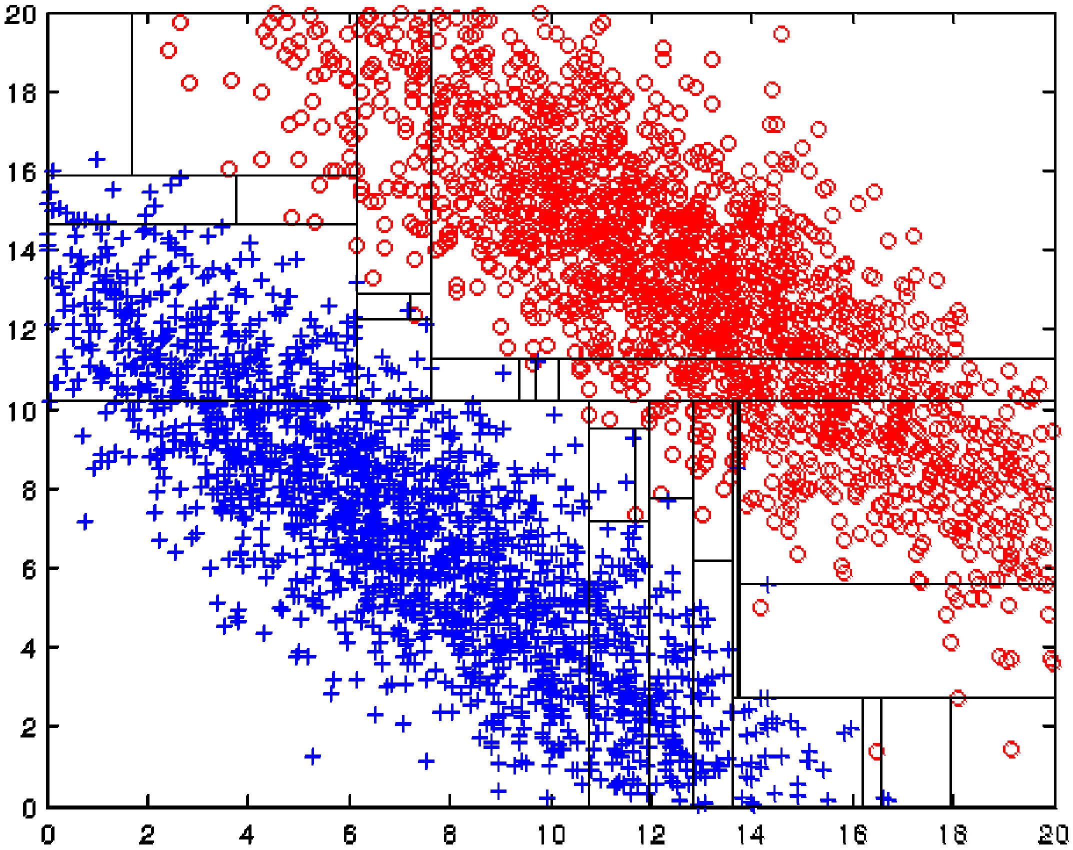
Entropy (X) : 0.99
Entropy (Y) : 0.99
Entropy (Z) : 0.98

Attribute Z will be chosen for splitting!





Limitations of single attribute-based decision boundaries



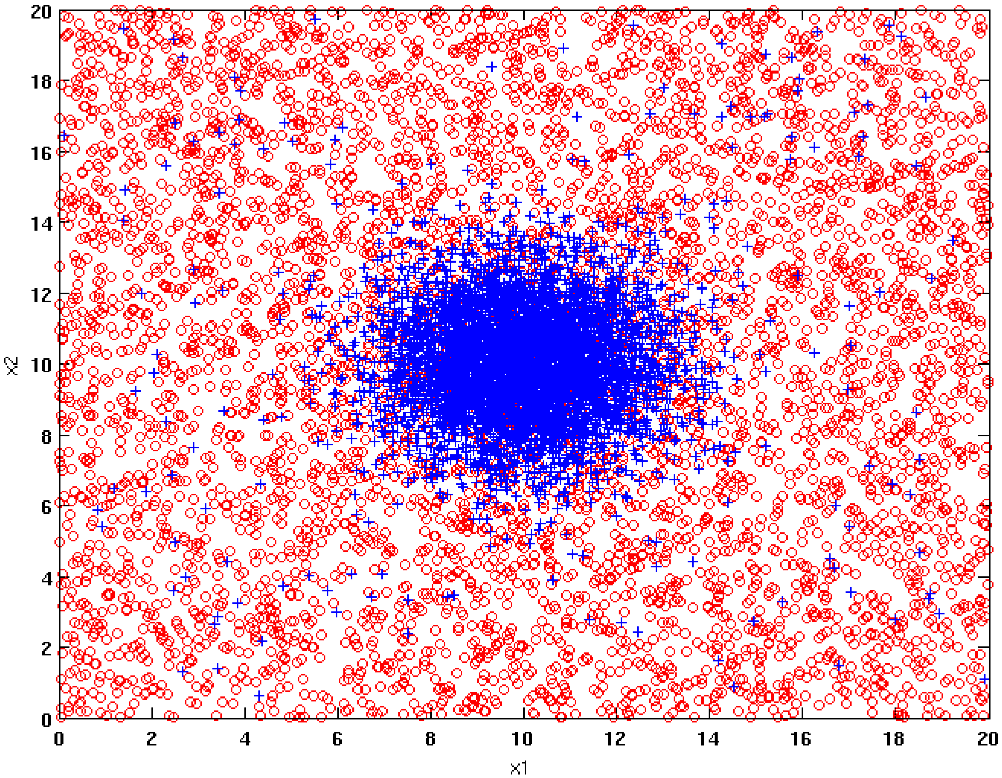
Both positive (+) and negative (o) classes generated from skewed Gaussians with centers at (8,8) and (12,12) respectively.

Test condition:
x +y < 20

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

Example Data Set



Two class problem:

+ : 5200 instances

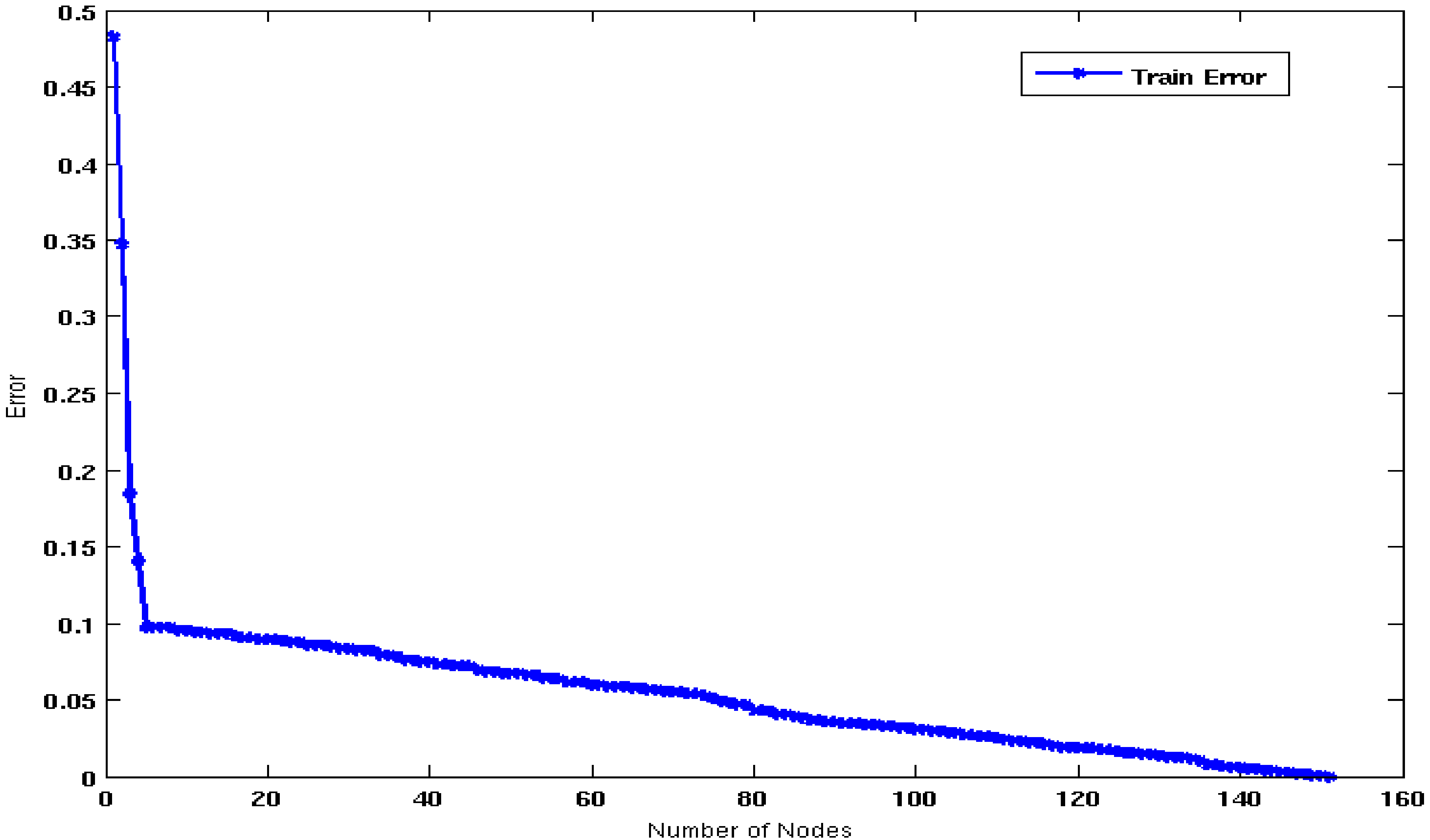
- **5000 instances generated from a Gaussian centered at (10,10)**
- **200 noisy instances added**

o : 5200 instances

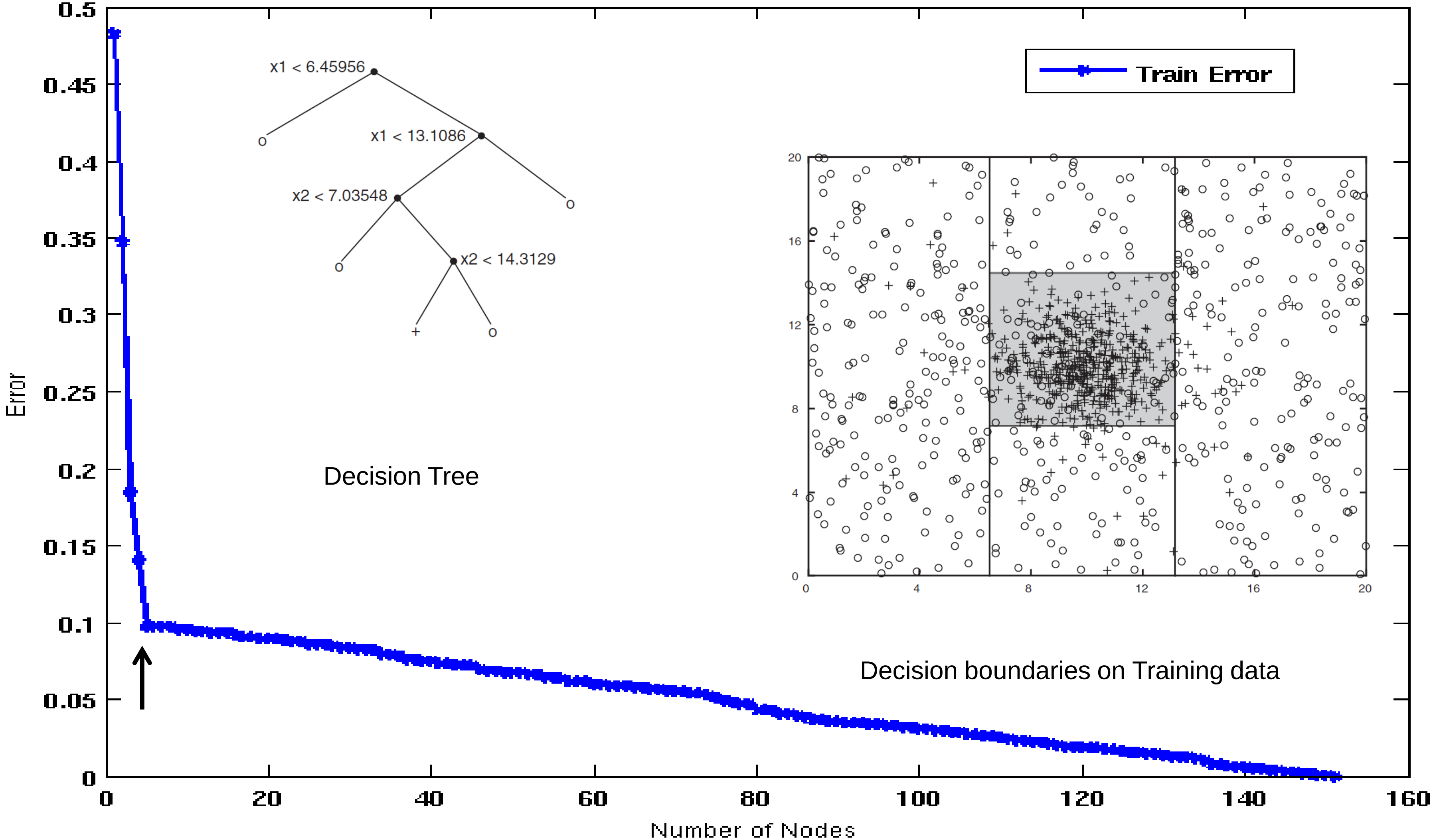
- **Generated from a uniform distribution**

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

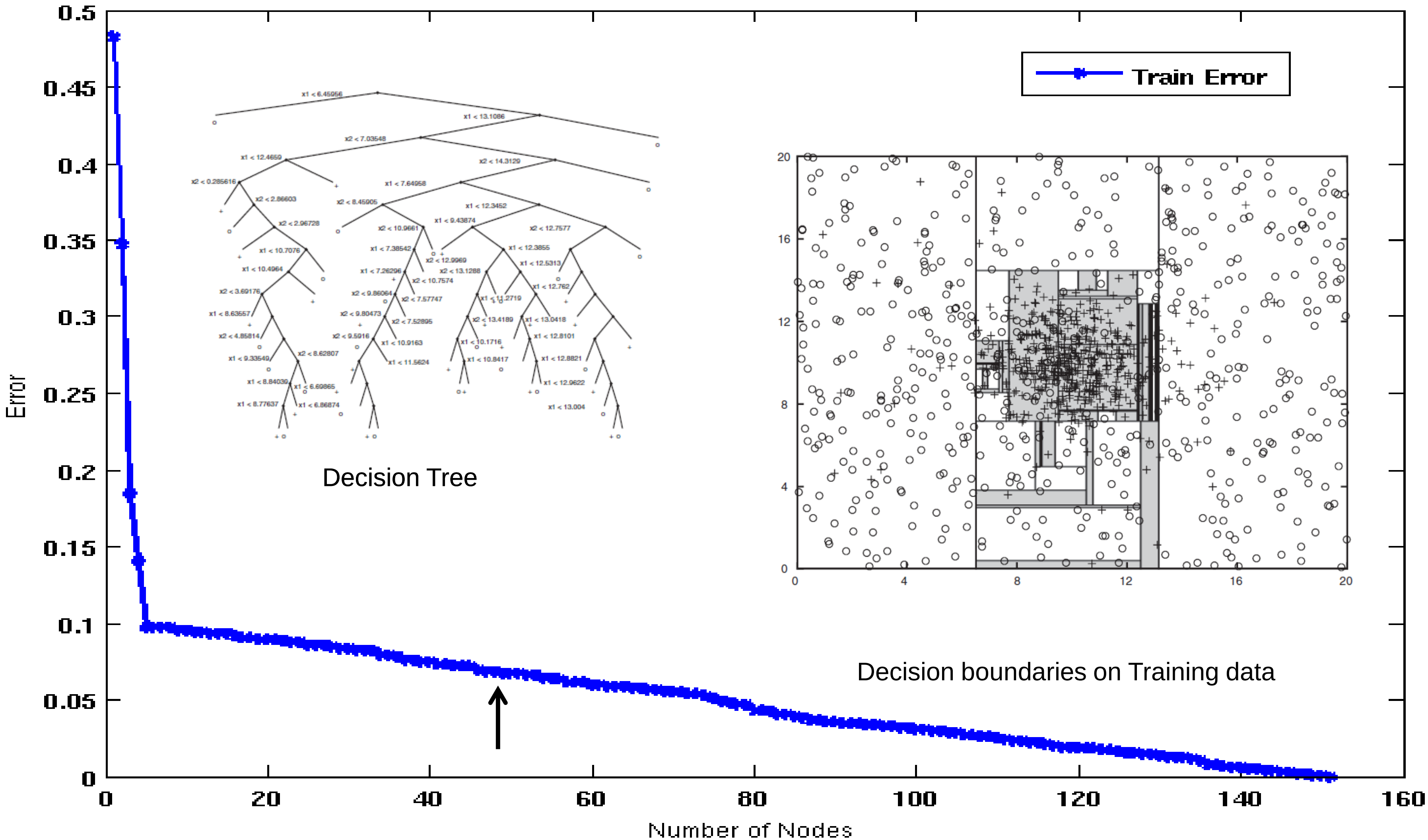
Increasing number of nodes in Decision Trees



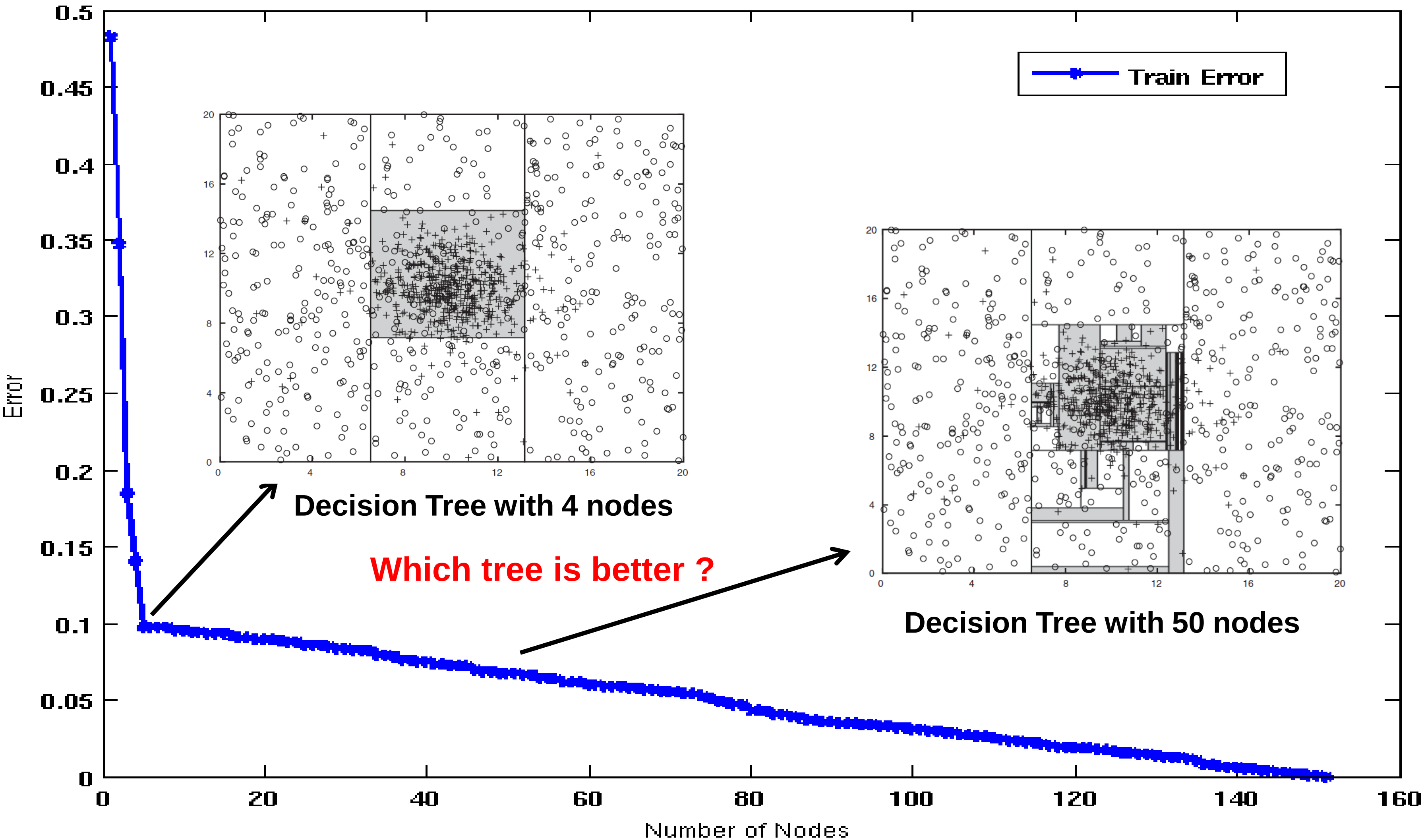
Decision Tree with 4 nodes



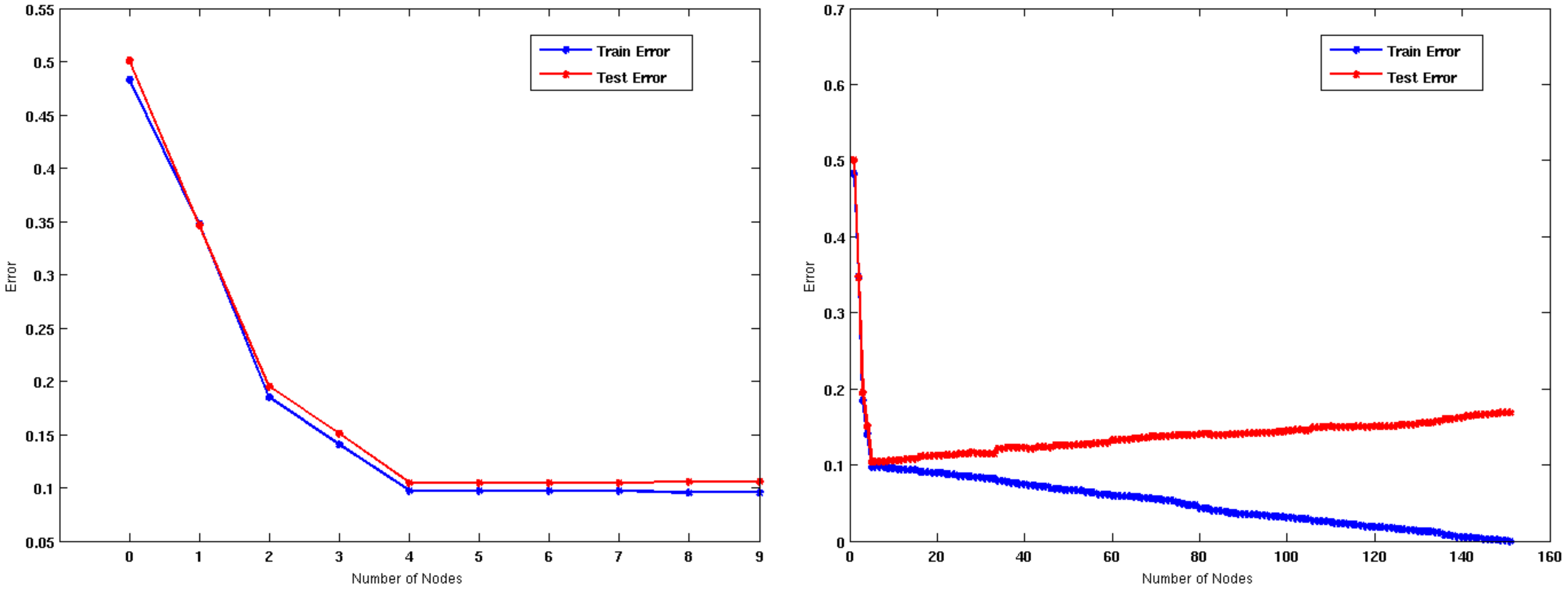
Decision Tree with 50 nodes



Which tree is better?



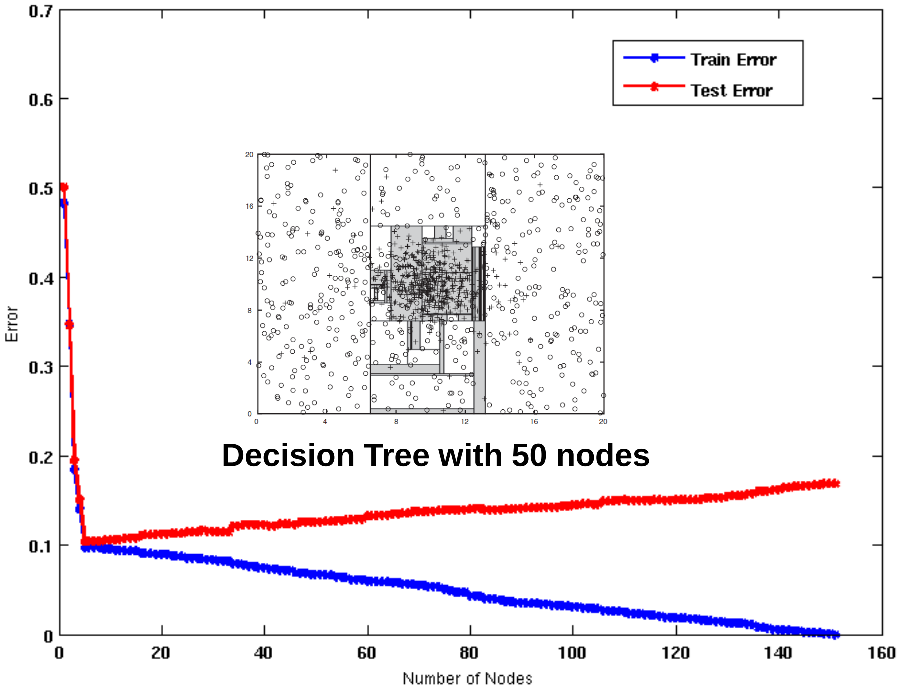
Model Overfitting

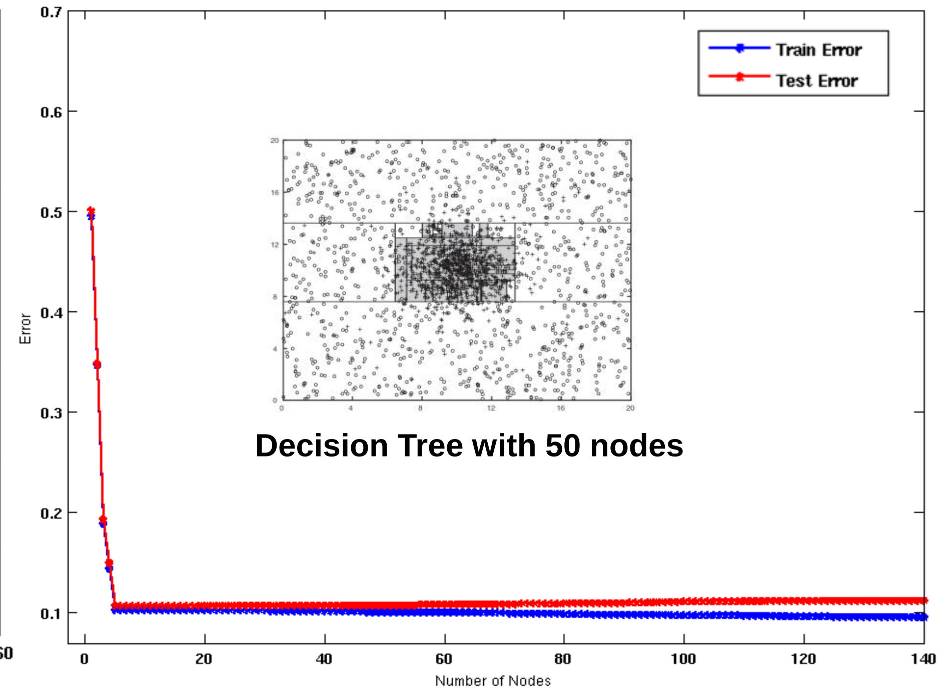


Underfitting: when model is too simple, both training and test errors are large

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

Model Overfitting

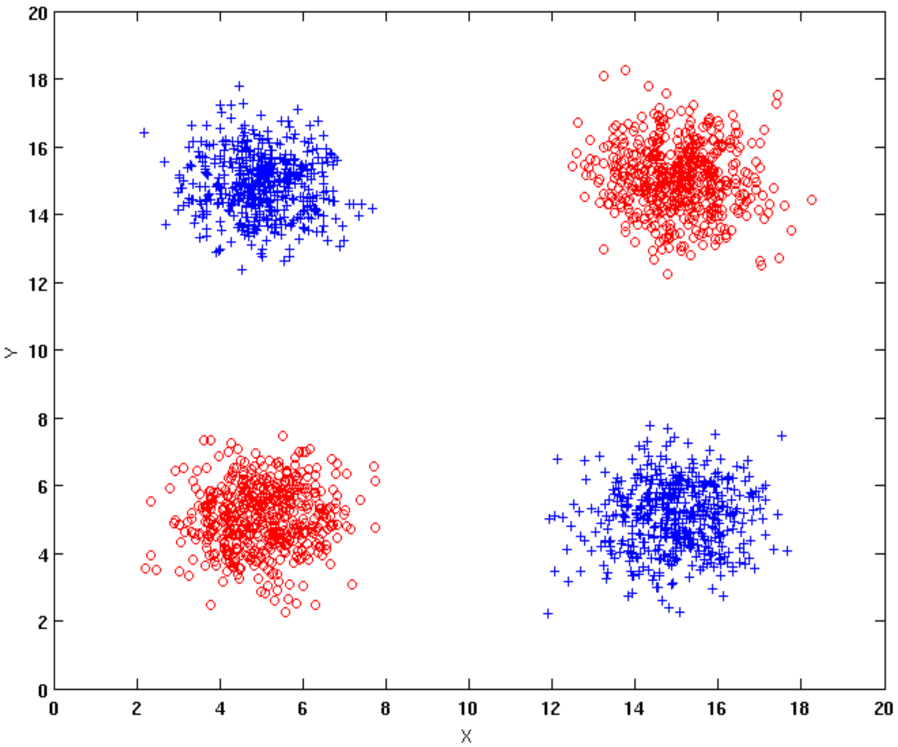


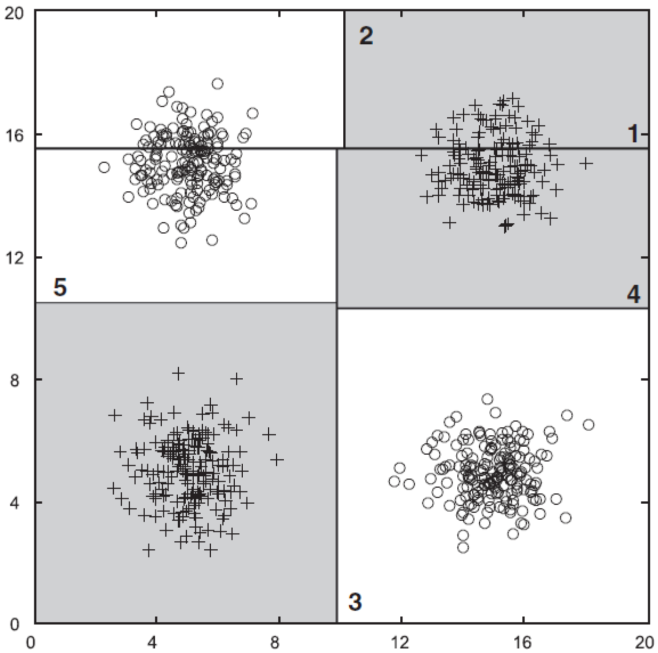


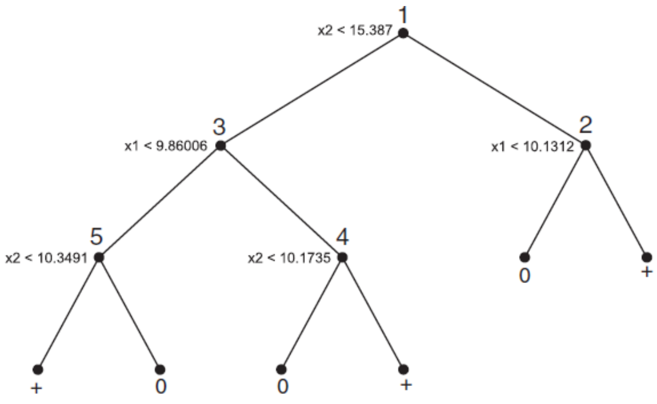
Using twice the number of data instances

- 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 – Another Example



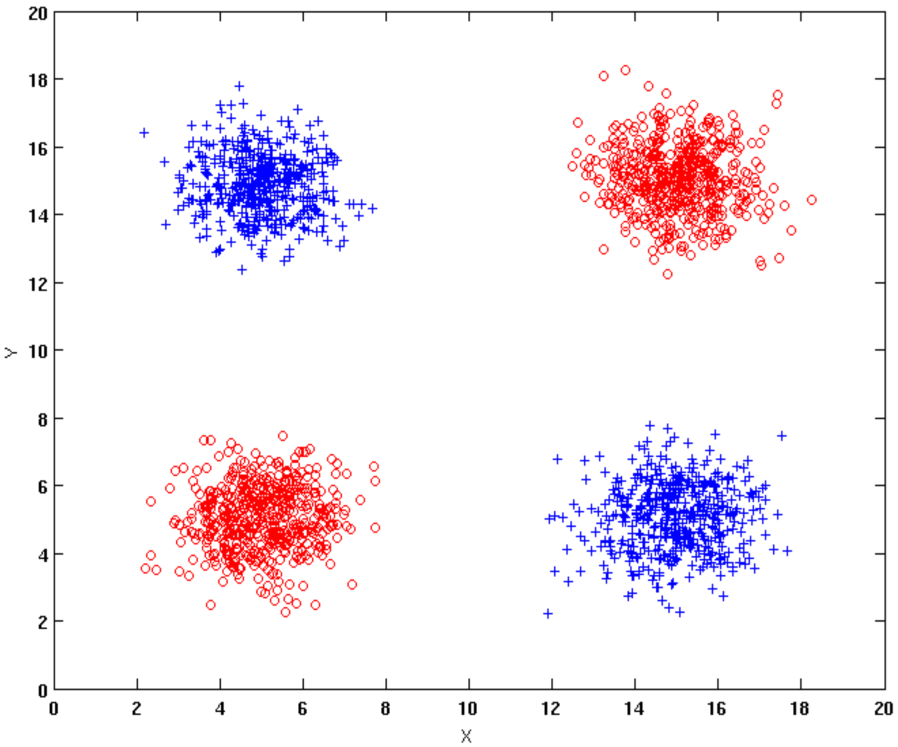




Use 30% of the data for training and 70% of the data for testing

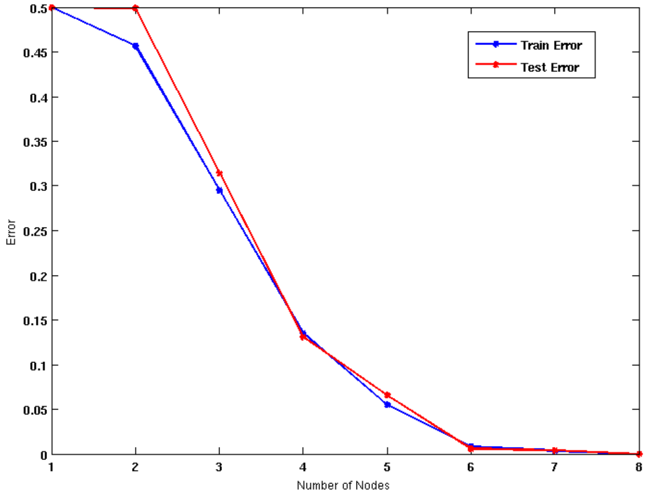
Using only X and Y as attributes

Model Overfitting – Another Example

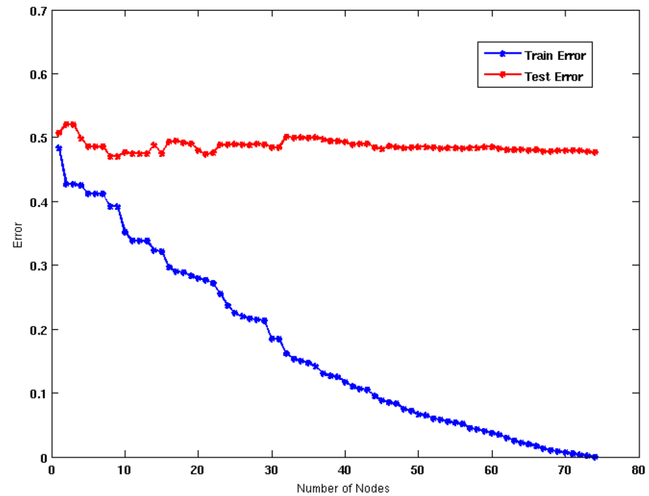


Use 30% of the data for training and 70% of the data for testing

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



Using only X and Y as attributes



Using additional 100 noisy variables

Notes on Overfitting

- Reasons for Model Overfitting
  - Limited Training Size
  - High Model Complexity

- Overfitting results in decision trees that are <u>more complex</u> 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
  - 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

Model Selection:

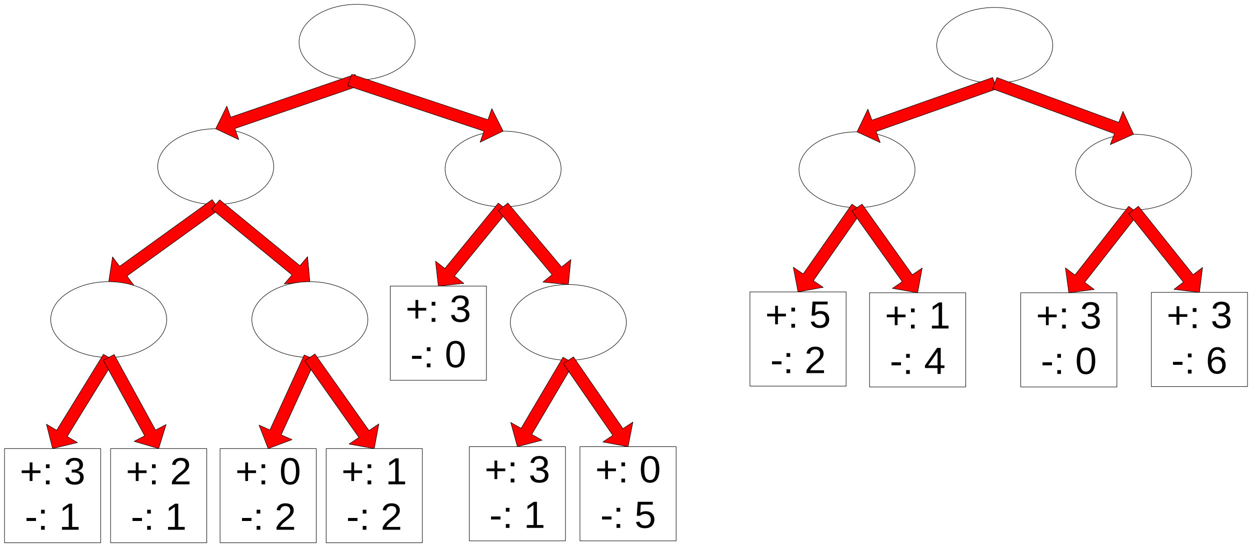
Incorporating Model Complexity

- 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 by errors in data
  - Therefore, one should include model complexity when evaluating a model

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

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

$e(T_L) = 4/24$

$e(T_R) = 6/24$

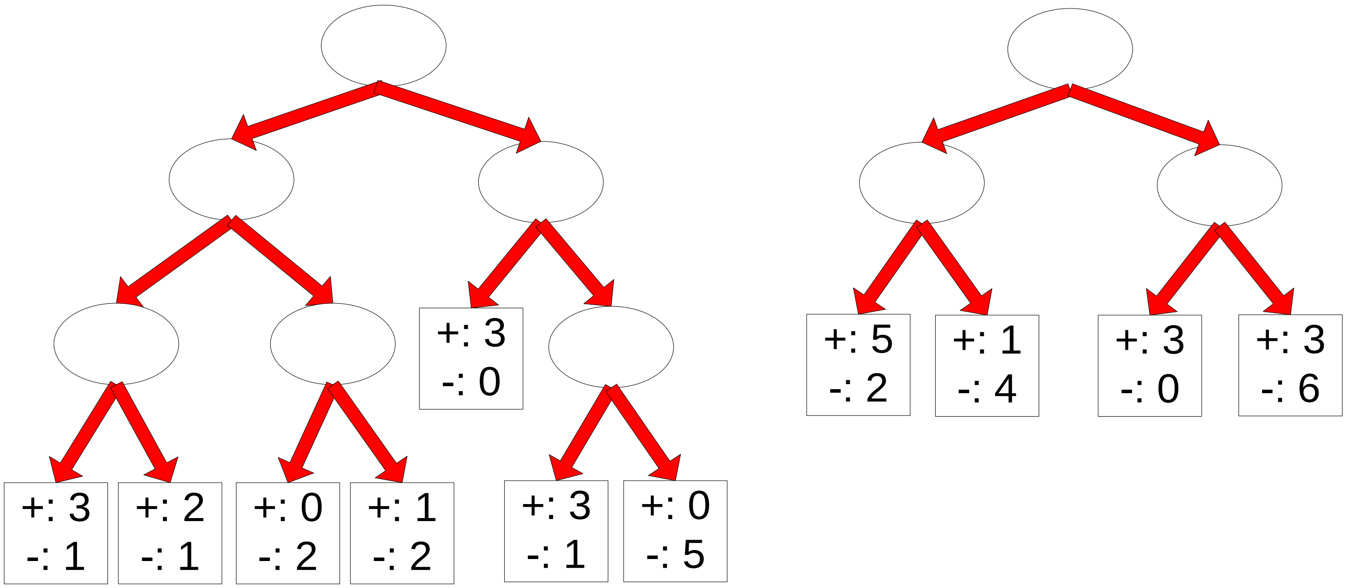
Estimating the Complexity of Decision Trees

- **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

$\mathbf{e(T_L) = 4/24}$

$\mathbf{e(T_R) = 6/24}$

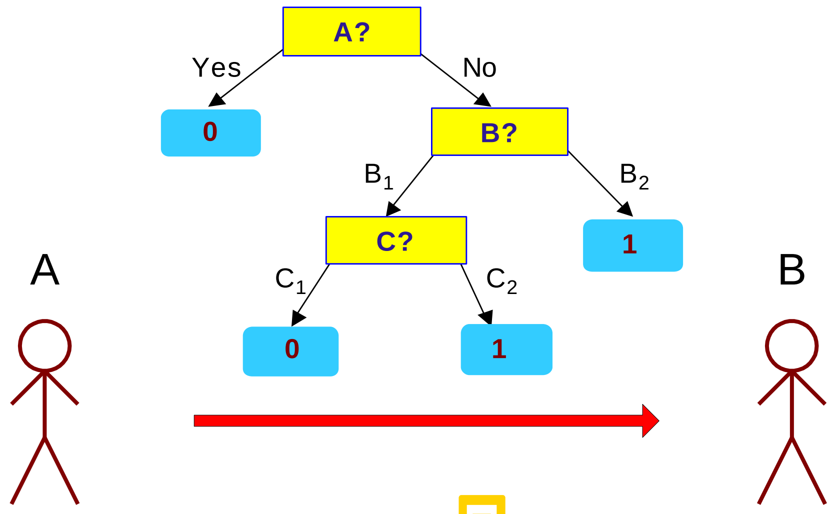
$\mathbf{\Omega = 1}$

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

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

Minimum Description Length (MDL)

X	y
X_1	1
X_2	0
X_3	0
X_4	1
…	…
X_n	1

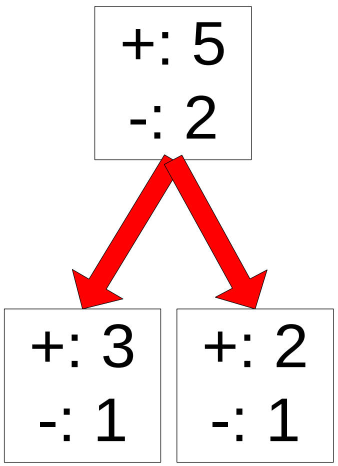


X	y
X_1	?
X_2	?
X_3	?
X_4	?
…	…
X_n	?

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

Estimating Statistical Bounds

Binomial distribution



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

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

Therefore, do not split

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
  - Subtree raising
    - Replace subtree with most frequently used branch

Example of Post-Pruning

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

Training Error (Before splitting) = 10/30

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

Training Error (After splitting) = 9/30

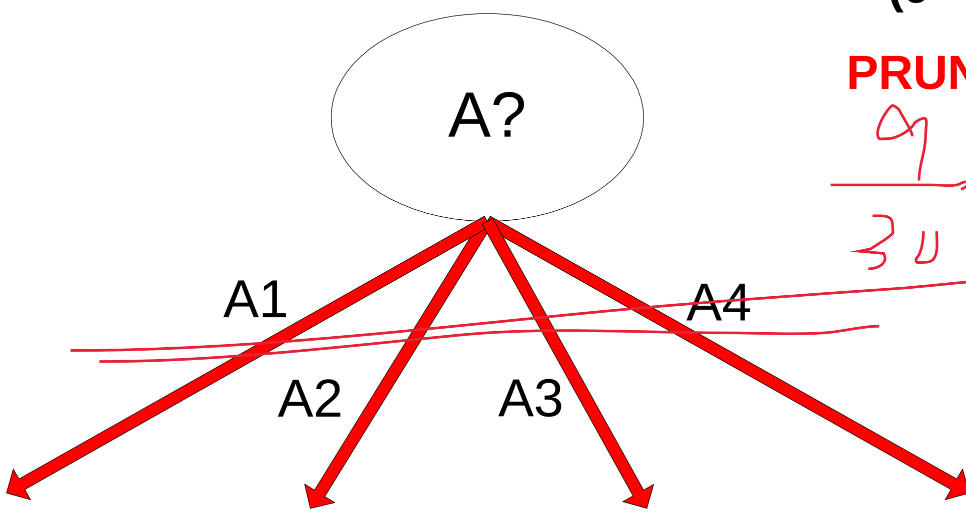
Pessimistic error (After splitting)

= (9 + 4 × 0.5)/30 = 11/30

PRUNE!

$\frac{9}{30} + 0.5 \times \frac{4}{30}$

砍!



Class = Yes	8
Class = No	4

Class = Yes	3
Class = No	4

Class = Yes	4
Class = No	1

Class = Yes	5
Class = No	1

Examples of Post-pruning

Decision Tree:

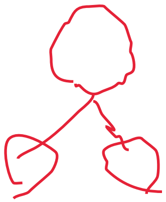
depth = 1 :
| breadth > 7 : **class 1**
| breadth <= 7 :
| | breadth <= 3 :
| | | ImagePages > 0.375 : **class 0**
| | | ImagePages <= 0.375 :
| | | | totalPages <= 6 : **class 1**
| | | | totalPages > 6 :
| | | | | breadth <= 1 : **class 1**
| | | | | breadth > 1 : **class 0**
| | width > 3 :
| | | 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

Model Evaluation

- Cross validation
  - Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Some variations:
    - Leave-one-out
    - Complete cross-validation
    - Repeated with random partitioning
    - Stratified cross-validation

3-fold cross-validation