

---


# **Introduction to Data Mining**

## **Classification**



# Agenda

---

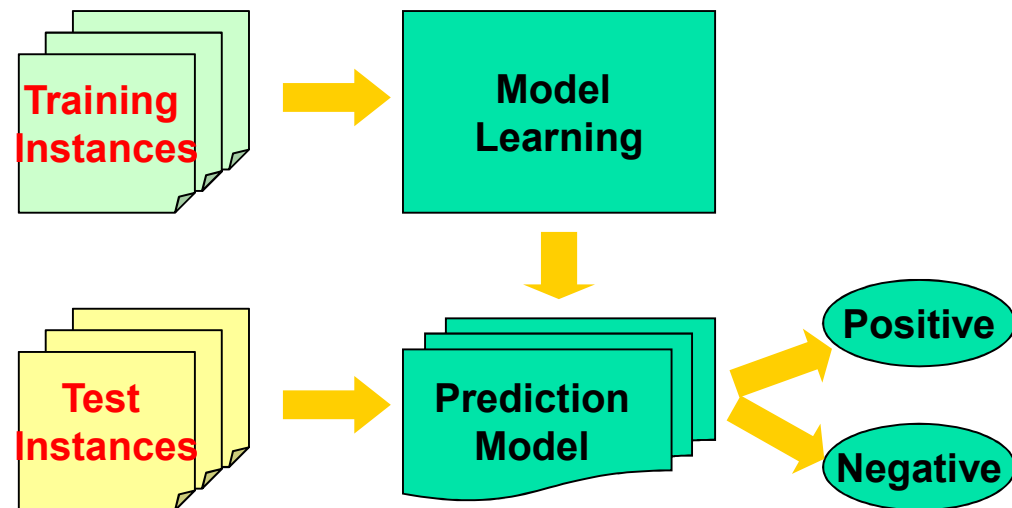
- Classification: Basic Concepts 
- Decision Tree Induction
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy:  
Ensemble Methods
- Bayes Classification Methods
- Summary

# Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data such as observations or measurements are accompanied by **labels** indicating the classes which they belong to
  - New data is classified based on the models built from the training set

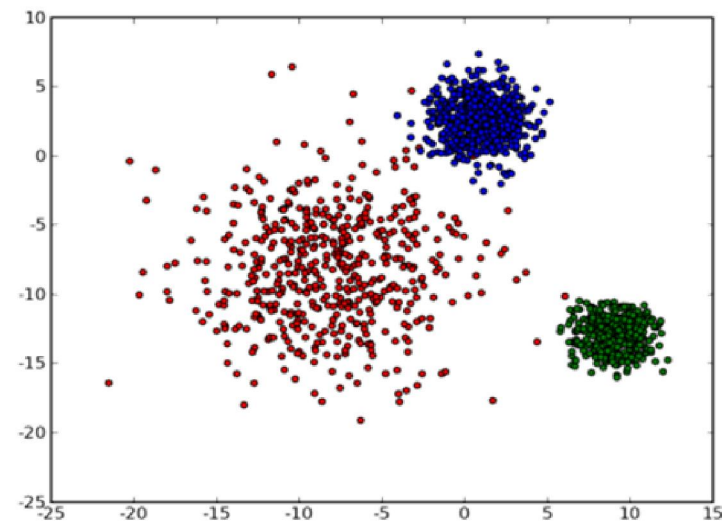
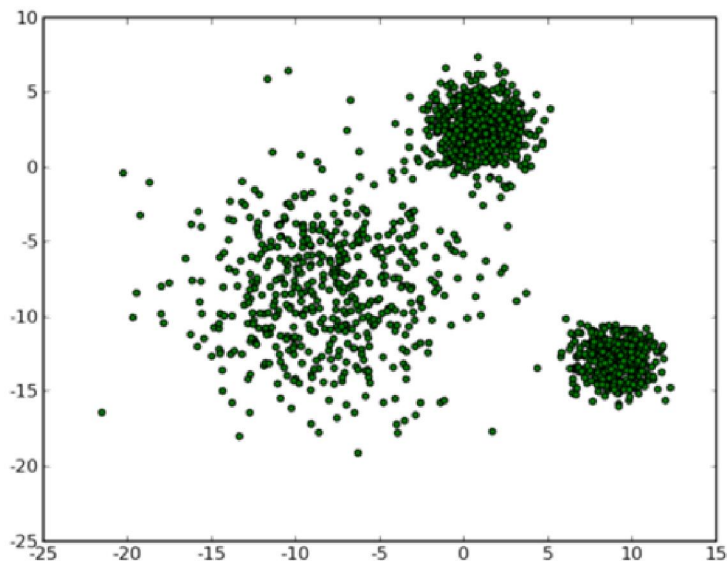
Training Data with class label:

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



# Supervised vs. Unsupervised Learning

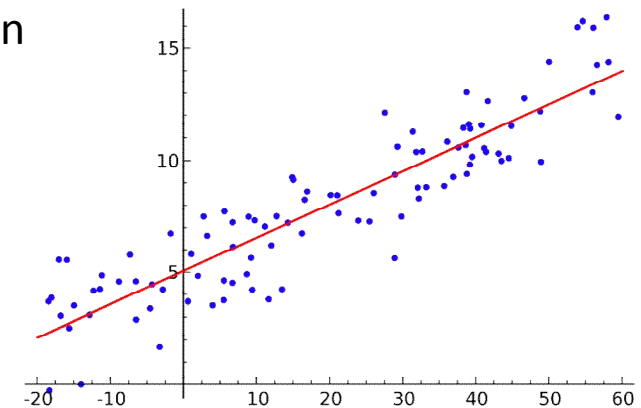
- Unsupervised learning (clustering)
  - The class labels of training data are unknown
  - Given a set of observations or measurements, establish the possible existence of classes or clusters in the data



# Prediction Problems: Classification vs. Numeric Prediction

---

- **Classification**
  - Predict categorical class labels (discrete or nominal)
  - Construct a model based on the training set and the **class labels** (the values in a classifying attribute) and use it in classifying new data
- **Numeric prediction**
  - Model continuous-valued functions (i.e., predict unknown or missing values)
- Typical applications of classification
  - Credit/loan approval
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is

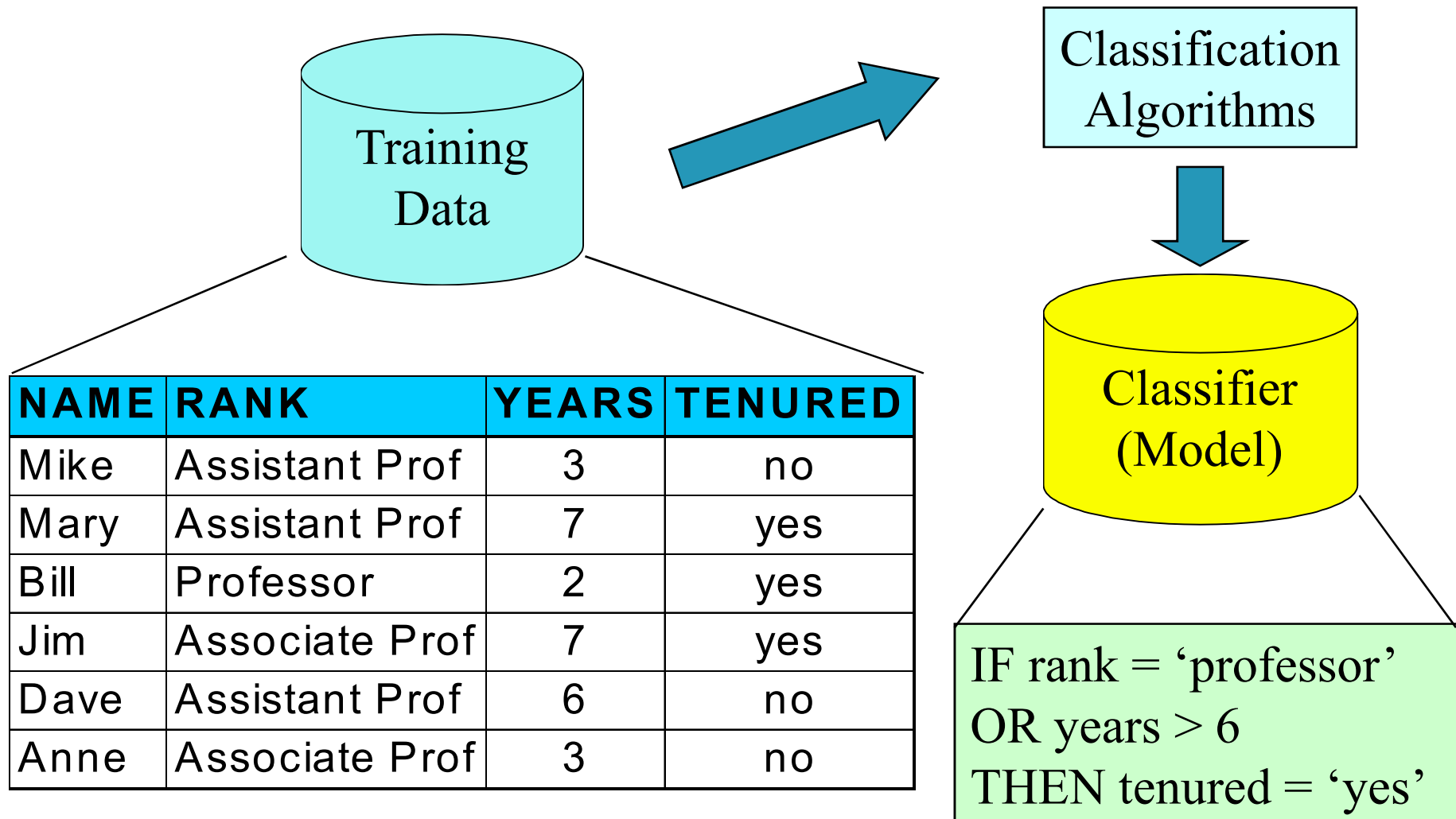


# Classification—Model Construction, Validation and Testing

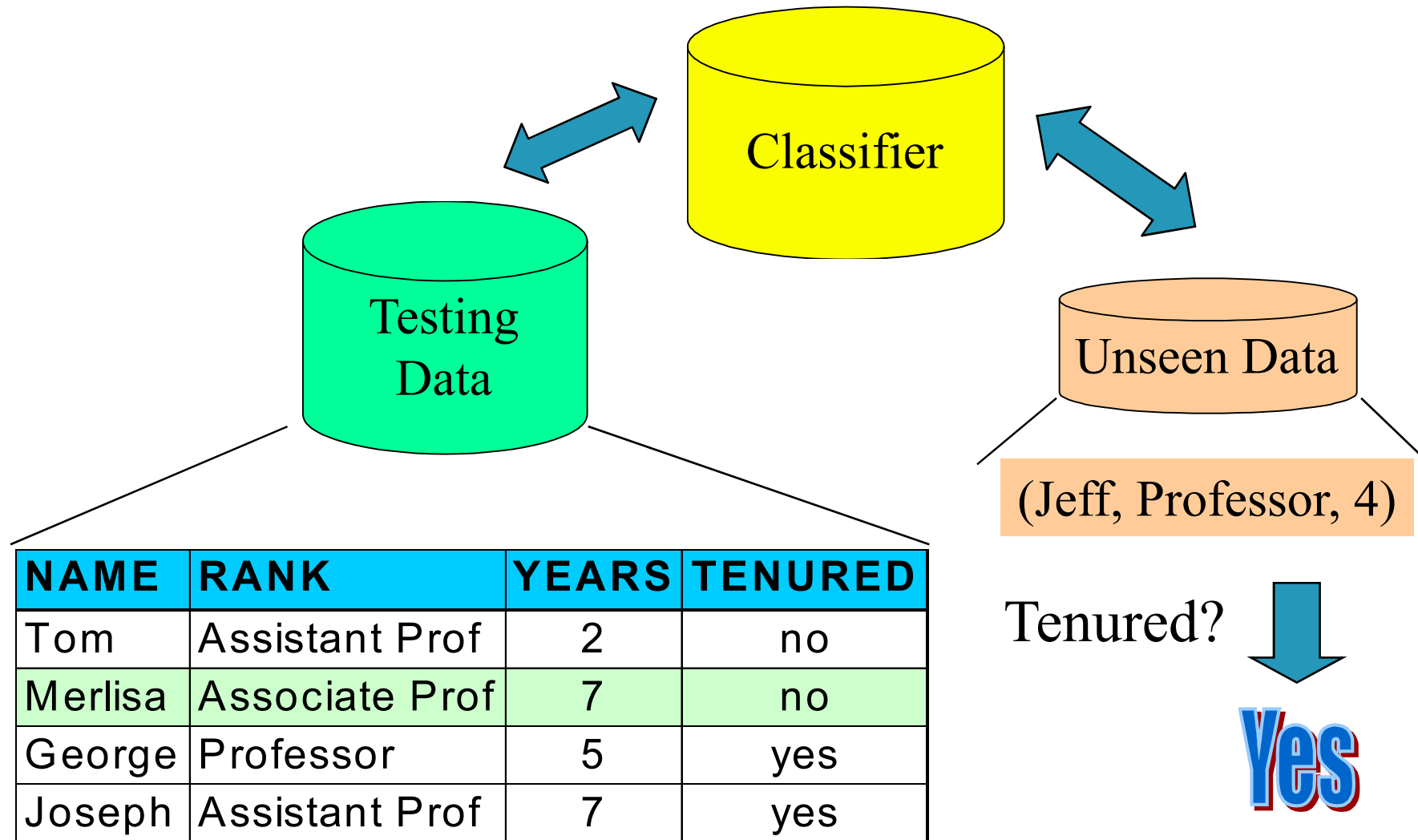
---

- **Model construction**
  - Each sample is assumed to belong to a predefined class (shown by the **class label**)
  - The set of samples used for model construction is **training set**
  - Model: Represented as decision trees, rules, mathematical formulas, or other forms
- **Model Validation and Testing:**
  - **Test:** Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - *Accuracy*: % of test set samples that are correctly classified by the model
    - Test set is independent of training set
  - **Validation:** If *the test set* is used to select or refine models, it is called **validation** (or development) **(test) set**
- **Model Deployment:** If the accuracy is acceptable, use the model to classify new data

# Process : Model Construction




# Process : Using the Model in Prediction





# Agenda

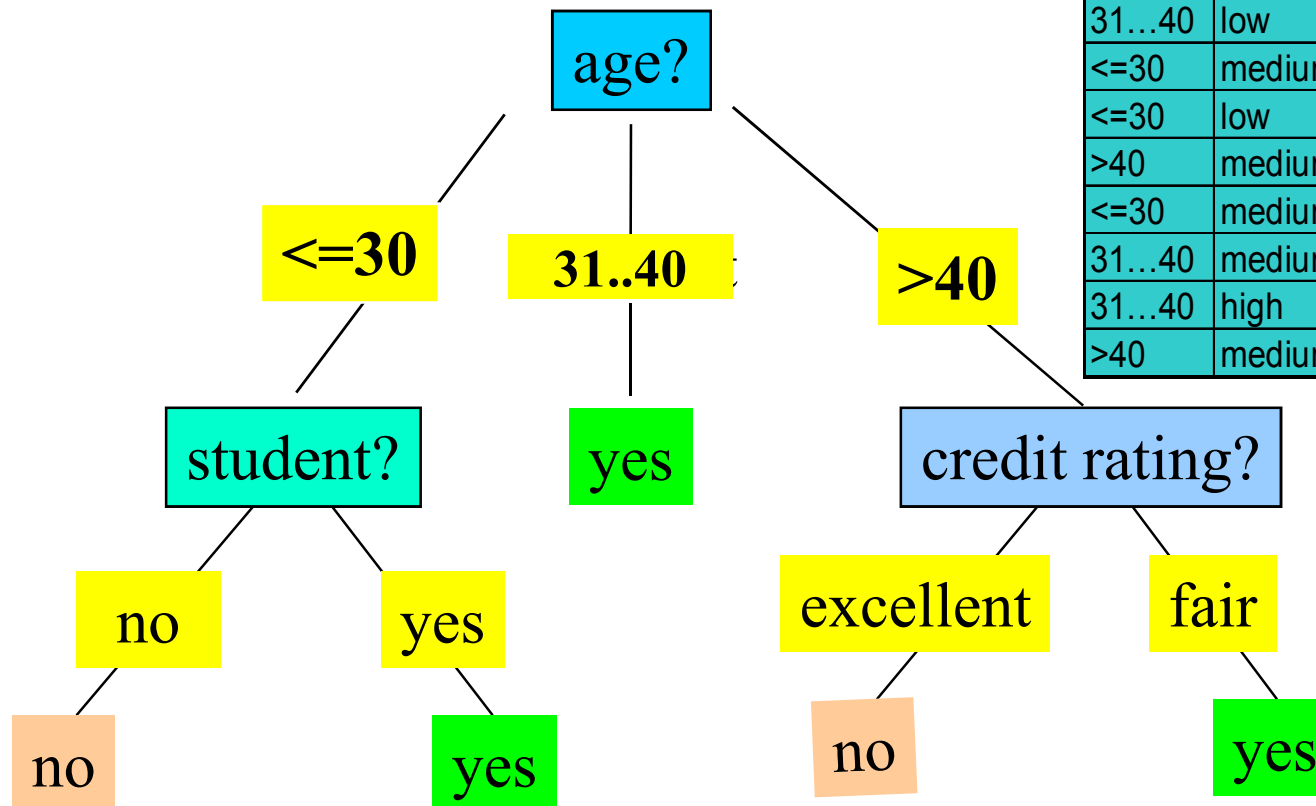
---

- Classification: Basic Concepts
- Decision Tree Induction 
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy:  
Ensemble Methods
- Bayes Classification Methods
- Summary

# Decision Tree Induction: An Example

- Training data set: Buys\_computer
- Resulting tree:

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



# Decision Tree Induction: Algorithm

---

- Basic algorithm
  - Tree is constructed in a **top-down, recursive, divide-and-conquer manner**
  - At start, all the training examples are at the root
  - Examples are partitioned recursively based on selected attributes
  - On each node, attributes are selected based on the training examples on that node, and 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
  - There are no samples left
- Prediction
  - **Majority voting** is employed for classifying the leaf

# 熵的基本概念

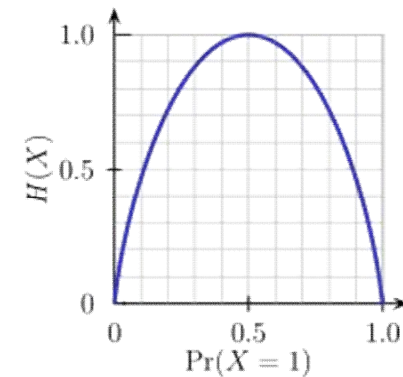
## ■ 信息熵 (Entropy) (信息论)

- 一条信息的信息量和它的不确定性有着直接的关系
- 信息量等于不确定性的多少
- 不确定性越大，熵越大
- 离散型随机变量X的熵定义为：

$$H(X) = - \sum_{x \in X} p(x) \log p(x)$$

例1： 假设世界杯决赛圈32强已经产生，那么随机变量“2018年俄罗斯世界杯足球赛32强中，谁是世界杯冠军？”的信息量是多少呢？

例2： 赌马比赛，四匹马A、B、C和D获胜的概率分别为 $\{1/2, 1/4, 1/8, 1/8\}$ 。设随机变量 $x \in \{A, B, C, D\}$ 表示哪一匹马获胜。



**m = 2**

# 熵的基本概念

- 条件熵(Conditional Entropy)
  - 假设有X和Y两个随机变量，要了解X。
  - 现已知X的随机分布 $p(X)$ ，即已知X的熵 $H(X)$ 。
  - 还知道Y的一些信息。
  - 在Y的条件下X的条件熵定义为：

$$H(X|Y) = - \sum_{x \in X, y \in Y} p(x, y) \log p(x|y)$$

$$H(X|Y, Z) = - \sum_{x \in X, y \in Y, z \in Z} p(x, y, z) \log p(x|y, z)$$

$$H(X) \geq H(X|Y) \geq H(X|Y, Z)$$

$$\begin{aligned} H(X|Y) &= \sum_{y \in Y} p(y) H(X|Y = y) \\ &= \sum_{y \in Y} p(y) \sum_{x \in X} p(x|y) \log p(x|y) \\ &= - \sum_{x \in X, y \in Y} p(x, y) \log p(x|y) \end{aligned}$$

# 熵的基本概念

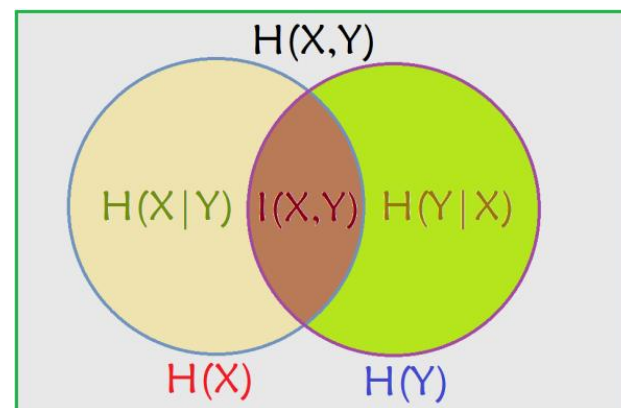
## ■ 互信息 (Mutual Information)

- 两个随机事件相关性的量化度量。
- 在了解Y的前提下，对于消除X不确定性所提供的信息量。
- 两个随机变量X和Y的互信息定义如下：

$$I(X; Y) = - \sum_{x \in X, y \in Y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$$

$$I(X; Y) = H(X) - H(X|Y)$$

例1： 闷热 下雨



# Information Gain: An Attribute Selection Measure

- Select the attribute with the highest information gain (used in typical decision tree induction algorithm: ID3/C4.5)
- Let  $p_i$  be the probability that an arbitrary tuple in  $D$  belongs to class  $C_i$ , estimated by  $|C_i \cap D|/|D|$
- 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  $A$

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

## Example: Attribute Selection with Information Gain

■ Class P: buys\_computer = “yes”

■ Class N: buys\_computer = “no”

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

age	p <sub>i</sub>	n <sub>i</sub>	I(p <sub>i</sub> , n <sub>i</sub> )
<=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, we can get

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

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



# How to Handle Continuous-Valued Attributes?

- Method 1: Discretize continuous values and treat them as categorical values
  - E.g., age: < 20, 20..30, 30..40, 40..50, > 50
- Method 2: Determine the **best split point** for continuous-valued attribute A
  - Sort the value A in increasing order:, e.g. 15, 18, 21, 22, 24, 25, 29, 31, ...
  - *Possible split point*: the midpoint between *each pair of adjacent values*
    - $(a_i + a_{i+1})/2$  is the midpoint between the values of  $a_i$  and  $a_{i+1}$
    - e.g.,  $(15+18)/2 = 16.5, 19.5, 21.5, 23, 24.5, 27, 30, \dots$
  - The point with the *maximum information gain* for A is selected as the **split-point** for A
- Split: Based on split point P
  - The set of tuples in D satisfying  $A \leq P$  vs. those with  $A > P$

## Gain Ratio: A Refined Measure

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

- Information gain measure is biased towards attributes with many values
- Gain ratio: Overcomes the problem (as a normalized version of 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)
- The attribute with the maximum gain ratio is selected as the splitting attribute
- Gain ratio is used in a popular algorithm C4.5 (a successor of ID3) by R. Quinlan
- Example
  - $SplitInfo_{income}(D) = -\frac{4}{14} \log_2 \frac{4}{14} - \frac{6}{14} \log_2 \frac{6}{14} - \frac{4}{14} \log_2 \frac{4}{14} = 1.557$
  - GainRatio(income) = 0.029/1.557 = 0.019

## Another Measure: Gini Index

- Gini index: Used in CART, and also in IBM Intelligent Miner
- If a data set  $D$  contains examples from  $n$  classes, gini index,  $gini(D)$  is defined as
  - $gini(D) = 1 - \sum_{j=1}^n p_j^2$ 
    - $p_j$  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  $gini_A(D)$  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**)

## Computation of Gini

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: D has 9 tuples in buys\_computer = yes

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- Suppose the attribute income partitions D into D<sub>1</sub> = {tuples with income {low, medium}} and 4 in D<sub>2</sub>

$$\begin{aligned} \blacksquare \quad gini_{income \in \{low, medium\}}(D) &= \frac{10}{14} gini(D_1) + \frac{4}{14} gini(D_2) \\ &= \frac{10}{14} \left( 1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2 \right) + \frac{4}{14} \left( 1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2 \right) = 0.443 = Gini_{income \in \{high\}}(D) \end{aligned}$$

- Gini<sub>{low, high}</sub> is 0.458; Gini<sub>{medium, high}</sub> is 0.450
- Thus, split on the {low, medium} (and {high}) since it has the lowest Gini index

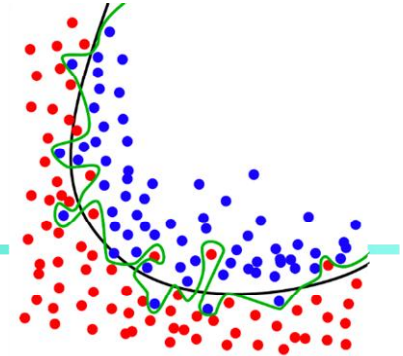
- All attributes are assumed continuous-valued
- May need other tools, e.g., clustering, to get the possible split values
- Can be modified for categorical attributes

# Comparing Three 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”

# Agenda

- Classification: Basic Concepts
- Decision Tree Induction
- Model Evaluation and Selection 
- Techniques to Improve Classification Accuracy:  
Ensemble Methods
- Bayes Classification Methods
- Summary

# Model Evaluation and Selection

---

- Evaluation metrics
  - How can we measure accuracy?
  - Other metrics to consider?
- Use **validation test set** of class-labeled tuples instead of training set when assessing accuracy
- Methods for estimating a classifier's accuracy
  - Holdout method
  - Cross-validation
  - Bootstrap
- Comparing classifiers:
  - ROC Curves



# Classifier Evaluation Metrics: Confusion Matrix

## ■ Confusion Matrix:

Actual class\Predicted class	$C_1$	$\neg C_1$
$C_1$	True Positives (TP)	False Negatives (FN)
$\neg C_1$	False Positives (FP)	True Negatives (TN)

- In a confusion matrix w.  $m$  classes,  $CM_{i,j}$  indicates # of tuples in class  $i$  that were labeled by the classifier as class  $j$

- May have extra rows/columns to provide totals

## ■ Example of Confusion Matrix:

Actual class\Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	6954	46	7000
buy_computer = no	412	2588	3000
Total	7366	2634	10000

# Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity

---

A\P	C	¬C	
C	TP	FN	P
¬C	FP	TN	N
	P'	N'	All

- **Classifier accuracy**, or recognition rate
  - Percentage of test set tuples that are correctly classified

$$\text{Accuracy} = (\text{TP} + \text{TN}) / \text{All}$$

- **Error rate**:  $1 - \text{accuracy}$ , or
$$\text{Error rate} = (\text{FP} + \text{FN}) / \text{All}$$

## □ **Class imbalance problem**

- One class may be *rare*
  - E.g., fraud, or HIV-positive
- Significant *majority of the negative class* and minority of the positive class
- Measures handle the class imbalance problem
  - **Sensitivity** (recall): True positive recognition rate
    - **Sensitivity = TP/P**
  - **Specificity**: True negative recognition rate
    - **Specificity = TN/N**

# Classifier Evaluation Metrics:

## Precision and Recall, and F-measures

- **Precision:** Exactness: what % of tuples that the classifier labeled as positive are actually positive?

$$P = \text{Precision} = \frac{TP}{TP + FP}$$

- **Recall:** Completeness: what % of positive tuples did the classifier label as positive?

$$R = \text{Recall} = \frac{TP}{TP + FN}$$

- Range: [0, 1]
- The “inverse” relationship between precision & recall
- **F measure (or F-score):** harmonic mean of precision and recall
  - In general, it is the weighted measure of precision & recall

$$F_{\beta} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

Assigning  $\beta$  times as much weight to recall as to precision)

- **F1-measure (balanced F-measure)**
  - That is, when  $\beta = 1$ ,

$$F_1 = \frac{2PR}{P + R}$$

## Classifier Evaluation Metrics: Example

- Use the same confusion matrix, calculate the measure just introduced

Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	<b>90</b>	<b>210</b>	300	30.00 ( <i>sensitivity</i> )
cancer = no	<b>140</b>	<b>9560</b>	9700	98.56 ( <i>specificity</i> )
Total	230	9770	10000	96.50 ( <i>accuracy</i> )

- Sensitivity =  $TP/P = 90/300 = 30\%$
- Specificity =  $TN/N = 9560/9700 = 98.56\%$
- Accuracy =  $(TP + TN)/All = (90+9560)/10000 = 96.50\%$
- Error rate =  $(FP + FN)/All = (140 + 210)/10000 = 3.50\%$
- Precision =  $TP/(TP + FP) = 90/(90 + 140) = 90/230 = 39.13\%$
- Recall =  $TP/(TP + FN) = 90/(90 + 210) = 90/300 = 30.00\%$
- F1 =  $2 P \times R / (P + R) = 2 \times 39.13\% \times 30.00\% / (39.13\% + 30\%) = 33.96\%$

# Classifier Evaluation: Holdout & Cross-Validation

- **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
- Repeated random sub-sampling validation: 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 distribution, in each fold is approximately the same as that in the initial data

# Classifier Evaluation: Bootstrap

---

- **Bootstrap**
  - Works well with small data sets
  - Samples the given training tuples uniformly *with replacement*
    - Each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
- Several bootstrap methods, and a common one is **.632 bootstrap**
  - A data set with  $d$  tuples is sampled  $d$  times, with replacement, resulting in a training set of  $d$  samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data end up in the bootstrap, and the remaining 36.8% form the test set (since  $(1 - 1/d)^d \approx e^{-1} = 0.368$ )
  - Repeat the sampling procedure  $k$  times, overall accuracy of the model:

$$Acc(M) = \frac{1}{k} \sum_{i=1}^k (0.632 \times Acc(M_i)_{test\_set} + 0.368 \times Acc(M_i)_{train\_set})$$

# Estimating Confidence Intervals: Classifier Models $M_1$ vs. $M_2$

- Suppose we have 2 classifiers,  $M_1$  and  $M_2$ , which one is better?
- Use 10-fold cross-validation to obtain  $\overline{err}(M_1)$  and  $\overline{err}(M_2)$
- These mean error rates are just *estimates* of error on the true population of *future* data cases
- What if the difference between the 2 error rates is just attributed to *chance*?
  - Use a **test of statistical significance**
  - Obtain **confidence limits** for our error estimates

# Estimating Confidence Intervals: Null Hypothesis

---

- Perform 10-fold cross-validation
- Assume samples follow a **t distribution** with  $k-1$  **degrees of freedom** (here,  $k=10$ )
- Use **t-test** (or **Student's t-test**)
- **Null Hypothesis:**  $M_1$  &  $M_2$  are the same
- If we can **reject** null hypothesis, then
  - we conclude that the difference between  $M_1$  &  $M_2$  is **statistically significant**
  - Chose model with lower error rate



# Estimating Confidence Intervals: t-test

- If only 1 test set available: **pairwise comparison**
  - For  $i^{\text{th}}$  round of 10-fold cross-validation, the same cross partitioning is used to obtain  $err(M_1)_i$  and  $err(M_2)_i$
  - Average over 10 rounds to get  $\overline{err}(M_1)$  and  $\overline{err}(M_2)$
  - **t-test** computes **t-statistic** with  $k-1$  **degrees of freedom**:

$$t = \frac{\overline{err}(M_1) - \overline{err}(M_2)}{\sqrt{var(M_1 - M_2)/k}} \quad \text{where}$$
$$var(M_1 - M_2) = \frac{1}{k} \sum_{i=1}^k \left[ err(M_1)_i - err(M_2)_i - (\overline{err}(M_1) - \overline{err}(M_2)) \right]^2$$

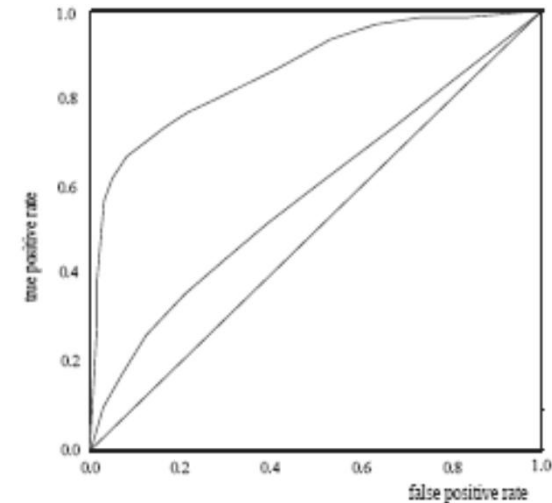
- If two test sets available: use **non-paired t-test**

$$\text{where } var(M_1 - M_2) = \sqrt{\frac{var(M_1)}{k_1} + \frac{var(M_2)}{k_2}},$$

where  $k_1$  &  $k_2$  are # of cross-validation samples used for  $M_1$  &  $M_2$ , resp.

# Model Selection: ROC Curves

- **ROC** (Receiver Operating Characteristics) curves: for visual comparison of classification models
- Originated from signal detection theory
- Shows the trade-off between the true positive rate and the false positive rate
- The area under the ROC curve (**AUC**: Area Under Curve) is a measure of the accuracy of the model
- Rank the test tuples in decreasing order: the one that is most likely to belong to the positive class appears at the top of the list
- The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model



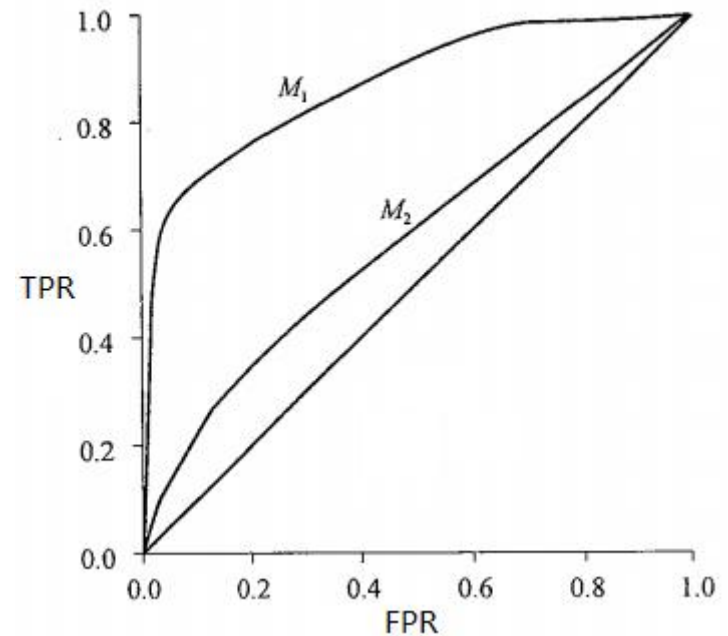
- Vertical axis represents the true positive rate
- Horizontal axis rep. the false positive rate
- The plot also shows a diagonal line
- A model with perfect accuracy will have an area of 1.0

# Model Selection: ROC Curves

元组编号	类	概率	TP	FP	TN	FN	TPR	FPR
1	P	0.90	1	0	5	4	0.2	0
2	P	0.80	2	0	5	3	0.4	0
3	N	0.70	2	1	4	3	0.4	0.2
4	P	0.60	3	1	4	2	0.6	0.2
5	P	0.55	4	1	4	1	0.8	0.2
6	N	0.54	4	2	3	1	0.8	0.4
7	N	0.53	4	3	2	1	0.8	0.6
8	N	0.51	4	4	1	1	0.8	0.8
9	P	0.50	5	4	1	0	1.0	0.8
10	N	0.40	5	5	0	0	1.0	1.0

$$TPR = \frac{TP}{P}$$

$$FPR = \frac{FP}{N}$$



# Issues Affecting Model Selection

---

- **Accuracy**
  - classifier accuracy: predicting class label
- **Speed**
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- **Robustness**: handling noise and missing values
- **Scalability**: efficiency in disk-resident databases
- **Interpretability**
  - understanding and insight provided by the model
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules

# Example

## Step 1: collecting data



Statlog (German Credit Data)

Description of the data set

### Statlog (German Credit Data) Data Set

Download: [Data Folder](#), [Data Set Description](#)

**Abstract:** This dataset classifies people described by a set of attributes as good or bad credit risks. Comes in two formats (one all numeric). Also comes with a cost matrix

Data Set Characteristics:	Multivariate	Number of Instances:	1000	Area:	Financial
Attribute Characteristics:	Categorical, Integer	Number of Attributes:	20	Date Donated	1994-11-17
Associated Tasks:	Classification	Missing Values?	N/A	Number of Web Hits:	430069

#### Source:

Professor Dr. Hans Hofmann  
Institut für Statistik und Ökonometrie  
Universität Hamburg  
FB Wirtschaftswissenschaften  
Von-Melle-Park 5  
2000 Hamburg 13



[Parent Directory](#)



[Index](#)

03-Dec-1996 04:07 150



[german.data](#)

17-Nov-1994 00:51 78K



[german.data-numeric](#)

17-Nov-1994 00:51 100K



[german.doc](#)

17-Nov-1994 00:51 4.6K

```
A11 6 A34 A43 1169 A65 A75 4 A93 A101 4 A121 67 A143 A152 2 A173 1 A192 A201 1
A12 48 A32 A43 5951 A61 A73 2 A92 A101 2 A121 22 A143 A152 1 A173 1 A191 A201 2
A14 12 A34 A46 2096 A61 A74 2 A93 A101 3 A121 49 A143 A152 1 A172 2 A191 A201 1
A11 42 A32 A42 7882 A61 A74 2 A93 A103 4 A122 45 A143 A153 1 A173 2 A191 A201 1
A11 24 A33 A40 4870 A61 A73 3 A93 A101 4 A124 53 A143 A153 2 A173 2 A191 A201 2
A14 36 A32 A46 9055 A65 A73 2 A93 A101 4 A124 35 A143 A153 1 A172 2 A192 A201 1
```

## Example

### Step 2: exploring and preparing the data

Install two packages:

```
install.packages("C50")
```

```
install.packages("gmodels")
```

V1: Status of existing checking account

V6: Savings account/bonds

V2: Duration in month

V5: Credit amount

V21: classification 1good/2bad

```
> credit <- read.table("german.data", sep = " ")
> table(credit$V1)

A11 A12 A13 A14
274 269  63 394
> table(credit$V6)

A61 A62 A63 A64 A65
603 103  63  48 183
> summary(credit$V2)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   4.0    12.0    18.0   20.9    24.0    72.0
> summary(credit$V5)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   250    1366    2320   3271    3972   18424
> table(credit$V21)

 1    2
700 300
```

## Example

---

### Step 2: exploring and preparing the data

Creating random training and test dataset

```
> set.seed(12345)
> credit_rand <- credit[order(runif(1000)),]
> summary(credit$V5)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   250   1366   2320   3271   3972   18424
> summary(credit_rand$V5)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   250   1366   2320   3271   3972   18424
> head(credit$V5)
[1] 1169 5951 2096 7882 4870 9055
> head(credit_rand$V5)
[1] 1199 2576 1103 4020 1501 1568
> credit_train <- credit_rand[1:900,]
> credit_test <- credit_rand[901:1000,]
> prop.table(table(credit_train$V21))

      1      2
0.7022222 0.2977778
> prop.table(table(credit_test$V21))

      1      2
0.68 0.32
```

## Example

### Step 3: training a model on the data

```
> library(C50)
> credit_model <- C5.0(credit_train[-21], as.factor(credit_train$V21))
> credit_model
```

```
Call:
C5.0.default(x = credit_train[-21], y
 = as.factor(credit_train$V21))
```

```
Classification Tree
Number of samples: 900
Number of predictors: 20
```

```
Tree size: 57
```

```
Non-standard options: attempt to group attributes
```

```
> summary(credit_model)
```

```
Call:
C5.0.default(x = credit_train[-21], y
 = as.factor(credit_train$V21))
```

```
C5.0 [Release 2.07 GPL Edition]
```

```
Tue Oct 30 16:37:07 2018
```

```
-----
Class specified by attribute `outcome'
```

```
Read 900 cases (21 attributes) from undefined.data
```

```
Decision tree:
```

```
V1 = A14: 1 (358/44)
V1 in {A11,A12,A13}:
...V20 = A202:
  ...V14 in {A142,A143}: 1 (17/1)
  : V14 = A141:
  :   ...V11 <= 3: 2 (2)
  :     V11 > 3: 1 (2)
```



## Example

### Step 4: evaluating model performance

```
> credit_pred <- predict(credit_model, credit_test)
> library(gmodels)
> CrossTable(credit_test$V21, credit_pred,
+           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
+           dnn = c('actual default', 'predicted default'))
```

Cell Contents

		N
		N / Table Total

Total observations in Table: 100

actual default	predicted default		Row Total
	1	2	
1	54 0.540	14 0.140	68
2	11 0.110	21 0.210	32
Column Total	65	35	100

## Example

### Step 5: improving model performance

```
> credit_boost10 <- C5.0(credit_train[-21], as.factor(credit_train$V21), trials = 10)
> credit_boost10
```

```
call:
C5.0.default(x = credit_train[-21], y = as.factor(credit_train$V21), trials = 10)
```

```
Classification Tree
Number of samples: 900
Number of predictors: 20

Number of boosting iterations: 10
Average tree size: 47.3

Non-standard options: attempt to group attributes
```

```
> summary(credit_boost10)
```

```
call:
C5.0.default(x = credit_train[-21], y = as.factor(credit_train$V21), trials = 10)
```

```
C5.0 [Release 2.07 GPL Edition] Tue Oct 30 16:54:54 2018
```

```
-----
Class specified by attribute `outcome'
```

```
Read 900 cases (21 attributes) from undefined.data
```

```
----- Trial 0: -----
```

```
Decision tree:
```

```
V1 = A14: 1 (358/44)
V1 in {A11,A12,A13}:
:...V20 = A202:
  :...V14 in {A142,A143}: 1 (17/1)
  : V14 = A141:
  :   :...V11 <= 3: 2 (2)
  :   : V11 > 3: 1 (2)
  V20 = A201:
  :...V3 in {A30,A31}: 2 (61/20)
  V3 in {A32,A33,A34}:
  :...V2 <= 11: 1 (76/13)
```

## Example

### Step 5: improving model performance

```
> credit_boost_pred10 <- predict(credit_boost10, credit_test)
> CrossTable(credit_test$V21, credit_boost_pred10,
+           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
+           dnn = c('actual default', 'predicted default'))
```

Cell Contents

	N
N / Table Total	

Total observations in Table: 100

actual default	predicted default		Row Total
	1	2	
1	63 0.630	5 0.050	68
2	16 0.160	16 0.160	32
Column Total	79	21	100

## Example

### Step 5: improving model performance

```
> error_cost <- matrix(c(0, 1, 4, 0), nrow = 2)
> error_cost
      [,1] [,2]
[1,]    0    4
[2,]    1    0
> credit_cost <- c5.0(credit_train[-21], as.factor(credit_train$v21),
+                      costs = error_cost)
Warning message:
no dimnames were given for the cost matrix; the factor levels will be used
> credit_cost_pred <- predict(credit_cost, credit_test)
> CrossTable(credit_test$v21, credit_cost_pred,
+            prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
+            dnn = c('actual default', 'predicted default'))
```

cell contents

		N
	N / Table Total	

Total observations in Table: 100

actual default	predicted default		Row Total
	1	2	
1	38 0.380	30 0.300	68
2	5 0.050	27 0.270	32
Column Total	43	57	100

# Agenda

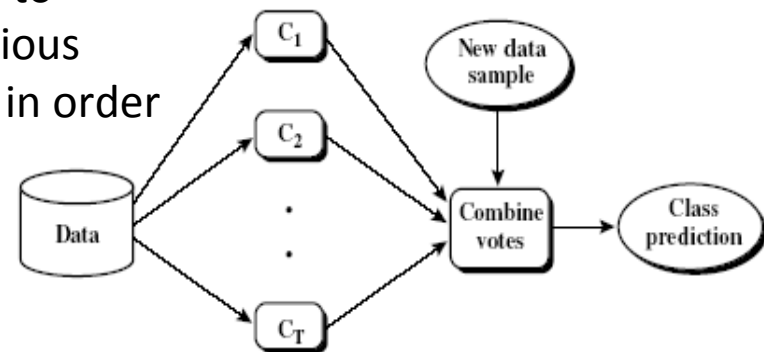
---

- Classification: Basic Concepts
- Decision Tree Induction
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy:  
Ensemble Methods
- Bayes Classification Methods
- Summary



# Ensemble Methods: Increasing the Accuracy

- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of  $k$  learned models,  $M_1, M_2, \dots, M_k$ , with the aim of creating an improved model  $M^*$
- Popular ensemble methods
  - Bagging: Trains each model using a subset of the training set, and models learned in parallel
  - Boosting: Trains each new model instance to emphasize the training instances that previous models mis-classified, and models learned in order



# Bagging: Bootstrap 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: It can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- Accuracy: Improved accuracy in prediction
  - Often significantly better than a single classifier derived from  $D$
  - For noise data: Not considerably worse, more robust

# Bagging: Bootstrap Aggregation

算法：装袋。装袋算法——为学习方案创建组合分类模型,其中每个模型给出等权重预测。

输入：

- $D$ :  $d$ 个训练元组的集合；
- $k$ : 组合分类器中的模型数；
- 一种学习方案（例如,决策树算法、后向传播等）

输出：组合分类器——复合模型 $M^*$ 。

方法：

- (1) **for**  $i = 1$  **to**  $k$  **do** // 创建 $k$ 个模型
- (2) 通过对 $D$ 有放回抽样, 创建自助样本 $D_i$ ;
- (3) 使用 $D_i$ 和学习方法导出模型 $M_i$ ;
- (4) **endfor**

使用组合分类器对元组 $x$ 分类：

让 $k$ 个模型都对 $x$ 分类并返回多数表决；



# Random Forest: Basic Concepts

- Random Forest (first proposed by L. Breiman in 2001)
  - A variation of bagging for *decision trees*
  - *Data bagging*
    - Use a subset of training data by sampling with replacement for each tree
  - *Feature bagging*
    - At each node use a random selection of attributes as candidates and split by the best attribute among them
  - Compared to original bagging, increases the diversity among generated trees
  - During classification, each tree votes and the most popular class is returned

# Random Forest

---

- Two Methods to construct Random Forest:
  - Forest-RI (*random input selection*): Randomly select, at each node,  $F$  attributes as candidates for the split at the node. The CART methodology is used to grow the trees to maximum size
  - Forest-RC (*random linear combinations*): Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)
- Comparable in accuracy to Adaboost, but more robust to errors and outliers
- Insensitive to the number of attributes selected for consideration at each split, and faster than typical bagging or boosting

# Boosting

- Analogy: Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy
- How boosting works?
  - **Weights** are assigned to each training tuple
  - A series of  $k$  classifiers is iteratively learned
  - After a classifier  $M_i$  is learned, the weights are updated to allow the subsequent classifier,  $M_{i+1}$ , to **pay more attention to the training tuples that were misclassified** by  $M_i$
  - The final  **$M^*$  combines the votes** of each individual classifier, where the weight of each classifier's vote is a function of its accuracy
- Boosting algorithm can be extended for numeric prediction
- Comparing with bagging: Boosting tends to have greater accuracy, but it also risks overfitting the model to misclassified data

# Adaboost (Freund and Schapire, 1997)

---

- Given a set of  $d$  class-labeled tuples,  $(\mathbf{X}_1, y_1), \dots, (\mathbf{X}_d, y_d)$
- Initially, all the weights of tuples are set the same ( $1/d$ )
- Generate  $k$  classifiers in  $k$  rounds. At round  $i$ ,
  - Tuples from  $D$  are sampled (with replacement) to form a training set  $D_i$  of the same size
  - Each tuple's chance of being selected is based on its weight
  - A classification model  $M_i$  is derived from  $D_i$
  - Its error rate is calculated using  $D_i$  as a test set
  - If a tuple is misclassified, its weight is increased; otherwise, it is decreased
- Error rate:  $err(\mathbf{X}_j)$  is the misclassification error of tuple  $\mathbf{X}_j$ . Classifier  $M_i$  error rate is the sum of the weights of the misclassified tuples:
$$error(M_i) = \sum_j^d w_j \times err(\mathbf{X}_j)$$
- The weight of classifier  $M_i$ 's vote is  $\log \frac{1 - error(M_i)}{error(M_i)}$

# Adaboost

算法: Adaboost. 一种提升算法——创建分类器的组合。每个给出一个加权投票。

输入:

- $D$ : 类标记的训练元组集。
- $k$ : 轮数 (每轮产生一个分类器)。
- 一种分类学习方案。

输出: 一个复合模型。

方法:

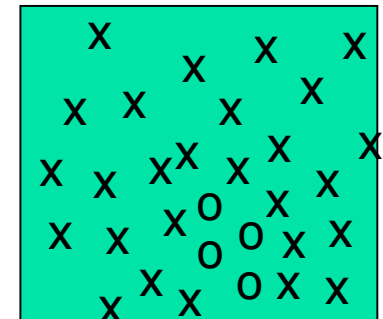
- (1) 将  $D$  中每个元组的权重初始化为  $1/d$ ;
- (2) **for**  $i = 1$  **to**  $k$  **do** // 对于每一轮
- (3) 根据元组的权重从  $D$  中有放回抽样, 得到  $D_i$ ;
- (4) 使用训练集  $D_i$  导出模型  $M_i$ ;
- (5) 计算  $M_i$  的错误率  $\text{error}(M_i)$  (8.34式)
- (6) **if**  $\text{error}(M_i) > 0.5$  **then**
- (7) 转步骤 (3) 重试;
- (8) **endif**
- (9) **for**  $D_i$  的每个被正确分类的元组 **do**
- (10) 元组的权重乘以  $\text{error}(M_i) / (1 - \text{error}(M_i))$ ; // 更新权重
- (11) 规范化每个元组的权重;
- (12) **endfor**

使用组合分类器对元组  $x$  分类:

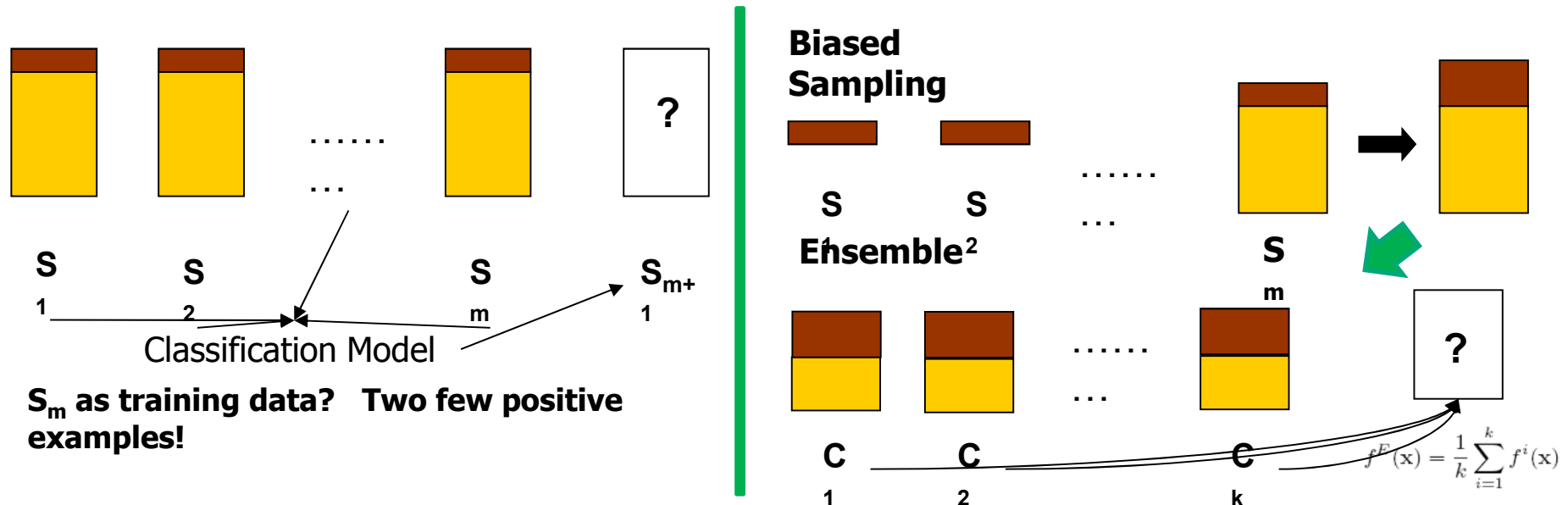
- (1) 将每个类的权重初始化为 0;
- (2) **for**  $i = 1$  **to**  $k$  **do** // 对于每个分类器
- (3)  $w_i = \log \frac{1 - \text{error}(M_i)}{\text{error}(M_i)}$ ; // 分类器的投票权重
- (4)  $c = M_i(x)$ ; // 从  $M_i$  得到  $x$  的类预测
- (5) 将  $w_i$  加到类  $c$  的权重;
- (6) **endfor**
- (7) 返回具有最大权重的类;

# Classification of Class-Imbalanced Data Sets

- Class-imbalance problem: Rare positive examples but numerous negative ones
  - E.g., medical diagnosis, fraud transaction, accident (oil-spill), and product fault
- Traditional methods assume a balanced distribution of classes and equal error costs: not suitable for class-imbalanced data
- Typical methods on imbalanced data in two-class classification
  - **Oversampling:** Re-sampling of data from positive class
  - **Under-sampling:** Randomly eliminate tuples from negative class
  - **Threshold-moving:** Move the decision threshold,  $t$ , so that the rare class tuples are easier to classify, and hence, less chance of costly false negative errors
  - **Ensemble techniques:** Ensemble multiple classifiers introduced above
- Still difficult for class imbalance problem on multiclass tasks



# Classifying Data Streams with Skewed Distribution




- Classify data stream with skewed distribution (i.e., rare events)
  - **Biased sampling:** Save only the positive examples in the streams
  - **Ensemble:** Partition negative examples of  $S_m$  into  $k$  portions to build  $k$  classifiers
  - Effectively reduce classification errors on the minority class

J. Gao, et al., "A General Framework for Mining Concept-Drifting Data Streams with Skewed Distributions", SDM'07

# Agenda

---

- Classification: Basic Concepts
- Decision Tree Induction
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy:  
Ensemble Methods
- Bayes Classification Methods 
- Summary



# Bayesian Classification: Why?

- A statistical classifier: performs *probabilistic prediction*, i.e., predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data
- Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

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

# Bayesian Theorem

- Given training data  $\mathbf{X}$ , *posteriori probability of a hypothesis*  $H$ ,  $P(H|\mathbf{X})$ , 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 written as  
posteriori = likelihood x prior/evidence
- Predicts  $\mathbf{X}$  belongs to  $C_2$  iff the probability  $P(C_i|\mathbf{X})$  is the highest among all the  $P(C_k|\mathbf{X})$  for all the  $k$  classes
- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

# Towards Naïve Bayesian Classifier

---

- Let  $D$  be a training set of tuples and their associated class labels, and each tuple is represented by an  $n$ -D attribute vector  $\mathbf{X} = (x_1, x_2, \dots, x_n)$
- Suppose there are  $m$  classes  $C_1, C_2, \dots, C_m$ .
- Classification is to derive the maximum posteriori, i.e., the maximal  $P(C_i | \mathbf{X})$
- This can be derived from Bayes' theorem

$$P(C_i | \mathbf{X}) = \frac{P(\mathbf{X} | C_i)P(C_i)}{P(\mathbf{X})}$$

- Since  $P(\mathbf{X})$  is constant for all classes, only

needs to be maximized

$$P(C_i | \mathbf{X}) = P(\mathbf{X} | C_i)P(C_i)$$

# Derivation of Naïve Bayes Classifier

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

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

- 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  $\mu$  and standard deviation  $\sigma$

and  $P(x_k | C_i)$  is

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

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

# Naïve Bayesian Classifier: Training Dataset

Class:

C1:buys\_computer = 'yes'

C2:buys\_computer = 'no'

Data sample

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 Bayesian Classifier: An Example

- $P(C_i)$ :  $P(\text{buys\_computer} = \text{"yes"}) = 9/14 = 0.643$   
 $P(\text{buys\_computer} = \text{"no"}) = 5/14 = 0.357$
  - Compute  $P(X|C_i)$  for each class
    - $P(\text{age} = \text{"<=30"} | \text{buys\_computer} = \text{"yes"}) = 2/9 = 0.222$
    - $P(\text{age} = \text{"<= 30"} | \text{buys\_computer} = \text{"no"}) = 3/5 = 0.6$
    - $P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"yes"}) = 4/9 = 0.444$
    - $P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$
    - $P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$
    - $P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"no"}) = 1/5 = 0.2$
    - $P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$
    - $P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$
  - **$X = (\text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit\_rating} = \text{fair})$** 
    - $P(X|C_i) : P(X|\text{buys\_computer} = \text{"yes"}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044$   
 $P(X|\text{buys\_computer} = \text{"no"}) = 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019$
    - $P(X|C_i) * P(C_i) : P(X|\text{buys\_computer} = \text{"yes"}) * P(\text{buys\_computer} = \text{"yes"}) = 0.028$   
 $P(X|\text{buys\_computer} = \text{"no"}) * P(\text{buys\_computer} = \text{"no"}) = 0.007$
- 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

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

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10)
- Use **Laplacian correction** (or Laplacian estimator)
  - *Adding 1 to each case*  
Prob(income = low) = 1/1003  
Prob(income = medium) = 991/1003  
Prob(income = high) = 11/1003
  - The “corrected” prob. estimates are close to their “uncorrected” counterparts



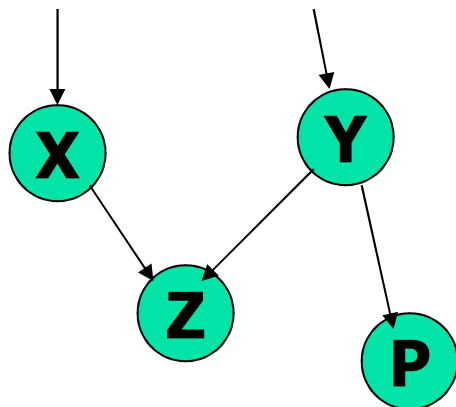
# Naïve Bayesian Classifier: Comments

---

- 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 Bayesian Classifier
- How to deal with these dependencies? Bayesian Belief Networks

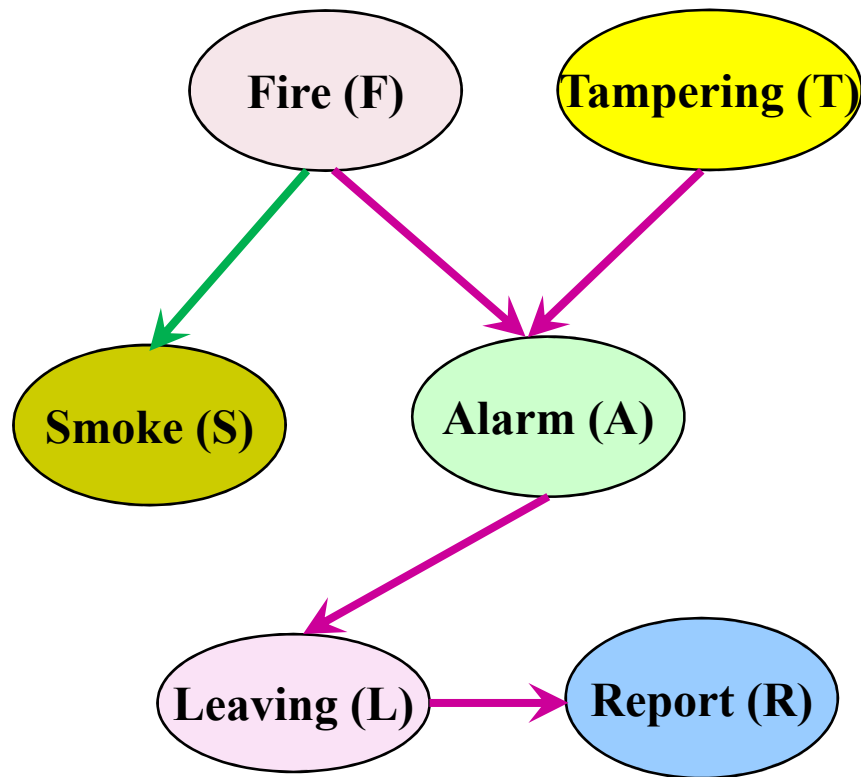
# Bayesian Belief Networks

- **Bayesian belief network** (also known as **Bayesian network**, **probabilistic network**): allows *class conditional independencies* between *subsets* of variables
- Two components: (1) A *directed acyclic graph* (called a structure) and (2) a set of *conditional probability tables* (CPTs)
- A (*directed acyclic*) graphical model of *causal influence* relationships
  - Represents dependency among the variables
  - Gives a specification of joint probability distribution



- ☐ Nodes: random variables
- ☐ Links: dependency
- ☐ X and Y are the parents of Z, and Y is the parent of P
- ☐ No dependency between Z and P
- ☐ Has no loops/cycles

# A Bayesian Network and Some of Its CPTs



## CPT: Conditional Probability Tables

Fire	Smoke	$\Theta_{s f}$
True	True	.90
False	True	.01

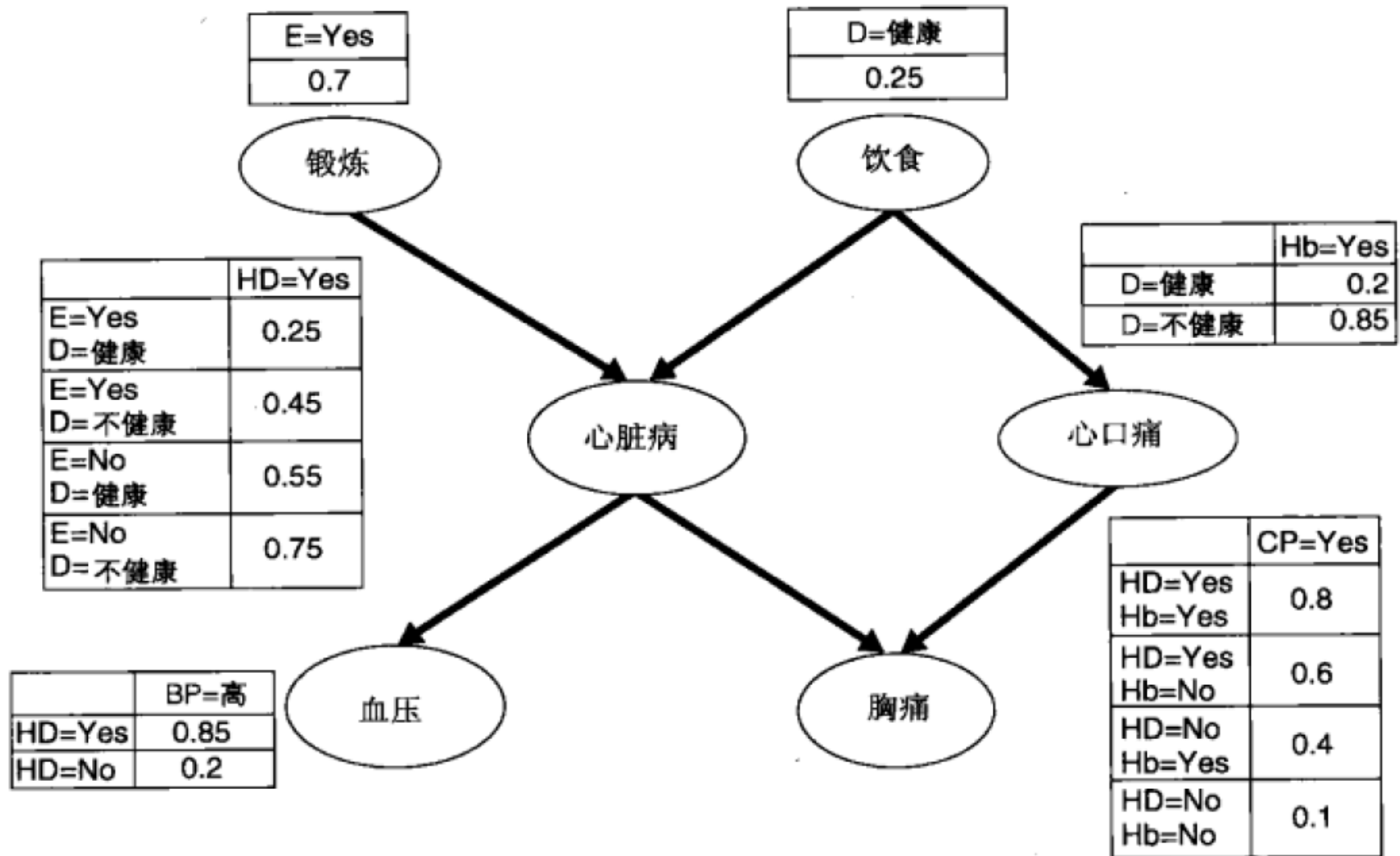
Fire	Tampering	Alarm	$\Theta_{a f,t}$
True	True	True	.5
True	False	True	.99
False	True	True	.85
False	False	True	.0001

CPT shows the conditional probability for each possible combination of its parents

Derivation of the probability of a particular combination of values of  $\mathbf{X}$ , from CPT:

$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i | \text{Parents}(x_i))$$

$$\begin{aligned}
 &P(\text{心脏病}=\text{No}|\text{锻炼}=\text{No}, \text{饮食}=\text{健康}) \\
 &= 1 - P(\text{心脏病}=\text{Yes}|\text{锻炼}=\text{No}, \text{饮食}=\text{健康}) \\
 &= 1 - 0.55 = 0.45
 \end{aligned}$$



---

情况一：没有先验信息 在没有任何先验信息的情况下，可以通过计算先验概率  $P(\text{HD}=\text{Yes})$  和  $P(\text{HD}=\text{No})$  来确定一个人是否可能患心脏病。为了表述方便，设  $\alpha \in \{\text{Yes}, \text{No}\}$  表示锻炼的两个值， $\beta \in \{\text{健康}, \text{不健康}\}$  表示饮食的两个值。

$$\begin{aligned} P(\text{HD}=\text{Yes}) &= \sum_{\alpha} \sum_{\beta} P(\text{HD}=\text{Yes} | E=\alpha, D=\beta) P(E=\alpha, D=\beta) \\ &= \sum_{\alpha} \sum_{\beta} P(\text{HD}=\text{Yes} | E=\alpha, D=\beta) P(E=\alpha) P(D=\beta) \\ &= 0.25 \times 0.7 \times 0.25 + 0.45 \times 0.7 \times 0.75 + 0.55 \times 0.3 \times 0.25 + 0.75 \times 0.3 \times 0.75 \\ &= 0.49 \end{aligned}$$

因为  $P(\text{HD}=\text{No}) = 1 - P(\text{HD}=\text{Yes}) = 0.51$ ，所以，此人不得心脏病的机率略微大一点。

情况二：高血压 如果一个人有高血压，可以通过比较后验概率  $P(\text{HD}=\text{Yes}|\text{BP}=\text{高})$  和  $P(\text{HD}=\text{No}|\text{BP}=\text{高})$  来诊断他是否患有心脏病。为此，我们必须先计算  $P(\text{BP}=\text{高})$ ：

$$\begin{aligned} P(\text{BP}=\text{高}) &= \sum_{\gamma} P(\text{BP}=\text{高}|\text{HD}=\gamma)P(\text{HD}=\gamma) \\ &= 0.85 \times 0.49 + 0.2 \times 0.51 = 0.5185 \end{aligned}$$

其中  $\gamma \in \{\text{Yes}, \text{No}\}$ 。因此，此人患心脏病的后验概率是：

$$\begin{aligned} P(\text{HD}=\text{Yes}|\text{BP}=\text{高}) &= \frac{P(\text{BP}=\text{高}|\text{HD}=\text{Yes})P(\text{HD}=\text{Yes})}{P(\text{BP}=\text{高})} \\ &= \frac{0.85 \times 0.49}{0.5185} = 0.8033 \end{aligned}$$

同理， $P(\text{HD}=\text{No}|\text{BP}=\text{高}) = 1 - 0.8033 = 0.1967$ 。因此，当一个人有高血压时，他患心脏病的危险就增加了。

---

情况三：高血压、饮食健康、经常锻炼身体 假设得知此人经常锻炼身体并且饮食健康。这些新信息会对诊断造成怎样的影响呢？加上这些新信息，此人患心脏病的后验概率：

$$\begin{aligned}& P(\text{HD}=\text{Yes}|\text{BP}=\text{高}, D=\text{健康}, E=\text{Yes}) \\&= \left[ \frac{P(\text{BP}=\text{高}|\text{HD}=\text{Yes}, D=\text{健康}, E=\text{Yes})}{P(\text{BP}=\text{高}|D=\text{健康}, E=\text{Yes})} \right] \times P(\text{HD}=\text{Yes}|D=\text{健康}, E=\text{Yes}) \\&= \frac{P(\text{BP}=\text{高}|\text{HD}=\text{Yes})P(\text{HD}=\text{Yes}|D=\text{健康}, E=\text{Yes})}{\sum_{\gamma} P(\text{BP}=\text{高}|\text{HD}=\gamma)P(\text{HD}=\gamma|D=\text{健康}, E=\text{Yes})} \\&= \frac{0.85 \times 0.25}{0.85 \times 0.25 + 0.2 \times 0.75} \\&= 0.5862\end{aligned}$$

而此人不患心脏病的概率是：

$$P(\text{HD}=\text{No}|\text{BP}=\text{高}, D=\text{健康}, E=\text{Yes}) = 1 - 0.5862 = 0.4138$$

因此模型暗示健康的饮食和有规律的体育锻炼可以降低患心脏病的危险。

# How Are Bayesian Networks Constructed?

- **Subjective construction:** Identification of (direct) causal structure
  - People are quite good at identifying direct causes from a given set of variables & whether the set contains all relevant direct causes
  - Markovian assumption: Each variable becomes independent of its non-effects once its direct causes are known
  - E.g.,  $S \leftarrow F \rightarrow A \leftarrow T$ , path  $S \rightarrow A$  is blocked once we know  $F \rightarrow A$
  - HMM (Hidden Markov Model): often used to model dynamic systems whose states are not observable, yet their outputs are
- **Synthesis from other specifications**
  - E.g., from a formal system design: block diagrams & info flow
- **Learning from data**
  - E.g., from medical records or student admission record
  - Learn parameters give its structure or learn both structure and parms
  - Maximum likelihood principle: favors Bayesian networks that maximize the probability of observing the given data set




# Training Bayesian Networks: Several Scenarios

---

- Scenario 1: Given both the network structure and all variables observable: *compute only the CPT entries*
- Scenario 2: Network structure known, some variables hidden: *gradient descent* (greedy hill-climbing) method, i.e., search for a solution along the steepest descent of a criterion function
  - Weights are initialized to random probability values
  - At each iteration, it moves towards what appears to be the best solution at the moment, w.o. backtracking
  - Weights are updated at each iteration & converge to local optimum
- Scenario 3: Network structure unknown, all variables observable: search through the model space to *reconstruct network topology*
- Scenario 4: Unknown structure, all hidden variables: No good algorithms known for this purpose
- D. Heckerman. [A Tutorial on Learning with Bayesian Networks](#). In *Learning in Graphical Models*, M. Jordan, ed. MIT Press, 1999.

# Agenda

---

- Classification: Basic Concepts
- Decision Tree Induction
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy:  
Ensemble Methods
- Bayes Classification Methods
- Summary 

# Summary (I)

---

- **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  $F_\beta$  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.

## Summary (II)

---

- **Significance tests** and **ROC curves** are useful for model selection.
- There have been numerous **comparisons of the different classification** methods; the matter remains a research topic
- No single method has been found to be superior over all others for all data sets
- Issues such as accuracy, training time, robustness, scalability, and interpretability must be considered and can involve trade-offs, further complicating the quest for an overall superior method

# Summary (III)

---

- Effective and advanced classification methods
  - Bayesian belief network (probabilistic networks)
  - Backpropagation (Neural networks)
  - Support Vector Machine (SVM)
  - Pattern-based classification
  - Other classification methods: lazy learners (KNN, case-based reasoning), genetic algorithms, rough set and fuzzy set approaches
- Additional Topics on Classification
  - Multiclass classification
  - Semi-supervised classification
  - Active learning
  - Transfer learning

---

*Thank You!*