Chi-Square Test

(based on Quinlan, Induction of Decision Trees, 1986)

- Notation
  - A - splitting (branching) attribute (e.g., Test)
  - v - domain size of attribute A (3: 0-600, 600-700, 700+)
  - C_i - subset of records containing value i of attribute A (Test < 600: 2)
  - c – number of classes (3: low, medium, high)
  - e_j - number of records belonging to class j in the entire data set C (*Low* = 1, *Medium* = 2, *High* = 2)
  - o_{ij} - number of records belonging to class j in subset C_i (Test < 600, GPA = Low : 1)
  - α - significance level

Chi-Square Test (cont.)

- **Null Hypothesis**: attribute *A* is <u>irrelevant</u> to classifying the records in data set *C*
- **Alternative Hypothesis**: attribute *A* <u>affects</u> the class distribution in data set *C*
- Expected number of class *j* records in C_i:

$$e'_{ij} = \frac{e_j}{\sum_{j=1}^{c} e_j} \sum_{j=1}^{c} o_{ij}$$

Statistic:

$$\sum_{j=1}^{c} \sum_{i=1}^{v} \frac{(o_{ij} - e'_{ij})^2}{e'_{ij}} \sim \chi_{\alpha}^{2}((v-1)(c-1))$$

Chi-Square Test – Student Example

$$e'_{ij} = \frac{e_j}{\sum_{j=1}^{c} e_j} \sum_{j=1}^{c} o_{ij}$$

- Entire data set (before splitting): e_{Low} = 1, e_{Medium} = 2, e_{High} = 2
- Splitting by *Test*

$$\sum_{j=1}^{c} o_{ij} = 2$$

  - Test < 600: *Low = 1, Medium = 1*
    - e'_{low} = (1/5)*2 = 0.4, e'_{medium} = (2/5)*2 = 0.8. e'_{high} = (2/5)*2 = 0.8.

$$\sum_{j=1}^{c} o_{ij} = 2$$

  - Test = 600-700: *Medium = 1, High = 1*
    - e'_{low} = (1/5)*2 = 0.4, e'_{medium} = (2/5)*2 = 0.8. e'_{high} = (2/5)*2 = 0.8.

$$\sum_{j=1}^{c} o_{ij} = 1$$

  - Test > 700: *High = 1*

Chi-Square Test – Student Example (cont.)

		Test Grade (i)			Total	p_j
GPA (j)		0-600	600-700	Over 700		
Low	Actual	1	0	0	1	0.2
	Expected	**0.4**	**0.4**	**0.2**	1	
	Statistic	**0.9**	**0.4**	**0.2**	**1.5**	
Medium	Actual	1	1	0	2	0.4
	Expected	**0.8**	**0.8**	**0.4**	2	
	Statistic	**0.05**	**0.05**	**0.4**	**0.5**	
High	Actual	0	1	1	2	0.4
	Expected	**0.8**	**0.8**	**0.4**	2	
	Statistic	**0.8**	**0.05**	**0.9**	**1.75**	
Total		2	2	1	**3.75**	5

Statistic:

$$\sum_{j=1}^{c} \sum_{i=1}^{v} \frac{(o_{ij} - e'_{ij})^2}{e'_{ij}} = 3.75$$

$$\chi^2_{0.05}(4) = 9.49$$

Conclusion: do not split the node on *Test Grade*

Pessimistic Error Pruning (PEP)

- Uses training set to estimate error on new data
- Error estimate (relative frequency with continuity correction)
    - probability of error (apparent error rate)
    - where
        - N = #examples
        - n_C = #examples in majority class

$$q = \frac{N - n_C + 0.5}{N}$$

5

Pessimistic Error Pruning (cont.)

- Error of a node *v* (if pruned)
  - where
    - N_v = #examples at node *v*
    - $n_{C,v}$ = #examples in majority class at node *v*

$$q(v) = \frac{N_v - n_{C,v} + 0.5}{N_v}$$

- Error of a subtree *T*
  - Where
    - l = leaf node of sub-tree *T*

$$q(T) = \frac{\sum_{l \in leafs(T)} (N_l - N_{C,l} + 0.5)}{\sum_{l \in leafs(T)} N_l}$$

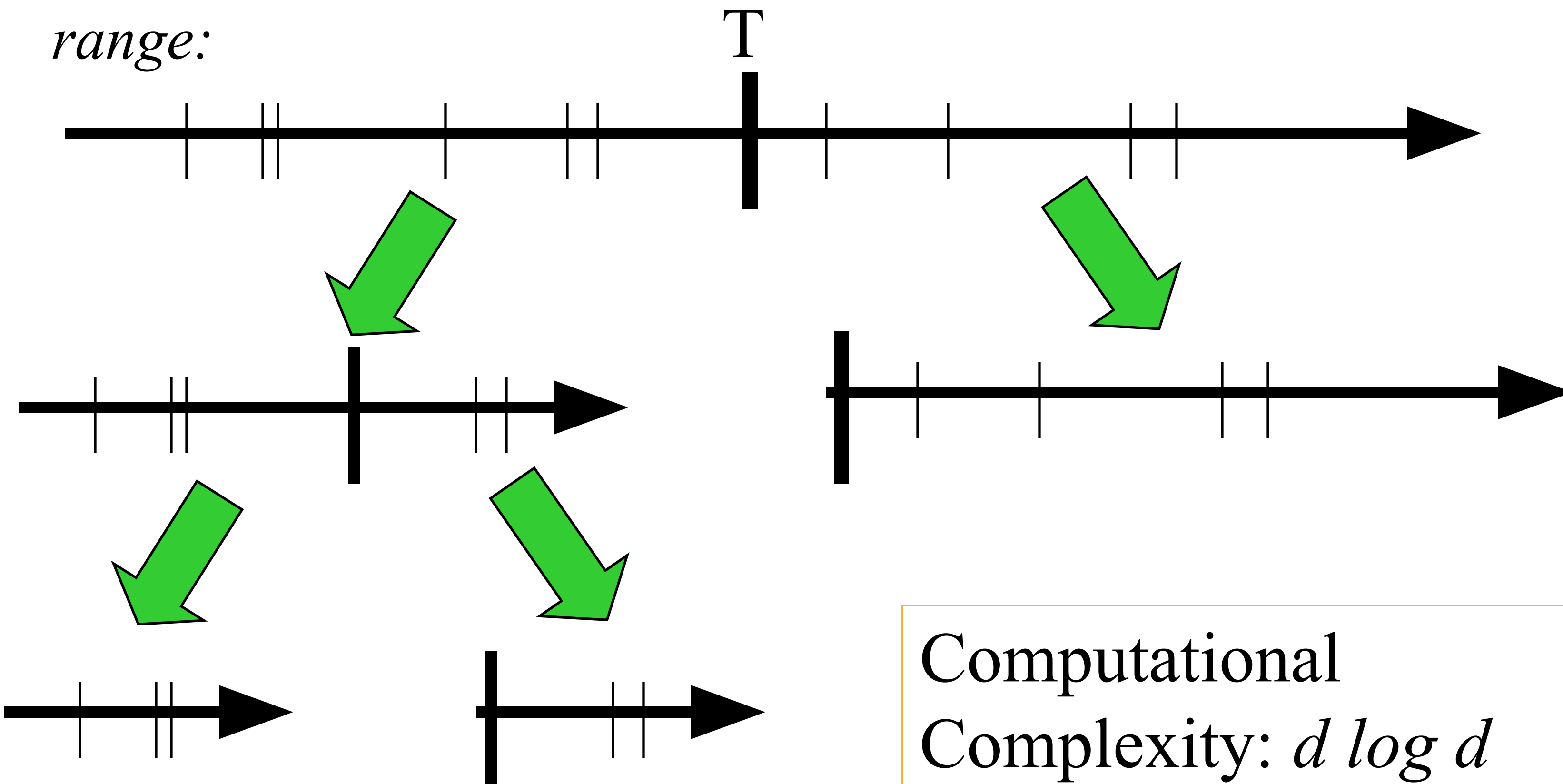
- Prune if $q(v) \leq q(T)$
- Prunes in bottom-up fashion
  - fast
  - considered a weakness (on accuracy)

Lecture No. 6 – Decision Tree Learning II

- Discretization of Continuous Attributes
- Alternative Splitting Rules
  - Information Gain Ratio
  - Gini Index
  - Twoing
- CART Overview
- Comparison of Decision Trees

Discretization Algorithm

Attribute range:



Computational Complexity: *d log d*

d – number of distinct values

Discretization Algorithm (continued)

- Notation
  - S - entire set of instances
  - A - attribute (feature)
  - T - threshold (partition boundary)
  - S_1 - set of instances below the threshold ($v \leq T$)
  - S_2 - set of instances above the threshold ($v > T$)

Discretization Algorithm (continued)

- Entropy induced by *T*:

$$E(A,T;S)=\frac{|S_1|}{|S|}Ent(S_1)+\frac{|S_2|}{|S|}Ent(S_2)$$

- Information Gain:
  - *Gain (A,T;S) = Ent (S) - E(A,T;S)*
- Example:

Record	1	2	3	4	5
Value	1	1.5	1.5	1.7	2.1
Class	0	1	0	1	1

Discretization Example

Record	1	2	3	4	5
Value	1	1.5	1.5	1.7	2.1
Class	0	1	0	1	1

Value <=	Pos (0)	Neg (1)	Total		Prob (0)	Prob (1)
1	1	0	1		1.00	0.00
1.5	2	1	3		0.67	0.33
1.7	2	2	4		0.50	0.50
2.1	2	3	5		0.40	0.60
Value >	**Pos (0)**	**Neg (1)**	**Total**		**Prob (0)**	**Prob (1)**
1	1	3	4		0.25	0.75
1.5	0	2	2		0.00	1.00
1.7	0	1	1		0.00	1.00
2.1	0	0	0			

Value <=	plogp	plogp	Total		Entropy	Info Gain
1	0.000		0.000		0.649	0.322
1.5	0.390	0.528	0.918		0.551	**0.420**
1.7	0.500	0.500	1.000		0.800	0.171
2.1	0.529	0.442	0.971		0.971	
Value >	**plogp**	**plogp**				
1	0.500	0.311	0.811			
1.5		0.000	0.000			
1.7		0.000	0.000			
2.1						

The best threshold

Lecture No. 6 – Decision Tree Learning II

- Rule Extraction
- Discretization of Continuous Attributes
- Alternative Splitting Rules
  - Information Gain Ratio
  - Gini Index
  - Twoing
- CART Overview
- Comparison of Decision Trees

Splitting Rules

- Possible splitting functions (rules)
  - Entropy (Information Gain and Gain Ratio)
  - Twoing
  - Gini Index

Information Gain Ratio

- ID3 selects the attribute which maximizes the mutual information (information gain):
  - $gain(A) = I(p, n) - E(A)$
    - $I(p, n)$ – unconditional entropy (does not depend on the choice of A)
    - $E(A)$ - conditional entropy after splitting the root node by the test attribute A
- The information gain is maximal when $E(A)$ is equal to zero
- $E(A) = 0$ if for each value of A
  - Either all examples are positive
  - Or all examples are negative

Gain Ratio (cont.)

- The problem with multi-valued and continuous attributes in noisy databases
  - The probability of a subset of examples to have the same class increases monotonically with a decrease in the subset size
    - The extreme case is a subset of one example
  - The average size of a subset decreases with an increase in the total number of attribute values (e.g., attribute Date)
- Conclusion
  - Information gain is biased towards multi-valued and continuous attributes

Gain Ratio (cont.)

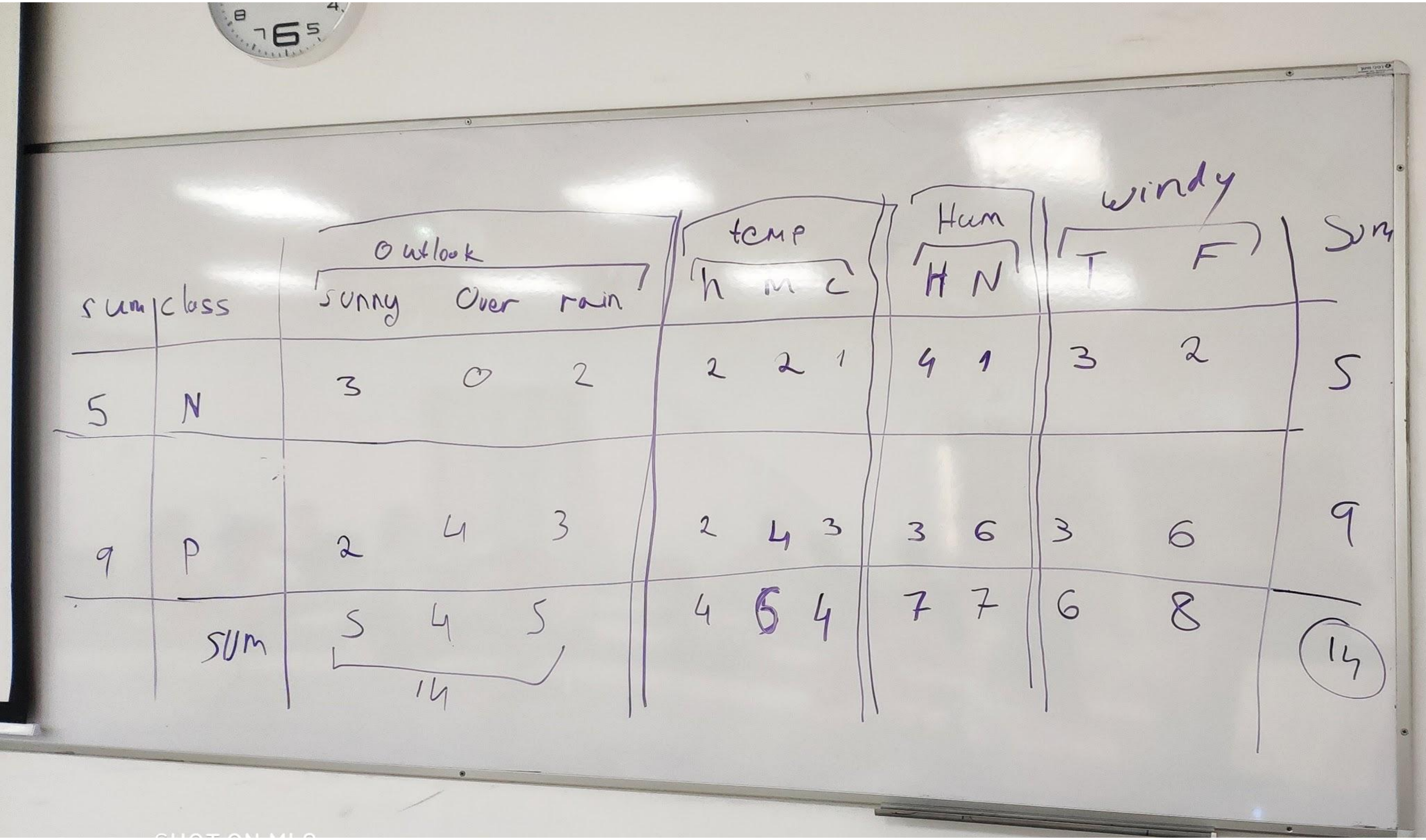
- The Gain Ratio Approach
  - “Punish” the multi-valued attributes via dividing (normalizing) their information gain by the *Split Information:*

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

  - The *Split Information* represents the *entropy* of the tested attribute (in contrast to the entropy of the target attribute)
  - The Gain Ratio: *Gain(A)/SplitInfo(A)*

Training Set

Outlook	Tempreature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N



sum	class	Outlook: sunny	Outlook: Over	Outlook: rain	temp: h	temp: m	temp: c	Hum: H	Hum: N	windy: T	windy: F	Sum
5	N	3	0	2	2	2	1	4	1	3	2	5
9	P	2	4	3	2	4	3	3	6	3	6	9
	sum	5	4	5	4	6	4	7	7	6	8	14

14

Gain ratios for weather data

Outlook		Temperature	
Info:	0.693	Info:	0.911
Gain: 0.940-0.693	0.247	Gain: 0.940-0.911	0.029
Split info: info([5,4,5])	1.577	Split info: info([4,6,4])	1.362
Gain ratio: 0.247/1.577	0.156	Gain ratio: 0.029/1.362	0.021

Humidity		Windy	
Info:	0.788	Info:	0.892
Gain: 0.940-0.788	0.152	Gain: 0.940-0.892	0.048
Split info: info([7,7])	1.000	Split info: info([8,6])	0.985
Gain ratio: 0.152/1	0.152	Gain ratio: 0.048/0.985	0.049

Gain Ratio – Student Example

שם פרטי	שם משפחה	מגדר	מקום לידה	ציון פסיכומטרי	ממוצע ציונים
First Name	**Last Name**	**Gender**	**Place of Birth**	**Test Grade**	**GPA**
David	Cohen	M	USA	Over 700	High
Ophir	Levy	M	Israel	600-700	Medium
Sharon	Grosman	F	Israel	600-700	High
Diana	Liberman	F	Russia	0-600	Medium
Anat	Klein	F	Israel	0-600	Low

Let us assume that the attribute "Place of Birth" has <u>three</u> possible values: USA, Israel, and Russia

Information Gain – *Place of Birth*

	Place of Birth			**Total**
	Israel	USA	Russia	
Low	1	0	0	1
p	0.333	0.000	0.000	
-logp	1.585	0.000	0.000	
Medium	1	0	1	2
p	0.333	0.000	1.000	
-logp	1.585	0.000	0.000	
High	1	1	0	2
p	0.333	1.000	0.000	
-logp	1.585	0.000	0.000	
Total	3	1	1	5
p	0.60	0.20	0.20	0.80
Entropy	1.585	0.000	0.000	**0.951**
Gain				**0.571**

Split Information and Gain Ratio

Student Example

	Place of Birth			**Total**
	Israel	USA	Russia	
Total	3	1	1	5
p	0.60	0.20	0.20	1.00
-logp	0.737	2.322	2.322	1.371
Gain				0.571
Gain Ratio				**0.416**

- Split Information

  Information Gain

  Information Gain Ratio

Gain Ratio (Test Grade) = **0.474**

Gain Ratio (Gender) = 0.176

Gain Ratio (Place of Birth) = 0.416

Gini index (CART™, 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
-

Reminder

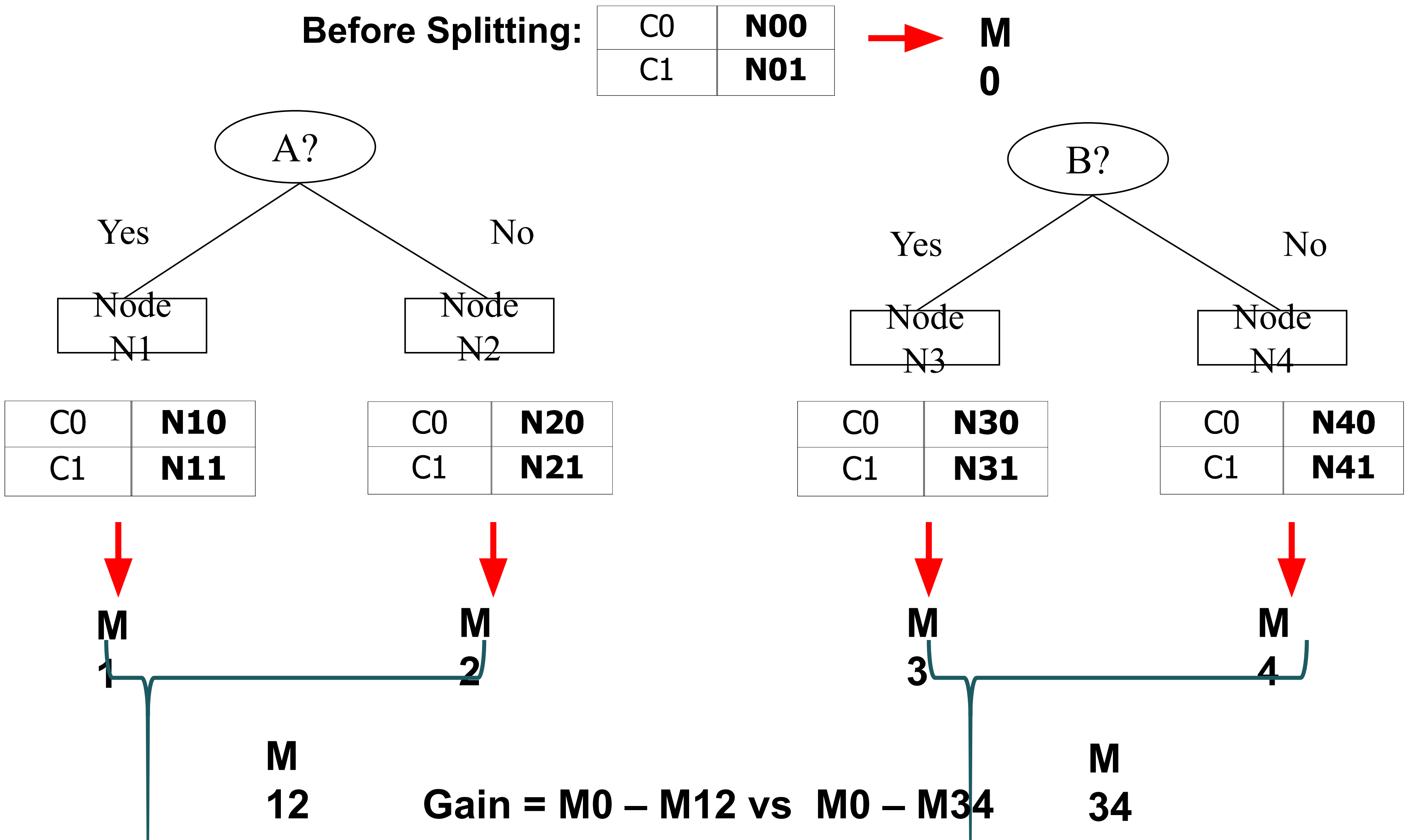
Impurity functions have to

1) achieve a maximum at the uniform distribution

2) achieve a minimum when $p_j = 1$

3) be symmetric with regard to their permutations.

How to Find the Best Split



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, implying least interesting information
- 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

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

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,

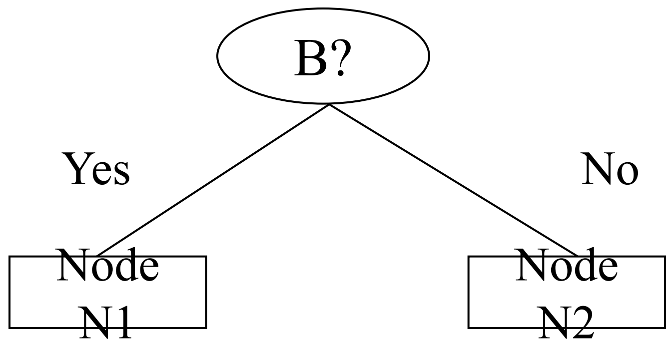
$$\boxed{GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)}$$

where, n_i = number of records at child i,

n = number of records at node p.

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)
$= 1 – (5/6)^2 – (2/6)^2$
$= 0.194$

Gini(N2)
$= 1 – (1/6)^2 – (4/6)^2$
$= 0.528$

	N1	**N2**
C1	**5**	**1**
C2	**2**	**4**
Gini=0.333		

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

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)

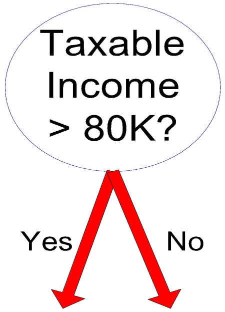
	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

	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 ≥ 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

Sorted Values

Split Positions

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	

- If a data set D contains examples from n classes, gini index, $gini(D)$ is defined as:

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

where p_j is the relative frequency of class j in D

- If a data set D is split on A into two subsets D_1 and D_2, the *gini* index *gini*(D) is defined as

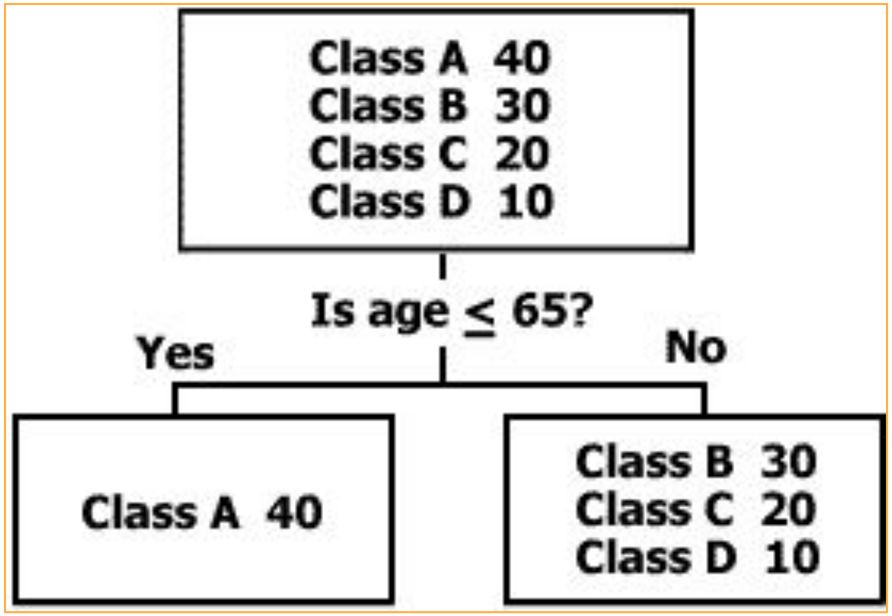
$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- Reduction in Impurity: $\Delta gini(A) = gini(D) - gini_A(D)$

6

Gini Splitting Rule

- Looks for the largest class in the data set and strives to isolate it from all other classes



Gini Index

- If a data set *T* contains examples from *n* classes, gini index, *gini(T)* is de

$$gini(T) = 1 - \sum_{j=1}^{n} p_j^2$$

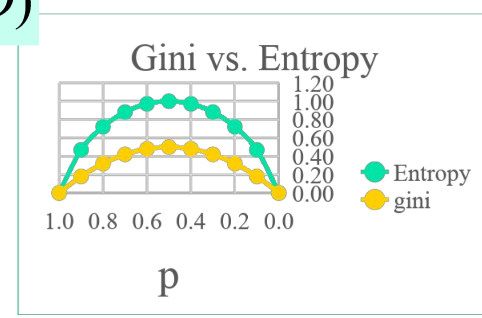
    - where p_j is the relative frequency of class *j* in *T*.

- If a data set *T* is split into two subsets T_1 and T_2 with sizes N_1 and N_2 respectively, the gini index of the split data contains examples from *n* classes, the gini index *gini(T)* is defined as

$$gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- Reduction in Impurity:
- The attribute provides the smallest $gini_{split}(T)$ (or the largest reduction in impurity) is chosen to split the node (need to enumerate all possible splitting points for



Twoing Rule

- Another splitting method is the **Twoing Rule**. This approach does not have anything to do with the impurity function.

- The intuition here is that the class distributions in the two child nodes should be as different as possible and the proportion of data falling into either of the child nodes should be balanced.

- The twoing rule: … s that maximizes:

$$\frac{p_L p_R}{4}\left[\sum_j |p(j|t_L) - p(j|t_R)|\right]^2$$

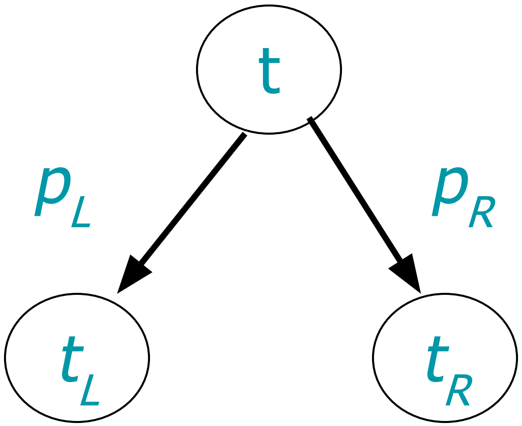
Twoing Splitting Rule (CART™)

Source: L. Breiman, J. Friedman, R. Olshen, and C. Stone (1984), *Classification and Regression Trees*, Pacific Grove: Wadsworth

- Maximize

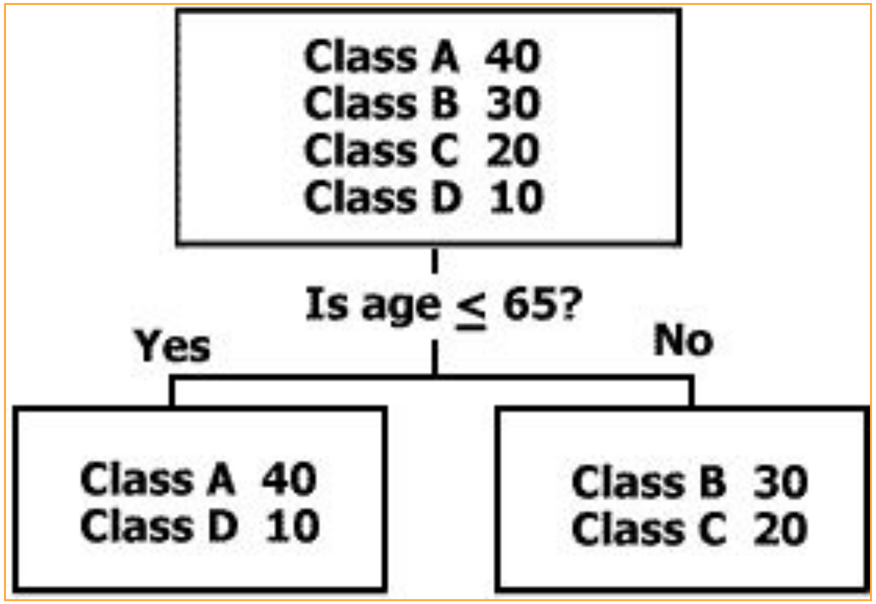
$$\boxed{\frac{p_L p_R}{4}\left[\sum_j \left| p(j/t_L) - p(j/t_R) \right|\right]^2}$$

- Notation
  - p_L – proportion of cases going to the left node
  - p_R – proportion of cases going to the right node
  - j – class index
  - $p\ (j/t_L)$ – probability of class j at the left node
  - $p\ (j/t_R)$ – probability of class j at the right node
- Attempts to find groups of up to 50% of the data each
- If impossible - power-modified twoing

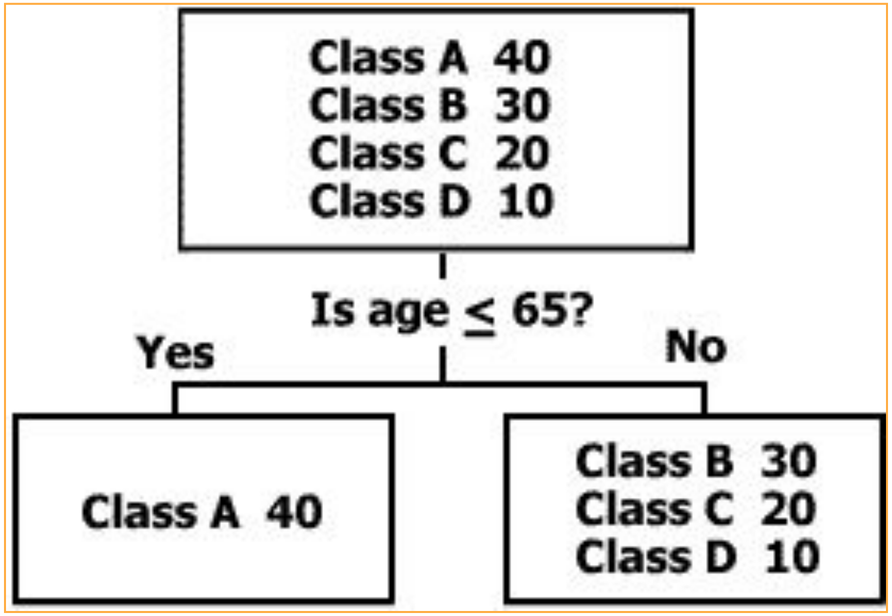


Twoing Splitting Rule - Example

Split 1



Split 2



Which split is better?

Twoing Splitting Rule - Example

$$\frac{p_L p_R}{4}\left[\sum_j \left| p(j/t_L) - p(j/t_R) \right| \right]^2$$

Split 1

Class:	A	B	C	D	Total	Pl/Pr
Age <= 65	40	0	0	10	50	0.5
Age > 65	0	30	20	0	50	0.5
Total					**100**	

Better split

Split 2

Class:	A	B	C	D	Total	Pl/Pr
Age <= 65	40	0	0	0	40	0.4
Age > 65	0	30	20	10	60	0.6
Total					**100**	

Prob			
A	B	C	D
0.80	0.00	0.00	0.20
0.00	0.60	0.40	0.00

Abs						
A	B	C	D	Total		Twoing
0.800	0.600	0.400	0.200	**2.000**		**0.250**

Abs						
A	B	C	D	Total		Twoing
1.000	0.500	0.333	0.167	**2.000**		0.240

Prob			
A	B	C	D
1.00	0.00	0.00	0.00
0.00	0.50	0.33	0.17

Using Splitting Rules (Gini, Twoing, Entropy)

- Gini -- is usually best for yes/no outcomes
- Twoing - similar to entropy but more flexible because it has a tuning parameter
  - excellent for multi-class outcomes
  - twoing excellent for hard to classify problems
    - problems where accuracy for all methods will be low
    - inherently difficult problems or low signal/noise ratio
- Entropy- popular in Machine Learning literature

Lecture No. 6 – Decision Tree Learning II

- Rule Extraction
- Discretization of Continuous Attributes
- Alternative Splitting Rules
  - Information Gain Ratio
  - Gini Index
  - Twoing
- CART Overview
- Comparison of Decision Trees

CART™ Algorithm
Main Steps

- Grow the maximal tree based on the entire data set
  - A binary splitting procedure
  - Splitting rules
  - Stopping criteria
- Derive a set of pruned sub-trees
  - Create “efficiency frontier”
- Select the best tree by using validation set or cross-validation

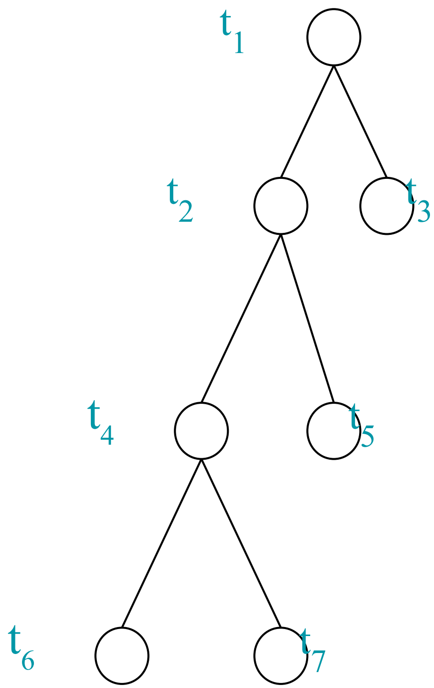
CART™ : Binary Splitting Procedure

- Continuous (Ordinal) Attributes
  - Each distinct value is considered for threshold
  - Branching rule: $x \le C$
  - M possible splits (M - number of distinct values)
- Nominal (Categorical) Attributes
  - The branching rule is determined separately for each possible value
  - $2^{M-1} - 1$ possible splits (M - number of values)

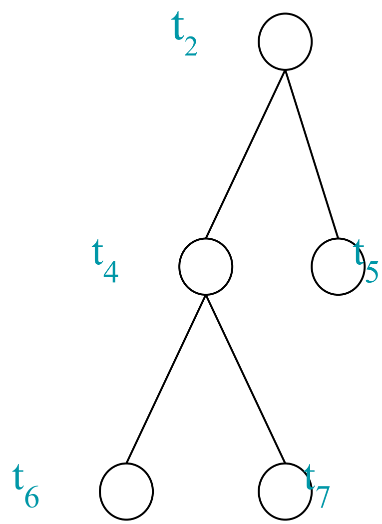
CART™ : Stopping Criteria

- Splitting is impossible
  - One case left in a node
  - All the cases in the node have the same target value
- Other reasons
  - Too few cases in the node (default = 10 cases)

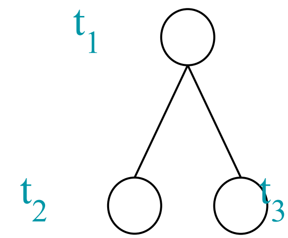
Pruning Trees



Tree T



Branch
T_{t2}



Sub-tree
$T - T_{t2}$

Deriving a set of pruned sub-trees

- Objective: minimizing the cost-complexity function

$$R_\alpha(T) = R(T) + \alpha \cdot \left|\widetilde{T}\right|$$

  - *T* - a tree
  - *R (T)* - the training error rate of a tree
  - R_α *(T)* - the cost-complexity of a tree
  - $\left|\widetilde{T}\right|$ - number of terminal nodes in a tree
  - *α* - complexity parameter (real number, greater than zero)

CART™ Pruning Algorithm

$$\boxed{R_\alpha(T) = R(T) + \alpha \cdot \left|\widetilde{T}\right|}$$

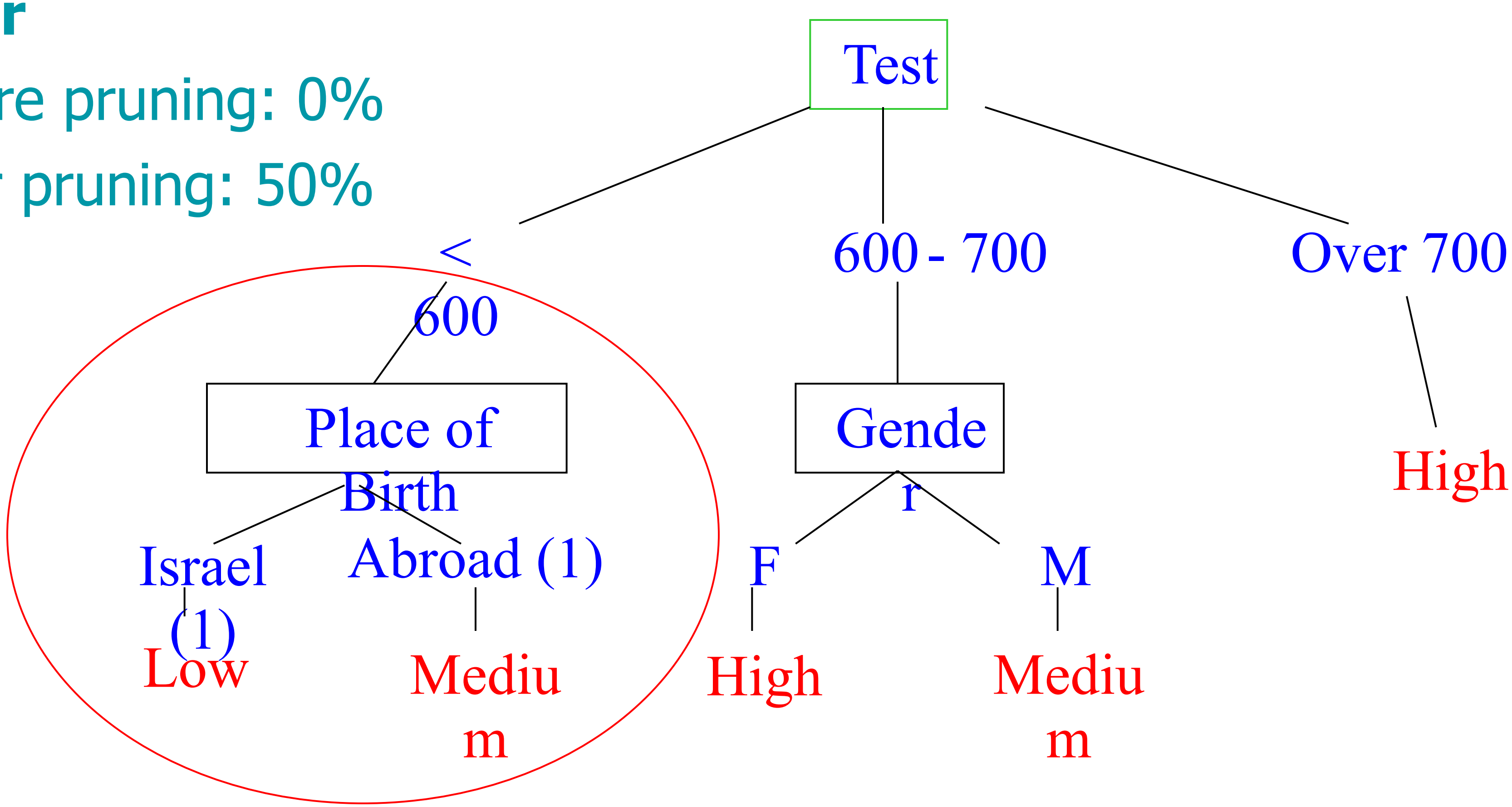
- Step 1 - Initialize the list of optimal trees with the maximal tree
- Step 2 - Initialize α = 0
- Step 3 - Increase α until the tree ceases to be optimal
- Step 4 - Find a new sub-tree, which is optimal with the new value of α
- Step 5 - Add the new sub-tree to the list of optimal trees.
- Step 6 - If the new sub-tree has more than one terminal node, go to Step 3. Otherwise, stop.

CART™ Student Example Maximal Tree ($\alpha = 0$)

Error

Before pruning: 0%

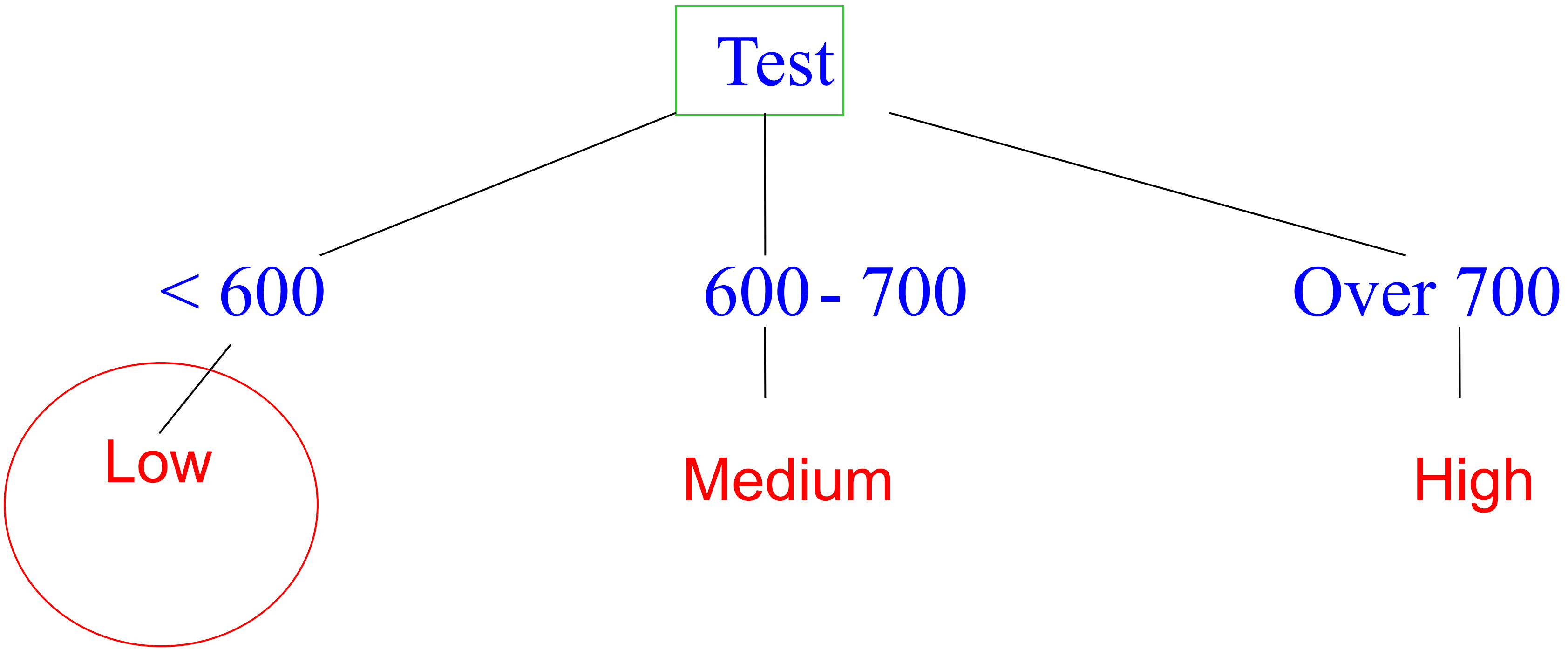
After pruning: 50%



CART™ Student Example (cont'd) Removing *Place of Birth*

- Cost-complexity of the single node t
  - $R_{\alpha}(\{t\}) = R(t) + \alpha*1 = 0.50 + \alpha$
- Cost-complexity of the branch T_t
  - $R_{\alpha}(T_t) = R(T_t) + \alpha*|\check{T}_t| = 0 + \alpha*2$
- The critical value of α
  - $R_{\alpha}(\{t\}) = R_{\alpha}(T_t)$
  - $0.50 + \alpha = 2\alpha$
  - $\alpha = 0.50$

CART™ Student Example (cont'd) New Sub-Tree (α = 0.50)



Lecture No. 6 – Decision Tree Learning II

- Rule Extraction
- Discretization of Continuous Attributes
- Alternative Splitting Rules
  - Information Gain Ratio
  - Gini Index
  - Twoing
- CART Overview
- Comparison of Decision Trees

Metrics for Performance Evaluation…

Confusion Matrix:

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

a: TP (true positive)

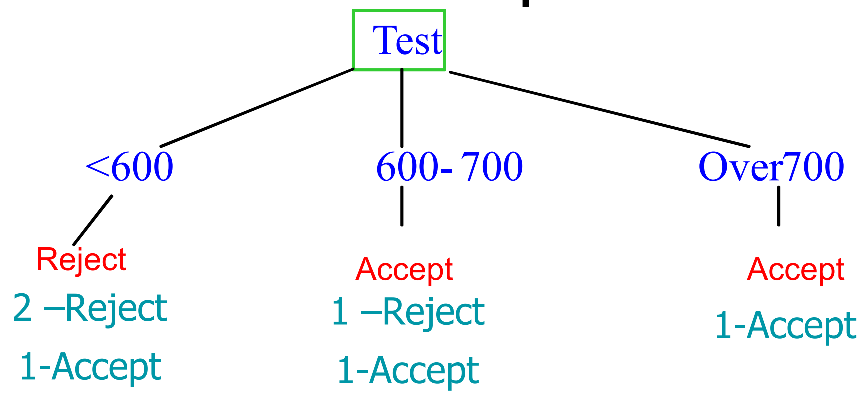
b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

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

Confusion Matrix Example



	PREDICTED CLASS		
		Class=Accept	Class=Reject
ACTUAL CLASS	Class=Accept	a = 2 (TP)	b = 1 (FN)
	Class=Reject	c = 1 (FP)	d = 2 (TN)

Accuracy = ?

Cost-Sensitive Measures

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

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

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

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

	PREDICTED CLASS		
		Class=Accept	Class=Reject
ACTUAL CLASS	Class=Accept	a = 2 (TP)	b = 1 (FN)
	Class=Reject	c = 1 (FP)	d = 2 (TN)

$p = ?$

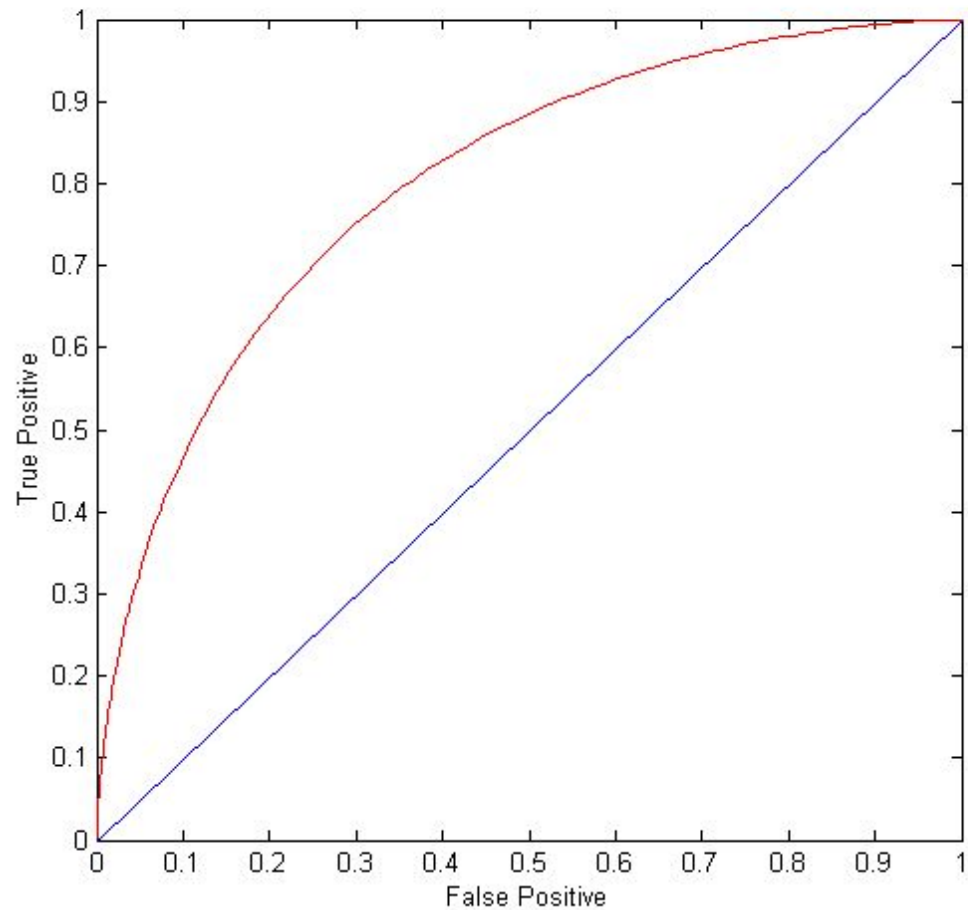
$r = ?$

$F = ?$

ROC Curve

TP Rate = TP / P

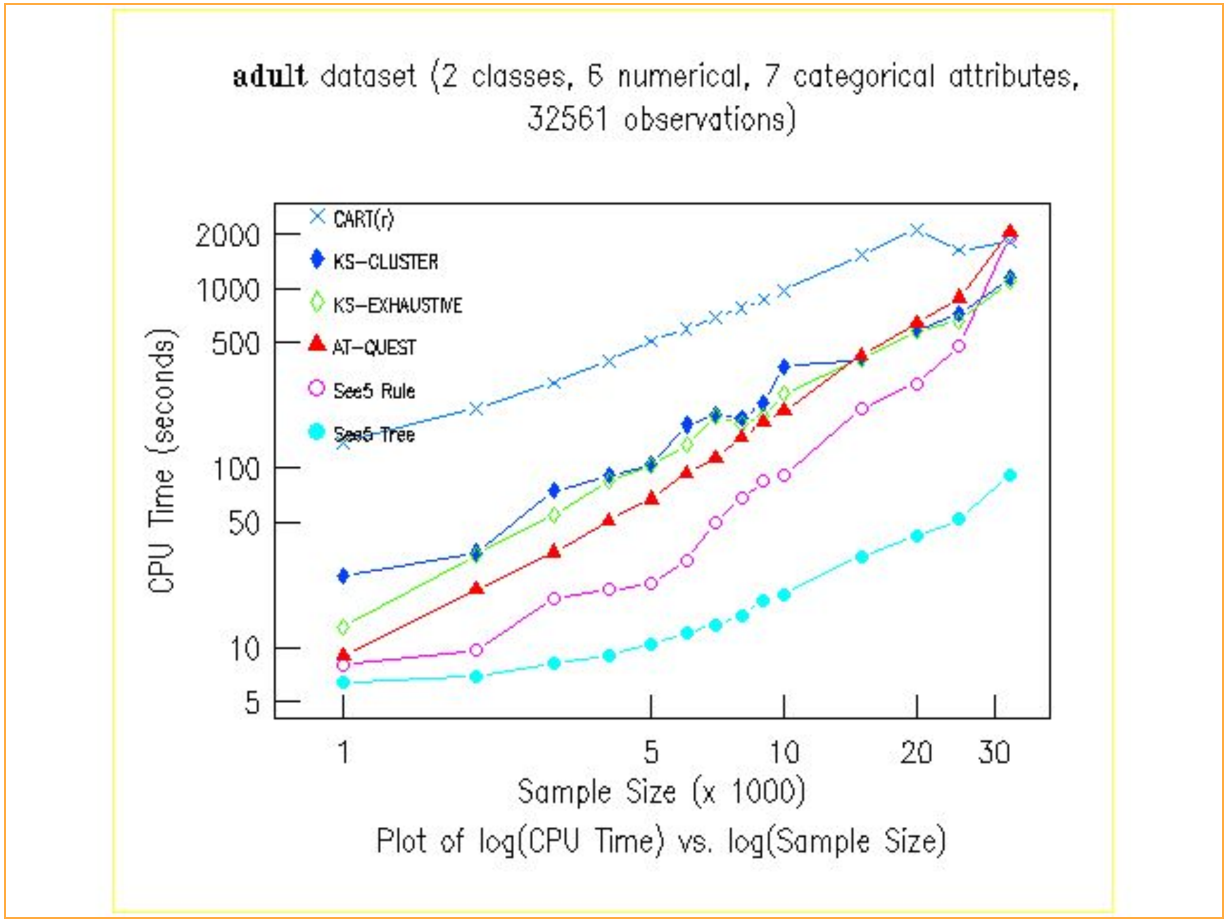
FP Rate = FP / N



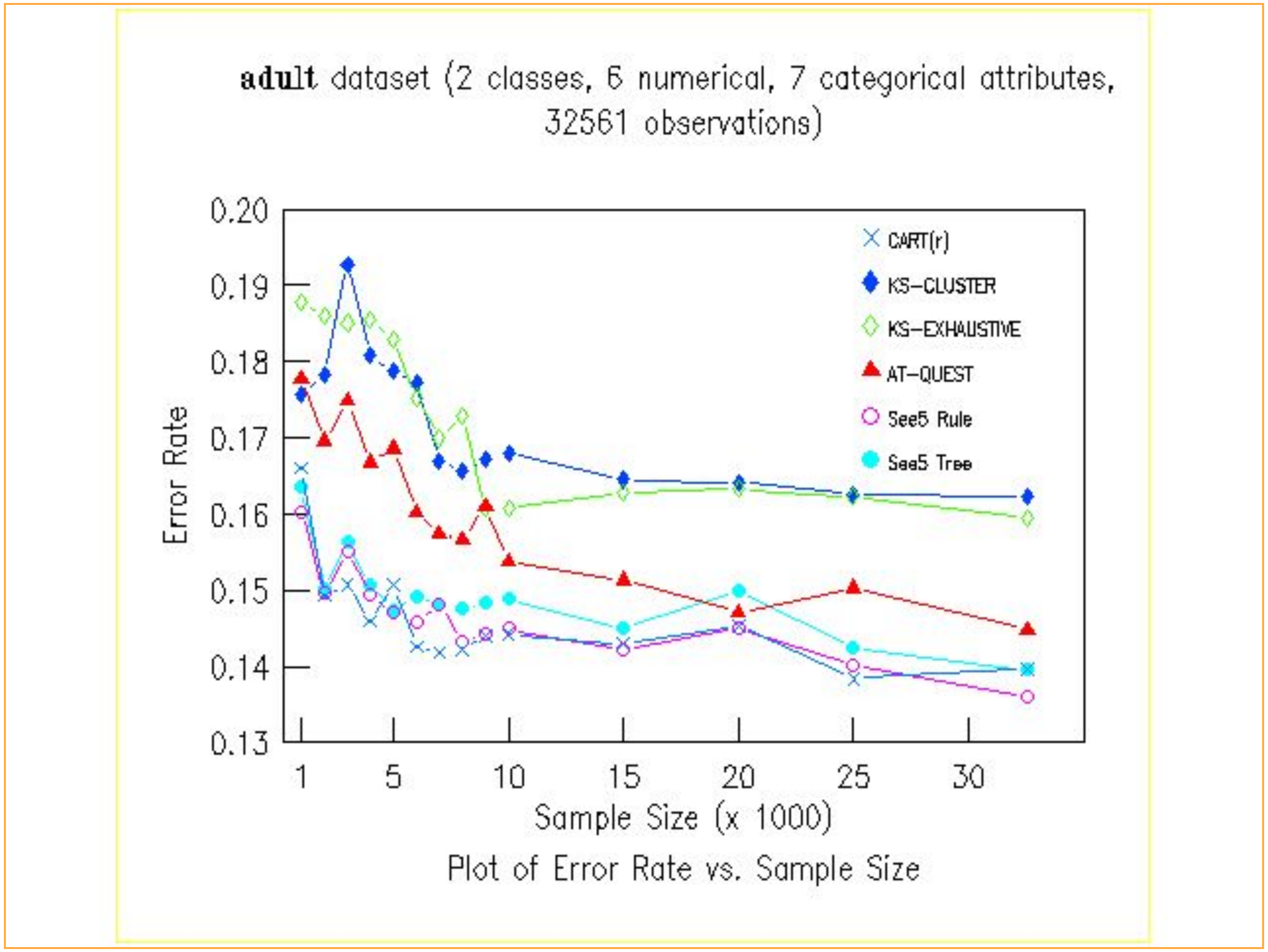
(TP,FP):

- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (1,0): ideal
- Area Under the ROC curve
  - Ideal:
    - Area = 1
  - Random guess:
    - Area = 0.5 (diagonal line)

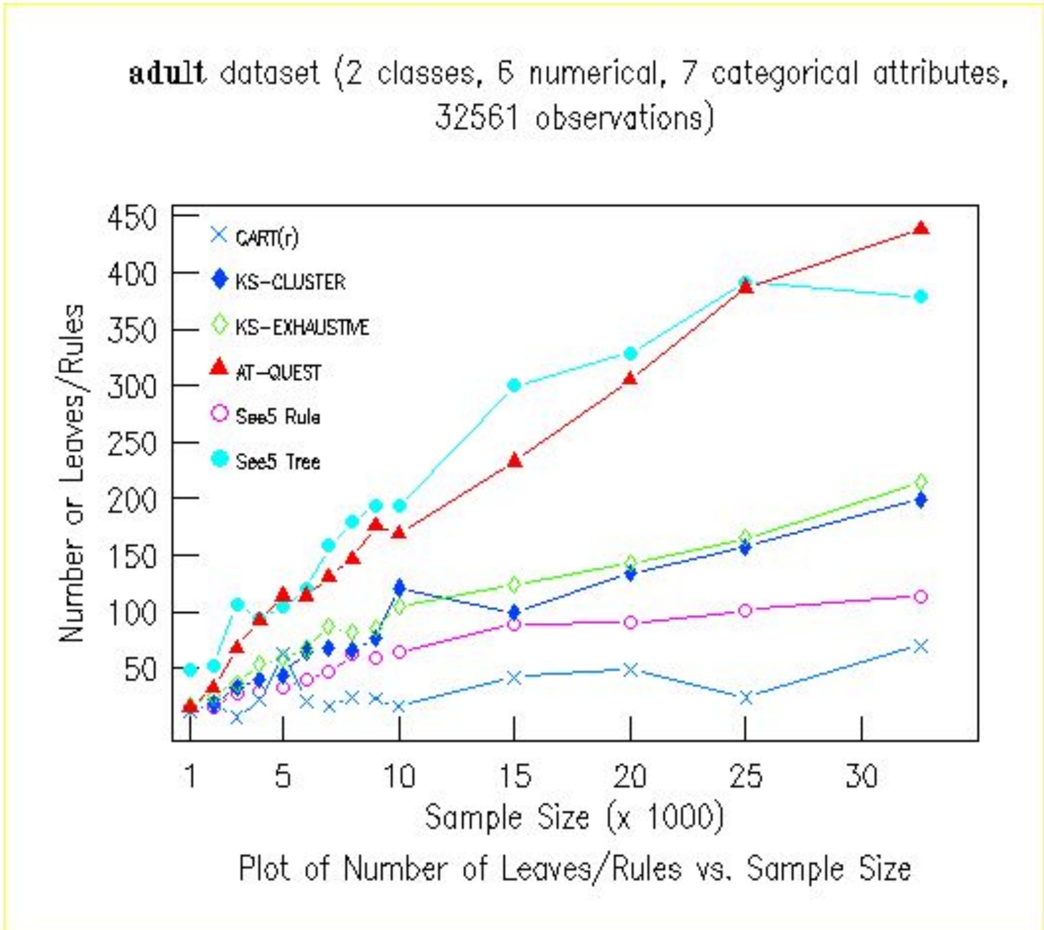
Comparison of Decision Trees
(based on Lim *et al*., Machine Learning, 40, 203–228, 2000)
Computational Complexity



Comparison of Decision Trees
Error Rate



Comparison of Decision Trees
Tree Size



Lectures No. 5-6: Summary

- Classification tasks involve *model construction* and *model testing*
- Decision trees are one of the most popular classification models
- Decision trees are usually constructed in *a top-down recursive divide-and-conquer manner*
- Overfitting can be avoided with *pre-pruning* and *post-pruning* techniques
- Most popular splitting criteria include *Gini*, *Twoing*, and *Entropy*