Ch. 8 – Classification: Basic Concepts

Prediction Problems: Classification vs. Numeric Prediction

- Classification
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Numeric Prediction
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical applications
  - Credit/loan approval: if a customer is good or bad credit risk
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is

Classification—A Two-Step Process

- Model construction: describing 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 training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - 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 (otherwise overfitting)
  - If the accuracy is acceptable, use the model to classify new data

Note: If *the test set* is used to select models, it is called validation (test) set

6

class label attribute

Predicting/Condition

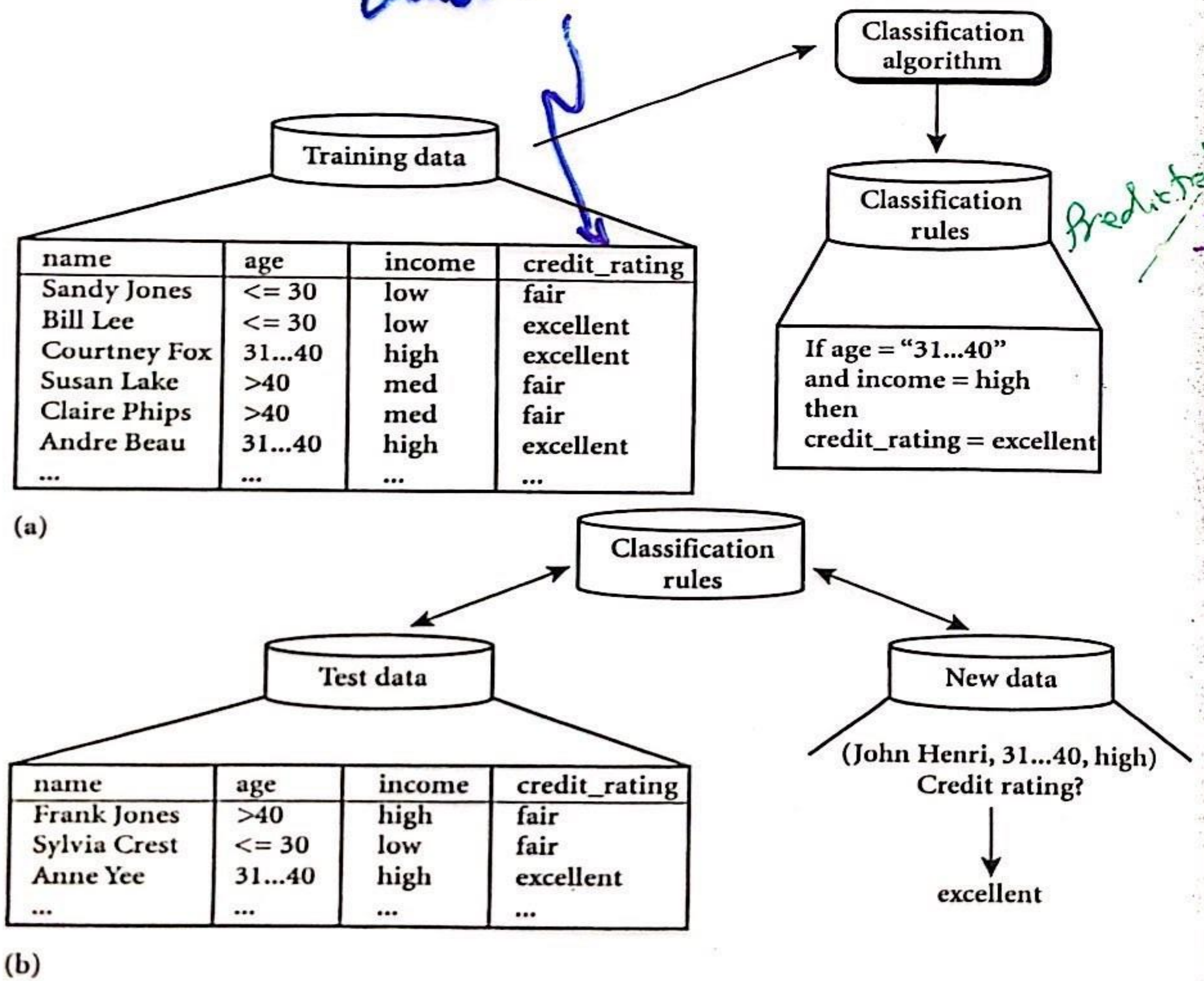
name

age

income

Predicted/Decision.

credit-rating



name	age	income	credit_rating
Sandy Jones	<= 30	low	fair
Bill Lee	<= 30	low	excellent
Courtney Fox	31...40	high	excellent
Susan Lake	>40	med	fair
Claire Phips	>40	med	fair
Andre Beau	31...40	high	excellent
...	...	...	...

name	age	income	credit_rating
Frank Jones	>40	high	fair
Sylvia Crest	<= 30	low	fair
Anne Yee	31...40	high	excellent
...	...	...	...

8

Figure 7.1 The data classification process: (a) *Learning:* Training data are analyzed by a classification algorithm. Here, the class label attribute is *credit_rating*, and the learned model or classifier is represented in the form of classification rules. (b) *Classification:* Test data are used to estimate the accuracy of the classification rules. If the accuracy is considered acceptable, the rules can be applied to the classification of new data tuples.

Decision Tree Methods

Quinlan

ID3 — Iterative Dichotomizer

C4.5

→ Information Gain

→ Gain Ratio

CART

- Expected Inform (still required)
- Entropy

purity

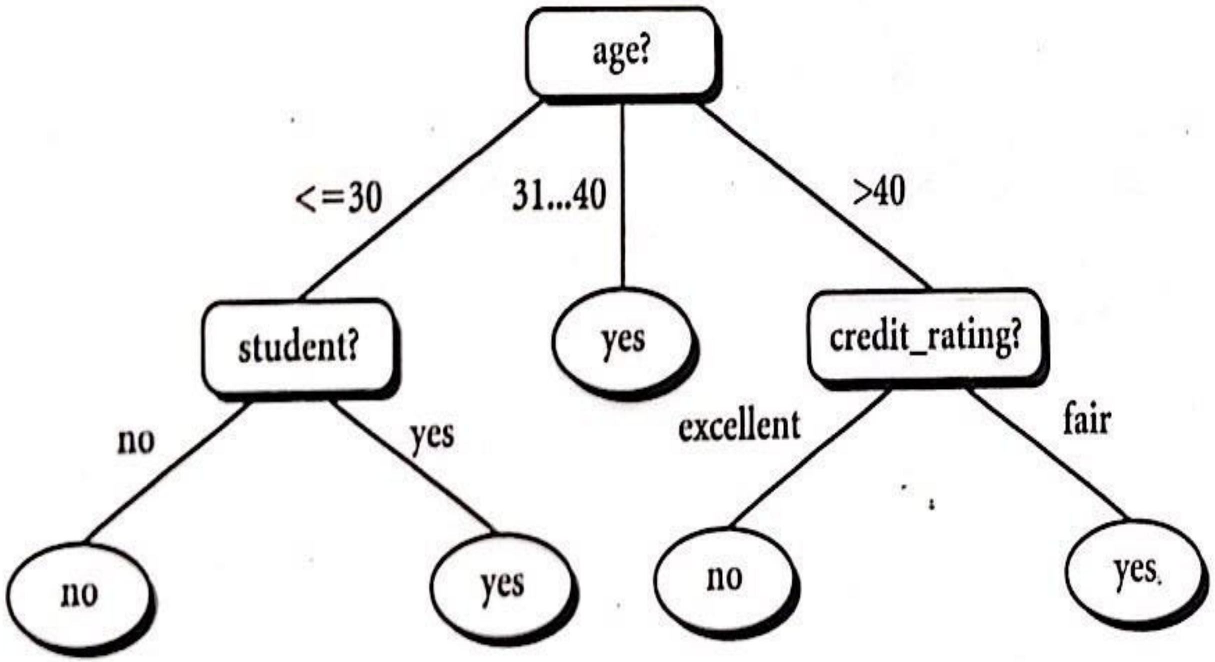
$\boxed{D}$ → C_1, C_2

Major

D_1 D_2 D_3

$|C_{1,D_1}|$ $|C_{1,D_2}|$ $|C_{1,D_3}|$

$|C_{2,D_1}|$ $|C_{2,D_2}|$



9.2
Figure ~~7~~.2 A decision tree for the concept *buys_computer*, indicating whether or not a customer at *AllElectronics* is likely to purchase a computer. Each internal (nonleaf) node represents a test on an attribute. Each leaf node represents a class (either *buys_computer* = *yes* or *buys_computer* = *no*).

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

Input:

- Data partition, D, which is 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. This criterion consists of a *splitting_attribute* and, possibly, either a *split point* or *splitting subset*.

Output: A decision tree.

Method:

(1) create a node N;
(2) **if** tuples in D are all of the same class, C **then**
(3)   return N as a leaf node labeled with the class C;
(4) **if** *attribute_list* is empty **then**
(5)   return N as a leaf node labeled with the majority class in D; // majority voting
(6) apply Attribute_selection_method(D, *attribute_list*) to **find** the "best" *splitting_criterion*;
(7) label node N with *splitting_criterion*;
(8) **if** *splitting_attribute* is discrete-valued **and**
    multiway splits allowed **then** // not restricted to binary trees
(9)   *attribute_list* ← *attribute_list* − *splitting_attribute*; // remove *splitting_attribute*
(10) **for each** outcome j of *splitting_criterion*
    // partition the tuples and grow subtrees for each partition
(11)   let D_j be the set of data tuples in D satisfying outcome j; // a partition
Dj
(12)   **if** D_j is empty **then**
(13)   attach a leaf labeled with the majority class in D to node N;
(14)   **else** attach the node returned by Generate_decision_tree(D_j, *attribute_list*) to node N;
    endfor
(15) return N;

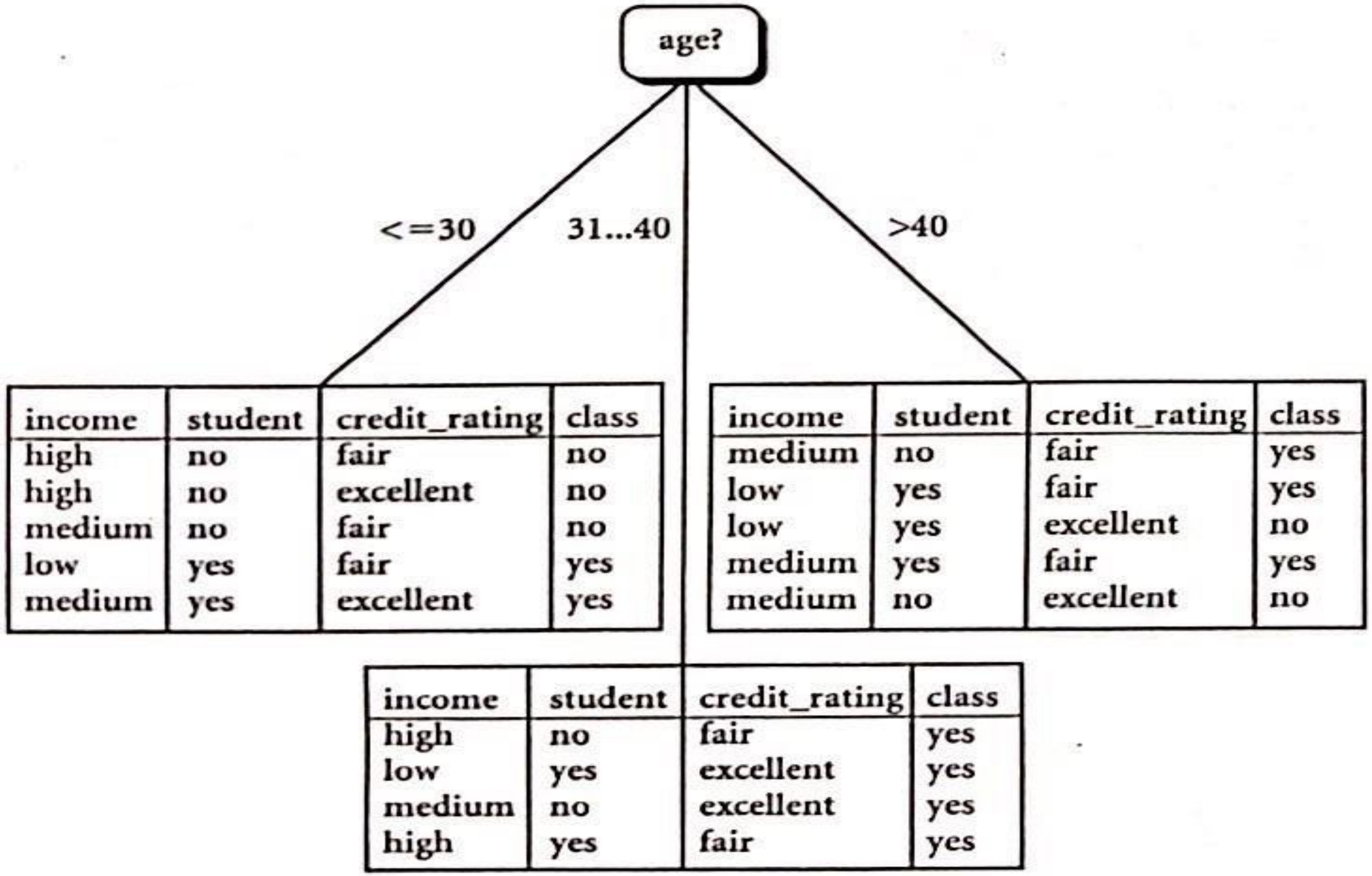
8

Figure 6.3 Basic algorithm for inducing a decision tree from training tuples.

	age			income			student		credit Rat.	
	age<30	31…40	>40	high	med	Low	yes	no	fair	excell
$X_2 =$	1	0	0	1	0	0	0	1	0	1

Table 7.1 Training data tuples from the *AllElectronics* customer database.

RID	*age*	*income*	*student*	*credit_rating*	*Class: buys_computer*
1	<=30	high	no	fair	no
2	<=30	high	no	excellent	no
3	31 . . . 40	high	no	fair	yes
4	>40	medium	no	fair	yes
5	>40	low	yes	fair	yes
6	>40	low	yes	excellent	no
7	31 . . . 40	low	yes	excellent	yes
8	<=30	medium	no	fair	no
9	<=30	low	yes	fair	yes
10	>40	medium	yes	fair	yes
11	<=30	medium	yes	excellent	yes
12	31 . . . 40	medium	no	excellent	yes
13	31 . . . 40	high	yes	fair	yes
14	>40	medium	no	excellent	no



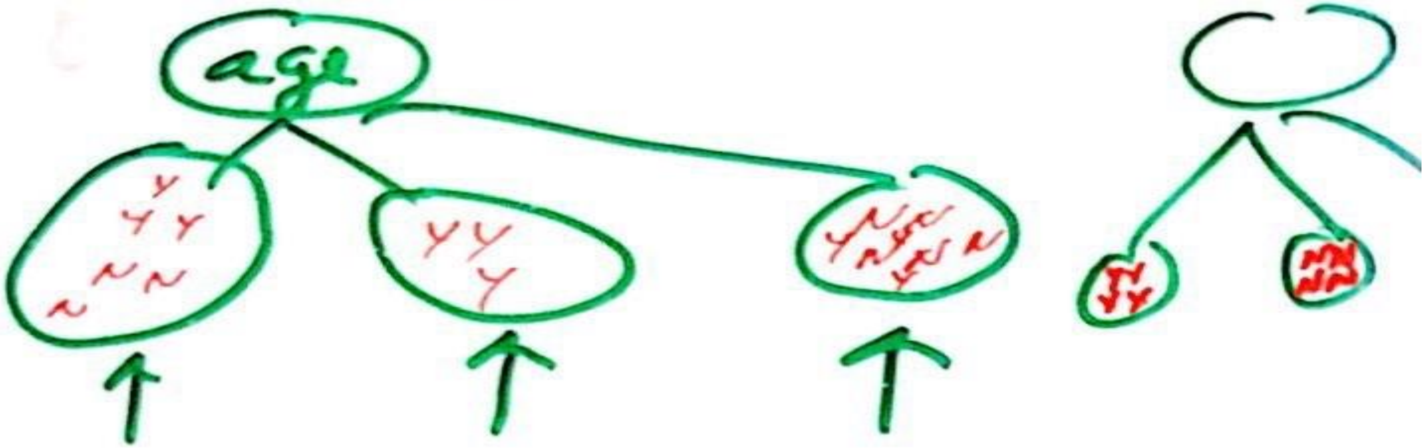
income	student	credit_rating	class
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

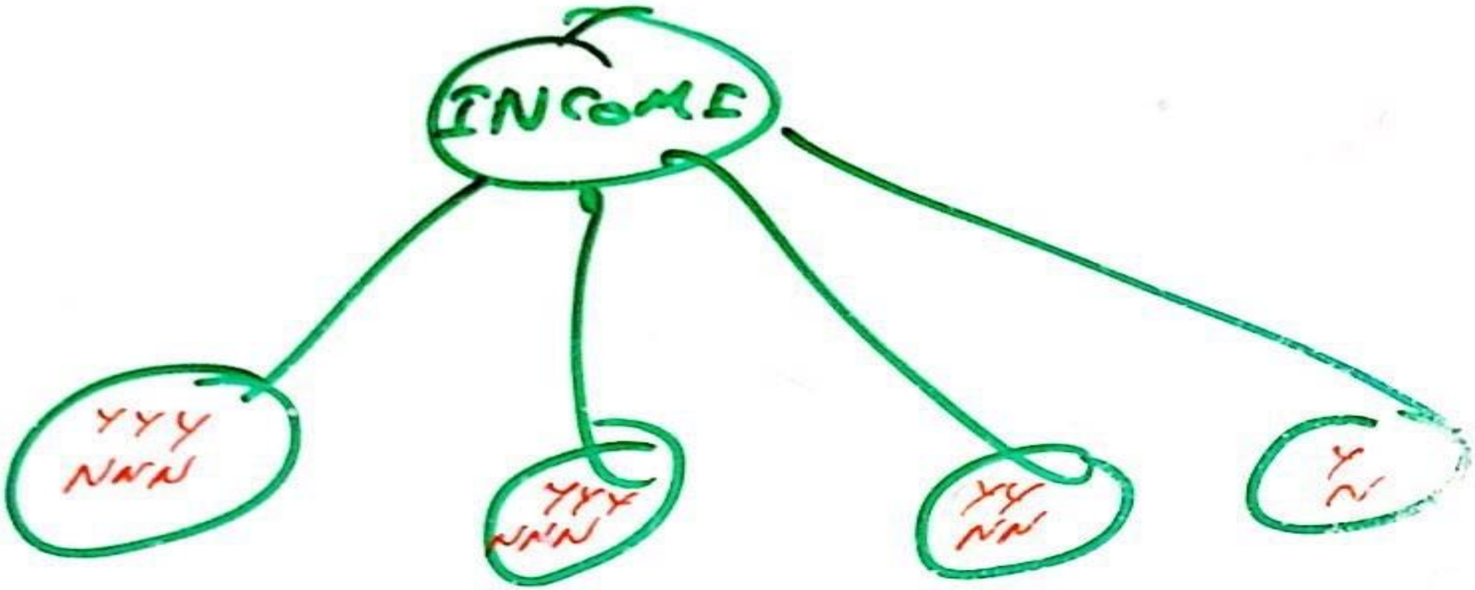
income	student	credit_rating	class
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

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

Figure 7.4 The attribute *age* has the highest information gain and therefore becomes a test attribute at the root node of the decision tree. Branches are grown for each value of *age*. The samples are shown partitioned according to each branch.

PURITY





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 D:

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

- Information gained by branching on attribute A

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

Attribute Selection: Information Gain

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

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

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

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

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

$\frac{5}{14}I(2,3)$ 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,

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

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

$\frac{5}{14}\log\frac{5}{14}$
0.5596

$\frac{9}{14}\log_{10}\frac{9}{14}$
$= 0.1233$

$\text{Inform Gain}_A(D)$

$= \text{Info}(D) - \text{Exp. Info}_A(D)$

$|C_{1,D}| = 9$ $\quad I(9,5)$

$|C_{2,D}| = 5$

$-\sum p_i \log(p_i)$

$-\frac{9}{14}\log\frac{9}{14} - \frac{5}{14}\log\frac{5}{14} = 0.94$

$|D_1| = 5 \qquad |D_2| = 4 \qquad |D_3| = 5 \leftarrow D_j$

$\text{Info}(D_1) = I(2,3) = -\frac{2}{5}\log\frac{2}{5} - \frac{3}{5}\log\frac{3}{5}$
$= 0.971$

$\text{Info}(D_2) = 0$

$\text{Info}(D_3) = I(3,2) = 0.971$

$\text{Info}_{age}(D) = \sum_{j=1}^{3} \frac{|D_j|}{|D|} \text{Info}(D_j)$

$= \frac{5}{14} \times 0.971 + 0 + \frac{5}{14} \times 0.971$
0.694

$\text{Info Gain}_{age}(D) = 0.94 - 0.694 = 0.246$

$$\log_a x = \frac{\log_{10} x}{\log_{10} a}$$

$$\log_{10} 2 = 0.3010$$

Computing Information-Gain for Continuous-Valued Attributes

Let attribute A be a continuous-valued attribute

Must determine the *best split point* for A

- Sort the value A in increasing order
- Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
  - $(a_i+a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}
- The point with the *minimum expected information requirement* for A is selected as the split-point for A

Split:

- D1 is the set of tuples in D satisfying A ≤ split-point, and D2 is the set of tuples in D satisfying A > split-point

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)

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

- GainRatio(A) = Gain(A)/SplitInfo(A)
  - Ex.

$$SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 1.557$$

gain_ratio(income) = 0.029/1.557 = 0.019

- The attribute with the maximum gain ratio is selected as the splitting attribute

$$SplitInfo_{age}(D) = I(5,4,5)$$

$$= -\frac{5}{14}\log\frac{5}{14} - \frac{4}{14}\log\frac{4}{14} - \frac{5}{14}\log\frac{5}{14}$$

$$= 1.4070$$

$$GainRatio_{age}(D) = 0.240 / 1.4070$$

$$= 0.175$$

Gini Index (CART, IBM IntelligentMiner)

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:

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

The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node (*need to enumerate all the possible splitting points for each attribute*)

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

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"

Naive Bayes Method

Bayes' Theorem: Basics

Total probability Theorem: $P(B) = \sum_{i=1}^{M} P(B|A_i)P(A_i)$

Bayes' Theorem: $$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$$

- Let **X** be a data sample ("*evidence*"): class label is unknown
- Let H be a *hypothesis* that X belongs to class C
- Classification is to determine P(H|**X**), (i.e., *posteriori probability):* the probability that the hypothesis holds given the observed data sample **X**
- P(H) (*prior probability*): the initial probability
  - E.g., **X** will buy computer, regardless of age, income, …
- P(**X**): probability that sample data is observed
- P(**X**|H) (likelihood): the probability of observing the sample **X**, given that the hypothesis holds
  - E.g., Given that **X** will buy computer, the prob. that X is 31..40, medium income

Prediction Based on Bayes' Theorem

Given training data **X**, *posteriori probability of a hypothesis* H, P(H|**X**), follows the Bayes' theorem

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$$

Informally, this can be viewed as

posteriori = likelihood x prior/evidence

Predicts **X** belongs to C_i iff the probability $P(C_i|\mathbf{X})$ is the highest among all the $P(C_k|X)$ for all the *k* classes

Practical difficulty: It requires initial knowledge of many probabilities, involving significant computational cost

Bayes Classifiation

Naive Bayes Method

Statistical

Estimate : probability

$X = (x_1, x_2, \dots x_n)$

Two classes - C_1, C_2

$$\left.\begin{array}{l} P(C_1/X) \\ P(C_2/X) \end{array}\right\}$$

Bayes Theorem

Naive Bayes → "simple"

c

$$P(c/x) = \frac{P(c,x)}{P(x)}$$

$$P(x/c) = \frac{P(c,x)}{P(c)}$$

$$P(x/c) \cdot P(c) = P(c/x) \cdot P(x)$$

$$P(c/x) = \frac{P(x/c_i)\, P(c_i)}{P(x)}$$

posteriori probability

marginal

class conditional densities → $\left\{ \begin{array}{l} P(x/c_1) \\ P(x/c_2) \end{array} \right.$

Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \ldots \times P(x_n|C_i)$$

- This greatly reduces the computation cost: Only counts the class distribution
- If A_k is categorical, $P(x_k|C_i)$ is the # of tuples in C_i having value x_k for A_k divided by $|C_{i, D}|$ (# of tuples of C_i in D)
- If A_k is continuous-valued, $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

$$g(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

and $P(x_k|C_i)$ is

$$P(\mathbf{X}|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

Naïve Bayes Classifier: Training Dataset

Class:

C1:buys_computer = ‘yes’

C2:buys_computer = ‘no’

Data to be classified:

X = (age <=30,

Income = medium,

Student = yes

Credit_rating = Fair)

age	income	student	credit_rating	_comp
<=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

Naïve Bayes Classifier: An Example

age	income	student	credit_rating	comp
<=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

$P(C_i)$: P(buys_computer = "yes") = 9/14 = 0.643

P(buys_computer = "no") = 5/14= 0.357

Compute $P(X|C_i)$ for each class

P(age = "<=30" | buys_computer = "yes") = 2/9 = 0.222

P(age = "<= 30" | buys_computer = "no") = 3/5 = 0.6

P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444

P(income = "medium" | buys_computer = "no") = 2/5 = 0.4

P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667

P(student = "yes" | buys_computer = "no") = 1/5 = 0.2

P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667

P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4

X = (age <= 30 , income = medium, student = yes, credit_rating = fair)

$P(X|C_i)$: P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044

P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

$P(X|C_i)*P(C_i)$: P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028

P(X|buys_computer = "no") * P(buys_computer = "no") = 0.007

Therefore, X belongs to class ("buys_computer = yes")

Avoiding the Zero-Probability Problem

Naïve Bayesian prediction requires each conditional prob. be **non-zero**. Otherwise, the predicted prob. will be zero

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

Ex. Suppose a dataset has 1000 samples in class *C_1*, where the distribution is: income=low (0), income= medium (990), and income = high (10)

Use **Laplacian correction** (or Laplacian estimator)

Adding 1 to each case

Prob(income = low/*C_1*) = 1/1003

Prob(income = medium/*C_1*) = 991/1003

Prob(income = high/*C_1*) = 11/1003

The "corrected" prob. estimates are close to their "uncorrected" counterparts

Naïve Bayes Classifier: Comments

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.
      Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayes Classifier

How to deal with these dependencies? Bayesian Belief Networks (Chapter 9)

Evaluation of Classifiers

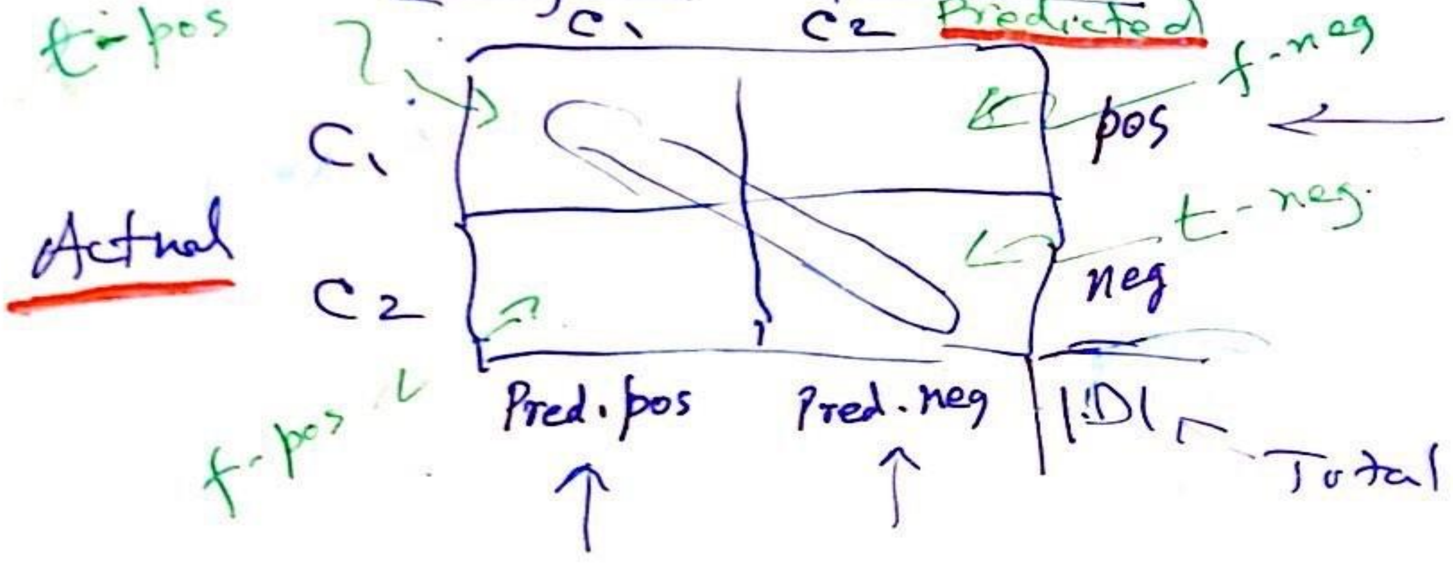
$$\text{sensitivity} = \frac{\text{t-pos}}{\text{pos}}$$

$$\text{specifiaty} = \frac{\text{t-neg}}{\text{neg}}$$

$$\text{precision} = \frac{\text{t-pos}}{\text{Pred. pos}}$$

$$\text{accuracy} = \frac{\text{t-pos} + \text{t-neg}}{\text{pos} + \text{neg}}$$

Confusion Matrix



Classifier Evaluation Metrics: Example

Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	**90**	**210**	300	30.00 (*sensitivity*
cancer = no	**140**	**9560**	9700	98.56 (*specificity)*
Total	230	9770	10000	96.40 (*accuracy*)

Precision = 90/230 = 39.13%   *Recall* = 90/300 = 30.00%

Evaluating Classifier Accuracy: Holdout & Cross-Validation Methods

- **Holdout method**
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - Random sampling: a variation of holdout
    - Repeat holdout k times, accuracy = avg. of the accuracies obtained
- **Cross-validation** (*k*-fold, where k = 10 is most popular)
  - Randomly partition the data into *k mutually exclusive* subsets, each approximately equal size
  - At *i*-th iteration, use D_i as test set and others as training set
  - Leave-one-out: *k* folds where *k* = # of tuples, for small sized data
  - ***Stratified cross-validation***: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data

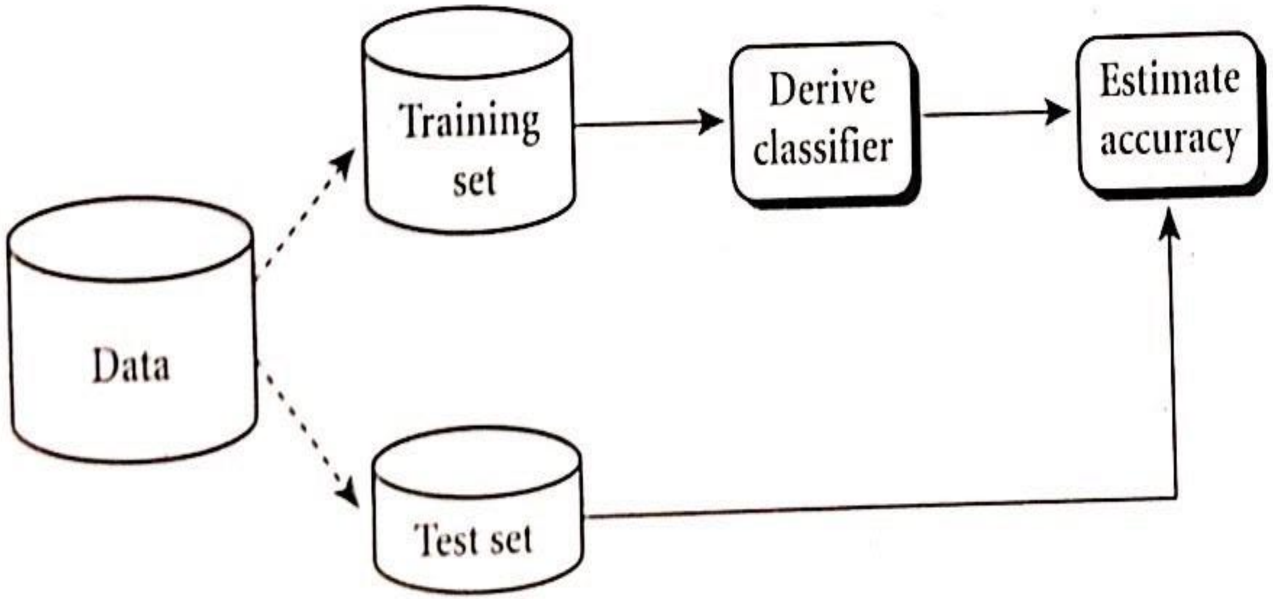


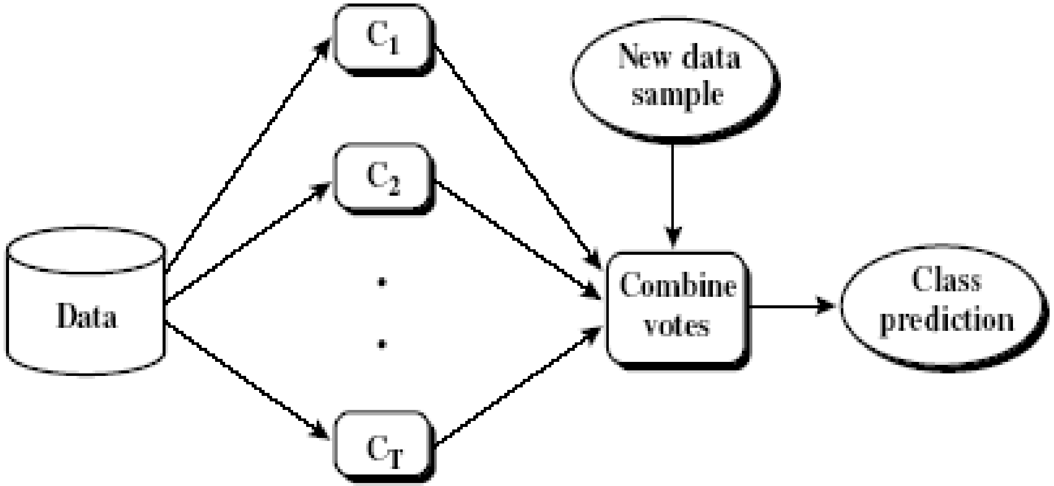
Figure ~~7~~**.16** Estimating classifier accuracy with the holdout method. i) Random subsampling

6.29

ii) K-fold
cross-validation.

iii) Stratified cross validation *

Ensemble Methods: Increasing the Accuracy



- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of k learned models, M_1, M_2, …, M_k, with the aim of creating an improved model M*
- Popular ensemble methods
  - Bagging: averaging the prediction over a collection of classifiers
  - Boosting: weighted vote with a collection of classifiers
  - Ensemble: combining a set of heterogeneous classifiers

Bagging: Boostrap Aggregation

- Analogy: Diagnosis based on multiple doctors' majority vote
- Training
  - Given a set D of *d* tuples, at each iteration *i*, a training set D_i of *d* tuples is sampled with replacement from D (i.e., bootstrap)
  - A classifier model M_i is learned for each training set D_i
- Classification: classify an unknown sample **X**
  - Each classifier M_i returns its class prediction
  - The bagged classifier M* counts the votes and assigns the class with the most votes to **X**
- Prediction: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- Accuracy
  - Often significantly better than a single classifier derived from D
  - For noise data: not considerably worse, more robust
  - Proved improved accuracy in prediction

Summary

- Classification is a form of data analysis that extracts models describing important data classes.
- Effective and scalable methods have been developed for decision tree induction, Naive Bayesian classification, rule-based classification, and many other classification methods.
- Evaluation metrics include: accuracy, sensitivity, specificity, precision, recall, F measure, and F_{β} measure.
- Stratified k-fold cross-validation is recommended for accuracy estimation. Bagging and boosting can be used to increase overall accuracy by learning and combining a series of individual models.