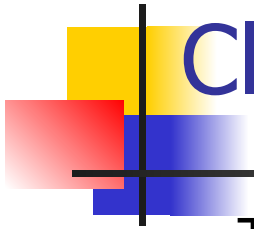
Classification

Jayanti Dansana

Classification Problem

- Given a database $D=\{t_1,t_2,\ldots,t_n\}$ and a set of classes $C=\{C_1,\ldots,C_m\}$, the Classification Problem is to define a mapping $f:D->C$ where each t_i is assigned to one class.
- Actually divides D into equivalence *classes*.
- Prediction is similar, but may be viewed as having infinite number of classes.

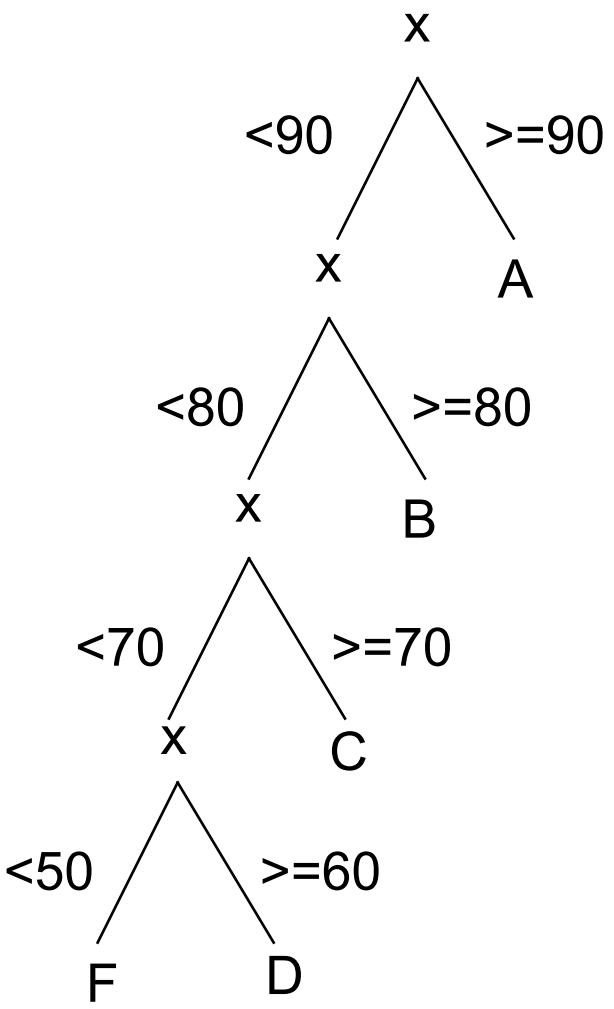
Classification Examples



- Teachers classify students' grades as A, B, C, D, or F.
- Identify mushrooms as poisonous or edible.
- Predict when a river will flood.
- Identify individuals with credit risks.
- Speech recognition
- Pattern recognition

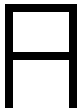
Classification Ex: Grading

- If x >= 90 then grade =A.
- If 80<=x<90 then grade =B.
- If 70<=x<80 then grade =C.
- If 60<=x<70 then grade =D.
- If x<50 then grade =F.

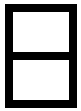


Classification Ex: Letter Recognition

View letters as constructed from 5 components:

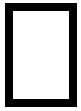


Letter A

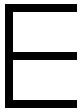


Letter B

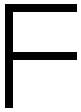
Letter C



Letter D



Letter E



Letter F

Classification: Definition

- Given a collection of records (*training set*)
  - Each record contains a set of *attributes*, one of the attributes is the *class level*.
- Find a *model* for class attribute as a function of the values of other attributes.
- 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.

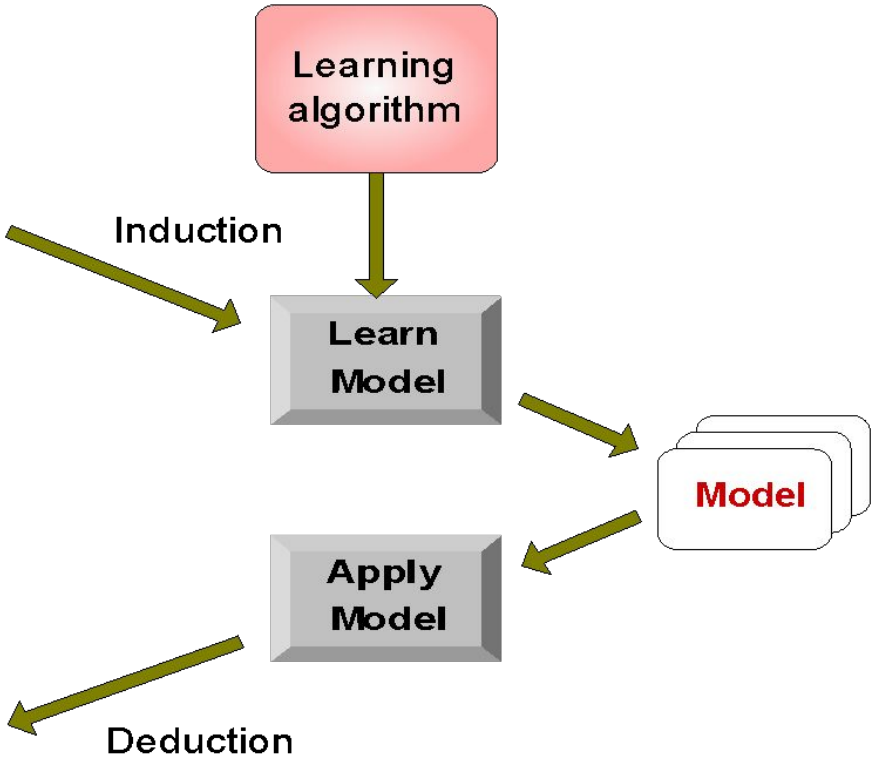
Illustrating Classification Task

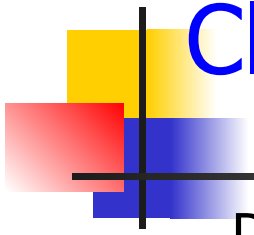
Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

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

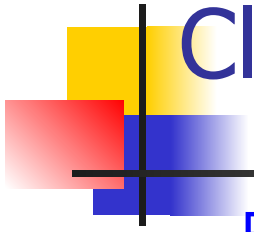




Classification Techniques

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

Classification Using Decision Trees



- Partitioning based: Divide search space into rectangular regions.
- Tuple placed into class based on the region within which it falls.
- DT approaches differ in how the tree is built: DT Induction
- Internal nodes associated with attribute and arcs with values for that attribute.
- Algorithms: ID3, C4.5, CART

Decision Tree

Given:

- D = $\{t_1, \ldots, t_n\}$ where $t_i = \langle t_{i1}, \ldots, t_{ih} \rangle$
- Database schema contains $\{A_1, A_2, \ldots, A_h\}$
- Classes $C = \{C_1, \ldots, C_m\}$

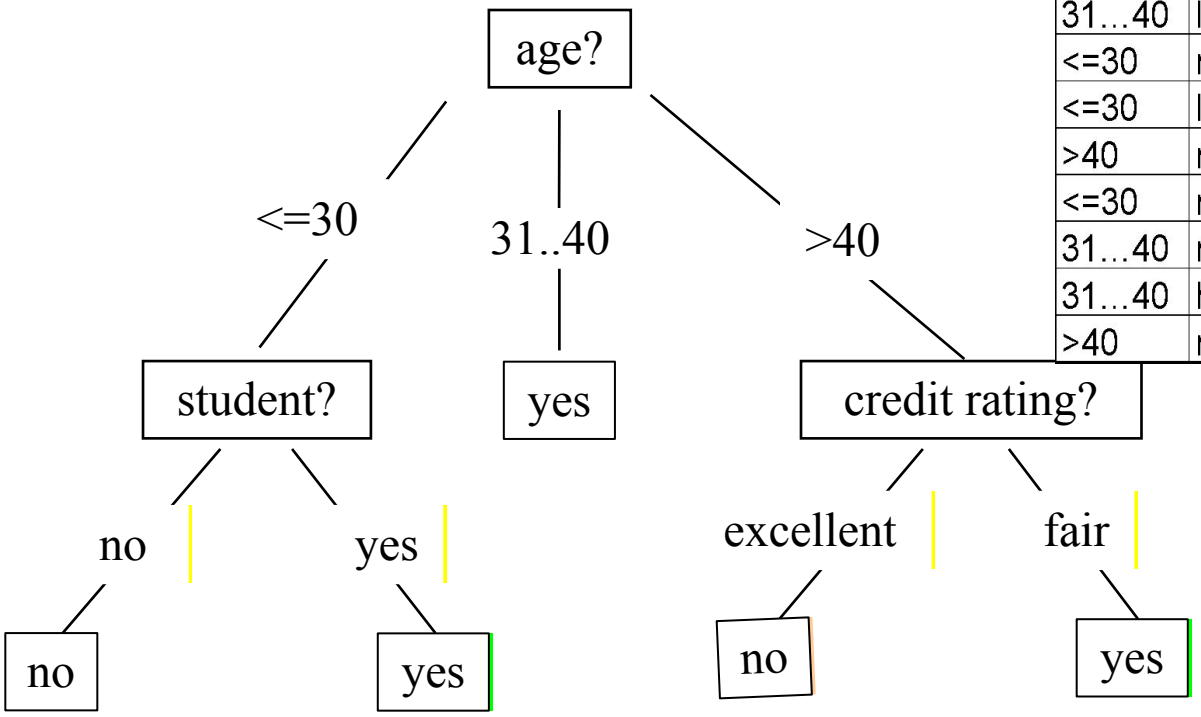
Decision or Classification Tree is a tree associated with D such that

- Each internal node is labeled with attribute, A_i
- Each arc is labeled with predicate which can be applied to attribute at parent
- Each leaf node is labeled with a class, C_j

Decision Tree Induction: An Example

- Training data set: Buys_computer
- Resulting tree:

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no



Classification by Decision Tree Induction

- Decision tree
  - A flow-chart-like tree structure
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
  - Leaf nodes represent class labels or class distribution
- Decision tree generation consists of two phases
  - Tree construction
    - At start, all the training examples are at the root
    - Partition examples recursively based on selected attributes
  - Tree pruning
    - Identify and remove branches that reflect noise or outliers
- Use of decision tree: Classifying an unknown sample
  - Test the attribute values of the sample against the decision tree

Algorithm for Decision Tree Induction

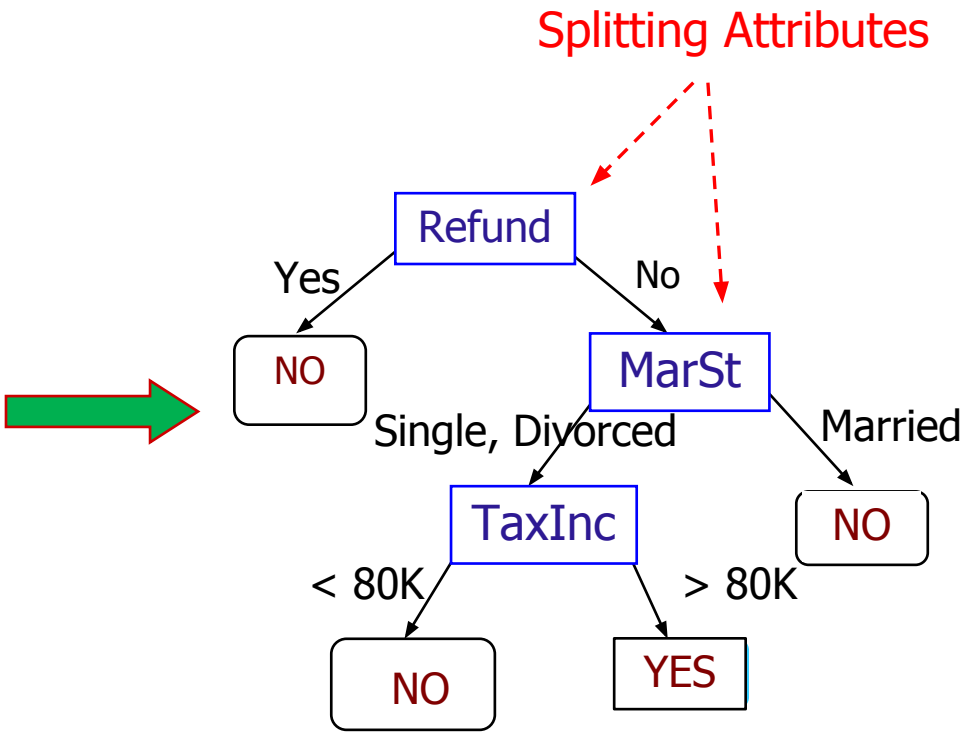
- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

Example of a Decision Tree

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

categorical categorical continuous class

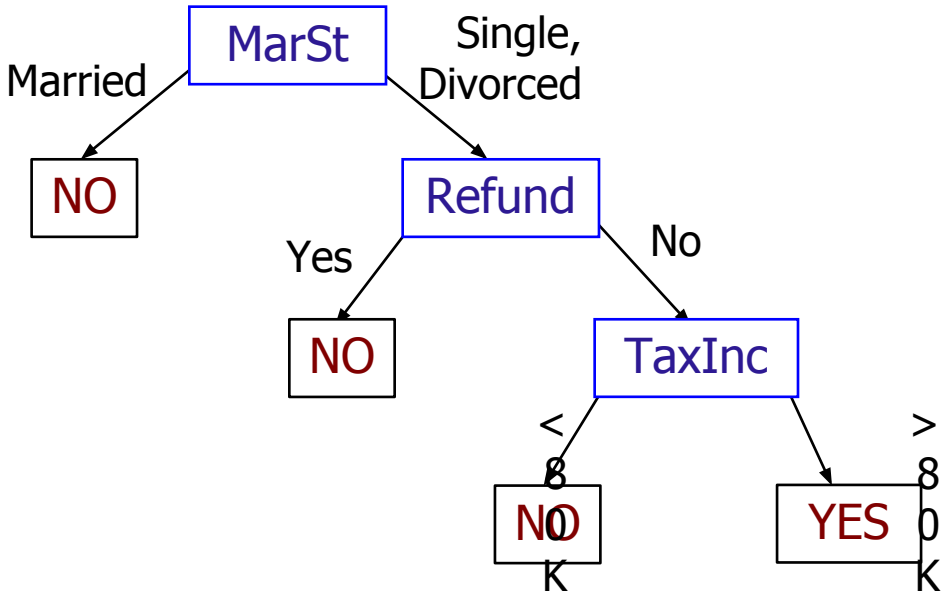
Training Data



Model: Decision Tree

Another Example of Decision Tree

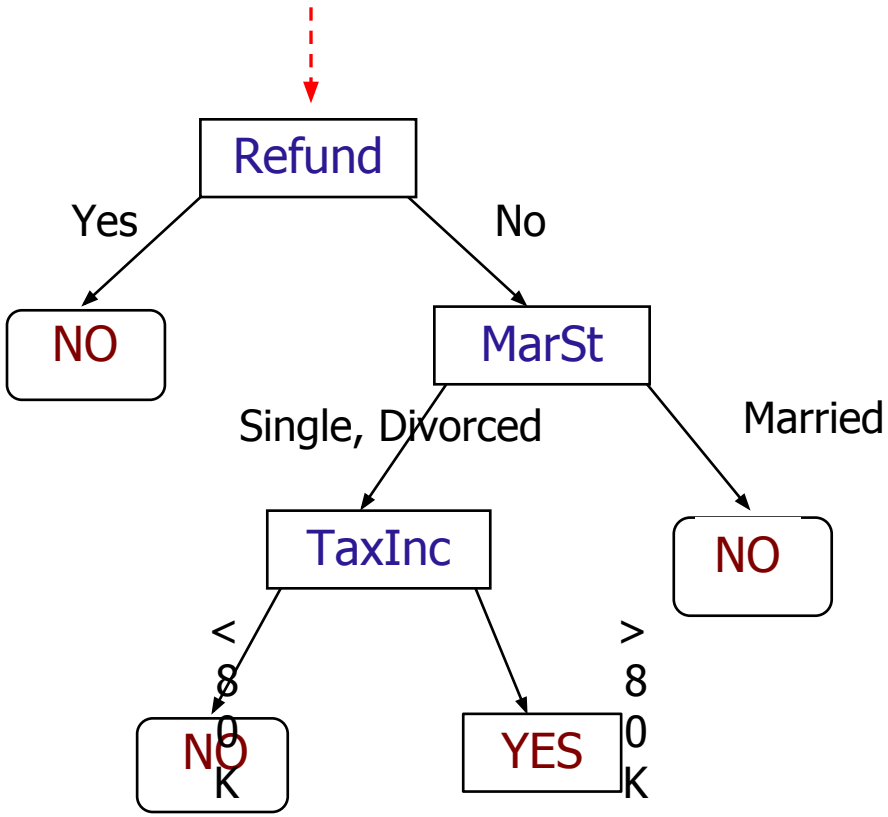
Tid	Refund	Marital Status	Taxable Income	Cheat
		categorical	continuous	class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



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

Apply Model to Test Data

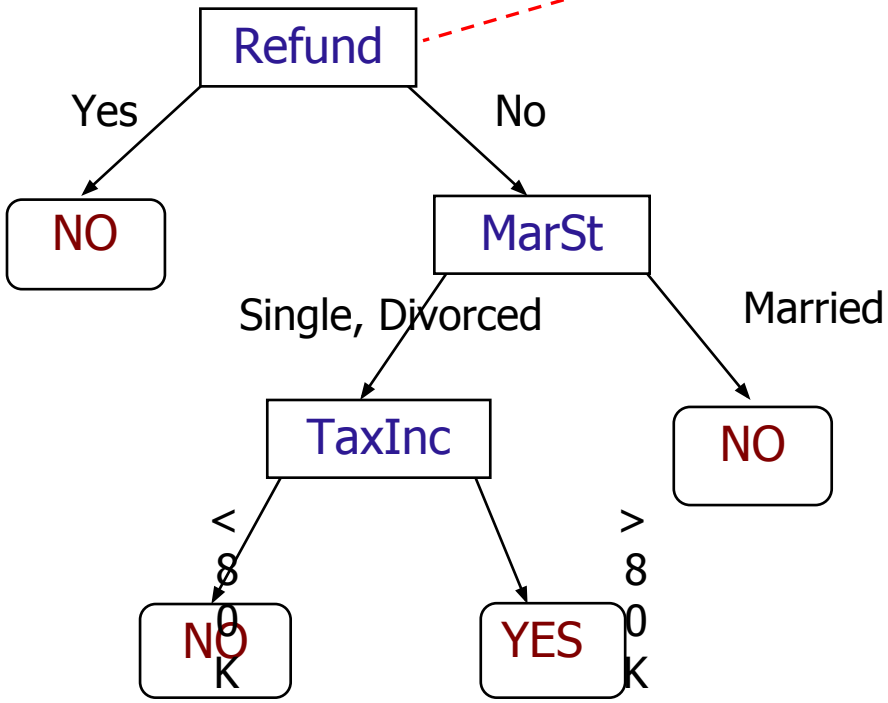
Start from the root of tree.



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



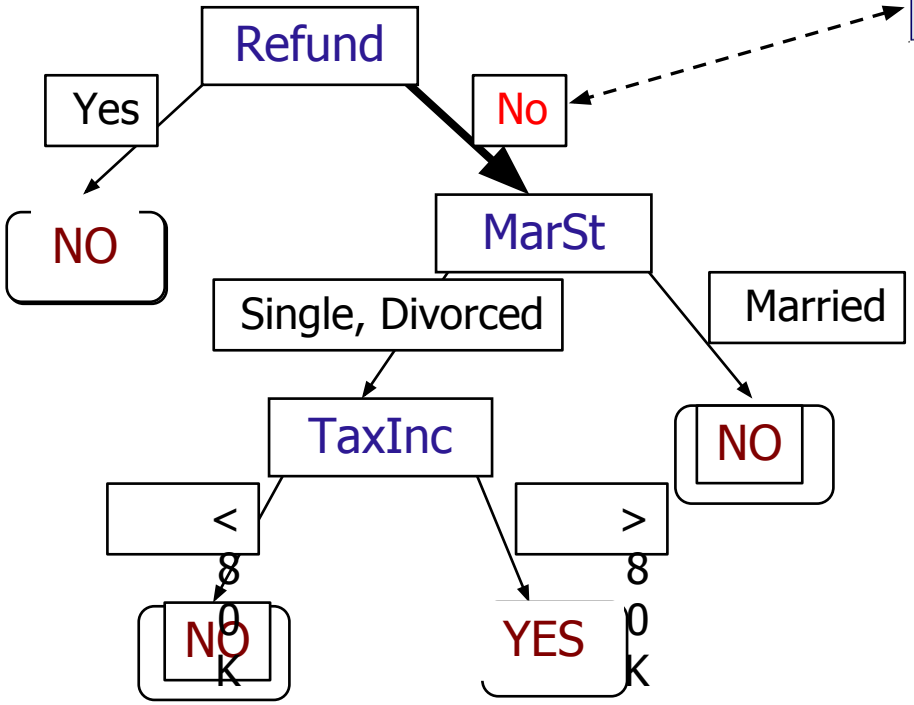
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

Test Data

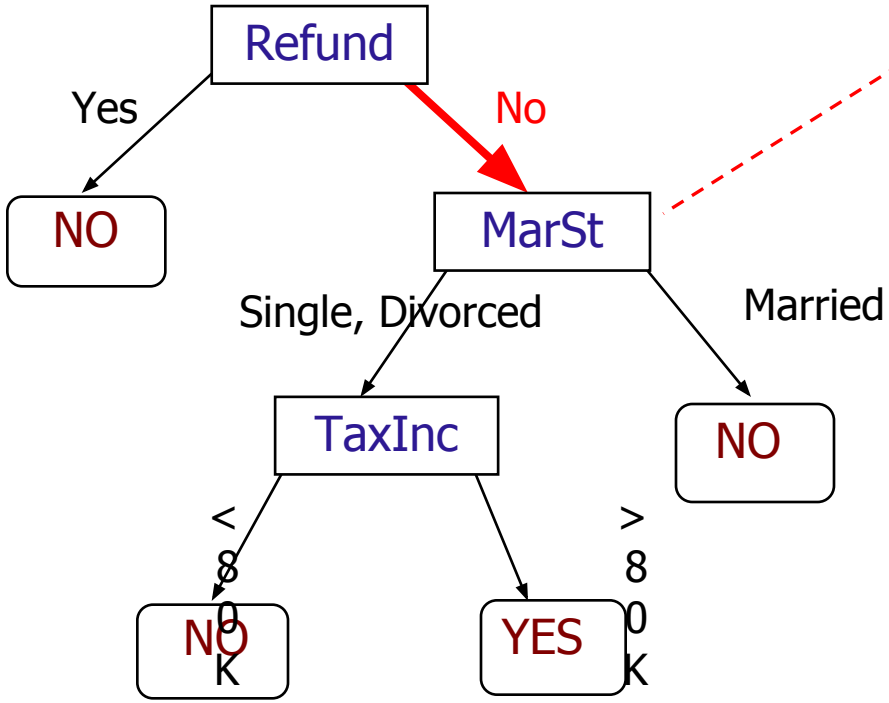
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

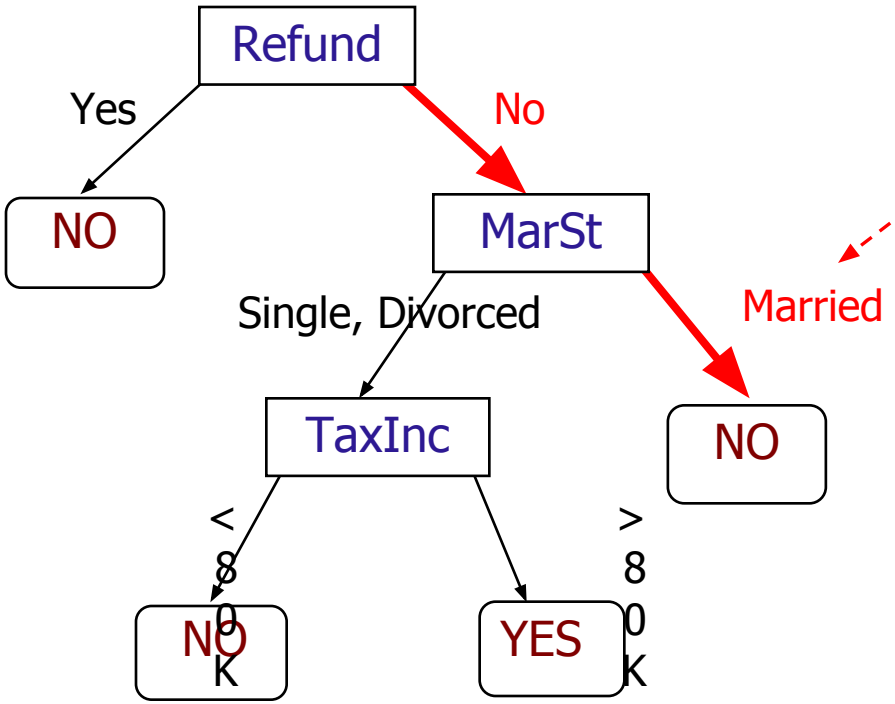
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

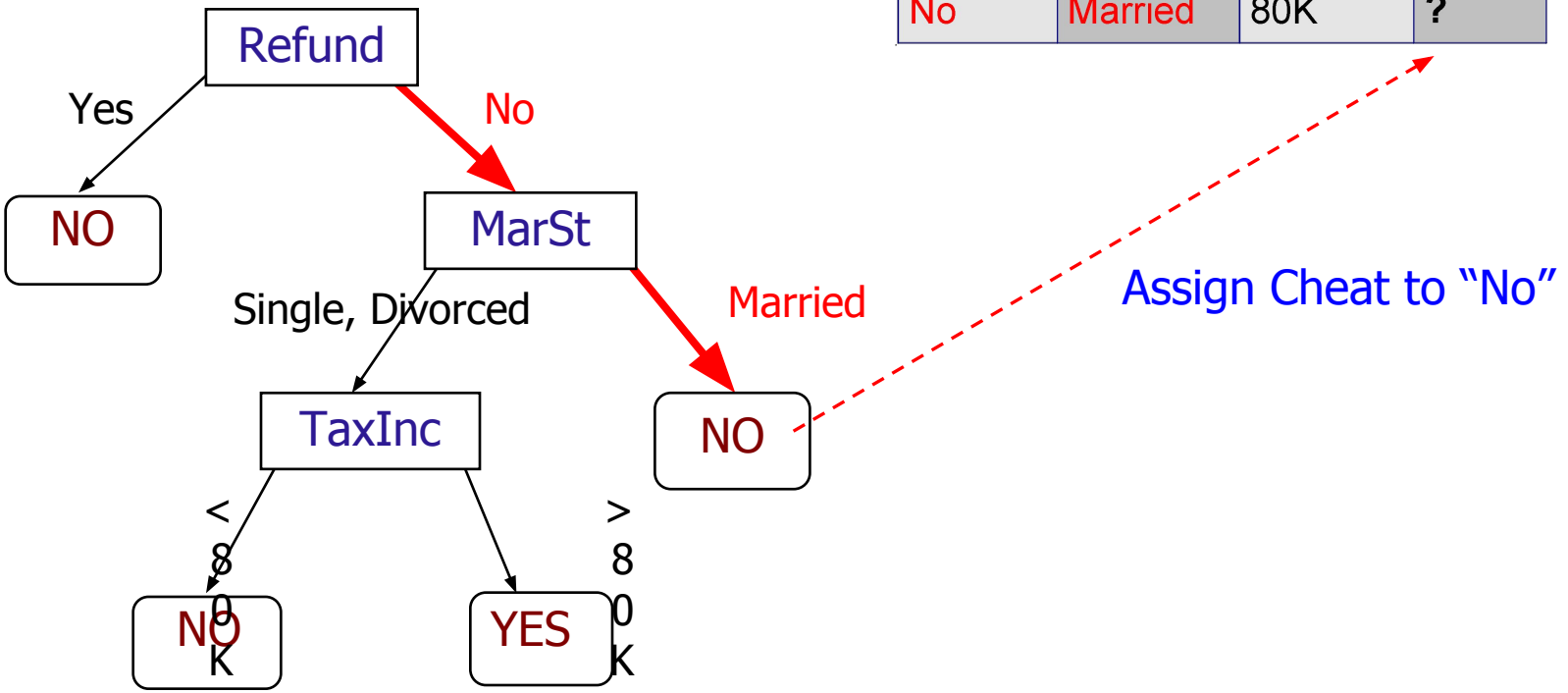
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

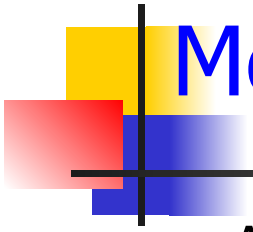
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



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

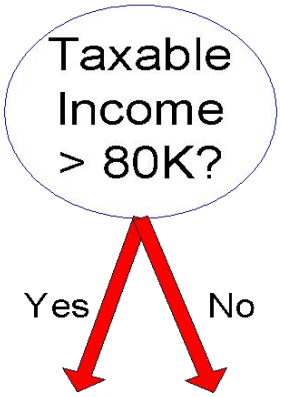


Methods For Splitting

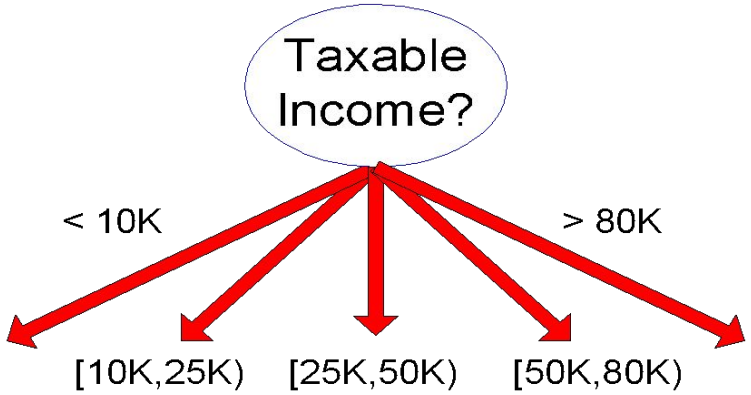
- A key step towards building a decision tree is to find an appropriate test condition for splitting data
- Categorical attributes
  - The test condition can be expressed as an attribute value pair (A = v?), whose outcomes are Yes / No, or as a question about the value of an attribute (A?)

Methods For Splitting

- Continuous attributes
  - The test condition can be expressed in terms of a binary decision (A < v ?) or (A >= v?), whose outcomes are Yes / No, or as a range query whose outcomes are $v_i <= A <= v_{i+1}$, for i = 1, 2, … k



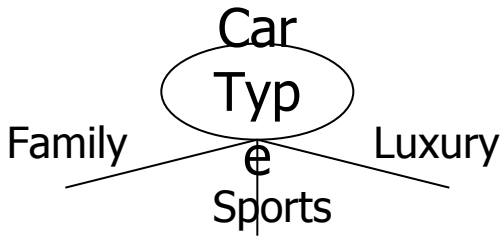
(i) Binary split



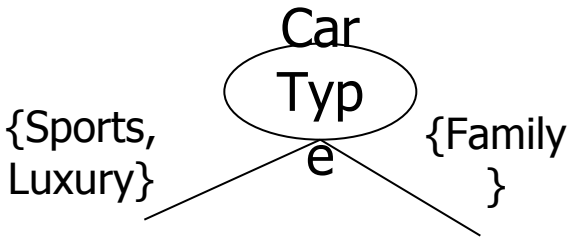
(ii) Multi-way split

Splitting Based on Nominal Attributes

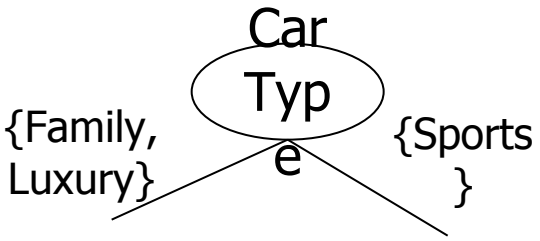
- Multi-way split: Use as many partitions as distinct values.



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

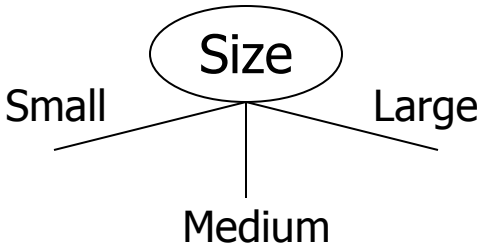


OR

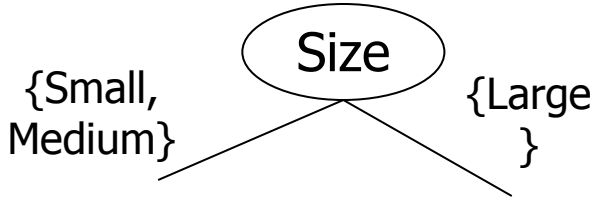


Splitting Based on Ordinal Attributes

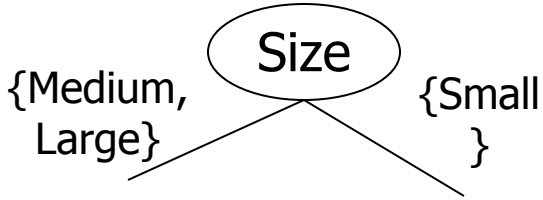
- Multi-way split: Use as many partitions as distinct values.

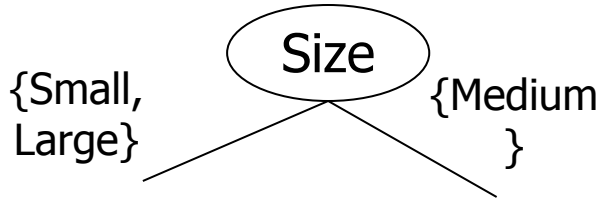


- Binary split: Divides values into two subsets.
  Need to find optimal partitioning.



OR



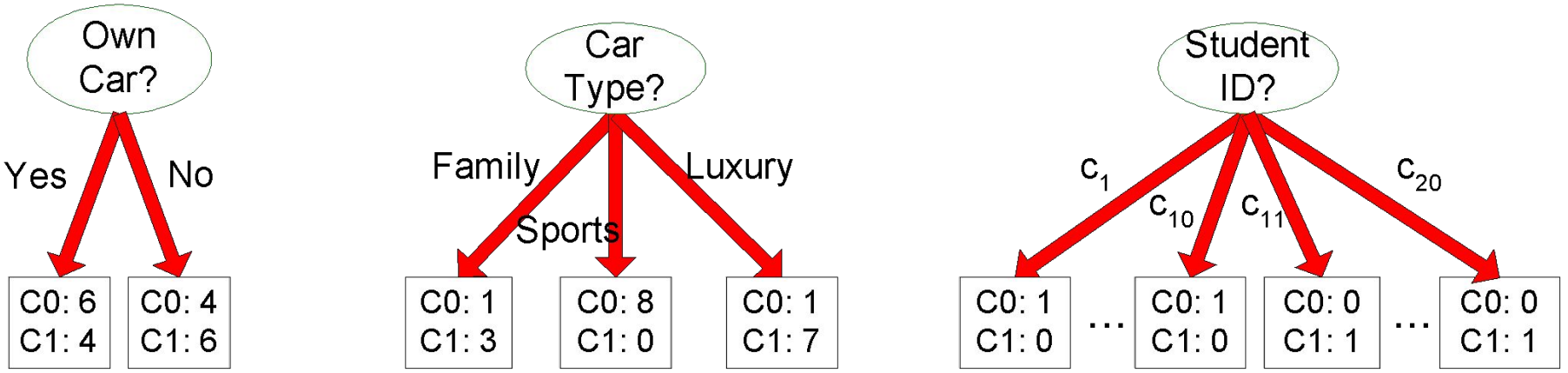


Splitting Based on Continuous Attributes

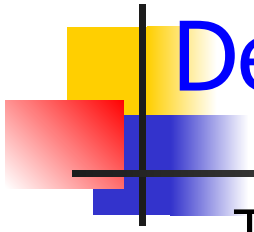
- Different ways of handling
  - Discretization to form an ordinal categorical attribute
    - Static – discretize once at the beginning
    - Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
  - Binary Decision: $(A < v)$ or $(A \geq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

How to determine the Best Split

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



Which test condition is the best?



Decision Tree Algorithms

- The basic idea behind any decision tree algorithm is as follows:
  - Choose the *best* attribute(s) to split the remaining instances and make that attribute a decision node
  - Repeat this process for recursively for each child
  - Stop when:
    - All the instances have the same target attribute value
    - There are no more attributes
    - There are no more instances

Algorithm for Decision Tree Induction (pseudocode)

Algorithm GenDecTree(Sample S, Attlist A)

1. Create a node N
2. If all samples are of the same class C then label N with C; terminate;
3. If A is empty then label N with the most common class C in S (majority voting); terminate;
4. Select a$\in$A, with the highest information gain; Label N with a;
5. For each value v of a:
   a. Grow a branch from N with condition a=v;
   b. Let S_v be the subset of samples in S with a=v;
   c. If S_v is empty then attach a leaf labeled with the most common class in S;
   d. Else attach the node generated by GenDecTree(S_v, A-a)

Splitting Criterion

- There are many test conditions one could apply to partition a collection of records into smaller subsets
- Various measures are available to determine which test condition provides the best split
  - Gini Index
  - Entropy / Information Gain
  - Classification Error

Attribute Selection Measure

- Gini index (IBM IntelligentMiner)
  - All attributes are assumed continuous-valued
  - Assume there exist several possible split values for each attribute
  - May need other tools, such as clustering, to get the possible split values
  - Can be modified for categorical attributes
- Information gain (ID3/C4.5)
  - All attributes are assumed to be categorical
  - Can be modified for continuous-valued attributes

Splitting Criterion: GINI

$$GINI(T) = 1 - \sum_i p_i^2$$

where p_i is the relative frequency of class i in T.
gini(T) is minimized if the classes in T are skewed.

Measure of Impurity

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

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

$$GINI(T) = 1 - \sum_i [p(i \mid T)]^2$$

C_1	0
C_2	6

$P(C_1) = 0/6 = 0 \quad P(C_2) = 6/6 = 1$

$Gini = 1 - P(C_1)^2 - P(C_2)^2 = 1 - 0 - 1 = 0$

C_1	1
C_2	5

$P(C_1) = 1/6 \quad P(C_2) = 5/6$

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

C_1	2
C_2	4

$P(C_1) = 2/6 \quad P(C_2) = 4/6$

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

C_1	3
C_2	3

$P(C_1) = 3/6 \quad P(C_2) = 3/6$

$Gini = 1 - (3/6)^2 - (3/6)^2 = 0.5$

Splitting Based on GINI

- Used in Classification And Regression Tree (CART), Supervised Learning in Quest (SLIQ), Scalable Parallelizable Induction of decision Tree (SPRINT).
- Splitting Criterion: Minimize Gini Index of the Split.
- When a node p is split into k partitions (children), the quality of split is computed as,

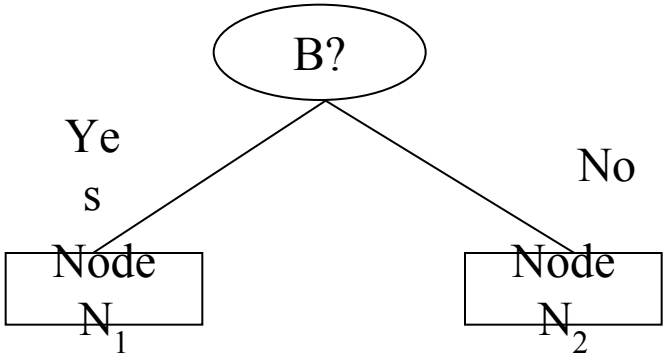
$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

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

The attribute providing smallest $gini_{split}(T)$ is chosen to split the node.

Binary Attributes: Computing GINI Index

- Splits into two partitions
- Effect of Weighing partitions:



	Parent
C_1	6
C_2	6
Gini = 0.500	

Gini(N_1)
= 1 – $(5/6)^2$ – $(2/6)^2$
= 0.194

Gini(N_2)
= 1 – $(1/6)^2$ – $(4/6)^2$
= 0.528

	N_1	N_2
C_1	5	1
C_2	2	4
Gini=0.333		

Gini(Children)
= 7/12 * 0.194 +
5/12 * 0.528
= 0.333

Splitting Based on GINI

- Gini index: Given a training set T, the target attribute takes on k different values (i.e. classes), then the gini index of T is defined as:

$$\text{Gini}(T) = 1 - \sum_{k=1}^{k} p_i^2$$

where p_i is the probability of T belonging to class i, that is, consider a probability distribution P = (p_1, p_2,...,p_k) and $\sum_{i=1}^{k} p_i = 1$

- For instance, if a data set has only one class, its gini index is $1 - 1^2 = 0$, which is a purity data set.
- On the other hand, if a probability distribution is uniform P = (1/k, 1/k,..., 1/k), its gini index achieves maximum.
-

Splitting Based on GINI

- Gini gain: Gini gain is the evaluation criterion for selecting the attributes A which is defined as
- GiniGain(A, T) = Gini(T) – Gini(A, T) = Gini(S) – $\sum_{i=1}^{n} \frac{|T_i|}{|T|} Gini(T_i)$

  where T_i is the partition of T induced by the value of attribute A. Assuming that features A has a continuous range. It has values in the training set, say they are, in increasing, A_1, A_2,...,A_m. Then for each value A_i, i = 1, 2,..., m, partition the records into two sets: the first set has the A values up to and include Ai; the second set has the A values greater than A_i. For each of these partitions, compute the gini gain of (A_i, T), i = 1, 2,..., m, and choose the partition that maximizes the ginigain. If all features are continuous, we will obtain a binary tree.

Splitting Based on GINI

Example: A small training set is shown below. There are 3 attributes which are A_1, A_2, and A_3. A_3 is a continuous attribute. The target set t has 2 classes which are P, N (i.e. Positive & Negative).

Instance	A_1	A_2	A_3	Target Class
1	T	T	1.0	P
2	T	T	6.0	P
3	T	F	5.0	N
4	F	F	4.0	P
5	F	T	7.0	N
6	F	T	3.0	N
7	F	F	8.0	N
8	T	F	7.0	P
9	F	T	5.0	N

Splitting Based on GINI

Solution: The gini index of A_1, A_2 is:

$Gini(t) = 1 - [(4/9)^2 + (5/9)^2] = 40/81$

$Gini(A_1 = T) = 1 - [(3/4)^2 + (1/4)^2] = 3/8$

$Gini(A_1 = F) = 1 - [(1/5)^2 + (4/5)^2] = 8/25$

$Gini(A_2 = T) = 1 - [(2/5)^2 + (3/5)^2] = 12/25$

$Gini(A_2 = F) = 1 - [(2/4)^2 + (2/4)^2] = 1/2$

$GiniGain(A_1) = Gini(t) - [4/9 \times Gini(A_1 = T) + 5/9 \times Gini(A_1 = F)] = 0.149$

$GiniGain(A_2) = Gini(t) - [5/9 \times Gini(A_2 = T) + 4/9 \times Gini(A_2 = F)] = 0.005$

According to the gini gain, the best split between A_1 and A_2 is A_1 due to it has the higher gini gain.

Splitting Based on GINI

Solution: For A_3 which is a continuous attribute, firstly sort all values in increasing order (i.e.1.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0) and then partition each position (i.e. 0.5, 2.0, 3.5,4.5, 5.5, 6.5, 7.5, 8.5) into 2 divisions which becomes a binary tree. The following calculation shows the gini gain on No.3 split position, that is, 3.5.

$$\text{GiniGain(SplitNo.3)} = \text{Gini(t)} - [2/9 \times \text{Gini(SplitNo.3left)} + 7/9 \times \text{Gini(SplitNo.3right)}]$$
$$= 40/81 - \{2/9 \times [1 - (1/2)^2 - (1/2)^2] + 7/9 \times [1 - (3/7)^2 - (4/7)^2]\} = 0.002$$

The results of every possible split are shown in following Table.

Table: Results of continuous attribute A_3

Split No.	1	2	3	4	5	6	7	8
Split Positions		0.5	2.5	3.5	4.5	5.5	6.5	7.5	8.5
GiniGain	0	0.077	0.002	0.049	0.005	0.012	0.049	0

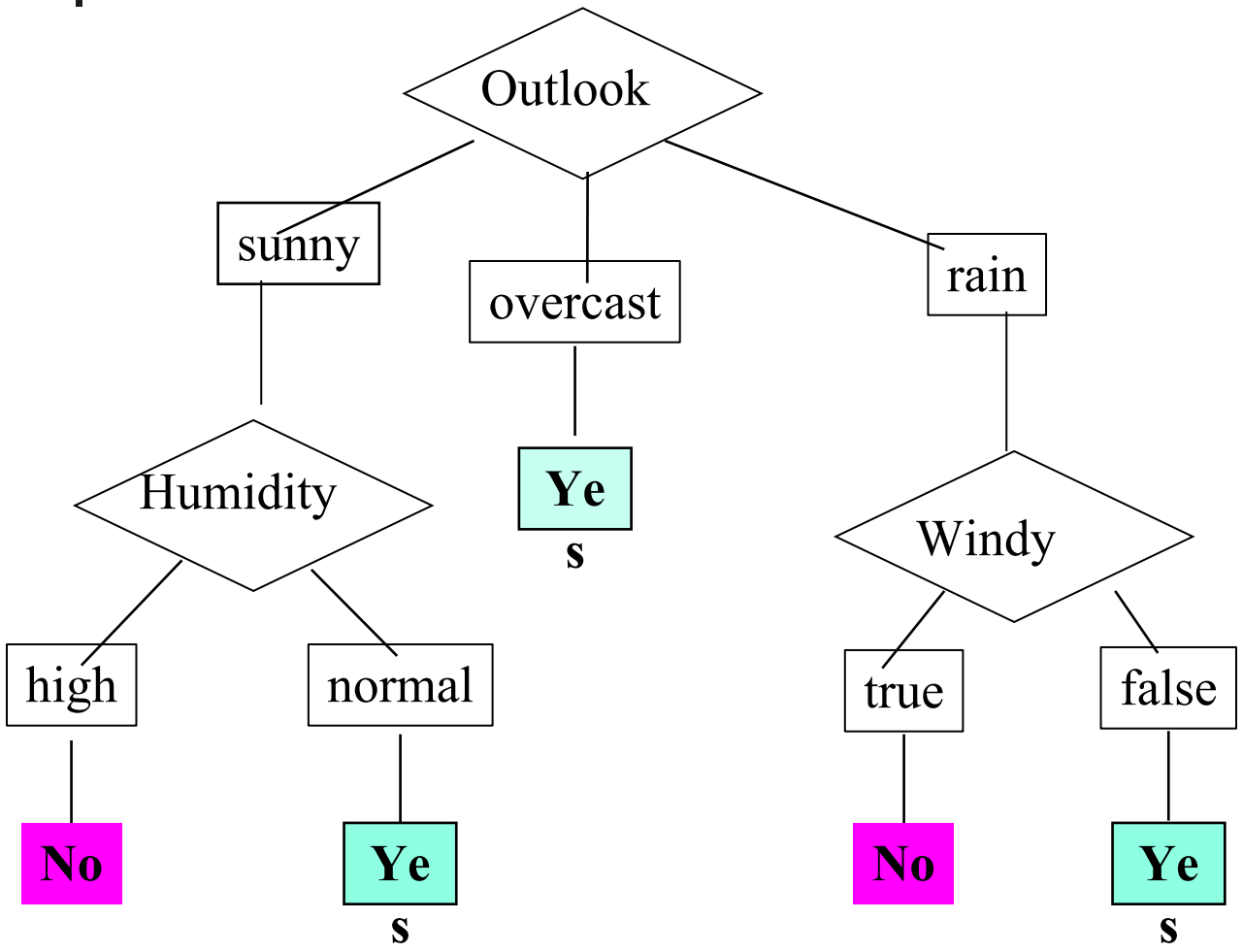
According to the results of A_1, A_2, A_3, the best split is A_1 due to its highest value of gini gain.

Weather Data: Play or not Play?

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

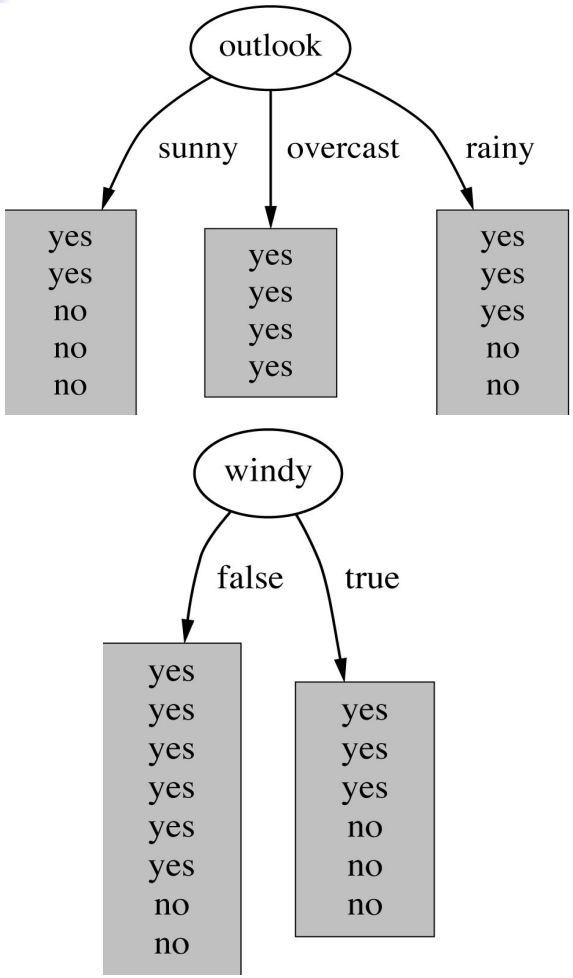
Note:
Outlook is the Forecast, no relation to Microsoft email program

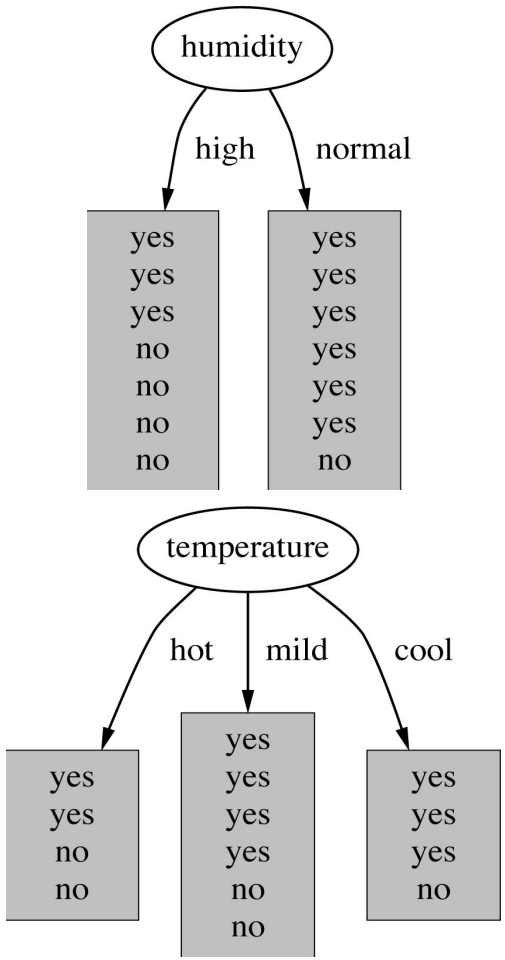
Example Tree for "Play?"



- An internal node is a test on an attribute.
- A branch represents an outcome of the test.
- A leaf node represents a class label or class label distribution.
- At each node, one attribute is chosen to split training examples into distinct classes as much as possible
- A new case is classified by following a matching path to a leaf node.

Which attribute to select?





A criterion for attribute selection

- Which is the best attribute?
  - The one which will result in the smallest tree
  - Heuristic: choose the attribute that produces the "purest" nodes
- Strategy: choose attribute that results in greatest information gain

Splitting Criteria based on INFO

- Entropy at a given node T:

$$Entropy(T) = -\sum_i p_i \log p_i$$

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

Examples for computing Entropy

$$Entropy(T) = -\sum_i p_i \log p_i$$

C_1	**0**
C_2	**6**

$P(C_1) = 0/6 = 0 \quad P(C_2) = 6/6 = 1$

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

C_1	**1**
C_2	**5**

$P(C_1) = 1/6 \quad P(C_2) = 5/6$

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

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

$P(C_1) = 2/6 \quad P(C_2) = 4/6$

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

Splitting Based on INFO

- 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 INFO

- Gain Ratio:

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO} \qquad SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

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
- Designed to overcome the disadvantage of Information Gain

Example: attribute "Outlook"

- "Outlook" = "Sunny":

$$\text{info}([2,3]) = \text{entropy}(2/5,3/5) = -2/5\log(2/5) - 3/5\log(3/5) = 0.971\,\text{bits}$$

- "Outlook" = "Overcast":

$$\text{info}([4,0]) = \text{entropy}(1,0) = -1\log(1) - 0\log(0) = 0\,\text{bits}$$

*Note: log(0) is not defined, but we evaluate 0*log(0) as zero*

- "Outlook" = "Rainy":

$$\text{info}([3,2]) = \text{entropy}(3/5,2/5) = -3/5\log(3/5) - 2/5\log(2/5) = 0.971\,\text{bits}$$

- Expected information for attribute:

$$\text{info}([2,3],[4,0],[3,2]) = (5/14)\times 0.971 + (4/14)\times 0 + (5/14)\times 0.971$$
$$= 0.693\,\text{bits}$$

Computing the information gain

- Information gain:

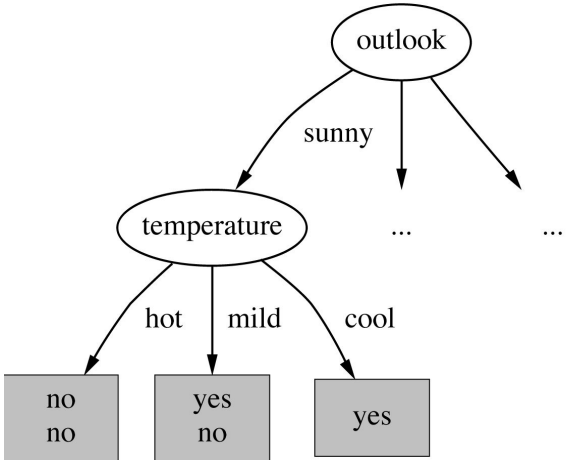
(information before split) – (information after split)

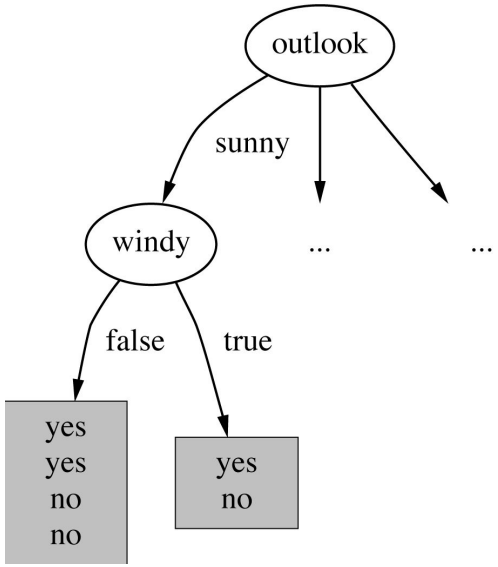
$$gain("Outlook") = info([9,5]) - info([2,3],[4,0],[3,2]) = 0.940 - 0.693$$
$$= 0.247 \text{ bits}$$

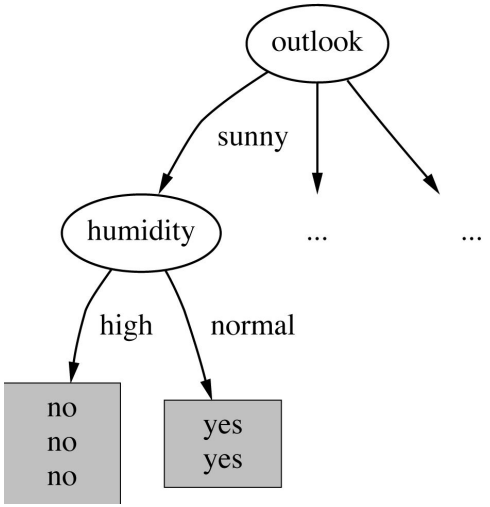
- Information gain for attributes from weather data:

$$gain("Outlook") = 0.247 \text{ bits}$$
$$gain("Temperature") = 0.029 \text{ bits}$$
$$gain("Humidity") = 0.152 \text{ bits}$$
$$gain("Windy") = 0.048 \text{ bits}$$

Continuing to split





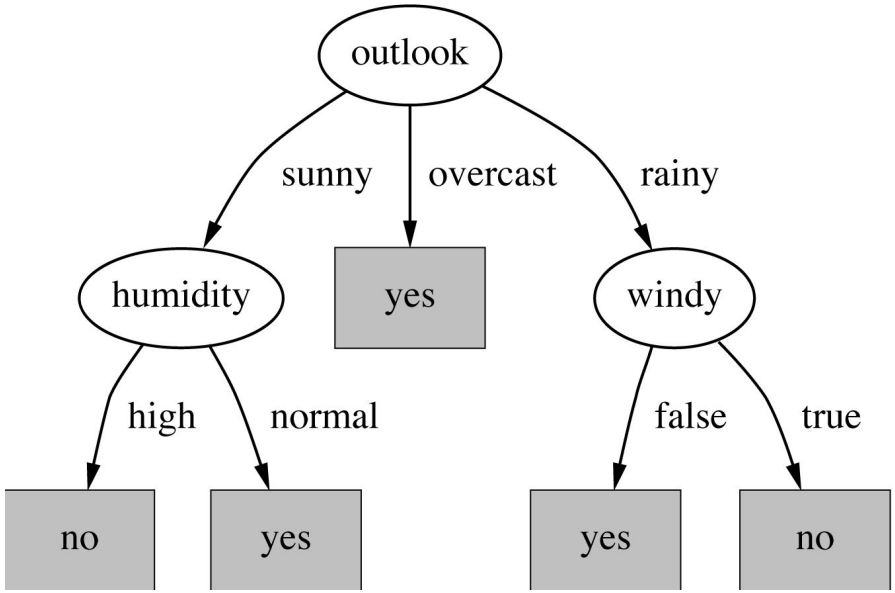


$$\text{gain("Humidity")} = 0.971\,\text{bits}$$

$$\text{gain("Temperature")} = 0.571\,\text{bits}$$

$$\text{gain("Windy")} = 0.020\,\text{bits}$$

The final decision tree



- Note: not all leaves need to be pure; sometimes identical instances have different classes

  ⇒ Splitting stops when data can't be split any further

Attribute Selection: Information Gain

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

$$Info_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$$

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

$\frac{5}{14}I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

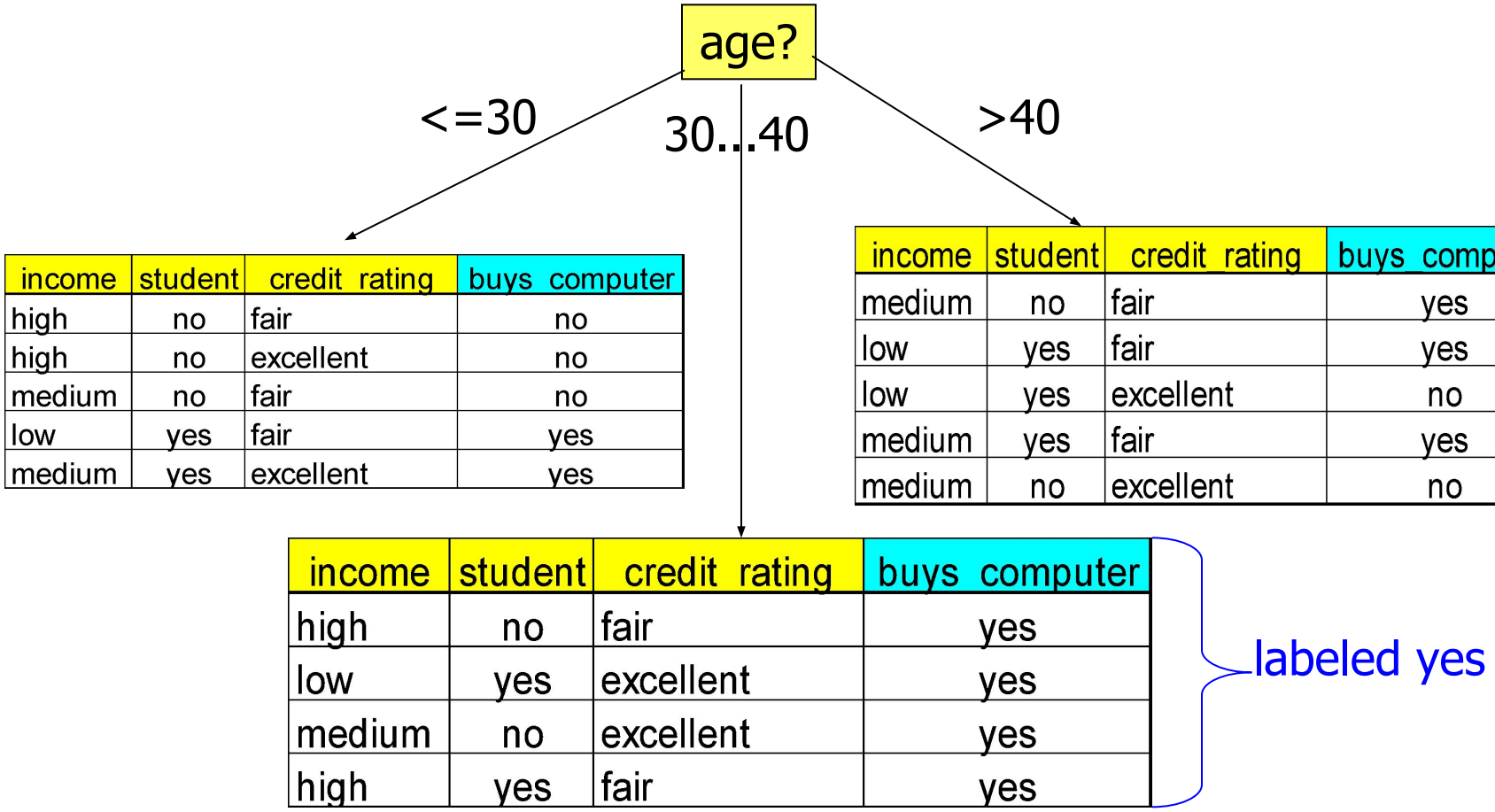
age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

Similarly,

$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$

Splitting the samples using *age*



Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2(\frac{|D_j|}{|D|})$$

  - GainRatio(A) = Gain(A)/SplitInfo(A)
- Ex. $SplitInfo_A(D) = -\frac{4}{14} \times \log_2(\frac{4}{14}) - \frac{6}{14} \times \log_2(\frac{6}{14}) - \frac{4}{14} \times \log_2(\frac{4}{14}) = 0.926$
  - gain_ratio(income) = 0.029/0.926 = 0.031
- The attribute with the maximum gain ratio is selected as the splitting attribute

Example

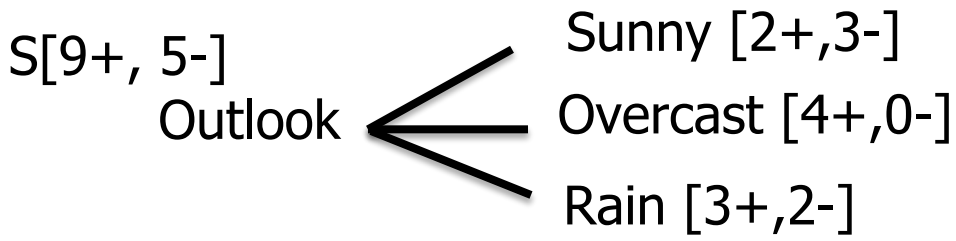
ID	Outlook	Temperature	Humidity	Wind	Play Tennis
1	Sunny	>25	High	Weak	No
2	Sunny	>25	High	Strong	No
3	Overcast	>25	High	Weak	Yes
4	Rain	15-25	High	Weak	Yes
5	Rain	<15	Normal	Weak	Yes
6	Rain	<15	Normal	Strong	No
7	Overcast	<15	Normal	Strong	Yes
8	Sunny	15-25	High	Weak	No
9	Sunny	<15	Normal	Weak	Yes
10	Rain	15-25	Normal	Weak	Yes
11	Sunny	15-25	Normal	Strong	Yes
12	Overcast	15-25	High	Strong	Yes
13	Overcast	>25	Normal	Weak	Yes
14	Rain	15-25	High	Strong	No

Tree induction example

- Entropy of data S

$$Info(S) = -9/14(\log_2(9/14)) - 5/14(\log_2(5/14)) = 0.94$$

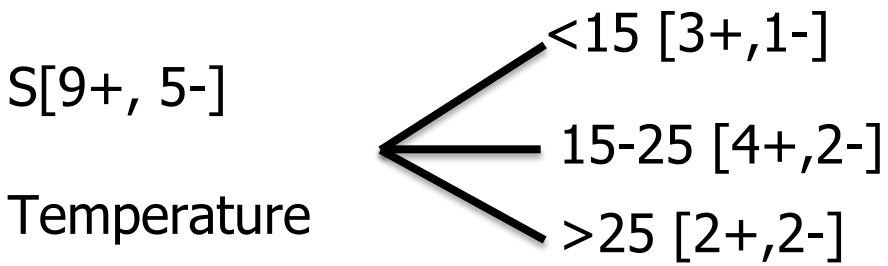
- Split data by attribute Outlook



$$Gain(Outlook) = 0.94 - 5/14[-2/5(\log_2(2/5)) - 3/5(\log_2(3/5))]$$
$$- 4/14[-4/4(\log_2(4/4)) - 0/4(\log_2(0/4))]$$
$$- 5/14[-3/5(\log_2(3/5)) - 2/5(\log_2(2/5))]$$
$$= 0.94 - 0.69 = 0.25$$

Tree induction example

- Split data by attribute Temperature

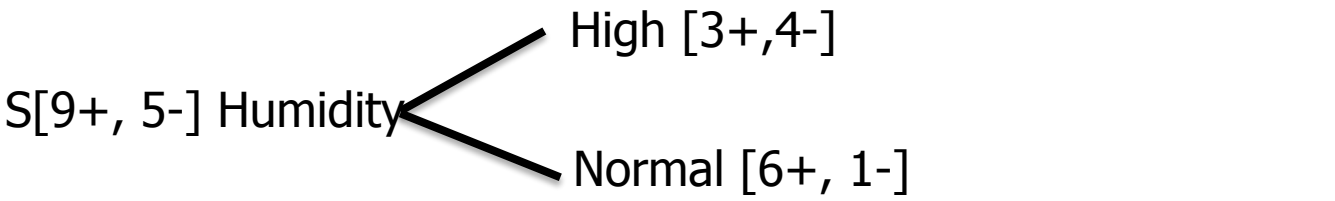


$$\text{Gain(Temperature)} = 0.94 - 4/14[-3/4(\log_2(3/4))-1/4(\log_2(1/4))] - 6/14[-4/6(\log_2(4/6))-2/6(\log_2(2/6))] - 4/14[-2/4(\log_2(2/4))-2/4(\log_2(2/4))]$$

$$= 0.94 - 0.91 = 0.03$$

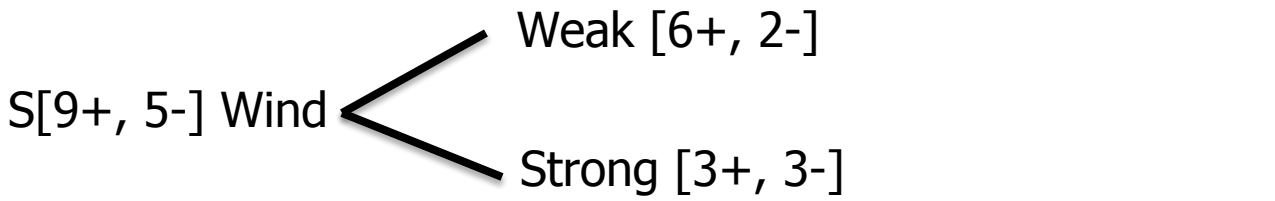
Tree induction example

- Split data by attribute Humidity



$$\text{Gain(Humidity)} = 0.94 - 7/14[-3/7(\log_2(3/7))-4/7(\log_2(4/7))] - 7/14[-6/7(\log_2(6/7))-1/7(\log_2(1/7))]$$
$$= 0.94 - 0.79 = 0.15$$

- Split data by attribute Wind

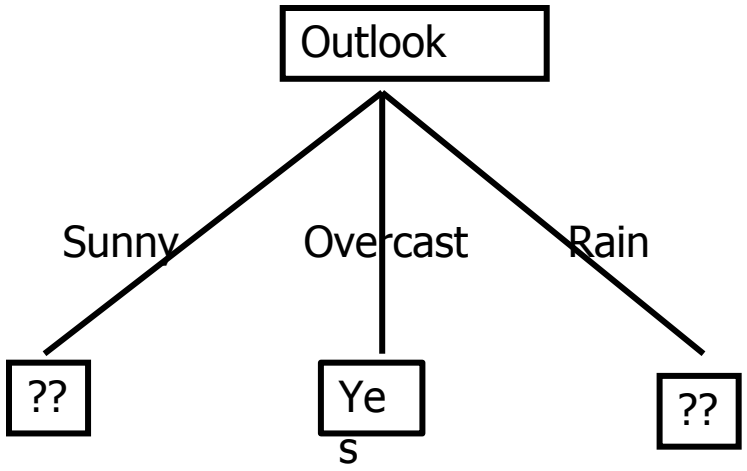


$$\text{Gain(Wind)} = 0.94 - 8/14[-6/8(\log_2(6/8))-2/8(\log_2(2/8))] - 6/14[-3/6(\log_2(3/6))-3/6(\log_2(3/6))]$$
$$= 0.94 - 0.89 = 0.05$$

Tree induction example

Outlook	Temperature	Humidity	Wind	Play Tennis
Sunny	>25	High	Weak	No
Sunny	>25	High	Strong	No
Overcast	>25	High	Weak	Yes
Rain	15-25	High	Weak	Yes
Rain	<15	Normal	Weak	Yes
Rain	<15	Normal	Strong	No
Overcast	<15	Normal	Strong	Yes
Sunny	15-25	High	Weak	No
Sunny	<15	Normal	Weak	Yes
Rain	15-25	Normal	Weak	Yes
Sunny	15-25	Normal	Strong	Yes
Overcast	15-25	High	Strong	Yes
Overcast	>25	Normal	Weak	Yes
Rain	15-25	High	Strong	No

Gain(Outlook) = 0.25
Gain(Temperature)=0.03
Gain(Humidity) = 0.15
Gain(Wind) = 0.05



- Entropy of branch Sunny

$$Info(Sunny) = -2/5(\log_2(2/5))-3/5(\log_2(3/5)) = 0.97$$

- Split Sunny branch by attribute Temperature

Sunny[2+,3-]
Temperature

- <15 [1+,0-]
- 15-25 [1+,1-]
- >25 [0+,2-]

Gain(Temperature)
= 0.97
$- 1/5[-1/1(\log_2(1/1))-0/1(\log_2(0/1))]$
$- 2/5[-1/2(\log_2(1/2))-1/2(\log_2(1/2))]$
$- 2/5[-0/2(\log_2(0/2))-2/2(\log_2(2/2))]$
= 0.97 – 0.4 = 0.57

- Split Sunny branch by attribute Humidity

Sunny[2+,3-]
Humidity

- High [0+,3-]
- Normal [2+, 0-]

Gain(Humidity)
= 0.97
$- 3/5[-0/3(\log_2(0/3))-3/3(\log_2(3/3))]$
$- 2/5[-2/2(\log_2(2/2))-0/2(\log_2(0/2))]$
= 0.97 – 0 = 0.97

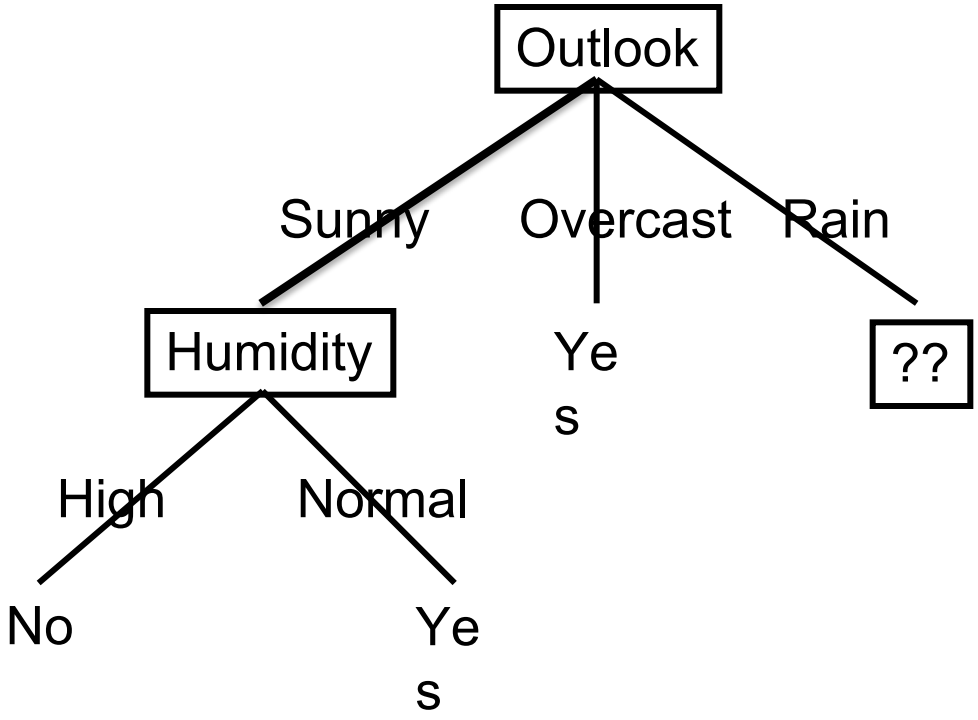
- Split Sunny branch by attribute Wind

Sunny[2+, 3-]
Wind

- Weak [1+, 2-]
- Strong [1+, 1-]

Gain(Wind)
= 0.97
$- 3/5[-1/3(\log_2(1/3))-2/3(\log_2(2/3))]$
$- 2/5[-1/2(\log_2(1/2))-1/2(\log_2(1/2))]$
= 0.97 – 0.95= 0.02

Tree induction example



- Entropy of branch Rain

Info(Rain) = -3/5(log2(3/5))-2/5($\log_2$(2/5)) = 0.97

- Split Rain branch by attribute Temperature

Rain[3+,2-]
Temperatur
e

<15 [1+,1-]

15-25 [2+,1-]

>25 [0+,0-]

Gain(Outlook)
= 0.97
– 2/5[-1/2($\log_2$(1/2))-1/2($\log_2$(1/2))]
– 3/5[-2/3($\log_2$(2/3))-1/3($\log_2$(1/3))]
– 0/5[-0/0($\log_2$(0/0))-0/0($\log_2$(0/0))]
= 0.97 – 0.95 = 0.02

- Split Rain branch by attribute Humidity

Rain[3+,2-]
Humidity

High [1+,1-]

Normal [2+, 1-]

Gain(Humidity)
= 0.97
– 2/5[-1/2($\log_2$(1/2))-1/2($\log_2$(1/2))]
– 3/5[-2/3($\log_2$(2/3))-1/3($\log_2$(1/3))]
= 0.97 – 0.95 = 0.02

- Split Rain branch by attribute Wind

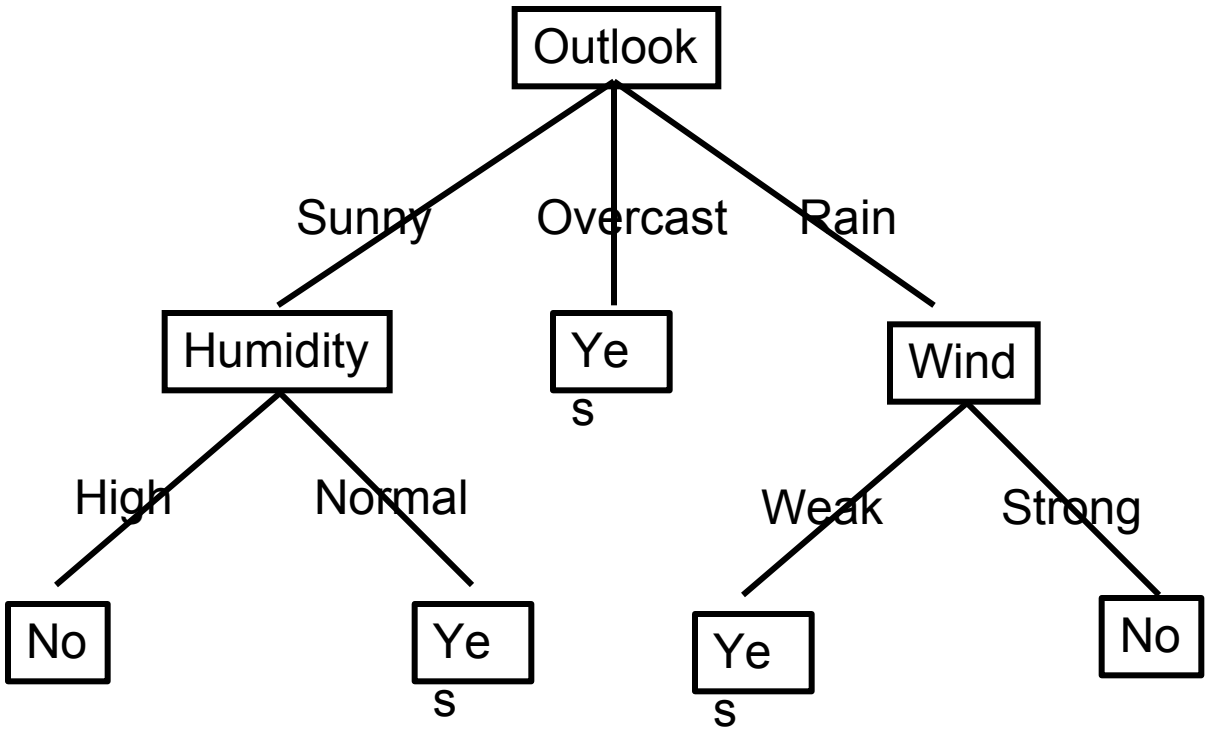
Rain[3+,2-]
Wind

Weak [3+, 0-]

Strong [0+, 2-]

Gain(Wind)
= 0.97
– 3/5[-3/3($\log_2$(3/3))-0/3($\log_2$(0/3))]
– 2/5[-0/2($\log_2$(0/2))-2/2($\log_2$(2/2))]
= 0.97 – 0 = 0.97

Tree induction example



Extracting Classification Rules from Trees

- Represent the knowledge in the form of IF-THEN rules
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction
- The leaf node holds the class prediction
- Rules are easier for humans to understand
- Example

  IF *age* = "<=30" AND *student* = "*no*" THEN *buys_computer* = "*no*"

  IF *age* = "<=30" AND *student* = "*yes*" THEN *buys_computer* = "*yes*"

  IF *age* = "31…40" THEN *buys_computer* = "*yes*"

  IF *age* = ">40" AND *credit_rating* = "*excellent*" THEN *buys_computer* = "*yes*"

  IF *age* = ">40" AND *credit_rating* = "*fair*" THEN *buys_computer* = "*no*"

Comparing Attribute Selection Measures

- The three measures, in general, return good results but
  - Information gain:
    - biased towards multivalued attributes
  - Gain ratio:
    - tends to prefer unbalanced splits in which one partition is much smaller than the others
  - Gini index:
    - biased to multivalued attributes
    - has difficulty when # of classes is large
    - tends to favor tests that result in equal-sized partitions and purity in both partitions

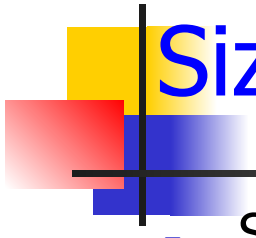
Pros and Cons of decision trees

- Pros
  - + Reasonable training time
  - + Fast application
  - + Easy to interpret
  - + Easy to implement
  - + Can handle large number of features
- Cons
  - – Cannot handle complicated relationship between features
  - – simple decision boundaries
  - – problems with lots of missing data

Overfitting and Tree Pruning

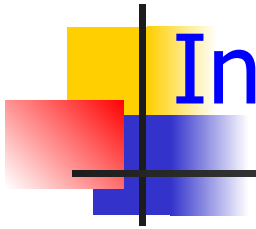
- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Prepruning: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Postpruning: Remove branches from a "fully grown" tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the "best pruned tree"

Approaches to Determine the Final Tree Size



- Separate training (2/3) and testing (1/3) sets
- Use cross validation, e.g., 10-fold cross validation
- Use all the data for training
  - but apply a statistical test (e.g., chi-square) to estimate whether expanding or pruning a node may improve the entire distribution
- Use minimum description length (MDL) principle:
  - halting growth of the tree when the encoding is minimized

Enhancements to Basic Decision Tree Induction



- Allow for continuous-valued attributes
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle missing attribute values
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- Attribute construction
  - Create new attributes based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication

Effect of Rule Simplification

- Rules are no longer mutually exclusive
  - A record may trigger more than one rule
  - Solution?
    - Ordered rule set
    - Unordered rule set – use voting schemes
- Rules are no longer exhaustive
  - A record may not trigger any rules
  - Solution?
    - Use a default class

Ordered Rule Set

- Rules are rank ordered according to their priority
  - An ordered rule set is known as a decision list
- When a test record is presented to the classifier
  - It is assigned to the class label of the highest ranked rule it has triggered
  - If none of the rules fired, it is assigned to the default class

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds
R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes
R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals
R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles
R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
turtle	cold	no	no	sometimes	?

Rule Ordering Schemes

- Rule-based ordering
  - Individual rules are ranked based on their quality
- Class-based ordering
  - Rules that belong to the same class appear together

Rule-based Ordering

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income<80K) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income>80K) ==> Yes

(Refund=No, Marital Status={Married}) ==> No

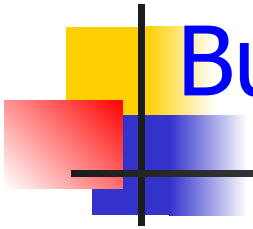
Class-based Ordering

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income<80K) ==> No

(Refund=No, Marital Status={Married}) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income>80K) ==> Yes



Building Classification Rules

- Direct Method:
  - Extract rules directly from data
  - e.g.: RIPPER, CN2, Holte's 1R
- Indirect Method:
  - Extract rules from other classification models (e.g. decision trees, etc).
  - e.g: C4.5 rules

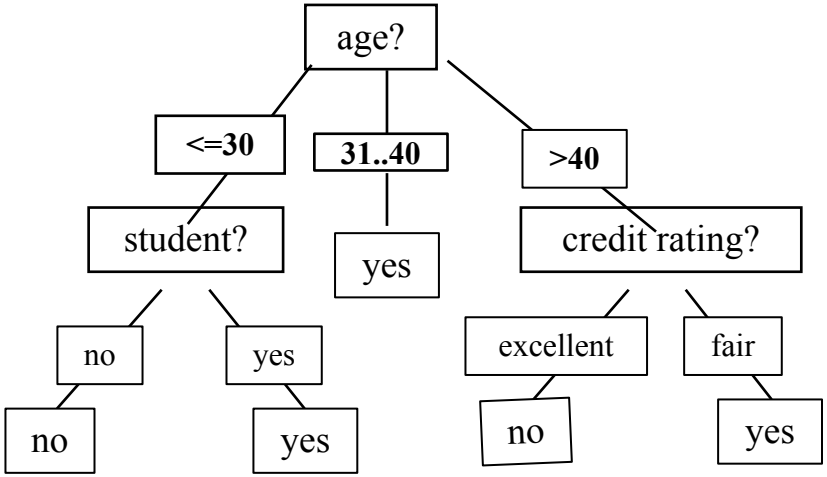
Using IF-THEN Rules for Classification

- Represent the knowledge in the form of IF-THEN rules
  R: IF *age* = youth AND *student* = yes THEN *buys_computer* = yes
  Rule antecedent/precondition vs. rule consequent
- Assessment of a rule: *coverage* and *accuracy*
  - n_{covers} = # of tuples covered by R
  - $n_{correct}$ = # of tuples correctly classified by R

  coverage(R) = n_{covers} /|D| /* D: training data set */
  accuracy(R) = $n_{correct}$ / n_{covers}
- If more than one rule are triggered, need **conflict resolution**
  - Size ordering: assign the highest priority to the triggering rules that has the "toughest" requirement (i.e., with the *most attribute tests*)
  - Class-based ordering: decreasing order of *prevalence or misclassification cost per class*
  - Rule-based ordering (**decision list**): rules are organized into one long priority list, according to some measure of rule quality or by experts

Rule Extraction from a Decision Tree

- Rules are *easier to understand* than large trees
- One rule is created *for each path* from the root to a leaf
- Each attribute-value pair along a path forms a conjunction: the leaf holds the class prediction
- Rules are mutually exclusive and exhaustive

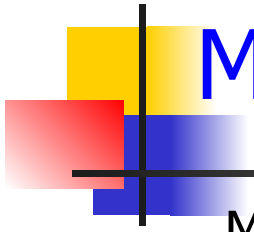


- Example: Rule extraction from our *buys_computer* decision-tree

IF *age* = young AND *student* = *no* THEN *buys_computer* = *no*
IF *age* = young AND *student* = *yes* THEN *buys_computer* = *yes*
IF *age* = mid-age THEN *buys_computer* = *yes*
IF *age* = old AND *credit_rating* = *excellent* THEN *buys_computer* = *no*
IF *age* = old AND *credit_rating* = *fair* THEN *buys_computer* = *yes*

Rule Induction: Sequential Covering Method

- Sequential covering algorithm: Extracts rules directly from training data
- Typical sequential covering algorithms: FOIL, AQ, CN2, RIPPER
- Rules are learned *sequentially*, each for a given class C_i will cover many tuples of C_i but none (or few) of the tuples of other classes
- Steps:
  - Rules are learned one at a time
  - Each time a rule is learned, the tuples covered by the rules are removed
  - Repeat the process on the remaining tuples until *termination condition*, e.g., when no more training examples or when the quality of a rule returned is below a user-specified threshold
- Comp. w. decision-tree induction: learning a set of rules *simultaneously*



Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

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)

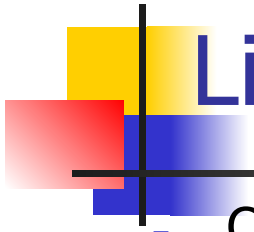
d: TN (true negative)

Metrics for Performance Evaluation…

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

- Most widely-used metric:

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



Limitation of Accuracy

- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10

- If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %
  - Accuracy is misleading because model does not detect any class 1 example

Cost Matrix

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

Computing Cost of Classification

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

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

Accuracy = 80%

Cost = 3910

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

Accuracy = 90%

Cost = 4255

Cost vs Accuracy

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

Cost	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	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

$N = a + b + c + d$

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

$\text{Cost} = p\,(a + d) + q\,(b + c)$

$= p\,(a + d) + q\,(N - a - d)$

$= q\,N - (q - p)(a + d)$

$= N\,[q - (q-p) \times \text{Accuracy}]$

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 is biased towards C(Yes|Yes) & C(No|Yes)
- F-measure is biased towards all except C(No|No)

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

Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

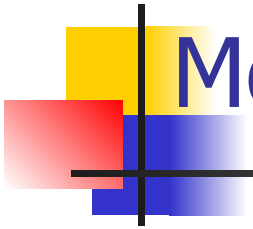
Methods for Performance Evaluation

- 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
- Random subsampling
  - Repeated holdout
- Cross validation
  - Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Leave-one-out: k=n
- Stratified sampling
  - oversampling vs undersampling
- Bootstrap
  - Sampling with replacement



Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?