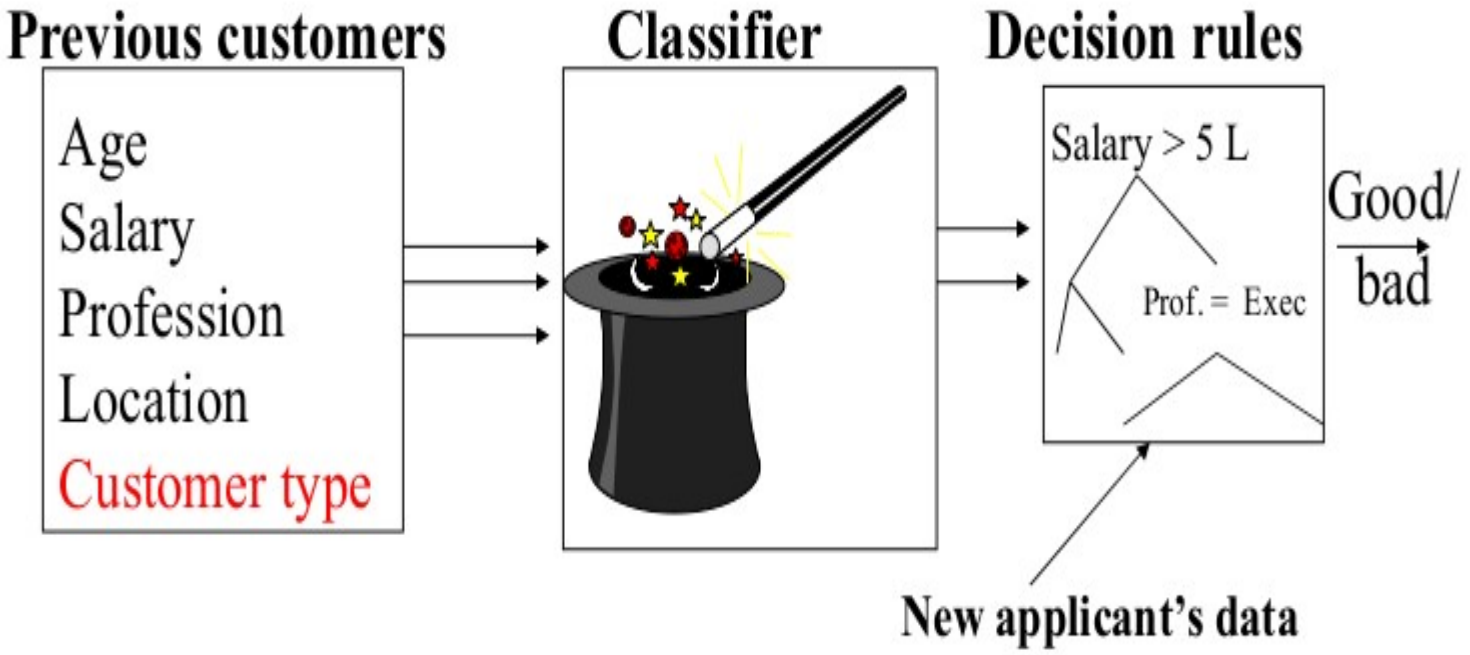
Decision Tree Classification

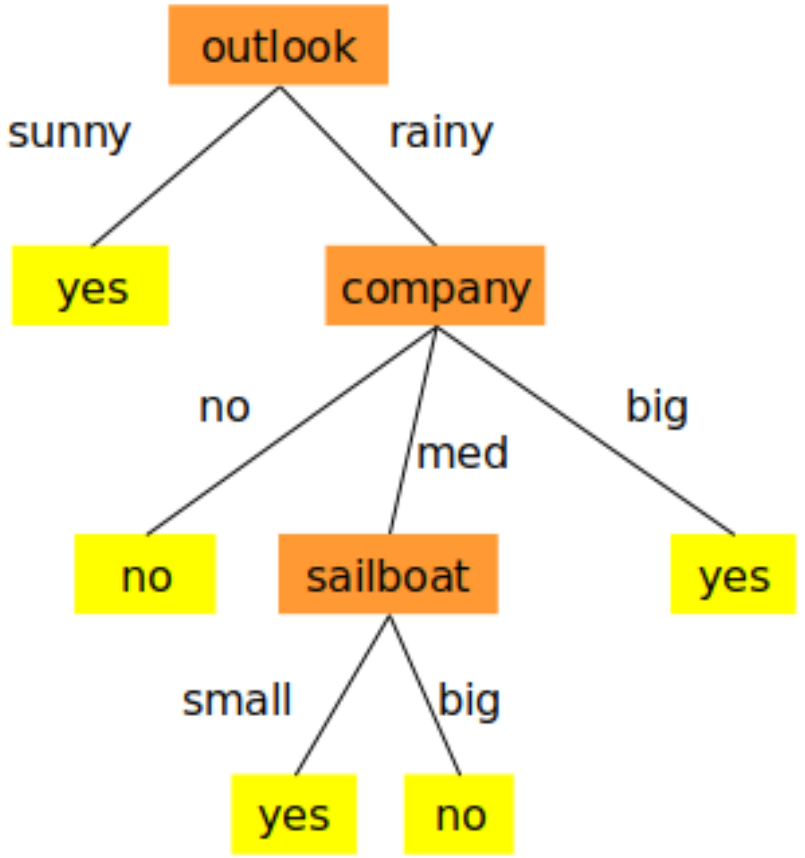
Setting

- Given old data about customers and payments, predict new applicant's loan eligibility



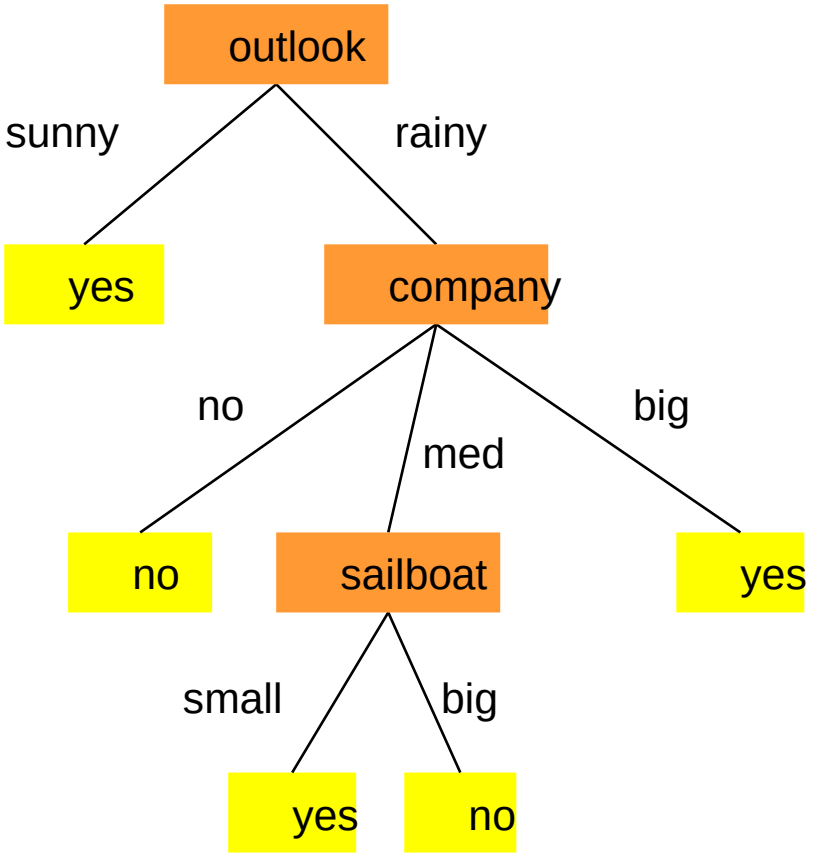
Example Dataset and Decision Tree(DT)

#	Attribute			Class
	Outlook	Company	Sailboat	Sail?
1	sunny	big	small	yes
2	sunny	med	small	yes
3	sunny	med	big	yes
4	sunny	no	small	yes
5	sunny	big	big	yes
6	rainy	no	small	no
7	rainy	med	small	yes
8	rainy	big	big	yes
9	rainy	no	big	no
10	rainy	med	big	no

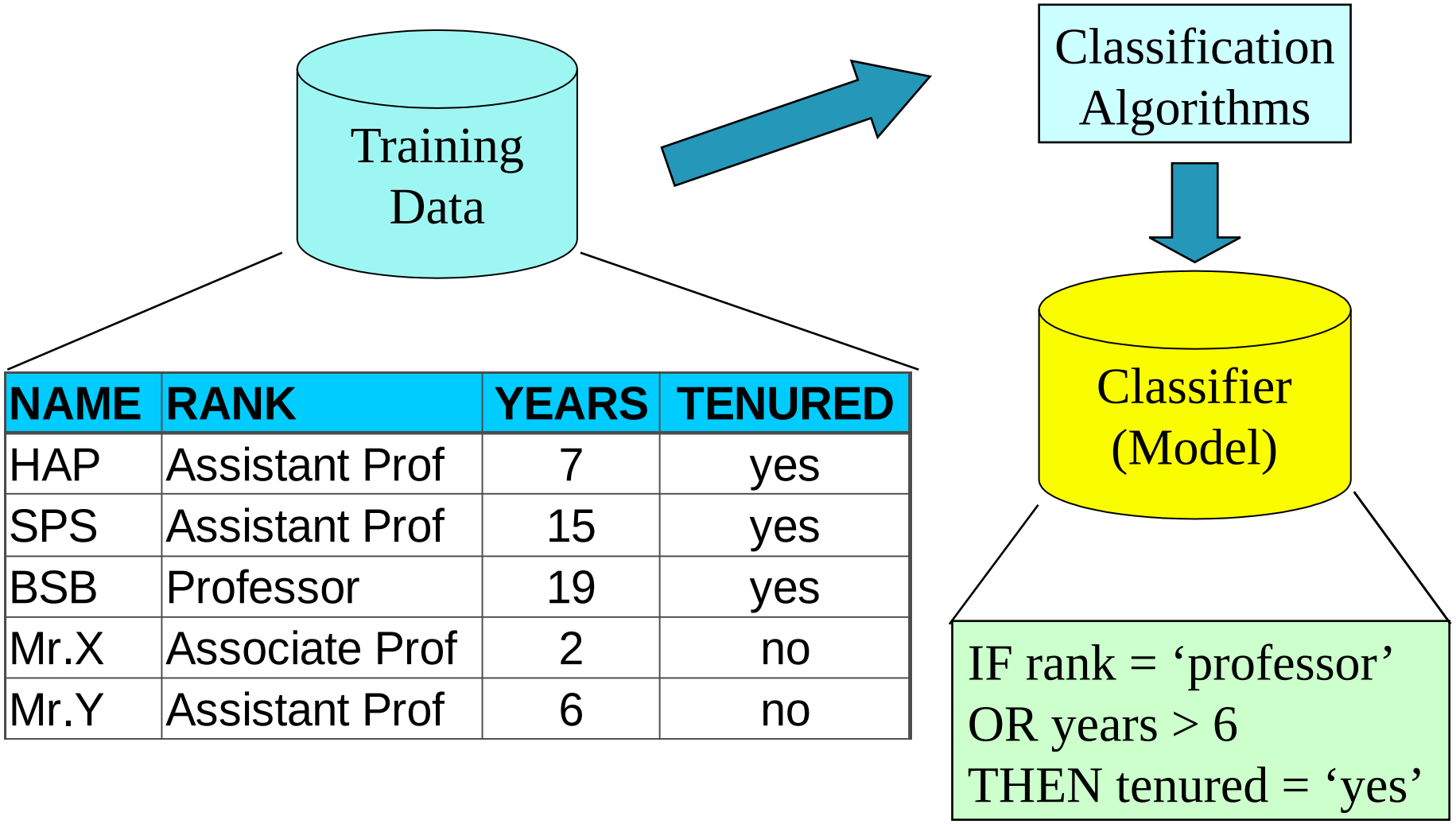


Classification

#	Attribute			Class
	Outlook	Company	Sailboat	Sail?
1	sunny	no	big	?
2	rainy	big	small	?

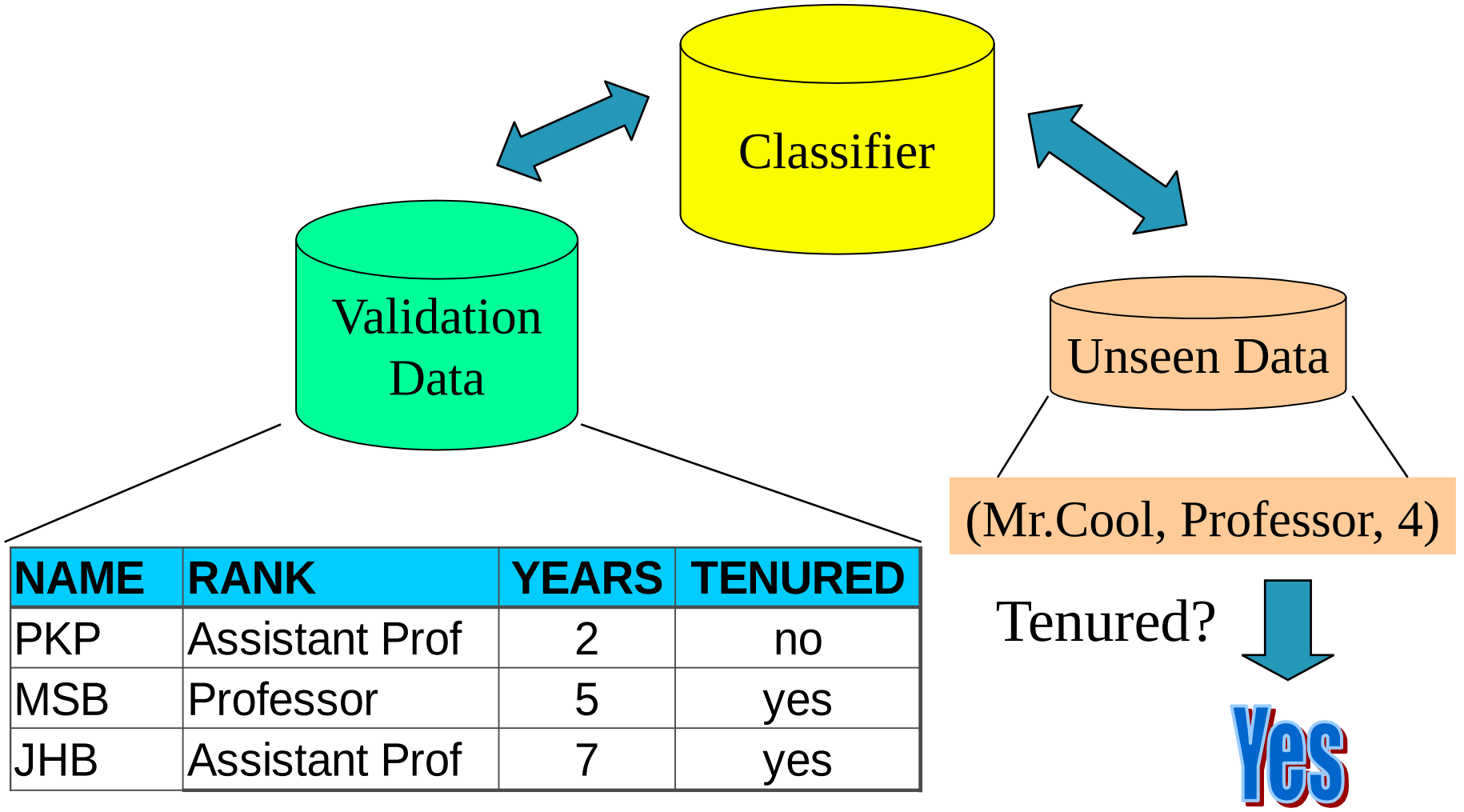


Model Construction



NAME	RANK	YEARS	TENURED
HAP	Assistant Prof	7	yes
SPS	Assistant Prof	15	yes
BSB	Professor	19	yes
Mr.X	Associate Prof	2	no
Mr.Y	Assistant Prof	6	no

Using model in prediction



NAME	RANK	YEARS	TENURED
PKP	Assistant Prof	2	no
MSB	Professor	5	yes
JHB	Assistant Prof	7	yes

Common terms used with Decision Tree

1. **Root Node:** It represents entire population or sample and this further gets divided into two or more homogeneous sets.
2. **Splitting:** It is a process of dividing a node into two or more sub-nodes.
3. **Decision Node:** When a sub-node splits into further sub-nodes, then it is called decision node.
4. **Leaf/ Terminal Node:** Nodes do not split is called Leaf or Terminal node.
5. **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning. You can say opposite process of splitting.
6. **Branch / Sub-Tree:** A sub section of entire tree is called branch or sub-tree.
7. **Parent and Child Node:** A node, which is divided into sub-nodes is called parent node of sub-nodes whereas sub-nodes are the children of parent node.

Decision Tree Induction

- Most of the existing systems to generate decision tree are based on Hunt's algorithm called "Top-Down Induction of Decision Tree (TDIDT)".
- Decision trees are constructed in a top-down recursive divide-and-conquer manner.
- Widely used decision tree algorithms are
  - ID3 (**I**terative **D**ichotomiser **3**)
  - C4.5
  - CART (**C**lassification **A**nd **R**egression **T**ree)

Inducing DT from Training Tuples

Algorithm: Generate decision tree: Generate a decision tree from the training tuples of data partition, D.

Input:

- Data partition, D: a set of training tuples and their associated class labels;
- attribute list: the set of candidate attributes;
- Attribute selection method: a procedure to determine the splitting criterion that “best” partitions the data tuples into individual classes.
  - Criterion consists of a splitting attribute and, possibly, either a split-point or splitting subset.

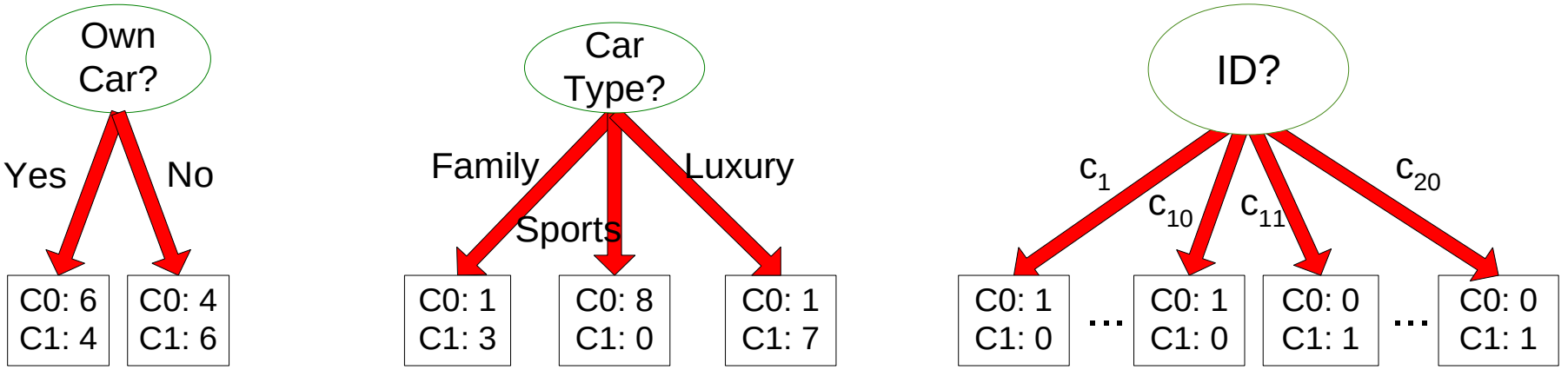
Output: A decision tree

Attribute Selection Methods

- Ex. information gain, Gain Ratio, Gini index.
- Whether the tree is strictly binary is generally driven by the attribute selection measure.
- Attribute selection measures, such as the Gini index, enforce the resulting tree to be binary.
- Others, like information gain, do not,
  - Therein allowing multiway splits
  - (i.e., two or more branches to be grown from a node)

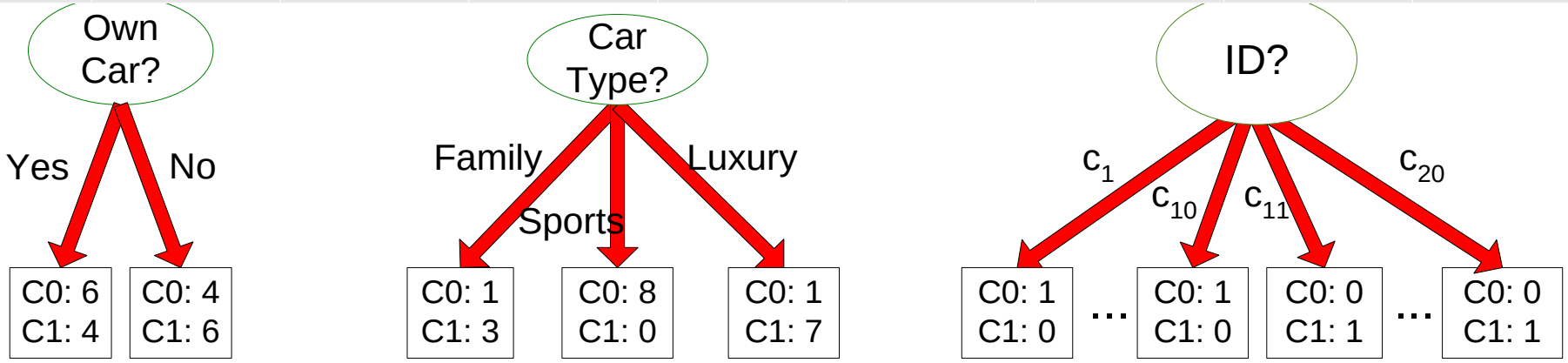
Attribute Selection Methods

- These methods are defined in terms of the class distribution of the records before and after splitting.
- Let p(i|t) denote the fraction of records belonging to class i at a given node t. We sometimes omit the reference to node t and express the fraction as p i .
- In a two-class problem, the class distribution at any node can be written as (p 0 ,p 1), where p 1 =1−p 0
- **Before Splitting: 10 records of class 0,**
  10 records of class 1



Which test condition is the best?

OC	CT	ID	Class		OC	CT	ID	Class
yes	Family	1	0		no	Family	11	1
yes	Sports	2	0		no	Family	12	1
yes	Sports	3	0		no	Family	13	1
yes	Sports	4	0		no	Luxury	14	1
yes	Sports	5	0		yes	Luxury	15	1
yes	Sports	6	0		yes	Luxury	16	1
no	Sports	7	0		yes	Luxury	17	1
no	Sports	8	0		yes	Luxury	18	1
no	Sports	9	0		no	Luxury	19	1
no	Luxury	10	0		no	Luxury	20	1



Which test condition is the best?

Attribute Selection Methods

- The class distribution before splitting is (0.5, 0.5) because there are an equal number of records from each class.
- If we split the data using the ***Own Car?*** attribute, then the class distributions of the child nodes are (0.6, 0.4) and (0.4, 0.6), respectively.
- Although the classes are no longer evenly distributed, the child nodes still contain records from both classes.
- Splitting on the second attribute, Car Type, will result in purer partitions.
- The measures developed for selecting the best split are often based on the degree of impurity of the child nodes.
  - The smaller the degree of impurity, the more skewed the class distribution.
  - For example, a node with class distribution (0, 1) has zero impurity, whereas a node with uniform class distribution (0.5, 0.5) has the highest impurity.
- Some of the popular impurity measures are Entropy, Gini and Classification Error.

Attribute Selection Methods

- The attribute selection measure provides a ranking for each attribute describing the given training tuples.
- The attribute having the best score for the measure is chosen as the splitting attribute for the given tuples.
- If the splitting attribute is continuous-valued or if we are restricted to binary trees, then, respectively, either a split point or a splitting subset must also be determined as part of the splitting criterion.
- We will discuss three popular attribute selection measures:

  1) Information gain

  2) Gain ratio

  3) Gini index

Information Gain Example: with nominal attributes

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Notations

- Let D, the data partition, is a training set of class-labeled tuples.
- Suppose the class label attribute has m distinct values defining m distinct classes, C_i (for i = 1,..., m).
- Let C_i,D be the set of tuples of class C_i in D.
- Let |D| and |C_i,D | denote the number of tuples in D and C_i,D , respectively

3-Steps

Expected Info:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i),$$

Information from Feature A:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

Information Gain:

$$Gain(A) = Info(D) - Info_A(D)$$

Solution

1. Above table presents a training set, D

2. The class:**buys_computer**, has two distinct values (namely, {yes, no}); therefore, there are two distinct classes (that is, m = 2).

Let class C1 correspond to yes and class C2 correspond to no.

The expected information needed to classify an instance in D:

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

3. Next, we need to compute the expected information requirement for each attribute. Let's start with the attribute **age**.

We need to look at the distribution of yes and no instances for each category of age.

a.For the age category **youth** : two yes instances and three no instances.

b.For the category **middle_aged**: four yes instances and zero no instances.

c.For the category **senior** : three yes instances and two no instances.

Solution

The expected information needed to classify an instance in D if the instances are partitioned according to **age** is:

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

$$\begin{aligned} Info_{age}(D) = & \frac{5}{14} \times (-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}) \\ & + \frac{4}{14} \times (-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}) \\ & + \frac{5}{14} \times (-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}) \\ = & 0.694 \text{ bits.} \end{aligned}$$

4. The gain in information from such a partitioning would be

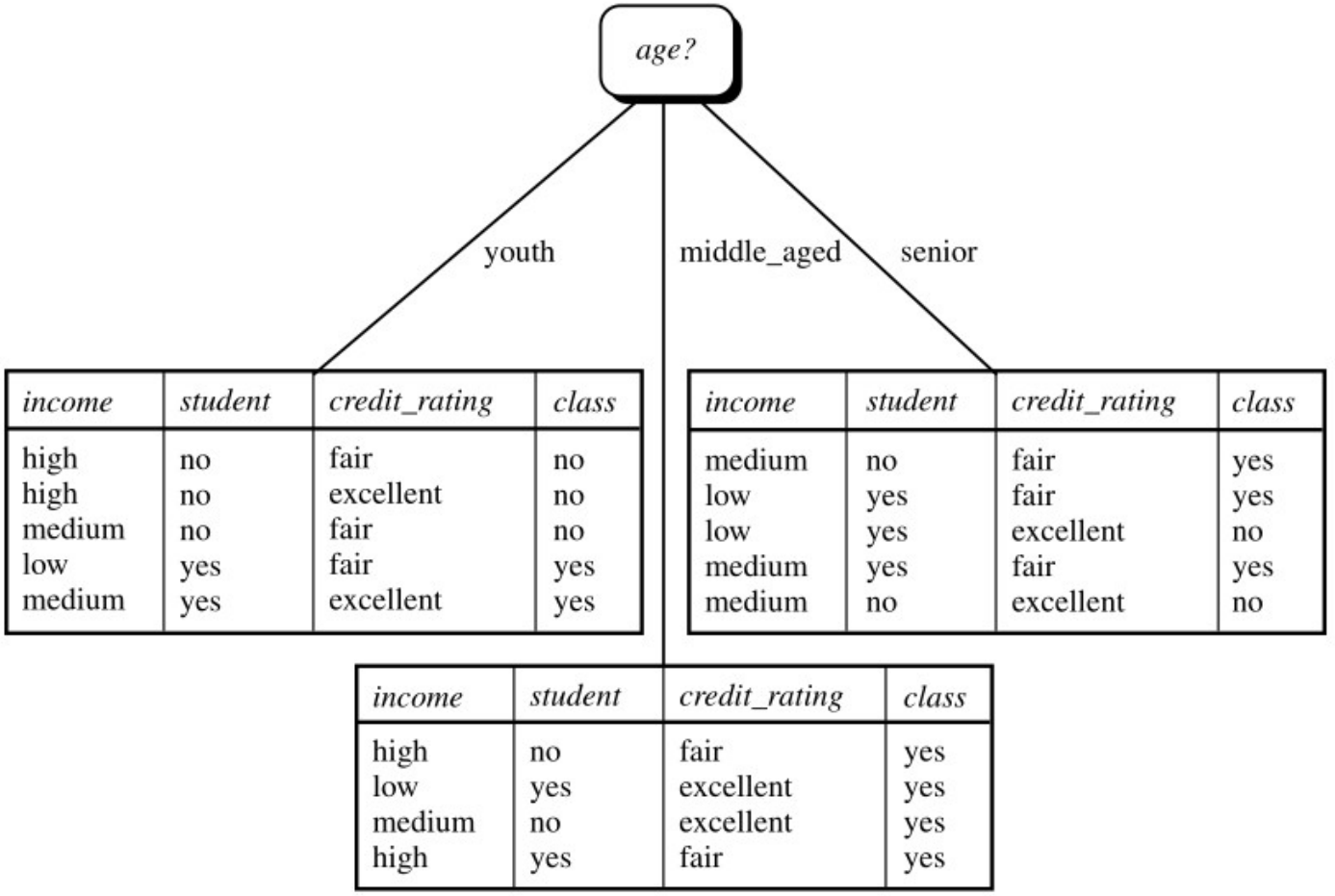
$$Gain(age) = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246 \text{ bits}$$

5. Similarly

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

6. Because **age** has the **highest information gain** among the attributes, it is selected as the splitting attribute.

Solution



income	*student*	*credit_rating*	*class*
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

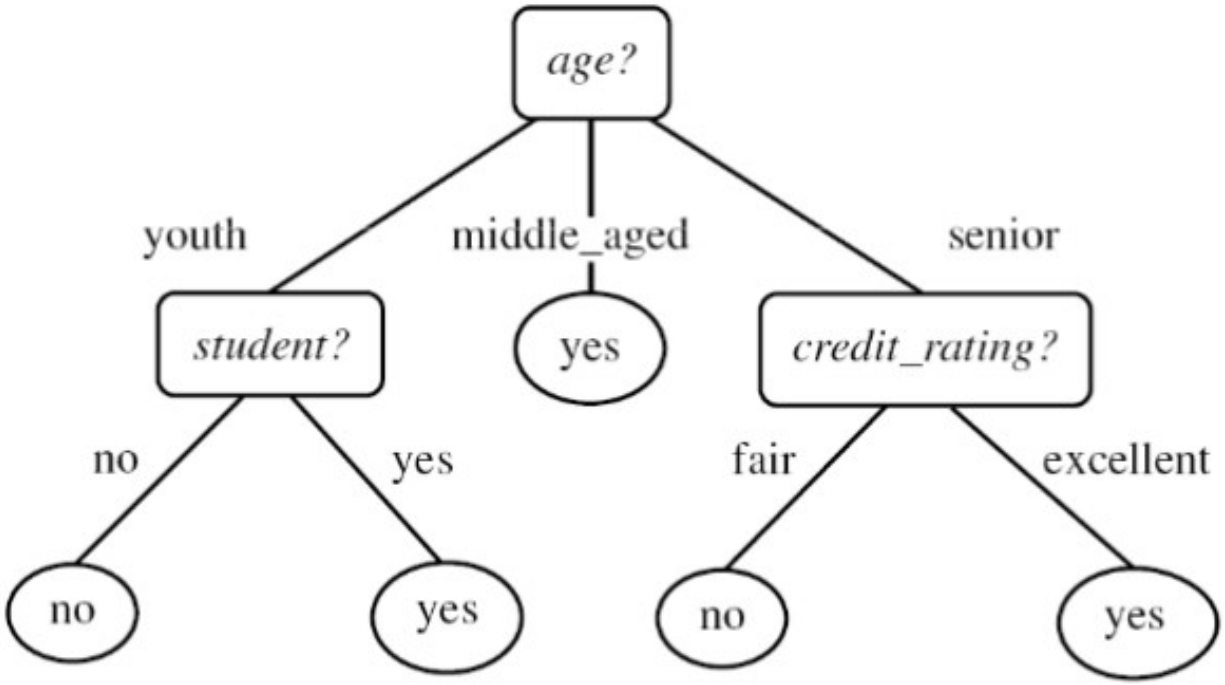
income	*student*	*credit_rating*	*class*
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

income	*student*	*credit_rating*	*class*
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

- Notice that the instances falling into the partition for age = middle_aged all belong to the same class.
- Because they all belong to class “yes,” a leaf should therefore be created at the end of this branch and labeled with “yes.”

Solution

- Final Decision Tree



Information Gain

- ID3 uses information gain as its attribute selection measure.
- Let node N represents or holds the tuples of partition D. The attribute with the highest information gain is chosen as the splitting attribute for node N.
- This attribute minimizes the information needed to classify the tuples in the resulting partitions and reflects the least randomness or “impurity” in these partitions.
- The expected information needed to classify a tuple in D is given by:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i),$$

  p_i is the nonzero probability that an arbitrary tuple in D belongs to class C_i and is estimated by $|C_i,D|/|D|$.

  A log function to the base 2 is used, because the information is encoded in bits.

  Info(D) is just the average amount of information needed to identify the class label of a tuple in D.

- Info(D) is also known as the entropy of D.

Information Gain

- Now, suppose we were to partition the tuples in D on some attribute A having v distinct values, {a 1 , a 2 , ... ,a v }, as observed from the training data.
- If A is discrete-valued, these values correspond directly to the v outcomes of a test on A. Attribute A can be used to split D into v partitions or subsets, {D 1 , D 2, ... , D v }, where D j contains those tuples in D that have outcome a j of A.
- These partitions would correspond to the branches grown from node N.
- How much more information would we still need (after the partitioning) to arrive at an exact classification? This amount is measured by

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

The term |D j | / |D| acts as the weight of the j th partition.

$Info_A(D)$ is the expected information required to classify a tuple from D based on the partitioning by A.

- The smaller the expected information required, the greater the purity of the partitions

Information Gain

- Information gain is defined as the difference between the original information requirement (i.e., based on just the proportion of classes) and the new requirement (i.e., obtained after partitioning on A).

$$Gain(A) = Info(D) - Info_A(D)$$

- Gain(A) tells us how much would be gained by branching on A.
- It is the expected reduction in the information requirement caused by knowing the value of A.
- The attribute A with the highest information gain, Gain(A), is chosen as the splitting attribute at node N.
- This is equivalent to saying that we want to partition on the attribute A that would do the "best classification," so that the amount of information still required to finish classifying the tuples is minimal (i.e., minimum InfoA(D)).