CHAPTER 3

Classification Learning

3.1 INTRODUCTION

Classification learning is a learning scheme for categorizing unseen examples into predefined classes based on a set of training examples. The learning algorithm generates a set of classification rules from a complete set of independent examples of instances and their corresponding categories, and then the generated rules are used to predict the classes or categories of novel instances.

The purpose of classification learning is to predict the classes of instances, in contrast with other methods. Association learning predicts not only classes but also attributes that are used in inducing the classes. In clustering, the classes are not predefined but are unknown at the point of learning, and the learning task is to define and identify the classes in the database. In numeric prediction or regression, classes are composed not of discrete categories but of continuous numeric values, but otherwise regression learning uses techniques very similar to classification learning and is sometimes considered a subtype of classification learning.

A typical application of classification learning requires the following characteristics: (1) predefined classes, (2) discrete data domains (except in regression learning), (3) a sufficient amount of training data, with at least as many examples as the number of classes, and (4) attribute values that are flat rather than structured data such that the values are fixed and each attribute has either a discrete or a numeric value.

Among the many algorithms used for classification learning, three major approaches to defining classification rules are found. The first approach uses a top-down, "divide and conquer" technique to induce knowledge rules by organizing all instances of the dataset into a "decision tree" based on a series

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

of outcomes of tests peformed on each attribute. The "divide and conquer" approach selects one attribute and partitions the dataset into subsets based on the outcome of a test, then recursively applies the process to the partitions until pure subsets are reached (all members are classified to only one class). This process of tree creation is in fact the process of heuristically searching for all possible classification rules. The classification rules can be directly generated from the tree by traversing through the paths from the root to each leaf. Several sophisticated systems have been developed during the last two decades. The most notable ones are **Cart** [Breiman 1984], **ID3** [Quinlan 1979, 1983, 1986], and its successors **C4.5** [Quinlan 1993] and **C5.0** [Quinlan 1997].

The second approach uses a top-down "separate and conquer" or "covering" technique that takes each class in turn and directly induces a set of rules, each covering as many instances of the class as possible (and excluding as few instances of other classes as possible) without erecting the tree first. After a rule is induced, the covered instances are excluded or separated from further induction. The "separate and conquer" approach is concerned with only one class at a time, and performs all the tests to quickly purify the subset, whereas both subsets may not be pure in the "divide and conquer" approach. Since there are some limitations in the representation of the classified rules, this process is less accurate than the "divide and conquer" approach, but faster because it does not follow the heuristic tree-searching procedure. Systems such as **Prism** [Cendrowska 1987], **Induct** [Gaines 1995], **IREP** [Furnkranz 1994], and **RIPPER** [Cohen 1995] use this approach.

The third approach, the "partial decision-tree" approach [Frank 1998], is a combination of the "divide and conquer" and "separate and conquer" approaches. It produces rules by inducing partial decision trees and separating the covered instances from further induction.

The goals of these approaches are to accurately and efficiently induce classification knowledge from a dataset. In this chapter, we will compare the three approaches mentioned above by introducing algorithms for each approach, with examples of how each algorithm is applied, and examine the advantages and disadvantages of each one.

Before we delve into the details of each algorithm, the representations of the knowledge extracted by the data-mining procedures are introduced. Each algorithm will use one of these forms to represent learned concepts or intermediate results.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

3.2 KNOWLEDGE REPRESENTATION

Two major kinds of knowledge representation are used in classification learning: the decision tree and the classification rule.

3.2.1 Classification Rules

Classification rules are the simplest representation of classification knowledge. Each classification rule is composed of a condition, which includes one or more tests, and a conclusion, which associates an instance with the class to which it belongs. Knowledge rules are expressed in the form "if the attribute *X* is *xxx* and the attribute *Y* is *yyy*, etc., then the instance *Z* belongs to class *zzz*." To each class there corresponds a set of relevant attribute values. After the complete set of rules generated from a dataset is sorted by the classes, they constitute a rule list used to represent the complete classification information over the dataset.

Classification rules are simple, straightforward, and easy to understand. The limitation of classification rules is that they cannot resolve conflicting information. If more than one rule have the same condition but specify different classes for the instance, additional attributes are required to resolve the contradiction and form a new knowledge rule.

3.2.2 Decision Trees

The decision trees shown in Figure 3.1 represent the integrated classification knowledge of a system. Each internal node represents a test over one or more attributes and corresponds to the condition of a classification rule. Each branch represents an outcome of the test. The leaf nodes indicate the classes corresponding to the conclusions of the classification rules. To identify an instance, move from the root down to the leaves according to the test results of successive internal nodes. The leaf arrived at shows the class if belongs to.

Classification rules and decision trees represent the same knowledge and are therefore interchangeable in most instances. We can directly generate rules from a simple tree. However, for a complex tree, the process of conversion is not that easy. Although classification rules are simpler and easier to understand than decision trees, there are potential problems with conflicting information. Moreover, decision trees give an integrated, complete view of classification information for the domain. On the other hand, classification

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

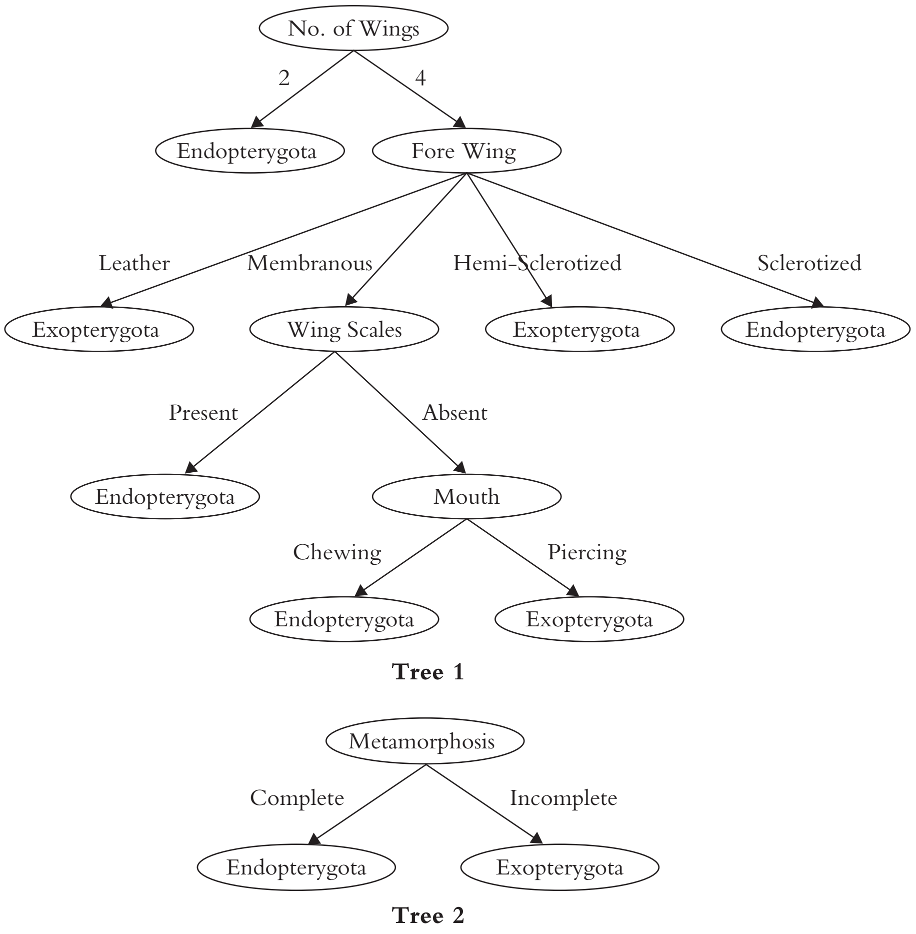

FIGURE 3.1 Decision trees derived from data in Table 3.1. Tree 1 is a random tree and tree 2 is a tree induced from the ID3 algorithm.

rules are the ultimate representation of the knowledge used in all algorithms for classification learning, whereas decision trees are only used in the "divide and conquer" and "partial decision tree" algorithms as an intermediate tool.

3.3 SEPARATE-AND-CONQUER APPROACH

The "separate and conquer" or "covering" technique takes each class and creates rules that cover as many instances of this class as possible while excluding as few instances of other classes as possible. This approach examines only one class at a time. For each class, it builds a rule by selecting

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

and adding tests to the rule until all the subset of instances covered by the rule are "pure" (e.g., all members belong to only one class). The covered subset of instances is then excluded from further processing. The rule-generation process continues until no more unclassified instances are left in the dataset. The advantage of this approach is time efficiency as a result of the following two characteristics: first, it creates knowledge rules directly without inducing an intermediate decision tree; secondly, it immediately excludes instances covered by a newly created rule from further induction.

The criteria used for test selection and the standard of "purity" vary among different algorithms using this approach. For example, in the Prism algorithm, it is assumed that the accuracy of a rule can be measured by the proportion of correct predictions it makes over the entire set of instances covered by the rule. A candidate rule is refined by selecting and adding tests that maximize this quantity until the rule reaches 100% purity or there are no more tests. In the Induct algorithm, the test selection is based on information gain, and this algorithm is robust against noisy data.

■ 3.3.1 Prism

This simple and straightforward covering algorithm works by first picking a class from the dataset for which to create a new rule having the class as its conclusion, and selectively adding tests to the condition of the rule, striving for maximum number of instances covered and 100% accuracy. The accuracy of a test is measured by the ratio of the number of positive instances p to the total number of instances covered by the rule t: p/t. The positive instances covered by the new rule then are removed from the dataset for further rule generation. Then, negative instances should remain in the dataset to await a later iteration of the process. This process continues until no more instances remain to be covered.

Let's consider a simple database in Table 3.1 as an example and show how the algorithm is applied to generate classification rules. This database contains information on morphological characteristics and suborders of ten insects. The purpose of this learning task is to create rules that associate insects to the right suborders according to their characteristics.

Let's randomly pick the suborder *Endopterygota* as the class of the first rule: "If ? , then the suborder is *Endopterygota*." Since the new rule initially has no tests in its condition part, it is not useful yet. It covers six instances. The

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

TABLE 3.1 Morphological characteristics and suborders of some insects

Name	No. of Wings	Forewing	Wing Scales	Mouth	Meta-morphosis	Hind Leg	Abdomen Needle	Suborder
Fly	2	Membrane	Absent	Sponging	Complete	Walking	Absent	Endopterygota
Wasp	4	Membrane	Absent	Chewing	Complete	Walking	Present	Endopterygota
Bee	4	Membrane	Absent	Chewing	Complete	Walking	Present	Endopterygota
Beetle	4	Sclerotized	Absent	Chewing	Complete	Walking	Absent	Endopterygota
Butter-fly	4	Membrane	Present	Siphoning	Complete	Walking	Absent	Endopterygota
Moth	4	Membrane	Present	Siphoning	Complete	Walking	Absent	Endopterygota
True Bug	4	Hemi-Sclerotized	Absent	Piercing	Incomplete	Walking	Absent	Exopterygota
Aphid	4	Membrane	Absent	Piercing	Incomplete	Walking	Absent	Exopterygota
Grass hopper	4	Leather	Absent	Chewing	Incomplete	Jumping	Absent	Exopterygota
Cock-roach	4	Leather	Absent	Chewing	Incomplete	Walking	Absent	Exopterygota

question is which *attribute* = *value* tests need to be added to the rule. The decision is made based on the accuracy of each possible test. For example, the accuracy of the test *number of wings* = 4 is measured by the ratio of the number of correct instances (*p*)—the number of instances satisfying the test—to the total number of instances (of any class) satisfying the test (*t*). We select the test with the highest accuracy, that is, *p*/*t* value. From Table 3.1, a list of the *p*/*t* values for each test is shown below:

Characteristics	***p/t***
No. of wings 2	*1/1*
4	*5/9*
Forewing membranous	*5/6*
sclerotized	*1/1*
hemi-sclerotized	*0/1*
leather	*0/2*

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

Wing with scales present	*2/2*
absent	*4/8*
Mouth sponging	*1/1*
chewing	*3/5*
siphoning	*2/2*
piercing	*0/2*
Hind leg walking	*6/9*
jumping	*0/1*
Abdomen needle present	*2/2*
absent	*4/8*
Metamorphosis complete	*6/6*
incomplete	*0/4*

From this list, we see that seven candidates are tied at a *p/t* value of 100%: *no. of wings* 2, *wing sclerotized*, *wing with scales present*, *mouth sponging*, *mouth siphoning*, *abdomen needle present*, and *metamorphosis complete*. However, because *metamorphosis complete* covers the most instances, we choose it as the first test to be added to the rule, yielding the following rule:

"If metamorphosis is complete, then the insect is in the suborder Endopterygota."

Do we need to add more terms to the new rule? Let's test the purity of the subset covered by the rule. In our case, the purity of the above rule is 100%. Therefore, no more terms need to be added. After the rule is formed, all six instances covered by the rule are then removed from the dataset for further processing. There are now four instances left in the dataset. Using the same strategy, we create the following additional rule:

"If metamorphosis is incomplete, the insect is in the suborder Exopterygota."

This second rule covers all four instances left in the dataset. So, no more rules are needed.

As this example shows, Prism works in three steps to build a classification rule:

1. First, it identifies the attribute-value pairs with the highest *p/t* ratio as candidate tests to add to the condition term of the rule.
2. If there is a tie among several tests, the test that covers the most positive instances will be selected.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

3. After the accuracy of the rule reaches 100% or there are no more `attributes` left, the process of adding additional tests to the condition term of the rule stops, and the positive instances covered by the rule are removed from the dataset.

The Prism algorithm is summarized in Figure 3.2 below:

FIGURE 3.2 The Prism rule-generation algorithm

```
For each class C
  Initialize E to the instance set
  While E contains instances in class C
    Create a rule R with an empty condition that predicts class C
    Until R is perfect (or there are no more attributes to use) do
      For each attribute A not mentioned in R, and each value V,
        Consider all possible tests A = V for the condition of R
    Select A and V to maximize the accuracy p/t
      (break ties by choosing the condition with
        the largest p/t ratio)
    Add A = V to R
      Remove the instances covered by R from E
```

In real-world applications, data is not always as sound and clean as in our example, especially in large databases. In these circumstances, the 100% accuracy requirement of the rule cannot be satisfied. The Prism algorithm cannot handle this situation and potentially useful information might be ignored. In some applications, a certain level of error is allowed for predictions.

To overcome the over-fitting difficulty in the Prism algorithm, Gaines and Compton [Gaines 1995] proposed a new system called Induct, which uses probability to measure the degree of "goodness" of rules.

3.3.2 Induct

The Induct algorithm inherited the idea of "separate and conquer" from Prism. Unlike Prism, the Induct algorithm does not use 100% accuracy as the measure of "goodness" for rules. Instead, it uses probability to measure the worth of a rule. Moreover, an additional post-pruning process is conducted after "perfect" rules are formed to trim off the over-fitted conditions according to the probability measures.

The key difference in Induct is the use of a probability measure to measure the "goodness" of a rule instead of the accuracy p/t, which is used in the Prism algorithm. This is the probability that a randomly chosen rule will have

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

accuracy the same as or higher than a given rule. If the odds are high for a randomly created rule to have better accuracy than the rule being tested, then the rule being tested is probably not a reliable one.

In general, the probability measurement agrees with the accuracy measurement of Prism: a rule with higher accuracy usually has a lower probability measure. The difference is that the probability is sensitive to not only the accuracy but also the coverage of the rule. A rule with high accuracy may still have a high probability measure if its coverage is very narrow. For example, consider the perfect rule "If wing is membranous and wing scales are present, then the suborder is *Endopterygota*." Given the data in Table 3.1, the accuracy of this rule is 2/2 or 100%, while the overall coverage of the dataset is 6/10. This rule covers two positive cases. The probability of a randomly chosen rule being at least as good as this rule is 0.33 (the details of the probability measure calculation will be explained below). When we loosen the condition of this rule by removing the last test, "wing scales are present," the rule's coverage increases to five positive instances out of six (accuracy of 83%). Although the accuracy dropped from 100% to 83%, the probability measure further drops to 0.071, since the new rule covers six instances instead of two.

To calculate the probability of a random rule having at least the same accuracy as the rule being tested, we first derive the probability of a random rule having an exact number of positive instances, i. This rule has the accuracy i/t, where t is the total number of instances the rule covers. For a better rule or a more accurate rule to be generated, the number of positive instances predicted must be greater than i. If the rule being tested covers t instances and i instances are positive, there exist $t - i$ possible "better" rules. The sum of the probabilities of all such "better" rules and of the rule with exactly i positive instances is the probability of a randomly generated rule having accuracy of at least i/t.

The probability of a rule that covers t instances and has exactly p positive instances in class c is expressed as follows:

Given Pr[*of t instances selected at random, exactly p instances are in class c*], *then the probability of a randomly generated rule having the same or better accuracy than i/t can be expressed as*

$$M(R) = \sum_{n=i}^{t} \text{Pr[of } t \text{ instances selected at random, exactly } n \text{ instances are in class } c]$$

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

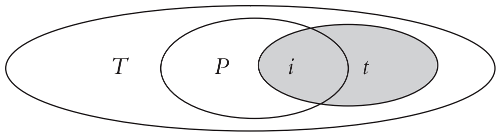

FIGURE 3.3 Relationship among various factors used in probability measurement

T :	total number of instances in the dataset
t :	total number of instances covered by the condition of rule R
P :	total number of positive instances belonging to class c in the dataset
i :	total number of positive instances covered by the condition of rule R

First, let's see how to derive the probability of a randomly chosen rule having exactly i positive instances when it covers t instances. As shown in Figure 3.3, the dataset has T total instances, of which P instances belong to class *c*. A randomly generated rule covers t instances, of which i instances belong to class *c*. We are selecting i instances of class *c* from the positive subset of the dataset with P instances and, at the same time, choosing $t - i$ instances from the rest of the instance set which contains $T - P$ instances. Therefore, the probability of a randomly selected rule R with i positive instances of class *c* can be expressed as follows:

$$\text{Pr[of } t \text{ instances selected at random, exactly } i \text{ instances are in the class } c] = \frac{\binom{P}{i}\binom{T-P}{t-i}}{\binom{T}{t}}$$

The probability distribution is hypergeometric. We assume that the selection is done "without replacement." For example, suppose that we derived a rule "If *tear production* rate is **not** *reduced* and *age* is *presbyopic* and *prescription* is *hypermetrope*, then the *recommended contact lens* is *none*" from the contact lens dataset in Table 3.2. The probability of the rule occurring randomly is calculated as follows with $T = 24$, $P = 15$, $t = 2$, and $i = 1$.

$$\text{Pr[of 2 instances selected at random, exactly 1 is in class } none] = \frac{\binom{15}{1}\binom{9}{1}}{\binom{24}{2}} = 0.49$$

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

From the equation above, we see that the probability of a random rule with accuracy 1/2 is 0.49. Thus, the probability of a randomly chosen rule that will do as well as or better than the above rule R is calculated as the sum of the probabilities of the random rules with one positive instance (accuracy 1/2) and two positive instances (accuracy 2/2), with $T = 24$, $P = 15$, $t = 2$, and $i = 1, 2$.

$$M(R) = \sum_{i}^{t} \text{Pr[of two instances selected at random, exactly } n \text{ instances are in class } none] = \sum_{i=1}^{2} \frac{\binom{15}{i}\binom{9}{2-i}}{\binom{24}{2}} = 0.49 + 0.38 = 0.87$$

The Induct algorithm is a two-stage process. The first stage is the rule-formation stage, which is adopted from the Prism algorithm. The second stage is the post-pruning stage. In the second stage, Induct trims off the tests of each rule one by one, in reverse order, until the probability measure increases. After this post-pruning process, the rules will no longer be over-fitting.

The procedure for rule generation and trimming, which is used to generate a good rule set, is summarized in Figure 3.4:

FIGURE 3.4 Rule generation algorithm with probability measurement

```
Initialize T to be the instance set;
repeat
  do
    for each class C for which T contains an instance
      (1) use the basic covering algorithm to create a perfect rule
          for class C;
      (2) calculate the probability measure M(R) for the rule;
      (3) calculate the probability measure M(R-) for the rule with
          the final condition removed;
      (4) if M(R-) < M(R), remove the final condition from the rule;
      (5) repeat steps (3) and (4) until M(R-) > M(R);
      (6) add the rule to the rule set;
      (7) remove all the instances covered by the rule from T.
    done
until T is empty;
```

Let's take Cendrowska's contact lens example to illustrate Induct's performance on a test dataset [Cendrowska 1987]. This dataset is shown in

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

TABLE 3.2 The contact lens data

Age	Spectacle Prescription	Astigmatism	Tear Production Rate	Recommended Contact Lens
Young	Myope	No	Reduced	None
Young	Myope	No	Normal	Soft
Young	Myope	Yes	Reduced	None
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	No	Reduced	None
Young	Hypermetrope	No	Normal	Soft
Young	Hypermetrope	Yes	Reduced	None
Young	Hypermetrope	Yes	Normal	Hard
Pre-presbyopic	Myope	No	Reduced	None
Pre-presbyopic	Myope	No	Normal	Soft
Pre-presbyopic	Myope	Yes	Reduced	None
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	No	Reduced	None
Pre-presbyopic	Hypermetrope	No	Normal	Soft
Pre-presbyopic	Hypermetrope	Yes	Reduced	None
Pre-presbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	No	Reduced	None
Presbyopic	Myope	No	Normal	None
Presbyopic	Myope	Yes	Reduced	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	No	Reduced	None
Presbyopic	Hypermetrope	No	Normal	Soft
Presbyopic	Hypermetrope	Yes	Reduced	None
Presbyopic	Hypermetrope	Yes	Normal	None

Table 3.2. It contains the relationships between the characteristics of patients and their appropriate contact lens types.

The process of applying the Induct algorithm to this dataset starts with selecting a class. Since the class "*recommended contact lens = none*" covers 15 instances, which is more than the number covered by any other class, we select

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

it as the first class for which to generate new rules. The list of accuracy (*p*/*t*) values of the tests for the class is given below:

Tests	***p*/*t***
Age is *young*	4/8
Age is *pre-presbyopic*	5/8
Age is *presbyopic*	6/8
Spectacle prescription is *myope*	7/12
Spectacle prescription is *hypermetrope*	8/12
Astigmatism is *no*	7/12
Astigmatism is *yes*	8/12
Tear production rate is *reduced*	12/12
Tear production rate is *normal*	3/12

From the list, the test with highest accuracy value (*p*/*t*) is "*tear production rate reduced*," which is 12/12. Therefore, the initial condition term of the rule should be "tear production rate reduced." Since the accuracy (*p*/*t*) value is 100%, we do not need to add more tests to the rule. Therefore, the first rule generated is

"If tear production rate is reduced, then the recommended contact lens is none."

After the instances covered by the above rule are excluded, the dataset is reduced as shown in Table 3.3. In the reduced dataset, there are still three instances of the class "*none*." More rules need to be created for the class *none*. A new list of accuracy (*p*/*t*) values is calculated to continue the process.

Tests	***p*/*t***
Age is *young*	0/4
Age is *pre-presbyopic*	1/4
Age is *presbyopic*	2/4
Spectacle prescription is *myope*	1/6
Spectacle prescription is *hypermetrope*	2/6
Astigmatism is *no*	1/6
Astigmatism is *yes*	2/6
Tear production rate is *normal*	3/12

From the list, "*age is presbyopic*" has the highest *p*/*t* value. Therefore, we choose "*age is presbyopic*" as the initial condition term of the second new rule. Since the *p*/*t* value is not 100%, we continue to add more terms to the rule.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

TABLE 3.3 Contact lens dataset after the first rule is created

Age	Spectacle Prescription	Astigmatism	Tear Production Rate	Recommended Contact Lens
Young	Myope	No	Normal	Soft
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	No	Normal	Soft
Young	Hypermetrope	Yes	Normal	Hard
Pre-presbyopic	Myope	No	Normal	Soft
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	No	Normal	Soft
Pre-presbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	No	Normal	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	No	Normal	Soft
Presbyopic	Hypermetrope	Yes	Normal	None

The two tests "*prescription is hypermetrope*" and "*astigmatism is yes*" have the same *p/t* values and the same coverage. We can randomly choose "*prescription is hypermetrope*" to add to the condition of the rule. Now the *p/t* value for the combined terms is 1/2. The new rule is still not pure. The next candidate test would be "*astigmatism is yes*." The *p/t* value after combining the three conditions is 1/1. Since the accuracy is 100%, no other condition is needed. The new rule covers only one instance. The second rule is the following:

> *"If age is presbyopic and prescription is hypermetrope and astigmatism is yes, then the recommended contact lens is none."*

Next, we drop the instance covered by this rule from the reduced dataset. It is important to point out that when more condition terms are added to a rule, at maximum only one test value can be used for the same attribute. For example, if the term "*age is presbyopic*" has already been selected, the term "*age is young*" cannot later be added to the rule, even if its *p/t* value is higher than any other test's, since the effect of combining the two test terms for the same attribute broadens rather than restricts the scope of the rule.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

TABLE 3.4 Contact lens dataset after all the instances associated with class "*none*" are removed

Age	Spectacle Prescription	Astigmatism	Tear Production Rate	Recommended Contact Lens
Young	Myope	No	Normal	Soft
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	No	Normal	Soft
Young	Hypermetrope	Yes	Normal	Hard
Pre-presbyopic	Myope	No	Normal	Soft
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	No	Normal	Soft
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	No	Normal	Soft

By repeating this procedure as described above, two more rules are generated for the class "*recommended contact lens is none*":

1. *"If age is presbyopic and prescription is myope and astigmatism is no, then the recommended contact lens is none."*
2. *"If age is pre-presbyopic and prescription is hypermetrope and astigmatism is yes, then the recommended contact lens is none."*

After these rules are induced for the class "*none*," the dataset is reduced as shown in Table 3.4.

Next, we take *soft* as the next class for which to induce rules. The tests and their accuracy (*p*/*t*) values for the class are listed as follows:

Tests	***p*/*t***
Age is *young*	2/4
Age is *pre-presbyopic*	2/3
Age is *presbyopic*	1/2
Spectacle prescription is *myope*	2/5
Spectacle prescription is *hypermetrope*	3/4
Astigmatism is *no*	5/5

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

Astigmatism is *yes*	0/4
Tear production rate is *normal*	5/9

The test "*Astigmatism is no*" has the highest accuracy value. So, we select it as the first term of the new rule:

"If astigmatism is no, then the recommended contact lens is soft."

It is the only conditional term needed for the new rule because the accuracy (p/t) ratio is 100%. After all the instances covered by the new rule (five total) are excluded from the reduced dataset, four instances are left in the dataset, as shown in Table 3.5.

Next, we take class *hard* which is the only class left in the dataset. The tests and their accuracy (p/t) values for the class are listed as follows:

Tests	***p/t***
Age is *young*	2/2
Age is *pre-presbyopic*	1/1
Age is *presbyopic*	1/1
Spectacle prescription is *myope*	3/3
Spectacle prescription is *hypermetrope*	1/1
Astigmatism is *yes*	4/4
Tear production rate is *normal*	4/4

From the list, the two tests "*astigmatism is yes*" and "*tear production rate is normal*" have the highest p/t ratios and coverage. We select "*astigmatism is yes*"

TABLE 3.5 Contact lens dataset after all rules associated with classes *none* and *soft* are inducted

Age	Spectacle Prescription	Astigmatism	Tear Production Rate	Recommended Contact Lens
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	Yes	Normal	Hard
Pre-presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Myope	Yes	Normal	Hard

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

as the first term of the new rule. Since the *p/t* value of the test is 100%, the new rule for class *hard* is as follows:

"If astigmatism is yes, then the recommended contact lens is hard."

Since the new rule covers all four instances left in the reduced dataset, the process of rule formation is finished. In summary, six rules are induced from the contact lens dataset by the procedure. The following are the rules listed in the sequence they were created:

<Generated rule list>

Class: none

1. *"If tear production rate is reduced, then the recommended contact lens is none";*
2. *"If age is presbyopic and prescription is hypermetrope and astigmatism is yes, then the recommended contact lens is none";*
3. *"If age is presbyopic and prescription is myope and astigmatism is no, then the recommended contact lens is none";*
4. *"If age is pre-presbyopic and prescription is hypermetrope and astigmatism is yes, then the recommended contact lens is none."*

Class: soft

5. *"If astigmatism is no, then the recommended contact lens is soft."*

Class: hard

6. *"If astigmatism is yes, then the recommended contact lens is hard."*

The rule list represents the complete classification information of the contact lens dataset. It can be used to diagnose and make correct recommendations for patients. To identify an instance for the correct class, the rules are considered in turn from the beginning of the list and compared against the attributes of the instance until a matching rule is found. It is very important that rules in the rule list be applied in the same sequence as they were created. Because of the exclusion process, the rules produced later are induced without the knowledge of the instances covered by the previous rules. If we apply the rules created later first, the identified class may be incorrect. For example, if a case has the same attributes as those in row 23 of Table 3.2 and we use the last (sixth) rule in the rule list first, the answer will be *hard* instead of the correct class *none*.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

Notice that each of the above rules in the list has an accuracy of 100%. They are potentially over-fitted. Next, we are going to post-prune the rules as introduced with Induct. Rules 1, 5, and 6 have only one conditional term and will not be considered in the pruning process. Rules 2, 3, and 4 have three conditional terms each. We will use the probability measure as the criterion to greedily trim off the over-fit conditions of these rules. Let's start with rule 2. As stated above, rules cannot be used independently. The real meaning of rule 2 is as follows:

> *"If tear production rate is **not** reduced and age is presbyopic and prescription is hypermetrope and astigmatism is yes, then the recommended contact lens is none."*

From Table 3.2, the *P*, *T*, *p*, *t* values and the probability (Pr) are as follows:

$$T = 24, \qquad P = 15, \qquad t = 1, \qquad i = 1$$

$$\text{M(R)} = \sum_{i=1}^{t} \text{Pr[of } t \text{ instances selected at random, exactly } i \text{ are in class } none] = \sum_{i=1}^{t} \frac{\binom{P}{i}\binom{T-P}{t-i}}{\binom{T}{t}} = 0.625$$

After trimming off the last conditional term "*astigmatism is yes*," the *P*, *T*, *i*, *t* values and the probability (Pr) are as follows:

$$T = 24, \qquad P = 15, \qquad t = 2, \qquad i = 1$$

$$\text{Pr} = 0.870$$

Since this probability is much higher than the previous value, according to the Induct algorithm the post-pruning process should stop here, and the original rule should be kept.

How does probability change if we continue to trim off additional terms? After trimming off the second to the last condition term "*prescription is hypermetrope*," the *P*, *T*, *p*, *t* values and the probability (Pr) are the following:

$$T = 24, \qquad P = 15, \qquad t = 4, \qquad p = 2$$

$$\text{Pr} = 0.870$$

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

After trimming off the third to the last term "*age is presbyopic*," the P, T, p, t values and the probability (Pr) measure are as follows:

$$T = 24, \qquad P = 15, \qquad t = 12, \qquad p = 3$$

$$\mathrm{Pr} = 1.00$$

As seen in the example above, the original rule has the lowest probability value. Therefore, we abandon the trimmed rules and keep the original one. The same principle can be applied to rules 3 and 4. No conditional terms are pruned off from these rules. Therefore, the rule list remains the same.

■ 3.3.3 REP, IREP, RIPPER

The preceding procedure for generating rules leads to over-fitted rules since it uses the same dataset to generate and evaluate the rules ([Pagallo 1990], [Brunk 1991], and [Cohen 1995]). Reduced Error Pruning (REP) and other algorithms such as Incremental Reduced Error Pruning (IREP) [Furnkranz 1994] and RIPPER [Cohen 1995] were proposed to correct this problem. These algorithms split the dataset into two parts: the growing set and pruning set. The growing set is used to form the over-fitted rules, while the pruning set is used to prune and evaluate these rules. The procedures for these algorithms are as follows:

1. First, the training dataset is randomly partitioned into two subsets, a growing set and a pruning set. Usually the growing set contains 2/3 of the instances.
2. Next, the rule-growing step is carried out to form an over-fitted rule.
3. Then, the rule is immediately pruned by deleting conditions in the reverse order until no deletion improves the prediction of the rule.

The error measurements are different from one algorithm to another. The probability measurement described in Induct could be used here. However, a success ratio V, which is a simpler measurement, is used in IREP:

$$V = (p + (N - n))/T,$$

where p is the number of positive instances covered by the rule, N is the total number of instances not belonging to this class in the pruning dataset, n is the number of negative (wrong) instances covered by the rule, and T is

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

the total number of instances in the dataset. The algorithm maximizes the success ratio for the rule. The problem with this measurement is that it cannot tell the difference between the two rules that identify the same number of correct instances with different error rates. For example, a rule *A* that covers 3,000 positive instances with 2,000 negative instances will have the same success ratio as that of another rule that covers 1,001 positive instances with 1 negative instance. In fact, the error rate of the latter is much smaller than the former. Therefore, the probability measurement is probably a better choice for improving the prediction accuracies of rules, although it involves more computation.

The IREP algorithm is summarized in Figure 3.5.

FIGURE 3.5 IREP rule-generation algorithm

```
initialize T to be the instance set;
repeat
  do
  (1) split T into a growing set and pruning set;
  (2) use the basic covering algorithm to create a
      perfect rule from the growing set;
  (3) prune the perfect rule against the pruning set;
  (4) if the error rate of the pruned rule exceeds 50%
      then
        return rule set;
      else
        add the rule to rule set;
        remove the instances covered by the rule from
        both the growing set and pruning set;
      endif
until T is empty;
return rule set;
```

A disadvantage of the REP-type algorithms is that certain important information is prevented from being used in the growing rules because some instances are split into the pruning set. Moreover, some wrong rules may be preserved since the pruning set may not contain enough information to detect the error because it has only one-third of the instances of the whole dataset.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

The algorithm is fast and efficient, but not as accurate as a decision tree. The accuracies of predicting classes for unseen instances using existing rules can be improved through a global optimization step on the rule set by revising and replacing individual rules with the method introduced in RIPPER.

RIPPER slightly modifies the strategy used in REP and IREP to form and prune individual rules. In addition, RIPPER adds an additional optimization step to improve the prediction accuracy of the rule set by revising, deleting, or replacing the pruned rules. During the optimization process, for each rule produced from IREP, two alternative rules are constructed: (1) a replacement rule and (2) a revision rule. The former is formed by growing from the empty condition list and then pruning the conditions to minimize the error rate for the whole rule set, not just for one rule. The latter is formed by adding conditions greedily to the original rule until the global error rate increases. The final decision about whether the rule set should include the original rule, the replacement rule, the revision rule, or none, is made based on the Minimal Description Length (MDL) heuristic, which will be explained later in the section on C4.5. The final step in the optimization is to form additional rules to cover the remaining instances left out of the above optimization process.

The rule sets produced by RIPPER are significantly more accurate than those produced by REP and IREP, and are competitive with those of C4.5 without seriously affecting the algorithm's efficiency [Cohen 1995].

■ 3.4 DIVIDE-AND-CONQUER APPROACH

The divide-and-conquer approach is one of the most reliable approaches in classification learning based on using decision trees to induce classification information. The decision tree recursively selects attributes to test and splits the dataset into subsets according to the outcome of the test until a subset is obtained that contains instances of only one class. The prediction based on the integrated decision tree is more complete and accurate than the one based on the independent knowledge rules generated by the separate-and-conquer approach. Several systems have been implemented using this approach. Notable ones among them are ID3 and its successors C4.5 and C5.0 and Cart.

In the following sections, the ID3 algorithm will be presented and explained along with an example to demonstrate the application of the method.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

Then, it will be compared with its successors, which are modified versions with additional features.

■ 3.4.1 ID3

The underlying strategy of ID3 is to develop a decision tree recursively from top to bottom by using the information gain measure of each attribute as the criterion to select the attribute to test when splitting the dataset into subsets.

As previously discussed, the leaves in the decision tree represent the classes, and the internal nodes represent the attribute-based tests, which are connected with the branches that represent the outcomes of each test. Only one test is used for each branch. The process of identifying the class of an instance starts from the root of the tree and follows branches down the appropriate path according to the outcome of the tests (internal nodes) until a leaf is reached. The leaf is the class to which the instance belongs.

More than one decision tree can be derived from the same dataset depending on the sequence in which attributes are tested. The topologies of some trees are simpler than others. From our insect dataset in Table 3.1, two possible trees can be derived, as shown in Figure 3.1. It is obvious that tree 1 is more complex than tree 2, even though they both classify the insects into the correct suborder. Of course, the simpler decision tree is the better one because it requires the fewest steps to reach the correct conclusion.

Now, the question is how to build the simplest tree. The tedious way is to explore all possible trees, then choose the simplest one among them. A better way is to build the simplest tree using certain strategies. The strategy of ID3 is to choose the attribute with the smallest information gain and use it as the next test to split the dataset. This strategy guarantees that the tree built is the shortest.

Next, we introduce the concept of information gain and how it is applied in the ID3 algorithm to build decision trees.

Information

Let T be the set of instances to be classified and $\{C_1, C_2, ..., C_k\}$ be the set of classes. If S is any set of instances, let $Freq(C_j, S)$ stand for the number of instances in S that belongs to class C_j. We will also use the standard notation

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

$|S|$ to denote the number of instances in the set S. If we select one instance at random from S, the probability of the instance belonging to class C_j is $Freq(C_j, S)/|S|$ and the information it conveys is $-\log_2 [Freq(C_j, S)/|S|]$ bits. Then, the overall expected information needed to assign instances in S into classes is the sum of the weighted average information conveyed by each class:

$$Info(S) = -\sum_{j=1}^{k} [Freq(C_j, S)/|S|] \times \log_2 [Freq(C_j, S)/|S|] \text{ bits}$$

where k is the number of classes. $Info(S)$ describes the distribution evenness of instances among the classes. If the instances are evenly distributed, $Info(S)$ has the maximum value. If the distribution of the instances is so askew that one class has all the instances and the rest have none, then $Info(S)$ is zero. In classification learning, $Info(S)$ is used to measure the class diversities of the subsets split by the attribute. We favor knowledge rules that split the subsets with less class diversity.

Now let us consider the information measurement after the dataset T has been partitioned in accordance with the test results on an attribute X that has n possible values. First, T is divided into n partitions: $T_1, T_2, \ldots, T_i, \ldots, T_n$. For each subset T_i, we can calculate its information measurement, $Info(T_i)$, by using the previous formula. The expected information measurement of the test on the attribute X is the weighted sum of all these values, expressed as follows:

$$Info_X(T) = \sum_{i=1}^{n} [|T_i|/|T|] \times Info(T_i)$$

which measures the average class purity of the subsets with respect to the test attribute X, where n is the number of possible values of X. The smaller $Info_X(T)$ is, the less diversified (in terms of the classes) the subsets split by the attribute X are. If $Info_X(T)$ is 0, no more testing and further splitting are necessary for the subsets split by attribute X.

Information Gain

Another form of information measurement is Information Gain, which is the total information of dataset T minus the information of the test attribute X:

$$Gain_X(T) = Info(T) - Info_X(T)$$

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

where $Info(T)$ is a constant for each dataset. Contrary to $Info_X(T)$, the bigger the value of $Gain_X(T)$, the less pure the subsets are. When building a decision tree, ID3 examines all candidate attributes and chooses an attribute X with *maximal* $Gain_X(T)$ or *minimal* $Info_X(T)$ as the branching test. And then the same process is recursively used to construct the decision tree until each of the subsets of the test belongs to one class. At this point, there is an attribute X with $Info_X(T)$ value of 0.

As an illustration, let's use our insect dataset in Table 3.1 to demonstrate how ID3 is applied to building a decision tree. The information that the dataset contains is represented by $Info(T)$, which can be obtained as shown below:

Two classes are $C_1 = Endopterygota$, $C_2 = Exopterygota$, and their frequencies are $Freq(C_1, T) = 6$, $Freq(C_2, T) = 4$ with $|T| = 10$.
Then,

$$Info(T) = -[6/10] \times \log_2[6/10] - [4/10] \times \log_2[4/10] = 0.971 \text{ bits}$$

The information gain of the attribute *number of wings* is

$$\begin{aligned}
&T_1 = 2\ wings,\quad T_2 = 4\ wings,\quad |T_1| = 1,\quad |T_2| = 9,\quad Freq(C_1, T_1) = 1,\\
&Freq(C_2, T_1) = 0, Info_{\text{no. wing}}(T_1) = 0\\
&Freq(C_1, T_2) = 5, Freq(C_2, T_2) = 4\\
&Info_{\text{no. wing}}(T_2) = -5/9 \times \log_2[5/9] - 4/9 \times \log_2[4/9] = 0.991\\
&Gain_{\text{no. wing}}(T) = Info(T) - [1/10 \times Info_{\text{no. wing}}(T_1) + 9/10 \times Info_{\text{no. wing}}(T_2)]\\
&\qquad = 0.971 - [0 + 9/10 \times 0.991] = 0.079
\end{aligned}$$

The information gain of the attribute *metamorphosis* is

$$\begin{aligned}
&T_1 = complete,\quad T_2 = incomplete,\quad |T_1| = 6,\quad |T_2| = 4\\
&Freq(C_1, T_1) = 6, Freq(C_2, T_1) = 0, Info_{\text{meta}}(T_1) = 0\\
&Freq(C_1, T_2) = 0, Freq(C_2, T_2) = 4, Info_{\text{meta}}(T_2) = 0\\
&Gain_{\text{meta}}(T) = Info(T) - [6/10 \times Info_{\text{meta}}(T_1) + 4/10 \times Info_{\text{meta}}(T_2)]\\
&\qquad = 0.971 - [0 + 0] = 0.971
\end{aligned}$$

The information gain of the attribute *forewings* is

$$\begin{aligned}
&T_1 = membranous,\quad T_2 = sclerotized,\quad T_3 = hemi\text{-}sclerotized,\quad T_4 = leather\\
&|T_1| = 6, |T_2| = 1, |T_3| = 1, |T_4| = 2\\
&Freq(C_1, T_1) = 5, Freq(C_2, T_1) = 1\\
&Info_{\text{f. wing}}(T_1) = -5/6 \times \log_2[5/6] - 1/6 \times \log_2[1/6] = 0.65\\
&Freq(C_1, T_2) = 1, Freq(C_2, T_2) = 0, Info_{\text{f. wing}}(T_2) = 0\\
&Freq(C_1, T_3) = 0, Freq(C_2, T_3) = 1, Info_{\text{f.wing}}(T_3) = 0
\end{aligned}$$

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

$$\begin{aligned}
&Freq(C_1, T_4) = 0, Freq(C_2, T_4) = 2, Info_{\text{f. wing}}(T_4) = 0\\
&Gain_{\text{f. wing}}(T) = Info(T) - [6/10 \times Info_{\text{f. wing}}(T_1) + 1/10 \times Info_{\text{f. wing}}(T_2)\\
&\qquad + 1/10 \times Info_{\text{f. wing}}(T_3) + 2/10 \times Info_{\text{f. wing}}(T_4)]\\
&\qquad = 0.971 - [6/10 \times 0.65 + 0 + 0 + 0] = 0.581
\end{aligned}$$

The information gain of attribute *mouth* is:

$$\begin{aligned}
&T_1 = sponging, \quad T_2 = chewing, \quad T_3 = siphoning, \quad T_4 = piercing\\
&|T_1| = 1, |T_2| = 5, |T_3| = 2, |T_4| = 2\\
&Freq(C_1, T_1) = 1, Freq(C_2, T_1) = 0, Info_{\text{mouth}}(T_1) = 0\\
&Freq(C_1, T_2) = 3, Freq(C_2, T_2) = 2,\\
&Info_{\text{mouth}}(T_2) = -3/5 \times \log_2[3/5] - 2/5 \times \log_2[2/5] = 0.97\\
&Freq(C_1, T_3) = 2, Freq(C_2, T_3) = 0, Info_{\text{mouth}}(T_2) = 0\\
&Freq(C_1, T_4) = 0, Freq(C_2, T_4) = 2, Info_{\text{mouth}}(T_4) = 0\\
&Gain_{\text{mouth}}(T) = Info(T) - [5/10 \times Info_{\text{mouth}}(T_2) + 1/10 \times Info_{\text{mouth}}(T_1)\\
&\qquad + 2/10 \times Info_{\text{mouth}}(T_3) + 2/10 \times Info_{\text{mouth}}(T_4)]\\
&\qquad = 0.971 - [5/10 \times 0.97 + 0 + 0 + 0] = 0.486
\end{aligned}$$

The information gain of attribute *wing scales* is:

$$\begin{aligned}
&T_1 = absent, \quad T_2 = present, \quad |T_1| = 8, \quad |T_2| = 2, \quad Freq(C_1, T_1) = 4,\\
&Freq(C_2, T_1) = 4\\
&Info_{\text{scales}}(T_1) = -4/8 \times \log_2[4/8] - 4/8 \times \log_2[4/8] = 1\\
&Freq(C_1, T_2) = 2, Freq(C_2, T_2) = 0, Info_{\text{scales}}(T_2) = 0\\
&Gain_{\text{scales}}(T) = Info(T) - [8/10 \times Info_{\text{scales}}(T_1) + 2/10 \times Info_{\text{scales}}(T_2)]\\
&\qquad = 0.971 - [0.8 + 0] = 0.171
\end{aligned}$$

The information gain of attribute *hind legs* is:

$$\begin{aligned}
&T_1 = walking, \quad T_2 = jumping, \quad |T_1| = 9, \quad |T_2| = 1,\\
&Freq(C_1, T_1) = 6, Freq(C_2, T_1) = 3\\
&Info_{\text{hind leg}}(T_1) = -3/9 \times \log_2[3/9] - 6/9 \times \log_2[6/9] = 0.918\\
&Freq(C_1, T_2) = 0, Freq(C_2, T_2) = 1, Info_{\text{hind leg}}(T_2) = 0\\
&Gain_{\text{hind leg}}(T) = Info(T) - [9/10 \times Info_{\text{hind leg}}(T_1) + 1/10 \times Info_{\text{hind leg}}(T_2)]\\
&\qquad = 0.971 - [9/10 \times 0.918 + 0] = 0.144
\end{aligned}$$

The information gain of attribute *abdominal needle* is the same as attribute *wing scales*, i.e., $Gain_{\text{Abdomen needle}}(T) = 0.171$.

The largest information gain is that of attribute *metamorphosis*. Therefore, it is chosen as the root of the decision tree to be constructed. Since all the branching nodes of this test are the two classes, *endopterygota* and *exopterygota*,

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

no further testing is needed. At this point, we can observe that the value of $Info(T)$ is the same as that of $Gain_{\text{metamorphosis}}(T)$. The final decision tree is tree 2 in Figure 3.1. Notice that in tree 1, which is created by randomly selecting the test attributes, four tests are needed to classify the insects into the right suborders. However, only one test is needed in tree 2, from which the following two knowledge rules can be derived:

Rule 1: If metamorphosis is complete, then the suborder is Endopterygota.

Rule 2: If metamorphosis is incomplete, then the suborder is Exopterygota.

Noise

One of the notable features of the ID3 series of algorithms that are popular in data mining is their ability to handle noisy data. A problem with the decision-tree construction method discussed above is that it requires 100% accuracy, as in the Induct algorithm. The sub-datasets at the terminal leaves of a decision tree must belong to a single class. We have already discussed the problem of over-fitting in Induct. The over-fitting problem in decision trees is even worse. The decision tree can grow exponentially, much quicker than classification rules. For a large database, the decision tree may become so complicated that the rules generated from the tree may be too trivial to be comprehensible. In Quinlan's experiment with a set of noisy datasets, the expected error rate of an over-fitted decision tree was higher than a tree created randomly. In ID3, Quinlan proposed a mechanism to stop further branching. This mechanism allows the leaves of a decision tree to have a certain degree of heterogeneity.

The mechanism is based on the chi-square test for stochastic independence. Suppose that an attribute X splits a dataset T into subsets $[T_1, T_2, \ldots, T_v]$, where T_i contains p_i and n_i instances of class P and N, respectively, and v is the number of possible values of the attribute X. If a value of attribute X is irrelevant to the class of an instance in T, the expected value p_i' of p_i should be

$$p_i' = p \times (p_i + n_i)/(p + n)$$

If n_i' is the corresponding expected value of $n_{i,}$ then

$$X^2 = \sum_{i=1}^{v} [(p_i - p_i')/p_i' + (n_i - n_i')/n_i']$$

is approximately chi-square with $v - 1$ degrees of freedom. The chi-square value is used to determine the confidence with which one can reject the hypothesis that X is independent of the class of instances in T.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

Suppose we need to determine if further testing on attribute X is needed. Then, the chi-square value is calculated using the value of the attribute in the formula above. If the value is lower than a pre-assigned threshold, say 95%, we cannot reject that the test on attribute X is irrelevant to the classification. Further dividing of the datasets is not needed. If the value is higher than the threshold, the test on the attribute is necessary. If no attribute is found to be relevant, then the tree should stop growing at the point of the subtree. The decision tree built this way will avoid over-fitting.

A Problem of ID3

A problem with the ID3 algorithm is that it favors attributes with more possible outcomes over attributes with fewer possible outcomes. One extreme case is that each outcome of the attribute test covers only one instance in the dataset. After the splitting of the dataset, each subset will contain only one instance of one class. Since its $Info_X(T)$ value will be zero, ID3 will favor this attribute. For prediction, however, such a decision tree is less informative. For example, in Table 3.1, a distinct *insect ID number* for each insect may be added to the insect database as an additional attribute, as shown in Table 3.6.

TABLE 3.6 Morphological characters and suborders of insects from Table 3.1 modified with a new column "Insect ID Number" added and all other columns omitted except "Name" and "Groups"

Name	Insect ID Number	Groups
Fly	1	Endopterygota
Wasp	2	Endopterygota
Bee	3	Endopterygota
Beetle	4	Endopterygota
Butterfly	5	Endopterygota
Moth	6	Endopterygota
True Bug	7	Exopterygota
Aphid	8	Exopterygota
Grasshopper	9	Exopterygota
Cockroach	10	Exopterygota

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

The information gain of the attribute *insect ID number* is calculated as follows:

$$|T_1| = 1, \quad |T_2| = 1, \quad |T_3| = 1, \quad |T_4| = 1, \quad |T_5| = 1,$$
$$|T_6| = 1, \quad |T_7| = 1, \quad |T_8| = 1, \quad |T_9| = 1, \quad |T_{10}| = 1.$$
$$Freq(C_1, T_1) = 1, Freq(C_2, T_1) = 0; Freq(C_1, T_2) = 1, Freq(C_2, T_2) = 0;$$
$$Freq(C_1, T_3) = 1, Freq(C_2, T_3) = 0; Freq(C_1, T_4) = 1, Freq(C_2, T_4) = 0;$$
$$Freq(C_1, T_5) = 1, Freq(C_2, T_5) = 0; Freq(C_1, T_6) = 1, Freq(C_2, T_6) = 0;$$
$$Freq(C_1, T_7) = 0, Freq(C_2, T_7) = 1; Freq(C_1, T_8) = 0, Freq(C_2, T_8) = 1;$$
$$Freq(C_1, T_9) = 0, Freq(C_2, T_9) = 1; Freq(C_1, T_{10}) = 0, Freq(C_2, T_{10}) = 1.$$
$$Info_{\text{Id\_no}}(T_1) = 0, Info_{\text{Id\_no}}(T_2) = 0, Info_{\text{Id\_no}}(T_3) = 0, Info_{\text{Id\_no}}(T_4) = 0, Info_{\text{Id\_no}}(T_5) = 0, Info_{\text{Id\_no}}(T_6) = 0, Info_{\text{Id\_no}}(T_7) = 0, Info_{\text{Id\_no}}(T_8) = 0, Info_{\text{Id\_no}}(T_9) = 0, Info_{\text{Id\_no}}(T_{10}) = 0$$
$$\begin{aligned} Info_{\text{Id\_no}}(T) = [&1/10 \times Info_{\text{Id\_no}}(T_1) + 1/10 \times Info_{\text{Id\_no}}(T_2) \\ &+ 1/10 \times Info_{\text{Id\_no}}(T_3) + 1/10 \times Info_{\text{Id\_no}}(T_4) \\ &+ 1/10 \times Info_{\text{Id\_no}}(T_5) + 1/10 \times Info_{\text{Id\_no}}(T_6) \\ &+ 1/10 \times Info_{\text{Id\_no}}(T_7) + 1/10 \times Info_{\text{Id\_no}}(T_8) \\ &+ 1/10 \times Info_{\text{Id\_no}}(T_9) + 1/10 \times Info_{\text{Id\_no}}(T_{10})] = 0 \text{ bits} \end{aligned}$$
$$Gain_{\text{seq\_no}}(T) = Info(T) - Info_{\text{Id\_no}}(T) = 0.971 - 0 = 0.971$$

The information gain of this attribute is one of the highest among the other attributes considered earlier. Therefore, it could be one of the candidates for the first test attribute selected for the decision tree. If we select this attribute as the root of the decision tree, it would have ten branches connected to the root with each sub-dataset identifying only one instance. Each rule converted from the tree will be supported by only one instance in the dataset. Therefore, it will provide very poor classification information for the purpose of classifying insects. The problem above can be compensated for by the gain ratio criterion that can be used for the selection of the attributes, which will be discussed in the next section on C4.5.

■ 3.4.2 C4.5 and C5.0

C4.5 uses the same basic strategy of ID3 with gain ratio as an added feature for the attribute selection criteria to be used when branching. Several factors are used to deal with features such as missing values, noisy data, and numeric values in generating rules from the trees. The C5.0 algorithm is an extended system based on C4.5 that shows improved performance over C4.5.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

Gain Ratio Criterion

Gain ratio is a remedy for the problem of favoring attributes with more possible values when determining node branching in the tree, as mentioned in algorithm ID3. It was first proposed by Quinlan and later included in his system C4.5. It takes into account how the dataset splits on each node. First, it derives a split information value that takes into account the number and the size of children nodes and ignores any information about the classes. The larger the number of possible values of an attribute, the greater the split information is. The Split Information is expressed as follows:

$$\textit{Split info}(x) = -\sum_{i=1}^{n} [\textit{Freq}\ (T_i) \times \log_2(\textit{Freq}(T_i))]$$

Here, n is the number of possible values of the attribute X and $\textit{Freq}(T_i)$ is the number of instances with attribute X of value i. The gain ratio is the ratio of the information gain used in ID3 divided by the Split Information, as expressed by the following formula:

$$\textit{Gain ratio}(X) = \textit{Gain}_X(T)/\textit{Split info}_X(T).$$

In the C4.5 system, the gain ratio is used instead of information gain when selecting the next attribute to be used to split the dataset, which corrects the problem of favoring more variant attributes. In our insect example, the split information for *insect ID number* is

$$\begin{aligned}\textit{Split info}_{\mathrm{ID}}(T) &= -\sum_{i=1}^{n} [\textit{Freq}(T_i) \times \log_2(\textit{Freq}(T_i))]\\ &= 10 \times 0.1 \times 3.3219 = 3.3219\end{aligned}$$

and its gain ratio is

$$\textit{Gain ratio}\ (\textit{ID}) = \textit{Gain}_{ID}(T)/\textit{Split info}_{ID}(T) = 0.971/3.3219 = 0.2923$$

The split information for *metamorphosis* is

$$\begin{aligned}\textit{Split info}_{Meta}(T) &= -\sum_{i=1}^{n} [\textit{Freq}\ (T_i) \times \log_2(\textit{Freq}(T_i))]\\ &= -\,6/10 \times \log_2(6/10) - 4/10 \times \log_2(4/10)\\ &= -0.4422 - 0.5288 = 0.971\end{aligned}$$

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

and its gain ratio is

$$Gain\ ratio(Meta) = Gain_{Meta}(T)/Split\ info_{Meta}(T) = 0.971/0.971 = 1.$$

The gain ratio of the metamorphosis attribute is much higher than that of the ID attribute. Therefore, when building a decision tree, we will select the metamorphosis attribute instead of the ID attribute as the first splitting node, even though they have the same information gain value.

Gain ratio may be over-compensated for by preferring attributes with unevenly divided sub-datasets. This may be fixed by choosing the attribute that maximizes the gain ratio, provided that the information gain for that attribute is at least as big as the average information gain of all the attributes examined.

Post-Pruning

For the over-fitting problem in ID3, Quinlan used chi-square testing as the criterion to stop tree growth. This approach to dealing with the over-fitting problem in decision trees is often referred to as "pre-pruning," in contrast to another approach called "post-pruning," which trims off over-fitted branches after a complete explorative tree is built. In C4.5, Quinlan abandoned the "pre-pruning" approach and adopted the "post-pruning" approach used in Cart [Breiman 1984], although the latter needs more computation for building parts of the tree. However, the cost is offset by the benefits due to the generation of more reliable results through a more thorough exploration of possible partitions.

The "post-pruning" starts after the complete over-fitted decision tree is created. The pruning process is carried out from the bottom and works upward to the root of the tree, and occurs when an internal node is replaced with a lower-level node (possibly a leaf node) according to the error rate estimation of the internal node and its child nodes. (If the estimated error rate of the internal node is lower than the combined weighted estimated error rate of all of its child nodes, the partitioning of the dataset into subsets at this point might cause adverse effects on the prediction using the decision tree.) Then the internal node is replaced by one of the lower-level nodes (maybe a leaf node), depending on which represents the majority of instances over the sub-dataset.

To estimate the error rate of the internal nodes, Quinlan borrowed concepts from statistics. Let's assume that each sub-dataset included in an internal node is a sample of the whole dataset population (in fact it is not). Given a confidence level, we can estimate the population (i.e., whole dataset) error

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

rate through the observed errors from the samples (the subsets). The probability of a random variable X, with 0 mean and a confidence range of $2z$ is $\Pr[-z \le X \le z] = c$, where c is the confidence level and z is the standard deviation of the variable X away from the mean. The z value can be obtained from the normal distribution with any given confidence level c. For example, if the confidence $c = 90\%$, then z is 1.65 from $\Pr[-1.65 \le X \le 1.65] = 90\%$. The implication is that there is a 90% chance that X lies between 1.65 standard deviations above and below the mean 0. If we increase the confidence level to 99%, z becomes 2.58, i.e., X lies on a wider area of random distribution. Quinlan used the upper-tailed probability to estimate the error rate. The one-tailed probability is expressed as $\Pr[X \ge z] = (1 - c)/2$ because the random distribution is symmetric. If c is 90%, the upper-tailed probability equals 5%. In fact, the random probability distribution in most statistics applications is expressed as one-tailed. Now let's see how to apply the above statistical measure to estimate the error rate of the internal nodes.

If E is the observed number of error instances, N is the total number of instances in the sub-dataset, and f is the observed error rate (i.e., $f = E/N$), then the above one-tailed random probability is expressed as $\Pr[(f - q)/\sqrt{q(1-q)/N} > z] = e$, where f (the random variable) minus the mean q (the estimated error rate of the population) is divided by $\sqrt{q(1-q)/N}$ (the standard deviation), and e is the upper-tailed confidence level. Given the one-tailed confidence level e, we can get the error rate q value by getting the z value first and then solving the inequality in the expression above, which is then converted as shown:

$$q = (f + z^2/2N + z\sqrt{f/N - f^2/N + z^2/4N^2})/(1 + z/N)$$

Since C4.5 used $e = 25\%$ as the default one-tailed confidence level, its corresponding z becomes 0.69.

Now that we have learned how to estimate the error rate of the decision-tree nodes, let's see how the estimated error rate can be applied to pruning the decision tree. The process can be summarized as follows:

1. Calculate the error rates of an internal node and all of its next-level children.
2. Sum the weighted error rates of these children, which is the combined error estimation of the direct children.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

3. Compare the estimate error of the internal node and the combined error rate of its children. If the estimated error rate of the internal node is larger than the children's combined error rate, the splitting of the decision tree at the internal node will improve its prediction. Therefore, pruning of the subtrees at this node will not be done. Otherwise, the splitting at the node will degrade the prediction based on the decision tree, and pruning of the subtree is appropriate here.
4. When pruning is appropriate, the pruning is done by replacing the internal node either by a leaf child or by any other lower-level internal node. The choice is made by identifying the child node covering the most instances. If the selected node is a leaf node, it simply replaces the parent node. If the replacing node is an internal node, the child subtree should be rearranged to include all instances covered by other children being trimmed off before the pruning. Therefore, replacing with a subtree is much more expensive than replacing with a leaf node.

Rule Set

After post-pruning is done to trim the decision trees, the knowledge rules can be read directly from the pruned tree. C4.5 uses another complicated and time-consuming procedure for deriving the rules. It converts the over-fitted decision tree into a set of rules with the following steps:

1. Over-fitted rules are generated from the decision tree. They are generated directly from the over-fitted decision tree.
2. *Optimization of single rules.*

   After the over-fitted rules are created directly from the over-fitted decision tree, C4.5 uses a try-and-test strategy to simplify conditions in the original rules without decreasing accuracy. The testing mechanism used here is the same as the confidence level testing used in the tree-pruning procedure mentioned above. First, a condition is temporarily removed from the rule. Then, the estimated upper limits of the error rates are compared. If the estimated upper limit of the error rate of a rule with the condition temporarily removed has the same or lower estimated upper limit of the error rate of the original rule without temporary removal of the condition, the deletion of the condition from the rule will not

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

degrade the performance of the rule under a certain confidence level. Therefore, the deletion of the condition will be finalized.

Now let's consider the complexity of multiple conditions in a rule. If the number of conditions in a rule is N, there are $N!$ possible combinations of the conditions. If we carry out an exhaustive search on all possible conjunctions of the conditions to determine whether to accept or reject the temporary rule, it takes $N!$ number of tests for the single rule. For a large database, the number of possible tests will become very huge. In C4.5, Quinlan used a "greedy" approach to delete the conditions, which produces reasonably accurate rules and is much faster than the exhaustive search. This approach deletes one condition at a time until no more conditions need to be removed. First, we calculate and list the estimated upper-limit error rates after deleting each condition from the original rule. Rules with estimated upper-limit error rates lower than the original rule will be accepted as new rules. If more than one condition qualifies, select the one with the lowest estimated upper-limit error rate. Since the deletion of the condition from the original rule may change the instance coverage, the default estimated upper-limit error rate of the new rule needs to be recalculated. From the discussion above, we can see that the maximal number of tests in greedy search is the same as in exhaustive deletion. Under this circumstance, all conditions are trimmed off. Since the conditions deleted are usually less than the total number of conditions in the rule, the greedy approach will speed up the process of single-rule optimization.

The possible adverse effect of rule optimization is that the rules produced from a decision tree may no longer be mutually inclusive or exclusive or both. This means that some instances may not be covered by any rule, while other instances may be covered by two or more rules. In C4.5, the former situation is resolved by adding a default rule to deal with the instances not covered by any rule. The conflict in the second situation is resolved by ranking the rules according to their priorities, and the rule with the highest priority is taken as the target rule.

3. *Optimization of rule set*
   In the previous two steps, we discussed ways to optimize individual rules. C4.5 further implemented a mechanism to optimize an entire rule set to improve the performance of the rule set as a whole. C4.5 first optimizes

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

the rules in a rule subset, each denoting a class, by using the Minimum Description Length (MDL) principle [Rissanen 1983a & 1983b]. Then, these subsets are ranked according to their priorities. If two conflicting rules share the same conditions but predict different classes, then they are distributed in separate rule subsets with different priority values. The rule in the rule subset with a higher priority will be used to identify the instances. The sequence of the rules within the same subset of the class is irrelevant. Lastly, the default rule is set for the instances not covered by any of the rules in the rule set.

Similar to upper-limit error estimation in individual rule formation, the MDL is applied in C4.5 as the criterion to optimize the rule subset for each class. The principle is that both a sender and a receiver have identical copies of training instances, but the sender's copy also specifies the class of each instance while the receiver's copy lacks any class information. The sender must communicate this missing information to the receiver by transmitting the classification theory together with exceptions to the theory. The sender may choose the complexity of the theory he sends, for example, a relatively simple theory with a substantial number of exceptions, or a more complete theory with fewer exceptions. The MDL principle states that the best theory derivable from the training data minimizes the number of bits required to encode the total message consisting of the theory together with its exceptions.

In our classification applications, the information that is derived through the theory (the set of rules for one class) and exceptions (misidentified instances) is the identification of the instances belonging to each target class. The purpose of MDL testing is to find the best subset of rules for a class from a number of possible combinations of the rules, which minimizes the encoding of the theory and the exceptions. The process is a bit complicated and the computation is time-consuming.

- The process first encodes each rule by calculating its associated information bits based on all conditions of the rule, which are then subtracted by the ordering credit of the conditions, since the relative sequence of the conditions is irrelevant to the conclusion of the rule. The value of the ordering information is $\log_2(k!)$ if there are k conditions on the rule.
- The theory information is the sum of the information bits of all rules in the subset, subtracted by the ordering information of the individual rules,

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

which is $\log_2(R!)$ if the total number of rules in the subset is R, since the ordering of the rules in the same class is irrelevant, as stated previously.

- The exceptions are encoded by summing the false-positive information bits and the false-negative information bits. The false-positive instances are the instances misidentified as belonging to different classes. The false-negative instances are those that are incorrectly excluded from their classes. If the rule covers r out of n training instances, with fp false-positive instances and fn false-negative instances, the information bits of the exceptions are

$$\log_2\binom{r}{fp} + \log_2\binom{n-r}{fn}$$

The information of the whole rule subset is the sum of the theory bits and the exceptions. This value measures the performance of the rule subset. The smaller the value, the better the performance of the rule subset. The purpose of optimization is to exclude from the subset certain rules that adversely affect the MDL value of the rule subset. The question is how to find the rule subset with the lowest MDL value from all possible combinations of the rules for the class. If the size of the rule set is small, we can do an exhaustive search on all possible combinations of the rules and find the subset with the lowest MDL value. C4.5 also adopted a simulated annealing method to find a near-best rule subset in a very large rule set. This method is more computationally effective and can produce more satisfactory results. The detailed description of the method is given in [Press 1988].

After the best rule subset is found for each class, the next step of the rule set optimization is to rank the rule subsets. If a rule with a higher priority in a rule subset positively identifies a false instance, then the rule that may give a correct answer in the subset with low priority may never get a chance to be checked. Therefore, the rule subset with a lower false-positive error rate should be ranked with a higher priority.

To those instances for which there is no corresponding rule, a default class should be given. It is reasonable to set the class that appears most frequently in the training database as the default class. Although the rule-set optimization process in C4.5 is rather lengthy and complicated, it is necessary to reduce errors of prediction, especially if the dataset contains a lot of noisy data.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

In the following, we are going to apply the techniques in the C4.5 algorithm to build a decision tree, post-prune the tree, and then induce knowledge rules from the contact lens database (Table 3.2). The final decision tree is shown in Figure 3.6. At the root, there are 3 classes and 24 instances in total. The information gain and gain ratio of the attributes available at this level are calculated as shown below:

Three Classes: $C_1 = none$, $C_2 = soft$, $C_3 = hard$
Total number of instances: $|T| = 24$
Frequencies of each class: $Freq(C_1, T) = 15$, $Freq(C_2, T) = 5$, $Freq(C_3, T) = 4$
Information bits of the node: $Info(T) = -[15/24] \times \log_2[15/24] - [5/24] \times \log_2[5/24] - [4/24] \times \log_2[4/24] = 1.326$ bits

Attribute: *Age*

$T_1 = young$, $|T_1| = 8$, $T_2 = pre\text{-}presbyopic$, $|T_2| = 8$,
$T_3 = presbyopic$, $|T_3| = 8$
$Freq(C_1, T_1) = 4$, $Freq(C_2, T_1) = 2$, $Freq(C_3, T_1) = 2$
$Info_{age}(T_1) = -4/8 \times \log_2[4/8] - 2/8 \times \log_2[2/8] - 2/8 \times \log_2[2/8] = 1.5$
$Freq(C_1, T_2) = 5$, $Freq(C_2, T_2) = 2$, $Freq(C_3, T_2) = 1$
$Info_{age}(T_2) = -5/8 \times \log_2[5/8] - 2/8 \times \log_2[2/8] - 1/8 \times \log_2[1/8] = 1.299$
$Freq(C_1, T_3) = 6$, $Freq(C_2, T_3) = 1$, $Freq(C_3, T_3) = 1$
$Info_{age}(T_3) = -6/8 \times \log_2[6/8] - 1/8 \times \log_2[1/8] - 1/8 \times \log_2[1/8] = 1.061$
$Gain_{age}(T) = Info(T) - [8/24 \times Info_{age}(T_1) + 8/24 \times Info_{age}(T_2) + 8/24 \times Info_{age}(T_3)] = 1.326 - 1.2633 = 0.0627$
$Gain\ Ratio = 0.0627/1.2633 = 0.0496$

Attribute: *Spectacle Prescription*

$T_1 = myope$, $|T_1| = 12$, $T_2 = hypermetrope$, $|T_2| = 12$
$Freq(\mathrm{C}_1, T_1) = 7$, $Freq(\mathrm{C}_2, T_1) = 2$, $Freq(\mathrm{C}_3, T_1) = 3$
$Info_{pres}(T_1) = -7/12 \times \log_2[7/12] - 2/12 \times \log_2[2/12] - 3/12 \times \log_2[3/12] = 1.384$

$Freq(C_1, T_2) = 8$, $Freq(C_2, T_2) = 3$, $Freq(C_3, T_2) = 1$

$Info_{pres}(T_2) = -8/12 \times \log_2[8/12] - 3/12 \times \log_2[3/12] - 1/12 \times \log_2[1/12] = 1.189$

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

$$\begin{aligned} Gain_{\text{pres}}(T) &= Info(T) - [12/24 \times Info_{\text{pres}}(T_1) + 12/24 \times Info_{\text{pres}}(T_2)] \\ &= 1.326 - 1.287 = 0.039 \end{aligned}$$

$$Gain\ Ratio = 0.039/1.326 = 0.02941$$

Attribute: *Astigmatism*

$$T_1 = yes, \quad |T_1| = 12, \qquad T_2 = no, \quad |T_2| = 12$$
$$Freq(C_1, T_1) = 8, Freq(C_2, T_1) = 0, Freq(C_3, T_1) = 4$$
$$Info_{\text{ast}}(T_1) = -8/12 \times \log_2[8/12] - 4/12 \times \log_2[4/12] - 0 = 0.918$$
$$Freq(C_1, T_2) = 7, Freq(C_2, T_2) = 5, Freq(C_3, T_2) = 0$$
$$Info_{\text{ast}}(T_2) = -7/12 \times \log_2[7/12] - 5/12 \times \log_2[5/12] - 0 = 0.098$$
$$\begin{aligned} Gain_{\text{ast}}(T) &= Info(T) - [12/24 \times Info_{\text{ast}}(T_1) + 12/24 \times Info_{\text{ast}}(T_2)] \\ &= 1.326 - 0.949 = 0.377 \end{aligned}$$
$$Gain\ Ratio = 0.377/1.\ 326 = 0.284$$

Attribute: *Tear Production Rate*

$$T_1 = reduced, \quad |T_1| = 12, \qquad T_2 = normal, |T_2| = 12$$
$$Freq(C_1, T_1) = 12, Freq(C_2, T_1) = 0, Freq(C_3, T_1) = 0$$
$$Info_{tear}(T_1) = 0$$
$$Freq(C_1, T_2) = 3, Freq(C_2, T_2) = 5, Freq(C_3, T_2) = 4$$
$$\begin{aligned} Info_{tear}(T_2) = &-3/12 \times \log_2[3/12] - 5/12 \times \log_2[5/12] \\ &-4/12 \times \log_2[4/12] = 1.555 \end{aligned}$$
$$\begin{aligned} Gain_{tear}(T) &= Info(T) - [12/24 \times Info_{tear}(T_1) + 12/24 \times Info_{tear}(T_2)] \\ &= 1.326 - 0.778 = 0.548 \end{aligned}$$
$$Gain\ Ratio = 0.548/1.326 = 0.413$$

Since the attribute *tear production rate* has the highest information gain ratio of 0.413, we select this attribute for the first test to partition the dataset. As was illustrated in Figure 3.6, the left branch has 12 *none* classified into only one class, so it requires no further splitting. The right branch contains three instances of *none*, five instances of *soft*, and four instances of *hard* classes:

$$C_1 = none, \qquad C_2 = soft, \qquad C_3 = hard, \quad |T| = 12$$
$$Freq(C_1, T) = 3, Freq(C_2, T) = 5, Freq(C_3, T) = 4$$
$$\begin{aligned} Info(T) = &-[3/12] \times \log_2[3/12] - [5/12] \times \log_2[5/12] \\ &-[4/12] \times \log_2[4/12] = 1.555 \text{ bits} \end{aligned}$$

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

Attribute: *Age*

$T_1 = young, \quad |T_1| = 4, \qquad T_2 = pre\text{-}presbyopic, \quad |T_2| = 4,$
$T_3 = presbyopic, \quad |T_3| = 4$
$Freq(C_1, T_1) = 0, Freq(C_2, T_1) = 2, Freq(C_3, T_1) = 2$
$Info_{age}(T_1) = -0 - 2/4 \times \log_2[2/4] - 2/4 \times \log_2[2/4] = 1$
$Freq(C_1, T_2) = 1, Freq(C_2, T_2) = 2, Freq(C_3, T_2) = 1$

$Info_{age}(T_2) = -1/4 \times \log_2[1/4] - 2/4 \times \log_2[2/4] - 1/4 \times \log_2[1/4] = 1.5$

$Freq(C_1, T_3) = 2, Freq(C_2, T_3) = 1, Freq(C_3, T_3) = 1$

$Info_{age}(T_3) = -2/4 \times \log_2[2/4] - 1/4 \times \log_2[1/4] - 1/4 \times \log_2[1/4] = 1.5$

$Gain_{age}(T) = Info(T) - [4/12 \times Info_{age}(T_1) + 4/12 \times Info_{age}(T_2) + 4/12$
$\times Info_{age}(T_3)] = 1.555 - 1.333 = 0.222$
$Gain\ Ratio = 0.222/1.555 = 0.143$

Attribute: *Spectacle Prescription*

$T_1 = myope, \quad |T_1| = 6, \qquad T_2 = hypermetrope, \quad |T_2| = 6$
$Freq(C_1, T_1) = 1, Freq(C_2, T_1) = 2, Freq(C_3, T_1) = 3$
$Info_{pres}(T_1) = -1/6 \times \log_2[1/6] - 2/6 \times \log_2[2/6] - 3/6 \times \log_2[3/6] = 1.459$

$Freq(C_1, T_2) = 2, Freq(C_2, T_2) = 3, Freq(C_3, T_2) = 1$

$Info_{pres}(T_2) = -2/6 \times \log_2[2/6] - 3/6 \times \log_2[3/6] - 1/6 \times \log_2[1/6]$
$= 1.459$

$Gain_{pres}(T) = Info(T) - [6/12 \times Info_{pres}(T_1) + 6/12 \times Info_{pres}(T_2)]$
$= 1.555 - 1.459 = 0.096$
$Gain\ Ratio = 0.096/1.555 = 0.061$

Attribute: *Astigmatism*

$T_1 = yes, \quad |T_1| = 6, \qquad T_2 = no, \quad |T_2| = 6$
$Freq(C_1, T_1) = 2, Freq(C_2, T_1) = 0, Freq(C_3, T_1) = 4$
$Info_{ast}(T_1) = -2/6 \times \log_2[2/6] - 0 - 4/6 \times \log_2[4/6] = 0.918$
$Freq(C_1, T_2) = 1, Freq(C_2, T_2) = 5, Freq(C_3, T_2) = 0$
$Info_{ast}(T_2) = -1/6 \times \log_2[1/6] - 5/6 \times \log_2[5/6] - 0 = 0.650$
$Gain_{ast}(T) = Info(T) - [6/12 \times Info_{ast}(T_1) + 6/12 \times Info_{ast}(T_2)] = 1.555 - 0.784$
$= 0.771$
$Gain\ Ratio = 0.771/1.555 = 0.496$

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

Since *astigmatism* has the highest gain ratio, it is selected as the next test to split the right branch into two subsets: *yes* and *no* branches. Let's examine the *no* branch first.

$$C_1 = soft, \quad |C_1| = 5, \qquad C_2 = none, \quad |C_2| = 1, \quad |T| = 6$$
$$Info(T) = -[5/6] \times \log_2[5/6] - [1/6] \times \log_2[1/6] = 0.650 \text{ bits}$$

Attribute: *Age*

$T_1 = young, \quad |T_1| = 2, \qquad T_2 = pre\text{-}presbyopic, \quad |T_2| = 2,$
$T_3 = presbyopic, \quad |T_3| = 2$
$Freq(C_1, T_1) = 2, Freq(C_2, T_1) = 0, Info_{age}(T_1) = -\ 2/2 \times \log_2[2/2] - 0 = 0$
$Freq(C_1, T_2) = 2, Freq(C_2, T_2) = 0, Info_{age}(T_2) = -\ 2/2 \times \log_2[2/2] - 0 = 0$
$Freq(C_1, T_3) = 1, Freq(C_2, T_3) = 1$
$Info_{age}(T_3) = -1/2 \times \log_2[1/2] - 1/2 \times \log_2[1/2] = 1$
$Gain_{age}(T) = Info(T) - [2/6 \times Info_{age}(T_1) + 2/6 \times Info_{age}(T_2) + 2/6 \times Info_{age}(T_3)]$
$= 0.650 - 0.333 = 0.617$
$Gain\ Ratio = 0.617/0.65 = 0.949$

Attribute: *Spectacle Prescription*

$T_1 = myope, \quad |T_1| = 3, \qquad T_2 = hypermetrope, \quad |T_2| = 3,$
$Freq(C_1, T_1) = 2, \qquad Freq(C_2, T_1) = 1$
$Info_{pres}(T_1) = -\ 2/3 \times \log_2[2/3] - 1/3 \times \log_2[1/3] = 0.918$
$Freq(C_1, T_2) = 3, Freq(C_2, T_2) = 0, Info_{pres}(T_2) = -\ 3/3 \times \log_2[3/3] - 0 = 0$
$Gain_{pres}(T) = Info(T) - [3/6 \times Info_{pres}(T_1) + 3/6 \times Info_{pres}(T_2)]$
$= 0.650 - 0.500 = 0.150$
$Gain\ Ratio = 0.15/0.65 = 0.231$

In this branch, the attribute *age* has the highest information gain ratio. This branch is divided into three subsets: *young, pre-presbyopic*, and *presbyopic*. The two branches *young* and *pre-presbyopic* are the leaf nodes, each with two instances of *soft*. The branch *presbyopic* has one instance of *soft* and one instance of *none*, which can be further partitioned by the attribute *prescription* to reach the leaf nodes.

Let's go back to the right branch of the *astigmatism* attribute, which is the *yes* branch as shown in Figure 3.6:

$$|T| = 6, \quad C_1 = hard, \qquad |C_1| = 4, \qquad C_2 = none, \quad |C_2| = 2,$$
$$Info(T) = -[4/6] \times \log_2[4/6] - [2/6] \times \log_2[2/6] = 0.918 \text{ bits}$$

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

Attribute: *Age*

$T_1 = \textit{young}, \quad |T_1| = 2, \qquad T_2 = \textit{pre-presbyopic}, \quad |T_2| = 2,$
$T_3 = \textit{presbyopic}, \quad |T_3| = 2,$
$\textit{Freq}(C_1, T_1) = 2, \textit{Freq}(C_2, T_1) = 0, \textit{Info}_{\text{age}}(T_1) = -\,2/2 \times \log_2[2/2] - 0 = 0$
$\textit{Freq}(C_1, T_2) = 1, \textit{Freq}(C_2, T_2) = 1$
$\textit{Info}_{\text{age}}(T_2) = -1/2 \times \log_2[1/2] - 1/2 \times \log_2[1/2] = 1$
$\textit{Freq}(C_1, T_3) = 1, \textit{Freq}(C_2, T_3) = 1$
$\textit{Info}_{\text{age}}(T_3) = -1/2 \times \log_2[1/2] - 1/2 \times \log_2[1/2] = 1$
$\textit{Gain}_{\text{age}}(T) = \textit{Info}(T) - [2/6 \times \textit{Info}_{\text{age}}(T_1) + 2/6 \times \textit{Info}_{\text{age}}(T_2) + 2/6 \times \textit{Info}_{\text{age}}(T_3)]$
$\qquad = 0.918 - 0.667 = 0.251$
$\textit{Gain Ratio} = 0.617/0.918 = 0.273$

Attribute: *Spectacle Prescription*

$T_1 = \textit{myope}, \quad |T_1| = 3, \qquad T_2 = \textit{hypermetrope}, \quad |T_2| = 3$
$\textit{Freq}(C_1, T_1) = 3, \textit{Freq}(C_2, T_1) = 0, \textit{Info}_{\text{pres}}(T_1) = -\,2/3 \times \log_2[2/3] - 0 = 0$
$\textit{Freq}(C_1, T_2) = 1, \textit{Freq}(C_2, T_2) = 2$
$\textit{Info}_{\text{pres}}(T_2) = -1/3 \times \log_2[1/3] - 2/3 \times \log_2[2/3] = 0.918$
$\textit{Gain}_{\text{pres}}(T) = \textit{Info}(T) - [3/6 \times \textit{Info}_{\text{pres}}(T_1) + 3/6 \times \textit{Info}_{\text{pres}}(T_2)]$
$\qquad = 0.918 - 0.459 = 0.359$
$\textit{Gain Ratio} = 0.359/0.918 = 0.391$

The attribute *prescription* is selected as the test to split the subset into two children along the branches *myope* and *hypermetrope*. The first branch includes only three instances of class *hard*. The second branch includes two instances of *none* and one instance of *hard* and is further split by the attribute *age*. The final decision tree is shown in Figure 3.6.

The decision tree in Figure 3.6 is a complete (perfect) tree. It may be over-fitted, though. There are nine leaf nodes on the tree, so nine rules could be derived from the tree. Notice that the number of rules generated by this method is much greater than the Induct algorithm we used earlier. Thus, the post-pruning technique is used to trim the tree. After that, a new set of rules is generated from the trimmed tree. We use the one-tailed probability of 25% as the confidence level. Its corresponding Z value is 0.69. The process will start from the lowest-level internal node on the right side of the tree (*prescription*). The error rates of the two leaf nodes are calculated as follows:

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

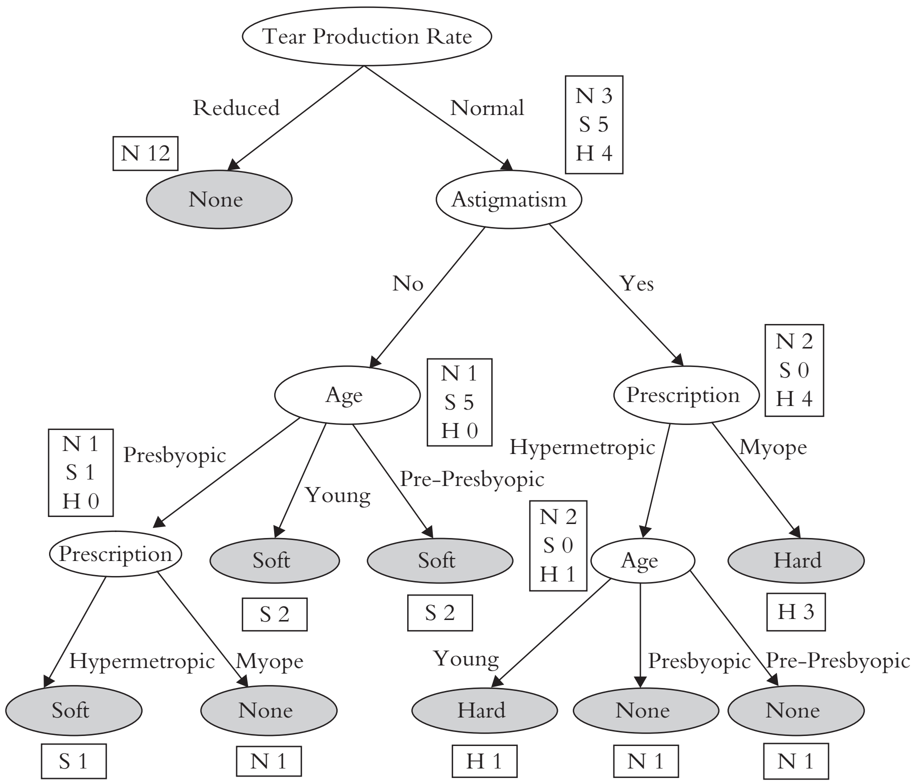

FIGURE 3.6 Decision tree generated from contact lens dataset

1. Non-shaded circles are the internal nodes representing test attributes.
2. Shaded circles are the leaf nodes representing classes.
3. Branches represent the test values.
4. Letters and digits in squares represent the number of instances per class for each node, such as *N* for *none*, *S* for *soft*, and *H* for *hard* lens.

Soft:

$$f = 0, \qquad z = 0.69 \text{ when } c = 25\%, N = 1,$$
$$q = (f + z^2/2N + z\sqrt{f/N - f^2/N + z^2/4N^2})/(1 + z^2/N)$$
$$= z^2/(N + z^2) = (0.69 \times 0.69)/(1 + 0.69 \times 0.69) = 0.322$$

None:

$$f = 0, \qquad z = 0.69 \text{ when } c = 25\%, \qquad N = 1,$$

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

$$q = (f + z^2/2N + z\sqrt{f/N - f^2/N + z^2/4N^2})/(1 + z^2/N)$$
$$= z^2/(N + z^2) = (0.69 \times 0.69)/(1 + 0.69 \times 0.69) = 0.322$$

The weighted average of the two children is $q = 1/2 \times 0.322 + 1/2 \times 0.322 = 0.322$. The estimated error rate of the internal node "prescription" itself is

$$f = 1/2 = 0.5, \quad N = 2,$$
$$q = (f + z^2/2N + z\sqrt{f/N - f^2/N + z^2/4N^2})/(1 + z^2/N)$$
$$= (0.5 + 0.69^2/(2 \times 2) + 0.69 \times \sqrt{f/N - f^2/N + z^2/4N^2})/(1 + 0.69^2/2)$$
$$= (0.5 + 0.119 + 0.2714)/1.24 = 0.719$$

Because the children's average error rate 0.322 is lower than the parent's 0.719, we keep the parent node. Let's move one level upward to the *age* node:

$$N = 6, \quad f = 1/6 = 0.167, \quad q = (0.167 + 0.0396 + 0.112)/1.079 = 0.295.$$

The estimated error rates of its two other children, *young* and *pre-presbyopic*, are calculated as follows:

Young:

$$N = 2, \quad f = 0, \quad q = z^2/(N + z^2) = 0.476/(2 + 0.476) = 0.192$$

Pre-presbyopic:

$$N = 2, \quad f = 0, \quad q = z^2/(N + z^2) = 0.476/(2 + 0.476) = 0.192$$

The weighted average error rate of the three children is $Q = (2/6) \times 0.192 + (2/6) \times 0.192 + (2/6) \times 0.719 = 0.368$. Because the error rate of the children (0.368) is larger than the error rate of the internal node *age* (0.295), we replace the internal node *age* with the leaf node *soft*, which is the dominant class in the subset. The replaced node has the error rate $q = 0.295$. After trimming off the nodes, the decision tree is as shown in Figure 3.7.

We now examine the right lowest-level internal node to move upwards one node along the path. The internal node *age* becomes the lowest-level internal node, as shown in Figure 3.7.

$$N = 3, \quad f = 1/3 = 0.333, \quad q = (0.333 + 0.079 + 0.204)/1.159 = 0.532$$

The error rates of its children node are as follows:

Young: $N = 1, \quad f = 0, \quad q = z^2/(N + z^2) = 0.476/(1 + 0.476)$
$= 0.322$

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

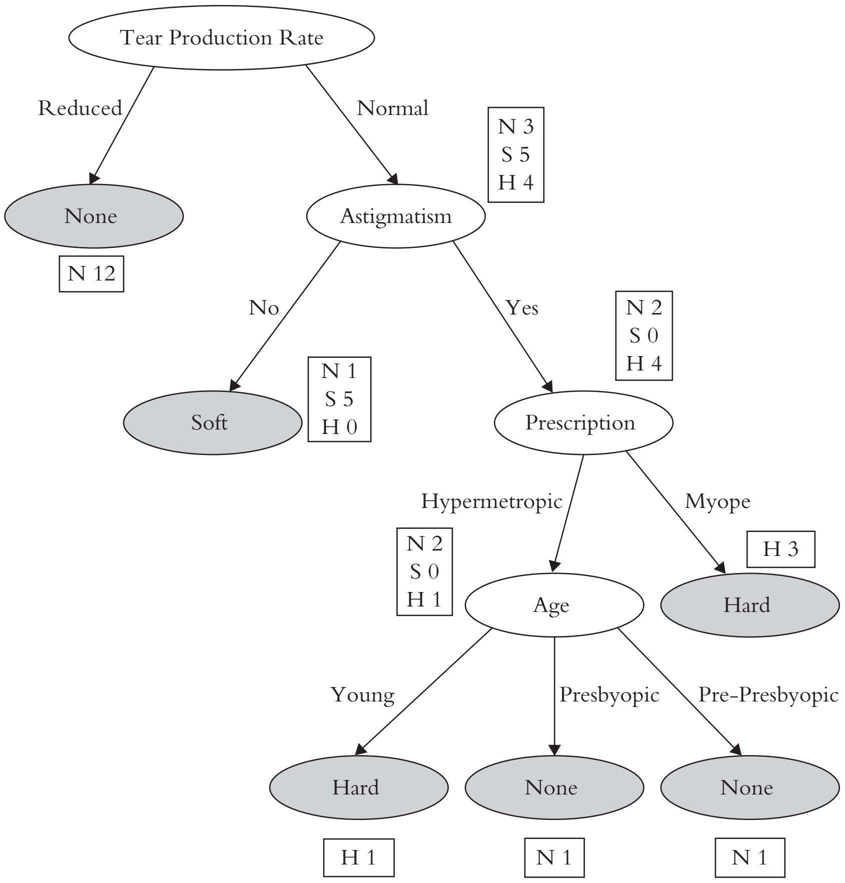

FIGURE 3.7 Decision tree trimmed from Figure 3.6

Pre-Presbyopic: $N = 1$, $f = 0$, $q = z^2/(N + z^2) = 0.476/(1 + 0.476)$
$= 0.322$

Presbyopic: $N = 1$, $f = 0$, $q = z^2/(N + z^2) = 0.476/(1 + 0.476)$
$= 0.322$

The weighted average is $q = (1/3) \times 0.322 + (1/3) \times 0.322 + (1/3) \times 0.322 = 0.322$. Since the internal error rate of the node *age* is bigger than those of its children, we keep the node *age*. Now the next node to examine is the internal node *prescription*:

$$N = 6, \quad f = 2/6 = 0.667, \quad q = \left(f + z^2/2N + z\sqrt{f/N - f^2/N + z^2/4N^2}\right)/(1 + z^2/N) = (0.333 + 0.040 + 0.139)/1.079 = 0.474$$

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

The right child leaf node *hard* has the following error rate:

$$N = 3, \qquad f = 0, \qquad q = z^2/(N + z^2) = 0.476/(3 + 0.476) = 0.137$$

Since the left child leaf node *age* has an error rate of 0.322, the weighted children's average error rate is $q = 3/6 \times 0.322 + 3/6 \times 0.137 = 0.230$, which is less than the internal node *prescription's* error rate 0.474. Therefore, we keep the internal node.

Now let's examine the internal node *astigmatism*:

$$f = 5/12, \qquad N = 12, \qquad q = (0.417 + 0.020 + 0.100)/1.040 = 0.516$$

The weighted error rate of the children *prescription* and *age* is

$$q = 6/12 \times 0.295 + 6/12 \times 0.474 = 0.385$$

Since the children's average error rate is less than the parent's, we keep the internal node *astigmatism*. Now the remaining node to examine is the root of the tree. The estimated error rate of the root is

$$N = 24, \qquad f = 9, \qquad q = (0.375 + 0.010 + 0.069)/1.020 = 0.445$$

The estimated error rate of child *none* is

$$N = 12, \qquad f = 0, \qquad q = z^2/(N + z^2) = 0.476/(12 + 0.476) = 0.038$$

The average estimated error rate of the children of the root is

$$q = 12/24 \times 0.038 + 12/24 \times 0.516 = 0.277$$

This is less than the parent's estimated rate. Therefore, we keep the root node.

In the post-pruning process above, we used the one-tailed confidence level $c = 0.25$ to prune the over-fit branches. The left internal node *age* is replaced by the leaf node *soft*. Other structures of the tree remain the same. The pruned tree has six leaf nodes from which new rules can be derived. Reading directly from the final pruned decision tree, six rules are generated as follows:

1. If *tear production rate* is *reduced*, then the *recommended contact lens* is *none*.
2. If *tear production rate* is *normal* and *astigmatism* is *no*, then the *recommended contact lens* is *soft*.
3. If *tear production rate* is *normal* and *astigmatism* is *yes* and *spectacle prescription* is *myope*, then the *recommended contact lens* is *hard*.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

4. If *tear production rate* is *normal* and *astigmatism* is *yes* and *spectacle prescription* is *hypermetropic* and *age* is *presbyopic*, then the *recommended contact lens* is *none*.
5. If *tear production rate* is *normal* and *astigmatism* is *yes* and *spectacle prescription* is *hypermetropic* and *age* is *young*, then the *recommended contact lens* is *hard*.
6. If *tear production rate* is *normal* and *astigmatism* is *yes* and *spectacle prescription* is *hypermetropic* and *age* is *pre-presbyopic*, then the *recommended contact lens* is *none*.

When we compare the rules generated before and after post-pruning, we can see that the number of rules drops from nine to six, at the expense of the accuracy dropping from 100% to 23/24 = 95.8%. When comparing these rules with those generated by the "separate and conquer" approach of the Induct algorithm, we see that the number of rules is the same in both lists, although the rules in the two lists are not identical.

3.5 PARTIAL DECISION TREE

The partial decision tree approach in classification learning is a mixture of the two previous approaches, "divide and conquer" and "separate and conquer." It uses a "separate and conquer" strategy, i.e., it builds a rule, removes the instances it covers, and recursively creates rules for the remaining instances until none are left. However, the method of creating a rule is different from that of the original "separate and conquer" approach. It uses a "partial decision tree" to induce individual rules, adopted from the "divide and conquer" method. A partial decision tree is built from the remaining dataset. Rules are directly generated out of the partial tree starting from the deepest leaf node in conjunction with all the nodes along the path towards the root. The partial tree is then discarded.

The approach has the efficiency of the "separate and conquer" approach because it excludes the instances used in producing the partial tree from the dataset, preventing those instances from participating in further rule induction. The accuracy of the algorithm is guaranteed by the partial tree induction process as in the "divide and conquer" approach. It may seem to be inefficient to build a different partial tree for each rule and discard it. However, the advantage of avoiding a lengthy and complicated rule-set optimization

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

FIGURE 3.8 Algorithm for building a partial decision tree

```
expand-subset (S){
    (1) choose a test T and use it to split a dataset
        into sub-datasets
    (2) sort the subsets into ascending order according
        to the information values
    (3) while [there is a subset X that has not yet been
              expanded AND all subsets expanded so far
              are leaves]
              expand-subset (X);
    (4) if (all the subsets expanded are leaves AND prun-
           ing is not necessary)
               undo expansion into subset and make node a
               leaf;
} /* the end of expand-subset(s) */
```

process—necessary in both the "separate and conquer" and "divide and conquer" approaches—far outweighs this expense. A partial tree is far less complicated than a full tree. The "separating" mechanism makes the partial tree even simpler for less general partial trees.

The key feature of the partial decision-tree approach is building a partial tree instead of a fully explored decision tree. A partial decision tree is a regular decision tree except that it contains branches with undefined subtrees—nodes that are not leaves. The process of creating a partial decision tree is summarized in Figure 3.8.

It first selects an attribute to test and splits the dataset into subsets, as in the regular decision-tree method. The information gain value is used as the criterion to choose the attribute to test, as in C4.5. Then the subsets are expanded in order of ascending number of bits of information. The reason for this is that a subset with a low number of bits of information is more likely to result in a smaller subtree and therefore to produce a more general rule. The nodes with a higher number of bits of information may never get expanded further for the partial tree. The process of expanding the nodes proceeds recursively until all the children of the lowest-level internal node are leaves. Then, the pruning procedure starts from the internal node. It checks to see if it is better to replace the node with a single leaf. The pruning

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

procedure continues backtracking, going through every internal node and its children that have been expanded.

After the pruning, the algorithm checks the children of lowest-level internal nodes to see if they are all leaves, since pruning causes shrinking into subtrees and may cause a higher-level unexpanded node to become a lowest-level node. If there is at least one unexpanded node, the expanding and pruning process will proceed from the unexpanded internal node with the next highest number of bits of information.

The procedure of expanding and pruning partial decision trees is illustrated in Figure 3.9. In the figure, the ellipses represent the internal nodes, the shaded color is for unexpanded nodes, and the unshaded color is for expanded nodes. The rectangle represents a leaf node. The numbers separated by a dot in the nodes indicate the level number and the sibling number of the level. The following explains the steps in the scenario of the figure in detail:

A. Select an attribute *A* to split dataset *S* into three subsets 2.1, 2.2, and 2.3 using the same method as in C4.5 algorithm.
B. Among the sibling nodes, node 2.2 has the lowest number of bits of information. Choose this node to expand (to split further) in the same way as in step A. Two subsets are created, nodes 3.1 and 3.2.
C. The subset 3.1 has fewer bits of information than the subset 3.2 and is further split into node 4.1 and a leaf node.
D. Between the two children of node 3.1, one node is a leaf node and the other non-leaf node 4.1 is expanded into two other leaf nodes.
E. The process of expanding stops at node 4.1 because both children are leaves. Next, the process of pruning starts at node 4.1 and goes upward. Suppose node 4.1 needs to be pruned and is, in fact, replaced by a leaf node. This results in step E of Figure 3.9.
F. The pruning process goes upward until a higher-level node with no unexpanded children is found. In our example, the highest-level node is node 2.2. At this point, nodes 3.1 and 2.2 are replaced by leaf nodes, as shown in step G of Figure 3.9.
G. The lowest level of the partial decision tree is now level 2. Among the sibling nodes of this level, only one is a leaf node. Therefore, further expanding is needed.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

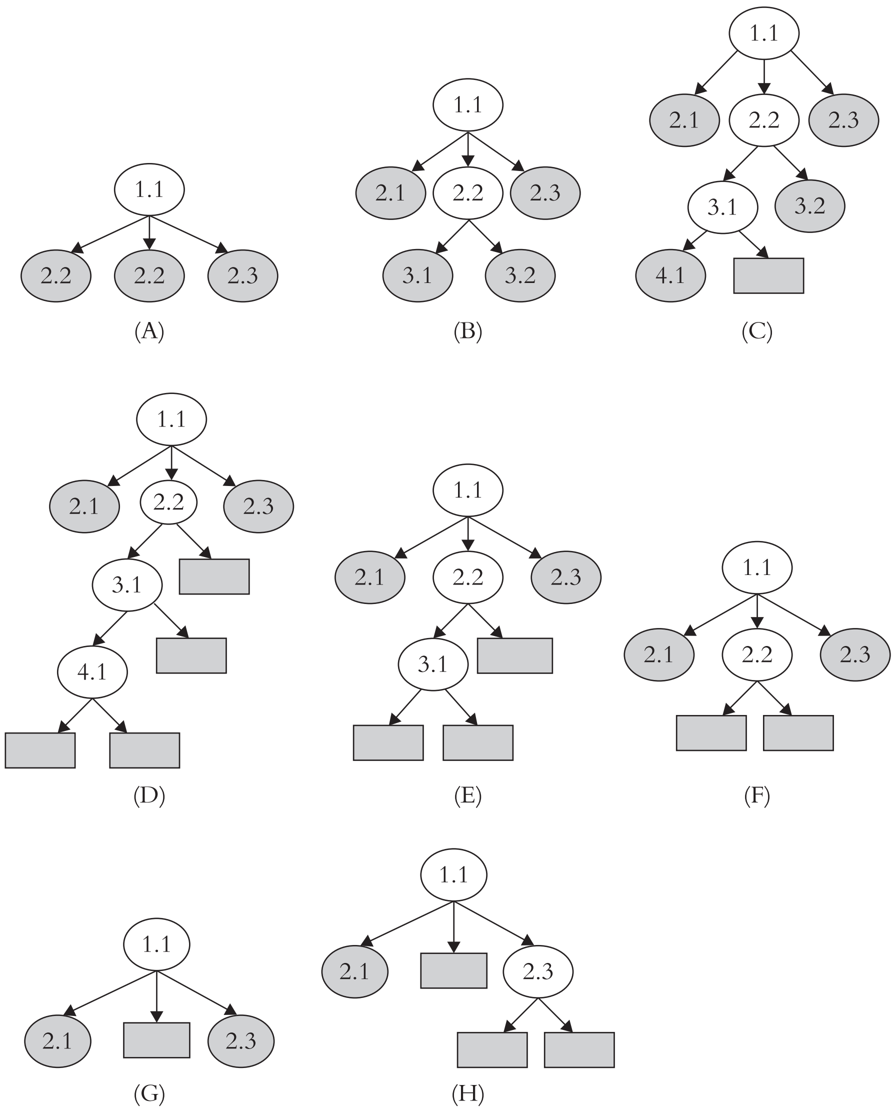

FIGURE 3.9 Expanding and pruning of a partial decision tree Ellipse: internal nodes (shaded ellipse: unexpanded nodes; unshaded ellipse: expanded nodes) Rectangles:leaf nodes.

H. Node 2.3 is expanded because it has fewer bits of information than node 2.1. Both children of node 2.3 are leaves. This triggers the pruning check again. If the pruning is not needed and node 2.3 is the highest-level node with no unexpanded children, the pruning stops here and this subtree is saved. At this point, we check to see if node expansion is necessary. Since

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

the lowest-level nodes are all leaves, no more expansion is needed. This tree (step H in Figure 3.9) is the final partial decision tree we get. A single rule can be derived by following the tree down to a lowest-level leaf.

This method is simple and surprisingly produces rule sets that compare favorably to those produced by C4.5 and C5.0 and are more accurate than those produced by RIPPER [Frank 1998]. Hence, the partial decision-tree approach outperforms (in accuracy) the RIPPER "separate and conquer" type system. The experiment also shows that the partial decision-tree approach greatly outperforms C4.5 in speed when the dataset is less noisy because perfect data prevents the algorithm from pruning partial trees. As the degree of noise increases, the speed gain of the partial-tree approach decreases. The overall time complexity of the partial-tree approach is $O(a \times n \log n)$, where a is the number of attributes and n is the number of instances in the dataset. The complexity of RIPPER is $O(a \times n \log^2 n)$.

Next, we use the contact lens database in Table 3.2 to illustrate how the partial decision-tree approach is applied to induce knowledge rules. The following steps illustrate the procedure of deriving the knowledge rules from the database:

1. At the root of the tree, we select the attribute *tear production rate*, which has the largest information gain ratio, as was shown in Figure 3.6.
2. Next, we determine which child is to be expanded. Since the left node *none* is a leaf node, there is no need to expand. The right child *astigmatism* needs to be expanded and split into two branches, *no* and *yes*.
3. Since both branches are internal nodes, we expand the *no* branch, which has bit of information 0.65, which is less than the *yes* branch, 0.918.
4. The *no* branch leads to the attribute *age*, which has the largest information gain ratio and is split into three children.
5. Among the three children, the branch *presbyopic* leads to an internal node and needs to be expanded.
6. The branch *presbyopic* is split into two leaves through the attribute *prescription*. The expanding process stops here. Next, we start the post-pruning on the partial tree, going upward from the bottom.
7. Using the same technique as in C4.5, the internal node *age* is replaced with a leaf node *soft*.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

8. Now, the node *astigmatism* becomes the lowest-level internal node. Its right branch is still not a leaf node and needs to be expanded.
9. The branch *astigmatism is yes* is split into two branches *hypermetropic* and *myopic* by the attribute *prescription*. The branch *hypermetropic* leads to the attribute *age* and splits into three leaf nodes.
10. The expanding procedure stops at this point and the post-pruning starts again toward the root of the partial tree.
11. No internal node is replaced in this round of post-pruning. The final partial tree is shown in Figure 3.10.
12. Six rules are generated from the partial tree.
13. The instances covered by the partial tree are excluded from the dataset. Since the partial tree covered all the instances of the dataset, no more partial trees need to be created.

Our example demonstrates that when the dataset doesn't contain any noisy data, the partial tree created by using the partial decision-tree approach is in fact the same as the decision tree created by the C4.5 algorithm.

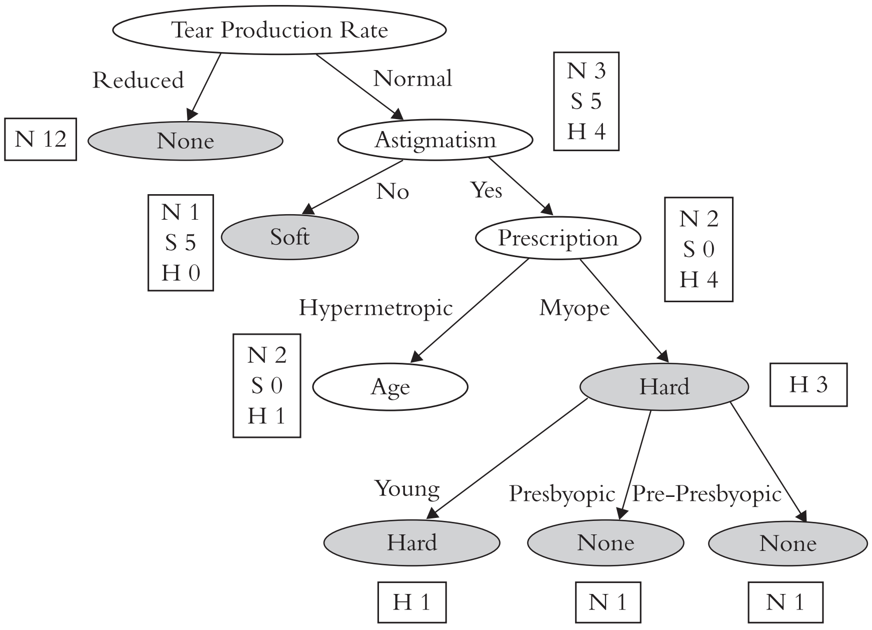

FIGURE 3.10 Partial decision tree derived from contact lens dataset

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

3.6 CHAPTER SUMMARY

In this chapter, several algorithms for classification methods in data mining were presented. A general introduction to classification and its role in data-mining applications was also given. Every approach to classification learning has its advantages and disadvantages. The beauty of the "separate and conquer" strategy is its simplicity, while the advantage of the "divide and conquer" approach is its accuracy. The partial decision-tree approach combines the benefits of both methods.

The Prism, Induct, REP, IREP, and RIPPER algorithms, have been used to elaborate the separate-and-conquer approach. As for the divide-and-conquer approach, methods such as ID3, C4.5, and C5.0 have been used to illustrate methods and applications using practical examples. The partial decision-tree approach in classification learning, a mixture of the "divide and conquer" and "separate and conquer" approaches, was discussed in depth with examples. The partial decision-tree approach uses a "separate and conquer" strategy to build a rule, removes the instances it covers, and recursively creates rules for the remaining instances until none are left. On the other hand, it is also used to induce individual rules, adopted from the "divide and conquer" method.

3.7 EXERCISES

1. Table 3.7 contains data on the effects of dietary fiber. Twelve female subjects were fed a controlled diet. Before each meal they ate crackers containing either bran fiber, gum fiber, a combination of both, or no fiber (control). Their caloric intake was monitored. Subjects reported any gastric or other problems. The table contains the following attributes:
   - Cracker: Type of fiber in the cracker
   - Diet: One of four diets (type of cracker)
   - Subject: Identification for each of the 12 subjects
   - Digested: Digested calories (difference between caloric intake and calories passed through system)
   - Bloat: Degree of bloating and flatulence reported by the subjects

   a. Use the Prism approach to generate decision rules from Table 3.7. Choose "bloat" as the class variable.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

TABLE 3.7 High-fiber diet plan

Cracker	Diet	Subject	Digested	Bloat
Control	1	3	1772.84	None
Bran	4	3	1752.63	Low
Combo	3	9	2121.97	Med
Gum	2	4	2558.61	High
Gum	2	1	2026.91	Med
Bran	4	1	2047.42	Low
Combo	3	1	2254.75	Low
Control	1	1	2353.21	Med
Combo	3	2	2153.36	None
Gum	2	2	2331.19	None
Bran	4	2	2547.77	None
Control	1	2	2591.12	None
Gum	2	3	2012.36	Low
Combo	3	3	1956.18	Low
Combo	3	4	2025.97	None
Bran	4	4	1669.12	None
Control	1	4	2452.73	None
Bran	4	5	2207.37	Low
Gum	2	5	1944.48	Med
Control	1	5	1927.68	Low
Combo	3	5	2190.1	High
Control	1	6	1635.28	None
Combo	3	6	1693.35	Low
Bran	4	6	1707.34	Low
Gum	2	6	1871.95	High
Gum	2	7	2245.03	None
Combo	3	7	2436.79	Low
Control	1	7	2667.14	Low
Bran	4	7	2766.86	None
Bran	4	8	2279.82	None

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

TABLE 3.7 High-fiber diet plan (*continued*)

Cracker	Diet	Subject	Digested	Bloat
Combo	3	8	1844.77	High
Gum	2	8	2002.73	High
Control	1	8	2220.22	Med
Control	1	9	1888.29	Low
Gum	2	9	1804.27	High
Bran	4	9	2293.27	Med
Bran	4	10	2357.4	None
Control	1	10	2359.9	None
Combo	3	10	2292.46	Low
Gum	2	10	2433.46	High
Gum	2	11	1681.86	Low
Control	1	11	1902.75	None
Bran	4	11	2003.16	None
Combo	3	11	2137.12	Med
Combo	3	12	2203.07	Med
Control	1	12	2125.39	Low
Gum	2	12	2166.77	Med
Bran	4	12	2287.52	None

b. List all possible decision rules that can be generated from Table 3.7 by following the Induct approach.

c. Use the IREP algorithm presented in this chapter to generate decision rules for the high-fiber diet plan database in Table 3.7.

2. Table 3.8 contains data that were collected on the genus of flea beetle *Chaetocnema*, which contains three species: *concinna* (Con), *heikertingeri* (Hei), and *heptapotamica* (Hep). Measurements were made on the width and angle of the aedeagus of each beetle. The table contains the following attributes:

- Width: The maximal width of the aedeagus in the forepart (in microns)
- Angle: The front angle of the aedeagus (1 unit = 7.5 degrees)
- Species: The species of flea beetle from the genus *Chaetocnema*

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

TABLE 3.8 Flea beetles

Width	Angle	Species	Width	Angle	Species
150	15	Con	137	9	Hep
147	13	Con	141	11	Hep
144	14	Con	138	9	Hep
144	16	Con	143	9	Hep
153	13	Con	142	11	Hep
140	15	Con	144	10	Hep
151	14	Con	138	10	Hep
143	14	Con	140	10	Hep
144	14	Con	130	9	Hep
142	15	Con	137	11	Hep
141	13	Con	137	10	Hep
150	15	Con	136	9	Hep
148	13	Con	140	10	Hep
154	15	Con	128	14	Hei
147	14	Con	129	14	Hei
137	14	Con	124	13	Hei
134	15	Con	129	14	Hei
157	14	Con	145	8	Hep
149	13	Con	140	11	Hep
147	13	Con	140	11	Hep
148	14	Con	131	10	Hep
120	14	Hei	139	11	Hep

The goal of the original study was to form a classification rule to distinguish the three species. Answer the following questions:

a. Use the ID3 strategy to generate decision rules from Table 3.8.
b. Generate classification rules from the table using C4.5 strategies.
c. Apply the partial decision-tree approach to generate all possible classification rules.
d. Compare and contrast all classification rules generated in (a), (b), and (c).

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

TABLE 3.8 Flea beetles (*continued*)

Width	Angle	Species	Width	Angle	Species
123	16	Hei	139	10	Hep
130	14	Hei	136	12	Hep
131	16	Hei	129	11	Hep
116	16	Hei	140	10	Hep
122	15	Hei	129	14	Hei
127	15	Hei	130	13	Hei
132	16	Hei	129	13	Hei
125	14	Hei	122	12	Hei
119	13	Hei	129	15	Hei
122	13	Hei	124	15	Hei
120	15	Hei	120	13	Hei
119	14	Hei	119	16	Hei
123	15	Hei	119	14	Hei
125	15	Hei	133	13	Hei
125	14	Hei	121	15	Hei

3. Table 3.9 contains data about 40 sampled elementary school students in terms of the following nine attributes:
   - Gender: Boy or girl
   - Race: White or other
   - Region: Rural or urban
   - School: Dooley or East
   - Grades: Rank of making good grades (1 = most important, 4 = least important)
   - Sports: Rank of being good at sports (1 = most important, 4 = least important)
   - Looks: Rank of being handsome/pretty (1 = most important, 4 = least important)
   - Money: Rank of having lots of money (1 = most important, 4 = least important)
   - Goal: Student's choice of personal goals (make good grades, be popular, be good at sports)

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

Assuming that the goal of the study is to generate classification rules to determine the role of the personal factors that affect students' perception about educational objectives, answer the following questions:

a. Use the ID3 strategy to generate decision rules from Table 3.9.
b. Generate classification rules from the table using C4.5 strategies.
c. Apply the partial decision-tree approach to generate all possible classification rules.
d. Compare and contrast all the classification rules generated in (a), (b), and (c) in terms of accuracy.

TABLE 3.9 Educational objectives of elementary school kids

Gender	Race	Region	School	Goal	Grades	Sports	Looks	Money
Boy	White	Rural	Dooley	Grades	4	1	2	3
Boy	Other	Rural	Dooley	Popular	4	3	2	1
Boy	White	Rural	Dooley	Sports	3	1	2	4
Boy	White	Rural	Dooley	Grades	1	2	3	4
Boy	White	Rural	Dooley	Sports	4	1	2	3
Boy	White	Rural	Dooley	Popular	4	2	1	3
Boy	White	Rural	Dooley	Sports	4	2	1	3
Boy	White	Rural	Dooley	Grades	4	3	1	2
Boy	White	Rural	Dooley	Sports	3	1	4	2
Boy	White	Rural	Dooley	Popular	4	2	1	3
Boy	White	Rural	Dooley	Sports	3	1	4	2
Girl	Other	Urban	East	Grades	1	2	3	4
Girl	White	Urban	East	Popular	4	3	1	2
Girl	White	Urban	East	Grades	1	3	2	4
Girl	Other	Urban	East	Sports	4	1	2	3
Girl	White	Urban	East	Grades	2	1	3	4
Girl	Other	Urban	East	Grades	2	4	1	3
Girl	White	Urban	East	Grades	3	2	1	4
Girl	White	Urban	East	Grades	3	2	1	4
Girl	White	Urban	East	Grades	1	2	3	4
Girl	White	Urban	East	Popular	2	1	3	4

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

TABLE 3.9 Educational objectives of elementary school kids (*continued*)

Gender	Race	Region	School	Goal	Grades	Sports	Looks	Money
Girl	White	Urban	East	Sports	1	3	2	4
Girl	White	Urban	East	Grades	1	2	3	4
Girl	White	Urban	East	Grades	1	2	3	4
Girl	White	Urban	East	Grades	4	3	1	2
Girl	White	Urban	East	Grades	1	2	4	3
Boy	White	Urban	East	Grades	3	1	2	4
Boy	White	Urban	East	Popular	1	3	2	4
Boy	White	Urban	East	Grades	3	1	4	2
Boy	White	Urban	East	Sports	1	2	4	3
Boy	White	Urban	East	Grades	3	1	2	4
Boy	White	Urban	East	Sports	2	1	4	3
Boy	White	Urban	East	Sports	2	3	1	4
Boy	White	Urban	East	Grades	2	1	4	3
Boy	Other	Urban	East	Grades	4	1	3	2
Boy	White	Urban	East	Grades	2	1	4	3
Boy	White	Urban	East	Grades	2	1	3	4
Girl	White	Urban	East	Grades	4	2	1	3
Girl	White	Urban	East	Grades	1	2	3	4
Girl	White	Urban	East	Grades	2	1	4	3

4. Table 3.10 contains the data from a statement by Texaco, Inc. to the Air and Water Pollution Subcommittee of the Senate Public Works Committee on June 26, 1973. Mr. John McKinley, President of Texaco, cited an automobile filter developed by Associated Octel Company as effective in reducing pollution. However, questions had been raised about the effects of filters on vehicle performance, fuel consumption, exhaust gas back pressure, and silencing. For the last question, he referred to the data shown in Table 3.10 as evidence that the silencing properties of the Octel filter were at least equal to those of standard silencers. The table contains the data of 36 cases with the following four attributes:
   - Noise: Noise level reading (in decibels)
   - Size: Vehicle size (1 = small, 2 = medium, 3 = large)

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

TABLE 3.10 Auto pollution filter noise

Noise	Size	Type	Side	Noise	Size	Type	Side
810	1	1	1	770	3	1	2
820	1	1	1	820	1	2	1
820	1	1	1	820	1	2	1
840	2	1	1	820	1	2	1
840	2	1	1	820	2	2	1
845	2	1	1	820	2	2	1
785	3	1	1	825	2	2	1
790	3	1	1	775	3	2	1
785	3	1	1	775	3	2	1
835	1	1	2	775	3	2	1
835	1	1	2	825	1	2	2
835	1	1	2	825	1	2	2
845	2	1	2	825	1	2	2
855	2	1	2	815	2	2	2
850	2	1	2	825	2	2	2
760	3	1	2	760	3	1	2
825	2	2	2	760	3	2	2
770	3	2	2	765	3	2	2

- Type: Filter type (1 = standard silencer, 2 = Octel filter)
- Side: Filter location (1 = right side, 2 = left side of car)

Answer the following questions to justify and support the claims made by Texaco by generating the appropriate classification rules:

a. Use the Prism approach to generate decision rules from Table 3.10. Choose "Noise" as the class variable.
b. List all possible decision rules that can be generated from Table 3.10 by following the Induct approach.
c. Apply the IREP algorithm presented in the chapter to the auto pollution filter noise database in Table 3.10 to generate the decision rules.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

3.8 SELECTED BIBLIOGRAPHIC NOTES

General ideas on classification rules and decision trees described in Section 3.2 are from [Baralis 2004], [Charu 2002], and [Deshpande 2002]. Various application of classification methods are described in [Charu 2004], [Diamantini 2000], [Islam 2004], [Lesh 1999], [Lutu 2002], and [Zaki 2003]. [Frank 1999b] and [Meretakis 2000] deal with classification on text databases, whereas [Shintani 1998] and [Yu 2003] present the efficient application of classification on large datasets using hierarchical properties.

The detailed descriptions of the ID3 algorithm in Section 3.4.1 were mainly taken from [Quinlan 1979], [Quinlan 1983], and [Quinlan 1986]. The contact lens example, which was originally taken from the world of ophthalmic optics, was adopted from [Cendrowska 1987]. The Prism algorithm in Section 3.3.1, a rule induction method based on ID3, was summarized from [Cendrowska 1987], whereas the Induct algorithm, which is an extension of Prism used to deal with noisy data, was described in Section 3.3.2 using methods provided in [Gaines 1995].

[Pagallo 1990], [Brunk 1991], [Richards 1998], and [Cohen 1995] discuss the errors resulting from favoring over-fitted rules in Induct. [Furnkranz 1994] and [Cohen 1995] propose REP, IREP, and RIPPER to correct this problem. Furthermore, [Cohen 1995] claims that RIPPER produces rules that are more accurate than REP and IREP and competitive with those of C4.5. The detailed discussion in Section 3.3.3 was taken from these references.

C4.5 uses the same basic strategy as ID3 with gain ratio as an added feature for the attribute selection criteria, whereas C5.0 is an extended version of C4.5 that shows improved performance. The description given in Section 3.4.2 for C4.5 and C5.0 was taken from [Quinlan 1993] and [Quinlan 1997]. [Mingers 1989] deals with the problem of overcompensating for gain ratio by preferring attributes with unevenly divided sub-datasets. The solution to this overcompensation problem was proposed in [Frank 1998] and [Frank 1999a]. The post-pruning method adopted by Quinlan for C4.5 was used in Cart and can be found in [Breiman 1984]. The MDL (Minimum Description Length) heuristic used in C4.5 to optimize the rules in a rule subset was described in [Rissanen 1983a] and [Rissanen 1983b]. C4.5 also adopted a simulated annealing method to find the near-best rule subset in a very large rule set, which is more computationally effective and can produce more satisfactory results. A detailed description of this method can be found in [Press 1988].

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

The partial decision-tree approach described in Section 3.5 was primarily based on the approach proposed in [Frank 1998]. [Skomorokhov 2000] dealt with classification trees in APL, whereas [Buja 2001] focused on tree-based regression and classification. [Sattler 2001], [Ankerst 1999], and [Ding 2002] discuss the decision-tree classification method, whereas [Stern 2004] deals with a classification tree analysis. Application of the data classification method to fraud detection is found in [Phua 2004] and [Bonchi 1999].

■ 3.9 CHAPTER BIBLIOGRAPHY

[Ankerst 1999] Mihael Ankerst, Christian Elsen, Martin Ester, and Hans-Peter Kriegel: "Visual classification: an interactive approach to decision tree construction," *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 1999.

[Baralis 2004] Elena Baralis, Silvia Chiusano, and Paolo Garza: "Data mining (DM): On support thresholds in associative classification," *Proceedings of the 2004 ACM Symposium on Applied Computing*, March 2004.

[Bonchi 1999] F. Bonchi, F. Giannotti, G. Mainetto, and D. Pedreschi: "A Classification-based Methodology for Planning Audit Strategies in Fraud Detection," *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999.

[Breiman 1984] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone: *Classification and Regression Trees*. Belmont, CA: Wadsworth International Group, 1984.

[Brunk 1991] Clifford Brunk, and Michael Pazzani: "Noise-tolerant relational concept learning algorithms," *Proceedings of the Eighth International Workshop on Machine Learning*, Ithaca, New York Morgan Kaufmann, 1991.

[Buja 2001] Andreas Buja and Yung-Seop Lee: "Data mining criteria for tree-based regression and classification," *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 2001.

[Cendrowska 1987] Jadzia Cendrowska: "PRISM: An Algorithm for Inducing Modular Rules," *International Journal of Human–Computer Studies*, 27(4), 349–370, Academic Press, 1987.

[Charu 2002] Charu C. Aggarwal: "Sequences and strings: On effective classification of strings with wavelets," *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, July 2002.

[Charu 2004] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. "On demand classification of data streams," *Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 2004.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

[Cohen 1995] W. Cohen: "Fast effective rule induction," *Proceedings of the 12th International Conference on Machine Learning,* 115–123, Morgan Kaufmann, 1995.

[Deshpande 2002] Mukund Deshpande and George Karypis: "Classification: Using conjunction of attribute values for classification," *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, November 2002.

[Diamantini 2000] Claudia Diamantini and Maurizio Panti: "An efficient and scalable data compression approach to classification," *ACM SIGKDD Explorations Newsletter*, 2(2), August 2004.

[Ding 2002] Qiang Ding, Qin Ding, and William Perrizo: "Database and digital library technologies: Decision tree classification of spatial data streams using Peano Count Trees," *Proceedings of the 2002 ACM Symposium on Applied Computing*, March 2002.

[Frank 1998] Eibe Frank and Ian H. Witten: "Generating Accurate Rule Sets Without Global Optimization," *15th International Conference on Machine Learning*, 144–151, Morgan Kaufmann, 1998.

[Frank 1999a] Eibe Frank and Ian H. Witten: "Making Better Use of Global Discretization," *16th International Conference on Machine Learning*, 115–123, Morgan Kaufmann, 1999.

[Frank 1999b] Eibe Frank, Gordon W. Paynter, Ian H. Witten: "Domain-Specific Keyphrase Extraction," *Proceedings of the 16th International Joint Conference on Articial Intelligence (IJCAI99)*, Stockholm, 1999.

[Furnkranz 1994] J. Furnkranz and G. Widmer: "Incremental Reduced Error Pruning (IREP)," in W. Cohen and H. Hirsh (eds.), *Proceedings of the 11th International Conference on Machine Learning (ML-94),* 70–77, Morgan Kaufmann, 1994.

[Gaines 1995] B. R. Gaines and P. Compton: "Induction of Ripple-Down Rules Applied to Modeling Large Databases," *Journal of Intelligent Information Systems*, 5, 211–228, Kluwer, 1995.

[Islam 2004] Md. Zahidul Islam and Ljiljana Brankovic: "A framework for privacy preserving classification in data mining," *Proceedings of the Second Workshop on Australasian Information Security, Data Mining and Web Intelligence, and Software Internationalization*, Volume 32, January 2004.

[Lesh 1999] Neal Lesh, Mohammed J. Zaki, and Mitsunori Ogihara: "Mining features for sequence classification," *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 1999.

[Lutu 2002] Patricia E. N. Lutu: "Data Mining: An integrated approach for scaling up classification and prediction algorithms for data mining," *Proceedings of the 2002 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement through Technology*, September 2002.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

[Meretakis 2000] Dimitris Meretakis, Dimitris Fragoudis, Hongjun Lu, and Spiros Likothanassis: "Scalable association-based text classification," *Proceedings of the Ninth International Conference on Information and Knowledge Management*, November 2000.

[Mingers 1989] J. Mingers: "An Empirical Comparison of Pruning Methods for Decision Tree Induction." *Machine Learning*, 4(2):227–243, November 1989.

[Pagallo 1990] G. Pagallo and D. Haussler: "Boolean Feature Discovery in Empirical Learning," *Machine Learning*, Vol. 5, No. 1, 71–99, 1990.

[Phua 2004] Clifton Phua, Damminda Alahakoon, and Vincent Lee: "Minority report in fraud detection: Classification of skewed data," *ACM SIGKDD Explorations Newsletter*, 6(1), June 2004.

[Press 1988] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling: *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, Cambridge, MA, 1988.

[Quinlan 1979] J. R. Quinlan: "Discovering Rules by Induction from Large Collections of Examples," in D. Mitchie (ed.), *Expert Systems in The Microelectronic Age*, pp. 168–201, 1979.

[Quinlan 1983] J. R. Quinlan: "Learning Efficient Classification Procedures and Their Application to Chess Endgames," in R. Michalski, J. Carbonell, and T. Mitchell (eds.), *Machine Learning: An Artificial Intelligence Approach*, Vol.1, pp. 463–482, Morgan Kaufmann, 1983.

[Quinlan 1986] J. R. Quinlan: "Induction of Decision Trees," *Machine Learning*, Vol. 1, No. 1, pp. 81–106, Kluwer, 1986.

[Quinlan 1993] J. R. Quinlan: *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.

[Quinlan 1997] J. R. Quinlan: "Learning Logical Definitions from Relations," *Machine Learning*, Vol. 5, pp. 239–266, 1997.

[Richards 1998] D. Richards and P. Compton: "Taking Up the Situated Cognition Challenge with Ripple Down Rules," *International Journal of Human–Computer Studies*, Vol. 49, No. 6, pp. 895–926, 1998.

[Rissanen 1983a] J. Rissanen: "A Universal Prior for Integers and Estimation by Minimum Description Length," *Annals of Statistics*, Vol. 11, No. 2, pp. 416–431, 1983.

[Rissanen 1983b] J. Rissanen: "A Universal Data Compression System," *IEEE Transactions on Information Theory*, Vol. 29, No. 5, pp. 656–664, 1983.

[Sattler 2001] Kai-Uwe. Sattler and O. Dunemann: "SQL Database Primitives for Decision Tree Classifiers," *Proceedings of the 10th International Conference on Information and Knowledge Management*, 2001.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

[Shintani 1998] T. Shintani and M. Kitsuregawa: "Parallel Mining Algorithms for Generalized Association Rules with Classification Hierarchy," *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, 1998.

[Skomorokhov 2000] A. Skomorokhov and V. Kutinsky: "Classification Trees in APL: Implementation and Application," *Proceedings of the 2001 Conference on APL: An Arrays Odyssey*, Vol. 31, No. 2, 2000.

[Stern 2004] S. E. Stern, S. Gregor, M. A. Martin, S. Goode, and J. Rolfe: "A Classification Tree Analysis of Broadband Adoption in Australian Households," *Proceedings of the 6th International Conference on Electronic Commerce*, 2004.

[Yu 2003] H. Yu, J. Yang, and J. Han: "Classifying Large Datasets Using SVMs with Hierarchical Clusters," *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

[Zaki 2003] M. J. Zaki and C. C. Aggarwal: "XRules: An Effective Structural Classifier for XML Data," *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.