

Classification Lecture Notes
cse634
Chapter 6

Professor Anita Wasilewska

Classification

(Data Mining Book Chapter 6)

- **PART ONE:**
- **Supervised learning and Classification**
- Data format: training and test data
- Concept, or class definitions and description
- Rules learned: characteristic and discriminant
- **Supervised learning** = classification process = building a classifier.
- Classification algorithms
- Evaluating predictive accuracy of a classifier: the most common methods for testing

Classification Algorithms (Models, Basic Classifiers)

- Decision Trees (ID3, C4.5)
- Neural Networks
- Bayesian Classifiers (Networks)
- Genetic Algorithm
- Rough Sets

Other Classification Methods

- k-nearest neighbor classifier
- Case-based reasoning
- Fuzzy set approaches

Classification Data Format

Learning Functionalities (1)

- **Classification Data Format:** a data table with key attribute removed .
- Special attribute, called a **class attribute** must be distinguished.
- The values of the class attribute are called **class labels**.
- The class label attribute is discrete-valued and unordered.
- Class attribute is **categorical** in that each value serves as a category, or class.
- The records in the classification data are called **tuples with their associated labels**.
- It means that we distinguish in a record its attribute part and class part.
- The attribute part is called data tuple, attribute vector, sample, example, instance, data point (with associate label).

Classification Data Example 1

- Example: Classification data with class attribute **C**.

a1	a2	a3	a4	C
1	1	m	g	c1
0	1	v	g	c2
1	0	m	b	c1

- This data consists of tuples (examples):
- $t_1 = (1, 1, m, g)$ with the class label c_1
- $t_2 = (0, 1, v, g)$ with the class label c_2
- $t_3 = (1, 0, m, b)$ with the class label c_1

Classification Data 1

- **Classification Data Format:** a data table with key attribute removed. Special attribute, called a class attribute must be distinguished. The values of the class attribute are called class labels. Example below has a class attribute: `buys_computer`.

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
30...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 Training Data 2 (with objects)

Learning Functionalities (2)

rec	Age	Income	Student	Credit_rating	Buys_computer
r1	<=30	High	No	Fair	No
r2	<=30	High	No	Excellent	No
r3	31...40	High	No	Fair	Yes
r4	>40	Medium	No	Fair	Yes
r5	>40	Low	Yes	Fair	Yes
r6	>40	Low	Yes	Excellent	No
r7	31...40	Low	Yes	Excellent	Yes
r8	<=30	Medium	No	Fair	No
r9	<=30	Low	Yes	Fair	Yes
r10	>40	Medium	Yes	Fair	Yes
r11	<=30	Medium	Yes	Excellent	Yes
r12	31...40	Medium	No	Excellent	Yes
r13	31...40	High	Yes	Fair	Yes
r14	>40	Medium	No	Excellent	No

Concept or Class definitions

Learning Functionalities (3)

- **Syntactically** a **Concept** or a **Class** is defined by the concept (class) attribute **C** and its value **V**
- **A Concept, Class (syntactic) description** is written as : **C=V**
- **Semantically**, a concept, or a class defined by the attribute **c** is the set **C** of all records for which the attribute **c** has a value **v**.

Concept or Class definition

Learning Functionalities (3)

- **Example:**

{ r1, r2, r6, r8, r14 } of the Classification Training Data 2 on the previous slide

Syntactically it is defined by the concept (class) attribute **buys_computer** and its value **no**

Concept (class) { r1, r2, r6, r8, r14 } description is: **buys_computer= no**

Concept, Class characteristics

Learning, Classification Functionalities (4)

Characteristics of a class (concept) C

is a set of attributes a_1, a_2, \dots, a_k , and their respective values v_1, v_2, \dots, v_k such that the intersection of set of all records for which $a_1=v_1 \ \& \ a_2=v_2 \ \& \ \dots \ \& \ a_k=v_k$ with set C is not empty

Characteristics description of C of is then syntactically written as $a_1=v_1 \ \& \ a_2=v_2 \ \& \ \dots \ \& \ a_k=v_k$

REMARK: A concept C can have many characteristic descriptions.

Concept, Class characteristic formula Learning, Classification Functionalities (5)

Definition:

A formula $a_1=v_1 \ \& \ a_2=v_2 \ \& \ \dots \ a_k=v_k$ (of a proper language) is called **a characteristic description** for **a class** (concept) **C**

If and only if

$R_1 = \{r: a_1=v_1 \ \& \ a_2=v_2 \ \& \ \dots \ a_k=v_k \} \wedge C = \text{not empty set}$

Concept, Class characteristics

Learning, classification Functionalities (6)

Example:

- Some of the **characteristic descriptions** of the concept (class) **C** with description: **buys_computer= no** are
 - Age= \leq 30 & income=high & student=no & credit_rating=fair
 - Age= $>$ 40& income=medium & student=no & credit_rating=excellent
 - Age= $>$ 40& income=medium
 - Age= \leq 30
 - student=no & credit_rating=excellent

Concept, Class characteristics

Learning, classification Functionalities (7)

- A formula
- $\text{Income}=\text{low}$ is a **characteristic description** of the concept (class) **C1** with description: **buys_computer= yes** and of the concept (class) **C2** with description: **buys_computer= no**
- A formula
- $\text{Age}\leq 30 \ \& \ \text{Income}=\text{low}$ is **NOT** the **characteristic description** of the concept **C1** with description: **buys_computer= no**

because:

$\{ r: \text{Age}\leq 30 \ \& \ \text{Income}=\text{low} \} \wedge \{ r: \text{buys_computer}=\text{no} \} =$
emptyset

Characteristic Formula

Any formula (of a proper language) of a form

IF concept (class) description
THEN characteristics

is called **a characteristic formula**

Example:

- **IF** buys_computer= no **THEN** income = low & student=yes & credit=excellent
- **IF** buys_computer= no **THEN** income = low & credit=fair

Characteristic Rule (1)

- A characteristic formula

IF concept (class) description

THEN characteristics

is called **a characteristic rule** (for a given database)

if and only if it is **TRUE** in the given database, i.e.

{r: concept description} & {r: characteristics} = not empty set

Characteristic Rule (2)

EXAMPLE:

The formula

- IF buys_computer= no THEN income = low & student=yes & credit=excellent

Is **a characteristic rule** for our database because

$\{r: \text{buys_computer} = \text{no}\} = \{r1, r2, r6, r8, r16\},$

$\{r: \text{income} = \text{low} \ \& \ \text{student} = \text{yes} \ \& \ \text{credit} = \text{excellent}\} = \{r6, r7\}$

and

$\{r1, r2, r6, r8, r16\} \wedge \{r6, r7\} = \text{not emptyset}$

Characteristic Rule (3)

EXAMPLE:

The formula

- IF buys_computer= no THEN income = low & credit=fair

Is **NOT** a characteristic rule for our database because

$\{r: \text{buys_computer} = \text{no}\} = \{r1, r2, r6, r8, r16\},$

$\{r: \text{income} = \text{low} \ \& \ \text{credit} = \text{fair}\} = \{r5, r9\}$

and

$\{r1, r2, r6, r8, r16\} \wedge \{r5, r9\} = \text{emptyset}$

Discrimination

- *Discrimination is the process which aim is to find rules that allow us to **discriminate** the objects (records) belonging to a given concept (one class) from the rest of records (classes)*

If characteristics then concept (class)

- *Example*
- ***If*** Age= \leq 30 & income=high & student=no & credit_rating=fair
then buys_computer= no

Discriminant Formula

A discriminant formula is any formula

If characteristics

then concept (class)

- Example:
- IF Age=>40 & inc=low THEN buys_comp= no

Discriminant Rule

- A discriminant formula

If characteristics

then concept (class)

is a ***DISCRIMINANT RULE*** (in a given database)

iff

{r: Characteristic} \sqsubseteq {r: concept}

Discriminant Rule

- **Example:**
- *A discriminant formula*

IF Age=>40 & inc=low
THEN buys_comp= no

IS NOT a discriminant rule in our data base
because

$\{r: \text{Age} \Rightarrow 40 \ \& \ \text{inc} = \text{low}\} = \{r5, r6\}$ is not a subset of the
set $\{r : \text{buys_comp} = \text{no}\} = \{r1, r2, r6, r8, r14\}$

Characteristic and discriminant rules

- The inverse implication to the characteristic rule is usually NOT a discriminant rule
- Example : the inverse implication to our characteristic rule: ***If*** buys_computer= no **then** income = low & student=yes & credit=excellent **is**
- ***If*** income = low & student=yes & credit=excellent **then** buys_computer= no
- The above rule is NOT a discriminant rule as it can't discriminate between concept with description buys_computer= no and buys_computer= yes
- (see records r7 and r8 in our training dataset)

DM Classification

Supervised Learning Goal (1)

- Given a data set and a concept, class **c** defined in this dataset **FIND a minimal set (or as small as possible set) characteristic, and/or discriminant rules, or other descriptions** for the concept **c**, or class, or classes.

DM Classification

Supervised Learning Goal (2)

- We also want these rules to involve as few attributes as it is possible, i.e. we want the rules to have **as short as possible length of descriptions.**

Classification

Supervised Learning

- The process of creating discriminant and/or characteristic rules, or other descriptions and **TESTING** them is called a **learning process**, and when it is finished we say that the concept has been learned (and tested) from examples (records in the dataset).
- It is called **a supervised learning** because we know the class labels of all examples.

A small, full set **DISCRIMINANT RULES** for concepts: *buys_comp=yes*, *buys_comp=no*

- The rules are:

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 = “no”

IF *age* = “<=30” AND *credit_rating* = “fair” THEN
buys_computer = “yes”

Rules testing

- In order to use rules for testing, and later when testing is done and predictive accuracy is acceptable we write rules in a **predicate form**:

IF *age(x, <=30)* AND *student(x, no)* THEN

buys_computer (x, no)

IF *age(x, <=30)* AND *student (x, yes)* THEN

buys_computer (x, yes)

- Attributes and their values of the new record x are matched with the IF part of the rule and the record is classified accordingly to the THEN part of the rule.

Test dataset

- The Test Dataset has the same format as the training dataset, i.e. the values of class attribute are known.
- We use it to evaluate the predictive accuracy of our rule
- **PREDICTIVE ACCURACY** of the set of rules, or any classification algorithm is a percentage of well classified data in the testing dataset.
- If the predictive accuracy is not high enough we chose a different learning and testing datasets and start process again
- There are many methods of testing the rules and they will be discussed later

Classification Data

- **Classification Data Format:** a data table with key attribute removed .
- **Special attribute, called a class attribute** must be distinguished.
- The values: C_1, C_2, \dots, C_n of the class attribute are called **class labels**.
- Example:

a1	a2	a3	a4	C
1	1	m	g	c1
0	1	v	g	c2
1	0	m	b	c1

Classification and Classifiers

- An algorithm (model, method) is called **a classification algorithm** if it uses the classification data i.e. data and its classification to build a set of patterns: discriminant and /or characteristic rules or other pattern descriptions.
- Those patterns are structured in such a way that we can use them **to classify unknown sets of objects:** unknown tuples, records.
- For that reason, and because of the goal a classification algorithm is often called shortly **a classifier.**
- The name **classifier** implies more than just classification algorithm.
- **A classifier is a final product of the data set and a classification algorithm.**

Building a Classifier

- Building a classifier consists of two phases:
training and testing.
- In both phases we use data (**training data set** and disjoint with it **test data set**) for which the class labels are known for ALL of the records.
- **We use** the training data set to create patterns (rules, trees, or to train a Neural or Bayesian network).
- **We evaluate** created patterns with the use of of test data, which classification is known.
- The measure for a trained classifier accuracy is called **predictive accuracy.**
- **The classifier is build** i.e. we terminate the process if it has been trained and tested and predictive accuracy was on an acceptable level.

Classification = Supervised Learning (book slide)

- **Classification = Supervised learning goal:**

Finding models (**rules**) that describe (**characterize**)
or/ and distinguish (**discriminate**) classes or
concepts for future prediction

Example: classify countries based on climate, or
classify cars based on gas mileage and use it to
predict classification of a new car on a base of
other attributes

Presentation: decision-tree, classification rules,
neural network

Classification vs. Prediction

(book slide)

- **Classification:**

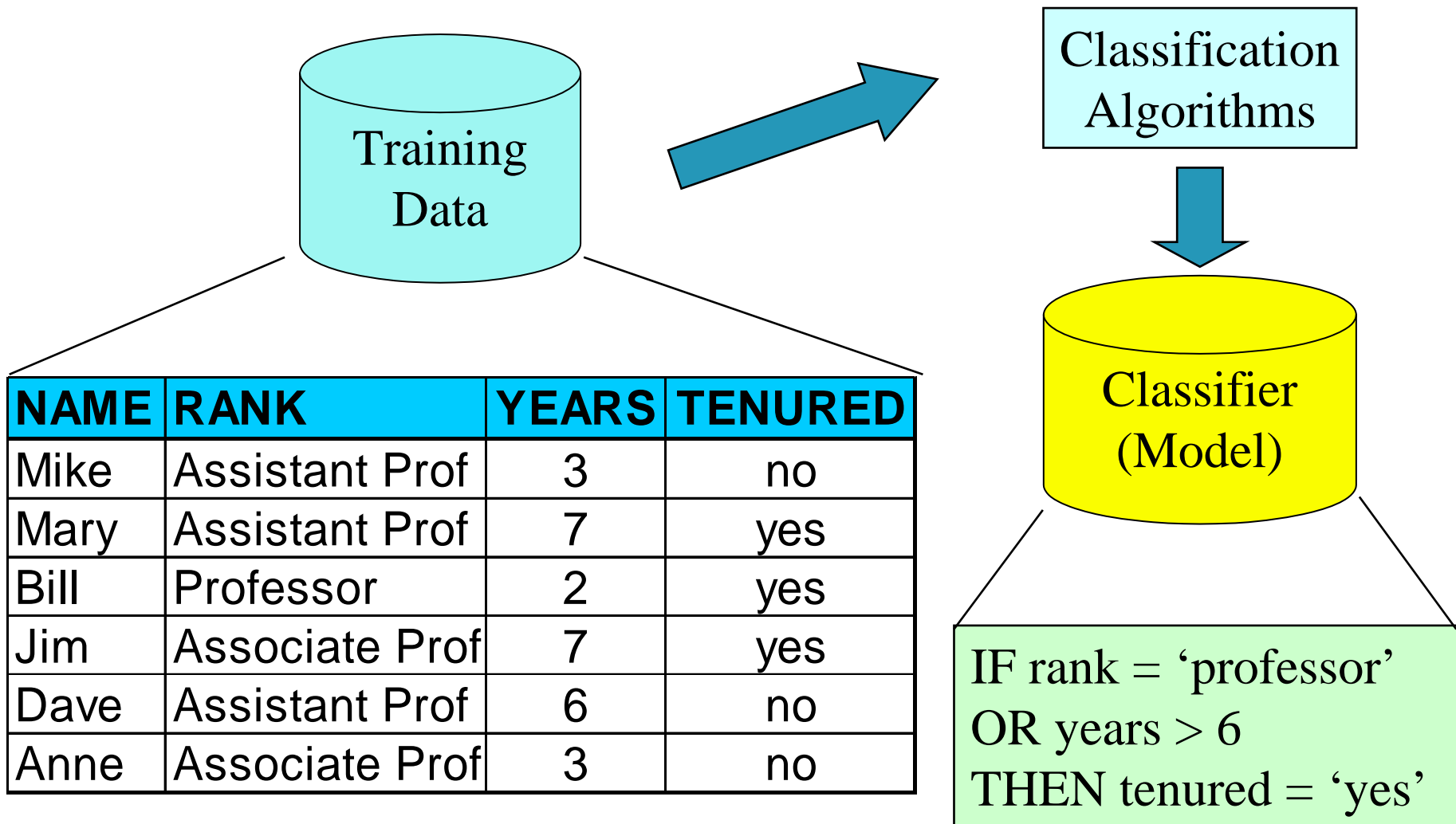
When a classifier is build it predicts categorical class labels of new data – classifies unknown data. We also say that it **predicts class labels** of the new data

Construction of the classifier (a model) is based on a training set in which the values of a decision attribute (**class labels**) are given and is tested on a test set

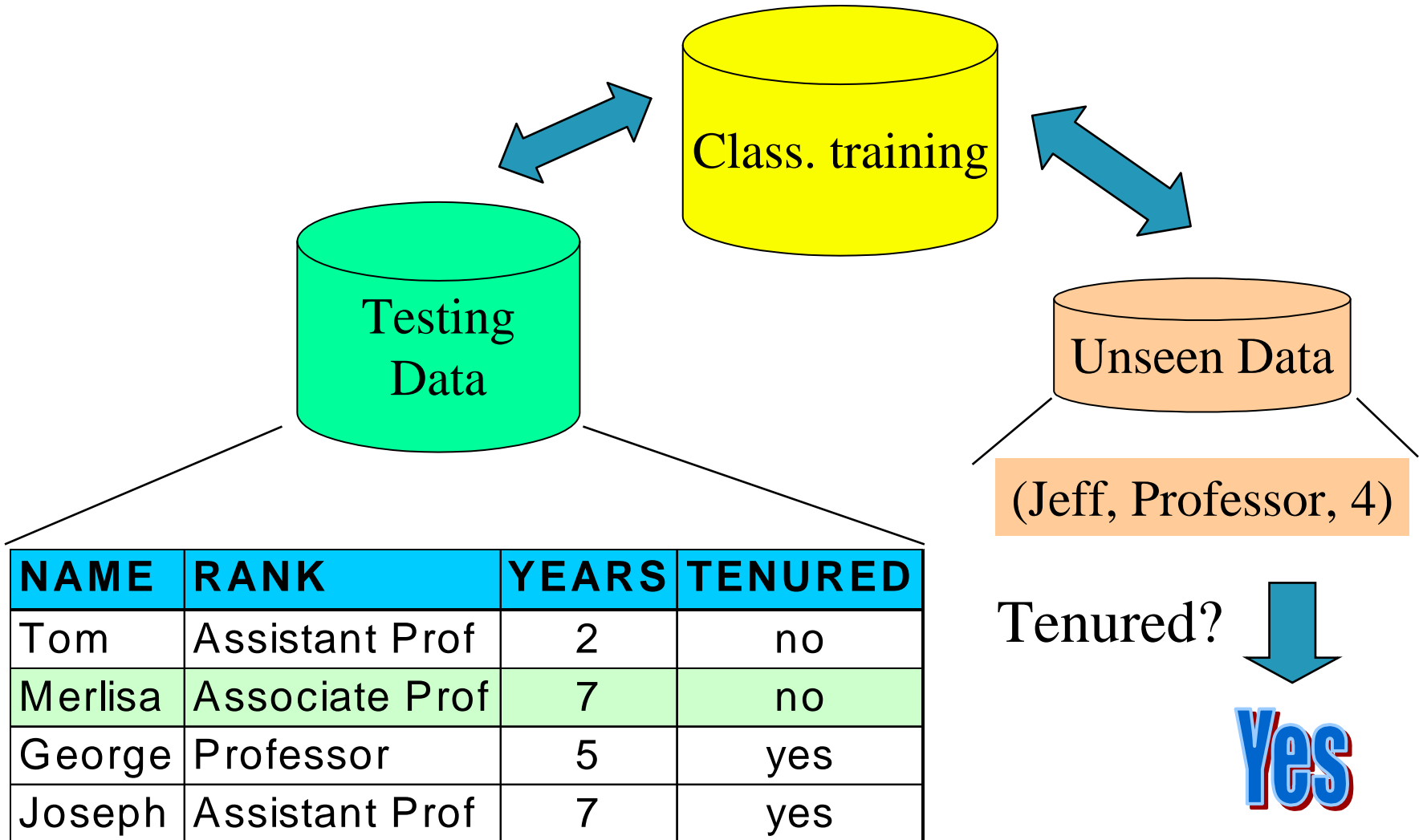
- **Prediction**

Statistical method that models continuous-valued functions, i.e., predicts unknown or missing values

Classification Process : a Classifier (book slide)



Testing and Prediction (book slide)



Classifiers Predictive Accuracy

- **PREDICTIVE ACCURACY** of a classifier is a percentage of well classified data in the testing data set.
- **Predictive accuracy depends heavily on a choice of the test and training data.**
- There are many methods of choosing test and training sets and hence evaluating the predictive accuracy. This is a separate field of research.
- **Basic methods are presented in TESTING CLASSIFICATION lecture Notes.**

Predictive Accuracy Evaluation

The main methods of predictive accuracy evaluations are:

- **Re-substitution** ($N ; N$)
- **Holdout** ($2N/3 ; N/3$)
- **x-fold cross-validation** ($N-N/x ; N/x$)
- **Leave-one-out** ($N-1 ; 1$),

where **N** is the number of instances in the dataset (see separate presentation)

- The process of building and evaluating a classifier is also called a **supervised learning**, or lately when dealing with large data bases a classification method in **Data Mining**.

Supervised vs. Unsupervised Learning (book slide)

- **Supervised learning (classification)**

Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations.

New data is classified based on a tested classifier

Supervised vs. Unsupervised Learning

(book slide)

- **Unsupervised learning (clustering)**

The class labels of training data is unknown

We are given a set of records (measurements, observations, etc.)

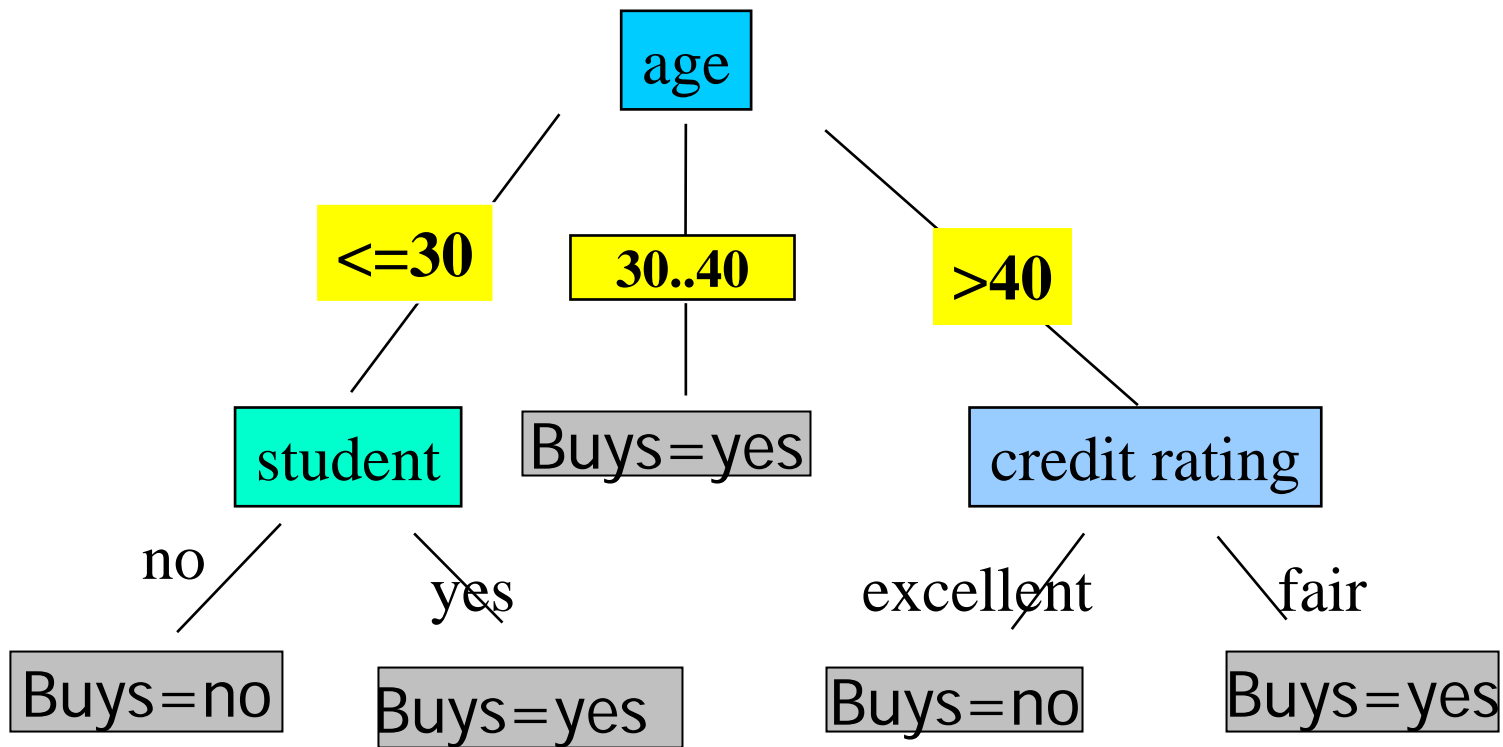
with the aim of establishing the existence of classes or **clusters** in the data

Part 2: Classification Algorithms (Models, Classifiers)

- Decision Trees (ID3, C4.5)
- Neural Networks
- Bayesian Networks
- Genetic Algorithms
- Rough Sets

PART 2: DECISION TREES:

An Example (book slide)



Classification by Decision Tree Induction

- **Decision tree**

A flow-chart-like tree structure

Internal node denotes an attribute

Branch represents the values of the node
attribute

Leaf nodes represent class labels or class
distribution

Classification by Decision Tree Induction

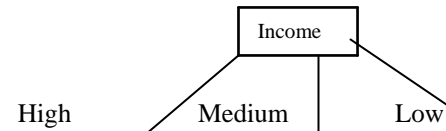
Basic Algorithm

- **The basic algorithm for decision tree construction** is a greedy algorithm that constructs decision trees in a top-down recursive divide-and-conquer manner.
- Given a training set D of classification data, i.e. a data table with a distinguished class attribute.
- This training set is recursively partitioned into smaller subsets (data tables) as the tree is being built.
- Tree **STARTS** as a single node (root) representing all training dataset (samples).
- We choose a node (root) an attribute from D. It is called a **SPLIT** attribute.
- **A branch is created** for each value of the node (root) attribute and **is labeled by this values** and the samples (it means the data table) are partitioned accordingly.
- The algorithm uses **the same process recursively** to form a decision tree at each partition.
- The recursive partitioning **STOPS** only when any one of the following conditions is true.
 1. all the samples (records) in the partition are of the same class, then the node becomes **the leaf labeled with that class**
 - **OR**
 - **2.** There is no remaining attributes on which the data may be further partitioned. In this case we apply **MAJORITY VOTING** to classify the node.
 - This involves **converting the node into a leaf** and labeling it with the most common class.
 - **3.** There are no tuples for a given branch, i.e. the partition is empty a leaf is created with the **majority class in the training data**

Training data
Class attribute: Buys_computer

rec	Age	Income	Student	Credit_rating	Buys_computer
r1	<=30	High	No	Fair	No
r2	<=30	High	No	Excellent	No
r3	31...40	High	No	Fair	Yes
r4	>40	Medium	No	Fair	Yes
r5	>40	Low	Yes	Fair	Yes
r6	>40	Low	Yes	Excellent	No
r7	31...40	Low	Yes	Excellent	Yes
r8	<=30	Medium	No	Fair	No
r9	<=30	Low	Yes	Fair	Yes
r10	>40	Medium	Yes	Fair	Yes
r11	<=30	Medium	Yes	Excellent	Yes
r12	31...40	Medium	No	Excellent	Yes
r13	31...40	High	Yes	Fair	Yes
r14	>40	Medium	No	Excellent	No

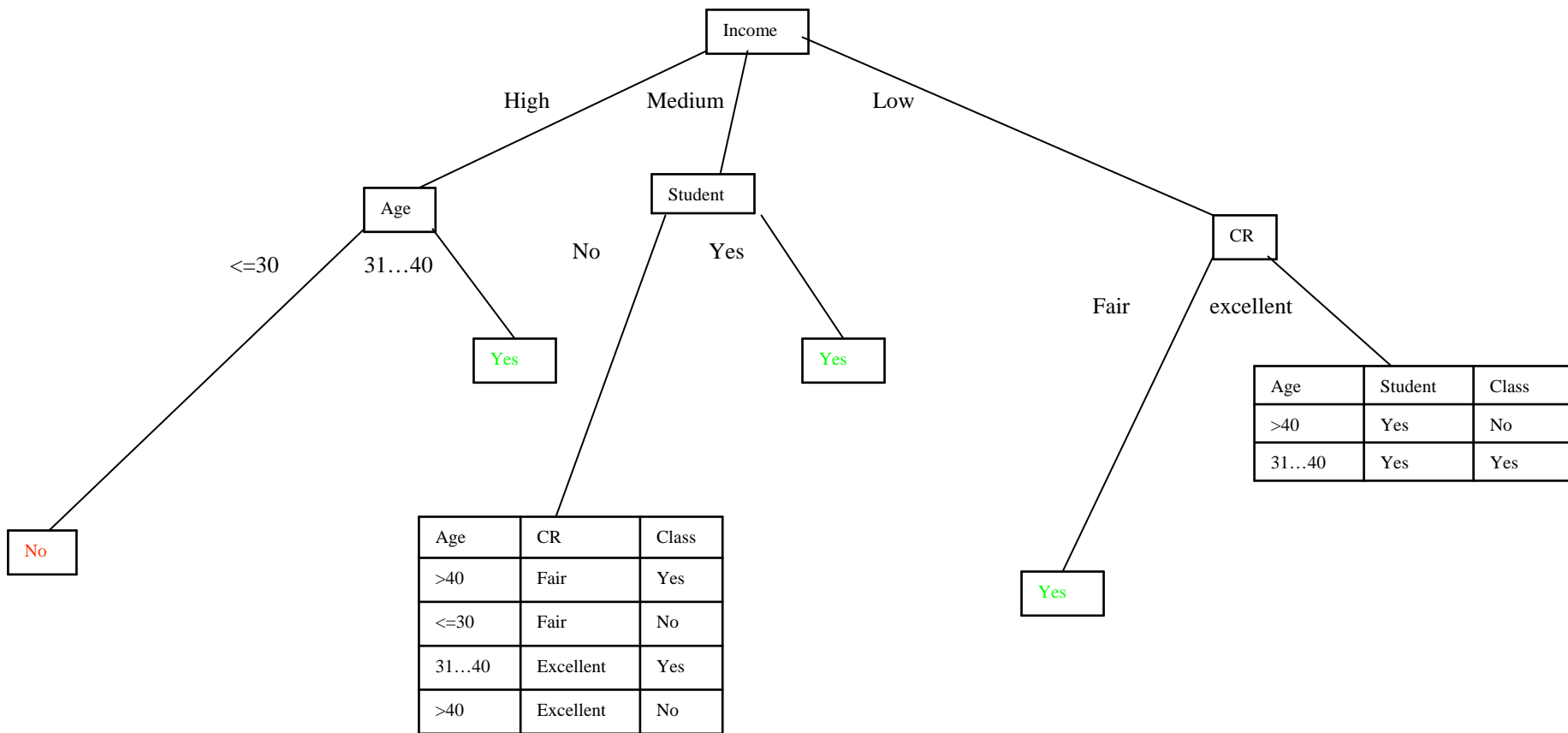
Decision Tree: Root Income



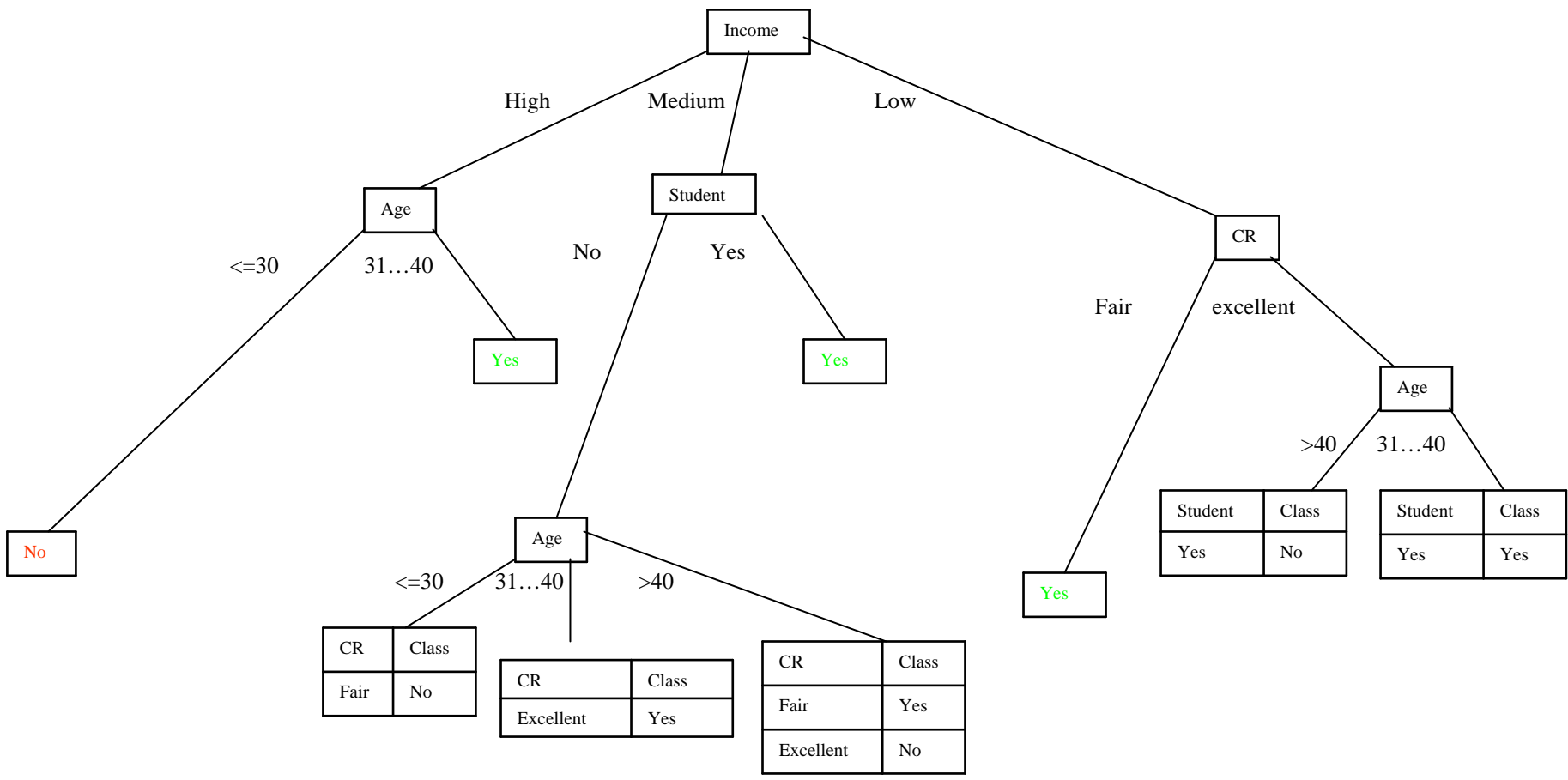
Age	Student	CR	Class
<=30	No	Fair	No
<=30	No	Excellent	No
31...40	No	Fair	Yes
31...40	Yes	Fair	yes

Age	Student	CR	Class
>40	No	Fair	Yes
<=30	No	Fair	No
>40	Yes	Fair	Yes
<=30	Yes	Excellent	yes
31...40	No	Excellent	yes
>40	No	Excellent	no

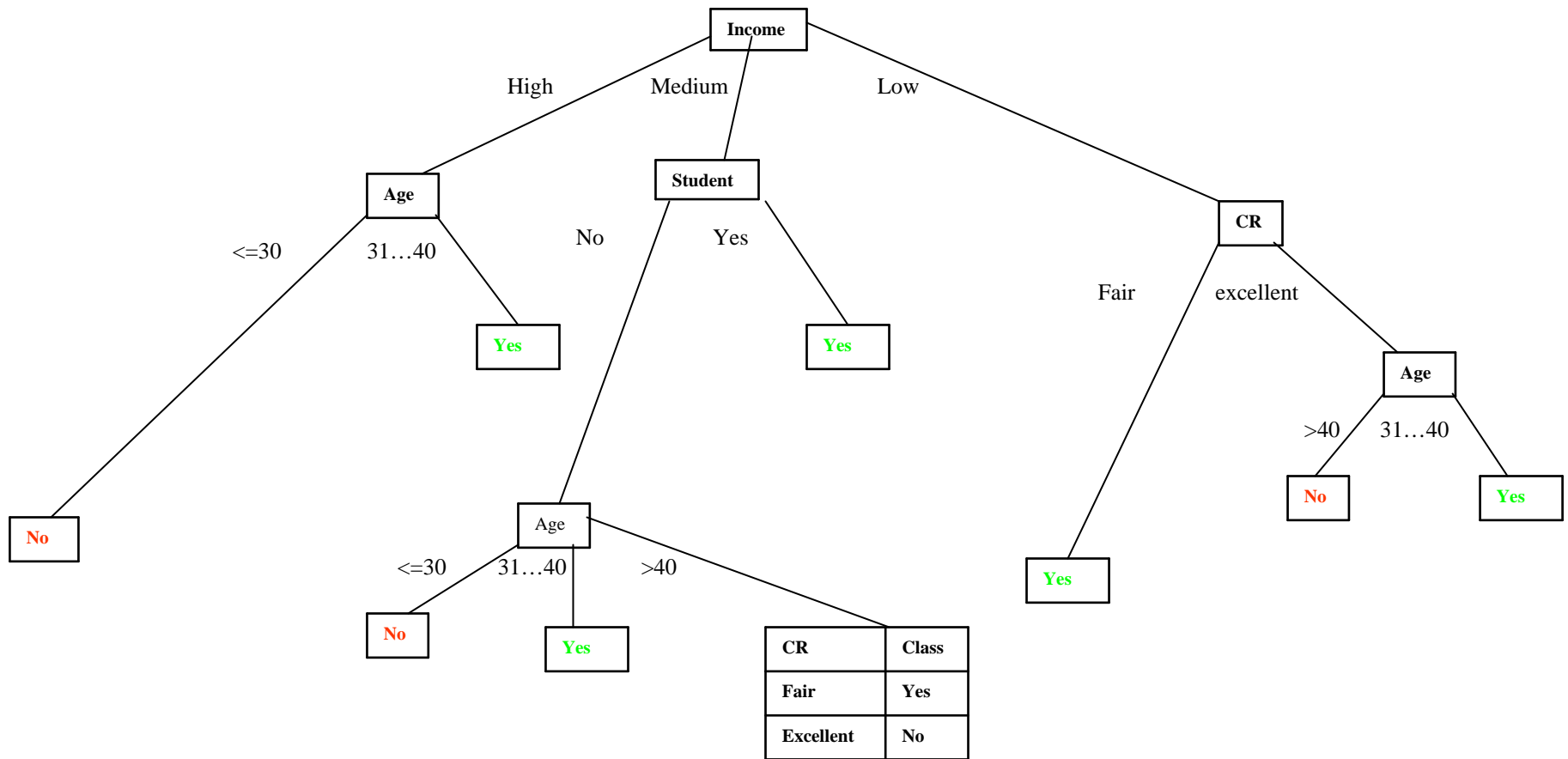
Age	Student	CR	Class
>40	No	Fair	Yes
>40	Yes	Excellent	No
31...40	Yes	Excellent	Yes
<=30	Yes	Fair	yes



Decision Tree next step

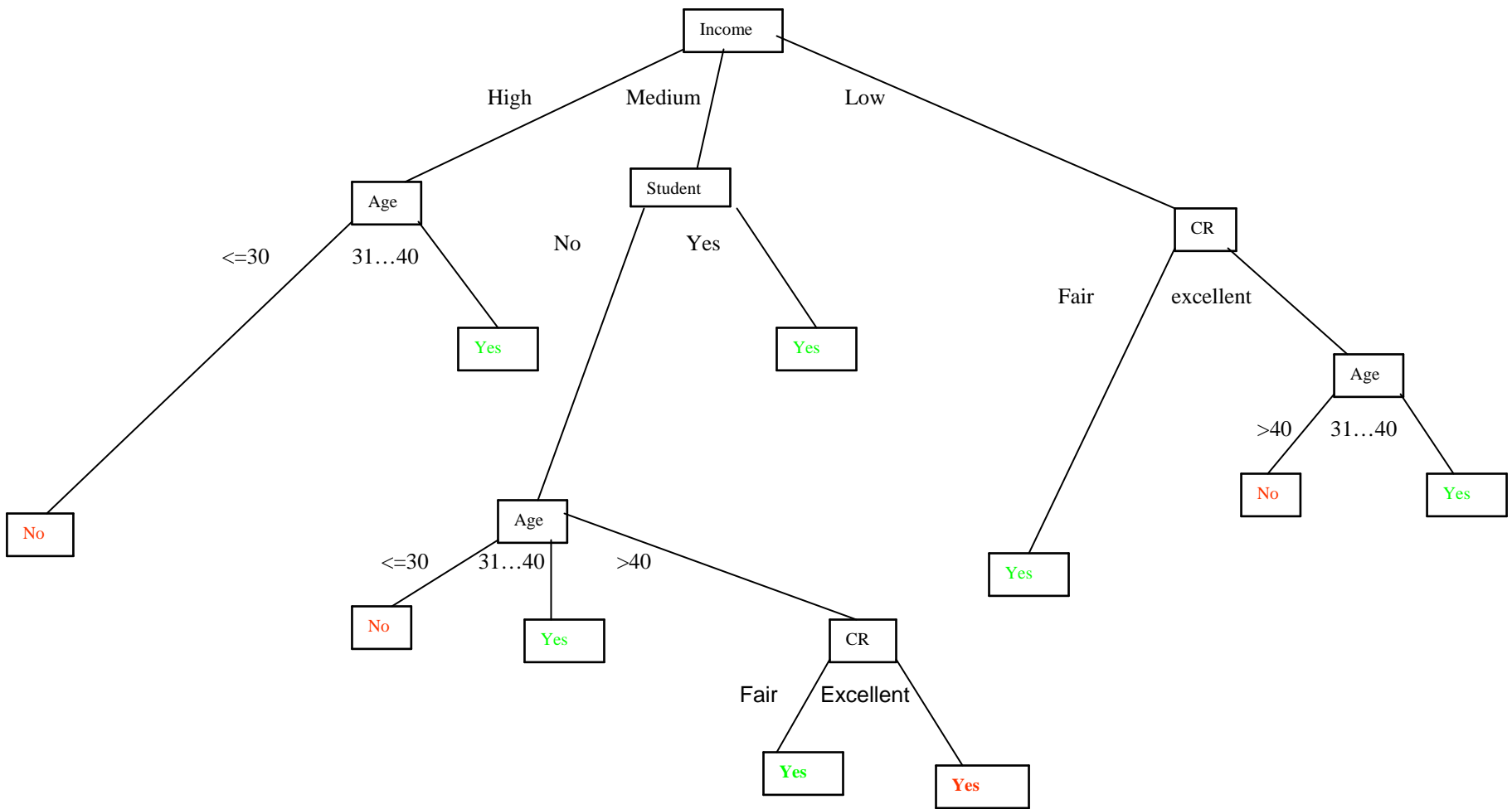


Decision Tree next step



Observe that on the branch age >40 we can't have majority voting; when we create node with the attribute CR both partitions are EMPTY; we create the leaves with majority class in the training data; i.e. YES

Decision Tree next step



Decision Tree last step

Classification by Decision Tree Induction (2)

Crucial point

Good choice of the root attribute and internal nodes attributes is a crucial point. Bad choice may result, in the worst case in a just another knowledge representation: relational table re-written as a tree with class attributes as the leaves.

- **Decision Tree Induction Algorithms differ on methods of evaluating and choosing the root and internal nodes attributes.**

Basic Idea of ID3/C4.5 Algorithm (1)

(book slide)

- The basic algorithm for decision tree induction is a greedy algorithm that constructs decision trees in a top-down recursive divide-and – conquer manner.
- The basic strategy is as follows.
- Tree **STARTS** as a single node representing all training dataset (samples)
- **IF** the samples are ALL in the same class, **THEN** the node becomes a LEAF and is labeled with that class
(or we may apply majority voting or other method to decide the class on the leaf)
- **OTHERWISE**, the algorithm uses an entropy-based measure known as *information gain* as a heuristic for selecting the **ATTRIBUTE** that will best separate the samples into individual classes. This attribute becomes the node-name (test, or tree split decision attribute)

Basic Idea of ID3/C4.5 Algorithm (2)

(book slide)

- **A branch is created** for each value of the node-attribute and **is labeled by this value** and the samples (it means the data table) are partitioned accordingly
- The algorithm uses **the same process recursively** to form a decision tree at each partition. Once an attribute has occurred at a node, it need not be considered in any other of the node's descendents
- The recursive partitioning **STOPS** only when any one of the following conditions is true.

Basic Idea of ID3/C4.5 Algorithm (3)

(book slide)

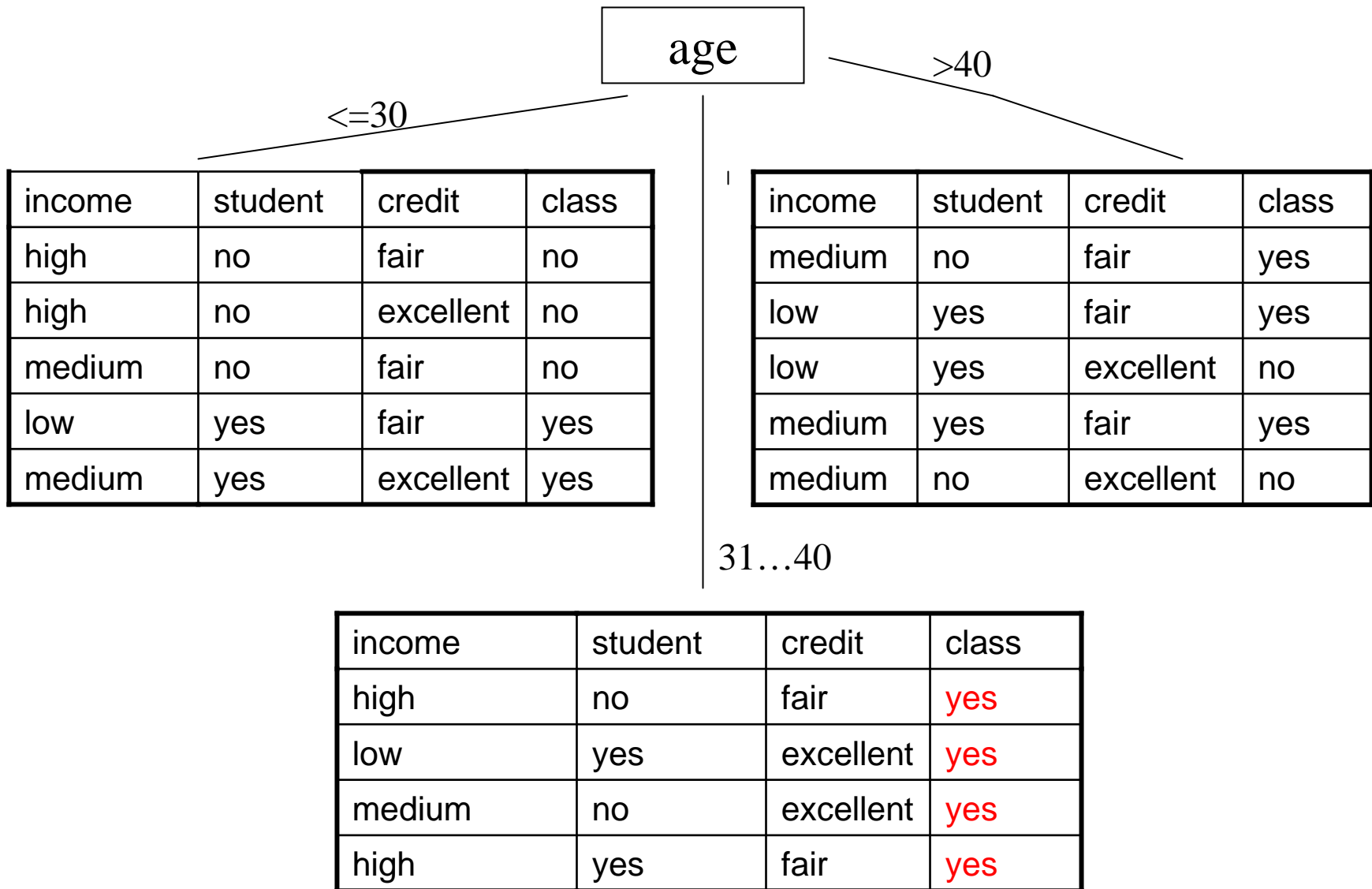
- All records (samples) for the given node belong to the same class **or**
- There are no remaining attributes on which the records (samples) may be further partitioning.
- In this case we convert the given node into a LEAF and label it with the class in majority among samples (*majority voting*)
- There is no records (samples) left – a leaf is created with majority vote for all training sample

Training Dataset (book slide)

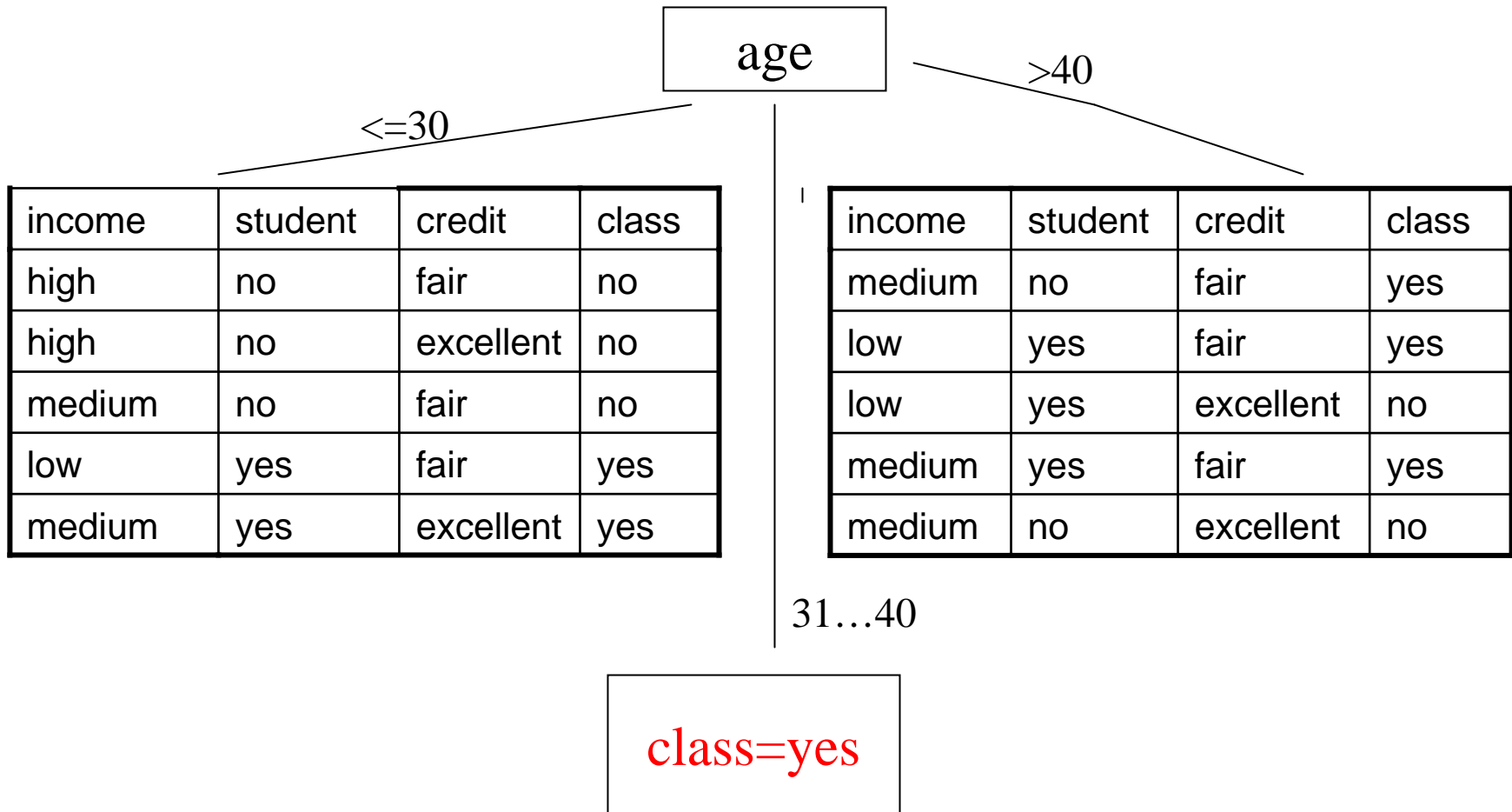
This follows an example from Quinlan's ID3

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
30...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

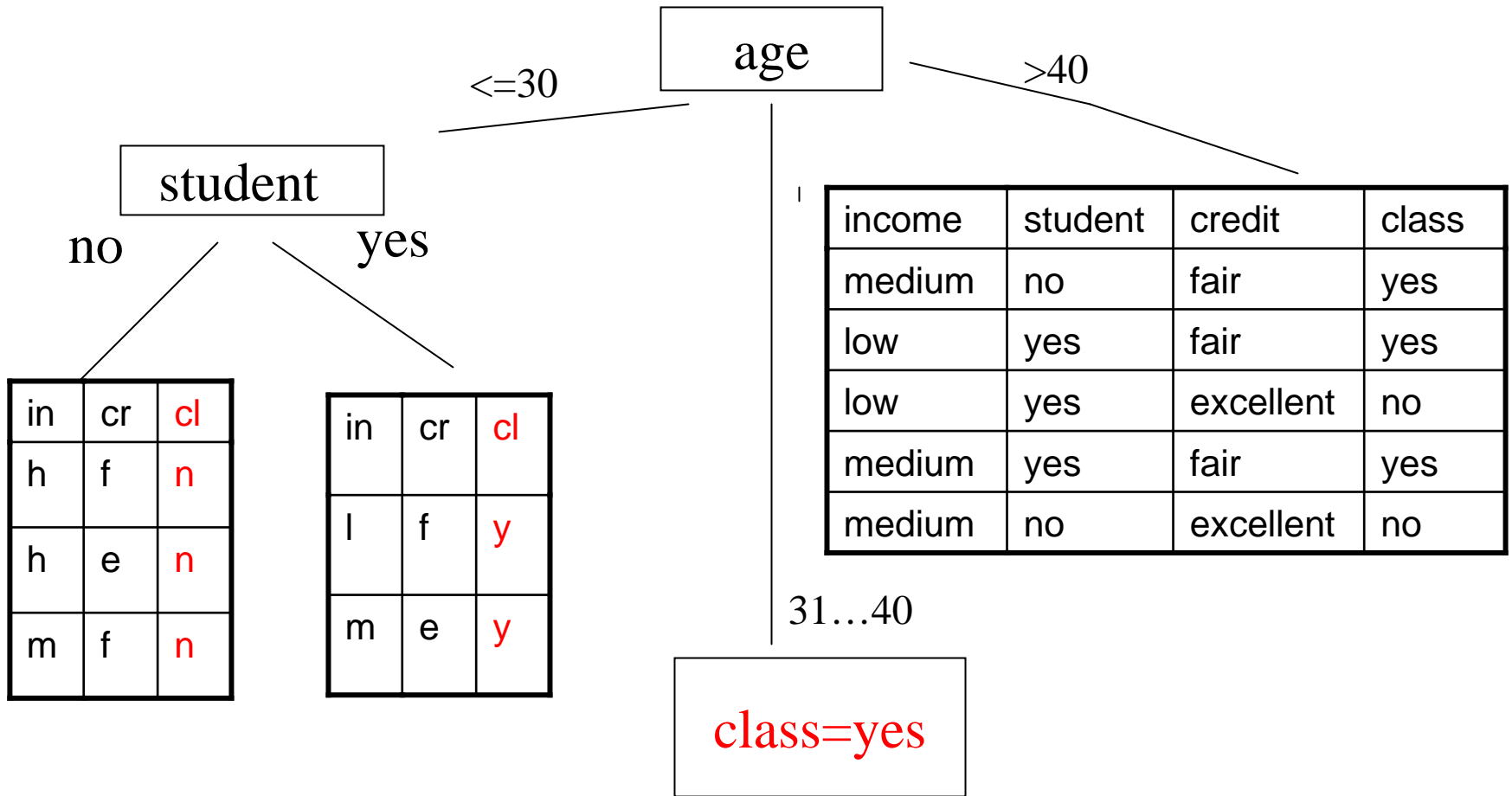
Example: Building The Tree: class attribute “buys”



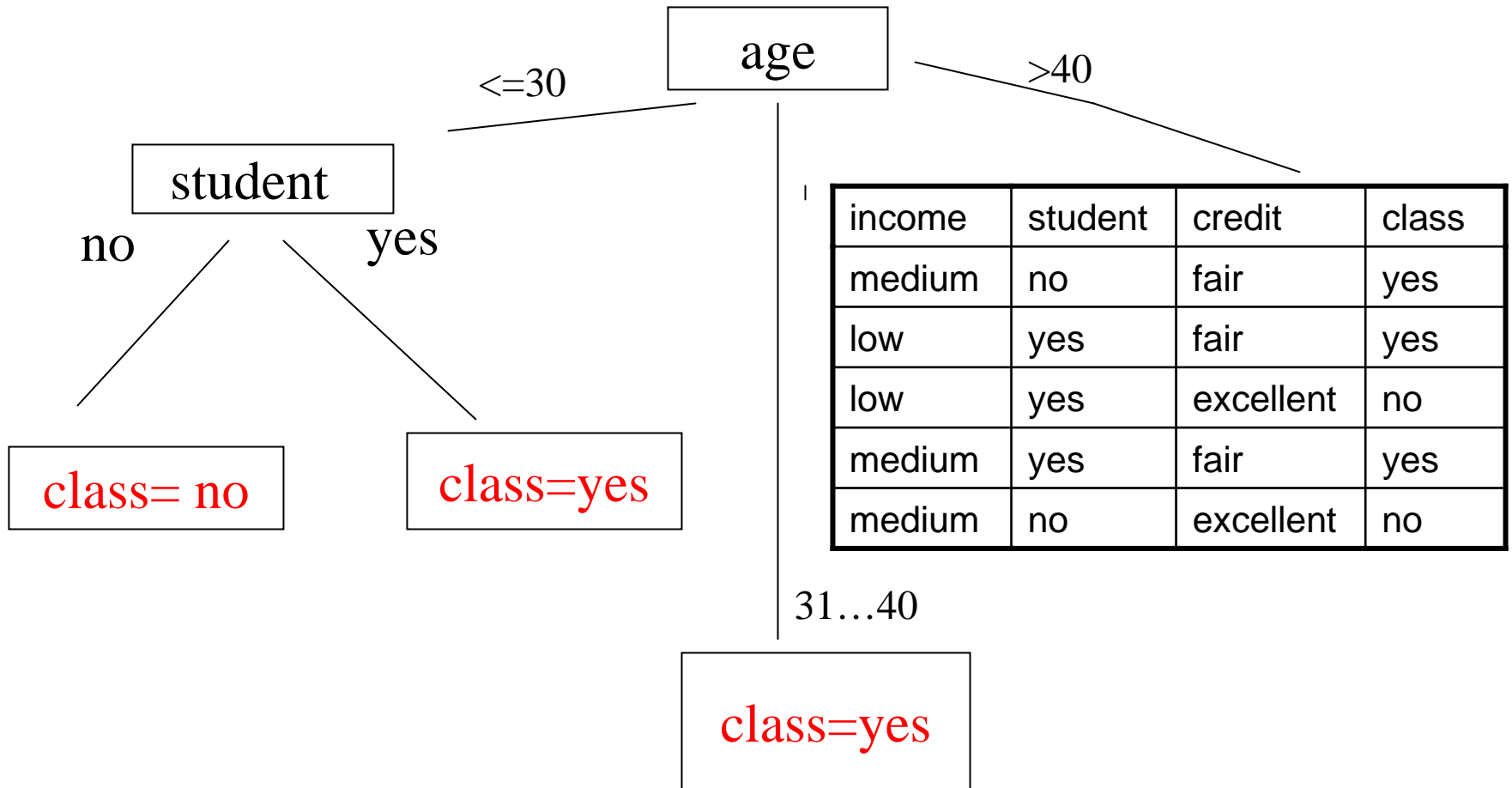
Example: Building The Tree: we chose "age"



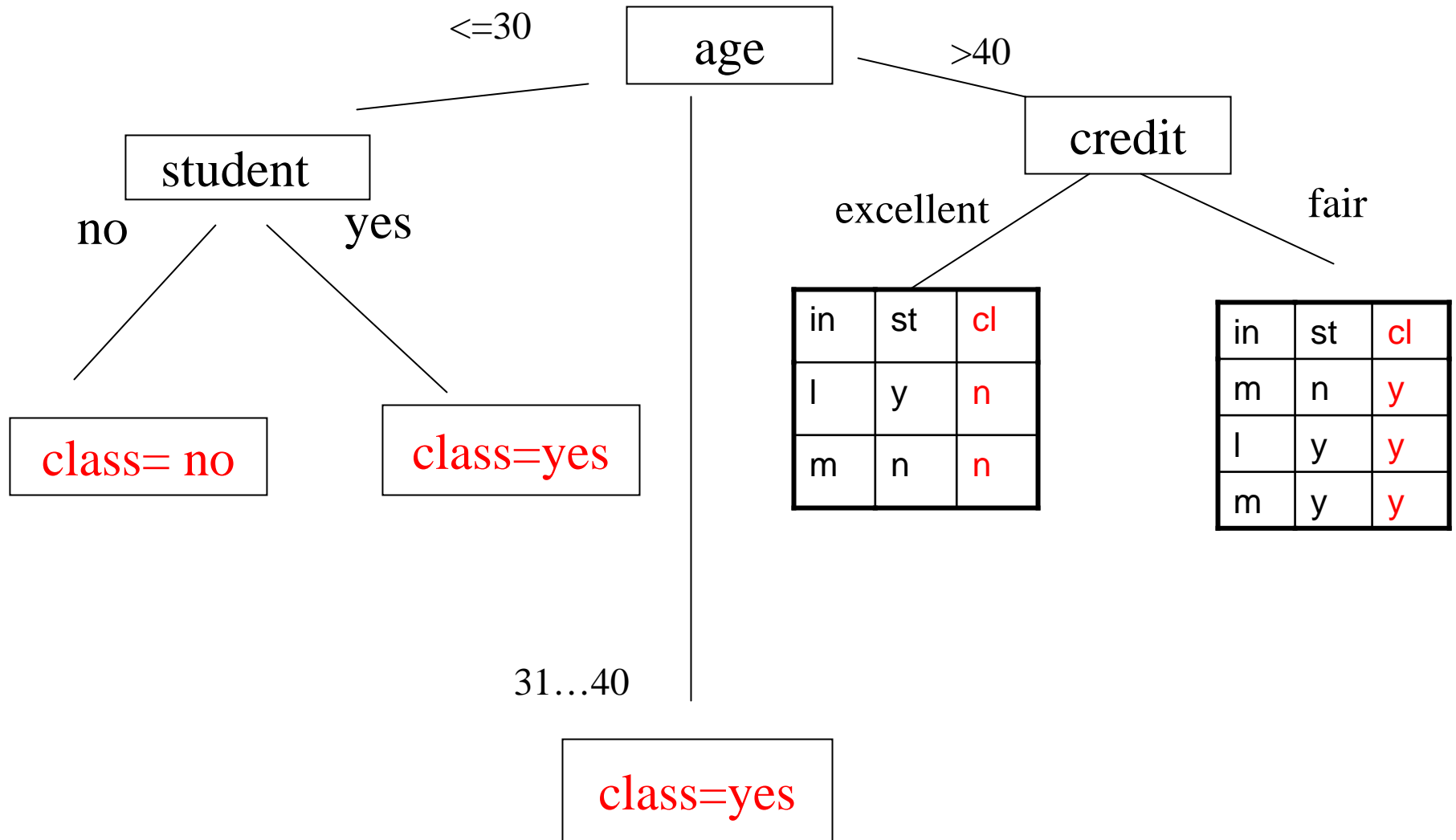
Example: Building The Tree: we chose "student" on ≤ 30 branch



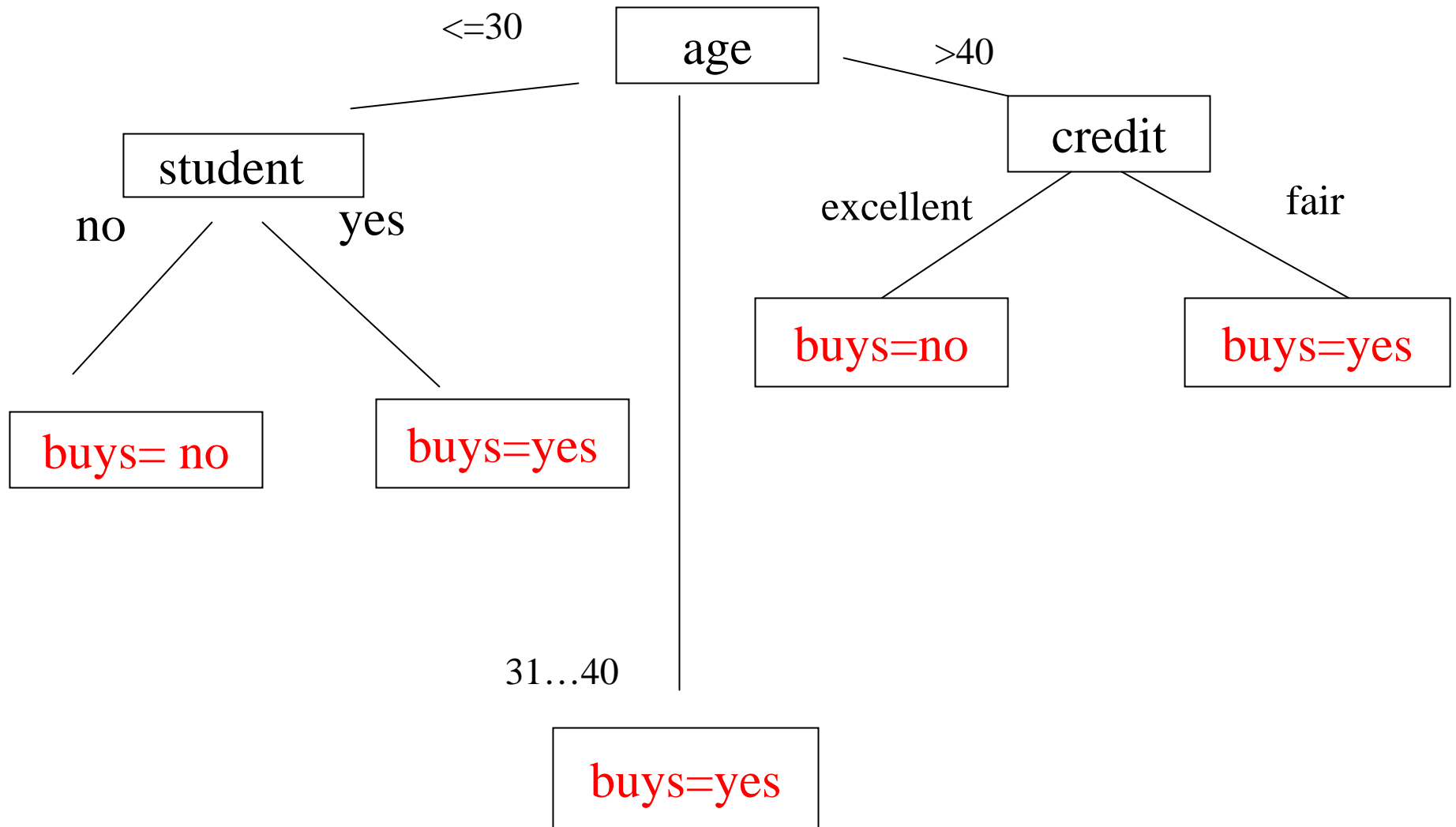
Example: Building The Tree: we chose "student" on ≤ 30 branch



Example: Building The Tree: we chose "credit" on >40 branch



Example: Finished Tree for class="buys"



Heuristics: Attribute Selection Measures

- **Construction of the tree depends on the order in which root attributes are selected.**
- Different choices produce different trees; some better, some worse
- Shallower trees are better; they are the ones in which classification is reached in fewer levels.
- These trees are said to be more efficient as the classification, and hence termination is reached quickly

Attribute Selection Measures

- Given a training data set (set of training samples) there are many ways to choose the root and nodes attributes while constructing the decision tree
- **Some possible choices:**
- Random
- Attribute with smallest/largest number of values
- Following certain order of attributes
- **We present here a special order: information gain as a measure of the goodness of the split**
- The attribute with the highest information gain is always chosen as the split decision attribute for the current node while building the tree.

Information Gain Computation (ID3/C4.5)

- ID3 uses **INFORMATION GAIN** as its attribute selection measure.
- This measure is based on pioneering work by Claude Shannon (yes, the same who invented Boolean representation for circuits) on information theory.
- Let node N represent or hold the tuples of partition (data table) D.
- **The attribute with the highest information gain is chosen as the splitting attribute for the node N.**
- **This attribute MINIMIZES the expected number of tests needed to classify the given tuples (records) in the resulting partition and reflects the least randomness, or “impurity” in these partitions.**

Information Gain Computation (ID3/C4.5)

- The **expected information** needed to classify a record in D is given by

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

- Where p_i is the probability that an arbitrary record in D belongs to class C_i and is estimated by the formula

$$|C_i(D)|/|D|$$

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

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

At this point the information we have is based solely on the proportions of records of each class.

Info(D) is also known as **ENTROPY** and denoted by **E(D)**

Information Gain Computation (ID3/C4.5)

- Assume that using attribute **A** a set **D** will be partitioned into sets $\{D_1, D_2, \dots, D_v\}$ (v is number of values of the attribute A).
- These partitions would correspond to the branches grown from the node N.
- Ideally, we would like this partitioning to produce an exact classification of the records; i.e. each partition to be **pure** (all record belong to one class).
- Usually, the some partitions are **impure**, i.e. contain record from different classes.
- **QUESTION:** how much more information would we still need (after partitioning) in order to arrive at an exact classification?

Information Gain Computation (ID3/C4.5)

- The measure of information needed after partitioning to arrive at an exact classification is given by

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

- The term

$$\frac{|D_j|}{|D|}$$

acts as a weight of the j th partition.

We use symbol $I(D_j)$ for $Info(D_j)$ as defined before.

- **$Info_A(D)$ is the expected information required to classify a record from D based on the partitioning by A .**
- The smaller the expected information (still) required the greater the **purity** of the partition.
- **INFORMATION GAIN** is hence defined by a formula:
- $Gain(A) = Info(D) - Info_A(D)$
- $Gain(A)$ tells us how much would be gained by branching on A .
- The attribute A with the highest information gain $Gain(A)$ is chosen as the splitting attribute at node N .

Attribute Selection Measure: Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i , estimated by $|C_{i,D}|/|D|$
- **Expected information** (entropy) needed to classify a tuple in D :

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

- **Information** needed (after using A to split D into v partitions) to classify a tuple in D :

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

- **Information gained** by branching on attribute A

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

Information Gain Computation (ID3/C4.5): Case of Two Classes

- Assume there are two classes, P (positive) and N (negative)

Let S be a training data set D consisting of s examples (records):

$$|S|=s$$

And S contains p elements of class P and n elements of class N

The amount of information $I(p,n) = Info(D)$, needed to decide if an arbitrary example (record) in S belongs to P or N is defined as

$$I(p, n) = -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n}$$

Information Gain Measure

- Assume that using attribute **A** a set S will be partitioned into sets $\{D_1, D_2, \dots, D_v\}$ (v is number of values of the attribute A)

If D_i contains p_i examples of P and n_i examples of N , the expected information $Info_A(D)$, or entropy $E(A)$, needed to classify objects in all sub-trees D_i is

$$E(A) = \sum_{i=1}^v \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

- The encoding information that would be **gained** by branching on A is

$$Gain(A) = I(p, n) - E(A)$$

Attribute Selection: Information Gain

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

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

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

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

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

Similarly,

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

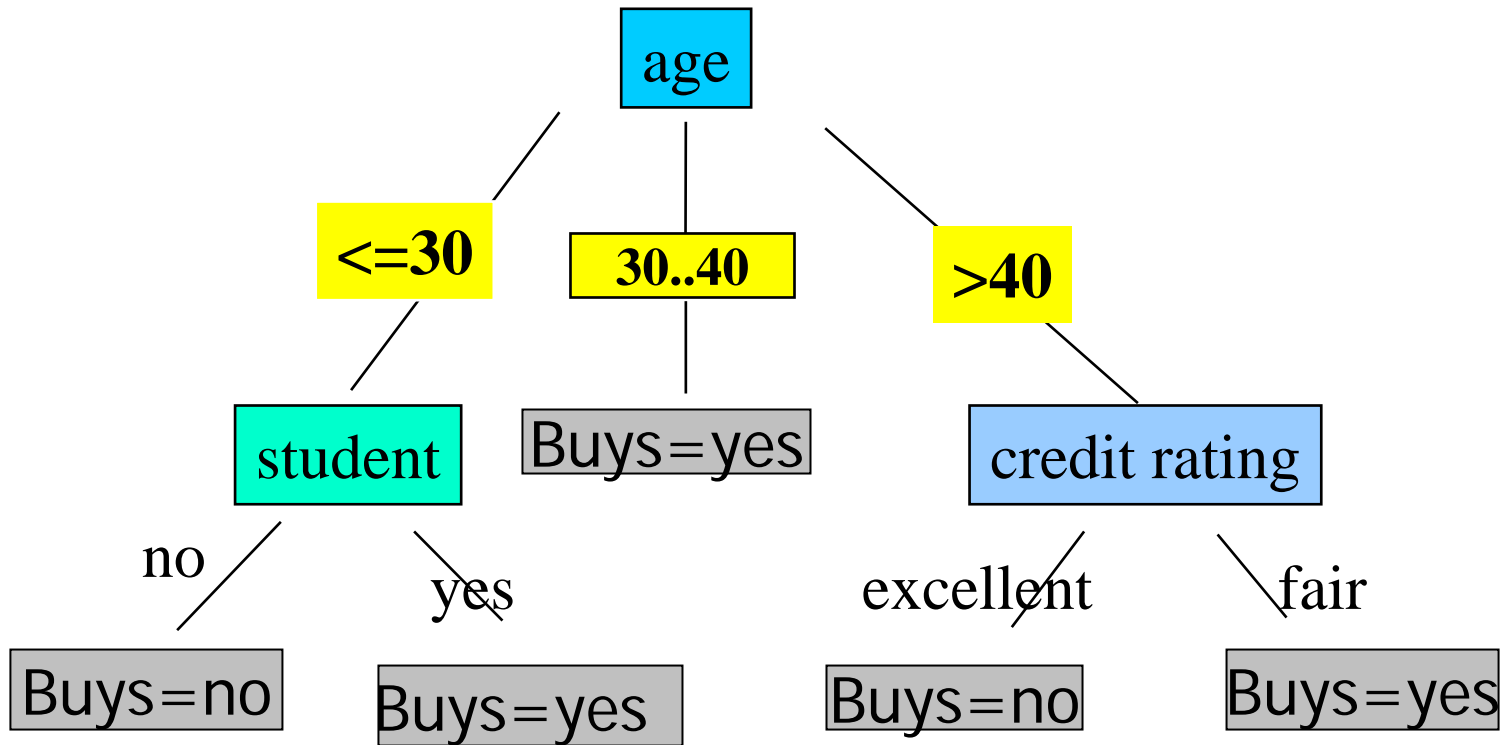
$$Gain(credit_rating) = 0.048$$

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

Extracting Classification Rules from Trees

- **Goal:** 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

The tree to extract rules from (book slide)



Extracting Classification Rules from Trees

(book slide)

- The rules are:

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 = “no”

IF *age* = “>40” AND *credit_rating* = “fair” THEN
buys_computer = “yes”

Rules format for testing and applications

- In order to use rules for testing, and later when testing is done and predictive accuracy is acceptable we write rules in a predicate form:

IF *age*(*x*, ≤ 30) AND *student*(*x*, *no*) THEN

buys_computer (*x*, *no*)

IF *age*(*x*, ≤ 30) AND *student* (*x*, *yes*) THEN

buys_computer (*x*, *yes*)

- Attributes and their values of the new record *x* are matched with the IF part of the rule and the record is classified accordingly to the THEN part of the rule.