Classification

Salvatore Orlando

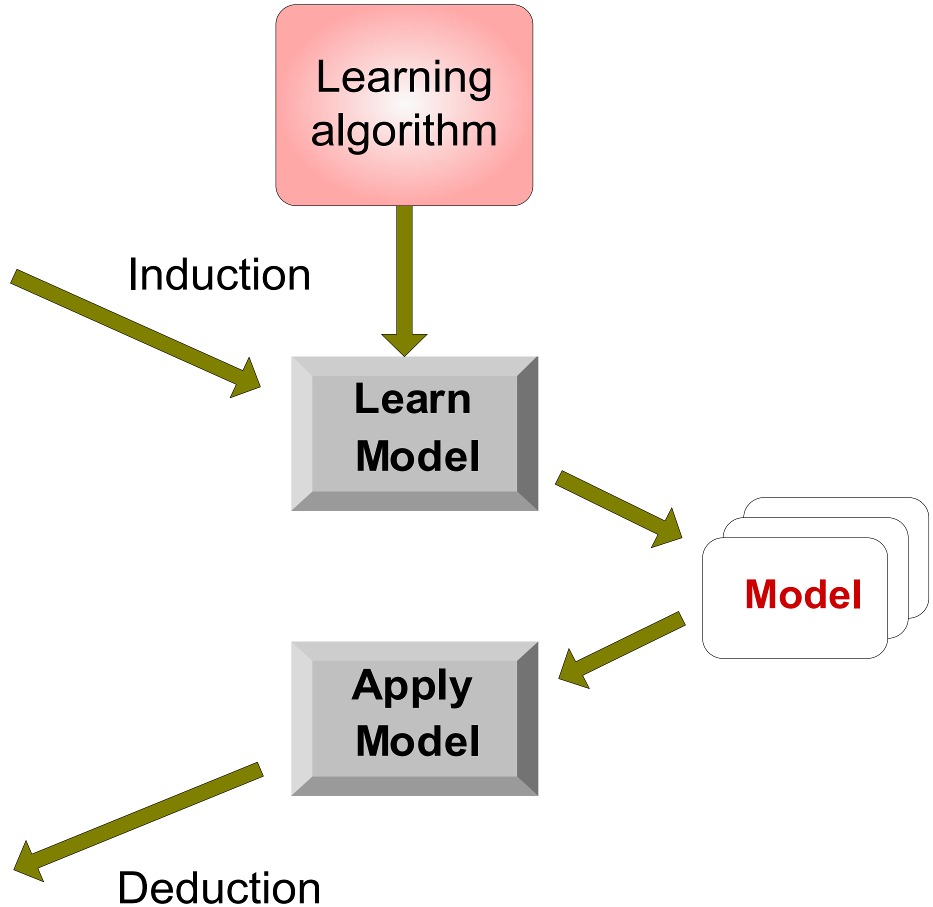
Classification: Definition

- **Given a collection of records (*training set*)**
  - **Each record contains a set of *attributes*, one of the attributes is the *class*.**
  - **The values of the class label represent the supervised knowledge**
- **Find a *model* for class attribute as a function of the values of other attributes**
  - **The function has to map a set of attributes X to a predefined class label *y***
- **Goal: previously unseen records should be assigned a class as accurately as possible.**
  - **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 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



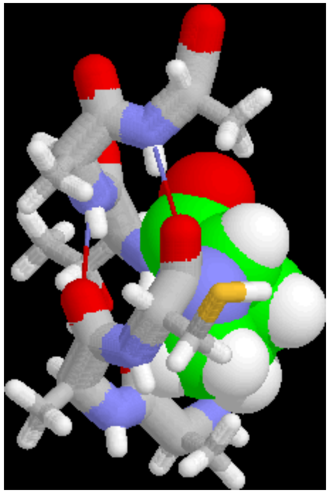
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

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

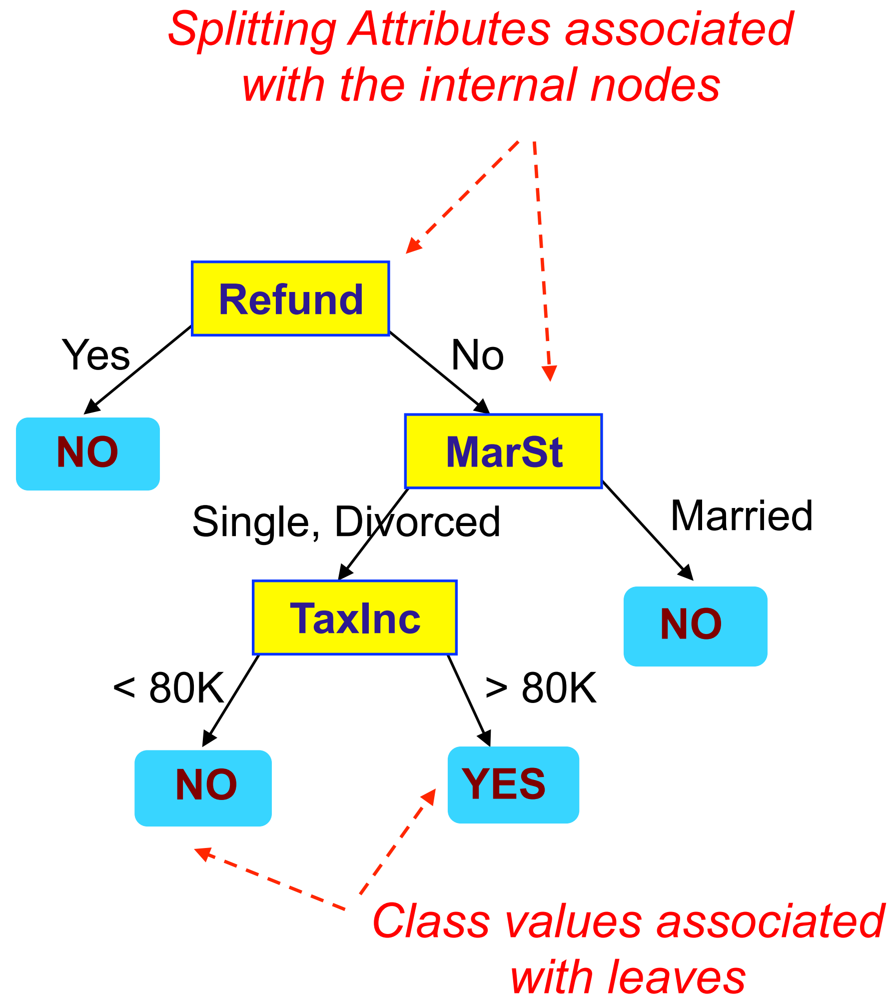
Classification Techniques

- Decision Tree based Methods
- Rule-based Methods
- Memory based reasoning
- Neural Networks
- Naïve Bayes and Bayesian Belief 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

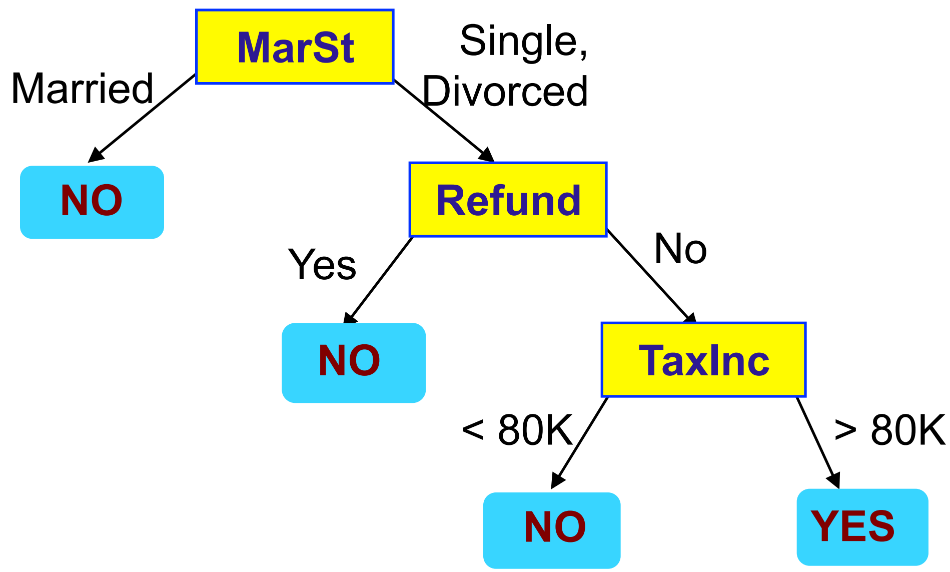


Training Data

Model: Decision Tree

Another Example of Decision Tree

	Refund	Marital Status	Taxable Income	Cheat
	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

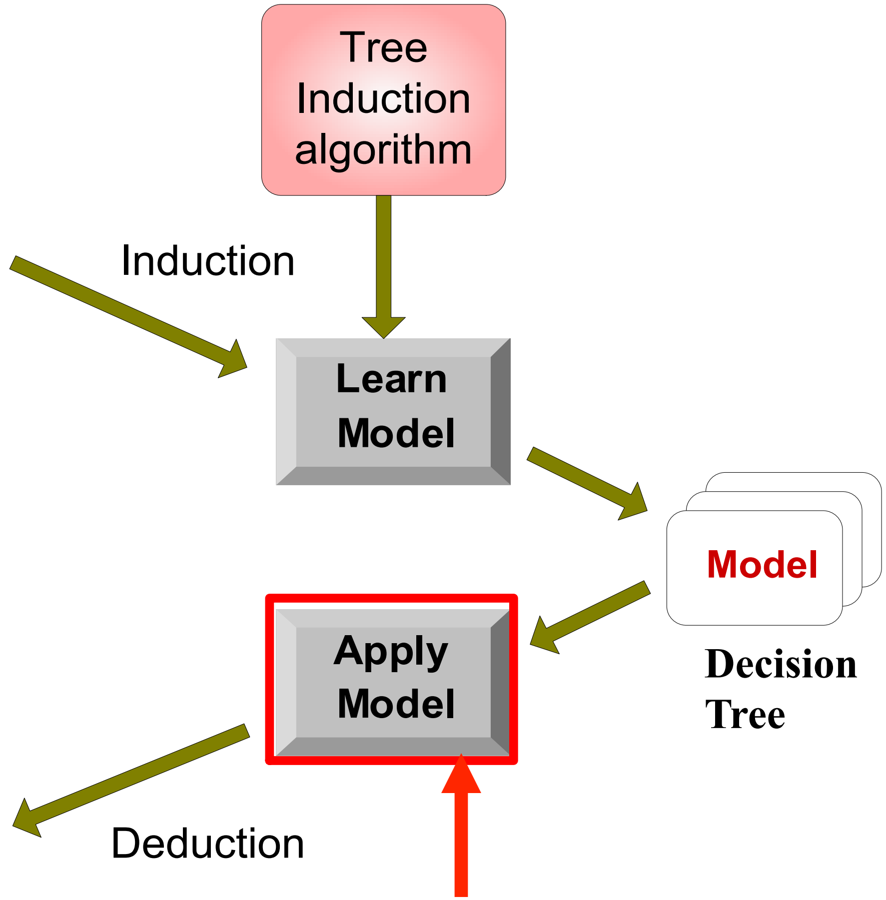


There could be more than one tree that fits the same data!

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

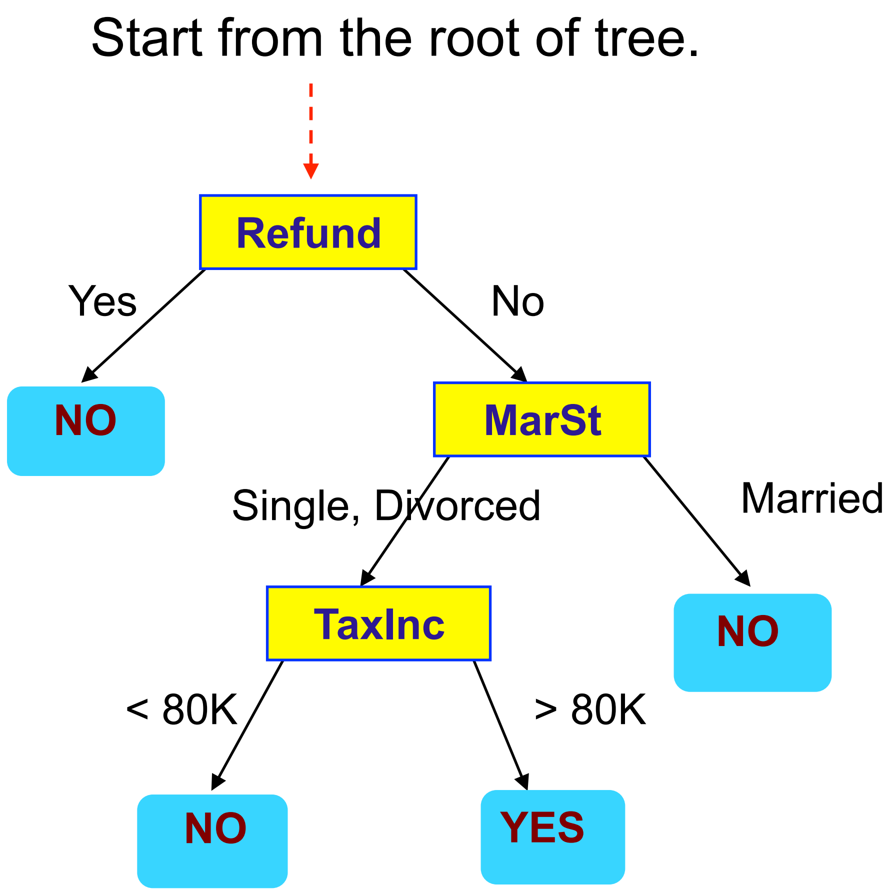
Training Set



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

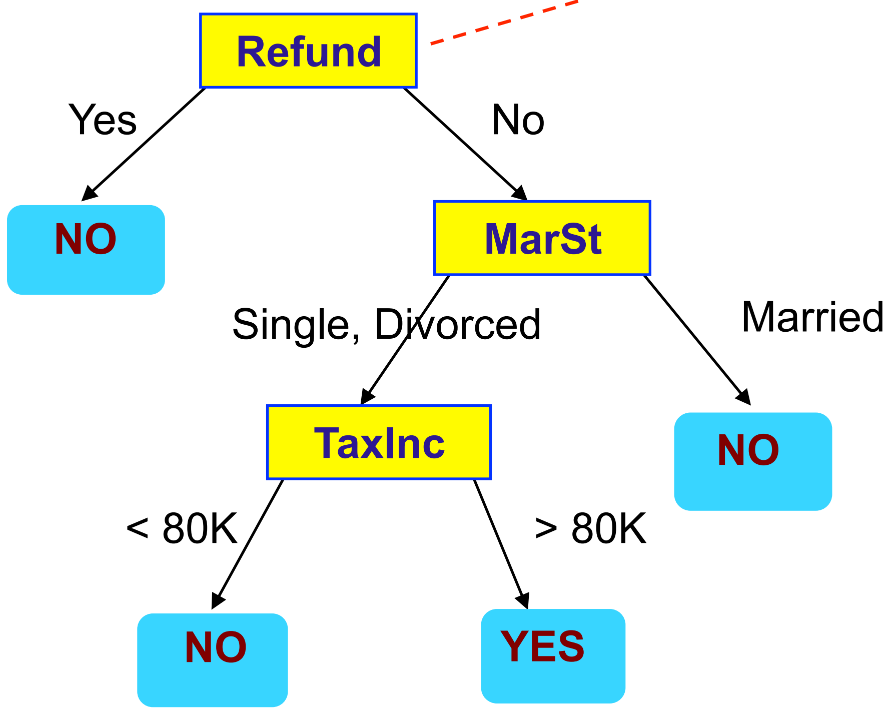
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



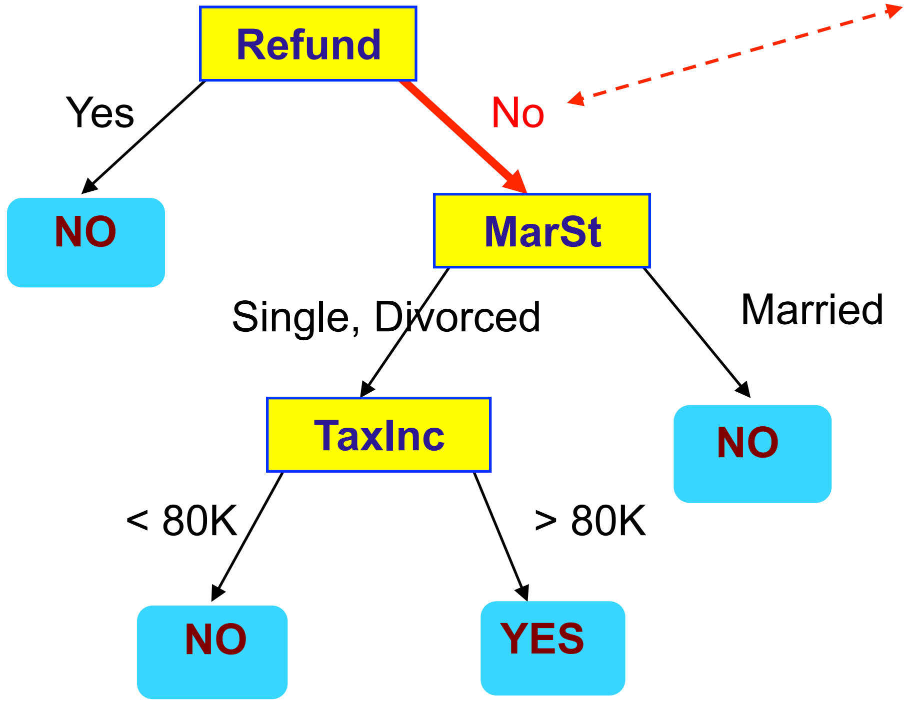
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

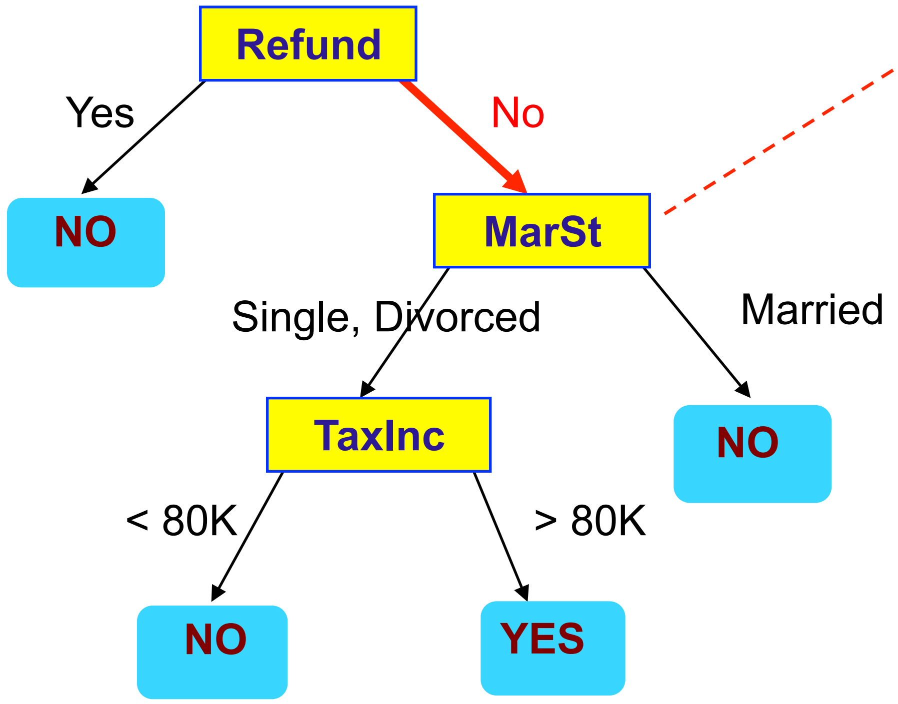
Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data



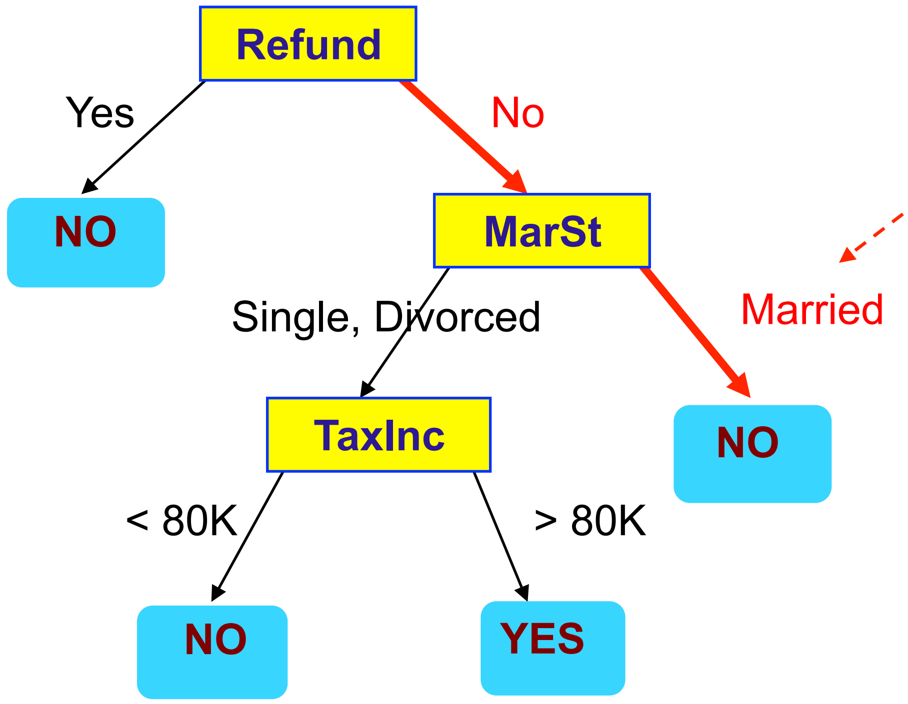
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

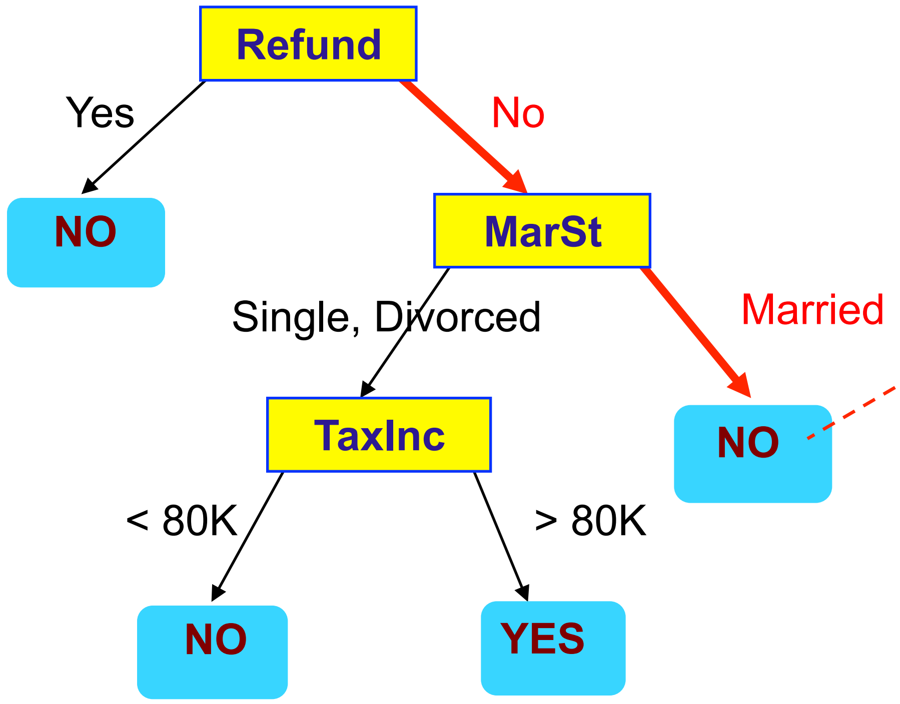
Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data



Test Data

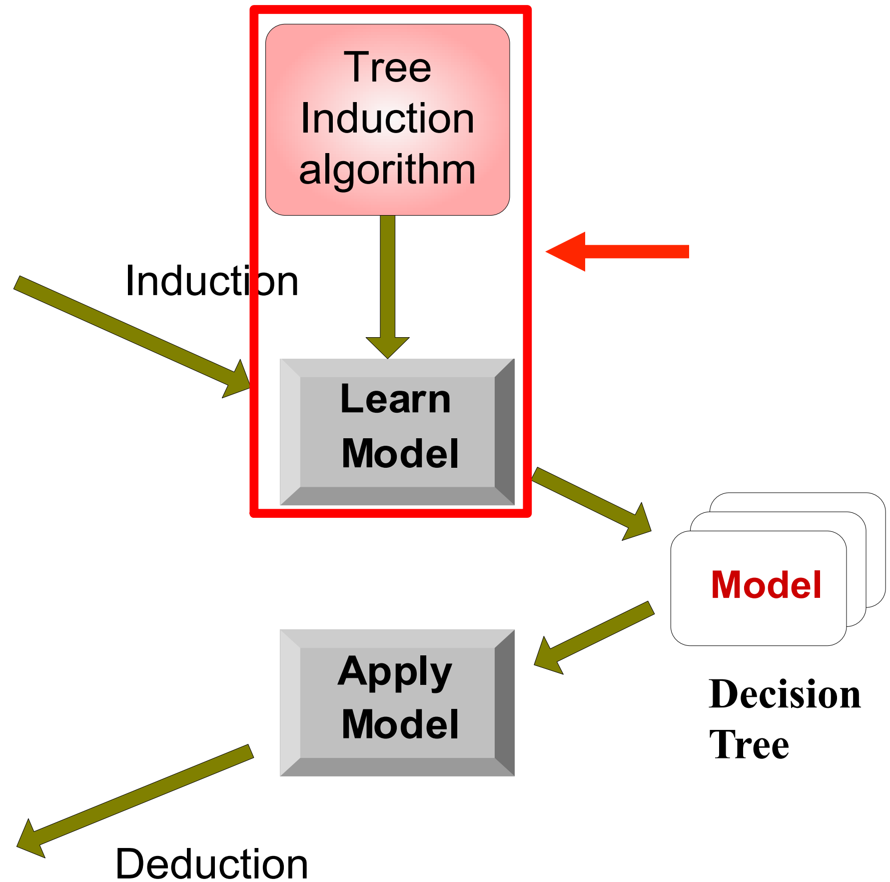
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

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

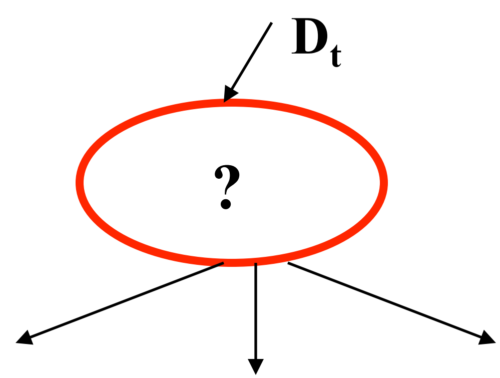
Decision Tree Induction

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

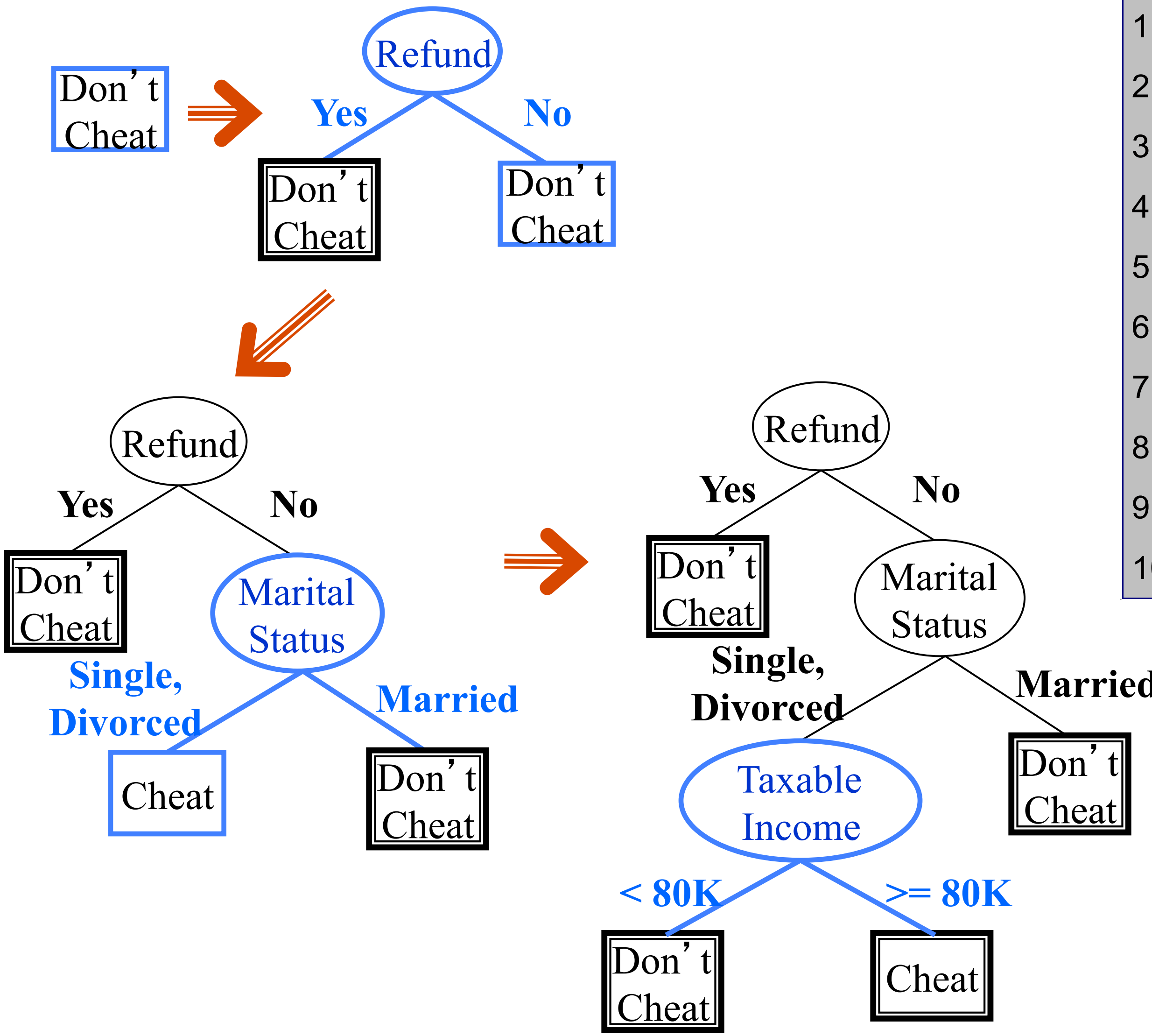
General Structure of Hunt's Algorithm

- Let D_t be the set of training records that reach a node *t*
  - At the beginning all the records are at the root
- 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 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, use an attribute test to split the data into smaller subsets. Recursively apply the procedure to each subset.

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 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**

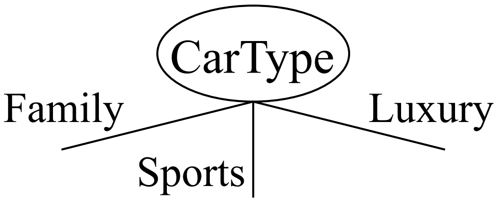
How to Specify Test Condition?

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

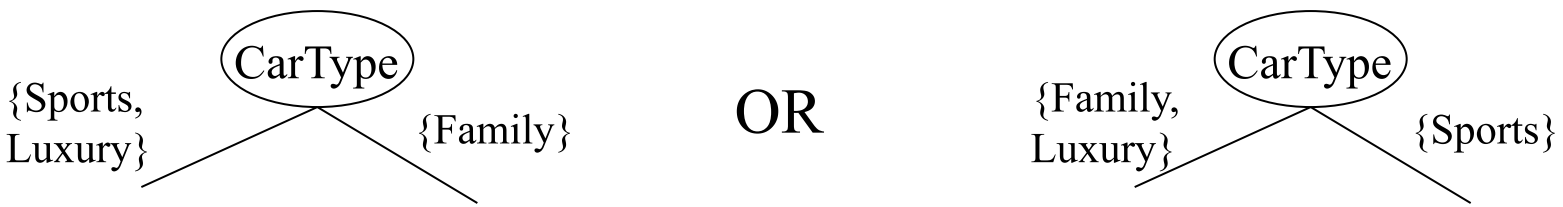
- **Depends on number of ways to split**
  - **2-way split**
  - **Multi-way split**

Splitting Based on Nominal Attributes

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

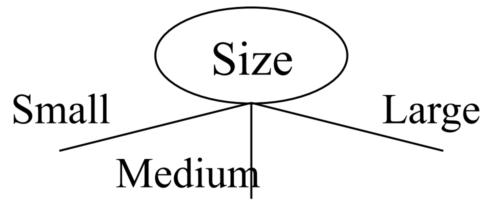


- **Binary split:** Divides values into two subsets. Need to find optimal partitioning.

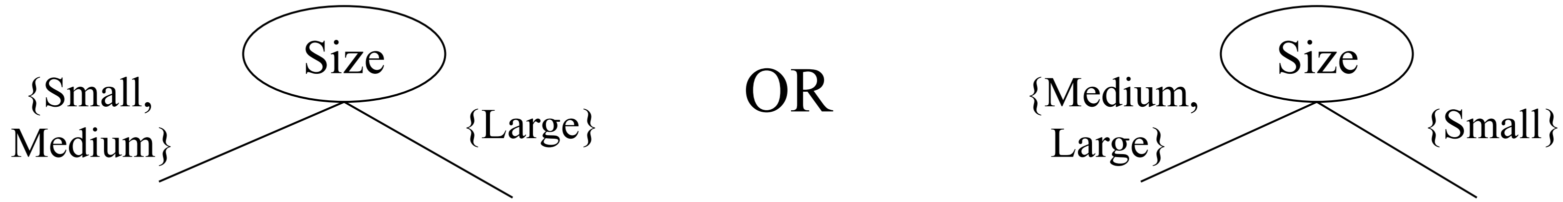


Splitting Based on Ordinal Attributes

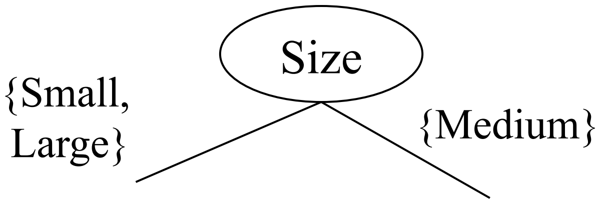
- **Multi-way split:** Use as many partitions as distinct values.



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



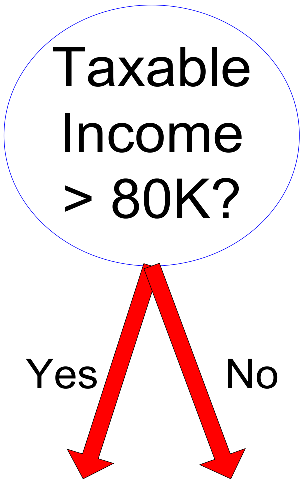
- **What about this split?**



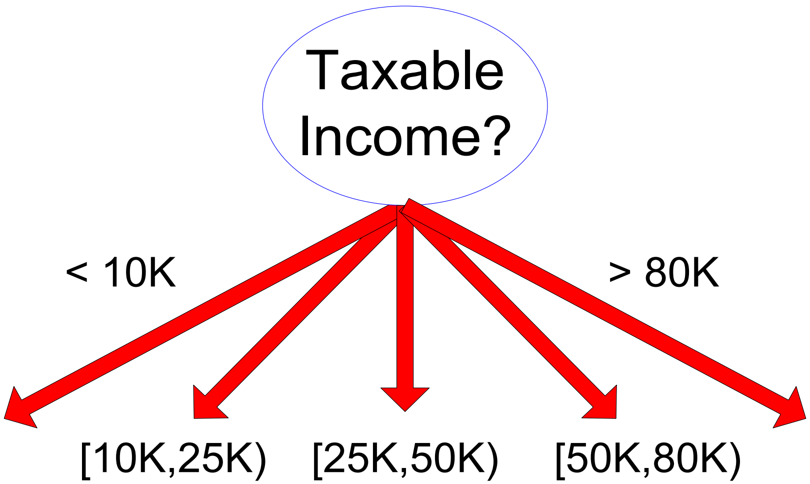
Splitting Based on Continuous Attributes

- **Different ways of handling**
  - **Discretization to form an ordinal categorical attribute**
    - **Static – discretize once at the beginning**
    - **Dynamic – ranges depends on the data, and can be found dynamically by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.**
  - **Binary Decision: (A < v) or (A ≥ v)**
    - **consider all possible splits and finds the best cut**
    - **can be more compute intensive**

Splitting Based on Continuous Attributes



(i) Binary split



(ii) Multi-way split

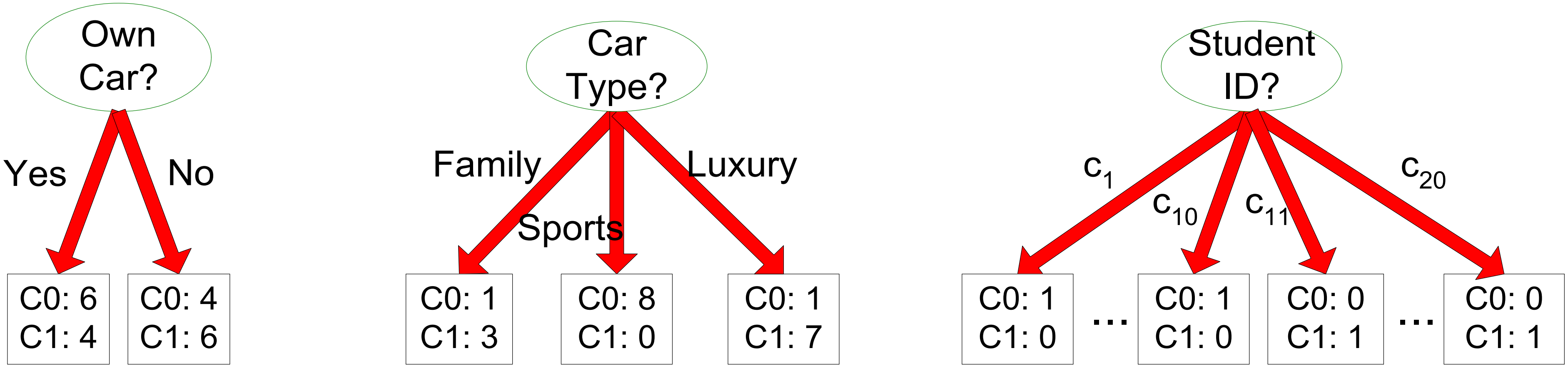
Tree Induction

- **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

How to determine the Best Split

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 homogeneous class distribution are preferred**
- **Need a measure of node impurity:**

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

Measures of Node Impurity

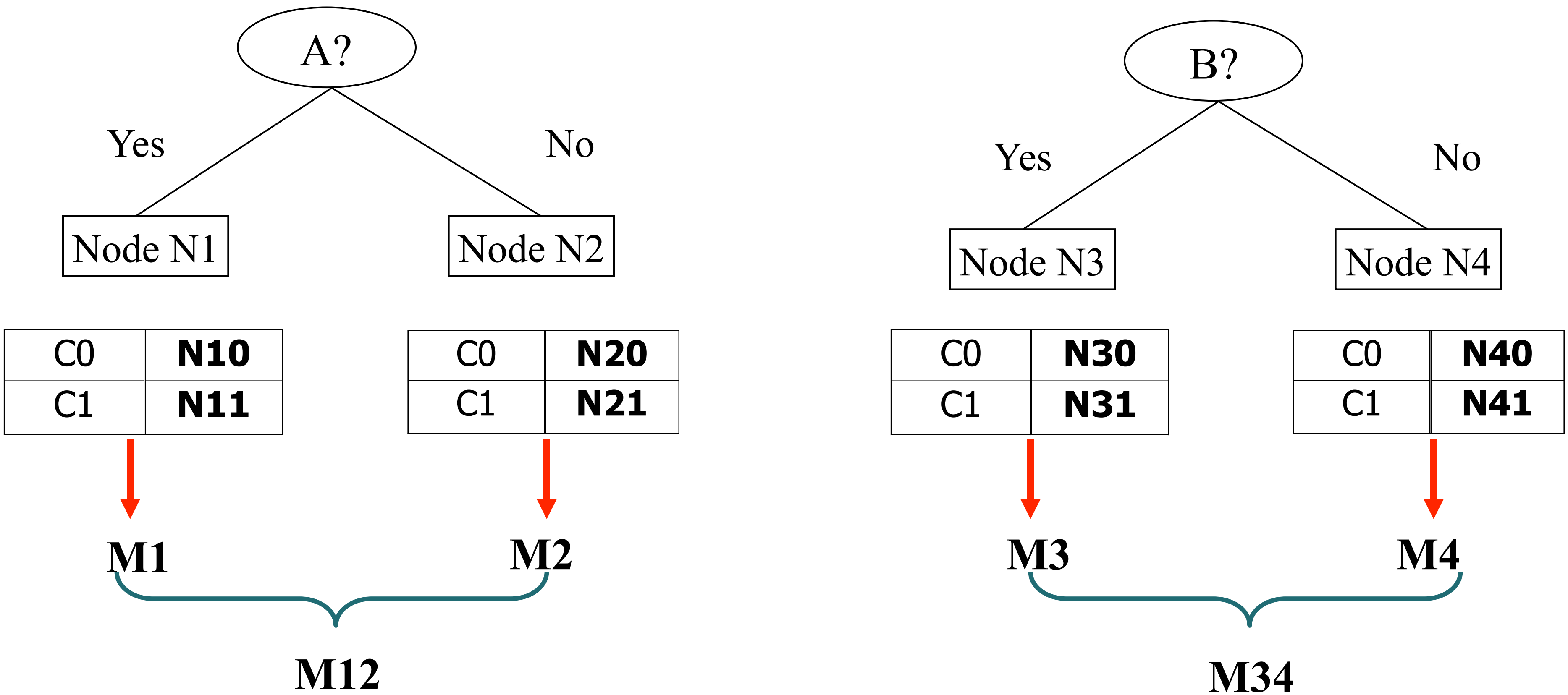
- **Gini Index**
- **Entropy**
- **Misclassification error**

How to Find the Best Split

Before Splitting:

C0	**N00**
C1	**N01**

→ **M0**



where N10+N20=N00
N11+N21=N01

where N30+N40=N00
N31+N41=N01

Gain = M0 – M12 vs. M0 – M34

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).

- **Maximum ($1 - 1/n_c$) when records are equally distributed among all classes**
- **Minimum (0.0) when all records belong to one class**

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	

- **This index is a measure of statistical dispersion developed by the Italian statistician and sociologist Corrado Gini (the index was published in 1912)**

Measure of Impurity: GINI

- **Min value of the index:**
  - **A class with a relative frequency of 1, all the others 0**

$$1 - \sum_{j=1}^{n_c} p_j^2 = 1 - 1^2 = 0$$

- **Max value of the index:**
  - **n_c classes with the same frequency:**

$$1 - \sum_{j=1}^{n_c} p_j^2 = 1 - \sum_{j=1}^{n_c} (\frac{1}{n_c})^2 = 1 - n_c(\frac{1}{n_c})^2 = 1 - \frac{1}{n_c}$$

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 CART, SLIQ, SPRINT.**
- **When a node p is split into k partitions (children), the quality of split is computed as,**

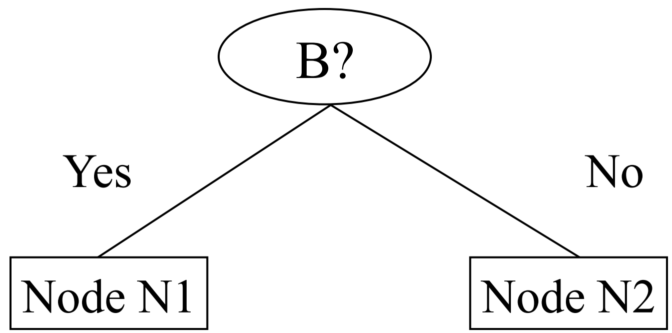
$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

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

n **= number of records at** ***father*** **node** ***p***

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=0.333		

Gini(Children)
$= 7/12 * 0.408 +$
$5/12 * 0.32$
$= 0.371$

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	2	1
C2	4	1	1
Gini	0.393		

Two-way split
(find best partition of values)

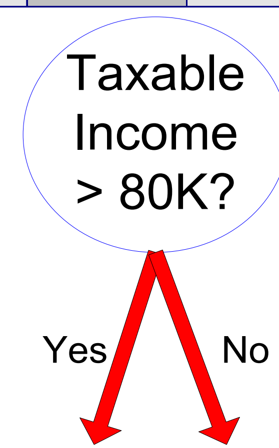
	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

	CarType	
	{Sports}	{Family, Luxury}
C1	2	2
C2	1	5
Gini	0.419	

Continuous Attributes: Computing Gini Index

- **Use Binary Decisions based on one value**
- **Several Choices for the splitting value**
  - **Number of possible splitting values = Number of distinct values**
- **Each splitting value has a count matrix associated with it**
  - **Class counts in each of the partitions, $A < v$ and $A \geq v$**
- **Simple method to choose best v**
  - **For each v, scan the database to gather count matrix and compute its Gini index**
  - **Computationally Inefficient! Repetition of work.**

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
  - 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
	Taxable 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	

Alternative Splitting Criteria based on Entropy

- **Entropy at a given node t:**

$$Entropy(t) = -\sum_{j} p(j \mid t) \log p(j \mid t)$$

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

- **Measures homogeneity 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 based computations are similar to the GINI index computations**

Measure of Impurity: Entropy

- **Min value of the index:**
  - **A class with a relative frequency of 1, all the others 0**

$$-\sum_{j=1}^{n_c}(p_j \log p_j) = -1 \log 1 = 0$$

- **Max value of the index:**
  - **n_c classes with the same frequency:**

$$-\sum_{j=1}^{n_c}(p_j \log p_j) = -\sum_{j=1}^{n_c}(\frac{1}{n_c} \log \frac{1}{n_c}) =$$

$$= -n_c \frac{1}{n_c} \log \frac{1}{n_c} = -(\log 1 - \log n_c) = \log n_c$$

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 Information Gain

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

Splitting Based on Information Gain

- **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 of the partitioning (large number of small partitions) is penalized!**
- **Used in CART, and designed to overcome the disadvantage of Information Gain**

Splitting Criteria based on Classification Error

- **Classification error at a node t :**

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

- **Measures misclassification error made by a node.**
  - **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: Classification Error

- **Min value of the index:**
  - **A class with a relative frequency of 1, all the others 0**

$$1 - \max_{j=1\ldots n_c} p_j = 1 - 1 = 0$$

- **Max value of the index:**
  - **n_c classes with the same frequency:**

$$1 - \max_{j=1\ldots n_c} p_j = 1 - \frac{1}{n_c}$$

Examples for Computing Error

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

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

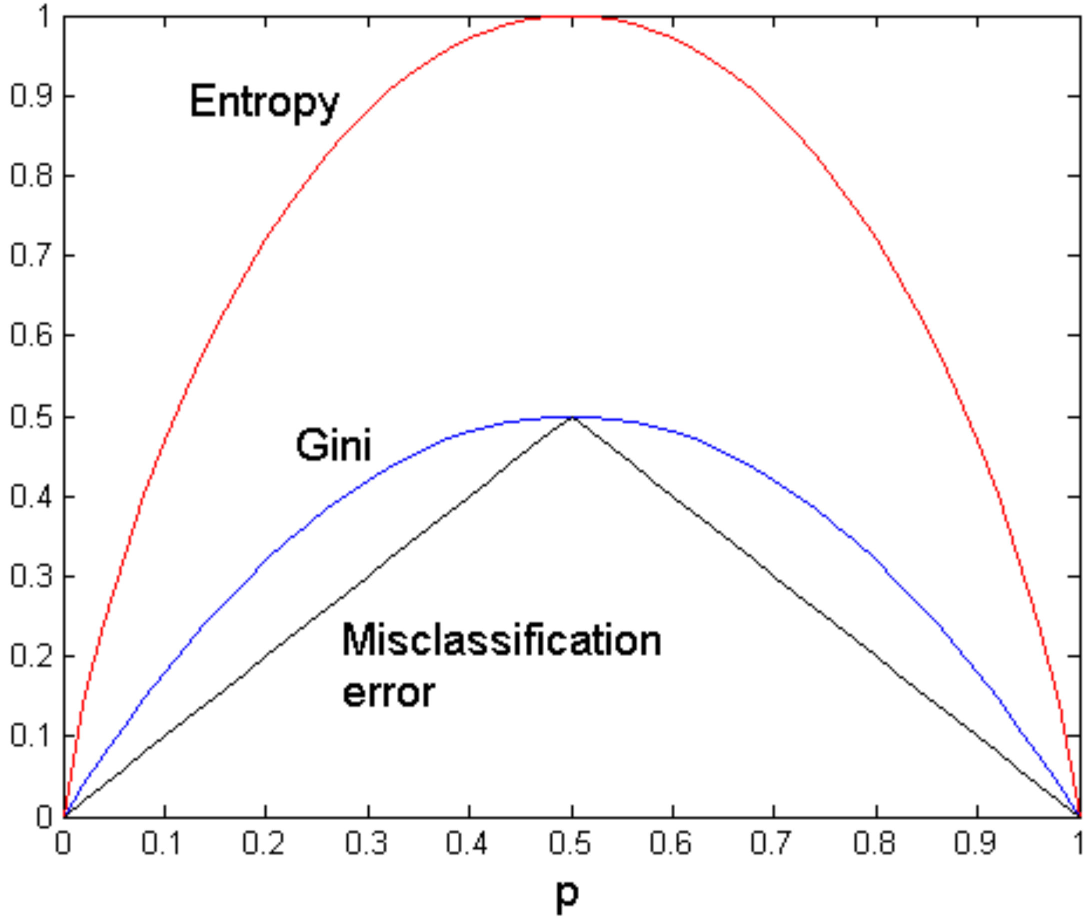
C1	**2**
C2	**4**

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

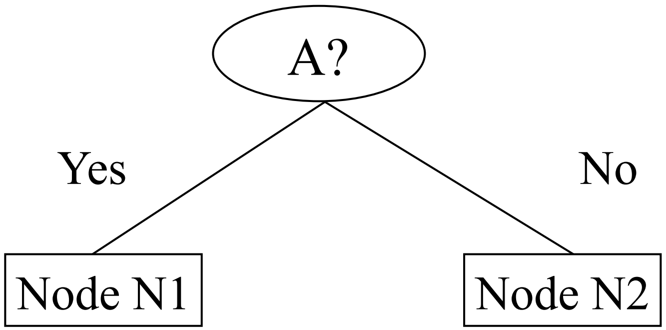
Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Comparison among Splitting Criteria

For a 2-class problem:



Misclassification Error vs Gini



	Parent
C1	7
C2	3
Gini = 0.42	

Gini(Parent)
$= 1 - (7/10)^2 - (3/10)^2$
$= 0.42$

ME(Parent) = $1 - 7/10 = 0.3$

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)
$= 3/10 * 0 + 7/10 * 0.489$
$= 0.342$

ME(N1) = $1 - 3/3 = 0$

ME(N2) = $1 - (4/7) =$
$= 0.428$

ME(Children) = $3/10 * 0$
$+ 7/10 * 0.428$
$= 0.2996$

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 the records have similar attribute values
- Early termination (to be discussed later)

Decision Tree Based Classification

- **Advantages:**
  - Inexpensive to construct
  - 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: 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.

Skeleton of a Decision Tree induction algorithm

Algorithm 7.3.1 (Generate_decision_tree) Generate a decision tree from the given training data.

Input: The training samples, *samples*, represented by discrete-valued attributes; the set of candidate attributes, *attribute-list*.

Output: A decision tree.

Method:

```
create a node N;
if samples are all of the same class, C then
    return N as a leaf node labeled with the class C;
if attribute-list is empty then
    return N as a leaf node labeled with the most common class in samples; // majority voting
select test-attribute, the attribute among attribute-list with the highest information gain;
label node N with test-attribute;
for each known value a_i of test-attribute // partition the samples
    grow a branch from node N for the condition test-attribute=a_i;
    let s_i be the set of samples in samples for which test-attribute=a_i; // a partition
    if s_i is empty then
        attach a leaf labeled with the most common class in samples;
    else attach the node returned by Generate_decision_tree(s_i, attribute-list - test-attribute);
```

- **This algorithm assumes that attributes are categorical**
  - **Multi-way split at each test node**
  - **When an attribute is selected, it is removed from the *attribute-list***

Decision Tree: Web Mining Application

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.

- **Web Usage Mining: exploit a Decision Tree to clean a log**
  - **Classify the accesses as performed by either humans or robots**
- **Each log record is an access, already segmented on the basis of the IP address**
  - **note Requested and Referrer/From page**

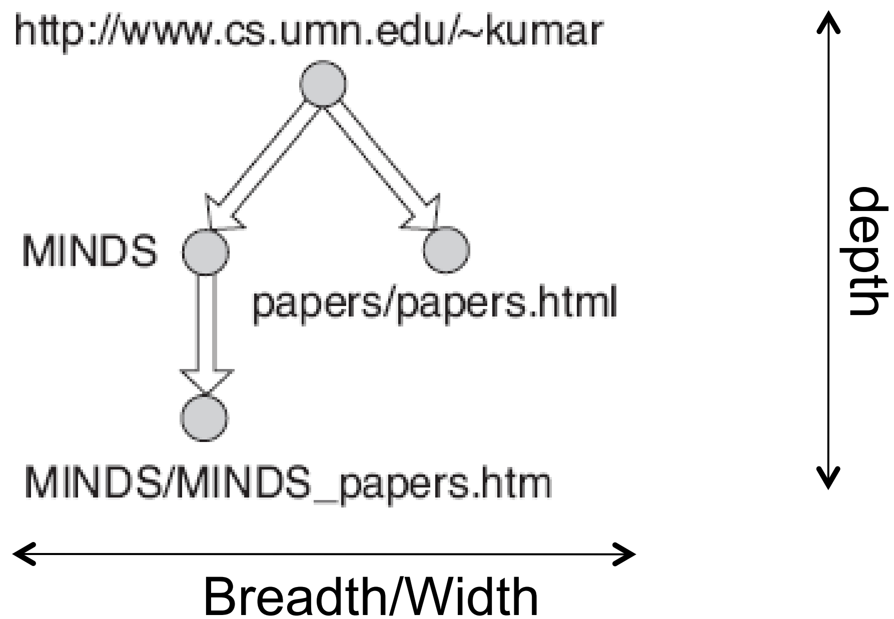
Decision Tree: Web Mining Application

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.

- **Each session is a directed graph**
  - **node: pages**
  - **edges: hyperlinks**

Decision Tree: Web Mining Application



Graph of session 1

Decision Tree: Web Mining Application

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.

- **Extraction of features characterizing the variuos sessions**

Decision Tree: Web Mining Application

- Build a trainig/test dataset (a portion of the Web log file), by annotating each record (supervised knowledge)
  - 2916 records
  - 50% clas 1 (Web Robots)
  - 50% class 0 (Human Users)
- 10% training dataset
- 90% test dataset

- Decision treee induction.
- The decision tree is used to classify and remove the robot sessions

Decision Tree: Web Mining Application

```
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
| | breadth > 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
```

- **The robots (*class* 1) have sessions with large “breadth/width” and short “depth”**
- **The humans (*class* 0) generate sessions with small “breadth”, but a larger “depth”**
  - **They surf the Web with exploration that are more focused**
- **The robots (some of them) do not download images**
- **The robot sessions are longer that the human ones**
- **The robots repeat the page request when they follow the back hyperlinks, whereas the humans do not repeat it due to the browser cache**

Metrics for Performance Evaluation

- **Focus on the predictive capability of a model**
  - **Rather than how fast it takes to classify or build models, scalability, etc.**

- **Confusion Matrix:**

	PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Metrics for Performance Evaluation

	PREDICTED CLASS	
	Class=Yes	**Class=No**
ACTUAL CLASS Class=Yes	a (TP)	b (FN)
ACTUAL CLASS Class=No	c (FP)	d (TN)

- **Most widely-used metric:**

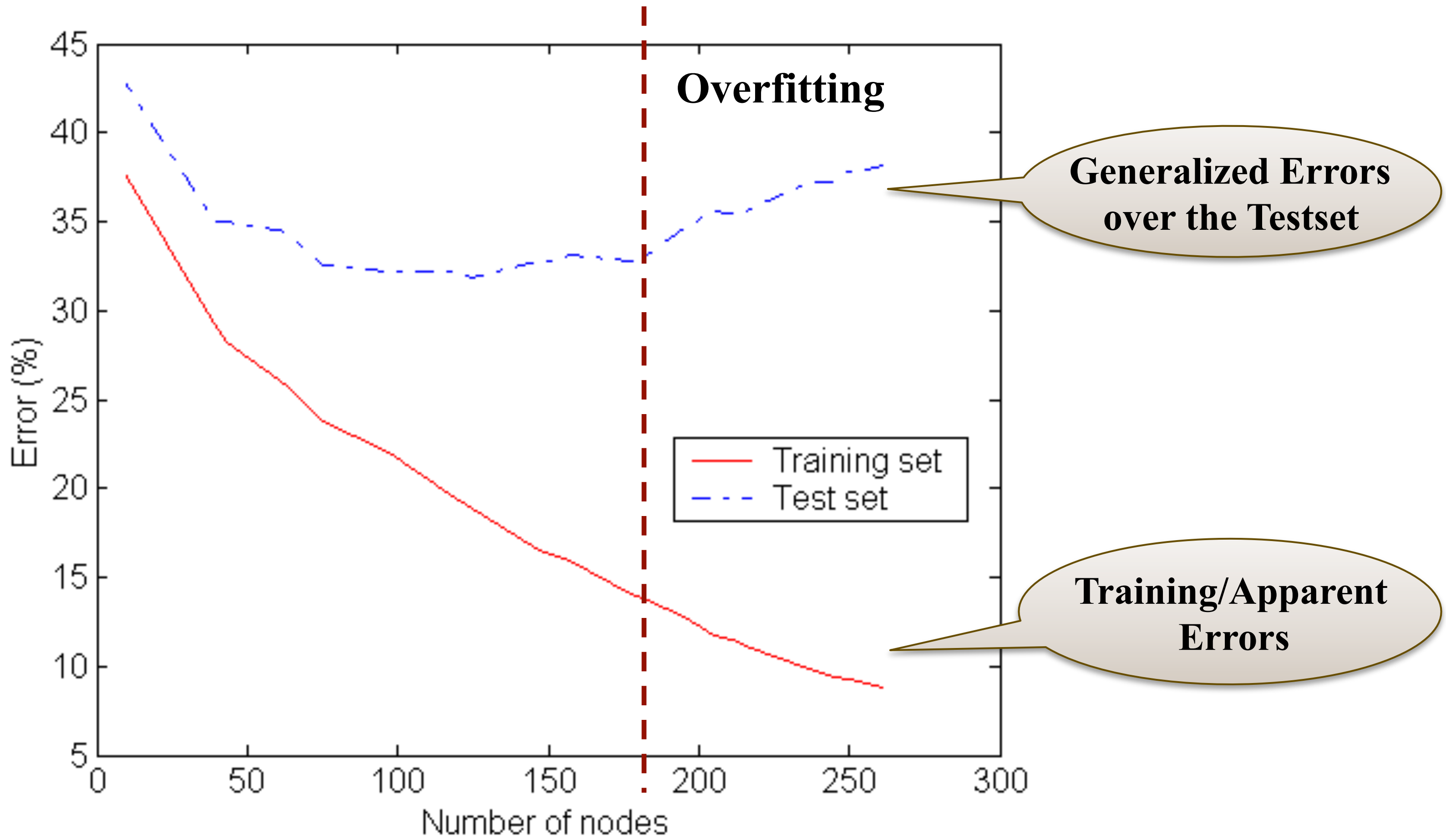
$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

$$Error\_rate = 1 - Accuracy = \frac{b+c}{a+b+c+d} = \frac{FN+FP}{TP+TN+FP+FN}$$

Practical Issues of Classification

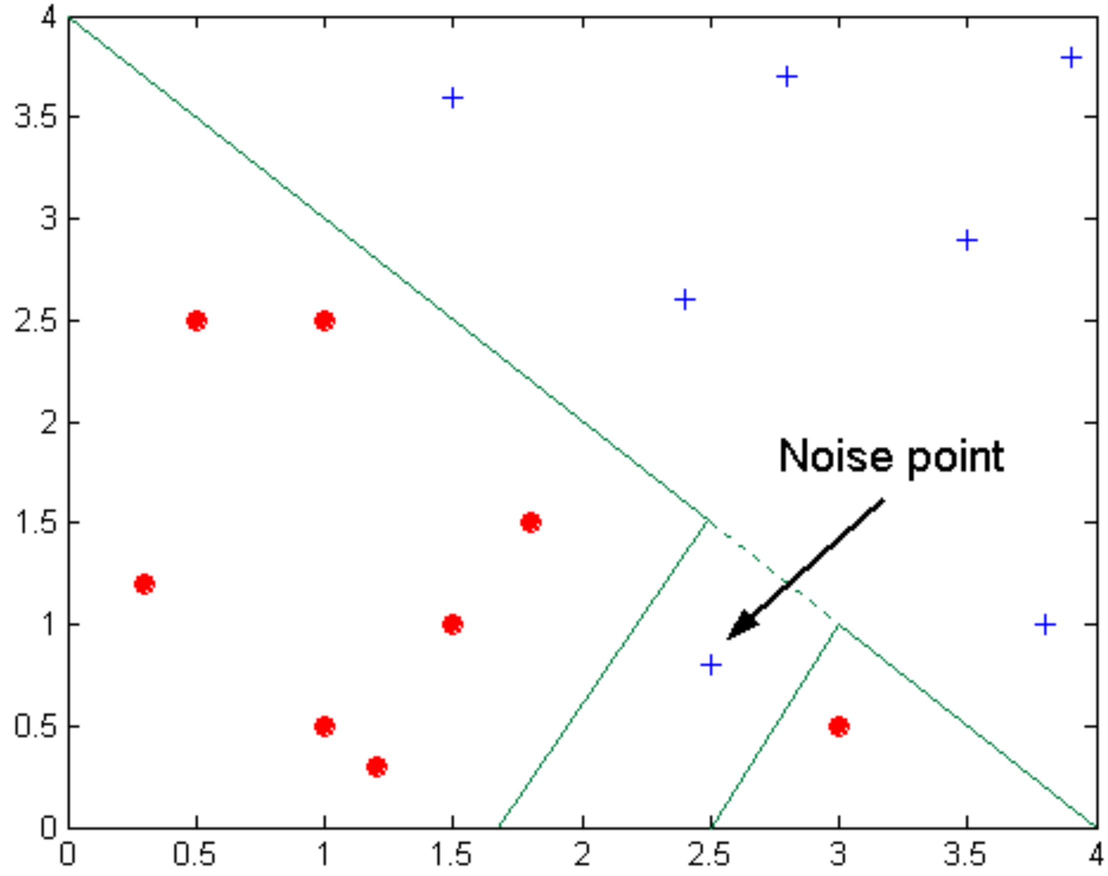
- **Underfitting and Overfitting**
- **Missing Values**
- **Costs of Classification**

Underfitting and Overfitting



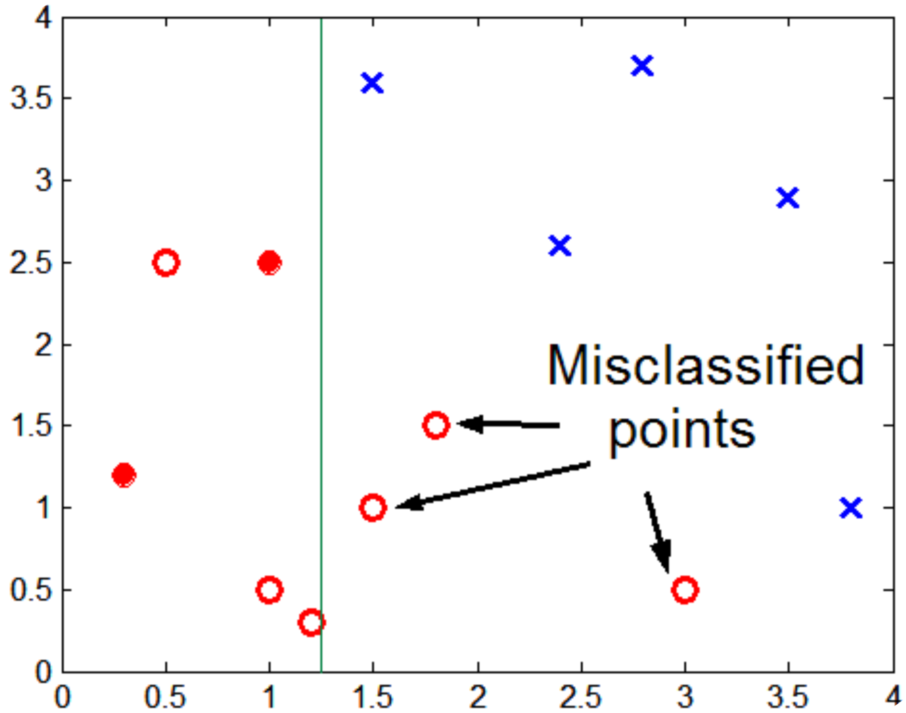
Underfitting: when model is too simple, both training and test errors are large

Overfitting due to Noise



Decision boundary is distorted by noise point

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

Notes on Overfitting

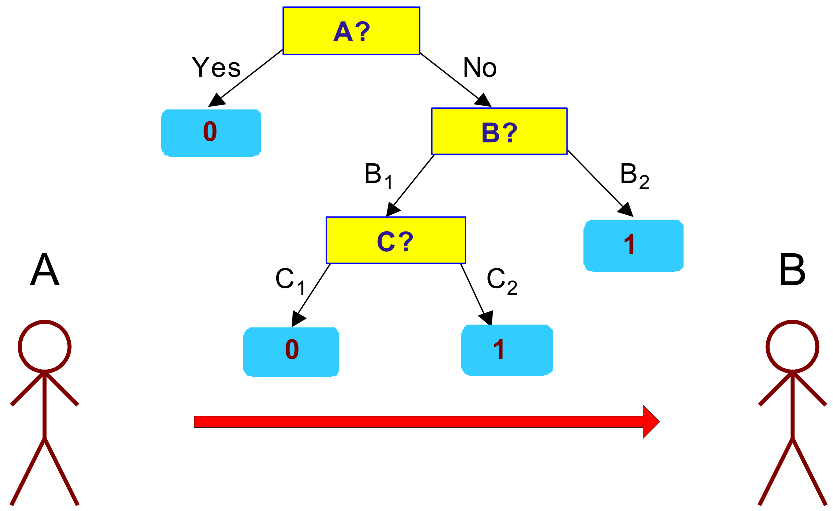
- **Overfitting results in decision trees that are more complex than necessary**
- **Training error no longer provides a good estimate of how well the tree will perform on previously unseen records**
- **Need new ways for estimating errors**

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

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) + Cost(Model)**
  - **Cost is the number of bits needed for encoding.**
  - **Search for the least costly model.**
- **Tradeoff**
  - **Cost(Data|Model) encodes the misclassification errors, and <u>decreases</u> for over-fitted models.**
  - **Cost(Model) uses node encoding (number of children) plus splitting condition encoding. This costs <u>increases</u> for over-fitted models.**

How to Address Over-fitting in DT

- **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).

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

Overfitting: 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
| | breadth > 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
```

Subtree Raising

Subtree Replacement

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
```

Handling Missing Attribute Values

- **Missing values affect decision tree construction/ use 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

Tid	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:

Entropy(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:

Entropy(Refund=Yes) = 0

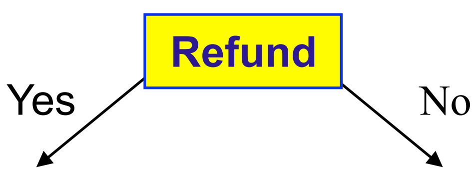
Entropy(Refund=No)
= -(2/6)log(2/6) – (4/6)log(4/6) = 0.9183

Entropy(Children)
= 0.3 (0) + 0.6 (0.9183) = 0.551

Gain = 0.9 × (0.8813 – 0.551) = 0.3303

Distribute Instances

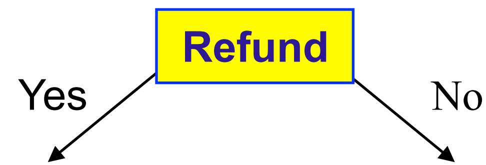
Tid	Refund	Marital Status	Taxable Income	Class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No



Class=Yes	0
Class=No	3

Cheat=Yes	2
Cheat=No	4

Tid	Refund	Marital Status	Taxable Income	Class
10	?	Single	90K	Yes



Class=Yes	0 + 3/9
Class=No	3

Class=Yes	2 + 6/9
Class=No	4

Probability that Refund=Yes is 3/9

Probability that Refund=No is 6/9

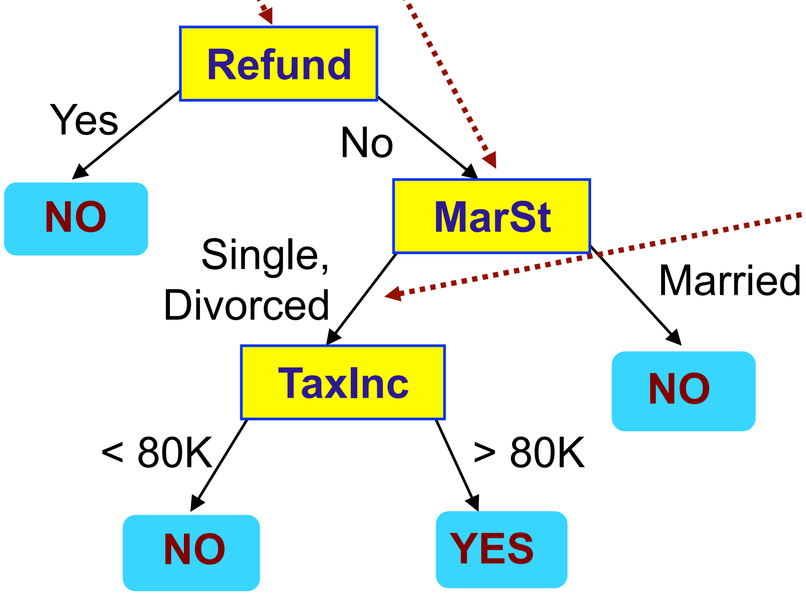
Assign record to the left child with weight = 3/9 and to the right child with weight = 6/9

Classify Instances

New record:

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Class**
11	No	?	85K	?

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



Probability that Marital Status = Married is 3/6.67

Probability that Marital Status ={Single,Divorced} is 3.67/6.67

Other Issues

- **Data Fragmentation**
- **Search Strategy**
- **Expressiveness**
- **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**

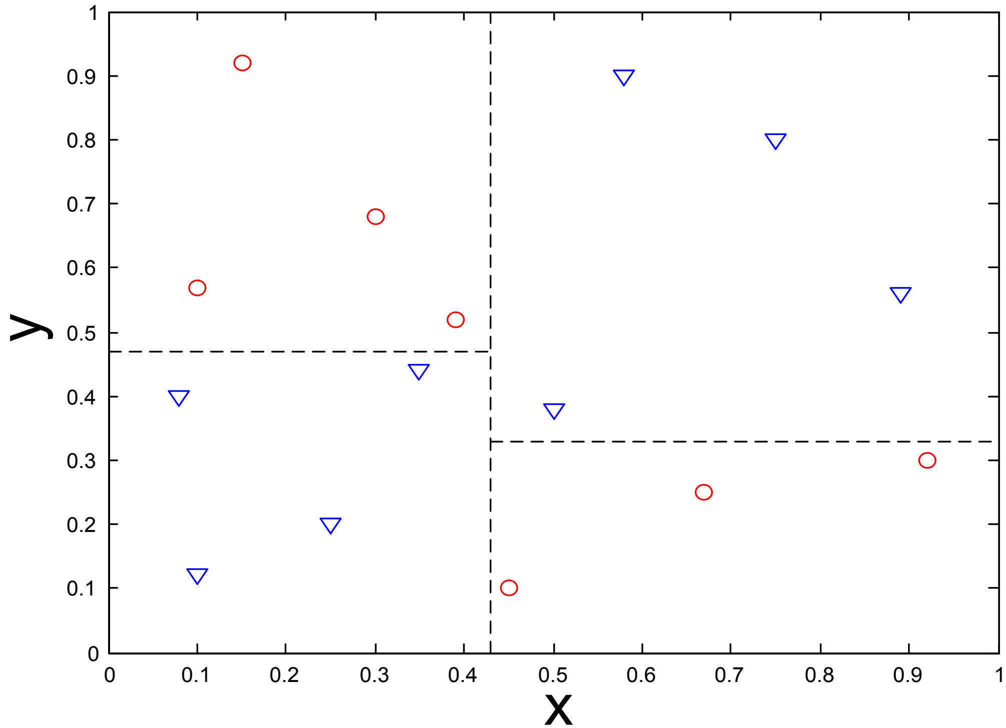
Search Strategy

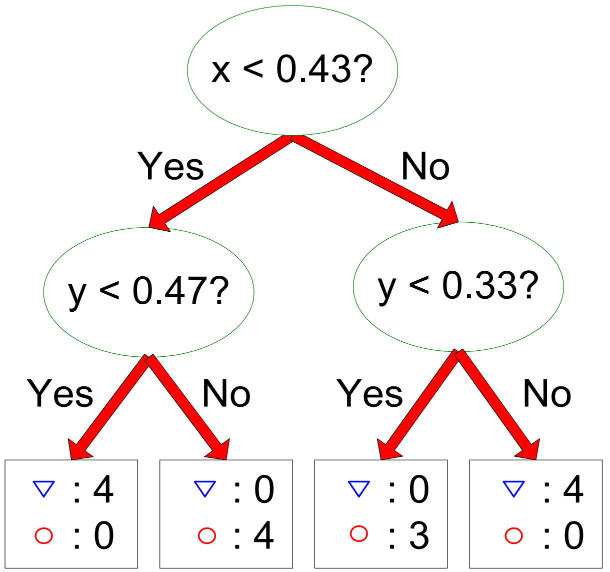
- **Finding an optimal decision tree is NP-hard**
- **The algorithm presented so far uses a greedy, top-down, recursive partitioning strategy to induce a reasonable solution**
- **Other strategies?**
  - **Bottom-up**
  - **Bi-directional**

Expressiveness

- Decision tree provides expressive representation for learning discrete-valued function
  - 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

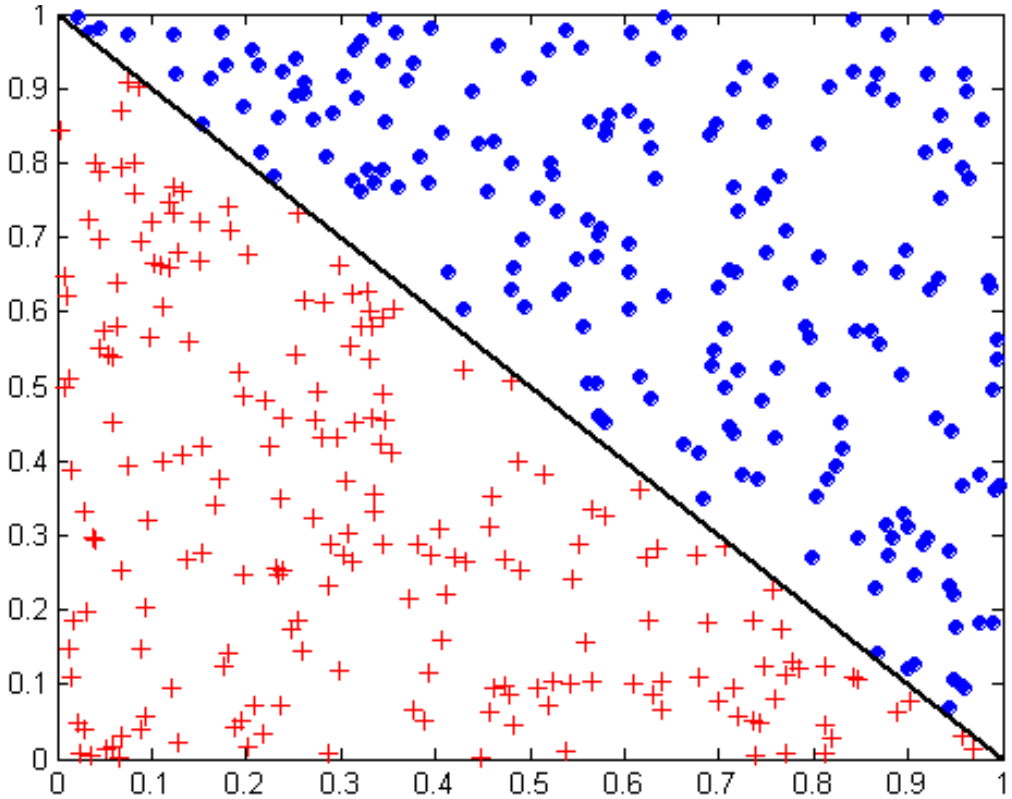
Decision Boundary

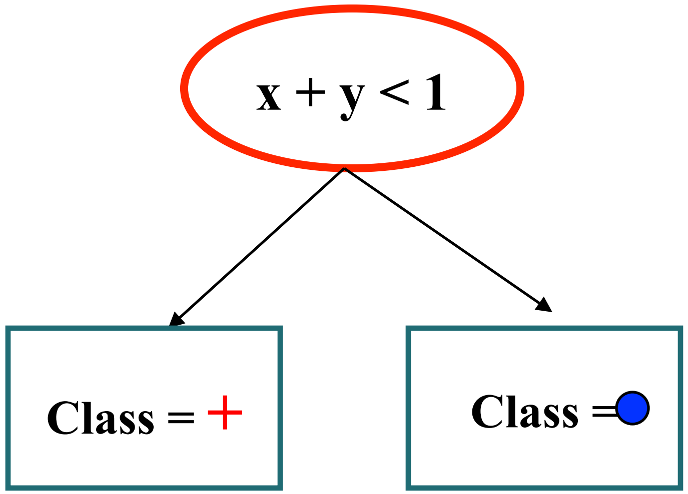




- Border line between two neighboring regions of different classes is known as decision boundary
- Decision boundary is parallel to axes because test condition involves a single attribute at-a-time

Oblique Decision Trees



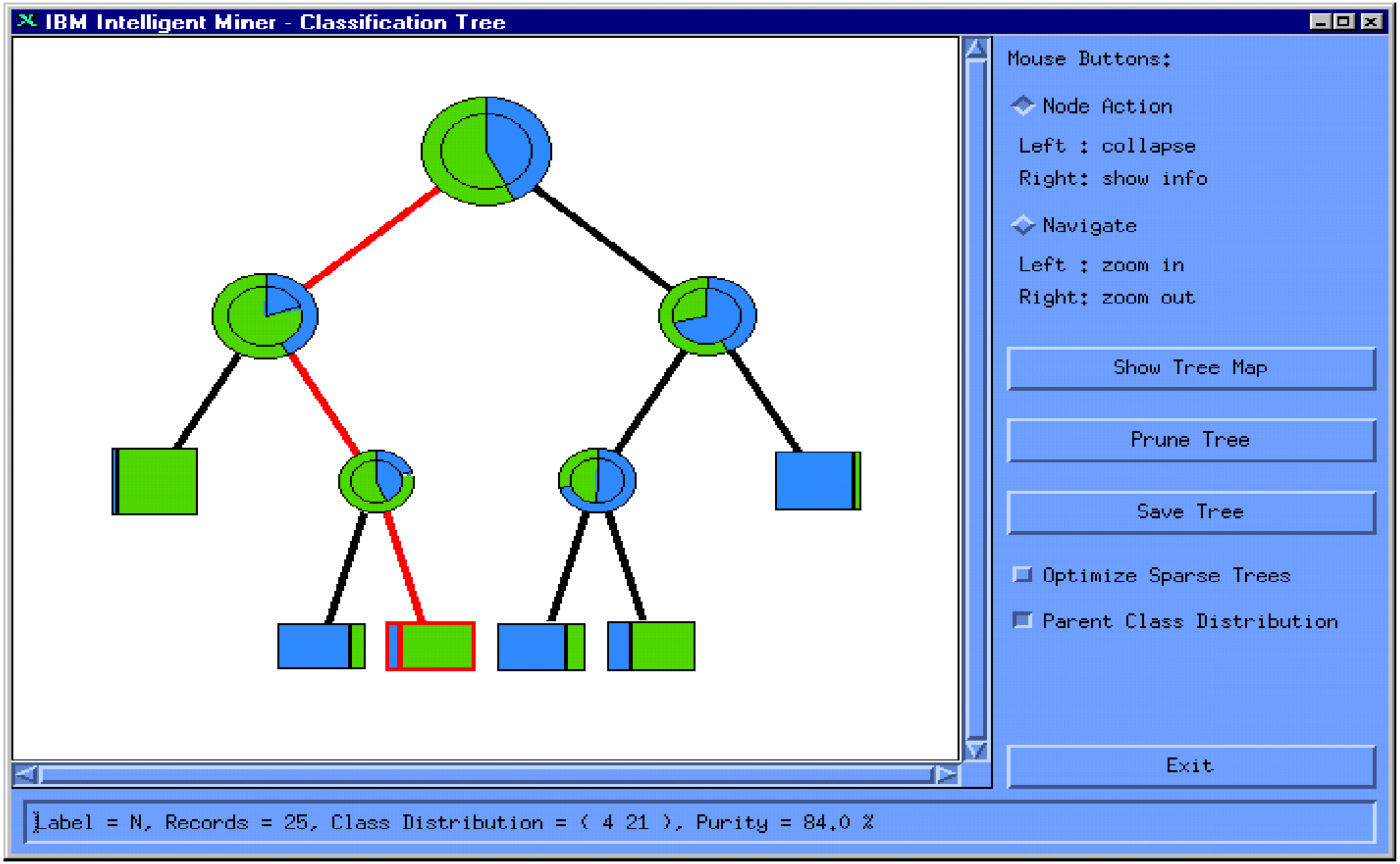


- Test condition may involve multiple attributes
- More expressive representation
- Finding optimal test condition is computationally expensive

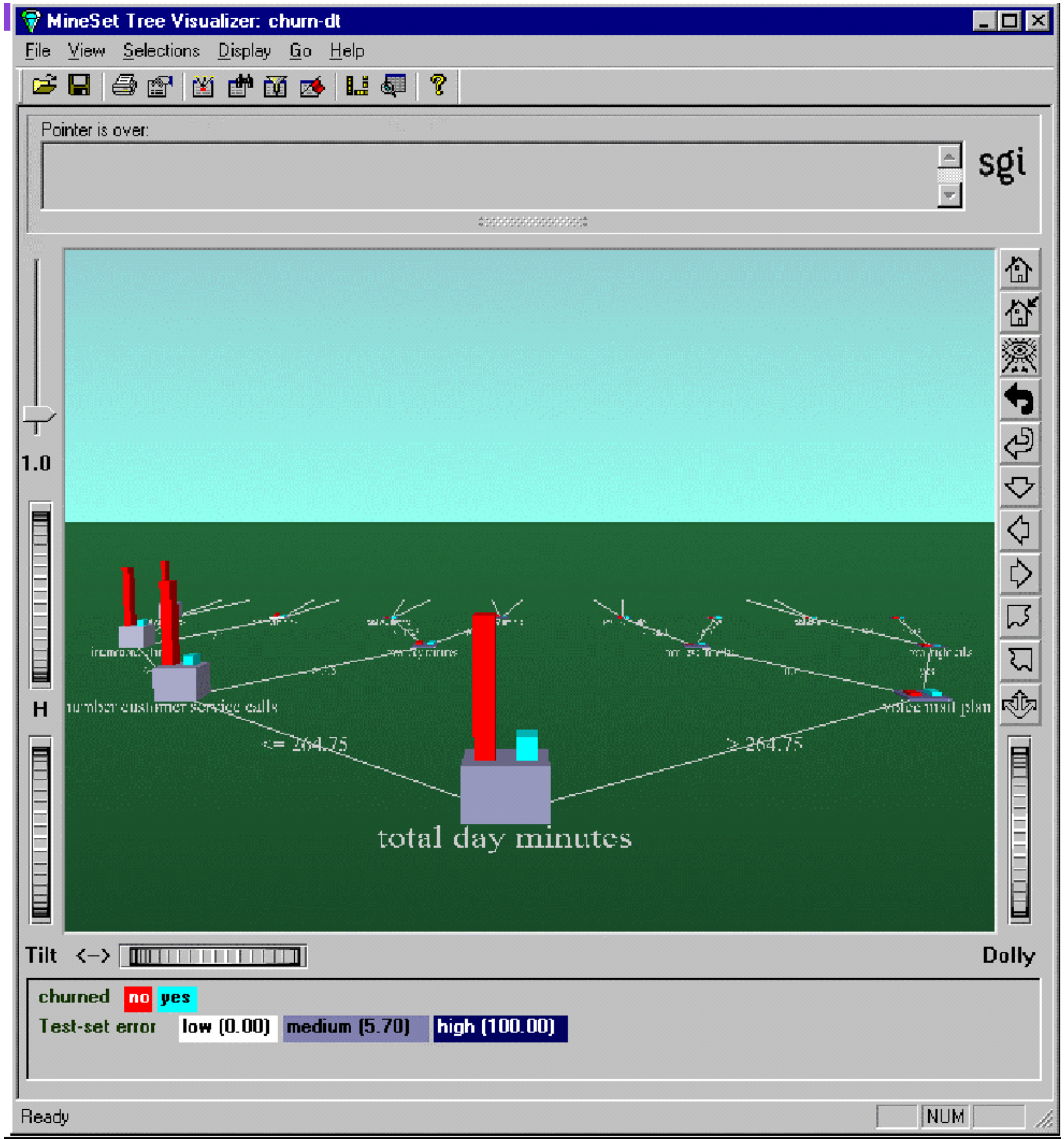
Presentation: decisiontree

```
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
| | breadth > 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
```

Presentation: decisiontree



Presentation: decisiontree



Metrics for Performance Evaluation

	PREDICTED CLASS	
	Class=Yes	**Class=No**
ACTUAL CLASS Class=Yes	a (TP)	b (FN)
ACTUAL CLASS Class=No	c (FP)	d (TN)

- **Most widely-used metric:**

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

$$Error\_rate = 1 - Accuracy = \frac{b+c}{a+b+c+d} = \frac{FN+FP}{TP+TN+FP+FN}$$

Limitation of Accuracy

- Consider a 2-class problem
  - Number of Class No examples = 9990
  - Number of Class Yes examples = 10 <= Rare Class

PREDICTED

ACTUAL	Yes	No
Yes	0 (TP)	10 (FN)
No	0 (FP)	9990 (TN)

- If model predicts everything to be class No, accuracy is 9990/10000 = 99.9 %
  - Accuracy is misleading because model does not detect any examples associated with class “No”

Other measures

- Sensitivity and Specificity: measures for binary classification test
- In many contexts: Rare class = Positive
  - In medical diagnostics: Positive = Disease
- True *positive* rate (TPR) or Sensitivity
  - *TPR = TP / (TP + FN) = TP / actual_pos* (previous example TPR=0)
  - *Fraction of positives returned over all the positives. Sensitivity relates to the test's ability to identify positive results*
- True *negative* rate (TNR) or Specificity
  - *TNR = TN / (TN + FP) = TN / actual_neg* (previous example TNR=9990/9990=1)
  - *Fraction of negatives returned over all the negatives. Specificity relates to the ability of the test to identify negative results.*
- Sensitivity vs. Recall
  - Typical measure in Information Retrieval

    Recall: $r = TP \ / \ (TP + FN)$ ← Sensitivity

    Precision: $p = TP \ / \ (TP + FP)$

    in IR the quantity *TP + FP* corresponds to all the documents that are considered relevant, and are thus returned by the search engine

Cost Matrix

	PREDICTED CLASS		
	C(i|j)	**Class=Yes**	**Class=No**
ACTUAL CLASS	**Class=Yes**	C(Yes|Yes)	C(No|Yes)
	Class=No	C(Yes|No)	C(No|No)

C(i|j): Cost of misclassifying class j example as class I

For instance, diagnosing disease for a healthy person (FP, since Disease=Positive) does not produce the same consequences as to predict health for an ill person (FN) => The cost of predicting health for an ill person should be higher!

Quantifying the consequences of a good or bad classification is a task of the domain expert

Computing Cost of Classification

Cost Matrix	PREDICTED CLASS		
ACTUAL CLASS	C(i|j)	+	-
	+	-1	100
	-	1	0

Model M_1	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	150	40
	-	60	250

Model M_2	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	250	45
	-	5	200

Accuracy = 80%

Cost = 60+4000-150=3910

Accuracy = 90%

Cost = 4255

Cost vs Accuracy

Count	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

Cost	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	p	q
	Class=No	q	p

Accuracy is proportional to cost if
1. C(Yes|No)=C(No|Yes) = q
2. C(Yes|Yes)=C(No|No) = p

$N = a + b + c + d$

$\text{Accuracy} = (a + d)/N$

$$\begin{aligned}\text{Cost} &= p\ (a + d) + q\ (b + c)\\ &= p\ (a + d) + q\ (N - a - d)\\ &= q\ N - (q - p)(a + d)\\ &= N\ [q - (q\text{-}p) \times \text{Accuracy}]\end{aligned}$$

Cost-Sensitive Measures

$$\text{Precision (p)} = \frac{a}{a+c}$$

$$\text{Recall (r)} = \frac{a}{a+b}$$

$$\text{F - measure (F)} = \frac{2rp}{r+p} = \frac{2a}{2a+b+c}$$

- Precision is biased towards C(Yes|Yes) & C(Yes|No)
- Recall/Sensitivity is biased towards C(Yes|Yes) & C(No|Yes)
- F-measure is biased towards all except C(No|No)

$$\text{Weighted Accuracy} = \frac{w_1 a + w_4 d}{w_1 a + w_2 b + w_3 c + w_4 d}$$

Methods for Performance Evaluation

- **How to obtain a reliable estimate of performance?**
- **Performance of a model may depend on other factors besides the learning algorithm:**
  - **Class distribution**
  - **Cost of misclassification**
  - **Size of training and test sets**

Methods of Estimation

- Holdout
  - Reserve 2/3 for training and 1/3 for testing
  - Issue: the two datasets are not independent
- Random subsampling for k times
  - Repeated holdout: $acc_{sub} = \Sigma_{i=1,k}\ acc_i/k$
- Cross validation
  - Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Leave-one-out: k=n
- Bootstrap
  - Sampling with replacement