Chapter 3
Classification and Regression Trees (CART)

3.1 Introduction

Recall that in stagewise regression, the results of a stage are fixed no matter what happens in subsequent stages. Earlier stages are not re-visited. Forward stepwise regression is not a stagewise procedure because all of the included regression coefficients are re-estimated as each new regressor is added to the model. In a similar fashion for backwards stepwise regression, all of the remaining regression coefficients are re-estimated as each additional regressor is dropped from the model.

If you look under the hood, conventional classification and regression trees (CART), also called decision trees in computer science, essentially a form of stagewise regression with predictors that are indicator variables. CART output is typically displayed in a tree-like structure, which accounts for how the technique is named. A defining feature of CART is the way in which predictors are transformed and selected.

Suppose one has a single quantitative response variable and several predictors. There is interest in $\hat{Y}|X$. The immediate task is to find the single best binary predictor from among a set of predictors, all of which may be numerical. To do this, two kinds of searches are undertaken. First, for each predictor, all possible binary splits of the predictor values are considered. For example, if the predictor is age in years, and there are age-values of 21 through 24, all possible splits maintaining order would be 21 versus 22–24, 21–22 versus 23–24, and 21–23 versus 24.

Ordinal predictors can be handled in the same fashion. For example, the possible answers to a questionnaire item might be "strongly agree," "agree," "can't say," "disagree," and "strongly disagree." Then, all possible binary splits of the data are considered with the order maintained i.e. in strength of agreement. Unlike conventional regression, ordinal predictors pose no special problems.

The original version of this chapter was revised: See the "Chapter Note" section at the end of this chapter for details. The erratum to this chapter is available at https://doi.org/10.1007/978-3-319-44048-4_10.

R.A. Berk, *Statistical Learning from a Regression Perspective*,
Springer Texts in Statistics, DOI 10.1007/978-3-319-44048-4_3

Closely related reasoning can be applied when a predictor is categorical. For instance, if the predictor is marital status with categories never married, married, and divorced, all possible splits would be never married versus married and divorced, married versus never married and divorced, and divorced versus never married and married. For categorical variables, there is no order to maintain.

How is the "best split" for each predictor defined? For quantitative response variables, the baseline is the response variable sum of squares. Each possible binary split of each predictor implies a different two-way partitioning of the data. For example, one partition might include all individuals under 25 years of age, and the other partition would then include all individuals 25 years of age or older. The response variable sum of squares is computed separately within each partition and added together. That sum will be equal to or less than the sum of squares for the response variable before the partitioning. The "best" split for each predictor is defined as the split that reduces the sum of squares the most.

With the best split of each predictor determined, the best split *overall* is determined as the second step. That is, the best split for each predictor is compared by the reduction in the sum of squares. The predictor with the largest reduction wins the competition. It is the predictor that when optimally split, leads to the greatest reduction in the sum of squares.

With the two-step search completed, the winning split is used to subset the data. In other words, the best split for the best predictor defines two subsets. For example, if the best split were to be 21–22 versus 23–24 years of age, all individuals 21–22 would form one subset, and all individuals 23–24 would form the other subset.

There are now two partitions of the original data, defined by best split within and between the predictors. Next, the same two-step procedure is applied to each partition separately; the best split within and between predictors for each subset is found. This leads to four partitions of the data, and once again, the two-step search procedure is undertaken separately for each. The recursive process can continue until there is no meaningful reduction in the sum of squares of the response variable. Then, the results are conventionally displayed as an inverted tree: roots at the top and canopy at the bottom.

As addressed shortly, the recursive partitioning results can be represented within a linear basis expansion framework. The basis functions are indicator variables defined by the best splits. With these determined, a regression of the response on the basis functions yields regression coefficients and fit statistics as usual. In practice, there is no need to translate the partitioning into a regression model; the partitioning results stand on their own as a regression analysis. Because the partitions are determined empirically from the data, the partitioning process introduces a form of model selection. This creates complications for any level II analysis.

The two-step search procedure is easily generalized to categorical response variables, but other performance measures are used rather than the sum of squares. Among the options to be discussed later is the deviance. The upside down tree display of key output remains.

There is a remarkably large number of tree-based statistical methods (Loh 2014). In this chapter, we consider Classification and Regression Trees (CART) introduced

by Breiman, Friedman, Olshen, and Stone in 1984. CART has been in use for about 30 years (Breiman et al. 1984) and remains a popular data analysis tool. We will focus on CART as it has traditionally been implemented. There are some interesting refinements and extensions (Chaudhuri et al. 1995; Lee 2005; Chipman et al. 1998; Loh 2014; Su et al. 2004; Choi et al. 2005; Hothorn et al. 2006; Zeileis et al. 2008), and even some major reformulations (Grubinger et al. 2014). There are also CART-like procedures such as CHAID (Kass 1980) and C5.0 (Quinlan 1993), which has a computer science lineage. A discussion of these variants would take us some distance from the approach emphasized in here in part because they treat CART primarily as a stand-alone data analysis tool. CART sometimes can be an effective stand-alone procedure as well, but more important for our purposes, it has become an integral component of statistical learning algorithms discussed in subsequent chapters. A discussion of CART provides an essential foundation for understanding those algorithms.

Chapter 2 was devoted almost entirely to quantitative response variables. Equal time and more is now given to categorical, and especially binary, response variables. As noted earlier, procedures that assign observations to classes are sometimes called "classifiers." When CART is used with categorical response variables, it is an example of a classifier. One grows a classification tree.

Categorical response variables introduce a number of issues that either do not apply to quantitative response variables, or apply only at a high level of abstraction. We now need to get this material on the table in part because it applies to classifiers in addition to CART. We also emphasize again the differences between level I, level II and level III regression analyses and remind readers of the critical difference between explanation and forecasting.

This is a somewhat plodding, tedious chapter. An effort has been made to include only the material that is really needed. But that's a lot, and it is probably necessary to slog through it all.

3.2 The Basic Ideas

We begin with conceptual overview of the CART computational machinery. Mathematical details are provided later. For a binary response variable coded "A" or "B," and predictors X and Z, Fig. 3.1 is the three-dimensional scatterplot illustrating a simple classification problem as it might be attacked by CART. CART's algorithm is called "greedy" because it searches for the best outcome without looking back to past splits or forward to future splits. The algorithm lives only in the present.

The vertical red line at, say, $Z = 3$ produces the first partition. It represents the best split in a competition between all possible splits of X or Z. The A values tend to be concentrated to the left, and the B values tend to be concentrated to the right. The green horizontal line at $X = 5$ produces the subsequent partition of the left subset. It represents the best split of the left partition. The upper left corner is now homogeneous in A. This is an ideal outcome. The yellow horizontal line at $X = -2$

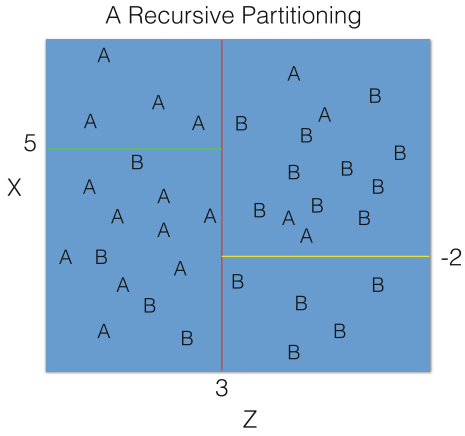

Fig. 3.1 A recursive partitioning for a binary response variable and predictors X and Z (The response is coded A or B. The *red line* shows the first partition. The *green* and *yellow lines* show the next two partitions.)

produces the best subsequent split of the right partition. The lower right corner is now homogeneous in B. This too is an ideal outcome. In principle, the lower left partition and the upper right partition would be further subdivided.

Figure 3.1 makes clear that CART constructs partitions with a series of straight-line boundaries perpendicular to the axis of the predictor being used. These may seem like serious constraints on performance. Why linear? Why perpendicular? They are simple to work with and can perform very well in practice.

The values at which the partitioning is done matter. For example, Fig. 3.1 reveals that cases with $Z \leq 3$ and $X > 5$ are always A. Likewise, cases with $Z > 3$ and $X \leq -2$ are always B. We are able describe all four conditional distributions of the binary response variable conditioning on the four partitions of X and Z. Within each partition, the proportion of A-values (or B-values) might be a useful summary statistic for the binary outcome. How do these proportions vary over the different partitions? We are still doing a regression analysis.

3.2.1 *Tree Diagrams for Understanding Conditional Relationships*

CART partitioning is often shown as an inverted tree. A tree visualization allows the data analyst to see how the data partitions were constructed and consider the conditional relationships implied. Explanation can be in play within a level I regression analysis.

Figure 3.2 is a simple, schematic illustration of an inverted tree. The full dataset is contained in the root node. The data are then broken into two mutually exclusive pieces. Cases with $X > C_1$ go to the right, and cases with $X \leq C_1$ go to the left. The latter are then in terminal node 1, which is not subject to any more subsetting; no meaningful improvements in fit can be made. The former are in an internal node that can be usefully subdivided further, and the internal node is partitioned again. Observations with $Z > C_2$ go to the right and into terminal node 3. Observations with $Z \leq C_2$ go to the left and into terminal node 2. Further subsetting is not productive.

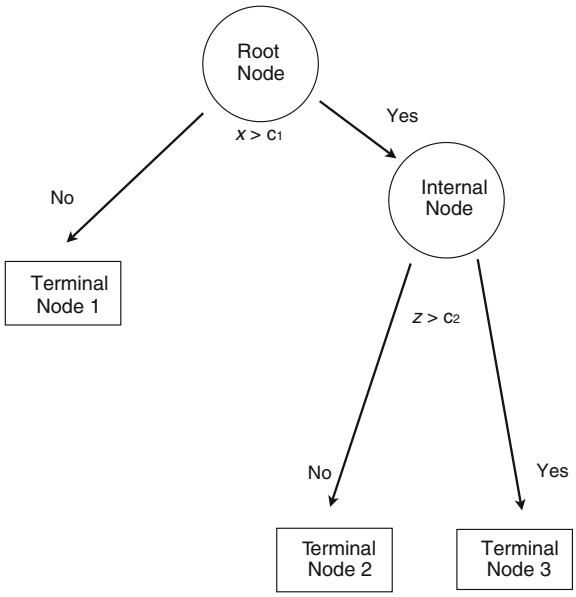

Fig. 3.2 A simple CART tree structure

In this case, all splits beyond the initial split of the root node imply, in regression language, interaction effects. The right partition imposed at the internal node only includes observations with X-values that are greater than C_1. Consequently, the impact of Z depends on observations with $X > C_1$, which is an interaction effect.

When there is no natural order to a predictor's values, the partitioning criterion selected is usually represented by the name of the variable along with the values that go to the right (or left, depending on the software) side. For example, if ethnicity is a predictor and there are five ethnicities represented by the letters a though e, the software might represent the partitioning criterion for a given split as *ethnicity* = *ade*. All cases belonging to ethnic groups a, d, and e are being placed in the right-hand partition.

Splits after the initial split do not have to represent interaction effects. If an immediately subsequent partitioning of the data uses the same predictor (with a different breakpoint), the result is an additional step in the step function for that predictor. A more complicated nonlinear function results, but not an interaction effect. In practice, however, most partitions of the data represent interaction effects.

It is easy to translate Fig. 3.2 into linear basis expansions. One just defines all of the terminal nodes with indicator variables, each of which is a function of one or more predictors (including the constant term). Thus,

$$f(X, Z) = \beta_0 + \beta_1[(I(x \le c_1)] + \beta_2[I(x > c_1 \ \& \ z \le c_2)] + \beta_3[I(x > c_1 \ \& \ z > c_2)]. \quad (3.1)$$

One can see the importance of interaction effects whenever two or more predictors are needed to construct the indicator variable. Interaction effects need to be kept in mind when CART tree diagrams are interpreted.

The application of CART is always an opportunity for a level I analysis. For a level II analysis, we once again must treat the data as random realizations from a nature's joint probability distribution. The estimation target is a classification or regression tree, having the same structure as the tree derived from the data, but as a feature of the joint probability distribution. Consequently, CART fitted values can be used as estimates of the corresponding, approximate response surface. The same level II reasoning can be applied to regression equations for the terminal nodes, such Eq. 3.1. But because CART has built in data snooping, the level II opportunities are somewhat limited.

To illustrate these initial ideas, consider passenger data from the sinking of the Titanic.[1] What predictors are associated with those who perished compared to those who survived? Figure 3.3 shows the CART results as an inverted tree. Survived is coded as a 1, and perished is coded as a 0. In each terminal node, the left number is the count of those who perished, and the right number is the count of those who survived. If the majority perished, the node is colored red. If the majority survived, the node is colored blue. Figure 3.4 contains the code.[2]

For this analysis, predictors used include:

1. sex — the gender of the passenger;
2. age — age of the passenger in years;
3. pclass — the passenger's class of passage;
4. sibsp — the number of siblings/spouses aboard; and
5. parch — the number of parents/children aboard.

The first split on sex. Males are sent down the left branch. Females are sent down the right branch. The two subsequent splits for males are at ages of 9.5 years and 2.5 years. For males, there are three terminal nodes. For the older males, 136 survived and 600 did not. Because the majority did not survive, the node is labeled with a 0 for not surviving. For males, only those 2.5 years old or younger were likely to survive (i.e., 24 to 3). For females, the splits are much more complicated but can be considered with the same sort of reasoning. For example, most of the females traveling in first or second class survived, and the terminal node is given a label of 1 accordingly (i.e., 233 to 17). If one takes the cinematic account seriously, the results broadly make sense, and a level I analysis of the data has been undertaken.

It is challenging to make a credible case for a level II analysis. What is the joint probability distribution responsible for the data? It could perhaps be the joint probability distribution for passengers on ocean liners using as a response of whether or not each survived the passage. But much more specificity would be needed. For

[1] Thanks go to Thomas Cason who updated and improved the existing Titanic data frame using the Encyclopedia Titanica.

[2] The procedure *rpart()* is authored by Terry Therneau and Beth Atkinson. The procedure *rpart.plot()* is authored by Stephen Milborrow. Both procedures are superb.

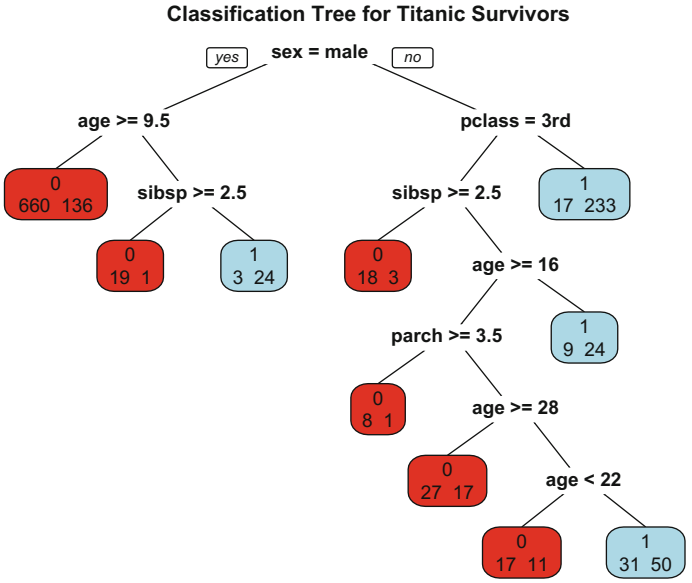

Fig. 3.3 A classification tree for the Titanic data: 1 = Survived, 0 = Perished (In each terminal node, the number who perished and the number who survived are shown, to the *left* and *right* respectively. In *red nodes*, the majority perished. In *blue nodes*, the majority survived. N = 1309)

```
### CART Code
library(PASWR) # To get the data
data(titanic3) # Load the data

library(rpart) # Load the CART library
library(rpart.plot) # Load the fancy plotting library

# Partition the data
out<-rpart(survived~sex+age+pclass+sibsp+parch, data=titanic3,
method="class")
#Plot the tree
prp(out,extra=1,faclen=10,varlen=15,cex=1.2,
    main="Classification Tree for Titanic Survivors",
    box.col=c("red","lightblue")[out$frame$yval])
```

Fig. 3.4 R code for the CART analysis of the Titanic data

example, it is likely that such a distribution would change with new ship technology and better training of captains, crews, and navigators. The sinking of the Titanic itself led to revisions of marine safety regulations governing such things as the number of lifeboats required on board. And even if a reasonable joint probability distribution could be defined, whether passengers perished or survived was not realized independently, conditioning on the available predictors. Indeed, that is part of the reason why the account of the Titanic sinking is so compelling. A much more complicated data realization process is required.

In addition, the partitioning algorithm is clearly an example of concerted data snooping. Therefore, in-sample performance is formally unjustified and potentially very misleading. Test data are required. What would be proper test data for the passengers on the Titanic? Unless there is a good answer, the fallback of cross-validation is also badly compromised. What exactly does an out-of-sample estimate mean?

3.2.2 Classification and Forecasting with CART

There is far more to the output from CART than a tree diagram. Indeed, perhaps the most important output is found in the terminal nodes. Suppose the response variable is binary, and the task is classification. Within each of the terminal nodes, the proportion of "successes" and proportion of "failures" can be calculated. These conditional proportions are often of significant descriptive interest as one kind of fitted values. For example, in Fig. 3.2 if the proportion of successes in terminal node 3 is .70, one can say for cases with $x > c_1$ and $z > c_2$ that the proportion of successes is .70. Analogous statements can be made about the other terminal nodes. For example, the proportion of success for cases with $x \leq c$ (terminal node 1) might be .25. Ideally, terminal node proportions will vary substantially, implying that the partitioning is making important distinctions between different kinds of cases. If one knows for any given case the value of x and the value of z, it really matters for the proportion of successes.

In addition, the conditional proportions can be used to attach class labels to terminal nodes that, in turn, can be assigned to observations. The class labels are a second kind of fitted value. If the majority of observations in a terminal node are As, all of the observations in that partition may be assigned to class A. If the majority of observations in a terminal node are Bs, all of the observations in that partition may be assigned to class B. These labels convey what is most typical in a terminal node and, therefore, is most typical for cases with a particular configuration of indicator variables. If all of the observations in a terminal node need to be placed into a single category, the terminal node class label provides the means. When CART is used in this manner, it is Bayes classifier, applied individually to each terminal node.

Think back to the discussion of logistic regression in Chap. 1. Recall that there were fitted proportions and fitted classes. Cases with fitted proportions that exceeded some threshold were assigned to one of the two classes. Cases with fitted proportions

that did not exceed that threshold, or were equal to it, were assigned to the other class. Much the same is going on within each of the terminal nodes of classification trees, where by the Bayes classifier, the threshold is .50.

But why would anyone care about the class labels? The class labels can be critical if new data are provided that contain the same predictors but with the binary outcome unknown. The labels are a good guess for the unknown binary outcome for each case. Often, this plays out as forecasting. Suppose one knows that observations with certain values for predictors fall in a particular terminal node, and that the majority of observations in that partition have, say, the outcome category A. Then, new observations that fall in that terminal node, but for which the response is unknown, sensibly might be predicted to be A as well. If a level II analysis is credible, the class label can be thought of as a fitted value to be used for forecasting.

3.2.3 Confusion Tables

If CART assigns classes to observations, it is certainly fair to ask how good those assigned classes actually are. Instructive assessments can be obtained from "confusion tables," briefly introduced earlier, which cross-tabulate the observed classes against the classes that CART assigns. Ideally, the two will correspond much of the time. But that will be a matter of degree, and confusion tables can provide useful measures of fit. We will consider confusion tables many times in the pages ahead, but a few details are important to introduce now. For ease of exposition at this point, the categorical outcome variable is binary.

Table 3.1 shows an idealized confusion table. There are two classes for the response variable: success and failure. The letters in the cells of the table are cell counts. For example, the letter "a" is the number of observations falling in the upper-left cell. All of the observations in that cell are characterized by an observed class of failure and a predicted class of failure. When the observations are from training data used to build the tree,"predicted" means "assigned." They are a product of the CART fitting process. If the observations are from test data not used to build the tree, "predicted" means "forecasted." The difference between fitting and forecasting is critical in the next several chapters.

In many situations, a split-sample strategy can be employed so that the values of tuning parameters are determined by fitting performance in the evaluation data.

Table 3.1 A confusion table

	Failure predicted	Success predicted	Model error
Failure	a	b	$b/(a+b)$
Success	c	d	$c/(c+d)$
Use error	$c/(a+c)$	$b/(b+d)$	Overall error $= \frac{(b+c)}{(a+b+c+d)}$

Once a satisfactory tree has been grown and honestly assessed with the test data, it is ready for use when the outcome class is not known, but the predictor values are.

There are generally four kinds of performance assessments that are made from confusion tables.

1. The overall proportion of cases incorrectly classified is an initial way to assess performance quality. It is simply the number of observations in the off-diagonal cells divided by the total number of observations. If all of the observations fall on the main diagonal, CART has, by this measure, performed perfectly. None of the observations have an actual class that does not correspond to the fitted class. When no cases fall in the main diagonal, CART is a total failure. All of the observations have an actual class that does not correspond to the fitted class. Clearly, a low proportion for this "overall error" is desirable, but how good is good depends on the baseline for fitting skill when no predictors are used. The real issue is how much better one does once the information in the predictors is exploited. A lot more is said about this shortly.
2. The overall error neglects that it will often be more important to be accurate for one of the response variable classes than for another. For example, it may be more important to correctly diagnose a fatal illness than to correctly diagnose excellent perfect health. This is where the row proportions shown in the far right-hand column become critical. One conditions on the actual class outcome. For each actual class, the row proportion is the number of observations incorrectly fitted divided by the total of observations of that class. Each row proportion characterizes errors made by the statistical procedure. When the true outcome class is known, how common is it for the procedure to fail to identify it?
   The two kinds of model failures are often called "false positives" and "false negatives." Successes incorrectly called failures are false negatives. Failures incorrectly called successes are false positives.[3] The row proportions representing the relative frequency of procedure-generated false negatives and false positives should, ideally, be small. Just as for overall error, the goal is to do better using the information contained in the predictors than could be done ignoring that information. But, the exercise now is done for each row separately. It is common for the procedure to perform better for one outcome than the other.
3. The column proportions address a somewhat different question. One conditions on the fitted class and computes the proportion of times a fitted class is wrong. Whereas the row proportions help evaluate how well the CART algorithm has performed, the column proportions help evaluate how useful the CART results are likely to be if put to work. They convey what would happen if a practitioner

[3]Here, the use of the class labels "success" and "failure" is arbitrary, so which off-diagonal cells contain "false positives" or "false negatives" is arbitrary as well. What is called a "success" in one study may be called a "failure" in another study.

used the CART results to impute or forecast. One conditions on either predicted success or on predicted failure from which two different estimates of errors in use can be obtained. Just as for model errors, it is common for the errors in use to differ depending on the outcome. The goal is much the same as for model error: for each column, to be wrong a smaller fraction of the time than if the predictors are ignored.

4. The ratio of the number of false negatives to the number of false positives shows how the results are trading one kind of error for the other. For example, if b is 5 times larger than c, there are five false positives for every false negative. This means that the CART procedure produces results in which false negatives are five times more important than false positives; one false negative is "worth" five false positives. Ratios such as this play a key role in our discussion later about how to place relative costs on false negatives and false positives.

In summary, confusion tables are an essential diagnostic tool. We rely on them in this chapter and all subsequent ones. They also raise some important issues that are very salient in the pages ahead.

3.2.4 CART as an Adaptive Nearest Neighbor Method

Not only is CART an essential component of many statistical learning procedures, it has direct links to other methods that on the surface might seem to be totally unrelated. In particular, it can be instructive to think about CART within an adaptive nearest neighbor framework. The partitions shown in Fig. 3.1 can be viewed as neighborhoods defined by nearest neighbors. But unlike conventional nearest neighbor methods, CART arrives at those neighborhoods adaptively. Like all stagewise (and stepwise) procedures, CART data snoops.

Consider, for example, terminal node 3 in Fig. 3.2. Within that node, are all observations whose values of x are greater than c_1, and whose values of z are greater than c_2. For these observations, a conditional mean or proportion can be computed. In other words, the nearest neighbors for either of these summary statistics are defined as all cases for which $x > c_1$ and $z > c_2$. All of the observations for which this condition holds can be used to arrive at a single numerical summary for the response variable.

The neighborhood represented by the terminal nodes is adaptive in three senses. First, information from the response variable is used to determine the neighborhood. A measure of fit is exploited to arrive recursively at the terminal node neighborhood. Second, because a large number of predictors and break points are examined, a large number of potential neighborhoods are evaluated before an actual neighborhood is formed. Third, the terminal node neighborhoods that result can be defined by different sets of predictors and different sets of break points. Both are arrived at inductively by the CART algorithm. For example, a given predictor can help define one terminal node, but not another. Even when a given predictor is used to define more than one

terminal node, it may enter at different stages of the partitioning and use different break points.

The terminal node neighborhoods can be constructed sequentially by where in the predictor space some step function for the response leads to the greatest difference in level. This follows from the desire to make the two resulting subsets as homogeneous as possible. Then, because for each split the single best predictor is chosen, each terminal node, and its implied neighborhood, can be defined using a subset of predictors. That is, one need not define nearest neighbors using the entire predictor space. This is in contrast to the multivariate lowess smoother discussed in the last chapter and is a way to fight back against the curse of dimensionality.

Within a level II analysis, there are also implications for the bias-variance tradeoff it fitted values are used to estimate the true response surface.[4] Suppose for a given terminal node a goal is to estimate the proportion of 1s for all observations in the terminal node neighborhood defined by a particular set of x-values. For example, suppose the binary response is whether a high school student graduates; graduation is coded as 1 and drop out is coded as 0. The terminal node neighborhood in question contains high school students, self-identified as Asian, who have family incomes in excess of $75,000. But suppose there is an omitted predictor variable. Even if it happened to be in the dataset, it would not be selected (e.g., the labor market for blue collar jobs). A more subtle error occurs when the correct predictor is chosen, but at the wrong breakpoint. (e.g., a grade point average < 2.0 rather than < 1.5). Because the neighborhood is not defined correctly, and there is a good chance that the estimated proportion will be biased with respect to the true response surface. A potential remedy is to further refine the terminal nodes by growing a larger tree. There is an opportunity for more predictors to determine the terminal node neighborhoods leading to a more homogeneous mixes of cases. But for a given sample size, a larger tree implies that on the average, there will be fewer cases in each terminal node. Although bias may be reduced, variance may be increased.

In summary, although smoothers, adaptive nearest neighbor methods, and CART come from very different traditions, they have important similarities. Additional and helpful connections between other statistical learning procedures will be addressed in subsequent chapters. We turn now to the CART details.

3.3 Splitting a Node

The first problem that the CART algorithm needs to solve is how to split each node using information contained in the set of predictors. For a quantitative predictor with m distinct values, there are $m - 1$ splits that maintain the existing ordering of values. So, $m - 1$ splits on that variable need to be evaluated. For example, if there are 50

[4] Recall that the response surface approximation itself can often be estimated in an asymptotically unbiased fashion.

distinct high school GPA scores possible, there are 49 possible splits that maintain the existing order. However, there are often algorithmic shortcuts that can capitalize, for instance, on ordering the splits by the size of the conditional mean or proportion. The same logic holds for ordinal predictors.

Order does not matter for categorical predictors. Consequently, a categorical variable with k categories has $(2^{k-1} - 1)$ possible splits. For example, if there are five ethnic group categories, there are 15 possible splits. Hence, although there are sometimes shortcuts here too, the computational burdens are generally much heavier for categorical variables. There are no order restrictions on how a categorical predictor is split.

Recall that starting at the root node, CART algorithm evaluates all possible splits of all predictor variables and picks the "best" single split overall. The best split of the variable selected is better than the best split of any other predictor. The data are then partitioned according to that best split. The same process is applied to all subsequent nodes until, at the extreme, all cases have been placed in a terminal node all their own. Because the final partitions do not overlap, each case can only be in one terminal node.

How is "best" to be formally defined? It is common to focus on the "impurity" of a node. Impurity is essentially heterogeneity. The goal is to have as little impurity (heterogeneity) overall as possible. Consequently, the best split is the one that reduces impurity the most. To help simplify the exposition that follows, assume a binary response variable coded 1 or 0. The term "success" is used to refer to outcomes coded 1 and the term "failure" to refer to outcomes coded 0.

A discussion of impurity can work using the proportions of successes and failures in a node or using the probabilities of a success or a failure for cases in that node. Most expositions favor probabilities and assume that they are known. There is really no didactic cost to this assumption.

Suppose that a dataset is realized from a joint probability distribution, so the concept of a probability can apply. Consider a given node, designated as node A. The impurity of node A is taken to be a nonnegative function of the probability that $y = 1$, written as $p(y = 1|A)$.

If A is a terminal node, ideally it should be composed of cases that are all equal to 1 or all equal to 0. Then $p(y = 1|A)$ would be 1.0 or 0.0. Intuitively, impurity is the smallest it can be. If half the cases are equal to 1 and half the cases are equal to 0, the probability is equal to .50. A is the most impure it can be because a given case is as likely to be a 1 as it is a 0.

One can more formally build on these intuitions. Let the impurity of node A be:

$$I(A) = \phi[p(y = 1|A)], \tag{3.2}$$

with $\phi \geq 0$, $\phi(p) = \phi(1-p)$, and $\phi(0) = \phi(1) < \phi(p)$. In other words, impurity is nonnegative, and symmetrical with a minimum when A contains all 0s or all 1s, and a maximum when A contains half of each.[5]

There remains a need to define the function ϕ. For classification, three definitions have been used in the past: Bayes error, the cross-entropy function, and the Gini index. In order they are:

$$\phi(p) = \min(p, 1-p); \tag{3.3}$$

$$\phi(p) = -p\ \log(p) - (1-p)\ \log(1-p); \tag{3.4}$$

and

$$\phi(p) = p\ (1-p). \tag{3.5}$$

All three functions for impurity are concave, having minimums at $p = 0$ and $p = 1$ and a maximum at $p = .5$. Entropy and the Gini index are the most commonly used, and in CART generally give very similar results except when there are more than two response categories. Then, there is some reason to favor the Gini index (Breiman et al. 1984: 111). The Gini index is more likely to partition the data so that there is one relatively homogeneous node having relatively few cases. The other nodes are then relatively heterogeneous and have relatively more cases. For most data analyses, this is a desirable result. Entropy tends to partition the data so that all of the nodes for a given split are about equal in size and homogeneity. This is generally less desirable. But the choice between the two impurity functions can depend on the costs associated with classification errors, which is a topic addressed shortly.

One might legitimately wonder why CART does not directly minimize classification error. Direct minimization of overall classification error is discussed in some detail by Breiman and his colleagues (1984: Sect. 4.1). In part because classification error is not continuous, there can be several splits for a given stage minimizing classification error. In addition, minimizing classification error at each stage has a tendency, like entropy, to produce a tree structure that is often more difficult to interpret. For now, we focus on node impurity as just defined. However, direct minimization of classification error resurfaces as a useful consideration when boosting is considered in Chap. 6.

For real applications, the probabilities are not likely to be known. Suppose one uses data to estimate the requisite probabilities. It should be apparent by now that obtaining good estimates involves conceptual and technical complications, but for didactic purposes, assume that the complications have been effectually addressed.

Building on Zhang and Singer (1999; Chaps. 2 and 4), for any internal node, we focus on a potential left "daughter" node A_L, and a right "daughter" node A_R. We wish to evaluate the usefulness of a potential partitioning of the data. Table 3.2

[5]The use of I in Eq. 3.2 for impurity should not be confused with the use of I to represent an indicator variable. The different meanings should be clear in context.

Table 3.2 Counts used to determine the usefulness of a potential split (The cell entries are counts, with the first subscript for rows and the second subscript for columns)

	Failure	Success	Total
Left Node: $x \leq c$	n_{11}	n_{12}	$n_{1.}$
Right Node: $x > c$	n_{21}	n_{22}	$n_{2.}$
	$n_{.1}$	$n_{.2}$	$n_{..}$

provides the information needed. The entries in each cell are counts, with rows as the first subscript and columns as the second subscript.

As before, we let $y = 1$ if there is a success and 0 otherwise. The estimate of $p(y = 1|A_L)$ is given by $n_{12}/n_{1.}$. Similarly, the estimate $p(y = 1|A_R)$ is given by $n_{22}/n_{2.}$.

Consider calculations for entropy as an example. Entropy impurity for the left daughter is

$$I(A_L) = -\frac{n_{11}}{n_{1.}}\log(\frac{\mathrm{n}_{11}}{\mathrm{n}_{1.}}) - \frac{\mathrm{n}_{12}}{\mathrm{n}_{1.}}\log(\frac{\mathrm{n}_{12}}{\mathrm{n}_{1.}}). \tag{3.6}$$

Entropy impurity for the right daughter is

$$I(A_R) = -\frac{n_{21}}{n_{2.}}\log(\frac{\mathrm{n}_{21}}{\mathrm{n}_{2.}}) - \frac{\mathrm{n}_{22}}{\mathrm{n}_{2.}}\log(\frac{\mathrm{n}_{22}}{\mathrm{n}_{2.}}). \tag{3.7}$$

Imagine that for the left daughter there are 300 observations with 100 successes and 200 failures. It follows that the impurity is $-.67(-.40) - .33(-1.11) = .27 + .37 = .64$. Imagine now that for the right daughter there are 100 observations with 45 successes and 55 failures. The impurity is $-.55(-.60) - .45(-.80) = .33 + .36 = .69$.

To put these numbers in context, it helps to consider the smallest and largest possible values for the impurity. The greatest impurity one could obtain would be for 50 % successes and 50 % for failures. The computed value for that level of impurity would be .693. For proportions of 1.0 or 0.0, the value of entropy impurity is necessarily 0. In short, the minimum value is 0, and the maximum is a little more than .69. The closer one gets to 50–50, where the impurity is the greatest, the closer one gets to .693. The impurity numbers computed are rather close to this upper bound and reflect, therefore, substantial heterogeneity found in both daughter nodes. It is likely that this split would not be considered to be a very good one.

Once all possible splits across all possible variables are evaluated in this manner, a decision is made about which split to use. But the impact of a split is not just a function of the impurity of a node. The importance of each node must also be taken into account. A node in which few cases are likely to fall should be less important than a node in which many cases are likely to fall. The former probably will not matter much, but the latter probably will.

We define the improvement resulting from a split as the impurity of the parent node minus the weighted left and right daughter impurities. If this is a large number,

entropy impurity is reduced substantially. More formally, the benefits of the split s for node A,

$$\Delta I(s, A) = I(A) - p(A_L)I(A_L) - p(A_R)I(A_R), \tag{3.8}$$

where $I(A)$ is the value of the parent impurity, $p(A_R)$ is the probability of a case falling in the right daughter node, $p(A_L)$ is the probability of a case falling in the left daughter node, and the rest is defined as before. The two probabilities can be estimated from information such as provided in Table 3.2; they are just the marginal proportions $n_{1.}/n_{..}$ and $n_{2.}/n_{..}$. They serve as weights.

$\Delta I(s, A)$ can be essentially the reduction in the deviance and thus, there is a clear link to the generalized linear model that can prove useful when different fitting procedures are compared. CART finds the best $\Delta I(s, A)$ for each variable. The variable and split with the largest value are then chosen to define the new partition. The same approach is applied to all subsequent nodes.

The CART algorithm can keep partitioning until there is one case in each node. There is then no impurity whatsoever. Such a tree is called "saturated." However, well before a tree is saturated, there will usually be far too many terminal nodes to interpret, and the number of cases in each will be quite small. The very small node sizes lead to very unstable results. Small changes in the data can produce trees with rather different structures and interpretations. One option is to prohibit the CART algorithm from constructing any terminal nodes with sample sizes smaller than some specified value. A second option is considered shortly. And we show in later chapters that there can be ways to work usefully with saturated trees, as long as there is a very large number of them.

3.4 Fitted Values

CART is a method to construct, using a predictors, a conditional distribution. Interest commonly centers on some measure of location. For classification problems, the conditional proportion is usually the measure. For regression problems, the measure is usually the conditional mean. Using linear basis expansions, explicit links to parametric regression can be made. It follows that most of the issues raised by parametric regression, and most of the concepts associated with parametric regression, carry over.

3.4.1 Fitted Values in Classification

As noted earlier, there are two kinds of classification fitted values for CART. First, within each terminal node, the proportion of observations for each of the classes are fitted values. They are conditional proportions that characterize the terminal node. For example, if the response variable is binary and the proportion of successes is .55, the fitted value for that terminal node is .55 (or .45 if one wants to focus on failures).

It can be useful to compare such fitted values across terminal nodes as part of a level I analysis.

If a level II analysis can be justified, the reasoning becomes more elaborate. One envisions the same neighborhoods in the joint probability distribution as found with the data. In each of those population neighborhoods, there are expectations for the proportion of successes. Those expectations are the estimation targets, and the in-sample proportions can be used as estimates. However, because the neighborhoods are a product of extensive data snooping, the quality of those estimates is unknown and possibly misleading. Having test data can really help. One "drops" the test data down the tree, and the fitted proportions will be asymptotically unbiased estimates of the tree-derived approximate response surface. Moreover, with the nonparametric bootstrap applied to the test data, asymptotically valid confidence intervals can be constructed around the estimates. An example using the bootstrap with CART is provided as part of the last exercise in this chapter.

The second kind of fitted values require an additional step: a class is assigned to each terminal node. As described above, the assigned class is determined by a majority vote (or a plurality if there are more than two response categories). Then within each terminal node, winning class is attached to each observation. The voting threshold (e.g., .50) has some of the features of a decision boundary, but only for one node at a time.

In-sample performance measures can follow directly. For any given terminal node, what proportion of the time is the assigned class the correct class? For example, if the proportion of successes in a terminal node is .90, the assigned class is "success," and that assigned class will be incorrect for 10 % of the observations in that node. The ideal real result would 0.0 %, which would follow if the terminal node were perfectly homogeneous. The issues are much the same as for the column calculations in confusion tables. How often is the assigned class wrong? One difference for terminal nodes is that the error rate for a terminal node is only for a subset of observations defined by a predictor neighborhood. Another difference is that a full confusion table cannot be constructed because for a given terminal node, only one class is assigned. The table would have one column and two rows.

But just as before, in-sample results can be misleading. It is better to work with test data with which the accuracy of assigned classes can be more honestly addressed. Better estimates of generalization error may be obtained, and once again, a non-parametric bootstrap can be applied to the test data to construct confidence intervals around the estimates of generalization error.

3.4.2 An Illustrative Prison Inmate Risk Assessment Using CART

A key issue for prison administrators is understanding which inmates are likely to place themselves and others in harm's way. Use of narcotics, assaults on prison

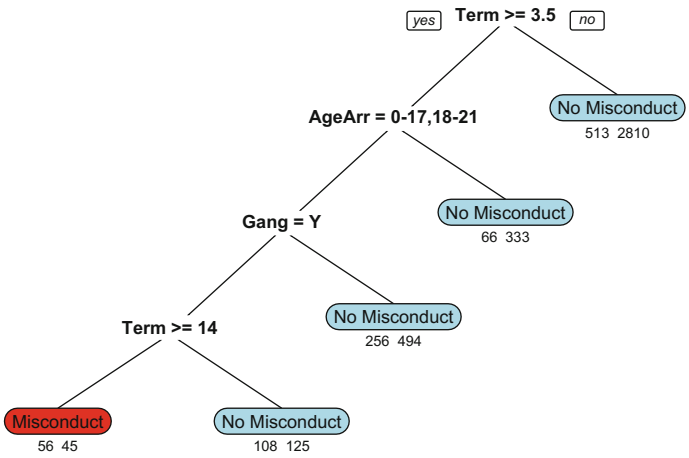

Fig. 3.5 CART recursive partitioning of the prison data (N = 4806)

guards, and homicides are examples. Although such events are relatively rare, they have very serious consequences. It follows that it would be very useful if such conduct could be anticipated. Then, for the high-risk inmates, preventive measures might be taken. For example, inmates from rival street gangs might be housed in different prisons. Low-risk inmates might be given far greater freedom to participate in job training and educational programs. A prerequisite, however, is a way to find effective predictors of misconduct in prison.

Using data from the administrative records of a large state prison system, Fig. 3.5 shows a classification tree suggesting which kinds of inmates are reported for some form of misconduct within 18 months of intake.[6] From Fig. 3.6, one can see that there are two tuning parameters. A relatively large minimum node sample size of 100 was imposed to stabilize the results and to keep the diagram very simple. More will be said soon about the complexity parameter *cp*, but a small value allows splits that do not reduce node impurity very much. According to the documentation of *rpart()*, *cp* is a threshold on the proportional improvement in fit. Only splits improving the fit by at least the value of *cp* will be permitted. A large value for the threshold means that acceptable splits must improve the fit a lot.[7]

The three predictors in Fig. 3.5 were selected by the CART procedure from a larger set of 12 predictors.

1. Term: Nominal sentence length in years. (The nominal sentence is the sentence given by the trial judge. Inmates are often released before their nominal sentence is fully served.)

[6] Because of confidentiality concerns, the data may not be shared.

[7] The tree diagram is formatted differently from the tree diagram used for the Titanic data to emphasize the terminal nodes.

```
library(rpart) # Load the CART library
library(rpart.plot) # Load the fancy plotting library

# Partition the data
out<-rpart(Fail~AgeArr+Gang+CDC+Jail+Psych+Term,
           data=temp, method="class",
           minbucket=100,cp=.001)
# Plot a tree
prp(out,extra=1,faclen=10,varlen=15,
box.col=c("red","lightblue")[out$frame$yval])
```

Fig. 3.6 R code for a CART analysis of prison misconduct

2. AgeArr: Age at arrival at the prison reception center in years using 16–20, 21–26, 27–35, and 36 or older.
3. Gang: gang membership with Y for "yes" and N for "no."

Terminal nodes are labeled "No Misconduct" if the majority in that node do not engage in misconduct and "Misconduct" if the majority do. The numbers within each terminal node show left to right the counts of misconduct cases and no misconduct cases respectively.[8] There are 4806 observations in the root node. Among these, observations are sent left if the nominal prison term is equal to or greater than 3.5 years and to the right if the nominal prison term is less than 3.5 years. Likewise, at each subsequent split, the observations meeting the condition specified are sent left.

The story is simple. Inmates with nominal prison terms over 14 years, who are under 22 years of age, and who are gang members are more likely than not to be reported for misconduct (i.e., 56 cases v. 45 cases). Inmates with nominal terms of less than 3.5 years are relatively unlikely to be reported for prison misconduct regardless of age, gang membership or any other 9 predictors in the data (i.e., 513 cases v. 2810 cases). In the other three terminal nodes, no-misconduct cases are the majority. One might say that a very long nominal terms puts an observation over the top but by itself is not associated with a preponderance of reported misconduct. Each of the five terminal nodes but the one on the far right are defined by interaction effects.

There are readily available (post hoc) explanations for these results. Judges impose far longer prison terms on individuals who have committed very serious, usually violent, crimes. The data suggest that the judges are on to something. And it is well known that young gang members can be especially difficult especially when housed with members of rival gangs.

At the same time, the partitions are arrived at inductively, and the regression model implied is no doubt badly misspecified. For example, there is no information on the

[8] In R, the character variable default order left to right is alphabetical.

security level in which the inmate is placed, and higher security levels are thought to reduce incidents of serious misconduct.[9]

Moreover, a conventional analysis using logistic regression would look very different. It is likely that most of the predictors would have been entered only as main effects, and unlikely that the 4-way interactions on the left branches would have been included. Neither approach is likely to be correct by conventional regression analysis thinking.

Nevertheless, suppose at intake, prison administrators wanted to intervene in a manner that could reduce prison misconduct. Absent other information, it might make sense to distinguish young gang members with very long sentences from other inmates. In effect, the prison administrators would be using the classification tree to make forecasts that would help inform placement and supervision decisions.

Although such thinking can have real merit, there are complications. We will have shifted into a level II analysis if the results are to be used for forecasting. One would need to consider how the data were realized, the impact the CART's extensive data snooping, and out-of-sample performance. In addition, the consequences of failing to identify a very dangerous inmate at intake can be enormously different from the consequences of incorrectly labeling a low-risk inmate as dangerous. The different consequences of can have very different costs. In the first instance, a homicide could result. In the second instance, an inmate might be precluded from participating in prison programs that could be beneficial. It stands to reason, therefore, that the differential costs of forecasting errors should be introduced into the CART algorithm. We turn to that now.

3.5 Classification Errors and Costs

Looking back at the earlier details on how splits are determined, there is little explicit concern about classification errors, but it can be shown that all three impurity functions treat classification errors symmetrically. Incorrectly classifying an A as a B, is treated the same as incorrectly classifying a B as an A. Similar reasoning carries over when the assigned classes are used for forecasting. The class assigned to a terminal node is determined by a majority vote. All observations in a given terminal node get a single vote that has the exact same weight for all. But is that always reasonable?

Consider the second terminal node from the left in Fig. 3.5. The vote is close, but the no misconduct cases win. It follows that the 108 inmates who in fact had reported misconduct are misclassified. But suppose each vote for no misconduct inmates counted as half a vote. The misconduct vote would carry the day, and the terminal node would be assigned to the misconduct class. There would then be 125 classification errors, but because each would count half as much as before, overall

[9]The issues are actually tricky and beyond the scope of this discussion. At intake, how an inmate will be placed and supervised are unknown and are, therefore, not relevant predictors. Yet, the training data need to take placement and supervision into account.

Table 3.3 CART confusion table for classifying inmate misconduct (N = 4816)

	Classify as no misconduct	Classify as misconduct	Model error
No misconduct	3762	45	.01
Misconduct	953	56	.95
Use error	.20	.45	Overall error = .21

error would be reduced from 108 to $125 \times .5 = 62.5$ Whereas the original cost ratio was 1 to 1, it is now 2 to 1. In concrete terms, a misclassified misconduct cases is taken to be twice as costly as a misclassified no misconduct case.

As we examine in depth shortly, introducing "asymmetric" costs for classification errors can be a game changer that actually begins with the criteria by which splits are determined and involves far more than re-weighting the votes in terminal nodes. We will see that by altering the prior, the measures of impurity in Eq. 3.8 are replaced by measures of the expected costs of classification errors when a split is determined.[10]

Whether there actually are asymmetric costs resulting from classification errors depends on what is done with the classification tree. If the CART results are just archived in some academic journal, there are probably no costs one way or the other. If the results are used to guide future research, costs can be a real issue. In genomic research, for example, follow-up research would be wasted if a genomic snip is incorrectly identified as important. Conversely, a significant research lead might be missed if an important genomic snip is incorrectly overlooked. These two costs are probably not be the same. In applied research, costs can be very important, as the prison example should make clear. A way is needed to build in the differential costs classification errors.

3.5.1 Default Costs in CART

Without any apparent consideration of costs, the CART algorithm can make classification decisions about the misconduct of inmates. But in fact, costs are built in. To see how, we need to examine Table 3.3, which is constructed from the prison misconduct analysis.

As noted earlier, tables of the form of Table 3.3 are often called confusion tables. They can summarize the classification performance (or as we see later, forecasting performance) of a particular classifier. Here, that classifier is CART. There is a row

[10] But as Therneau and Atkinson (2015: Sect. 3.3.2) state,"When altered priors are used, they affect only the choice of split. The ordinary losses and priors are used to compute the risk of the node. The altered priors simply help the impurity rule choose splits that are likely to be good in terms of the risk." For example, the deviance or mean squared error are computed as usual to show how much better the fit has become.

for each actual outcome. There is a column for each classified outcome. Correct classifications are in the main diagonal. Misclassifications are the off-diagonal elements. Thus, we learn that 998 out of 4806 cases were incorrectly classified. But, how good this is depends on the baseline.

Had no predictors been used, classification could have been done from the misconduct marginal distribution alone. Applying the Bayes classifier, all cases could have been classified as having no reported misconduct. Then, 999 out of 4806 of the cases would have been incorrectly classified. Clearly, there is no meaningful improvement by this yardstick.

However, there is lots more going on in the table. The overall fit ignores how well CART does when the two response variable categories separated. Consider first what happens when one conditions on the actual class. In this case, the absence of misconduct can be classified very well — 99 % of the cases are classified correctly. In contrast, instances of misconduct are misclassified about 95 % of the time. The overall error rate masks these important differences. CART performs very well when there is no misconduct and very poorly when there is misconduct.

The columns in Table 3.3 are also instructive. Now the conditioning is with respect to the assigned class, not the actual class. If the no misconduct class is assigned, it is wrong for about 20 % of the observations. If the misconduct class is assigned, it is wrong for about 45 % of the observations. So, mistakes are relatively more common when misconduct is assigned, but we do better with the misconduct class that one might expect. If one is thinking ahead to possible forecasting applications, there may be some hope.

Where are costs in all this? Key information about costs is contained in the two off-diagonal cells. There are 45 no misconduct cases incorrectly classified and 953 misconduct cases incorrectly classified. The former one might call false positives and the latter one might call false negatives. The ratio of the cell counts is $953/45 = 21.2$. There are about 21 false negatives for each false positive. Stated a little differently, one false positive is "worth" about 21 false negatives, and it's cost is, therefore, about 21 times greater. According to the confusion table, it is 21 times more costly to misclassify a case of no misconduct (i.e., a false positive) than to misclassify a case of misconduct (i.e., a false negative).

All of the performance results in the table depend on the 21 to 1 cost ratio produced by default, and one has to wonder if corrections administrators and other stakeholder think that the 21 to 1 cost ratio makes sense. The analysis is shaped by treating false positives as far more costly than false negatives. In practice, prison misconduct false negatives are usually thought to be more costly (Berk 2012), so this analysis may well have it upside down. And if that's right, all of the various measures of classification performance are highly suspect and potentially very misleading.

Important lessons follow. First, the CART algorithm (and every other classification procedure for that matter) *necessarily* introduces the costs of classification errors at least implicitly when classes are assigned. There is no way to circumvent this. Even if the data analyst never considers such costs, they are built in. To not consider the relative costs of misclassifications is to leave the relative cost determinations to the data and the classification algorithm. Second, the way cases are classified will

vary depending on the cost ratios. As a result, the entire confusion table can change dramatically depending on the cost ratio. Finally, the classes assigned can serve as forecasts when the predictor values are known and the outcome class is not. But if the assigned classes depend on the false negative to false positive cost ratio, so do the forecasts.

If costs are so important, there is a need to understand how they are incorporated into the CART algorithm. This will set the stage for a data analyst to introduce costs explicitly. In other words, it is desirable — some might say essential — to treat the relative costs of classification/forecasting errors as an *input* to the algorithm. Unless this is done, the results risk being unresponsive to the empirical questions being asked.

3.5.2 Prior Probabilities and Relative Misclassification Costs

The marginal distribution of any categorical response variable will have a proportion of the observations in each response category. In the prison example, .21 of the inmates had a reported incident of misconduct, and .89 of the inmates had not. However, before looking at the data, one might already hold strong beliefs from past research or other information about what those marginal proportions should be. For example, the design through which the data were collected may have over-sampled inmates reported for misconduct in order to have a sufficient number of them in the study. But for many uses of the results, it would make sense to weight the observations back to the actual proportion of inmates who engage in misconduct. For a level II analysis, proportions can be conceptualized as the "prior probabilities" associated with the response variable. The word "prior" comes from Bayesian statistical traditions in which the "prior" refers to the beliefs of the data analyst, before the data are examined, about the probability density or distribution of some parameter.

There has been some work within Bayesian traditions capitalizing on several different kinds of CART priors (Chipman et al. 1998; 1999), including a "pinball prior" for tree size and some features of tree shape (Wu et al. 2007). That is, key features of the tree itself are given a prior probability distribution. The ideas advanced are truly interesting, and have led to some important statistical learning spinoffs (Chipman et al. 2010). But in practice, the technical complications are considerable, and it is not even clear that there will often be credible information available to make such priors more than tuning parameters. Tuning parameters are important and useful, but they are not a feature of probability distributions specified before the data analysis begins. Consequently, for present purposes, we will use the term "prior" only with reference to the marginal probability distribution of the response variable. Consistent with most exposition of CART, we will proceed as if a level II analysis is appropriate, but in broad brush strokes the lessons learned apply as well to level I approaches.

Assume that a credible level II analysis has been undertaken with CART. Drawing heavily on Therneau and Atkinson (2015), suppose the training data has N

observations and C classes for the response variable. The CART algorithm produces K terminal nodes. Define π_i for $i = 1, 2, \ldots,$ as the prior probability of being in class i. For the binary cases, i is here represented by 1 or 2. $L(i, j)$ is the loss matrix. The elements in the main diagonal (i.e., $i = j$) are the costs of a correct classification, assumed to be 0.0. The off-diagonal elements (i.e. $i \neq j$) are the costs of classification errors, assumed to be positive.

A is a terminal node, and $\tau(x)$ is the true class for an observation, where x represents the vector of predictor variable values for that observation. We let $\tau(A)$ be the class assigned to node A. N_i and N_A are the number of observations in the sample that are in class i and in node A, respectively, with N_{iA} the number of observations of class i in node A. The following relationships in each terminal node hold.

1. $P(A)$ is the probability of cases appearing in node A, which is equivalent to $\sum_{i=1}^{C} \pi_i P[x \in A|\tau(x) = i]$, where π_i is a prior probability. For each class i, the prior probability of a case being in class i is multiplied by the probability that class i cases will be in node A. This product is summed over classes. $P(A)$ can be estimated by $\sum_{i=1}^{C} \pi_i(N_{iA}/N_i)$. Because prior probabilities figure directly in these calculations, prior probabilities can affect the tree structure.
2. Then, $p(i|A)$ is the conditional probability of class i given that a case is in node A, or $P[\tau(x) = i|x \in A]$. The value of $p(i|A)$ can be estimated by the number of cases of class i in node A, divided by the total number of cases in that node and equals $\pi_i P[x \in A|\tau(x) = i]/P[x \in A]$, which can also be estimated by $\pi_i(N_{iA}/N_i)/\sum_{i=1}^{C} \pi_i(N_{iA}/N_i)$. The conditional probability of a case with true class i landing in A depends in part on the prior probability that a case is class i to begin with.
3. $R(A)$ is the "risk" associated with node A, such that $\sum_{i=1}^{C} p(i|A)L(i, \tau(A))$. In other words, the risk associated with node A is for a binary response the conditional probability of a case of type $i = 1$ falling in that node times costs that follow, plus the conditional probability of a case of type $i = 2$ falling in that node times costs that follow. Risk is a function of both the conditional probabilities and the costs. Because the conditional probabilities depend on the prior probabilities, the prior probabilities affect risks.
4. $R(T)$ is the risk of the entire tree T, which equals $\sum_{j=1}^{K} P(A_j)R(A_j)$, where A_j is for each of the K terminal nodes of the tree. We are now just adding the total risk associated with each terminal node, weighted by the probability of cases falling in that node. This can also be seen as the "expected cost" of the entire tree.

To help make these concepts more concrete, Fig. 3.7 provides a numerical illustration constructed from fictional training data. There are 10,000 high school students in the data. 2000 dropped out (D) and 8000 graduated (G). From these figures one can estimate the "objective" prior as $p(D) = .2$, and $p(G) = .8$. It is called "objective" because it is estimated empirically. We will see later that the objective prior can be altered in useful ways. Shown is a confusion table for terminal node A in which the actual outcome is tabulated by the classified outcome. For example, there are 50 false negatives and 250 false positives. On the margins of the table are the row and column

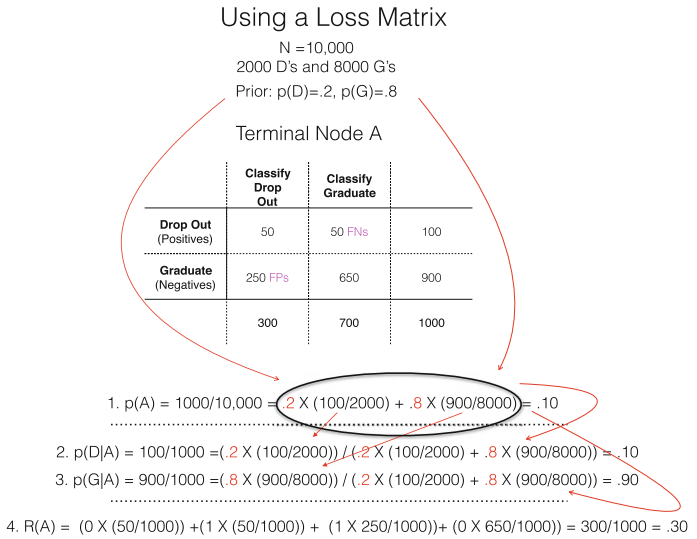

Fig. 3.7 Terminal node A calculations for Dropouts (D) and Graduates (G)

totals, not the usual error rates. With the table in place, one can illustrate how the expressions just described are employed.

What is the estimated probability of a case falling in this terminal node? One can immediately intuit that the probability is 1000/10,000 = .10. That result can be unpacked as shown in line #1, consistent with the formal expressions above. The key point is that the estimated prior distribution (in red) plays a role.

What is the estimated conditional probability that the case will be a high school drop out, given that a case lands in terminal node A? One can immediately intuit that the estimated conditional probability is 100/1000 = .10. Line #2 shows how this probability can be unpacked and again the prior is a player, consistent with the formal expression above. Likewise, what is the estimated conditional probability that the case is a high school graduate, given that a case lands in terminal node A? The intuitive answer is $900/1000 = .90$ which is unpacked in line #3. As before, the prior is involved.

Finally, in line #4 we get to the risk associated with terminal node A. The four probabilities come from the four interior entries in the table. These probabilities are affected by the prior through the conditional probabilities on the margins of the table. For the risk calculations, we need to introduce costs. As before, there are no costs associated with correct classifications. For incorrect classifications, we use for now a cost of 1.0. The loss matrix $L(i, j)$ has 0s along the main diagonal and 1s in the off-diagonal cells.

Again building on intuition, the risk for terminal node A is the sum of the estimated conditional probabilities of a case falling in each cell of the table, each multiplied by the cost of falling in that cell, given that the observation has landed in terminal

node A to begin with. In effect, risk is an expected cost, and with a value of 1.0 for the cost of misclassifications, the expected risk is nothing more than the proportion of cases misclassified. That value for this example is .30.

For example, there are 1000 cases in the table. The estimated conditional probability of falling in the upper left cell is $50/1000 = .05$. Because this cell only contains correct classifications, the cost is 0.0. It makes no contribution to the risk. For the lower left hand cell, the estimated conditional probability is $250/1000 = .25$. Because this cell contains misclassifications (i.e., false positives), the estimated conditional probability is multiplied by a cost-value of 1.0. The same reasoning applies to the two cells on the right side of the table. When all of the expected costs are summed, the result is .30. When the cost of a false positive or a false negative is equal to 1, it makes sense that the risk for terminal node A is simply the proportion misclassified.

There are four general lessons.

1. The calculations just illustrated apply to each terminal node and, therefore, the full tree. When all misclassifications are given a cost of 1.0, the risk for the tree as a whole is the total number of cases misclassified.
2. The prior's effects cascades down all the way to the final risk calculations. These calculations depend on the estimated probabilities associated with each cell that in turn are influenced by all of the row and column estimated conditional probabilities. If the prior is different, what follows will be different as well. In short, the expected number of classification errors in a terminal node is affected by the priors. Where these are distributed within the confusion table is affected too.
3. Replacing $L(i \neq j) = 1$ with $L(i \neq j) = m$, where m is some constant, just scales up or down the risk by some arbitrary amount and makes no difference to the CART algorithm. It is a bit like measuring a person's height in inches rather than feet. Each misclassification has a cost of 12 rather than 1.
4. In this example, the costs of false positives and false negatives are taken to be the same. That is the usual CART default. Therefore, if one just lets the data determine everything, it is the same as (a) making the costs of all classification errors the same and (b) taking the empirical distribution of the response as the appropriate prior distribution. This is exactly what was done for the results in Fig. 3.5.

But what does one do if as in Table 3.3 the empirical balance of false negatives to false positives is unsatisfactory? It would seem that the most direct response would be to alter the costs in the loss matrix to make them asymmetric. The off-diagonal elements in the loss matrix would not longer be the same. False negatives are then made more (or less) costly relative to false positives. For example, instead of the loss matrix $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, one might use the loss matrix $\begin{bmatrix} 0 & 10 \\ 1 & 0 \end{bmatrix}$.

Because false positives and false negatives are now being weighted differently, one might well expect CART output to be affected. Looking back at second righthand side term in the expression in line #4, imagine that the 1 was replaced by a 10. False negatives are made 10 times more costly. It makes sense that in an effort to minimize

(or at least reduce) risk, the algorithm would aggressively seek ways to reduce the number of false negatives, even if it meant increasing substantially the number of false positives. That, in turn, would presumably impact the classification tree and the confusion table.

One possible approach is to use the loss matrix as an argument in the Gini function (Therneau and Atkinson 2015: Sect. 3.2). As the "generalized" Gini function, impurity is defined then as

$$G(p) = (1/2) \sum_i \sum_j L(i, j) p_i p_j. \tag{3.9}$$

Unfortunately, this approach does not work in practice (Therneau and Atkinson 2015: 7). $G(p)$ is not necessarily concave, which was a key motivation for each of the three impurity functions. In addition, calculation of the generalized Gini function "symmetrizes" the loss matrix. It is as if the loss matrix were added to its transpose. For the binary case, any cost asymmetry is lost. Both problems apply as well to categorical response variables with more than two categories, although consequences for the content of the loss matrix are a bit more complicated.

However, one need not work with the Gini index directly. Recall from the algebraic treatment several pages back that the probability of class i in node A, $p(i|A)$, can be estimated by the expression $\pi_i(N_{iA}/N_i)/\sum_{i=1}^{C} \pi_i(N_{iA}/N_i)$, and its associated risk is $p(i|A)L(i, \tau(A))$. Therefore, the risk associated with a class in a given node is scaled by the product of the prior probabilities and the entries in the loss matrix. This was implicit in expression #4 in Fig. 3.7. To see the consequences, suppose there exist a new $\tilde{\pi}$ and a new $\tilde{L}$ so that

$$\tilde{\pi}_i \tilde{L}(i, j) = \pi_i L(i, j). \tag{3.10}$$

The risks are identical, and it does not matter what the particular values of $\tilde{\pi}$ and $\tilde{L}$ happen to be as long as the equality holds. This opens the door for lots of possibilities. If one just thinks of the righthand side as the weight given to the classification errors for class i in a given node, and if more or less weight is desired, one can alter either the prior, or the costs, or both. In practice, it is less work to alter one of them, and the choice can depend on how the software is written. In the binary response case, if one wanted to alter the weights by altering just the prior distribution to π_i^* from π_i, one would use

$$\tilde{\pi}_i^* = \frac{\pi_i L_i^*}{\pi_i L_i^* + \pi_j L_j^*}. \tag{3.11}$$

With index for each response class, the values of π_i are the probabilities associated with the empirical prior distribution. The values of L_i^* are the new costs. Because of the normalization, all that matters in the loss matrix is relative costs. Thus, one just has to know, for example, that the cost of one kind of classification error is three times the cost of another kind of classification error, not their actual values.

Let's try an example of changing the prior so that the reasoning is clear. Suppose for the prison data one were to let the data determine everything. Then, the empirical prior distribution is about .8 for no misconduct and about .2 for misconduct. The cost of a false negative or a false positive is taken to be 1.0.

Suppose we want the cost of a false negative to be twice the cost of a false positive; the 1 to 1 ratio would be 1 to 2. For no misconduct, we let $\pi_1 \times 1.0 = .80 \times 1.0 = .80$. For misconduct, we let $\pi_2 \times 2.0 = .20 \times 2.0 = .40$. These values need to be normalized so that as probabilities they sum to 1.0. Normalizing π_1^*, we compute $(4/5)/(4/5 + 2/5) = .67$. Normalizing π_2^*, we compute $(2/5)/(4/5 + 2/5) = .33$.

Thus, a 1 to 2 cost ratio for a false positive to a false negative can be imposed using for the prior distribution .67 and .33. There is no need to change the values in $L(i, j)$, which in effect, still have off-diagonal cost elements $L(i \neq j) = 1$. Finally, the same results may be obtained by using Eq. 3.11 with $\pi_1 L_1^* = .80 \times 1.0$ and $\pi_2 L_2^* = .20 \times 2.0$.

It would also be handy if analogous procedures were available for categorical response variables with more than two response categories. However, with more than two response categories, there is more than one cost ratio and often, more adjustments are needed than can be properly captured in a revised prior distribution. We will return to this matter in subsequent chapters.

There still is more to the story. Adjusting the prior only affects one feature of the fitting process, and as Fig. 3.7 shows, terminal nodes are also affected by various conditional proportions in the data. If the goal is to have a certain cost ratio in the off-diagonal cells of a *confusion table for the entire tree*, there is absolutely no guarantee that altering the prior by Eq. 3.8 will get the job done.

In practice, therefore, the prior is a tuning parameter used to arrive at desirable off-diagonal cell counts in a confusion table for the entire tree. For example, if the goal is to have in a confusion table a ratio of 5 to 1 for false negatives versus false positives, the prior calculated by Eq. 3.11 may have to be computed with values for L_i^* and L_j^* that are very different from 5 to 1. Some trial and error will be required. Examples are provided later.

Treating the prior (or loss matrix) as a means to arrive at an acceptable cost ratio in a confusion table may seem a bit unprincipled. But this can be fully consistent with Breiman's algorithmic perspective introduced in Chap. 1. As will become more apparent in the pages ahead, statistical learning and machine learning are *procedures* motivated by fitted values that perform well out-of-sample. Getting a responsive cost ratio is part of that performance. Stated only a bit too cavalierly, in this setting, the ends justify the means.

To summarize, when the CART solution is determined solely by the data, the prior distribution is empirically defined, and the costs in the loss matrix are the same for all classification errors. Equal costs are being assigned even if the data analyst makes no conscious decision about them. Should the balance of false negatives to false positives in a confusion table be unsatisfactory, that balance can be changed. The prior distribution can be altered, and with some trial and error, a more satisfactory cost ratio usually can be obtained. An example follows shortly.

3.6 Pruning

With the discussion of costs behind us, we can now return to the problem of overly complex trees and what can be done. Recall that setting a minimum sample size for each terminal node is one strategy. In *rpart()*, the relevant tuning parameter is *minbucket*. Another strategy is to require some minimum reduction in impurity before a new partitioning of the data is undertaken. In *rpart()*, the relevant tuning parameter is *cp*. Still another strategy to constrain the size of the tree is called "pruning." The pruning process removes undesirable branches by combining nodes that do not reduce heterogeneity sufficiently in trade for the extra complexity added. The process starts at the terminal nodes and works back up the tree until all of the remaining nodes are satisfactory. One can think of the tuning parameters *minbucket* and *cp* as serving to "pre-prune" the tree. The material on pruning that follows draws heavily on Therneau and Atkinson (2015, Sect. 4).

Of late, pruning has not gotten a lot of attention. The problem that pruning addresses is very real. But, as CART has become superseded, pruning has become less salient. Consequently, the discussion of pruning here is relatively short. The main objective is to highlight some important issues raised in the previous chapter that figure significantly in the pages ahead.

For a tree T, recall that the overall risk over K terminal nodes is

$$R(T) = \sum_{j=1}^{K} P(A_j)R(A_j). \tag{3.12}$$

This is the sum over all terminal nodes of the risk associated with each node, each risk first multiplied by the probability of a case falling in that node. It might seem that a reasonable pruning strategy would be to directly minimize Eq. 3.12. What could be better than that? Unfortunately, that would leave a saturated tree untouched. CART would construct enough terminal nodes so that all were homogeneous, even if that meant one node for each observation. With all terminal nodes homogeneous, the risk associated with each would be zero. But, the result would be unstable nodes, serious overfitting of the data, and far too much detail to usefully interpret.

The solution is much like what was seen in the previous chapter. A penalty is introduced for complexity, and under the true model perspective in a level II context, the bias–variance tradeoff reappears. For larger trees with a given sample size, there will be fewer classification errors, implying less bias. But larger trees will have terminal nodes with fewer cases in each, which implies greater instability and hence, greater variance. The trick is to find a sensible balance.

To take complexity into account in CART, a popular solution has been to define an objective function for pruning, called "cost complexity," that includes an explicit penalty for complexity. The penalty is not based on the number of parameters, as in conventional regression, or a function of roughness, as in smoothing. For CART, the penalty is a function of the number of terminal nodes. More precisely, one attempts to minimize

$$R_\alpha(T) = R(T) + \alpha|T|. \tag{3.13}$$

$R_\alpha(T)$ has two parts: the total costs of the classification errors for the tree T, and a penalty for complexity. For the latter, $\alpha \geq 0$ is the complexity parameter playing much the same role as λ in smoothing splines. In place of some measure of roughness, $|T|$ is the number of terminal nodes in tree T.

The value of α quantifies the penalty for each additional terminal node. The larger the value of α, the heavier is the penalty for complexity. When $\alpha = 0$, there is no penalty and a saturated tree results. So, α is the means by which the size of the tree can be determined.

Breiman et al. (1984: Sect. 3.3) prove that for any value of the complexity parameter α, there is a unique, smallest subtree of T that minimizes cost complexity. Thus, there cannot be two subtrees of the same size with the same cost complexity. Given α, there is a unique solution.

In many CART implementations, there are ways the software can select a reasonable value for α, or for parameters that play the same role (Zhang and Singer 1999: Sect. 4.2.3). These defaults are often a good place to start, but will commonly lead to results that are unsatisfactory. The tree selected may make a tradeoff between the variance and the bias that is undesirable for the particular analysis being undertaken. For example, there may be far too much detail to be usefully interpreted.

Alternatively, one can specify by trial and error a value of α that leads to terminal nodes, each with a sufficient number of cases, and that can be sensibly interpreted. Interpretation will depend on both the number of terminal nodes and the kinds of cases that fall in each, so a substantial number of different tree models may need to be examined.

More recent thinking on pruning replaces α with cp. Thus,

$$R_{cp}(T) \equiv R(T) + cp * |T| * R(T_1), \tag{3.14}$$

where $R(T_1)$ is the tree with no splits, $|T|$ is now the number of splits for a tree, and R is the risk as before. The value of cp ranges from 0 to 1. When $cp = 0$, one has a saturated tree. When $cp = 1$, there are no splits. A key advantage over α is that cp is unit free and easier to work with. It can be used to pre-prune a tree, much like the minimum bucket size, or can be tuned with procedures such as cross-validation. Sometimes it can be used in both roles for single analysis.

In practice, whether one determines tree complexity by pre-pruning or pruning (or both) seems to make little practical difference. The goal is to construct a useful classification tree. How exactly that is accomplished is less important as long as the steps undertaken and the various results evaluated are recorded so that the work can be replicated.

The major difficulty is that for a level II analysis, a very aggressive model selection exercise is being undertaken. Beyond the data snooping done by CART itself, there is a search over trees. As already noted many times, all statistical inference can be badly compromised. And the best solution, when feasible, is to have training data,

evaluation data, and test data that can be used in much the same way they were used for tuning smoothers. Absent such data, a level I analysis may well have to suffice.

3.6.1 Impurity Versus $R_{cp}(T)$

At this point, one might wonder again why CART does not use Eq. 3.14 from the start when a tree is grown instead of some measure of node impurity. $R_{cp}(T)$ would seem to have built in all of the end-user needs very directly.

As mentioned earlier, the rationale for not using a function of classification errors as a fitting criterion is discussed in Breiman et al. (1984: Sect. 4.1). As a technical matter, there can be at any given node, no single best split. But perhaps a more important reason is that less satisfactory trees can result. Consider two splits. For the first, there are two nodes that are about equally heterogeneous. For the second, one node is far more heterogeneous than the other. Suppose the two splits reduce impurity effectively the same amount. Yet, minimizing some function of classification errors could lead to the first split being chosen even though the second split was preferable. For the second split, the less heterogeneous node might serve as a terminal node, or might readily lead to one. The more heterogeneous node would be more subject to further partitioning. For the first split, both nodes would likely be partitioned substantially further. In general, therefore, more complicated tree structures will follow.

There can be good subject matter reasons as well. Thinking back to the prison example, finding a single node that was filled almost completely with misconduct cases would be a very useful result, even if the other terminal nodes were quite heterogeneous. In contrast, having all of the terminal nodes with roughly the same proportions of misconduct and no misconduct cases, would not be very useful. The point of the exercise is find subsets of inmates with different proclivities for misconduct. Using node impurity as a splitting criterion will largely prevent this kind of problem.

3.7 Missing Data

Missing data are a common problem for statistical analyses, and CART is no exception. Broadly stated, missing data creates for CART the same kinds of difficulties created for conventional linear regression. There is the loss of statistical power with the reduction in sample size and a real likelihood of bias insofar as the observations lost are not effectively a random sample of the total. That is, the data are now randomly realized from a new joint probability distribution that does include cases like those that are missing from the data on hand.

There is one and only one ironclad solution to missing data regardless of the form of data analysis: don't have any. The message is that it pays to invest heavily in the

data collection so that missing data do not materialize or are very rare. A fallback position is to try to correct the missing data after the data are collected. There are other alternatives to be sure, but all are risky.

Three kinds of missing data mechanisms are commonly considered. For a given variable or sets of variables, there may be no information about certain observations, and that information is "missing completely at random." By "missing completely at random," one means that the mechanism by which the data are lost is equivalent to simple random sampling. A more complicated case is when the information is missing "conditionally at random." By that one means that after conditioning on certain variables, the data are now missing completely at random. For example, the subset of cases that are male may have information about age that is missing completely at random. Finally, the information may be missing systematically. For example, income levels above $100,000 a year may not be recorded regardless of the values of other variables. In a given data set, any combination of these missing data mechanisms may be operating, and the mix shapes what, if any, corrective measures the data analyst employs.

The easiest response to missing data in a multivariate setting is "listwise deletion." If for any case in the data any variable has a missing entry, that case is struck from the data. If the data are missing completely at random, the price is solely a smaller number of observations. A related response in a multivariate setting is "pairwise" deletion. The analysis proceeds with listwise deletion but only for each pair of variables. For example, if for least squares regression some of the predictors have missing values, the cross-product matrix of predictors is assembled for all pairs of predictors based on the number of complete observations for each pair. Many of the computed covariances can be based on different numbers of observations.[11] The most demanding response to missing data is imputation. The basic idea is to replace each missing value with a value that is a useful approximation of what the missing value would have been. There are many different ways this can be done.

Imputation introduces important complications. To begin, one does not ordinarily want to impute values for the response variable. The risk is that artificial relationships between the response and the predictors will be built into the analysis. Consequently, imputation is typically used with predictors only. If there are missing values for the response variable, listwise deletion is usually the prudent choice.

Imputation for predictors alone is hardly problem free. Because the imputed values are rarely the same as what the missing value would have been, measurement error is introduced. Even random measurement error in predictors can bias estimation.[12]

---

[11] In a least squares regression setting, this is generally not a good idea because the covariance matrix may no longer be positive definite.

[12] Consider a conventional regression with a single predictor. Random measurement error (i.e., IID mean 0), will increase the variance of the predictor, which sits in the denominator of slope expression. Asymptotically, $\hat{\beta} = \frac{\beta}{1+\sigma_\epsilon^2/\sigma_x^2}$, where σ_ϵ^2 is the variance of the measurement error and σ_x^2 is the variance of the predictor. The result is a bias toward 0.0. When there is more than one predictor, each with random measurement error, the regression coefficients can be biased toward 0.0 or away from 0.0.

Another difficulty is that an imputed value is typically noisy and that noise should be considered in how the data are analyzed. In practice this can mean imputing missing data several times and taking any random variation into account. Finally, imputed values for a given predictor will often be less variable than the natural variation in that predictor. The reduction in variability can make estimates from the data less precise and invalidate standard errors.

A detailed discussion of missing data is beyond the scope of this book, and excellent treatments are easily found (Little and Rubin 2015). But it is important to consider how missing data can affect CART in particular and what some of the more common responses can be.

3.7.1 Missing Data with CART

Some of the missing data options for CART overlap with conventional practices, and some are special to CART. For the latter, we emphasize the CART options within *rpart()*. In either case, there are statistical and subject-matter issues that must be considered in the context of *why* there are missing data to begin with. The "why" helps determine what the best response to the missing data should be.

Just as in conventional practice, listwise deletion is always an option, especially when one can make the case that the data are missing completely at random. If the data are missing conditionally at random, and the requisite conditioning variables are in the dataset, it is sometimes possible to build those conditioning variables into the analysis. Then one is back to missing data by, in effect, simple random sampling.

A second set of options is to impute the data outside of CART itself. To take a simple illustration, a predictor with the missing data is regressed on complete data predictors with which it is likely to be related. The resulting regression equation can then be used to impute what the missing values might be.

For example, suppose that for employed individuals there are some missing data for income. But income is strongly related to education, age, and occupation. For the subset of observations with no missing income data, income is regressed on education, age, and occupation. Then, for the observations that have missing income data, values for the three predictors can be inserted into the estimated regression equation. Predicted values follow that can be used to fill in the holes for the income variable. At that point, CART can be applied as usual.

A useful extension of this strategy is to sample randomly from the predictive distribution to obtain several imputed values for each missing value. CART can then be applied to the data several times with different imputations in place. One can at least get a sense of whether the different imputed values make an important difference (He 2006). If they do, there are averaging strategies derived from procedures addressed in the next chapter.

A third option is to address the missing data for predictors within CART itself. There are a number of ways this might be done. We consider here one of the better

approaches, and the one available with *rpart()* in R. As before, we only address missing data for predictors.

The first place where missing data will matter is when a split is chosen. Recall that

$$\Delta I(s, A) = I(A) - p(A_L)I(A_L) - p(A_R)I(A_R), \tag{3.15}$$

where $I(A)$ is the value of the parent impurity, $p(A_R)$ is the probability of a case falling in the right daughter node, $p(A_L)$ is the probability of a case falling in the left daughter node, $I(A_R)$ is the impurity of the right daughter node, and $I(A_L)$ is the impurity of the left daughter node. CART tries to find the predictor and the split for which $\Delta I(s, A)$ is as large as possible.

Consider the leading term on the righthand side. One can calculate its value without any predictors and so, there are no missing values to worry about. However, to construct the two daughter nodes, predictors are required. Each predictor is evaluated as usual, but using only the predictor values that are not missing. That is, $I(A_R)$ and $I(A_L)$ are computed for each optimal split for each predictor using only the data available for the given predictor. The associated probabilities $p(A_R)$ and $p(A_L)$ are re-estimated for each predictor based on the data actually present. This is essentially pairwise deletion.

But determining the split is only half the story. Now, observations have to be assigned to one of the two daughter nodes. How can this be done if the predictor values needed are missing? CART employs a sort of "CART-lite" to impute those missing values by exploiting "surrogate variables."

Suppose there are ten other predictors $x_1 - x_{10}$ that are to be included in the CART analysis, and suppose there are missing observations for x_1 only, which happens to be the predictor chosen to define the split; the split defines two categories for x_1.

The predictor x_1 now becomes a binary response variable with the two classes determined by the split. CART is applied with binary x_1 as the response and $x_2 - x_{10}$ as potential splitting variables. As before pairwise deletion is employed. Only one partitioning is allowed; a full tree is not constructed. The nine predictors are then ranked by the proportion of cases in x_1 that are misclassified. Predictors that do not do substantially better than the marginal distribution of x_1 are dropped from further consideration.

The variable with the lowest classification error for x_1 is used in place of x_1 to assign cases to one of the two daughter nodes when the observations on x_1 are missing. That is, a predicted class for x_1 is used when the actual classes for x_1 are missing. If there are missing data for the highest ranked predictor of x_1, the second highest predictor is used instead. If there are missing data for the second highest ranked predictor of x_1, the third highest ranked predictor is used instead, and so on. If each of the variables $x_2 - x_{10}$ has missing data, the marginal distribution of the x_1 split is used. For example, if the split is defined so that $x_1 < c$ sends observations to the left and $x_1 \geq c$ sends cases to the right, cases with data missing on x_1, which have no surrogate to use instead, are placed along with the majority of cases.

This is a reasonable, but ad hoc, response to missing data. One can think of alternatives that might perform better. But the greatest risk is that if there are lots of

missing data and the surrogate variables are used, the correspondence between the results and the data, had they been complete, can become very tenuous. In practice, the data will rarely be missing completely at random or even conditionally at random. Then, if too many observations are manufactured, a new kind of generalization error will be introduced. The irony is that imputation can fail just when it is needed the most. Imputation can be most helpful when there are only a few instances of missing data for single predictors, but the cases with missing data vary over those predictors. Listwise deletion can then decimate the dataset. Imputation may then be a better approach.

But perhaps the best advice is to avoid the use of surrogate variables. The temptations for misuse are great, and there is no clear missing data threshold beyond which imputation is likely to produce misleading results. Imputation of the missing values for the predictors will often be a software option, not a requirement. (But check what the default is.)

Alternatively, one should at least look carefully at the results with and without using surrogates. Results that are substantially different need to be reported to whomever is going to use the findings. There may then be a way to choose on subject matter grounds which results are more sensible. Sometimes neither set will be sensible, which takes us back to where we began. Great efforts should be made to avoid missing data.

3.8 Statistical Inference with CART

As before, there is no statistical inference for a level I analysis. For a level II analysis, one again must make the case that the data are realized from a joint probability distribution of subject-matter interest. Both Y and X are random variables.

Consistent with the discussion in Chap. 1, one should probably give up on the idea of trying to estimate the true tree structure or the true response surface as features of the joint probability distribution responsible for the data. Even if all of the requisite predictors are available in the data, there is no reason to think that the use of step functions coupled with the stagewise fitting procedure will lead to a model that is at least first order correct.

A potentially more viable approach is to estimate an explicit, tree-based, approximation of the true response surface. For reasons already discussed and soon to be elaborated, the partitions themselves, usually displayed as an inverted tree, should be of little interest. Rather, the tree's fitted values can be very important whether those fitted values are the terminal node proportions or the assigned terminal node classes. The estimation target is those fitted values derived from a tree with the same features as the tree grown with the training data.

But there is a new wrinkle. All of the procedures discussed so far do not engage in model selection once the tuning parameters are determined. The defining features of the procedure are then fixed, and the procedure can simply be applied with test

data. To take our earliest example, linear least squares with the selected predictors can be used with the test data, and there are no model selection issues.

CART is fundamentally different. To apply CART to new data means that the data partitioning begins again. Even if all tuning parameters are already determined, extensive model selection is undertaken. Test data do not solve this problem because the test data will be subject to the new partitioning.

Probably the best way to proceed is to reformulate the estimation problem. Interest in the CART response surface approximation remains, but CART is not allowed to partition the test data. Rather, the data partitions determined using the training data and the evaluation data are fixed. This means that the tree structure and terminal nodes are fixed as well. The test data are used solely for prediction. Test data predictor values are dropped down the tree and fitted values computed. Those fitted values serve as asymptotically unbiased estimates of the CART approximation of the true response surface. An honest confusion table can follow. Here are the steps.

1. For a set of potential values for tuning parameters, fit classification trees in the *training* data. Key tuning parameters are likely to be values for the prior and the *cp*.
2. Drop the *evaluation* data down each tree, and compute the fitted values. For classification trees, then construct an evaluation data confusion table. From those tables, performance measures can be obtained.
3. Select a "best" classification tree.
4. Using the selected tree, drop the *test* data down the selected tree and compute the fitted values.
5. Cross-tabulate the predicted outcome classes in the test data by the actual test data response classes. Construct a confusion table from that cross-tabulation. That confusion table provides an asymptotically unbiased estimate of the population confusion table approximation if the population realizations were dropped down the selected tree. The overall misclassification rate provides an estimate of generalization error. If misclassification costs are asymmetric, the different classification errors should be weighted by their relative costs should an overall measure of performance be desired.
6. Apply a nonparametric bootstrap to the *test* data to obtain a family of confusion tables and instructive summary statistics. The distribution of those summary statistics can be used to characterize uncertainty in the estimate of generalization error. Consistent with the earlier discussion of generalization error, the training data and tree structure are fixed. We will explain more about the bootstrap in the next chapter. Also, the last exercise for this chapter addresses the bootstrap with code provided.

Compared to the options offered in the last chapter, the level II formulation is more restricted. Results from the fitting procedure are fixed. They are not re-computed with the test data, so an important source of uncertainty has been neglected. We are back within a generalization error framework discussed in Chap. 1. The training data and the results from the training data are a given. One still must make the case that the

data are realized from a substantively relevant population, but with that credibly accomplished, a level II analysis can proceed.

There have been some very interesting efforts to think about statistical inference with CART when there is a single dataset. In particular, Horthorn and colleagues (2006) provide a rationale and R library *party* for using permutation procedures to help determine which splits are justified couple with tests of the null hypothesis that there is actually no association between the response and the split selected. However, it is not clear how the permutation tests properly take into account the full set of tests undertaken, the dependence between tests undertaken for a given split, and test results from earlier splits. The impact of tuning also seems to have been overlooked. In addition, the realized values of the predictors are treated as fixed (once realized). It is not apparent how one thinks about forecasting applications when the realized x-values are treated as fixed. What does one do with a new realized case having x-values not represented in the training data?

Options from a Bayesian perspective have also been proposed (Chipman et al. 1998, 1999). A key feature of the Bayesian approach is the use of several different kinds of prior distributions. But, if Bayesian approaches are to be taken seriously as inferential tools, the priors must be taken seriously as well. In practice, the prior distributions typically are used to tune the results and are really not very different from CART tuning parameters like *cp* or *minbuckets*. That's fine, but undermines Bayesian statistical inference.

A reasonable overall assessment is that there are many unsolved problems with CART level II analyses. Lots of interesting work is in progress about post-model selection inference more generally, but at the moment, no fully satisfactory solutions exist. This may well explain why most CART applications are effectively level I analyses.

3.9 From Classification to Forecasting

Forecasting with CART was covered in pieces earlier. Here is the material summarized and in one place.

When a classification tree is used for forecasting, a level II analysis is being undertaken, and the goal is usually to forecast a class. The training data and tree grown from the training data are treated as fixed. There are new cases for which the predictor values are known but the response class is not. New cases are "dropped down" the tree. The class previously assigned to the terminal node in which a case lands, is the forecasted class. Alternatively, the forecasting process can be understood getting fitted values from the regression equation characterizing all of the terminal nodes (e.g., Eq. 3.1).

A key assumption is that the new cases whose outcomes need to be forecasted are realized from the same joint probability distribution as the training data. Unless the joint probability distribution is for a well defined, finite population, and the new cases have be selected by probability sampling, there is no way to know if the assumption

is true. In practice, the validity of the assumption is usually a matter of degree that needs to be argued on subject-matter grounds.[13]

In the prison example introduced earlier, the mix of new inmates who are recently admitted must be effectively the same as the mix of inmates in the training data. The prison setting, staffing, and regulations are also unchanged. These issues need to be addressed by individuals who are very familiar with local scene. Sometimes, data may be brought to bear. Is there evidence, for example, of temporal trends in gang affiliations that inmates might bring into the prison? However, the data will usually not have all of the relevant information and whatever the results, it is difficult to know how large changes need to be before the forecasts become insufficiently accurate.

Probably the best strategy is to keep track of forecasting accuracy over time. For each inmate, actual outcomes can be empirically determined for some reasonable amount of elapsed time after intake (e.g., 2 years). Those outcomes can be compared to the forecast at intake. With a sufficient number of inmates, a confusion table can be constructed. Such analyses can be repeated with new inmates over time so that changes in forecasting accuracy are documented. At some point, declines in forecasting accuracy may be sufficient for stakeholders to ask for an update of the classification tree.

In the past, forecasting was an important application for CART. There are now better forecasting tools, for reasons that will soon be apparent. Nevertheless, CART is often a foundation for these better methods and many of the issues resurface.

3.10 Varying the Prior and the Complexity Parameter

Figure 3.8 shows again the tree diagram for the CART analysis of inmate misconduct. Recall that the empirical distribution of the response variable was used as the prior distribution, the costs were assumed to be the same for false negatives and false positives, and the number of terminal nodes was constrained by setting the minimum terminal node sample size.

Recall also the confusion table. It is reproduced here as Table 3.4. One questionable feature of the table was that there were about 21 false negatives for each false positive, implying that false positives were far more costly than false negatives.

Conversations with prison officials indicated that from their point of view, false negatives were worse than false positives. Failing to anticipate inmate misconduct, which could involve fatal violence, was of far greater concern than incorrectly labeling an inmate as high risk. When pushed, a prison official said that the cost of a false negative was at least twice the cost of a false positive. Hence, the earlier analysis got things upside down.

[13] If the population is finite, there technically is no joint probability distribution. There is multivariate histogram. This was discussed in Chap. 1.

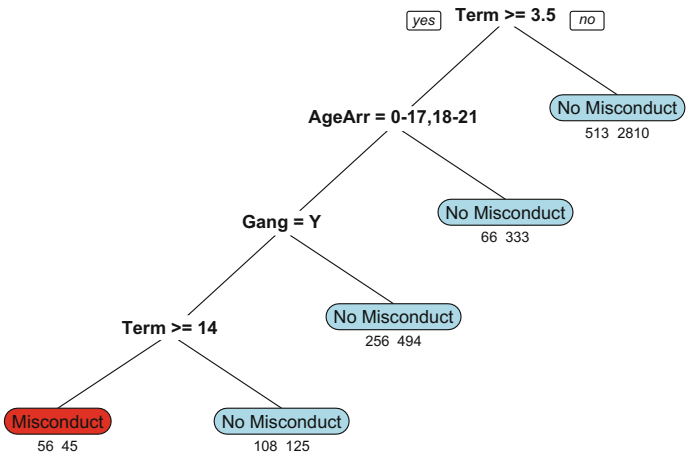

Fig. 3.8 CART recursive partitioning of the prison data with default costs (The *red nodes* represent misconduct, the *blue nodes* represent no misconduct, N = 4806.)

Table 3.4 CART confusion table for classifying inmate misconduct with the default cost ratio (Even with the same data, N's can vary over different analyses because of missing data patterns, $N = 4816$)

	Classify as no misconduct	Classify as misconduct	Model error
No misconduct	3762	45	.01
Misconduct	953	56	.95
Use error	.20	.45	Overall error = .21

Figure 3.9 shows the tree diagram that results when the target cost ratio of false negatives to false positives in the confusion table is 2 to 1. The code is shown in Fig. 3.10.

Setting appropriate values for *prior* and *cp* is explicit tuning. The process can involve some trial and error and in this illustration, the task was complicated by a tree with relatively few terminal nodes. When there are few terminal nodes, the distribution of fitted classes can be lumpy so that getting the empirical cost ratio right in the confusion table may not be possible. With a specified prior of 52 % misconducts and 48 % no misconducts, the confusion table cost ratio was approximately 2.5 false positives for every false negative. False negatives are about 2.5 times more costly. The results do not change materially with cost ratios between approximately 2 and 3.5.

At first, the terminal nodes in Fig. 3.9 may seem a little odd. There is not a single terminal node in which there are more misconduct cases than no misconduct cases and yet, there are three terminal models with misconduct as the fitted class. The reason is that the counts shown in each terminal node are weighted by the new prior.

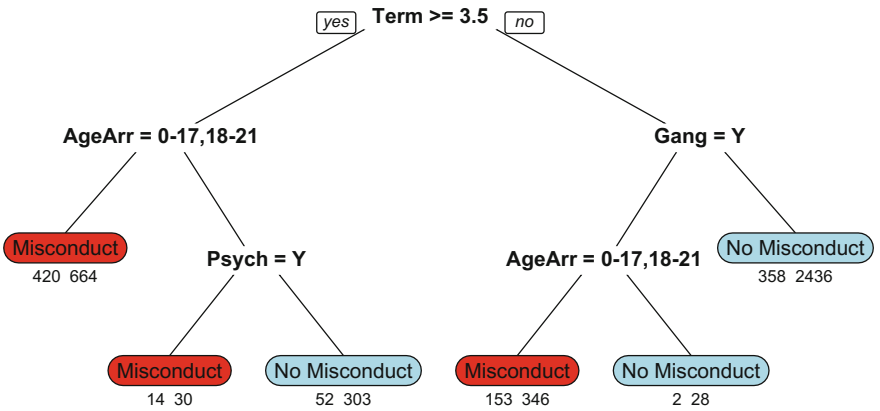

Fig. 3.9 CART recursive partitioning of the prison data with 2 to 1 target cost ratio (N = 4797)

```
# Partition the data
out<-rpart(Fail~AgeArr+Gang+CDC+Jail+Psych+Term,
           data=temp, method="class",
           parms = list(prior = c(.52,.48)),cp=.004)
# Plot a Tree
prp(out,extra=1,faclen=10,varlen=15,under=T
    box.col=c("red","lightblue")[out$frame$yval])
```

Fig. 3.10 R code for the CART analysis of prison misconduct with a 2 to 1 target cost ratio

Misconduct cases constitute about 21 % of the observations but are now treated as if they constitute a little over half. The number of misconduct cases in each terminal node is multiplied by about 2.5 when the fitted class is determined. The ratio of the new prior for misconduct of .52 to the old prior misconduct of .21 is about 2.5.

The partitioning in Fig. 3.9 returns a somewhat different story from the partitioning in Fig. 3.8. Misconduct is the fitted class for young inmates with long nominal terms. Gang membership is not required. For older inmates with long nominal terms, a diagnosed psychological problem leads to a misconduct fitted class. Inmates with shorter nominal terms who are young and gang members are also classified as misconduct cases. All of these results are produced by interaction effects. There are no terminal nodes defined by main effects.

Table 3.5 shows dramatic changes. Increasing relative the costs of false negatives relative to false positives leads to more terminal nodes with misconduct as the assigned class. This is precisely what was intended. As a result, there is a dramatic increase in CART's ability to accurately classify misconduct and in trade, a dramatic reduction in CART's ability to accurately classify no misconduct cases. Overall error is increased a bit, but as already explained, the overall error rate is misleading when the costs of classification errors are asymmetric. The overall error rate treats all clas-

Table 3.5 CART confusion table for classifying inmate misconduct with 2 to 1 target cost ratio (N = 4797)

	Classify as no misconduct	Classify as misconduct	Model error
No misconduct	2767	1040	.27
Misconduct	412	578	.42
Use error	.13	.64	Overall error = .30

sification errors the same. Finally, the use errors have changed as well. When no misconduct is the fitted class, it is incorrect 13 % of the time rather than 20 % of the time. In trade, the use error when misconduct is the fitted class increases from 45 % to 64 %.

The confusion table is constructed from training data only and is no doubt subject to overfitting. For a level I analysis, that comes with the territory. Nevertheless, the changes in use error could have important implications for level II forecasting because the fitted class becomes the forecasted class for new cases when that outcome is not known. By making false negatives more costly, the CART algorithm is, in effect, accepting weaker statistical evidence when misconduct is the fitted class or when misconduct is forecasted; the algorithm is trying harder not to overlook potential misconduct cases. The flip side is that for no misconduct to be the fitted or forecasted class, the algorithm requires stronger statistical evidence. As result, classification and forecasting accuracy (i.e., conditioning on the fitted class) will decline for misconduct outcomes and improve for no misconduct outcomes.

There is nothing mandatory about these results. One might think that there is some sensible way to define an optimal result and a way to produce it. But, even when one has training data, evaluation data and test data, optimality is usually a pipe dream. As a technical matter, several different configurations of tuning parameters values often can lead to very similar results.

The preferred cost ratio complicates matters further. It should be determined by stakeholders, not the data analyst, and depend on decisions that will be informed by the CART tree and confusion table. Such decision-making responsibilities are usually is not included in the job description of a data analyst.

But even with a given cost ratio, how well CART performs is not easily reduced to a single number. For example, some observers might want the tree structure to make subject matter sense. There is no single number by which that can be measured. Other observers may just want a tree structure that is easily interpreted. Again, there is no single number. Still other observers may care little about the tree structure and focus instead on classification accuracy. However, there can be several measures of classification accuracy extracted from a confusion table. In short, there will often be a substantial disconnect between the goals of explanation, classification and forecasting. Even when one dominates, there may well be no single evaluation yardstick to optimize.

There also can be pragmatic complications. In this illustration, prison officials might want to place all inmates forecasted to be problematic in high security settings. But high security incarceration is very expensive (about like Harvard tuition), and if too many misconduct cases are forecasted, the requisite resources will not be available. In effect, there is some threshold above which the number of false positives and true positives breaks the bank. There can be as well legal and political complications if too many false positives are forecasted. Prison officials can be criticized for "over-incarceration."

In short, a reasonable stance is that CART will sometimes provide useful information but rarely definitive guidance. How helpful the information is will depend on the information with which it competes. When decision-makers are operating in an information vacuum, very weak CART results can be valuable. When a lot is already known, strong CART results can be irrelevant. Fortunately, there are much more powerful statistical learning procedures coming.

3.11 An Example with Three Response Categories

In broad brush strokes, there is no formal problem extending CART to three or more response variable categories. But the bookkeeping is much more demanding and getting the cost ratios right is sometimes quite difficult. To see how this happens, we reanalyze the prison data with the three-category response variable: serious and substantial misconduct, some less serious form of misconduct, and no misconduct. About 78 % of the cases have no reported misconduct, about 20 % have minor reported misconduct, and about 2 % have serious and substantial reported misconduct. The 2 % represents very rare cases that ordinarily present a daunting classification challenge. The available predictors are the same as before.

The first hurdle is arriving at sensible cost ratios for classification errors. Prison administrator were concerned about even minor misconduct because it can be a test of staff authority and lead to more serious problems. Still, reported cases of serious and substantial misconduct were of somewhat greater concern. After some back and forth, the following cost ratios were provisionally agreed upon, which can be compared to the results in Table 3.6. The agreement is about as good as one can get without a large number of terminal nodes.

- Misclassifying a "substantial" as a "none" was taken to be about 5 times worse than misclassifying a "none" as a "substantial." In fact, cell 31/cell 13 = 188/31 = 6.1
- Misclassifying a "substantial" as a "some" was taken to be about about 2 times worse than misclassifying a "some" as a "substantial." In fact, cell 23/cell 32 = 117/70 = 1.7
- Misclassifying a "some" as a "none" was taken to be about 2 times worse than misclassifying a "none" as a "some." In fact, cell 12/cell21 = 1181/301 = 2.0

Table 3.6 CART confusion table for classifying inmate misconduct with three outcome classes and target cost ratios ($N = 4736$)

	Classify as none	Classify as some	Classify as substantial	Model error
None	2438	1181	188	.36
Some	301	443	117	.49
Substantial	31	70	37	.73
Use error	.11	.74	.89	Overall error = .39

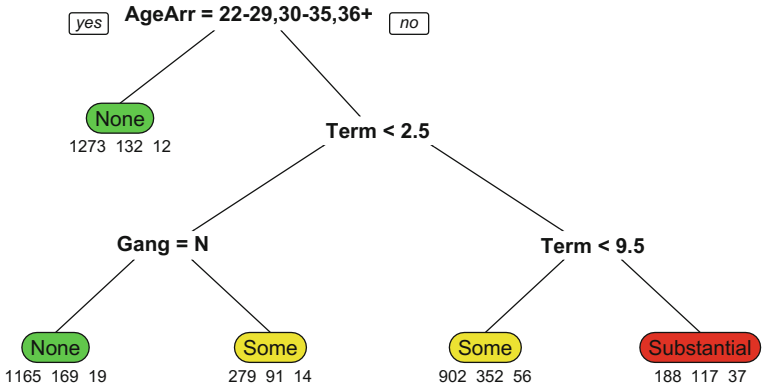

Fig. 3.11 CART recursive partitioning of the prison data with three outcome classes (*Red* terminal nodes represent substantial misconduct, *yellow* terminal nodes represent some misconduct, and *green* terminal nodes represent no misconduct. The order of the numbers below each terminal node is alphabetical: none, some, substantial. $N = 4806$)

```
library(rpart) # Load the CART library
library(rpart.plot) # Load the fancy plotting library
# Partition the data
out<-rpart(Fail3Way~AgeArr+Gang+CDC+Jail+Psych+Term,
           data=temp, method="class",
           parms = list(prior = c(.35,.35,.30)),cp=.01)
#Plot a tree
prp(out,extra=1,faclen=10,varlen=15,under=T
    box.col=c("green","yellow","red")[out$frame$yval])
```

Fig. 3.12 R code for the analysis of prison misconduct

Given reasonable cost ratios in the confusion table (to which we will return), Fig. 3.11 shows the associated classification tree. The R code is provided in Fig. 3.12. With three outcome classes, one reads the classification tree much like before. The main difference is that in each terminal node there is one count for each class arranged in alphabetical order from left to right. As before, the outcome with the largest prior-

weighted count determines the class assigned to a terminal node. But here is where the bookkeeping starts to matter. Because the prior has three classes, the weighting is more complicated. Tuning is more intricate. For example, 2 % of the cases in the empirical priors are reported for serious and substantial misconduct, but cost-sensitive prior arrived at assigns a value of 30 %. Such cases are upweighted by a factor of about 15. At the other extreme, the no misconduct cases account for 78 % of the cases in the empirical prior but only 35 % of the tuned cost-sensitive prior. These cases are down weighted by a factor of about .45. In short, all of the votes in each terminal node are given more or less weight than their raw counts indicate.

With the weighting, Fig. 3.11 has no substantive surprises. Young inmates with longer nominal sentences land in the only terminal node assigned the class of substantial or serious misconduct. Older inmates and younger inmates with shorter terms and no gang associations land in the two terminal nodes that have no misconduct as the assigned class. But if the latter are gang members, they placed in the node assigned some, less consequential misconduct. There is also an apparent break point for nominal sentences less than 9.5. years. Inmates with terms of more than 2.5 years but less than 9.5 years land in a terminal node with the "some" assigned class.

The performance measures in the confusion table are interpreted as before with one important exception. With three outcome categories, there are always two ways to misclassify. For example, when there is no misconduct, the classification is wrong about a third of the time. But the vast majority of those errors are not for cases of serious and substantial misconduct. Perhaps, the performance is better than it first seems. When the class assigned is no misconduct, it is correct nearly 90 % of the time. As before, the cost ratios require that there be strong statistical evidence before a case is classified as a no misconduct. Related reasoning can be applied to the other two outcomes.

Outcomes with more than two classes are common and often very desirable. For example, forecasting how an inmate will do on parole has historically been done so that any form of recidivism counts as a failure. The absence of recidivism is a success. Of late and in response to expressed needs to criminal justice stakeholders, more than two outcome classes are increasingly being used (Berk 2012). One might want to distinguish between new arrests for crimes of violence and new arrests for crimes in which there is no violence. There are then three outcome classes: no arrest, an arrest for a crime that is not violent, and an arrest for a violent crime.

Just as with the earlier, two-outcome analysis of prison misconduct, the results are derived solely from training data. Ideally, one would work with training data, evaluation data and test data. In a forecasting setting, the three data sets can be essential because before forecasting with real consequences is undertaken, one must have a classification tree that is tuned well and provides an honest assessment of out-of-sample performance.

3.12 Some Further Cautions in Interpreting CART Results

Just as for any data analysis procedure, the output from CART always demands scrutiny before substantive conclusions are reached. There are commonly three kinds of potential problems: inappropriate response functions, unstable tree structures, and unstable classifications. All can produce results, which if taken at face value, risk serious interpretive errors.

3.12.1 Model Bias

On this point, we can be brief. Should the goal of a CART analysis be to determine the true response surface in the parent joint probability distribution, disappointment will follow. Even if all of the required predictors are included in the data, there is absolutely no guarantee that the correct function will be discovered. Why step functions? Why various high-order interactions? And no claims can be made that the stagewise partitioning will perform as well as a less constrained approach. In short, it is likely that by the standard of truth, the $\hat{f}(\mathbf{X})$ is substantially wrong.

For a level I analysis, these limitations are unfortunate, but useful insights may still follow. For a level II analysis, estimates of an approximate response surface in the joint probability distribution using the training data alone will also be biased in unknown ways. Because of the predictor selection process, all statistical inference (including confidence intervals and statistical tests) is compromised. However, asymptotically unbiased estimates of the tree approximation of the true response surface can be constructed when there is sufficient test data, and one can accept a perspective in which the training data and the CART partitioning are fixed.

3.12.2 Model Variance

Estimation variance is not an issue for a CART level I analysis because the enterprise is descriptive. Estimation variance is a significant problem for CART level II analyses. Indeed, one of the reasons why CART is no longer a popular data analysis procedure is that it can be very unstable.

The most obvious difficulty is the stability in classes assigned to terminal nodes. Each fitted class is determined by a vote within a single terminal node. No estimation strength is borrowed from other nodes. Nor is there any smoothing over nodes. Consequently, when the number of observations in a node is small, the results of the within-node vote could come out very differently in a new realized dataset. The instability can be especially troubling when the within-node votes are close. With very small changes in the composition of node, the assigned class can change. If the assigned classes are unstable, generalization error will be inflated. As noted earlier,

however, if small samples within nodes is a concern, the problem can be addressed at least in part with tuning. But as also noted, the risk is an increase in bias with respect to the true response surface.

Instability is even a greater problem in the partitioning process. Just as with the assignment of classes to terminal nodes, very small differences based on very few observations can make a critical difference. Moreover, the stagewise process guarantees that when an unstable splitting decisions is made, its consequences cascade down through all subsequent splits. Recall also that although the "best" split is chosen at each stage, there may very little difference in fitting performance between the best split and the second best split.

Just as with conventional stepwise regression, instability in tree structure is exacerbated with predictors that are strongly correlated. Figure 3.13 provides an illustration. We again have a binary outcome represented by A or B, and X and Z as predictors. The tight ellipse in the figure means that the two predictors have a strong linear relationship.

The first split is shown by the red vertical line. The B outcome dominates to the left, and the A outcome dominates to the right. For the partition to the right, two possible partitions are shown. One partition, shown with the green vertical line, splits on Z a second time. The result would be a more complex step function linking Z and the outcome. The other partition, shown with the horizontal yellow line, splits on X. The result would be an interaction effect between X and Z. The two alternatives imply very different tree structures and subject-matter interpretations.

If the vertical green line is chosen, the partition at the upper left favors As 4 to 1. If the horizontal yellow line is chosen, the partition at the upper left favors As 3 to 1. Clearly the difference in impurity is small, and the result of the single, circled A. With a new data realization, a entirely different split will likely be chosen.

Figure 3.14 addresses the same issues when the two predictors are not as strongly related in a linear fashion. The As and Bs are more spread out on the $X-Z$ plane. After the initial split, the next split is determined by a larger number of observations. The difference in impurity between the horizontal and vertical splits no longer depends

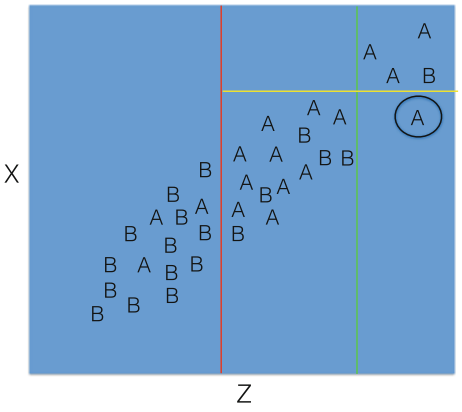

Fig. 3.13 Greater instability in CART partitioning when X and Z are highly correlated (The *red vertical line* is the first split. The *yellow* and *green lines* represent two possible choices for the second split.)

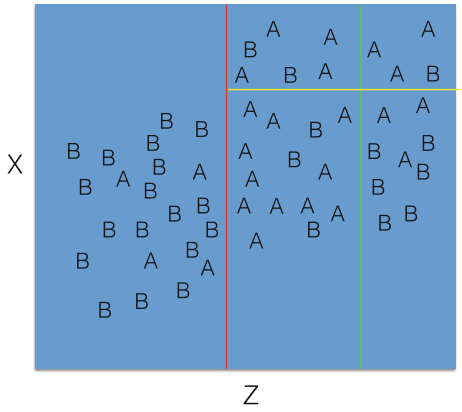

Fig. 3.14 Less instability in CART partitioning when X and Z are not highly correlated

on a single observation. The horizontal split is more homogeneous in A than the vertical split (i.e., 7 out of 10 v. 6 out of 13). Recall that in practice, there are other statistical features of the competing splits to consider, but the point here is with less dependence between X and Z, more stable partitions are likely.

In summary, CART classifications will likely be more stable than the structure of the classification tree. Nevertheless, it is always important to study the distribution of classes within each terminal node. When there are few cases or when the vote is close, the class assigned can be unreliable. Instability in tree structure is more difficult to deal with, but an examination of the sequence of splitting decision can be helpful. (In *rpart()*, that information is easily accessed.) Much as for the terminal nodes, one can consider how many observations determined the split and how much better the winning split was compared to its competitors.

3.13 Regression Trees

The emphasis in this chapter has been on categorical response variables. For reasons that were both pedagogical and practical, classification has been the primary topic. By concentrating on categorical response variables, the full range of fundamental issues surrounding CART are raised.

But CART is certainly not limited to categorical response variables. Quantitative response variables are also fair game. And with the discussion of categorical response variables largely behind us, a transition to quantitative response variables is relatively straightforward. It is possible to be brief.

Perhaps the major operational complication is the kind of regression method to be applied. For example, there are three options in *rpart()* labeled as "anova", "poisson," and "exp." The first is for conventional quantitative response variables such as income. The second is for count response variables such as the number of hurricanes in a given hurricane season. The third is for survival response variables

such as the time from release from prison to a new arrest. Code to be written, options available, and output differ a bit over the three. Beneath the hood, the algorithmic details differ a bit too. The splitting criterion is now some variant on the deviance.

Consider a conventional regression application as an illustration. Node impurity is represented by the response within-node sum of squares:

$$i(\tau) = \sum (y_i - \bar{y}(\tau))^2, \tag{3.16}$$

where the summation is over all cases in node τ, and $\bar{y}(\tau)$ is the mean of those cases. Then, much as before, the split s is chosen to maximize

$$\Delta(s, \tau) = i(\tau) - i(\tau_L) - i(\tau_R). \tag{3.17}$$

Recall that for the split decision when the response is categorical, the Gini impurities are weighted by the proportion of cases in each potential daughter node. For the sum of squares impurities, there is no weighting. But because the sum of squares is a sum over observations, the number of cases in each daughter node matters; there is implicit weighting. One can be more direct by reformulating the problem in units of variances. Impurity for the parent node is represented by the variance of its y-values. Likewise, impurity in the derived nodes is represented by each of their variances. Before they are added, the two variances are weighted by the proportion of cases in their respective node. The proportion weights provide the opportunity to employ different weights. Ishwaran (2015) examined three kinds: no weighting, weighting by the node proportions and weighting by the node proportions squared (called "heavy" weighting). Conventional weighting is said to work best for regression trees when predictors are very noisy.

No asymmetric cost weights can be used because there is no reasonable way to consider false positives and false negatives without a categorical response variable. Ideally, one would alter the way impurity is computed. This is easily done for quantile regression applications, but not for least squares applications. There is interesting work in progress on quantile regression trees (Chaudhuri and Loh 2002; Bhat et al. 2011).

To get the impurity for the entire tree, one sums over all terminal nodes to arrive at $R(T)$. In *rpart()*, regression trees can be pruned using the *cp* tuning parameter. Just as with categorical response variables, one can also "pre-prune" using the *cp*.

The summary statistic for each terminal node is usually the node's mean. In principle, a wide variety of summary statistics could be computed (e.g., the median). And just like for conventional regression, one can compute overall measures of fit. Unfortunately, with all of the searching over possible splits and predictors, it is not clear how to adjust for the degrees of freedom used up. There is no summary measure of fit that can account for this form of overfitting. As usual, the best course of action is to use test data to get an honest assessment of fit but as discussed earlier, this is has it own complications.

Just as in parametric regression, the fitted values can be used for forecasting. Each new observation for which the outcome is unknown is dropped down the tree. The fitted value for the terminal node in which an observation lands become the forecast. Typically, that is the conditional mean.

All of the earlier concerns about CART still apply, save for those linked to the classes assigned. Potential bias and instability remain serious problems. Possible remedies, insofar as they exist, are also effectively the same.

3.13.1 A CART Application for the Correlates of a Student's GPA in High School

Figure 3.15 shows a regression tree for applicants to a large public university. Figure 3.16 shows the code. The response variable is a student's GPA in high school. Predictors include the verbal SAT score, the mathematics SAT score, gender, and household income. Grade point average can be as high as 5.0 because the scoring gives performance in advanced placement classes extra weight. The mean GPA in each node is shown along with the number of cases in that node.[14]

Students with Math SAT scores above 695 and verbal SAT scores above 675 have the highest mean grade point average. Their mean is 4.2. Students with verbal SAT scores less than 565 and math SAT scores below 515 have the lowest mean grade point average. Their mean is 3.4. The two terminal nodes are characterized by interaction effects between the verbal and math SATs. All of the other branches suggest rather complicated and not easily explained relationships. For example, students with verbal SAT scores below 565 and math SAT scores above 515, get an extra boost if they are from families with incomes over $77,000. That students from higher income household do a bit better is no surprise. But why that boost only materializes within a certain set of SAT score values is not apparent. The role of gender is even more obscure. Claims are sometimes made that one of CART's assets is the ease with which a classification or regression tree can be interpreted. In practice, splits based on reductions in impurity often do not lead to results with credible subject-matter interpretations.

There are no confusion tables for quantitative response variables. But one can get a sense of fit performance from Fig. 3.17 produced by *rsq.rpart()*.[15] Both plots show the number of splits on the horizontal axis. On the vertical axis for the plot on the left is the usual R^2. The Apparent R^2 (in blue) is computed from the data used to grow the regression tree. The X relative R^2 (in green) is computed using cross-validation in an effort to get a more honest measure of fit. In this case, the two are almost identical and approach .20 as the number of splits increases. The cross-validation R^2 levels out a bit sooner. There is little improvement in the quality of the fit when the

[14] These data cannot be shared.

[15] The function defaults to only black and white. If you want color (or something else) you need to get into the source code. It's not a big deal.

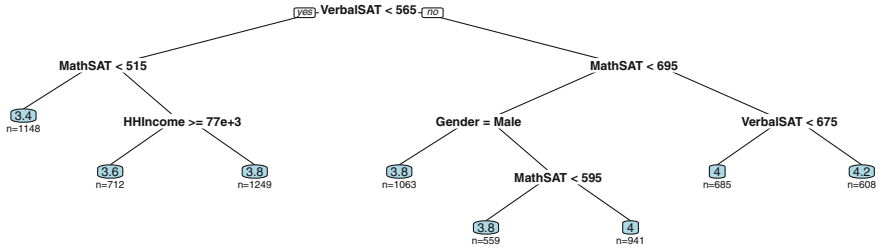

Fig. 3.15 CART recursive partitioning of high school GPA data (The number in each terminal node is the conditional mean. $N = 6965$)

```
# Construct CART Partitions
out<-rpart(GPA~VerbalSAT+MathSAT+HHIncome+Gender,
           data=temp, method="anova", cp=.005)
# Construct a CART tree
prp(out,extra=1,faclen=10,varlen=15,under=T)
# Get Fit Plots
par(mfrow=c(1,1))
rsq.rpart(out)
```

Fig. 3.16 R code for the analysis of the high school GPA data

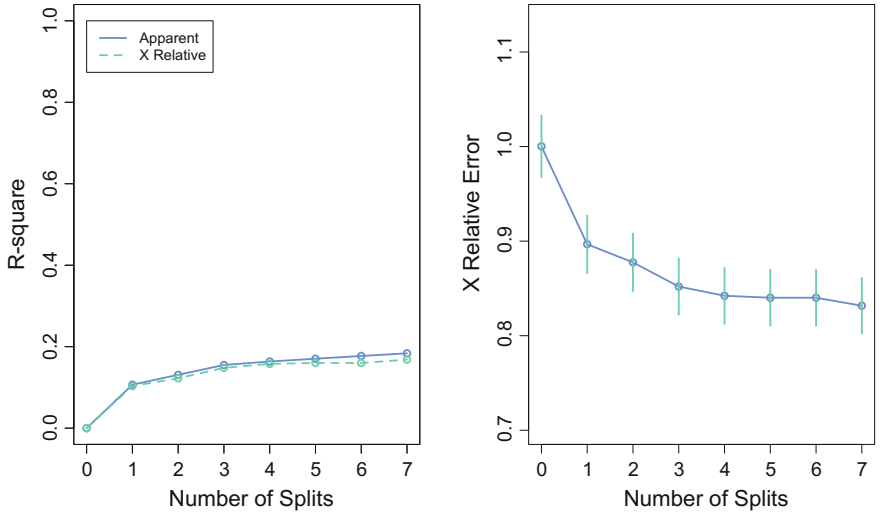

Fig. 3.17 Plots of GPA regression fit. (The *left* figure shows the increase in R^2 with increases in the number of splits. The *solid line* is computed in-sample. The *dashed line* is computed through cross-validation. The *left* figure shows the reduction in relative fitting error with increases in the number of splits. The *vertical lines* are *error bars*.)

number greater than 3. It is also apparent from the similarity of the two lines that the CART search procedures do not seem to have produced problematic overfitting in this instance. This follows from the large number of observations and small number of predictors. Relative to the number of observations, the amount of data snooping is modest. Relatively few degrees have been spent.

The plot on the right side shows the relative improvement in fit. On the vertical axis is the proportional reduction in mean squared error. The vertical lines shows plus and minus one standard error estimated by cross-validation. For this plot, we see that there is little evidence of systematic improvement after the first split, and no evidence of systematic improvement after the 3rd split. The first two errors bars do not overlap at all. Subsequent error bars overlap to varying degrees. The availability of these error bars is helpful, but given all of the problems with CART level II analyses, they should not be taken literally. Moreover, there seems to be no formal justification for using "the 1-SE rule" rather some some other rule (e.g., a 2-SE rule).

For this application, one could probably make a good case for a level II analysis. A relevant joint probability distribution could probably be defined with each observation realized independently. Figure 3.17 suggests that the consequences of data snooping may not be serious. And the number of observations is large enough to make good use of asymptotics. One might want to grow a new regression tree with up to three splits and base any interpretations or applications on that single tree, especially if this were done in test data. The estimation target would be the fitted values from that tree as a feature of the joint probability distribution.

3.14 Multivariate Adaptive Regression Splines (MARS)

Multivariate Adaptive Regression Splines (MARS) can be viewed as another kind of smoother, in the traditions of the last chapter, or as a twist on classification and regression trees. This brief discussion will build on the latter and especially regression trees because MARS assumes that the response variable is quantitative. An excellent and far more extensive exposition of MARS can be found in Hastie et al. (2009: Sect. 9.4).

A key difference between CART and MARS is in the nature of the basis functions used. The MARS formulation is the broadly familiar

$$f(X) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(X), \tag{3.18}$$

where as before, there are M weighted basis functions $h_m(X)$. But each $h_m(X)$ is a product variable composed of linear piecewise splines. This takes a little explaining.

Just as for CART, the basis functions are determined by searching over all predictors and breakpoints. In CART, this leads to a step function. In MARS, this leads to a V-shaped function composed of two linear piecewise splines, with its minimum

value of 0.0 at the breakpoint chosen. The two splines are mirror images of one another; hence the V-shape. Hastie et al. (2009: 322) call the two splines a "reflected pair."

Figure 3.18 is an illustration with a hypothetical breakpoint at 0.5. To the left of breakpoint, the values of the basis function are the positive values of the predictor X subtracted from the breakpoint value. To the right of breakpoint, the values of the basis function are the positive values of the breakpoint subtracted from the predictor X. For CART, the analog to Fig. 3.18 is a step function having a basis function value of 0.0 to the left of the breakpoint and a basis function value of 1.0 to the right. The linear splines formulation does not work for categorical predictors. When there are categorical predictors, MARS goes back to indicator variables just like CART.

Another important difference from CART is that as various breakpoints and predictors are considered, they are *multiplied* by the previous linear piecewise functions already included. The linear basis expansion is undertaken with the sum of products of linear piecewise splines. Exactly how this works need not concern us here. But the use of linear piecewise splines combined with sums of products means that there is no tree representation to inspect.

Like regression trees, each new product is evaluated by the reduction in the error sum of squares. Substantial overfitting can result, so an analogue to pruning is avail-

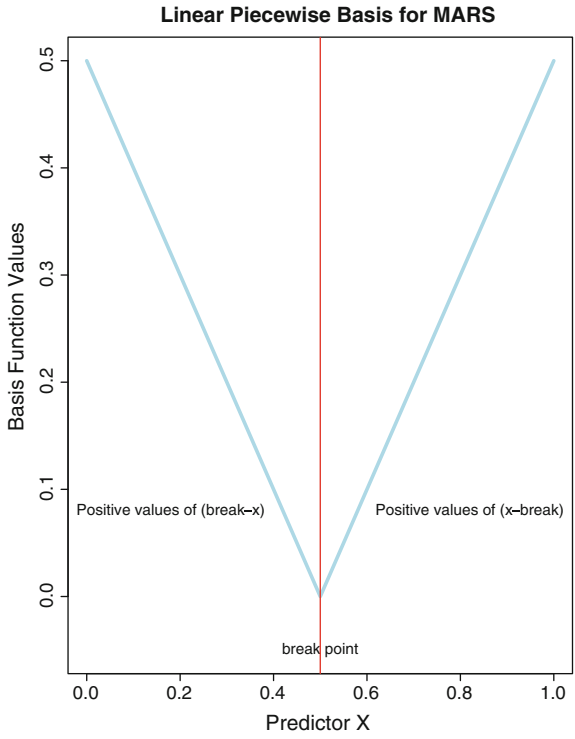

Fig. 3.18 MARS linear basis function with an illustrative break point at 0.5

able. Various tuning parameters are also available to help determine how complex a model is permitted.

MARS output can include the equation actually estimated, and an ANOVA-type partitioning of the explained variance to represent predictor "importance." There are, therefore, parallels to conventional regression. A lot more is said about variable "importance" in the next several chapters.

MARS can be extended to classification tasks. For both classification and regression, it can perform better than CART when step function approximations of $f(\mathbf{X})$ are unsatisfactory. But it has many of the same limitations. MARS has been largely superseded by more recent statistical learning procedures and unlike CART, has not been incorporated into any of these formulations.

3.15 Summary and Conclusions

CART can sometimes be an effective statistical learning tool. It is relatively easy to use, builds directly on familiar regression procedures, does not demand great computing power, and generates output that can be presented in an accessible manner. CART also provides a useful way to introduce the costs of classification errors. However, CART also has some important limitations.

First, if one wants an unbiased estimate of the true $f(\mathbf{X})$, there is no reason to believe that CART's $\hat{f}(X)$ will provide it or that it will even come close. Despite the flexible ways in which CART can respond to data, substantial bias is a real possibility. All fitting procedures are limited by the data available. Key predictors may not be available. But even if the requisite data are available, the use of step functions and the stagewise estimation procedure are significant complications. One can always settle for a level I analysis. A fallback level II strategy is to work with approximate population trees and test data. Asymptotically valid statistical inference, including statistical tests and confidence intervals, can follow.

Second, the splitting decisions can be very unstable. A few observations can in some situations dramatically affect which variables are selected and the precise values used for partitioning the data. Then, all subsequent partitions can be affected. This instability is closely related to overfitting, which can substantially limit the generalizability of the results. The findings from the data examined may not generalize well to other random realizations from the same population (let alone to data from other populations).

Third, for classification, the classes assigned to terminal nodes can be unstable too. When the computed proportions for the terminal node classes are about the same, the movement of just a few cases from one side of a proportion threshold to the other can change the assigned class. It is important to carefully inspect the distribution of proportions in each terminal node to get a sense of how serious the instability problem may be.

Fourth, even moderately elaborate tree diagrams will seriously tax substantive understanding. The problem is not just complexity. CART is trying in a single-

minded manner to use associations in the data to maximize the homogeneity of its data partitions. How those associations come to be represented may have nothing remotely to do with subject matter understandings or how subject matter experts think about those associations. High order interaction effects are dramatic illustration.

Exercises

Problem Set 1

The purpose of this exercise is to provide an initial sense of how CART compares to conventional linear regression when the response variable is quantitative.

1. To begin, construct a regression dataset with known properties:

```
x1=rnorm(300)
x2=rnrom(300)
error=2*rnorm(300)
y1=1+(2*x1)+(3*x2)+error
```

   Apply conventional linear regression using *lm()*. Then apply *rpart()*, and print the tree using *prp()* from the library *rpart.plot()*. Compare the regression output to the way in which the data were actually generated. Compare the tree diagram to the way in which the data were actually generated. Compare how well linear regression and CART fit the data. For CART, use *rsq.rpart()* from the library *library(rpart.plot)* to consider the fit. What do you conclude about the relative merits of linear regression and CART when the $f(X)$ is actually linear and additive?
2. Now, redefine the two predictors as binary factors and reconstruct the response variable.

```
x11=(x1 > 0)
x22=(x2 > 0)
y=1+(2*x11)+(3*x22)+error
```

   Proceed as before comparing linear regression to CART. How do they compare? What do you conclude about the relative merits of linear regression and CART when the $f(X)$ is actually a step function and additive?
3. Under what circumstances is CART likely to perform better than linear regression? Consider separately the matter of how well the fitted values correspond to the observed values and the interpretation of how the predictors are related to the response.

Problem Set 2

The goal of the following exercises is to give you some hands-on experience with CART in comparison to some of the procedures covered in earlier chapters. An initial hurdle is getting R to do what you want. There are lots of examples in the chapter. Also, make generous use of *help()* and I have provided a number of hints along the way. However, I have tried to guide you to results in the least complicated way

possible and as a consequence, some of the more subtle features of CART are not explored. Feel free to play with these in addition. You can't break anything.

Load the data set called "frogs" from the DAAG library. The data are from a study of ecological factors that may affect the presence of certain frog populations. The binary response variable is pres.abs. Use the help command to learn about the data. For ease of interpretation, limit yourself to the following predictors: altitude, distance, NoOfPools, NoOfSites, avrain, meanmin and meanmax.

1. Use logistic regression from *glm()* to consider how the predictors are related to whether frogs are present. Which predictors seem to matter? Do their signs make sense?
2. Using the procedure *stepAIC()* from the *MASS* library with the default for stepwise direction, find the model that minimizes the AIC. Which predictors remain? Do their signs make sense?
3. Using *table()*, construct a confusion table for the model arrived at by the stepwise procedure. The observed class is pres.abs. You will need to assign class labels to cases to get the "predicted" class. The procedure *glm()* stores under the name "fitted.values" the estimated conditional probabilities of the presence of frogs. If the probability is greater than .5, assign a 1 to that case. If the probability is equal to or less than .5, assign a 0 to that case. Now cross-tabulate the true class by the assigned class. What fraction of the cases is classified incorrectly? Is classification more accurate for the true presence of frogs or the true absence of frogs? What is a rationale for using .5 as the threshold for class assignment? What is the cost ratio in the table? What are its implications for an overall measure of classification performance? (Hint: some classifications are likely to be relatively more costly than others. This needs to be taken into account for all confusion tables, not just those from CART.)
4. Using your best model from the stepwise procedure, apply the generalized additive model. You can use *gam()* in either the *gam* or *mvcv* library. Use smoothers for each predictor. Let the procedure decide how many degrees of freedom to use for each smooth. Look at the numerical output and the smoothed plots. How do the results compare to those from logistic regression?
5. Construct a confusion table for the model arrived at through GAM. Once again, the observed class is pres.abs. Use the same logic as applied previously to *glm()* to determine the assigned class. What fraction of the cases is classified incorrectly? Is classification more accurate for the true presence of frogs or the true absence of frogs? How do these results compare to the GLM results? (Hint: don't forget to cost-weight the overall measure of classification accuracy.)
6. Going back to using all of the predictors you began with, apply CART to the frog data via the procedure *rpart()* in the library *rpart*. For now, accept all of the default settings. But it is usually a good idea to specify the method (here, method="class") rather than let *rpart()* try to figure it out from your response variable. Use the *print()* command to see some key numerical output. Try to figure out what each piece of information means. Use *rpart.plot()* to construct a tree diagram. What predictors does CART select as important? How do they

compare with your results from GLM and GAM? How do the interpretations of the results compare?

7. Use *predict()* to assign class labels to cases. You will need to use the help command for *predict.rpart()* to figure out how to do this. Then construct a confusion table for the assigned class and the observed class. What fraction of the cases is classified incorrectly? Is classification more accurate for the true presence of frogs or the true absence of frogs? How do these results compare to the GLM and GAM results? If the three differ substantially, explain why you think this has happened. Alternatively, if the three are much the same explain why you think this has happened.
8. Run the CART analysis again with different priors. Take a close look at the information available for *rpart()* using the help command. For example, for a perfectly balanced prior in *rpart()* you would include *parms=list(prior= c(.50,.50))*. Try a prior of .5 for presence and then a prior of .30 for presence. (For this *rpart()* parameter, the prior probability of 0 comes first and the prior probability of 1 comes second.) What happens to the ratio of false negatives to false positives? What happens to the overall amount of cost-weighted classification error compared to the default?
9. Using Eq. 3.11 set the prior so that in the confusion table false negatives are ten times more costly than false positives (with pres.abs = 1 called a "positive" and pres.abs = 0 called a "negative"). Apply CART. Study the output from *print()*, the tree diagram using *rpart.plot()*, and the confusion table. What has changed enough to affect your interpretations of the results? What has not changed enough to affect your interpretations of the results?
10. Construct two random samples with replacement of the same size as the dataset. Use the *sample()* command to select at random the rows of data you need and use those values to define a new sample with R's indexing capability, *x[r,c]*. For the two new samples, apply CART with the default parameters. Construct a tree diagram for each. How do the two trees compare to each other and to your earlier results with default settings? What does this tell you about how stable your CART results are and about potential problems with overfitting.
11. Repeat what you have just done, but now set the minimum terminal node size to 50. You will need the argument *control = rpart.control (minbucket = 50))* in your call to *rpart()*. How do the three trees compare now? What are the implications for overfitting in CART?

Problem Set 3

Here is another opportunity to become familiar with CART, but this time with a quantitative response variable. From the library *car*, load the data set "Freedman." The dataset contains for 100 American cities the crime rate, population size, population density, and percent nonwhite of the population. The goal is to see what is associated with the crime rate.

1. Using the *gam()* from the library *gam*, regress the crime rate on the smoothed values of the three predictors. Examine the numerical output and the plots. Describe how the crime rate is related to the three predictors.
2. Repeat the analysis using *rpart()* and the default settings. Describe how the crime rate is related to the three predictors. How do the conclusions differ from those using the generalized additive model?
3. Plot the fitted values from the GAM analysis against the fitted values from the CART analysis. The fitted values for *gam()* are stored automatically. You will need to construct the fitted values for CART using *predict.rpart()*. What would the plot look like if the two sets of fitted values corresponded perfectly? What do you see instead? What does the scatterplot tell you about how the two sets of fitted values are related?
4. Overlay on the scatterplot the least squares line for the two sets of fitted values using *abline()*. If that regression line had a slope of 1.0 and an intercept of 0.0, what would that indicate about the relationship between the two sets of fitted values? What does that overlaid regression line indicate about how the two sets of fitted values are related?
5. Using *scatter.smooth()*, apply a lowess smoother to the scatterplot of the two sets of fitted values. Try several different spans. What do you conclude about the functional form of the relationship between the two sets of fitted values?
6. For the GAM results and the CART results, use *cor()* to compute separately the correlations between the fitted values and the observed values for the crime rate. What procedure has fitted values that are more highly correlated with the crime rate? Can you use this to determine which modeling approach fits the data better? If yes, explain why. If no, explain why.

Problem Set 4

1. At a number of places, the bootstrap has been mentioned and applied. The basic idea behind the bootstrap is easy enough in broad brushstrokes. But the details can be challenging even for math-stat types because there are important subtleties and many different bootstrap variants. Fortunately, for many of the applications that have been discussed and that will be discussed, uncertainty in some important features of the output can be usefully addressed with procedures available in R. Currently, *boot()* is popular because of its great flexibility. But it has a relatively steep learning curve, and some readers may prefer to write their own bootstrap code.

   The code below produces a bootstrap distribution of the proportion of correct classifications in a CART confusion table using the Titanic data and the outcome class of survival. From the empirical distribution, one can consider the sampling distribution for the proportion correctly classified and construct a confidence interval.

   The code is meant to only be illustrative. As already mentioned, it probably makes little sense to consider a level II analysis for the Titanic data. However, code like this will work for test data should a level II analysis be appropriate. The main difference is that there would be no new CART fit for each bootstrap sample.

The rpart step shown in the code would not be included. Rather, the rpart-object would be computed outside of the function and called when *predict()* was used with test data.
Run the code and consider the output. Then try it with another dataset and test data. Interpret the CART confusion table and the distribution of the proportion of cases correctly classified.

```
# Application of nonparametric bootstrap for CART
library(PASWR)
library(boot)
library(rpart)
data("titanic3")
attach(titanic3)
temp<-data.frame(survived,pclass,sex,age) # Select variables
working<-na.omit(temp) # Remove NAs
detach(titanic3)

# Define the function to be bootstraped
confusion<-function(data,i) # i is the index for the bootstrap sample
{
  working2<-working[i,] # names the bootstrap sample for each i
  out<-rpart(survived~sex+age+pclass,data=working2,method="class")
  preds<-predict(out,data=working2)
  conf<-table(preds[,2]>.5,working2$survived) # Confusion table as usual
  fit<-(conf[1,1]+conf[2,2])/dim(working2)[1] # Proportion correct
  return(fit)
}

# Apply the bootstrap and examine the output
fitting<-boot(working,confusion,R=300) # Look at the object
plot(fitting)
quantile(fitting$t,probs=c(.025,.975))
```