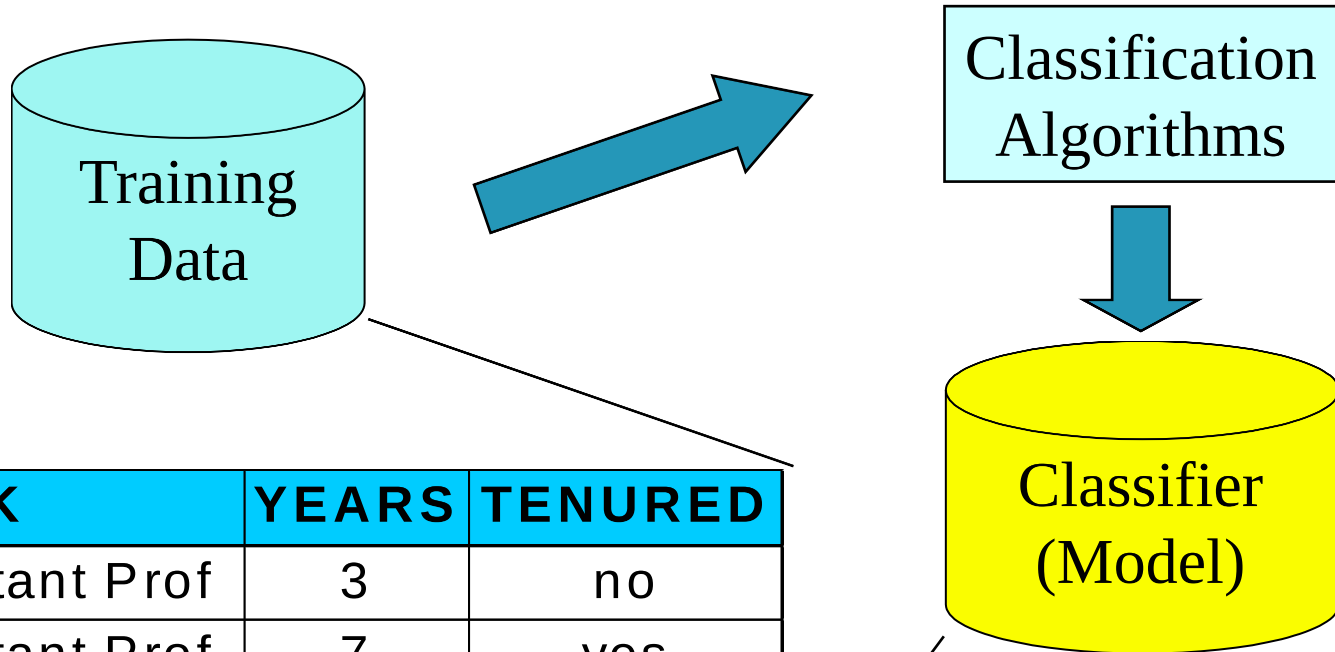
Classification—A Two-Step Process

- Step 1 - Model construction
  - describe a set of predetermined classes
    - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
    - The set of tuples used for model construction is the training set
    - The model is represented as classification rules, decision trees, or mathematical formulae
- Step 2 - Model usage
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set
  - Use model to classify future or unknown objects

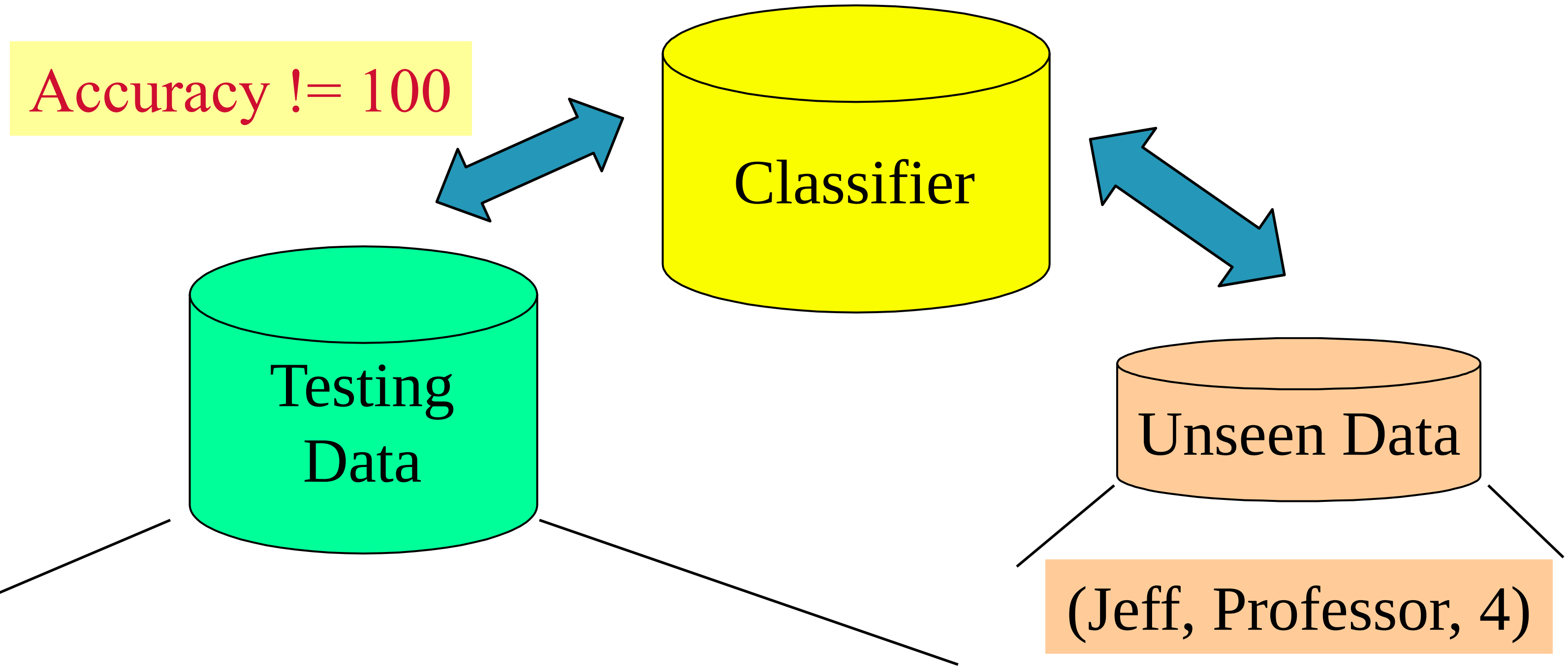
Classification Process (1): Model Construction



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

Classification Process (2): Use the Model in Prediction



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

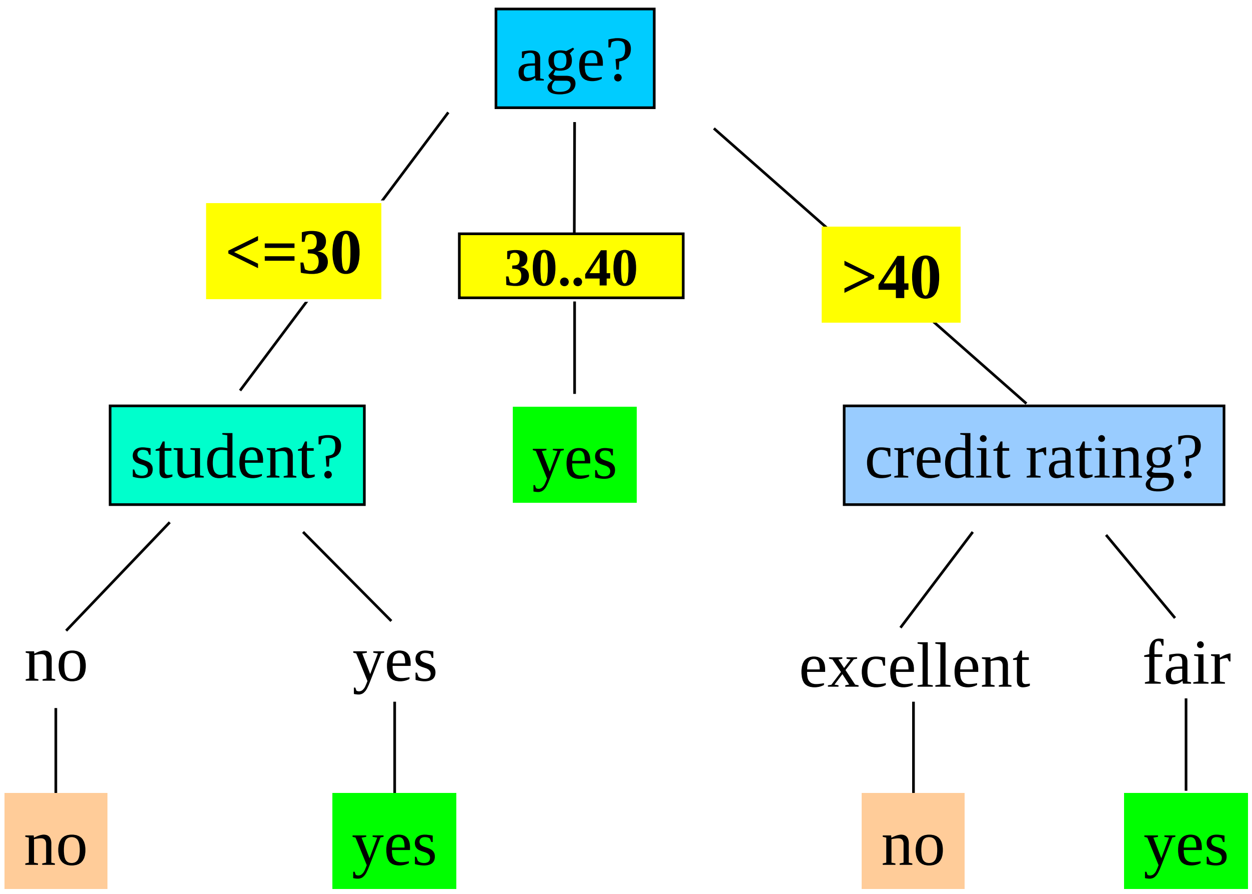
Classification by Decision Tree Induction

- 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

Training Dataset

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

Example: A Decision Tree for "buys_computer"



Non-leaf nodes – test on an attribute

Leaf nodes – class (buys_computer)

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

Algorithm for Decision Tree Induction (continued)

- Basic algorithm (generate_decision_tree)
  - Create a node N
  - If *samples* are all of the same class, C then
    - Return N as a leaf node labeled with the class C
  - If *attribute-list* is empty then
    - Return N as a leaf node labeled with most common class in *sample*
  - Select test-attribute, the attribute with highest info gain from *attribute-list*
  - Label node N with *test-attribute*
  - For each known value a_i of *test-attribute*
    - Grow a branch from node N for the condition *test-attribute*=a_i
    - Let s_i be the set of samples in *samples* for which *test-attribute*=a_i
    - If s_i is empty then
      - Attach a leaf labeled with the most common class in *samples*
    - Else attach the node returned by generate_decision_tree(s_i, *attribute-list*)

Information Gain (attribute selection measure)

- Select the attribute with the highest information gain
- Assume there are two classes, *P* and *N*
  - Let the set of examples *S* contain *p* elements of class *P* and *n* elements of class *N*
  - The amount of information , needed to decide if an arbitrary example in *S* belongs to *P* or *N* is defined as or expected information to classify a tuple:-

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

Information Gain in Decision Tree Induction

- Assume that using attribute A, a set *S* will be partitioned into sets {S_1, S_2, …, S_v}
  - If S_i contains p_i examples of *P* and n_i examples of *N*, the entropy, or the **expected information needed to classify objects in all sub-trees** S_i is

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

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

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

Attribute Selection by Information Gain Computation

- Class P: buys_computer = “yes”
- Class N: buys_computer = “no”
- I(p, n) = I(9, 5) =0.940
- Compute the entropy for *age*:

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

$$E(age) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.69$$

Hence

$$Gain(age) = I(p,n) - E(age) = .25$$

Similarly

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

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*"

Bayesian Classification: Why?

- Probabilistic learning: Calculate explicit probabilities for hypothesis, among the most practical approaches to certain types of learning problems
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct. Prior knowledge can be combined with observed data.
- Probabilistic prediction: Predict multiple hypotheses, weighted by their probabilities
- Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

Play-tennis example: estimating $P(x_i|C)$

Outlook	Temperature	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

2 classes – p (play), n (don't play)

P(p) = 9/14	
P(n) = 5/14	

outlook	
P(sunny | p) = 2/9	**P(sunny | n) = 3/5**
P(overcast | p) = 4/9	**P(overcast | n) = 0**
P(rain | p) = 3/9	**P(rain | n) = 2/5**
temperature	
P(hot | p) = 2/9	**P(hot | n) = 2/5**
P(mild | p) = 4/9	**P(mild | n) = 2/5**
P(cool | p) = 3/9	**P(cool | n) = 1/5**
humidity	
P(high | p) = 3/9	**P(high | n) = 4/5**
P(normal | p) = 6/9	**P(normal | n) = 2/5**
windy	
P(true | p) = 3/9	**P(true | n) = 3/5**
P(false | p) = 6/9	**P(false | n) = 2/5**

ASSIGNMENT

- KDD For Insurance Risk Assessment: A Case Study

✓Decision tree techniques to identify significant areas of risk within an insurance portfolio.

✓The real world dataset used contains information about policies and insurance claims on those policies.

✓Historical data is used to estimate parameters of the model