Data Mining:

3. Klassifikation

A) Basic Concepts, Decision Trees

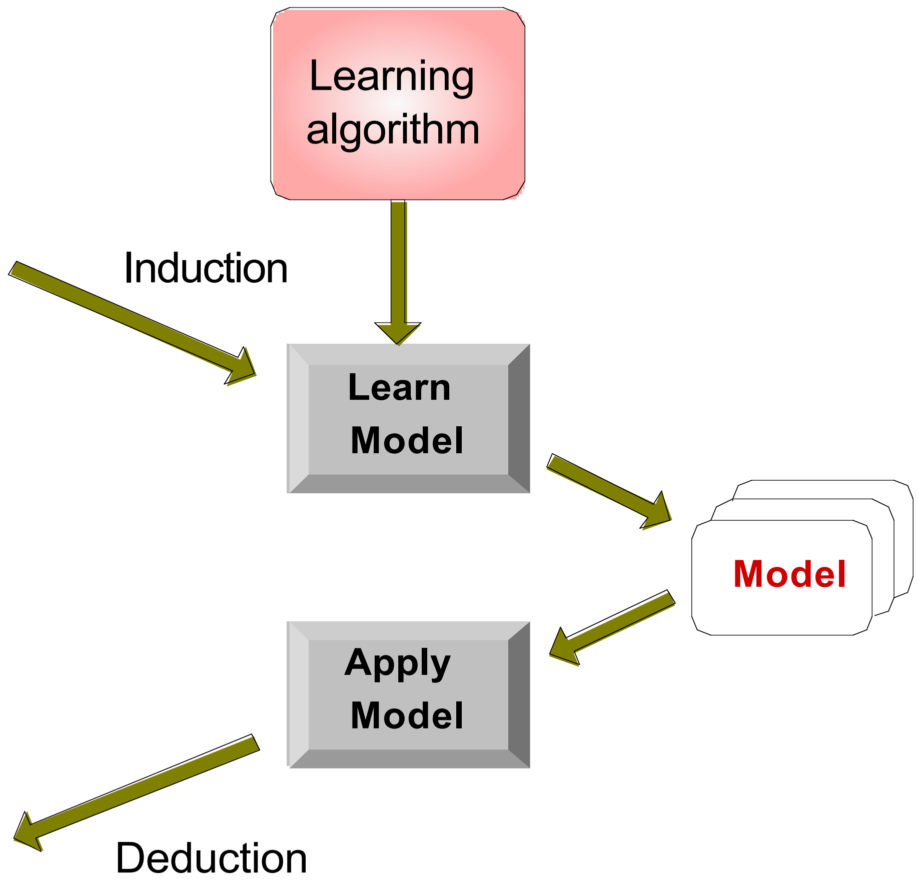
Classification: Definition

- Given a collection of records (*training set*)
  - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Goal 1: Find a *model* for the class attribute as a function of the values of other attributes.
- Goal 2: Previously unseen records should be assigned a class as accurately as possible (*prediction*).
- Validation: A *test set* is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model, and with test set used to validate it.

Illustrating Classification Task

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

Training Set



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

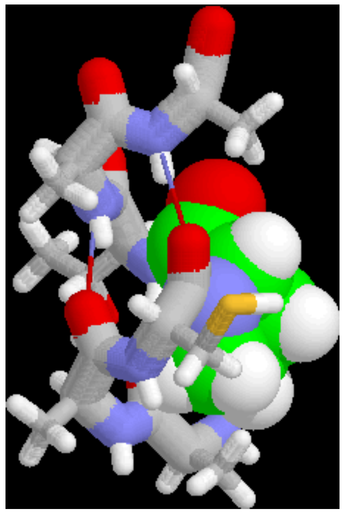
Test Set

Examples of Classification Task

- Predicting tumor cells as benign or malignant

- Classifying credit card transactions as legitimate or fraudulent

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



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

Classification Techniques

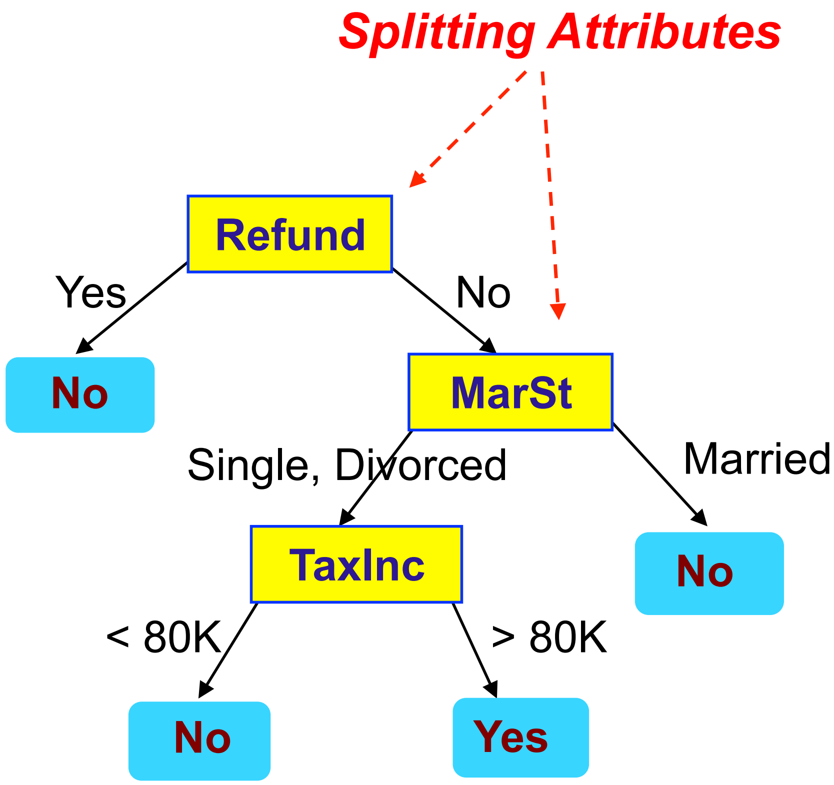
- Decision Tree based Methods
- Rule-based Methods
- Naïve Bayes and Bayesian Belief Networks
- Neural Networks
- Support Vector Machines

Example of a Decision Tree

categorical	categorical	continuous	class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

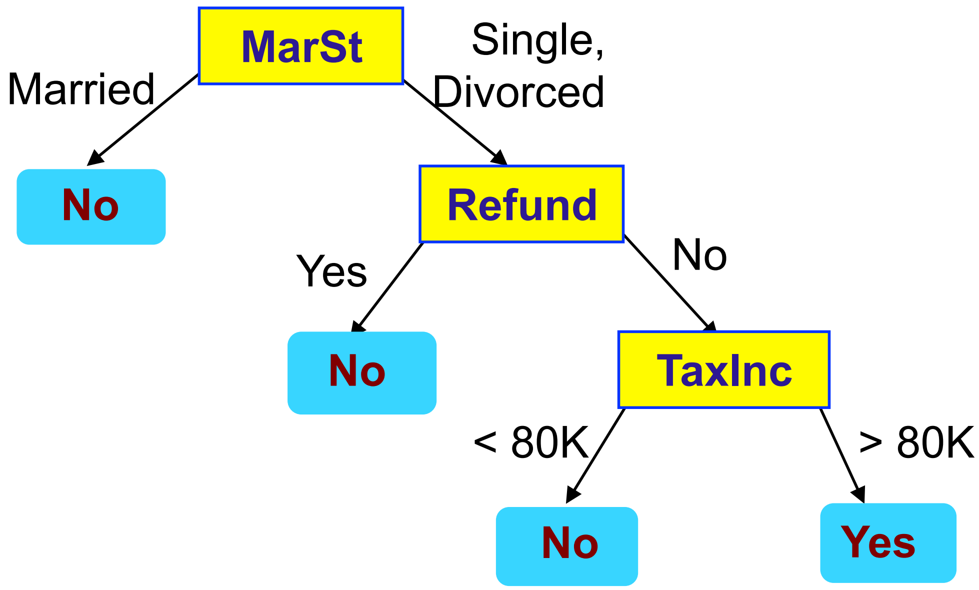
Training Data



Model: Decision Tree

Another Example of Decision Tree

	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

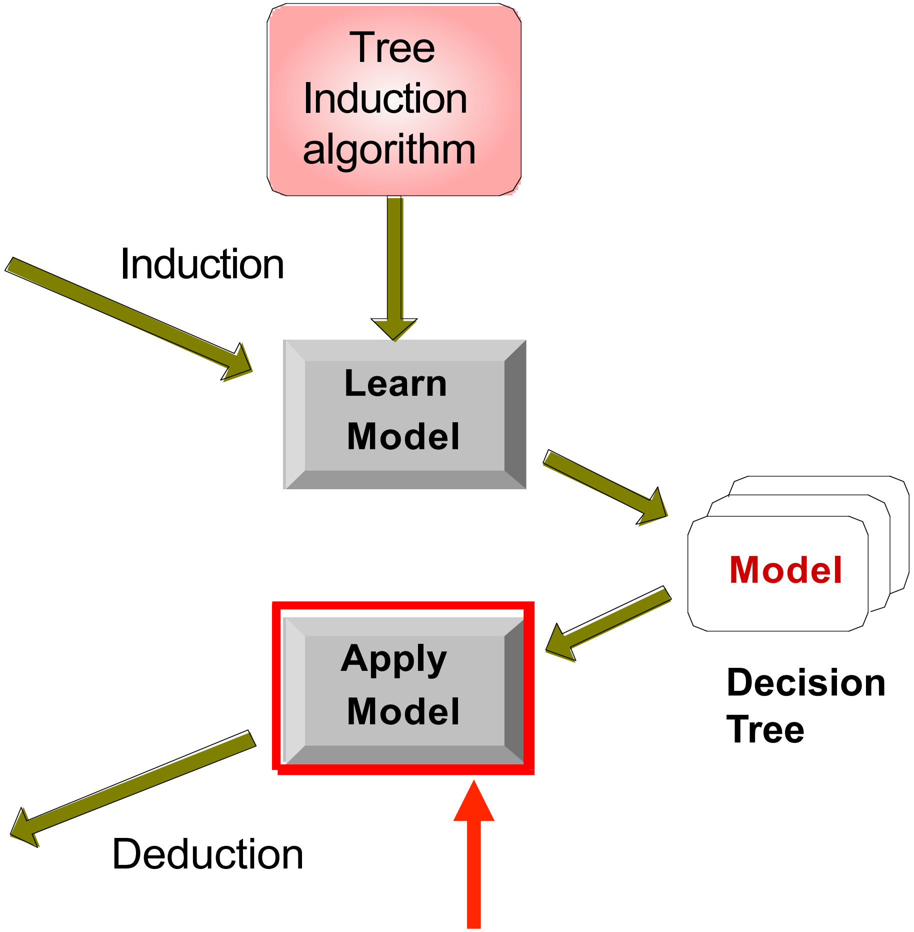


There could be more than one tree that fits the same data.

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

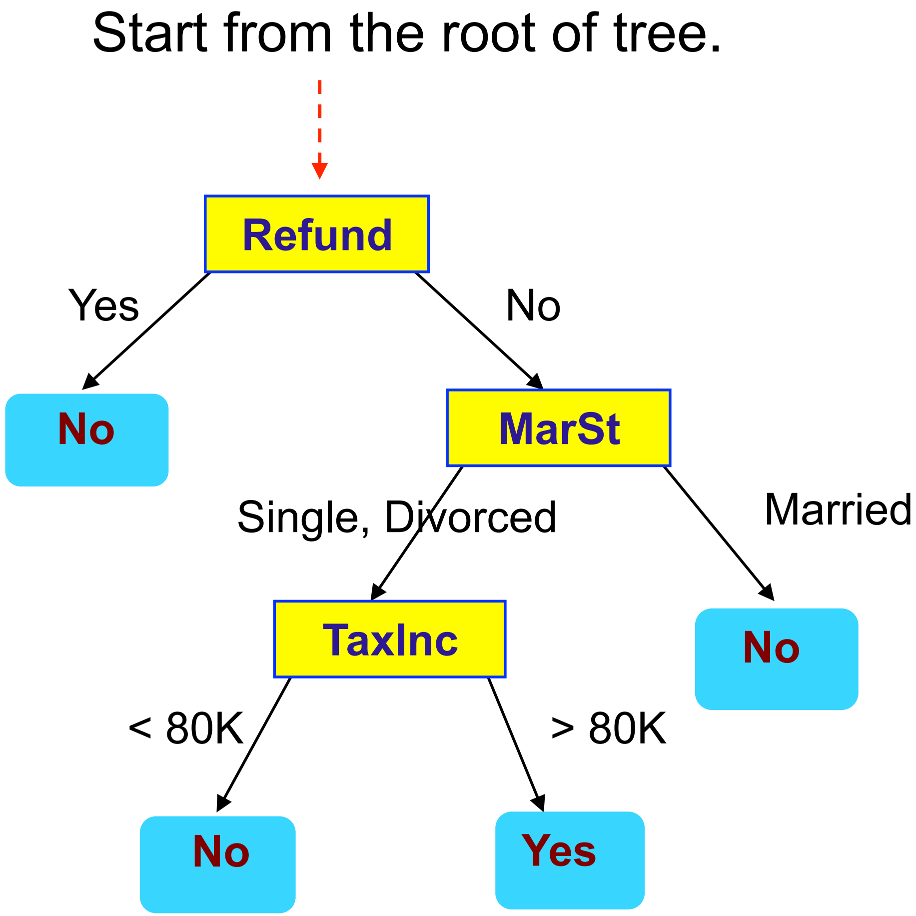
Training Set



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

Apply Model to Test Data



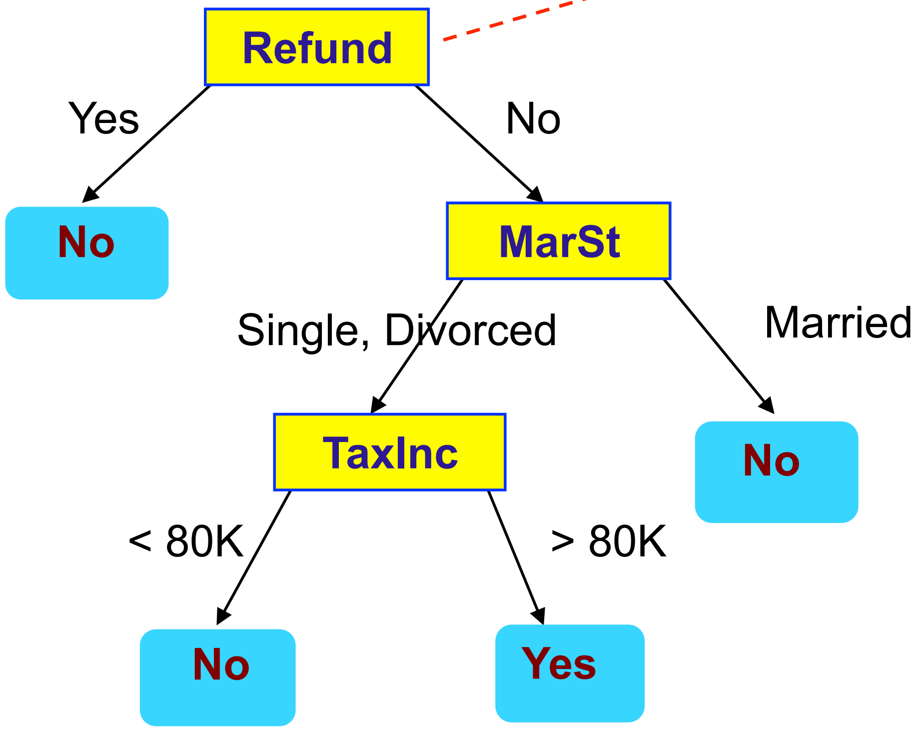
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

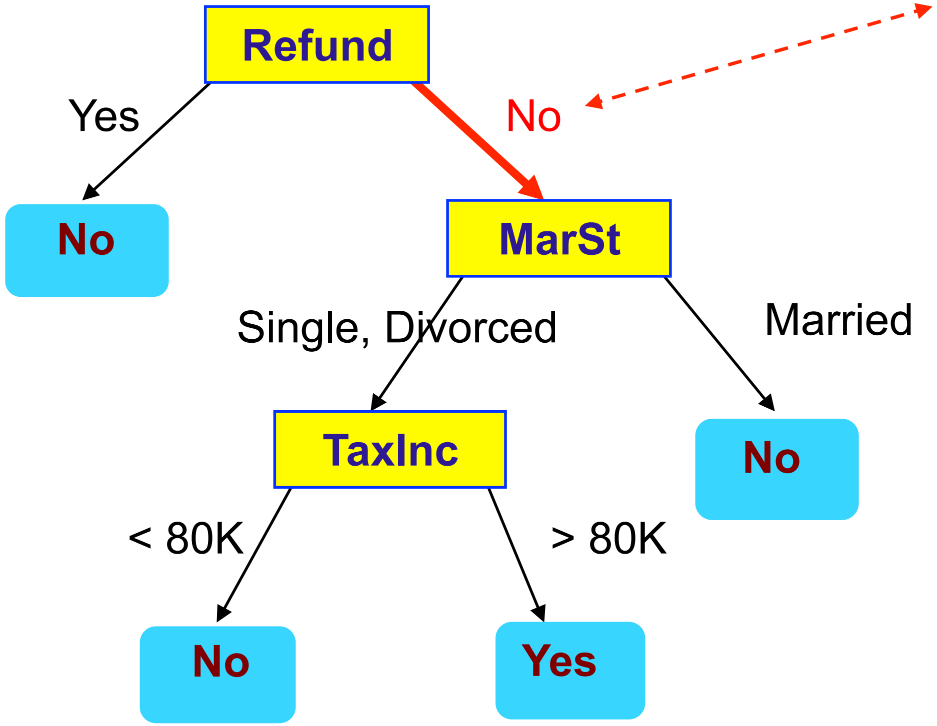
Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



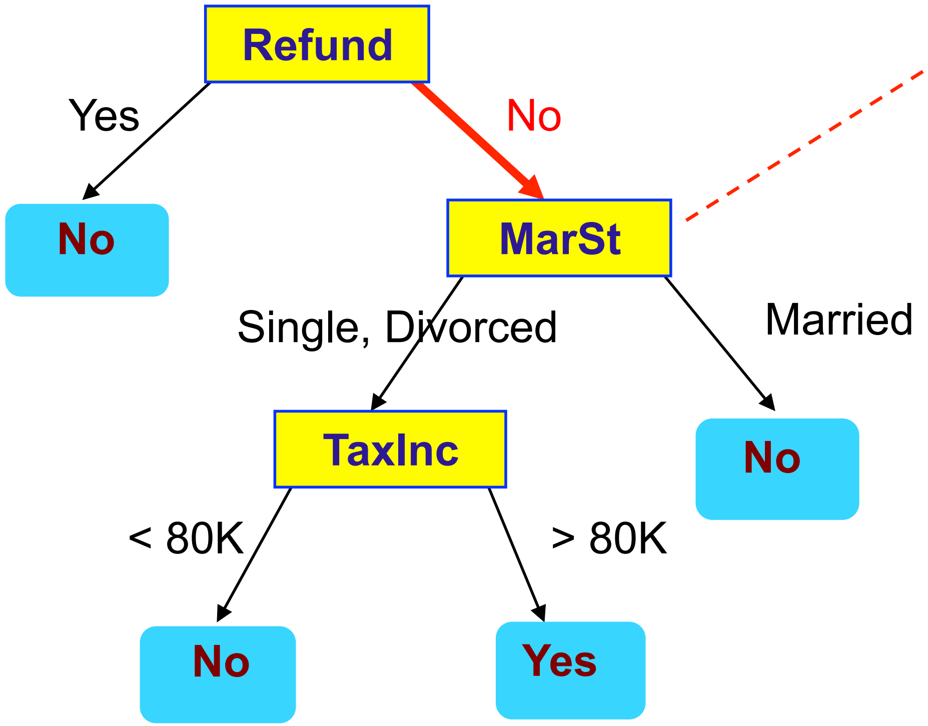
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

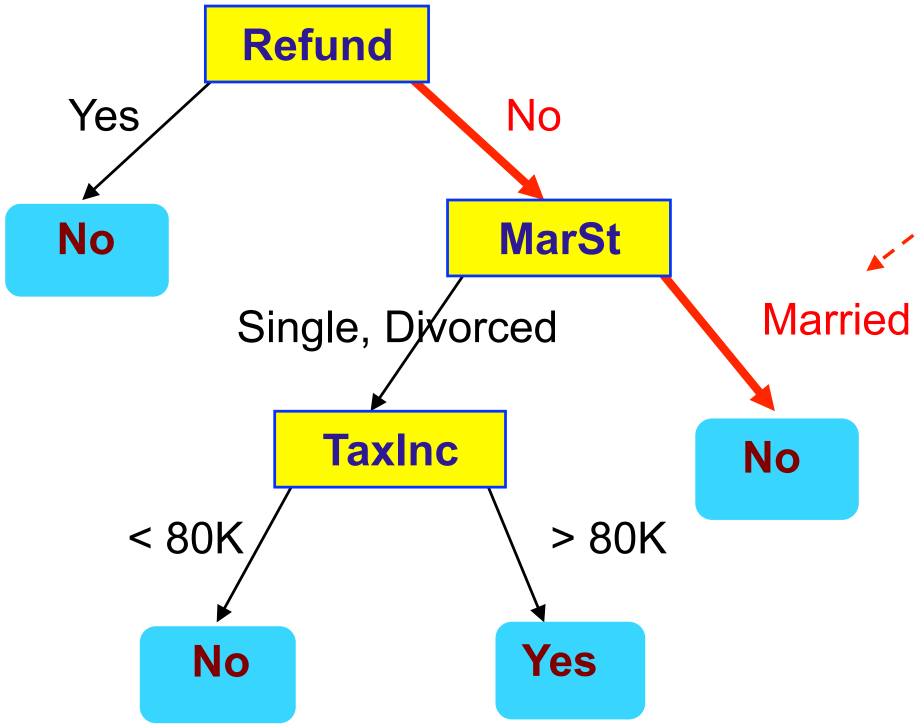
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

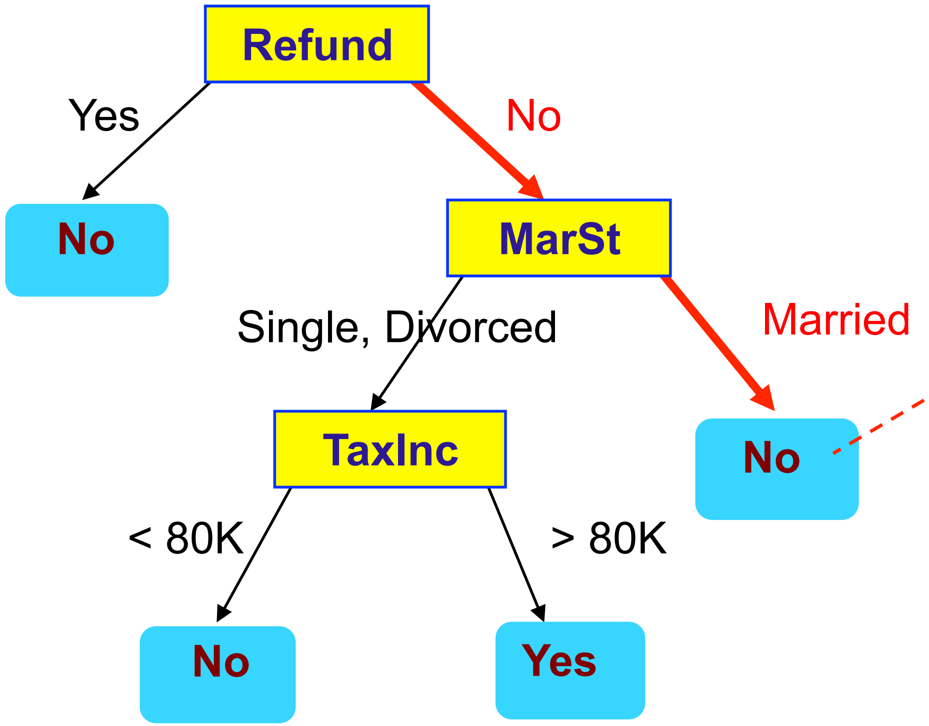
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Assign Cheat to "**No**"

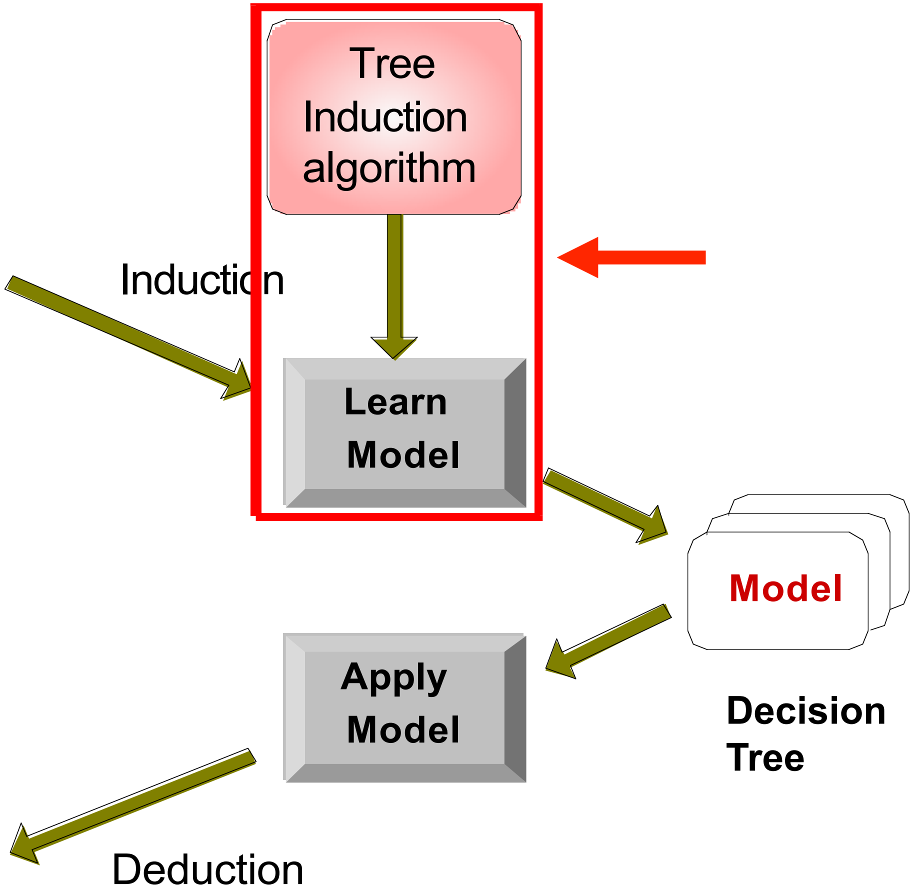
Decision Tree Classification Task

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

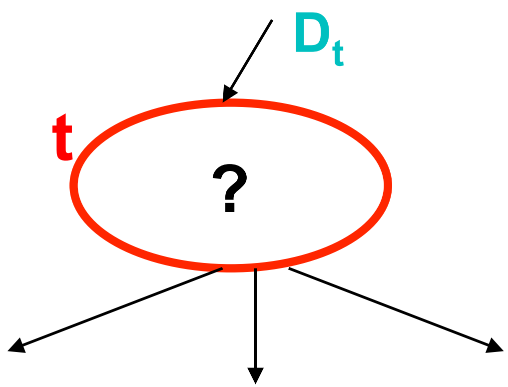
- Many Algorithms:
  - Hunt's Algorithm (one of the earliest)
  - CART
  - ID3, C4.5
  - SLIQ,SPRINT

General Structure of Hunt's Algorithm

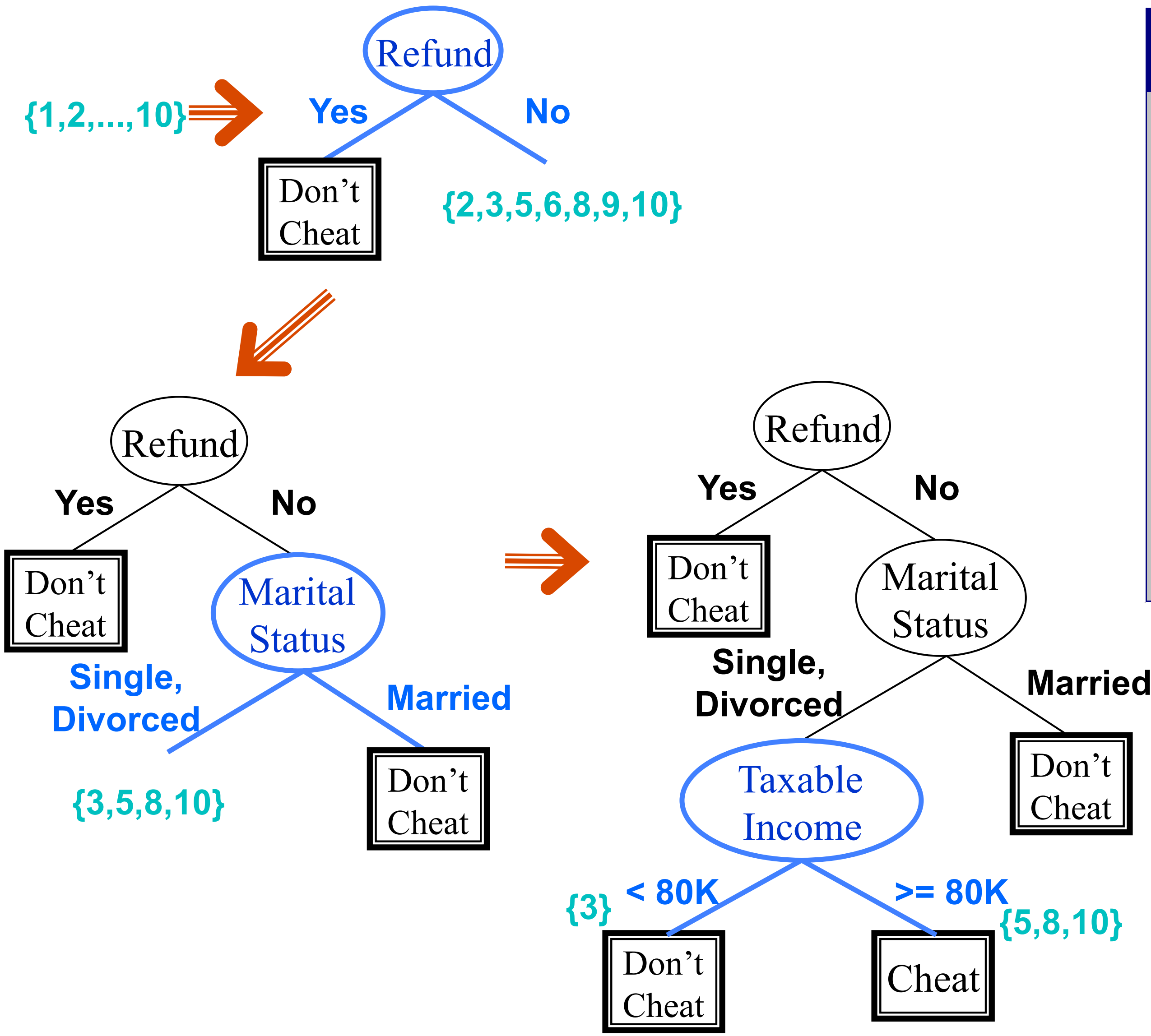
- Generate a new node t; return pointer.
- Let D_t be the set of training records that reach this node t (implicit parameter)
- General Procedure:
  - If D_t contains records that all belong to the same class y_t, then t is a leaf node labeled as y_t
  - If D_t is an empty set, then t is a leaf node labeled by the default class, y_d
  - If D_t contains records that belong to more than one class, *find an attribute test to split* the data into smaller subsets.

    Recursively apply the procedure to each subset to construct subtrees.

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Hunt's Algorithm



Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Tree Induction

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

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

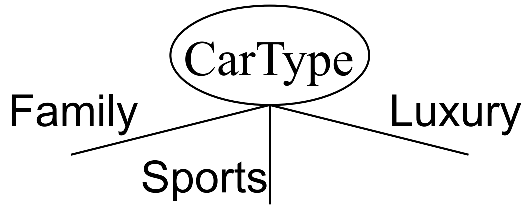
How to Specify the Test Condition?

- Depends on attribute types
  - Categorial (“nominal”)
  - Categorial and ordered (“ordinal”)
  - Continuous

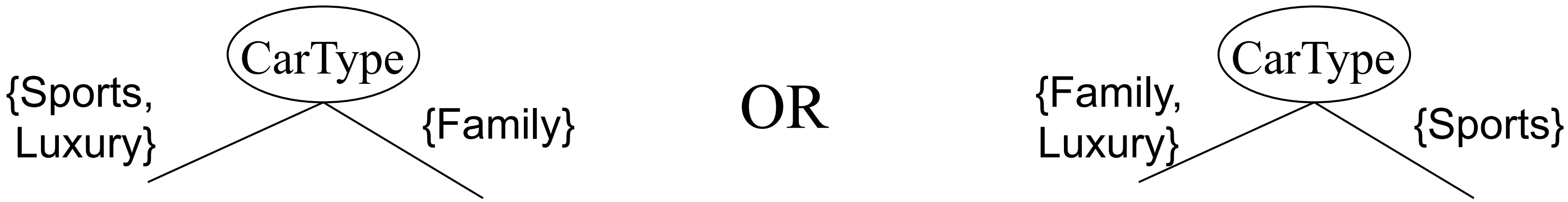
- Depends on number of ways to split
  - Binary (2-way) split
  - Multi-way split

Splitting Based on Nominal Attributes

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

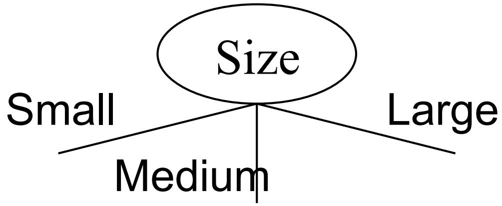


- Binary split: Divides values into two subsets. Need to find optimal partitioning.

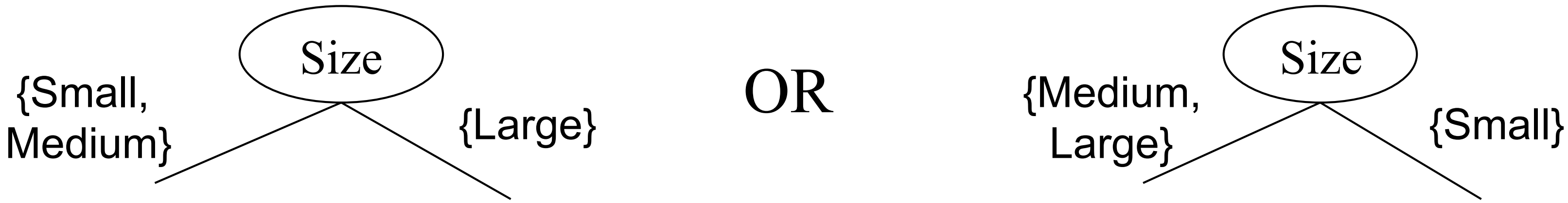


Splitting Based on Ordinal Attributes

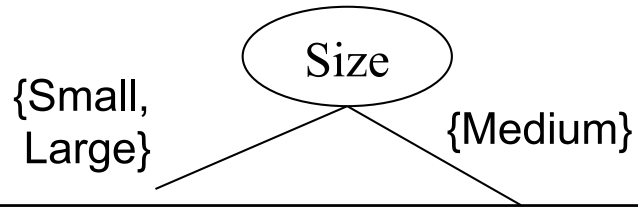
- Multi-way split: Use as many partitions as distinct values.



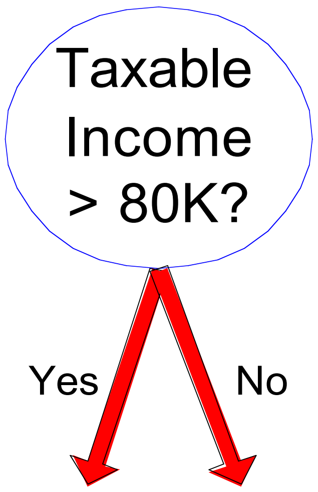
- Binary split: Divide values into two subsets. Need to find optimal partitioning.



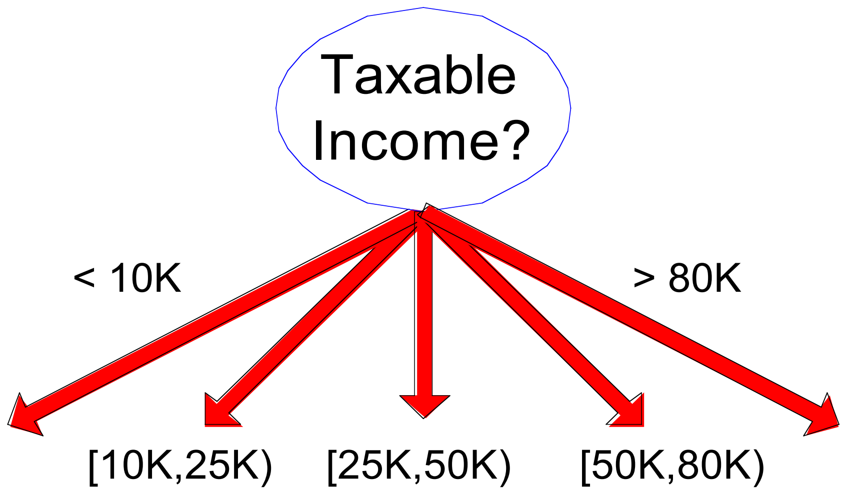
- What about this split?



Splitting Based on Continuous Attributes



(i) Binary split



(ii) Multi-way split

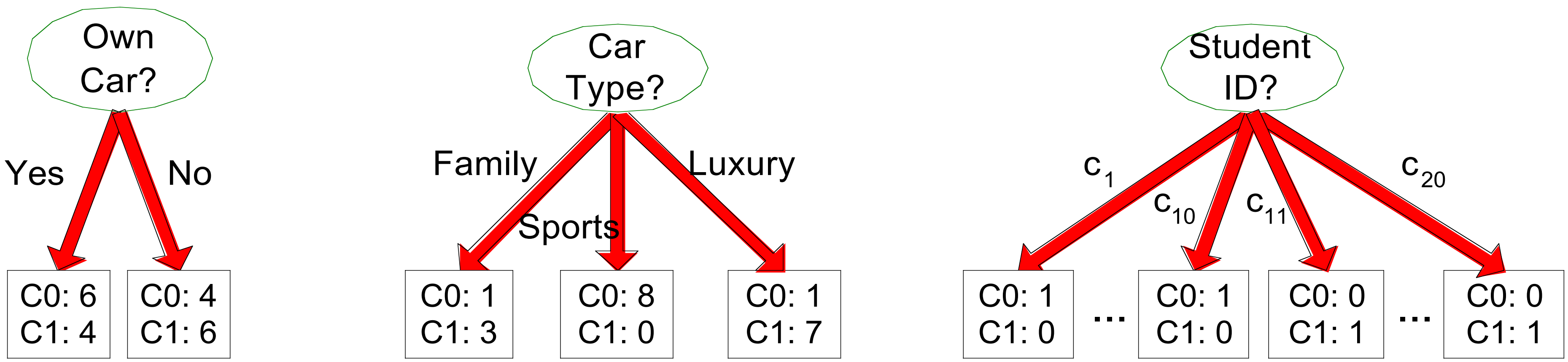
Splitting Based on Continuous Attributes

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

How to Determine the Best Split

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

After Splitting:



Which split (choice of attribute and choice of test condition) is the best?

How to Determine the Best Split

- Greedy approach:
  – Nodes with homogeneous class distribution are preferred

- Needs a measure of node impurity:

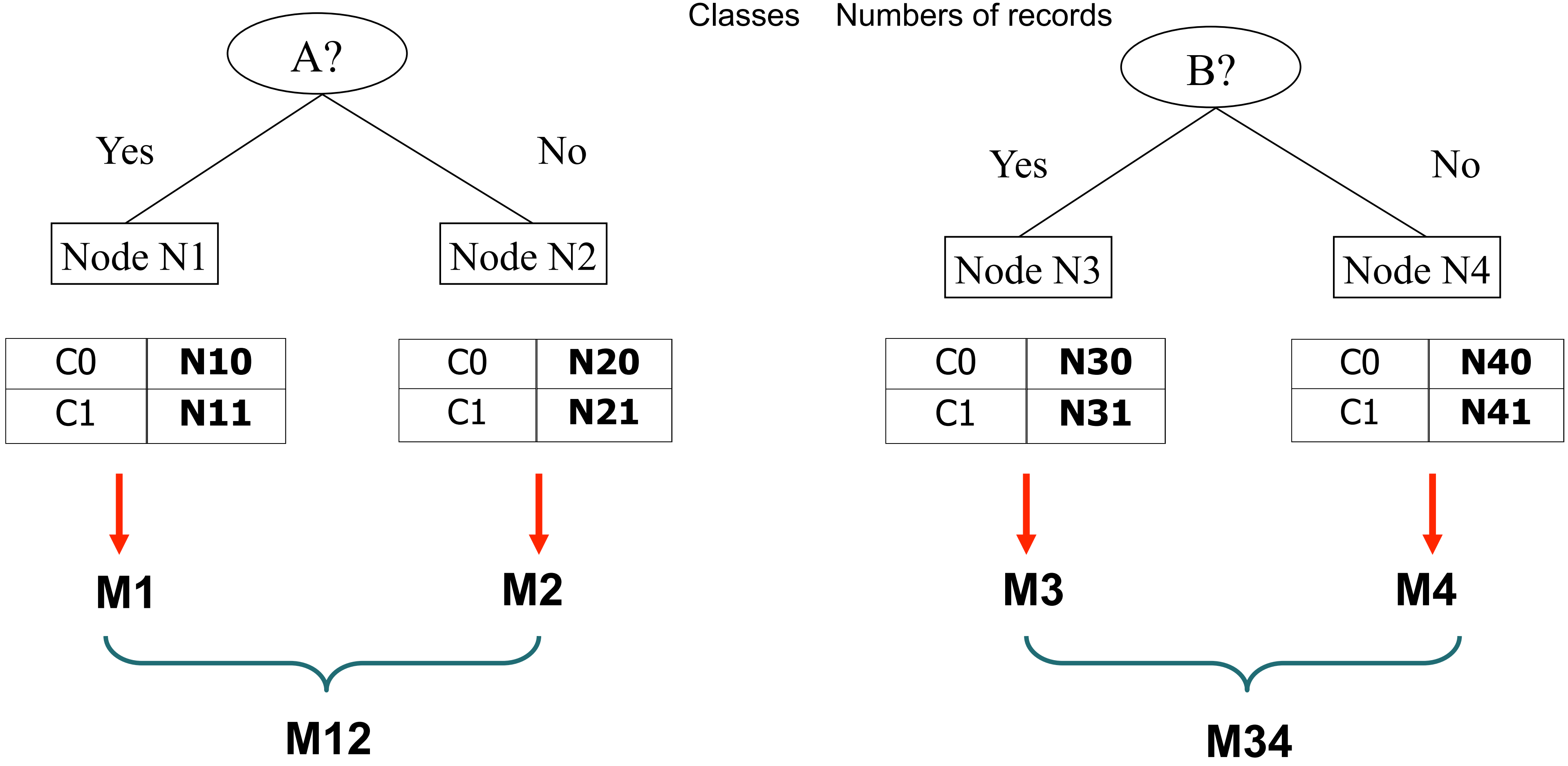
C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous **High degree of impurity**	**More homogeneous** **Low degree of impurity** ***preferred***

How to Determine the Best Split

Before Splitting:

Classes	Numbers of records
C0	**N00**
C1	**N01**

→ **M0** Measure



Maximize **Gain** = **M0 – M12** vs **M0 – M34** ! I.e. Minimize child measures **M12** vs **M34**.

Measures of Node Impurity

- Gini Index
- Entropy
- Misclassification Error

Measure of Impurity: GINI

[Corrado Gini: ital. Statistiker]

- 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, $j=1 \ldots n_c$)

– Maximum ($1 - 1/n_c$) when records are equally distributed among all classes, implying impurest information (n_c=number of classes)

– Minimum (0.0) when all records belong to one class, implying purest information

- Example:

C1	**0**
C2	**6**
Gini=0.000	

C1	**1**
C2	**5**
Gini=0.278	

C1	**2**
C2	**4**
Gini=0.444	

C1	**3**
C2	**3**
Gini=0.500	

Examples for Computing GINI

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

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

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

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

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

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

Gini = 1 – (1/6)^2 – (5/6)^2 = 0.278

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

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

Gini = 1 – (2/6)^2 – (4/6)^2 = 0.444

Splitting Based on GINI

- Used in algorithms CART, SLIQ, SPRINT.
- When a parent node is split into *k* partitions (children), the measure of this split is computed as the weighted average

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

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

- Since we want to maximize the difference

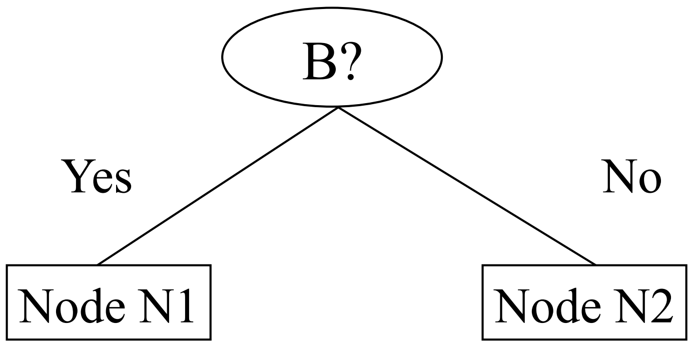
$GINI$(parent node) - $GINI_{split}$,

we have to find a split with minimal $GINI_{split}$ value.

Binary Attributes: Computing GINI Index

- Splits into two partitions
- Effect of weighing partitions:
  - Larger and purer partitions are sought for.

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



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

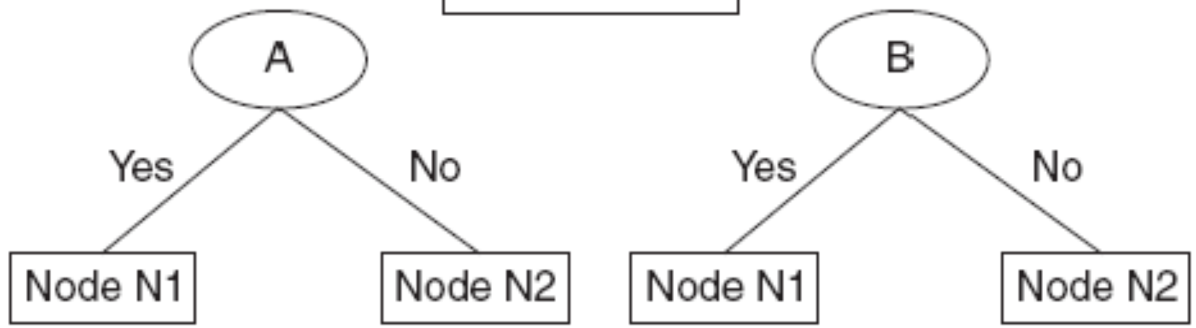
Gini(N2)
= 1 – $(1/5)^2$ – $(4/5)^2$
= 0.32

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

$Gini_{split}$(Children)
= 7/12 * 0.408 +
5/12 * 0.32
= 0.371

Binary Attributes: Computing GINI Index

	Parent
C0	6
C1	6
Gini = 0.500	



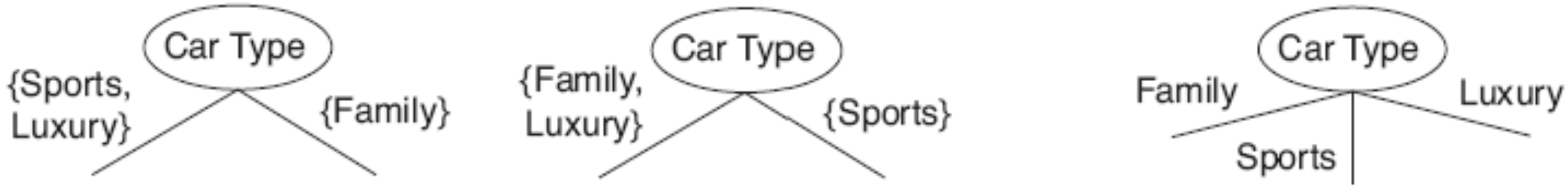
	N1	N2
C0	4	2
C1	3	3
$Gini_{split} = 0.486$		

	N1	N2
C0	1	5
C1	4	2
$Gini_{split} = 0.371$		

← Prefer this split

Figure 4.14. Splitting binary attributes.

Nominal Attributes: Computing GINI Index



	Car Type	
	{Sports, Luxury}	{Family}
C0	9	1
C1	7	3
$Gini_{split}$	0.468	

	Car Type	
	{Sports}	{Family, Luxury}
C0	8	2
C1	0	10
$Gini_{split}$	0.167	

(a) Binary split

	Car Type		
	Family	Sports	Luxury
C0	1	8	1
C1	3	0	7
$Gini_{split}$	0.163		

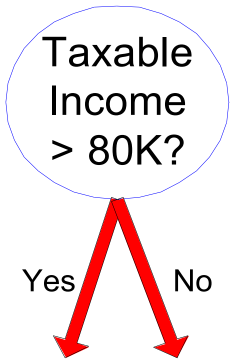
(b) Multiway split

Figure 4.15. Splitting nominal attributes.

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(N)+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! $O(N^2)$ Repetition of work.

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**



Continuous Attributes: Computing GINI Index

- For efficient computation: for each attribute,
  – Sort the attribute on values: O(N log N)
  – Linearly scan these values, each time updating the count matrix and computing GINI index: O(N)
  – Choose the split position that has the least GINI index: within latter step

Class	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
$Gini_{split}$	0.420		0.400		0.375		0.343		0.417		0.400		*0.300*		0.343		0.375		0.400		0.420	

Figure 4.16. Splitting continuous attributes.

Alternative Measure

- Entropy at a given node t:

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

(Note: $p(j\,|\,t)$ is the relative frequency of class j at node t; 0log0:=0)

– Measures information content of a node.
  - Maximum ($\log n_c$) when records are equally distributed among all classes implying least information
  - Minimum (0.0) when all records belong to one class, implying most information

– Entropy computations are similar to Gini-index computations

Examples for Computing Entropy

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

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

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

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

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

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

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

Splitting Based on Entropy

- Again, the gain (measure at parent - avg measure of children) of a split has to be maximized (here called Information Gain):

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

$$(\ldots) = Entropy_{split}$$

- Used in algorithms ID3 and C4.5
- Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.
- Avoiding this disadvantage: Use binary splits only
  or use Gain Ratio instead of Gain …

Splitting, Adjusted

- 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!
 – E.g. k partitions of same size $1/k$: SplitINFO = $\log_2 k$
 – Used in C4.5
 – Designed to overcome the disadvantage of Inf.Gain

Yet another measure

- Misclassification error at a node *t* (with classes *j*):

$$\boxed{Error(t) = 1 - \max_{j} p(j \mid t)}$$

  – Measures misclassification error made by a node.
    - Maximum ($1 - 1/n_c$) when records are equally distributed among all classes, implying maximally unclear classification
    - Minimum (0.0) when all records belong to one class, implying no misclassification

  – Simplest measure, but least differentiating.

Examples for Computing Error

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

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

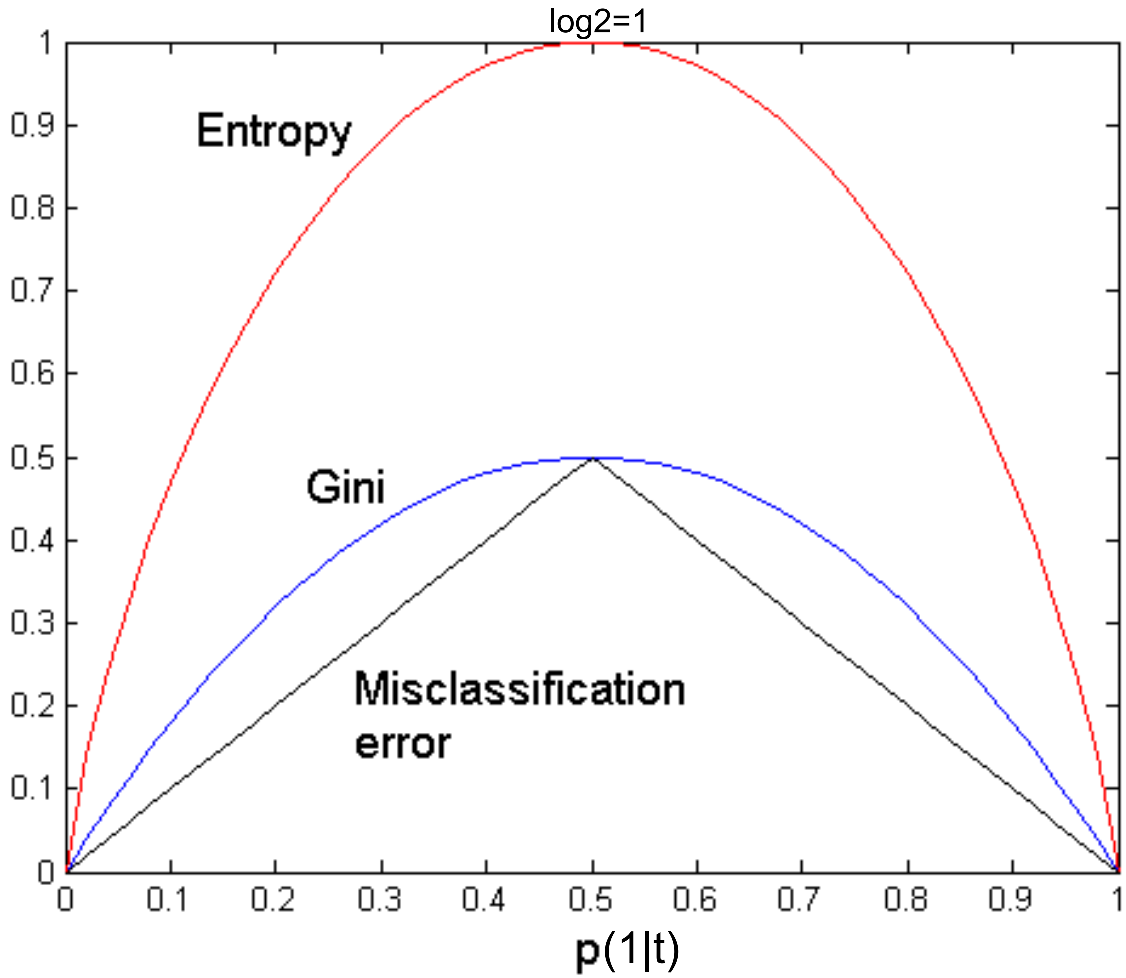
C1	**2**
C2	**4**

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

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

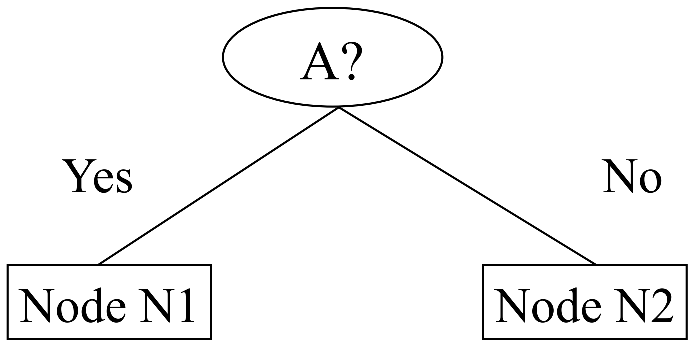
Comparison among Impurity Measures

For a binary classification problem:



(values of one node, not of a split)

Misclassification Error vs Gini



	Parent
C1	**7**
C2	**3**
Gini = 0.42, Error = 0.3	

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

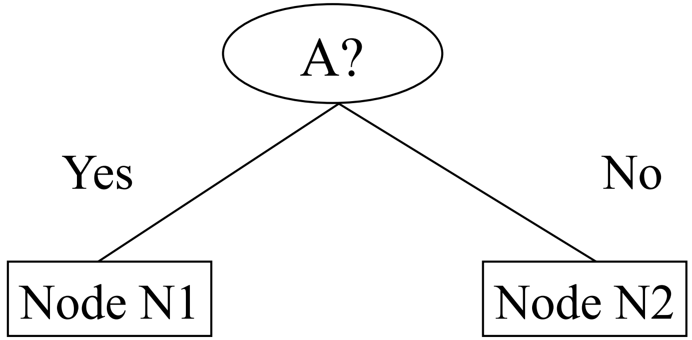
Gini(N2)
= 1 – $(4/7)^2$ – $(3/7)^2$
= 0.489

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

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

Gini improves !!

Example: Misclassification Error vs Gini



	Parent
C1	**7**
C2	**3**
Gini = 0.42, Error = 0.3	

Error(N1)
= 1 – max(3/3,0/3)
= 0

Error(N2)
= 1 – max(4/7,3/7)
= 3/7

	N1	N2
C1	**3**	**4**
C2	**0**	**3**
Gini=0.342, Error=0.3		

Error(Children)
= 3/10 * 0
+ 7/10 * 3/7
= 0.3

Gini improves,
Error does not !!

Tree Induction

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

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

Stopping Criteria for Tree Induction

- Stop expanding a node when all the records belong to the same class
- Stop expanding a node when all represented records have similar attribute values
- Early termination (using default or majority class) to avoid overfitting the model [to be discussed later]

Tree Induction Algorithm

Algorithm A skeleton decision tree induction algorithm.

```
TreeGrowth (E, F)
if stopping_cond(E,F) = true then
    leaf = createNode().
    leaf.label = Classify(E).
    return leaf.
else
    root = createNode().
    root.test_cond = find_best_split(E, F).
    let V = {v|v is a possible outcome of root.test_cond }.
    for each v ∈ V do
        E_v = {e | root.test_cond(e) = v and e ∈ E}.
        child = TreeGrowth(E_v, F).
        add child as descendent of root and label the edge (root → child) as v.
    end for
end if
return root.
```

E training records, F attribute set, *label* assigned class (usually, the class *j* with maximal *p(j|t)*)

Decision Tree Based Classification

- Advantages:
  - Inexpensive to construct
    - but many splitting options may have to be calculated
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Accuracy is comparable to other classification techniques for many simple data sets

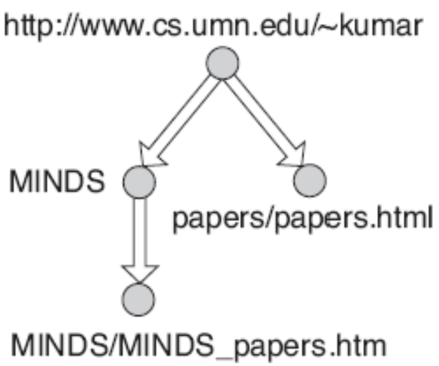
Example Algorithm: C4.5

- Simple depth-first construction.
- Uses Information Gain
- Sorts Continuous Attributes at each node.
- Needs entire data to fit in memory.
- Unsuitable for Large Datasets.
  - Needs out-of-core sorting.
- You can download the software or use it in Weka.

An Application: Web Robot Detection

Session	IP Address	Timestamp	Request Method	Requested Web Page	Protocol	Status	Number of Bytes	Referrer	User Agent
1	160.11.11.11	08/Aug/2004 10:15:21	GET	http://www.cs.umn.edu/~kumar	HTTP/1.1	200	6424		Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
1	160.11.11.11	08/Aug/2004 10:15:34	GET	http://www.cs.umn.edu/~kumar/MINDS	HTTP/1.1	200	41378	http://www.cs.umn.edu/~kumar	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
1	160.11.11.11	08/Aug/2004 10:15:41	GET	http://www.cs.umn.edu/~kumar/MINDS/MINDS_papers.htm	HTTP/1.1	200	1018516	http://www.cs.umn.edu/~kumar/MINDS	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
1	160.11.11.11	08/Aug/2004 10:16:11	GET	http://www.cs.umn.edu/~kumar/papers/papers.html	HTTP/1.1	200	7463	http://www.cs.umn.edu/~kumar	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
2	35.9.2.2	08/Aug/2004 10:16:15	GET	http://www.cs.umn.edu/~steinbac	HTTP/1.0	200	3149		Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7) Gecko/20040616

(a) Example of a Web server log.



(b) Graph of a Web session.

Attribute Name	Description
totalPages	Total number of pages retrieved in a Web session
ImagePages	Total number of image pages retrieved in a Web session
TotalTime	Total amount of time spent by Web site visitor
RepeatedAccess	The same page requested more than once in a Web session
ErrorRequest	Errors in requesting for Web pages
GET	Percentage of requests made using GET method
POST	Percentage of requests made using POST method
HEAD	Percentage of requests made using HEAD method
Breadth	Breadth of Web traversal
Depth	Depth of Web traversal
MultiIP	Session with multiple IP addresses
MultiAgent	Session with multiple user agents

(c) Derived attributes for Web robot detection.

Figure 4.17. Input data for Web robot detection.

An Application: Web Robot Detection

```
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 <= 0.322:  class 0
| | | | | RepeatedAccess > 0.322:  class 1
| MultiAgent = 1:
| | totalPages <= 81:  class 0
| | totalPages > 81:  class 1
```

class 1: web robots
class 0: human users

Figure 4.18. Decision tree model for Web robot detection.