Classifiers

- Naïve Bayesian Classifiers
- K-Nearest Neighbour
- Decision Trees
- Linear Regression
- Logistic Regression
- Neural Networks
- Support Vector Machines

NAIVE BAYES CLASSIFIER

Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm. Naive Bayes classifiers gives high accuracy and more speed on large datasets.

Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features. For example, a loan applicant is desirable or not depending on his/her income, previous loan and transaction history, age, and location. Even if these features are interdependent, these features are still considered independently. This assumption simplifies computation, and that's why it is considered as naive. This assumption is called class conditional independence.

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

P(h): the probability of hypothesis h being true (regardless of the data). This is known as the prior probability of h.

P(D): the probability of the data (regardless of the hypothesis). This is known as the prior probability.

P(h|D): the probability of hypothesis h given the data D. This is known as posterior probability.

P(D|h): the probability of data d given that the hypothesis h was true. This is known as posterior probability.

Naive Bayes classifier calculates the probability of an event in the following steps:

Step 1: Calculate the prior probability for given class labels

Step 2: Find Likelihood probability with each attribute for each class

Step 3: Put these value in Bayes Formula and calculate posterior probability.

Step 4: See which class has a higher probability, given the input belongs to the higher probability class.

Naive Bayes Algorithm – Example.

Bayes Theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$P(A|B)$ = Conditional probability of A given B

$P(B|A)$ = Conditional probability of B given A

$P(A)$ = Probability of event A

$P(B)$ = Probability of event B.

Problem statement:

For the given tuple classify whether the Customer will buy the computer or not. (CL = YES OR NO).

X = A = {age = youth, income = medium, Student = Yes, C_R = fair}

The Prior Probability of each class (Yes, No) is

$P(\text{buys\_computer} = \text{Yes}) = 9/14 = 0.643$

$P(\text{buys\_computer} = \text{No}) = 5/14 = 0.357$

Then for the given tuple find Conditional probability:

$P(\text{age} = \text{youth} \mid \text{buys\_computer} = \text{yes}) = \frac{2}{9} = 0.222$

$P(\text{income} = \text{medium} \mid \text{buys\_computer} = \text{yes}) = \frac{3}{5} = 0.600$

$P(\text{Student} = \text{yes} \mid \text{buys\_computer} = \text{yes}) = 6/9 = 0.667$

$P(\text{C\_R} = \text{fair} \mid \text{buys\_computer} = \text{yes}) = 6/9 = 0.667$

$P(X \mid \text{buys\_computer} = \text{yes}) = 0.222 \times 0.444 \times 0.667 \times 0.667$

$= 0.044$

$P(X \mid \text{buys\_computer} = \text{yes})\ P(\text{buys\_computer} = \text{yes})$

$= 0.044 \times 0.643 = 0.028$

This is the Conditional probability of X (given tuple) to the outcome that buys_computer = yes.

$P(age = Youth \mid buys\text{-}computer = no) = \frac{3}{5} = 0.600$

$P(income = medium \mid buys\text{-}computer = no) = \frac{2}{5} = 0.400$

$P(student = yes \mid buys\text{-}computer = no) = \frac{1}{5} = 0.200$

$P(C\text{-}R = fair \mid buys\text{-}computer = no) = \frac{2}{5} = 0.400$

$P(X \mid buys\text{-}computer = No) = 0.600 \times 0.4 \times 0.2 \times 0.4$

$= 0.019.$

$P(X \mid buys\text{-}computer = No) P(buys\text{-}computer = No)$

$= 0.019 \times 0.357 = 0.007$

Finally,

$P(X \mid buys\text{-}computer = yes)\ P(buys\text{-}computer = yes) >$

$P(X \mid buys\text{-}computer = No)\ (P(buys\text{-}computer = No))$

(i.e) $0.028 > 0.007$,

Therefore, the Naive Bayesian classifier predicts buys-computer = Yes for the given tuple X.

Example 2. Naive Bayesian classifier:

Problem Statement: For the given dataset, using Naive Bayes's approach predict the outcome.

$X = \{Red, Domestic, SUV\}$

Colour	Type	Origin	Stolen
Red	Sports	Domestic	Yes
Red	Sports	Domestic	No
Red	Sports	Domestic	Yes
Yellow	Sports	Domestic	No
Yellow	Sports	Imported	Yes
Yellow	SUV	Imported	No
Yellow	SUV	Imported	Yes
Yellow	SUV	Domestic	No
Red	SUV	Imported	No
Red	Sports	Imported	Yes

$P(Red|Yes) = \frac{3}{5} = 0.6$

$P(Domestic|Yes) = \frac{2}{5} = 0.4$

$P(SUV|Yes) = \frac{1}{5} = 0.2$

$P(X|Yes) = 0.6 \times 0.4 \times 0.2 = 0.048$

$P(X|Yes)P(Yes) = 0.048 \times \frac{5}{10}$

$= 0.048 \times 0.5$

$= 0.024.$

$P(Red|No) = \frac{2}{5} = 0.4$

$P(Domestic|No) = \frac{3}{5} = 0.6$

$P(SUV|No) = \frac{2}{5} = 0.4$

$P(X|No) = 0.4 \times 0.6 \times 0.4$

$= 0.096$

$P(X|No)P(No) = 0.096 \times 5/10 = 0.096 \times 0.5$

$= 0.0480$

Finally

$P(X|No)P(No) > P(X|Yes)P(Yes)$

$0.0480 > 0.024$

Therefore the outcome for the given tuple is "No".

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING
ITA5007-Data Mining and Business Intelligence
MCA

Topic :

1. Naïve bayes examples

Description:

Naïve Bayes Formula:

P (A|B) =P (B|A). P(A) /P(B)

P(A|B) is posterior probability (The probability of A being true given that B is true)

P(B|A) is Likelihood (The Probability of B being true given that A is true)

P(A) is class prior probability (The probability of A being true)

P(B) is predictor prior probability (the probability of B being true)

Example 1:

In the given dataset, apply Naïve Bayes' algorithm and predict that if a fruit has the following properties then which type of fruit it is. Fruit (X)= { Yellow, Sweet,Long}

Fruit	Yellow	Sweet	Long	Total
Mango	350	450	0	650
Banana	400	300	350	400
Orange	50	100	50	150
Total	800	850	400	1200

If the fruit is of type Mango,

P(X|Mango)=P(Yellow|Mango).P(Sweet|Mango).P(Long|Mango)

P(Yellow|Mango)=P(Mango|Yellow).P(Yellow)/P(Mango)

=(350/800 * 800/1200) / (650/1200)

=0.53

P(Sweet|Mango)=P(Mango|Sweet).P(Sweet)/P(Mango)

=(450/850*850/1200) /(650/1200)

=0.69

P(Long|Mango) = P(Mango|Long).P(Long)/P(Mango)

= (0/400 *400/1200)/(650/1200)

=0

P(X|Mango)= P(Yellow|Mango).P(Sweet|Mango).P(Long|Mango) = 0.53*0.60*0=0

If the fruit is of type Banana,

P(X|Banana)=P(Yellow| Banana).P(Sweet| Banana).P(Long| Banana)

P(Yellow| Banana)=P(Banana |Yellow).P(Yellow)/P(Banana)

=1

P(Sweet| Banana)=P(Banana|Sweet).P(Sweet)/P(Banana)0.75

P(Long| Banana)=0.875

P(X| Banana)= P(Yellow| Banana).P(Sweet| Banana).P(Long| Banana) = 1*0.75*0.875 =0.65

If the fruit is of type Orange,

P(X| Orange)=P(Yellow| Orange).P(Sweet| Orange).P(Long| Orange)

P(Yellow| Orange)=P(Orange |Yellow).P(Yellow)/P(Orange)

=0.33

P(Sweet| Orange)=0.66

P(Long| Orange)=0.33

P(X| Orange)= P(Yellow| Orange).P(Sweet| Orange).P(Long| Orange) = 1*0.75*0.875 =0.072

The Probability that the fruit is a Banana is more than other values, so the fruit is Banana.

Example 2:

In the given dataset, apply Naïve Bayes Algorithm and predict the outcome, whether
Car ={Red, Domestic, SUV}
has been stolen or not.

Colour	Type	Origin	Stolen
Red	Sports	Domestic	Yes
Red	Sports	Domestic	No
Red	Sports	Domestic	Yes
Yellow	Sports	Domestic	No
Yellow	Sports	Imported	Yes
Yellow	SUV	Imported	No
Yellow	SUV	Imported	yes
Yellow	SUV	Domestic	No
Red	SUV	Imported	No
Red	Sports	Imported	Yes

X={Red, Domestic, SUV}
P(Red|Yes) =P(Yes|Red)* P(Red) / P(Yes)
= (3/5 *5/10) / (5/10) = 3/5
P(Domestic|Yes) =P(Yes|Domestic). P(Domestic) / P(Yes)
= (2/5* 5/10)/ (5/10)=2/5
P(SUV|Yes) =P(Yes|SUV). P(SUV) / P(Yes)
= (1/4 *4/10)/(5/10) =1/5

P(X/Yes)= P(Red|Yes)* P(Domestic|Yes)* P(SUV|Yes) =3/5*2/5*1/5=6/125=0.024

P(Red|No) =P(No|Red). P(Red) / P(No)
= (2/5 *5/10) / (5/10) = 2/5
P(Domestic| No) =P(No |Domestic). P(Domestic) / P(No)
= 3/5
P(SUV| No) =P(No |SUV). P(SUV) / P(No)
= 2/5
P(X/No)= P(Red|No)* P(Domestic|No)* P(SUV|No) =2/5*3/5*2/5=12/125=0.072

X={Red, Domestic, SUV} set of values according to Naïve Bayes calculation lies in **No** category that implies that the car is **not stolen**.

Questions:

S.No	Swim	Fly	Crawl	Class
1	Fast	No	No	Fish
2	Fast	No	Yes	Animal
3	Slow	No	No	Animal
4	Fast	No	No	Animal
5	No	Short	No	Bird
6	No	Short	No	Bird

7	No	Rarely	No	Animal
8	Slow	No	Yes	Animal
9	Slow	No	no	Fish
10	Slow	No	Yes	Fish
11	No	Long	No	Bird
12	Fast	No	No	Bird

Let X =(Slow, Rarely, No), Which class label is appropriate for X ?
Solution : class label is animal

binary splits and that is based on the notion of purity of partitions, such as the gini index. BOAT uses a lower bound on the attribute selection measure in order to detect if this "very good" tree, T', is different from the "real" tree, T, that would have been generated using the entire data. It refines T' in order to arrive at T.

BOAT usually requires only two scans of D. This is quite an improvement, even in comparison to traditional decision tree algorithms (such as the basic algorithm in Figure 6.3), which require one scan per level of the tree! BOAT was found to be two to three times faster than RainForest, while constructing exactly the same tree. An additional advantage of BOAT is that it can be used for incremental updates. That is, BOAT can take new insertions and deletions for the training data and update the decision tree to reflect these changes, without having to reconstruct the tree from scratch.

6.4 Bayesian Classification

"What are Bayesian classifiers?" Bayesian classifiers are statistical classifiers. They can predict class membership probabilities, such as the probability that a given tuple belongs to a particular class.

Bayesian classification is based on Bayes' theorem, described below. Studies comparing classification algorithms have found a simple Bayesian classifier known as the *naive Bayesian classifier* to be comparable in performance with decision tree and selected neural network classifiers. 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*. It is made to simplify the computations involved and, in this sense, is considered "naïve." *Bayesian belief networks* are graphical models, which unlike naïve Bayesian classifiers, allow the representation of dependencies among subsets of attributes. Bayesian belief networks can also be used for classification.

Section 6.4.1 reviews basic probability notation and Bayes' theorem. In Section 6.4.2 you will learn how to do naïve Bayesian classification. Bayesian belief networks are described in Section 6.4.3.

6.4.1 Bayes' Theorem

Bayes' theorem is named after Thomas Bayes, a nonconformist English clergyman who did early work in probability and decision theory during the 18th century. Let $\boldsymbol{X}$ be a data tuple. In Bayesian terms, $\boldsymbol{X}$ is considered "evidence." As usual, it is described by measurements made on a set of n attributes. Let H be some hypothesis, such as that the data tuple $\boldsymbol{X}$ belongs to a specified class C. For classification problems, we want to determine $P(H|\boldsymbol{X})$, the probability that the hypothesis H holds given the "evidence" or observed data tuple $\boldsymbol{X}$. In other words, we are looking for the probability that tuple $\boldsymbol{X}$ belongs to class C, given that we know the attribute description of $\boldsymbol{X}$.

$P(H|\boldsymbol{X})$ is the **posterior probability**, or *a posteriori probability*, of H conditioned on $\boldsymbol{X}$. For example, suppose our world of data tuples is confined to customers described by

the attributes *age* and *income*, respectively, and that $\boldsymbol{X}$ 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 $P(H|\boldsymbol{X})$ reflects the probability that customer $\boldsymbol{X}$ will buy a computer given that we know the customer's age and income.

In contrast, $P(H)$ is the **prior probability**, or *a priori probability,* of H. For our example, this is the probability that any given customer will buy a computer, regardless of age, income, or any other information, for that matter. The posterior probability, $P(H|\boldsymbol{X})$, is based on more information (e.g., customer information) than the prior probability, $P(H)$, which is independent of $\boldsymbol{X}$.

Similarly, $P(\boldsymbol{X}|H)$ is the posterior probability of $\boldsymbol{X}$ conditioned on H. That is, it is the probability that a customer, $\boldsymbol{X}$, is 35 years old and earns $40,000, given that we know the customer will buy a computer.

$P(\boldsymbol{X})$ is the prior probability of $\boldsymbol{X}$. Using our example, it is the probability that a person from our set of customers is 35 years old and earns $40,000.

"*How are these probabilities estimated?*" $P(H)$, $P(\boldsymbol{X}|H)$, and $P(\boldsymbol{X})$ may be estimated from the given data, as we shall see below. **Bayes' theorem** is useful in that it provides a way of calculating the posterior probability, $P(H|\boldsymbol{X})$, from $P(H)$, $P(\boldsymbol{X}|H)$, and $P(\boldsymbol{X})$. Bayes' theorem is

$$P(H|\boldsymbol{X}) = \frac{P(\boldsymbol{X}|H)P(H)}{P(\boldsymbol{X})}. \tag{6.10}$$

Now that we've got that out of the way, in the next section, we will look at how Bayes' theorem is used in the naive Bayesian classifier.

6.4.2 Naïve Bayesian Classification

The **naïve Bayesian** classifier, or **simple Bayesian** classifier, works as follows:

1. Let D be a training set of tuples and their associated class labels. As usual, each tuple is represented by an n-dimensional attribute vector, $\boldsymbol{X} = (x_1, x_2, \ldots, x_n)$, depicting n measurements made on the tuple from n attributes, respectively, $A_1, A_2, \ldots, A_n$.

2. Suppose that there are m classes, $C_1, C_2, \ldots, C_m$. Given a tuple, $\boldsymbol{X}$, the classifier will predict that $\boldsymbol{X}$ belongs to the class having the highest posterior probability, conditioned on $\boldsymbol{X}$. That is, the naïve Bayesian classifier predicts that tuple $\boldsymbol{X}$ belongs to the class C_i if and only if

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

   Thus we maximize $P(C_i|\boldsymbol{X})$. The class C_i for which $P(C_i|\boldsymbol{X})$ is maximized is called the *maximum posteriori hypothesis*. By Bayes' theorem (Equation (6.10)),

$$P(C_i|\boldsymbol{X}) = \frac{P(\boldsymbol{X}|C_i)P(C_i)}{P(\boldsymbol{X})}. \tag{6.11}$$

3. As $P(\boldsymbol{X})$ is constant for all classes, only $P(\boldsymbol{X}|C_i)P(C_i)$ need be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are

equally likely, that is, $P(C_1) = P(C_2) = \cdots = P(C_m)$, and we would therefore maximize $P(\boldsymbol{X}|C_i)$. Otherwise, we maximize $P(\boldsymbol{X}|C_i)P(C_i)$. Note that the class prior probabilities may be estimated by $P(C_i) = |C_{i,D}|/|D|$, where $|C_{i,D}|$ is the number of training tuples of class C_i in D.

4. Given data sets with many attributes, it would be extremely computationally expensive to compute $P(\boldsymbol{X}|C_i)$. In order to reduce computation in evaluating $P(\boldsymbol{X}|C_i)$, the naive assumption of **class conditional independence** is made. This presumes that the values of the attributes are conditionally independent of one another, given the class label of the tuple (i.e., that there are no dependence relationships among the attributes). Thus,

$$\begin{aligned} P(\boldsymbol{X}|C_i) &= \prod_{k=1}^{n} P(x_k|C_i) \\ &= P(x_1|C_i) \times P(x_2|C_i) \times \cdots \times P(x_n|C_i). \end{aligned} \tag{6.12}$$

We can easily estimate the probabilities $P(x_1|C_i), P(x_2|C_i), \ldots, P(x_n|C_i)$ from the training tuples. Recall that here x_k refers to the value of attribute A_k for tuple $\boldsymbol{X}$. For each attribute, we look at whether the attribute is categorical or continuous-valued. For instance, to compute $P(\boldsymbol{X}|C_i)$, we consider the following:

(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, then we need to do a bit more work, but the calculation is pretty straightforward. A continuous-valued attribute is typically assumed to have a Gaussian distribution with a mean μ and standard deviation σ, defined by

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{6.13}$$

so that

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}). \tag{6.14}$$

These equations may appear daunting, but hold on! We need to compute μ_{C_i} and σ_{C_i}, which are the mean (i.e., average) and standard deviation, respectively, of the values of attribute A_k for training tuples of class C_i. We then plug these two quantities into Equation (6.13), together with x_k, in order to estimate $P(x_k|C_i)$. For example, let $\boldsymbol{X} = (35, \$40{,}000)$, where A_1 and A_2 are the attributes *age* and *income*, respectively. Let the class label attribute be *buys_computer*. The associated class label for $\boldsymbol{X}$ is *yes* (i.e., *buys_computer* = *yes*). Let's suppose that *age* has not been discretized and therefore exists as a continuous-valued attribute. Suppose that from the training set, we find that customers in D who buy a computer are 38 ± 12 years of age. In other words, for attribute *age* and this class, we have $\mu = 38$ years and $\sigma = 12$. We can plug these quantities, along with $x_1 = 35$ for our tuple $\boldsymbol{X}$ into Equation (6.13) in order to estimate *P(age = 35|buys_computer = yes)*. For a quick review of mean and standard deviation calculations, please see Section 2.2.

5. In order to predict the class label of $\boldsymbol{X}$, $P(\boldsymbol{X}|C_i)P(C_i)$ is evaluated for each class C_i. The classifier predicts that the class label of tuple $\boldsymbol{X}$ is the class C_i if and only if

$$P(\boldsymbol{X}|C_i)P(C_i) > P(\boldsymbol{X}|C_j)P(C_j) \quad \text{for } 1 \leq j \leq m, j \neq i. \tag{6.15}$$

In other words, the predicted class label is the class C_i for which $P(\boldsymbol{X}|C_i)P(C_i)$ is the maximum.

"*How effective are Bayesian classifiers?*" Various empirical studies of this classifier in comparison to decision tree and neural network classifiers have found it to be comparable in some domains. In theory, Bayesian classifiers have the minimum error rate in comparison to all other classifiers. However, in practice this is not always the case, owing to inaccuracies in the assumptions made for its use, such as class conditional independence, and the lack of available probability data.

Bayesian classifiers are also useful in that they provide a theoretical justification for other classifiers that do not explicitly use Bayes' theorem. For example, 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.

Example 6.4 **Predicting a class label using naïve Bayesian classification.** We wish to predict the class label of a tuple using naïve Bayesian classification, given the same training data as in Example 6.3 for decision tree induction. The training data are in Table 6.1. The data tuples are described by the attributes *age*, *income*, *student*, and *credit_rating*. The class label attribute, *buys_computer*, has two distinct values (namely, {*yes, no*}). Let C_1 correspond to the class *buys_computer* = *yes* and C_2 correspond to *buys_computer* = *no*. The tuple we wish to classify is

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

We need to maximize $P(\boldsymbol{X}|C_i)P(C_i)$, for $i = 1, 2$. $P(C_i)$, the prior probability of each class, can be computed based on the training tuples:

$$P(buys\_computer = yes) = 9/14 = 0.643$$
$$P(buys\_computer = no) = 5/14 = 0.357$$

To compute $P\boldsymbol{X}|C_i)$, for $i = 1, 2$, we compute the following conditional probabilities:

$$P(age = youth \mid buys\_computer = yes) = 2/9 = 0.222$$
$$P(age = youth \mid buys\_computer = no) = 3/5 = 0.600$$
$$P(income = medium \mid buys\_computer = yes) = 4/9 = 0.444$$
$$P(income = medium \mid buys\_computer = no) = 2/5 = 0.400$$
$$P(student = yes \mid buys\_computer = yes) = 6/9 = 0.667$$
$$P(student = yes \mid buys\_computer = no) = 1/5 = 0.200$$
$$P(credit\_rating = fair \mid buys\_computer = yes) = 6/9 = 0.667$$
$$P(credit\_rating = fair \mid buys\_computer = no) = 2/5 = 0.400$$

Using the above probabilities, we obtain

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

Similarly,

$$P(\boldsymbol{X}|buys\_computer = no) = 0.600 \times 0.400 \times 0.200 \times 0.400 = 0.019.$$

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

$$P(\boldsymbol{X}|buys\_computer = yes)P(buys\_computer = yes) = 0.044 \times 0.643 = 0.028$$

$$P(\boldsymbol{X}|buys\_computer = no)P(buys\_computer = no) = 0.019 \times 0.357 = 0.007$$

Therefore, the naïve Bayesian classifier predicts *buys_computer = yes* for tuple $\boldsymbol{X}$. ■

"*What if I encounter probability values of zero?*" Recall that in Equation (6.12), we estimate $P(\boldsymbol{X}|C_i)$ as the product of the probabilities $P(x_1|C_i), P(x_2|C_i), \ldots, P(x_n|C_i)$, based on the assumption of class conditional independence. These probabilities can be estimated from the training tuples (step 4). We need to compute $P(\boldsymbol{X}|C_i)$ for *each* class ($i = 1, 2, \ldots, m$) in order to find the class C_i for which $P(\boldsymbol{X}|C_i)P(C_i)$ is the maximum (step 5). Let's consider this calculation. For each attribute-value pair (i.e., $A_k = x_k$, for $k = 1, 2, \ldots, n$) in tuple $\boldsymbol{X}$, we need to count the number of tuples having that attribute-value pair, per class (i.e., per C_i, for $i = 1, \ldots, m$). In Example 6.4, we have two classes ($m = 2$), namely *buys_computer = yes* and *buys_computer = no*. Therefore, for the attribute-value pair *student = yes* of $\boldsymbol{X}$, say, we need two counts—the number of customers who are students and for which *buys_computer = yes* (which contributes to $P(\boldsymbol{X}|buys\_computer = yes)$) and the number of customers who are students and for which *buys_computer = no* (which contributes to $P(\boldsymbol{X}|buys\_computer = no)$). But what if, say, there are no training tuples representing students for the class *buys_computer = no*, resulting in $P(student = yes|buys\_computer = no) = 0$? In other words, what happens if we should end up with a probability value of zero for some $P(x_k|C_i)$? Plugging this zero value into Equation (6.12) would return a zero probability for $P(\boldsymbol{X}|C_i)$, even though, without the zero probability, we may have ended up with a high probability, suggesting that $\boldsymbol{X}$ belonged to class C_i! A zero probability cancels the effects of all of the other (posteriori) probabilities (on C_i) involved in the product.

There is a simple trick to avoid this problem. We can assume that our training database, D, is so large that adding one to each count that we need would only make a negligible difference in the estimated probability value, yet would conveniently avoid the case of probability values of zero. This technique for probability estimation is known as the **Laplacian correction** or **Laplace estimator**, named after Pierre Laplace, a French mathematician who lived from 1749 to 1827. If we have, say, q counts to which we each add one, then we must remember to add q to the corresponding denominator used in the probability calculation. We illustrate this technique in the following example.