

Classification and Prediction

Introduction

Databases are rich with hidden information that can be used for **intelligent decision making**

Classification and prediction are two forms of data analysis that can be used to **extract models** describing important data classes or to predict future data trends.

Classification predicts categorical (discrete, unordered) labels, prediction models continuous valued functions.

Ex: Build a classification model to categorize bank loan applications as either safe or risky. Prediction model to predict the expenditures in dollars of potential customers given their income and occupation.

Most algorithms are **memory resident**, assuming a small data size. Recent data mining research developing scalable classification and prediction techniques capable of handling **large disk-resident data**.

What is Classification?

A bank loans officer needs analysis of her data in order to learn which loan applicants are “safe” and which are “risky” for the bank.

A marketing manager at AllElectronics needs data analysis to help guess whether a customer with a given profile will buy a new computer.

A medical researcher wants to analyze breast cancer data in order to predict which one of three specific treatments a patient should receive.

In each of these examples, the data analysis task is **classification**, where a model or classifier is constructed to **predict categorical labels**,

- “safe” or “risky” for the loan application data;
- “yes” or “no” for the marketing data;
- “treatment A,” “treatment B,” or “treatment C” for the medical data.

What is Prediction?

A **marketing manager** would like to predict how much a given customer will spend during a sale at AllElectronics.

This data analysis task is an example of numeric prediction, where the model constructed predicts a **continuous-valued function**, or ordered value. This model is a **predictor**.

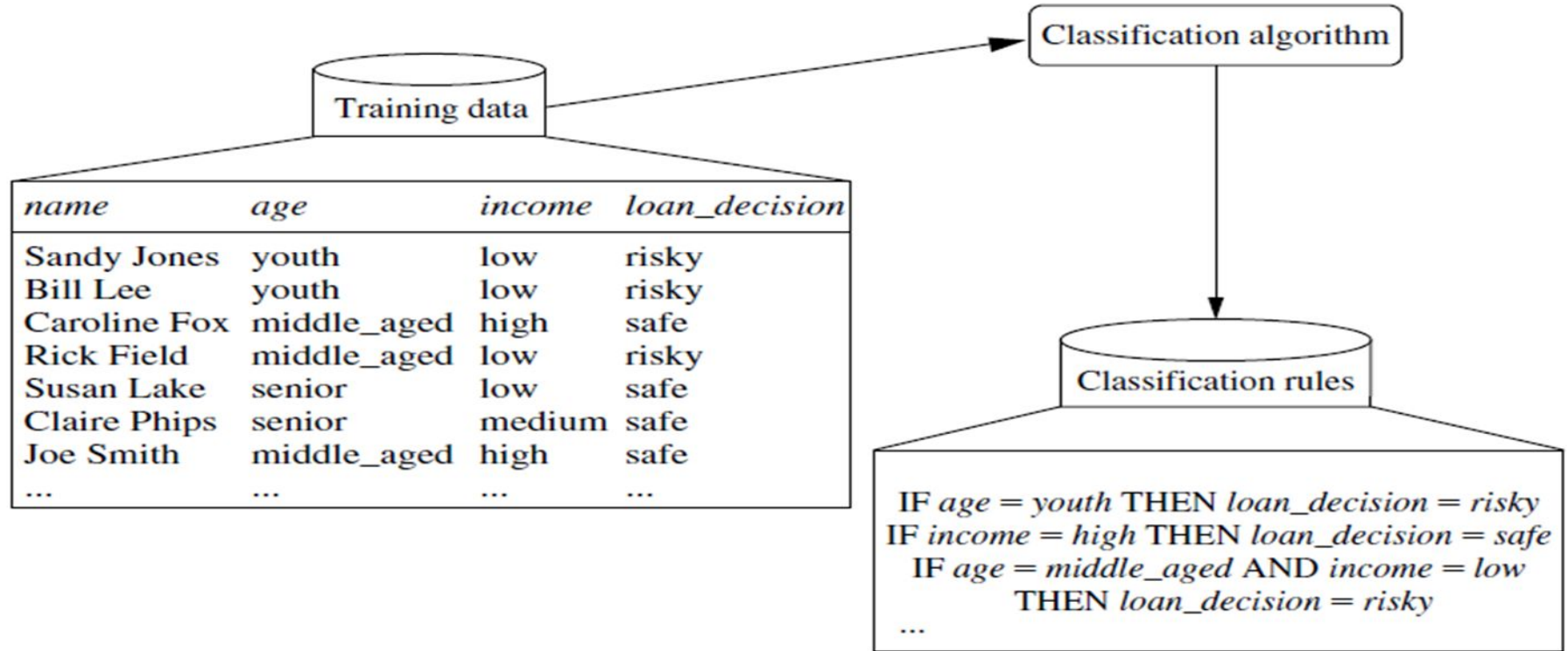
Regression analysis is a statistical methodology that is most often used for numeric prediction.

Classification and numeric prediction are the two major types of prediction problems.

General approach to Classification

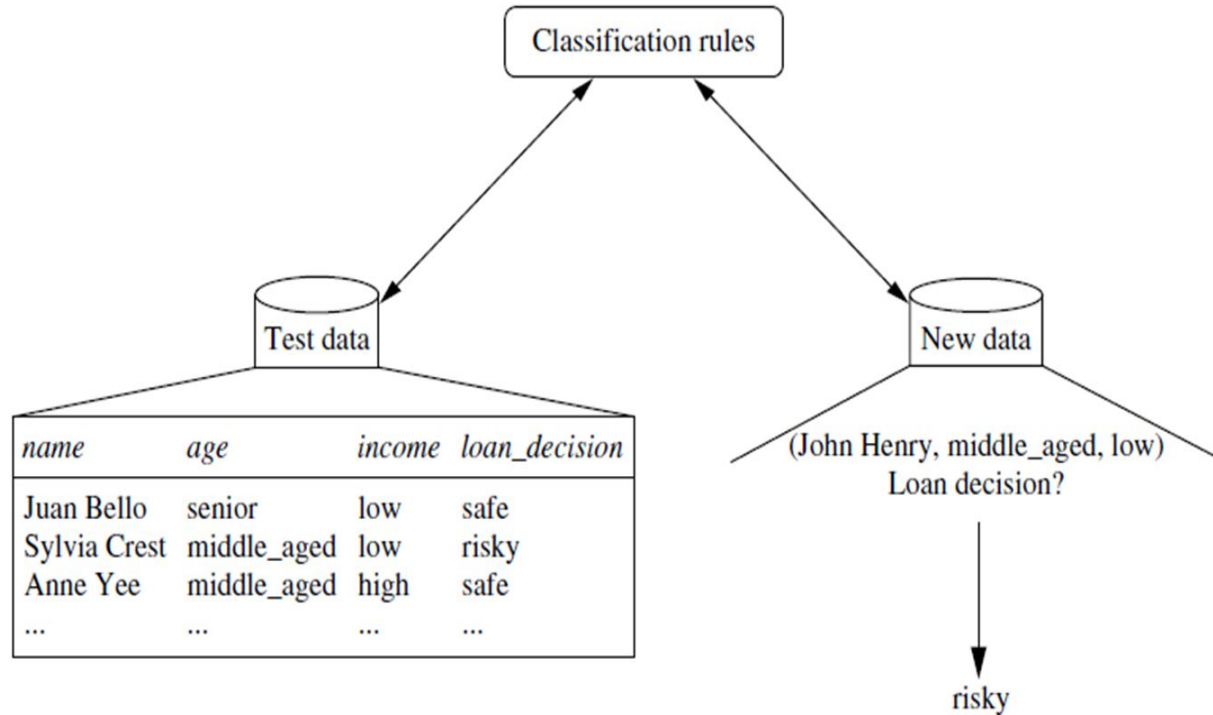
- **Data classification** is a two-step process,
 - *Learning* - where a classification model is constructed(training phase)
 - *Classification* - where the model is used to predict class labels for given data(testing phase)
- **Learning step:** a classifier or a classification algorithm builds the classifier by analyzing or “learning from” a **training set**
 - **Training set** made up of database tuples and their associated class labels.
 - The class label attribute is **discrete-valued and unordered** and *categorical* (or nominal) in that each value serves as a category or class.
- **Classification Step**
 - **Training tuples** are randomly sampled from the database under analysis.
 - Data tuples can be referred to as *samples, examples, instances, data points, or objects*
 - Because the class label of each training tuple *is provided*, this step is also known as **supervised learning**

General approach to Classification



•**Learning:** Training data are analyzed by a classification algorithm. Here, *loan_decision* is the class label attribute, and classification rules represents the learned model or classifier.

General approach to Classification



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

General approach to Classification

- This first step of the classification process can also be viewed as the learning of a mapping or function, $y=f(X)$
 - X is a test data tuple input to a function
 - y is a associated class label as output.
- This mapping is represented in the form of classification rules, decision trees, or mathematical formulae.

Classification Accuracy

First, the predictive accuracy of the classifier is estimated.

If we were to use the training set to measure the classifier's accuracy, this estimate would likely be optimistic, because the classifier tends to overfit the data.

Therefore, a test set is used, made up of test tuples and their associated class labels. They are independent of the training tuples, meaning that they were not used to construct the classifier.

The accuracy of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier. The associated class label of each test tuple is compared with the learned classifier's class prediction for that tuple.

Issues Regarding Classification and Prediction

The following preprocessing steps may be applied to the data to help improve the accuracy, efficiency, and scalability of the classification or prediction process.

Data cleaning: To remove or reduce noise and the treatment of missing values. Although most classification algorithms have some mechanisms for handling noisy or missing data, this step can help reduce confusion during learning.

Relevance analysis: **Correlation analysis** can be used to identify whether any two given attributes are statistically related.

For example, a strong correlation between attributes A1 and A2 would suggest that one of the two could be removed from further analysis.

Attribute subset selection can be used to find a reduced set of attributes such that the resulting probability distribution of the data classes is as close as possible to the original distribution obtained using all attributes.

Such analysis can help improve classification efficiency and scalability.

Issues Regarding Classification and Prediction

Data transformation and reduction

The data may be transformed by normalization, particularly when neural networks or methods involving distance measurements are used in the learning step.

The data can also be transformed by generalizing it to higher-level concepts. Concept hierarchies may be used for this purpose. This is particularly useful for continuous valued attributes.

Because generalization compresses the original training data, fewer input/output operations may be involved during learning.

Data can also be reduced by applying many other methods, ranging from wavelet transformation and principle components analysis to discretization techniques, such as binning, histogram analysis, and clustering.

Comparing Classification and Prediction Methods

Classification and prediction methods can be compared and evaluated according to the following criteria:

Accuracy: The accuracy refers to the ability of a given classifier to correctly predict the class label of new or previously unseen data (i.e., tuples without class label information). Similarly, the accuracy of a predictor refers to how well a given predictor can guess the value of the predicted attribute for new or previously unseen data.

Speed: This refers to the computational costs involved in generating and using the given classifier or predictor.

Robustness: This is the ability of the classifier or predictor to make correct predictions given noisy data or data with missing values.

Scalability: This refers to the ability to construct the classifier or predictor efficiently given large amounts of data.

Interpretability: This refers to the level of understanding and insight that is provided by the classifier or predictor. Interpretability is subjective and therefore more difficult to assess.

Classification by Decision Tree Induction

Decision tree induction is the learning of decision trees from class-labeled training tuples.

A decision tree is a flowchart-like tree structure,

- each **internal node** (non leaf node) denotes a test on an attribute
- each **branch** represents an outcome of the test
- each **leaf node** (or terminal node) holds a class label.

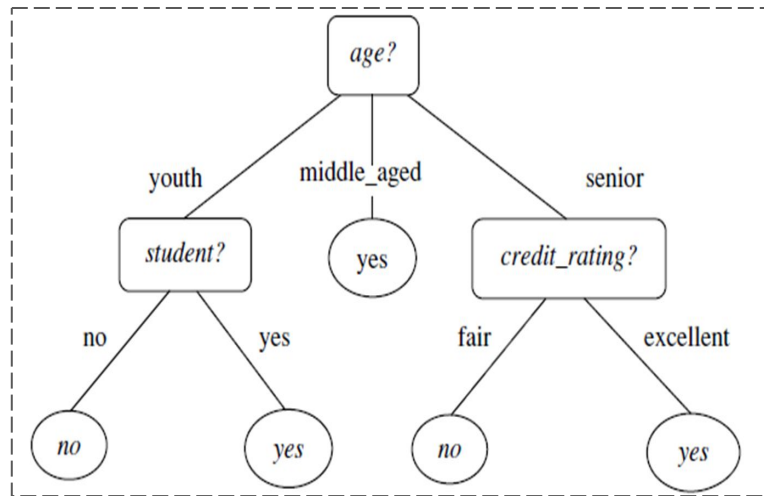


Fig: A decision tree for the concept *buys_computer*, indicating whether an *AllElectronics* customer is likely to purchase a computer.

Classification by Decision Tree Induction

“How are decision trees used for classification?”

- Given a tuple, X , for which the associated **class label is unknown**, the attribute values of the tuple are tested against the decision tree.
- A path is traced from the root to a leaf node, which holds the class prediction for that tuple. Decision trees can easily be converted to classification rules.

“Why are decision tree classifiers so popular?”

- The construction of decision tree classifiers does not require any domain knowledge or parameter setting.
- Decision trees can handle multidimensional data.
- The learning and classification steps of decision tree induction are simple and fast with good accuracy.
- Widely used for classification in many application areas such as medicine, manufacturing and production, financial analysis, astronomy, and molecular biology, etc.,

Decision Tree Induction Algorithm

- During the late 1970s and early 1980s, J. Ross Quinlan, a researcher in machine learning, developed a decision tree algorithm known as **ID3 (Iterative Dichotomiser)**.
- This work expanded on earlier work on **concept learning systems**, described by E. B. Hunt, J. Marin, and P. T. Stone. Quinlan later presented **C4.5 (a successor of ID3)**, which became a benchmark to which newer supervised learning algorithms are often compared.
- In 1984, a group of statisticians (L. Breiman, J. Friedman, R. Olshen, and C. Stone) published the book Classification and Regression Trees (CART), which described the generation of binary decision trees.

Decision Tree Induction Algorithm

- ID3, C4.5, and CART adopt a greedy (i.e., non backtracking) approach in which decision trees are constructed in a **top-down recursive divide-and-conquer** manner.
- Starts with a training set of tuples and their associated class labels. The training set is recursively partitioned into smaller subsets as the tree is being built.

Basic algorithm for inducing a decision tree from training tuples

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.

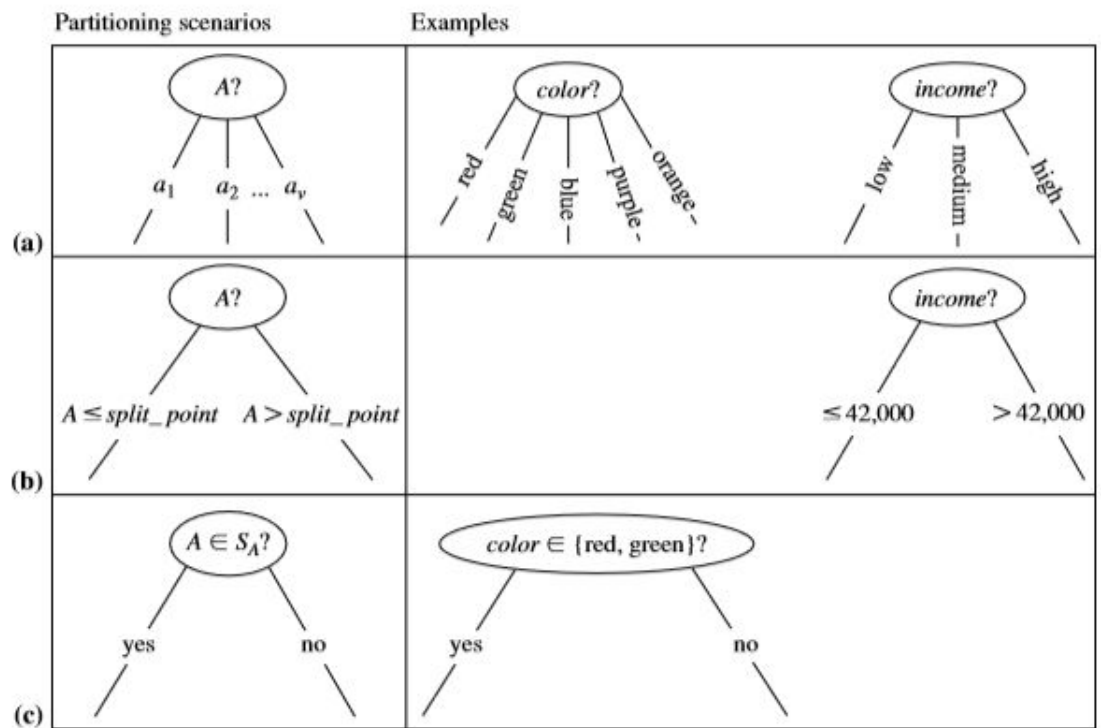
Method:

- (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* $\leftarrow$ *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
- (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 ;

Information Gain
Gain Ratio
Gini Index

Pure

The node N is labeled with the splitting criterion (step 6)



This figure shows three possibilities for partitioning tuples based on the splitting criterion, each with examples. Let A be the splitting attribute. (a) If A is discrete-valued, then one branch is grown for each known value of A . (b) If A is continuous-valued, then two branches are grown, corresponding to $A \leq \text{split_point}$ and $A > \text{split_point}$. (c) If A is discrete-valued and a binary tree must be produced, then the test is of the form $A \in S_A$, where S_A is the splitting subset for A .

Decision Tree Induction Algorithm

The recursive partitioning stops only when any one of the following terminating conditions is true:

- All the tuples in partition D (represented at node N) belong to the same class (steps 2 and 3).
- There are no remaining attributes on which the tuples may be further partitioned (step 4). In this case, majority voting is employed (step 5). This involves converting node N into a leaf and labeling it with the most common class in D.
- There are no tuples for a given branch, that is, a partition D_j is empty (step 12). In this case, a leaf is created with the majority class in D (step 13).

The computational complexity of the algorithm given training set D is

$$O(n \times |D| \times \log(|D|))$$

where n is the number of attributes describing the tuples in D

$|D|$ is the number of training tuples in D

Attribute Selection Measures: Decision Tree Induction

- An attribute selection measure is a heuristic for selecting the splitting criterion that “best” separates given data partition, D , of class-labeled training tuples into individual classes.
- Provides a ranking for each attribute describing the given training tuples.
- The attribute having the **best score** for the measure is chosen as the splitting attribute for the given tuples.
- If the splitting attribute is **continuous-valued** or if we are restricted to binary trees, then, either a **split point or a splitting subset** must also be determined as part of the splitting criterion.

Three popular attribute selection measures are:

—**Information gain, gain ratio, and Gini index.**

Attribute Selection Measures: Information Gain

Used Notations:

D , be a training set of class-labeled tuples.

m is the number of distinct values defining m distinct classes, C_i (for $i=1, \dots, m$).

$C_{i,D}$ be the set of tuples of class C_i in D .

$|D|$ and $|C_{i,D}|$ denote the number of tuples in D and $C_{i,D}$, respectively.

Attribute Selection Measures: Information Gain

- ID3 uses **information gain** as its attribute selection measure.
- Let node N represent or hold the tuples of partition D . Choose the attribute with highest information gain as a splitting attribute for node N .
- This attribute minimizes the information needed to classify the tuples in the resulting portions and reflects the least randomness or **impurity** in the partitions.
- The expected information needed to classify a tuple D is given by

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

- p_i is the non zero probability that an arbitrary tuple in D belongs to class C_i and is estimated by $|C_{i,d}|/|D|$
- Log function to the base 2 is used, because the information is encoded in bits
- $Info(D)$ is also known as **Entropy** of D , is a avg amount of info needed to identify the class label of a tuple in D

Attribute Selection Measures: Information Gain

How much more information would we still need (after the partitioning) to arrive at an exact classification? This amount is measured by

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

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

$Info_A(D)$ is the **expected information** required to classify a tuple from D based on the partitioning by A . The **smaller the expected information** (still) required, the greater the purity of the partitions.

Attribute Selection Measures: Information Gain

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

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

Class-Labeled Training Tuples from the *AllElectronics* Customer Database

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

Example

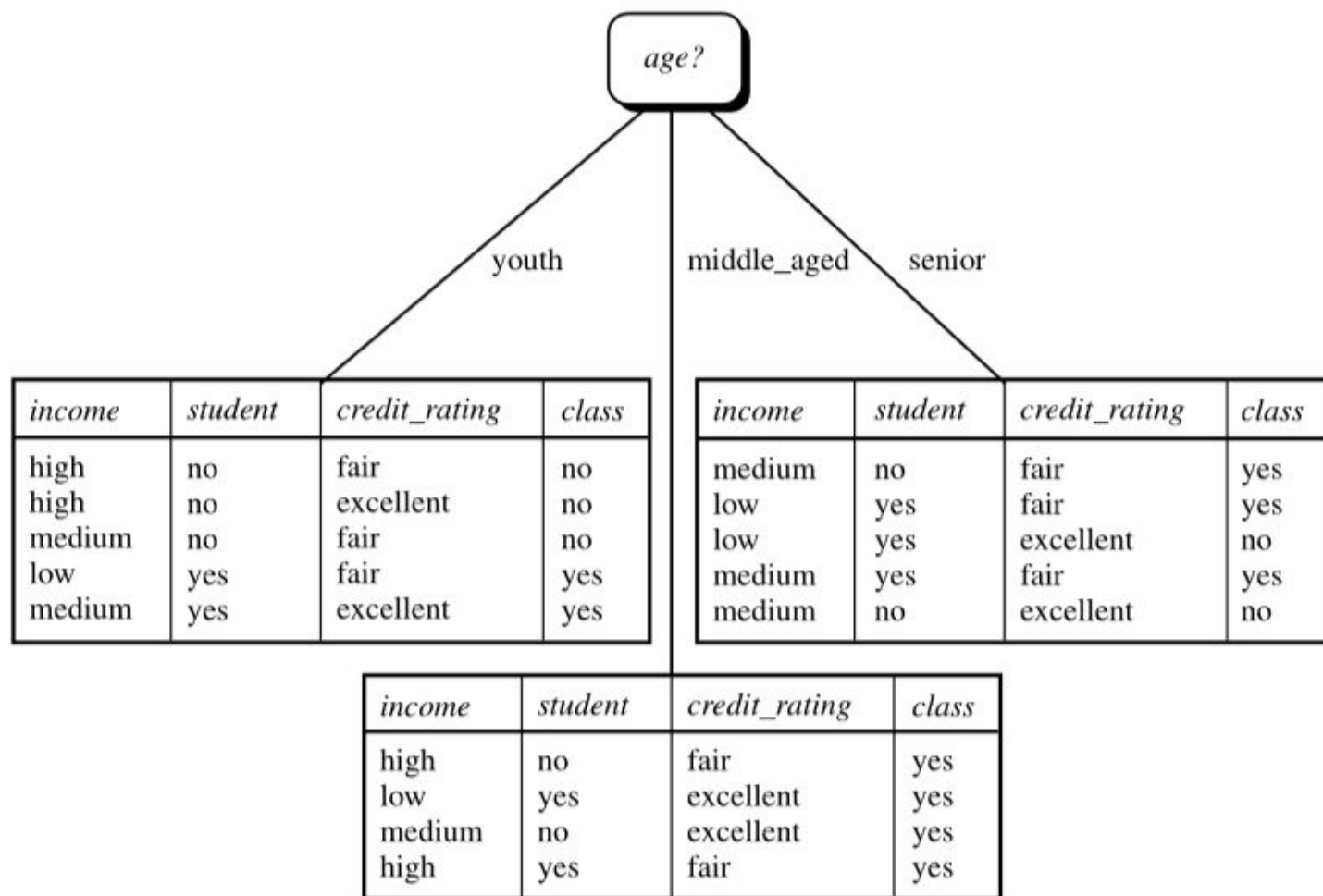
Classification by Decision Tree Induction

A (root) node N is created for the tuples in D. To find the splitting criterion for these tuples, we must compute the information gain of each attribute.

$$Info(D) = -\sum_{i=1}^m p_i \log_2(p_i) \quad Info(D) = -\frac{9}{14} \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits.}$$

Next, compute the expected information requirement for each attribute.

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



Classification by Decision Tree Induction

Solve Problem

Bayesian Classification

Bayesian classifiers are statistical classifiers

Predict class membership probabilities (the probability that a given tuple belongs to a particular class)

Comparing classification algorithms have found a simple Bayesian classifier known as the naïve Bayesian classifier

Bayesian classifiers have also exhibited high accuracy and speed when applied to large databases

Naïve Bayesian classifiers assume that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is called class conditional independence.

Bayes' Theorem

Let X be a data tuple. In Bayesian terms, X is considered “evidence.”

Let H be some hypothesis such as that the data tuple X belongs to a specified class C .

For **classification problems**, we want to determine $P(H|X)$, the probability that the hypothesis H holds given the “evidence” or observed data tuple X .

In other words, we are looking for the probability that tuple X belongs to class C , given that we know the attribute description of X .

Bayes' Theorem

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

$P(H|X)$ is the **posterior probability** of H conditioned on X .

Ex: customers described by the attributes age and income.

X is a 35-year-old customer with an income of \$40,000.

Suppose that H is the hypothesis that our customer will buy a computer.

Then $P(H|X)$ reflects the probability that customer X will buy a computer given that we know the customer's age and income.

$P(H)$ is the **prior probability** of H

This is the probability that any given customer will buy a computer, regardless of age, income, or any other information, for that matter.

$P(H)$, which is independent of X .

Bayes' Theorem

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

$P(X|H)$ is the posterior probability of X conditioned on H .

That is, it is the probability that a customer, X , is 35 years old and earns \$40,000, given that we know the customer will buy a computer.

“How are these probabilities estimated?”

$P(H)$, $P(X|H)$, and $P(X)$ may be estimated from the given data.

Bayes' theorem is useful in that it provides a way of calculating the posterior probability, $P(H|X)$, from $P(H)$, $P(X|H)$, and $P(X)$.

Naïve Bayesian Classification

Naïve Bayesian classifier.

1. Let D be a training set of tuples and their associated class labels. Each tuple is represented by an n -dimensional attribute vector, $X=(x_1, x_2, \dots, x_n)$, depicting n measurements made on the tuple from n attributes, respectively, $A_1, A_2, \dots, A_n$.
2. There are m classes, $C_1, C_2, \dots, C_m$. Given a tuple, X , the classifier will predict that X belongs to the class having the **highest posterior probability**, conditioned on X .

That is, the naïve Bayesian classifier predicts that tuple X belongs to the class C_i if and only if

$$P(C_i|X) > P(C_j|X) \quad \text{for } 1 \leq j \leq m, j \neq i.$$

Naïve Bayesian Classification

Thus, we maximize $P(C_i|X)$. The class C_i for which $P(C_i|X)$ is maximized is called the maximum posteriori hypothesis.

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

Naïve Bayesian Classification

3. As $P(X)$ is constant for all classes, only $P(X|C_i)P(C_i)$ needs to be maximized.

If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, that is, $P(C_1)=P(C_2)=\dots=P(C_m)$, and we would therefore maximize $P(X|C_i)$.

Otherwise, we maximize $P(X|C_i)P(C_i)$.

Class prior probabilities may be estimated by $P(C_i)=|C_i,D|/|D|$

Naïve Bayesian Classification

4. Given data sets with many attributes, it would be extremely computationally expensive to compute $P(X|C_i)$.

To reduce computation in evaluating $P(X|C_i)$, the naïve assumption of **class-conditional independence** is made.

This presumes that the attributes' values are conditionally independent of one another, given the class label of the tuple

$$\begin{aligned} P(X|C_i) &= \prod_{k=1}^n P(x_k|C_i) \\ &= P(x_1|C_i) \times P(x_2|C_i) \times \cdots \times P(x_n|C_i). \end{aligned}$$

Naïve Bayesian Classification

For each attribute, we look at whether the attribute is categorical or continuous-valued.

- (a) If A_k is categorical, then $P(x_k|C_i)$ is the number of tuples of class C_i in D having the value x_k for A_k , divided by $|C_i, D|$, the number of tuples of class C_i in D .
- (b) If A_k is continuous-valued, is typically assumed to have a Gaussian distribution with a mean μ and standard deviation σ , defined by

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

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

Naïve Bayesian Classification

5. To predict the class label of X , $P(X|C_i)P(C_i)$ is evaluated for each class C_i . The classifier predicts that the class label of tuple X is the class C_i if and only if

$$P(X|C_i)P(C_i) > P(X|C_j)P(C_j) \quad \text{for } 1 \leq j \leq m, j \neq i.$$

Under certain assumptions, it can be shown that many neural network and curve-fitting algorithms output the *maximum posteriori hypothesis*, as does the naïve Bayesian classifier

Predict a class label using Naïve Bayesian Classification

Let C1 correspond to the class buys_computer = yes and C2 correspond to buys_computer = no.

The tuple we wish to classify is

$X = (\text{age} = \text{youth}, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit rating} = \text{fair})$

We need to maximize $P(X|C_i)P(C_i)$, for $i=1, 2$.

Naïve Bayesian Classification

The prior probability of each class,

$$P(\text{buys computer}=\text{yes})=9/14=0.643$$

$$P(\text{buys computer}=\text{no})=5/14=0.357$$

Naïve Bayesian Classification

To compute $P(X|C_i)$, for $i=1,2$, we compute the following conditional probabilities

$$P(\text{age}=\text{youth}|\text{buys computer}=\text{yes}) = 2/9 = 0.222$$

$$P(\text{age}=\text{youth}|\text{buys computer}=\text{no}) = 3/5 = 0.600$$

$$P(\text{income}=\text{medium}|\text{buys computer}=\text{yes}) = 4/9 = 0.444$$

$$P(\text{income}=\text{medium}|\text{buys computer}=\text{no}) = 2/5 = 0.400$$

$$P(\text{student}=\text{yes}|\text{buys computer}=\text{yes}) = 6/9 = 0.667$$

$$P(\text{student}=\text{yes}|\text{buys computer}=\text{no}) = 1/5 = 0.200$$

$$P(\text{credit rating}=\text{fair}|\text{buys computer}=\text{yes}) = 6/9 = 0.667$$

$$P(\text{credit rating}=\text{fair}|\text{buys computer}=\text{no}) = 2/5 = 0.400$$

Naïve Bayesian Classification

$$\begin{aligned} P(X|\text{buys computer}=\text{yes}) &= P(\text{age}=\text{youth}|\text{buys computer}=\text{yes}) \times \\ &P(\text{income}=\text{medium}|\text{buys computer}=\text{yes}) \times P(\text{student}=\text{yes}|\text{buys computer}=\text{yes}) \times \\ &P(\text{credit rating}=\text{fair}|\text{buys computer}=\text{yes}) \\ &= 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044. \end{aligned}$$

Similarly, $P(X|\text{buys_computer} = \text{no}) = 0.600 \times 0.400 \times 0.200 \times 0.400 = 0.019$.

To find the class, C_i , that maximizes $P(X|C_i)P(C_i)$, we compute

$$P(X|\text{buys_computer} = \text{yes}) P(\text{buys_computer}=\text{yes}) = 0.044 \times 0.643 = 0.028$$

$$P(X|\text{buys_computer} = \text{no}) P(\text{buys_computer}=\text{no}) = 0.019 \times 0.357 = 0.007$$

Therefore, the naïve Bayesian classifier predicts **buys_computer = yes** for tuple X.