Preface

Over the past 20 years, there has been an incredible change in the size, structure, and types of data collected in the social and behavioral sciences. Thus, social and behavioral researchers have increasingly been asking the question "What do I do with all of this data?" The goal of this book is to help answer that question.

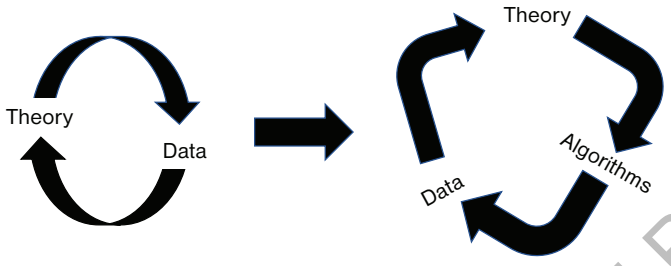
With advances in data collection, there has been a corresponding expansion in the understanding of the complexity underlying the relationships between variables. Nonlinear effects and interactions are now regularly posed as hypotheses, aided by larger sample sizes that afford adequate statistical power. Further, predicting an outcome with only a few covariates of interest, while only assessing linear relationships, is now recognized as severely limiting. While this approach was common in the past, due to smaller dataset sizes and statistical software limitations, computer-assisted data collection and new software have helped overcome such challenges.

In the past, particularly in academic research, certain types of data were only analyzed with specific types of statistical models such as analysis of variance (ANOVA) and linear regression, which were closely aligned with the theoretical motivation underlying the study. However, the advent of novel data collection methods has resulted in new data types (e.g., text), extracted from a variety of sources (e.g., brain imaging, social network), as well as larger collections of traditional survey data. As a result, there is an incredible degree of flexibility in the choice of algorithms. This complicates modern statistical applications, as researchers and practitioners have to contend with an additional dimension, specifically, "Which algorithm or algorithms should I apply?" and "How does this algorithm align with the theoretical motivations of my study?"

It is our viewpoint that in social and behavioral research, to answer the question "What do I do with all of this data?", one needs to know the latest advances in the algorithms and think deeply about the interplay of statistical algorithms, data, and theory.

An important distinction between this book and most other books in the area of machine learning is our focus on theory. To address the interplay

FIGURE1. The complexity of modern data analysis necessitates a transition from primarily focusing on the interplay of data and theory, to now understanding how data, theory, and algorithms are integrated in practice.



of machine learning, data, and theory (see Figure 1), we start with detailing our perspective in Chapter 2 to address the question we often receive when teaching classes or workshops on machine learning, namely: "Can machine learning analyses be incorporated into my traditional confirmatory research?" The follow-up question is often: "Given the exploratory nature of machine learning, how can we be sure our results are trustworthy?" We address this question specifically in Chapter 3 by providing details on a number of cross-validation strategies that help prevent overfitting.

One glance at the table of contents from any of the recently published books on machine learning, data mining, statistical learning, data science, or artificial intelligence reveals a dizzying array of algorithms not detailed in traditional statistics textbooks. This book is different in a number of ways. The first is our aforementioned focus on theory. While Chapter 1 provides an orientation to the book's organization, the primary substance of our book begins with a focus on theory (Chapter 2)—namely, how machine learning fits into research that has traditionally been done from a hypothesis-driven perspective. This is followed by a chapter (Chapter 3) on principles, specifically, how practitioners can apply machine learning algorithms to produce trustworthy results. These chapters set the stage for our discussion of algorithms for univariate outcomes (Chapters 4–6); however, in contrast to other books, we focus largely on regularization and tree-based methods. This better allows us to discuss the integration of these algorithms with complex models that are commonly applied in social and behavioral research, namely, latent variable models. This enables us to provide additional detail on handling measurement error, an extremely important component of survey data, longitudinal data analysis, and the assessment of heterogeneity through the identification of subgroups. Measurement is the focus of Chapters 7 and 8, followed by a discussion of mod-

eling longitudinal data (Chapter 9), and assessing heterogeneity (Chapter 10). Finally, we focus the last two chapters on alternative data types, with an introduction to text analysis (Chapter 11), where we detail the processing of text data and implementing commonly applied algorithms, and social network data (Chapter 12), with an emphasis on network modeling.

Chapters 1 through 6, 9, and 11 have been used as the primary source material for an advanced undergraduate and graduate course. Further, this book can be used as a supplementary reading for courses on regression, multivariate, longitudinal data, and structural equation modeling, among others. While Chapters 4 through 6 have considerable overlap with content found in other books oriented toward supervised learning, Chapter 7 and the subsequent chapters provide a more detailed/advanced account of machine learning methodologies.

Chapters pair a breadth in coverage of methodologies and algorithms with a depth in focusing on fundamental topics, which are detailed at the beginning of each chapter in a "Key Terminology" section to prepare readers for the fundamental concepts of each chapter. Further, we end Chapters 3–12 with a "Computational Time and Resources" section that discusses how to put each method into practice, denoting the key R packages that can be used. Every application of machine learning detailed in this book was programmed in the R statistical environment. While the book does not detail R code, code for all analyses is provided on the book's website. Readers can apply this code to reproduce every example in the book.

Acknowledgments

Each author of this book became interested in machine learning because of mentorship, collaboration, and friendship with John (Jack) J. McArdle, who passed away before this book was completed. Jack was one of the first psychological researchers who became interested in machine learning, writing a host of papers and an edited book on the topic dating back to the early 2010s. Jack provided the inspiration and motivation for each one of us to pursue research in machine learning, spurred by his novel application of machine learning to understanding attrition in longitudinal studies and heterogeneity in longitudinal trajectories. Simply put, without Jack, this book would never have been written.

In addition, we would like to thank the helpful reviewers of the book who guided us in the development of the manuscript. These reviewers were initially anonymous, but they agreed to have their identities revealed now, and we would like to thank them for their insightful feedback: Sonya K. Sterba, Department of Psychology and Human Development, Vanderbilt University; George Marcoulides, Mays Business School, Texas A&M Univer-

sity; and Alexander Christensen, Department of Psychology and Human Development, Vanderbilt University.

Finally, we would like to thank the staff at The Guilford Press, particularly C. Deborah Laughton, whose patience allowed the writing of this book to continue through the COVID-19 pandemic.

5 Decision Trees

Decision trees are one of the core algorithms of machine learning, both as an individual algorithm, as well as forming the basis for many more complex algorithms (Chapter 6). Their primary benefit is the level of interpretability afforded by the visually depicted tree structure along with propensity to identify interaction effects without manual specification. We spend the majority of this chapter focused on describing the tree creation process and how to interpret the tree structure, while ending with details on the specific implementations and nuances in application.

5.1 Key Terminology

- Decision tree. Also commonly referred to as classification and regression trees, decision trees refer to an algorithm that recursively partitions the predictor space to create a nonlinear mapping from predictors to an outcome. The resultant mapping is then depicted as an interpretable tree structure.
- Partition. The process of dividing the dataset into subgroups based on predictor cutoff values.
- Heterogeneity. In contrast to homogeneity, heterogeneity refers to the existence of between-person variability, namely, that assuming that an entire sample represents a singular population is untenable.
- Node. A group of cases in a tree structure. The root node refers to the entire sample that has yet to be partitioned, while the terminal node is the final subgroup at the end of the partitioning process.
- Surrogate split. A split in the decision tree that does not show up in the final tree as it did not produce as much of an improvement in fit as the chosen split, but is used for cases with missingness on the predictor used in the chosen split.

- Pruning. The process of creating a large decision tree, followed by pruning or removing splits to create a smaller, more generalizable tree.

- Variable importance. A summary statistic for how much an individual predictor contributes to explaining variability in the outcome. This is used primarily as a way to understand which predictors are important in algorithms (trees and ensembles; see Chapter 6) that are not as interpretable as linear regression.

5.2 Introduction

Decision trees were first introduced in the 1960s by Morgan and Sonquist (1963). However, the use of the method did not capture the attention of the statistics community until 20 years later with the work of Breiman, Friedman, Olshen, and Stone (1984). First termed *automatic interaction detection*, the purpose of the algorithm was to efficiently identify interaction effects when the number of predictors was large. In large datasets, the sequential testing of a large number of models and possible combinations of variables quickly depreciates the statistical properties of the models (i.e., false positives; Morgan, 2005).

As there are a number of variants to decision tree algorithms, we refer to decision trees as the *umbrella methodology*, and refer to specific algorithms that fall under this umbrella terminology as such. For instance, the most popular variant of decision trees is *classification and regression trees (*CART; Breiman et al., 1984). Note that classification and regression trees is also commonly used to refer to decision trees, while CART refers to a specific algorithm.

Decision trees' popularity can be attributed to their easy-to-understand tree structure, a structure that mimics the way that humans make decisions. We elaborate on this further below. To create a tree structure, the algorithm proceeds as follows: First, divide the predictor space into all possible unique values. Across these unique predictor values, the algorithm iteratively partitions the dataset to determine an improvement in model fit. This involves placing each observation into one of two groups, of which each group member receives the same predicted response. The actual partitioning of the dataset functions differently based on the type of each predictor variable. For quantitative variables, each unique value is tested in order: For example, if predictor values range from 50 to 100, the process is as follows: Everyone with a predictor value equal to 50 versus everyone with a predictor value greater than 50, then everyone with a

predictor value equal to 50–51 versus everyone with greater than 51, until every possible successive value is tested in this manner. Given m unique sorted values, this involves testing $m - 1$ unique splits. For nominal variables, as there is no ordering to the values, all possible groupings are tested. If there are m categories, this involves testing $2^{m-1} - 1$ dichotomous splits. Particularly for predictors with large numbers categories, this can quickly become computationally burdensome.

Predictions derived from the tree structure are then compared to each observation's actual response to calculate error (model misfit). For continuous outcomes, this is generally mean squared error, for categorical it can be accuracy or two alternatives, such as entropy or the gini index. After every possible partition has been tested, and an error is calculated and assigned to this specific partitioning, the split that results in the largest reduction in error is chosen. Effectively, the algorithm has created two new datasets: dataset A contains every observation that adheres to one side of the best splitting rule and dataset B contains all other observations.

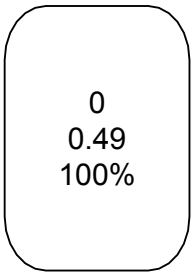
At its core, decision trees are an atheoretical method for creating subgroups. This assumes that identifying heterogeneity (creating subgroups) with respect to the observations in the sample will explain some portion of the outcome.

5.2.1 Example 1

To make the process of creating a tree structure more concrete, we use data from the National Survey on Drug Use and Health, as was detailed in the previous chapter. This dataset was split into a subset to have equal proportions of the suicidal ideation (last 12 months; `SUICTHINK`) outcome, as well as selecting cases that received questions about depression. Predictors included variables that measure feelings of worthlessness (`worthless`; 1 for Yes), whether depression problems interfered with work or personal life (`interfere`; 1-5, 5 = extremely), whether symptoms of depression are felt everyday (`dep`; 1 = Yes), `sex` (1 = Female), `age`, and `race`. Note that in contrast to the logistic regression analyses, it is not recommended to dummy code categorical predictors. Instead, these should be treated as nominal, as it is worthwhile to test a one-versus-all scheme for group differences (as opposed to just a comparison to the reference group).

Decision trees first start with a root node, as displayed in Figure 5.1.

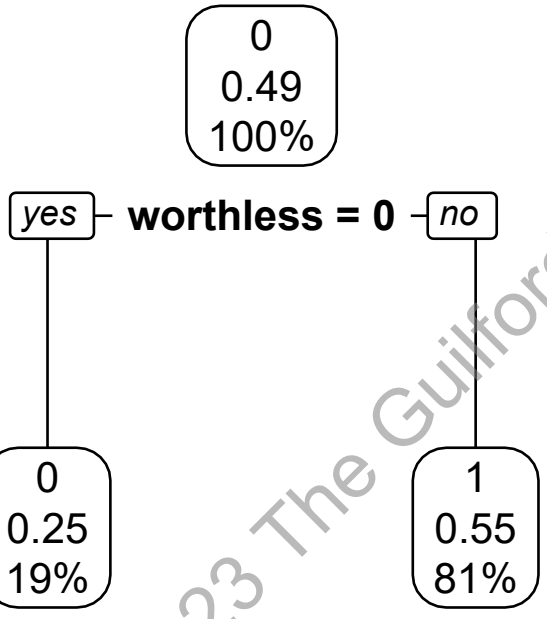
FIGURE 5.1. The root node from the NSDUH dataset. No splits have occurred, as every observation is treated as a homogenous sample. "0" refers to the predicted response for each case in the node, 0.49 is the mean of the outcome, and 100% denotes that the entire sample is contained in this node.



For classification trees, the prediction for each observation reflects the distribution of both classes, which in our example is 0.49, or 49% 0s. In regression trees, this would be the mean of the outcome. In our example, those with a history of suicide attempts (1) and those without a history (0) are roughly distributed 50–50.

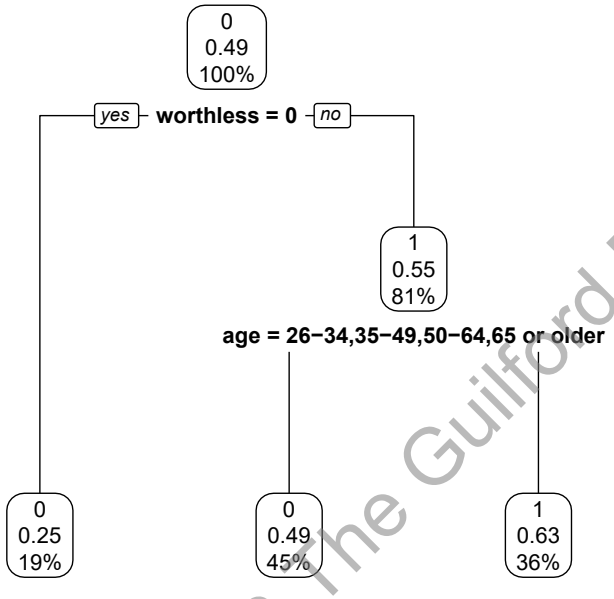
To create a first split, the algorithm tests all possible splits, according to the rules dictated by the predictor variable types (which is important to check the class of the predictors before creating the tree). This resulted in the following split depicted in Figure 5.2.

FIGURE 5.2. First split in the NSDUH dataset. The split occurs between the values of 0 and 1 on `worthless`.



This split occurred between 0 and 1 on `worthless`, with those having a value of 0 (19% of sample) receiving a predicted probability of suicide attempt of 0.25. Note that this predicted probability is simply the class proportion in each node (e.g., 25% of the sample in this node have 1s). Similar to logistic regression, decision trees output predicted probabilities, with class assignment dependent on the probability cutoff (generally 0.5). Cases with a `worthless` value of 1 received a predicted probability of 0.55 (81%) of the sample. Now that we effectively have two subgroups of cases based on `worthless`, the algorithm proceeds by trying to partition each subgroup further. This results in the next set of splits is given in Figure 5.3.

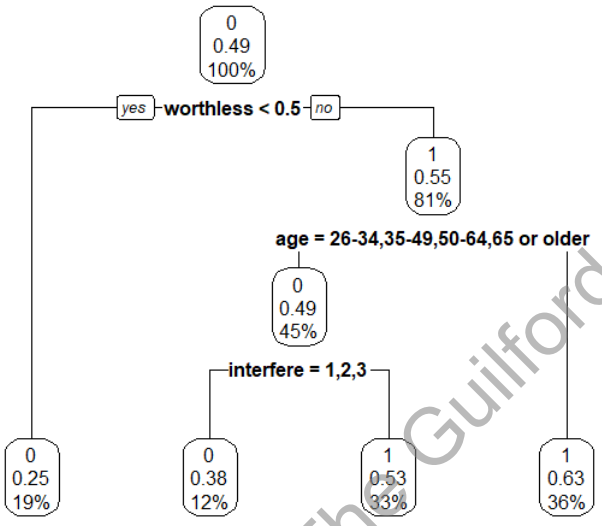
FIGURE 5.3. Second split in the NSDUH dataset. Cases with values of 1 on `worthless` are further split between ages of 18–25 and older.



The next split occurs among the 81% of the sample with a value of 1 on `worthless`. The split, on `age`, partitions those whose `age` falls in the category of 18–25 into one node, and those in the older `age` categories into the other node. These nodes then receive different class predictions based on their predicted probabilities. Note that those that were partitioned with a response of 1 on `worthless` did not receive a further split. Although this could be attributed to a number of aspects, the smaller sample size (19% or 190) could be partially accountable.

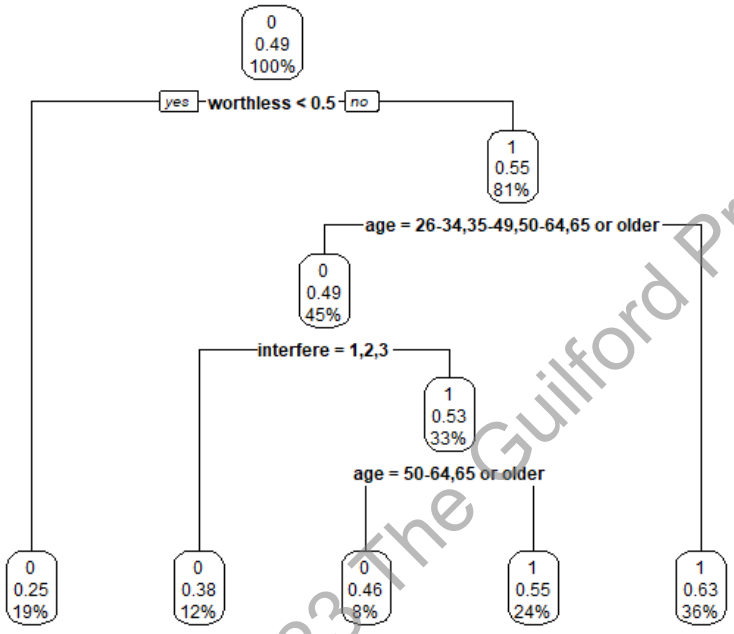
The next step of splitting occurred among those with `worthless` values of 1 and in the older `age` categories, as displayed in Figure 5.4. This split occurred between values of 1–3 and 4–5 on `interfere`. Those with values of 1–3 mostly had outcome values of 0 (0.36), whereas the partition of cases with values of 4 or 5 were approximately evenly split (0.53).

FIGURE 5.4. Example tree with four terminal nodes. The third split occurred among those cases that are older than 18–25, between the values of 3 and 4 on `interfere`.



In this final tree (we set the maximum depth to four), displayed in Figure 5.5, a split occurs among cases that we just placed in the node with `interfere` values of 4 or 5. Cases with `age` values of 50–64 or 65 or older were partitioned from those that had age values of 26–49. Note that this is the second split on age, with cases having an age of 18–25 already placed in a terminal node. For the current split, it results in slightly different predicted probabilities of a suicide attempt, with older individuals having slightly lower probabilities (0.46 versus 0.55).

FIGURE 5.5. Final tree model. The final split occurred among cases with `interfere` values of 4 or 5, occurring between the ages of 25–49 and 50 or older.



5.3 Describing the Tree

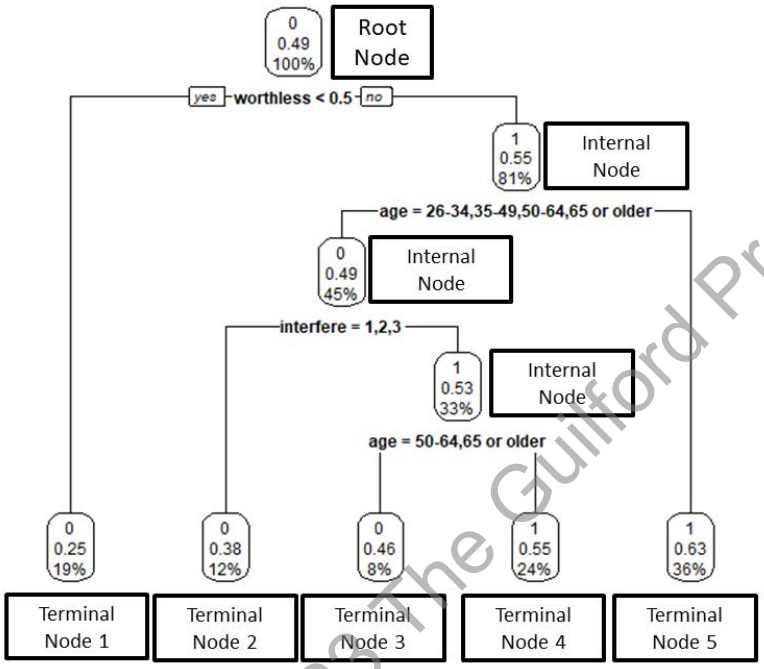
The steps taken to interpret the resulting subgroups can be described as a set of rules, summarized in Table 5.1.

TABLE 5.1. Example Splitting Rules. Note that we could have alternatively included their predicted probabilities, as this is closer to what decision trees attempt to optimize.

Splitting Rule	Prediction	Probability
`worthless` = 0	0	0.25
`worthless` = 1 & `age` > 25 & `interfere` = 1-3	0	0.36
`worthless` = 1 & `age` > 25 & `interfere` = 4-5 & `age` > 49	0	0.46
`worthless` = 1 & `age` > 25 & `interfere` = 4-5 & `age` < 50	1	0.55
`worthless` = 1 & `age` =18-25 & `age` = 18-25	1	0.63

Now that we have a final decision trees model displayed as the rules in Table 5.1 and Figure 5.5, we can be more explicit about the terminology used to describe the tree. This is depicted in Figure 5.6.

FIGURE 5.6. Terminology for each node.



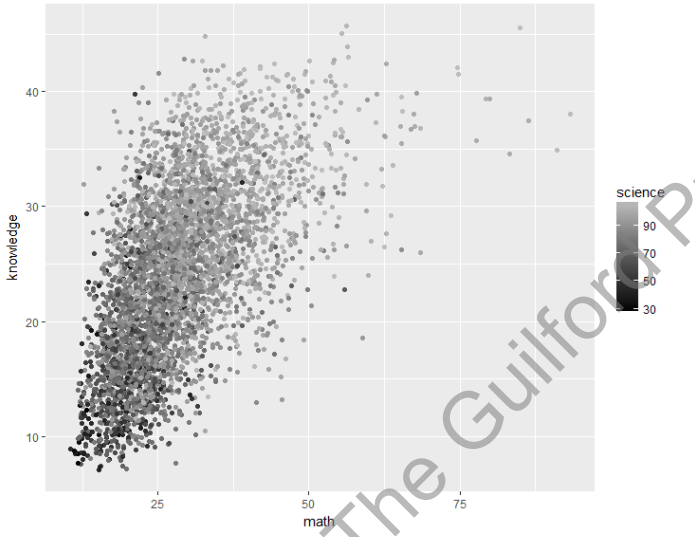
In this figure, the first (topmost) node is termed the *root* node, while each additional node that is then further split is referred to as an *internal* node. Finally, those nodes that are not further split (bottommost) are referred to as *terminal* nodes. We have already described the splitting functions (rules) that dictate where people are placed. Note that there are alternative terminologies used to describe the parts of the tree. One alternative is the use of *parent* and *child* nodes, used in a relative way to describe splitting the cases in the parent node that results in two child nodes. Lastly, the terminal nodes care also referred to as the *leaves* of the tree

5.3.1 Example 2

The same process that can be used for decision trees and categorical outcomes (classification trees) applies to continuous outcomes (regression trees). To portray this and take it a step further in understanding the mapping between predictor and response, we will use data from the ECLS-K. For this example, we will use math and general knowledge scores to predict science scores in the eighth grade. To get a better sense of the rela-

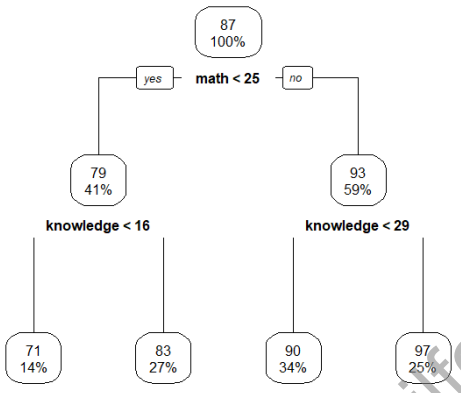
tionships between variables, Figure 5.7 is a scatterplot between math and knowledge, where each point is colored by the science score.

FIGURE 5.7. Three variable scatterplot.



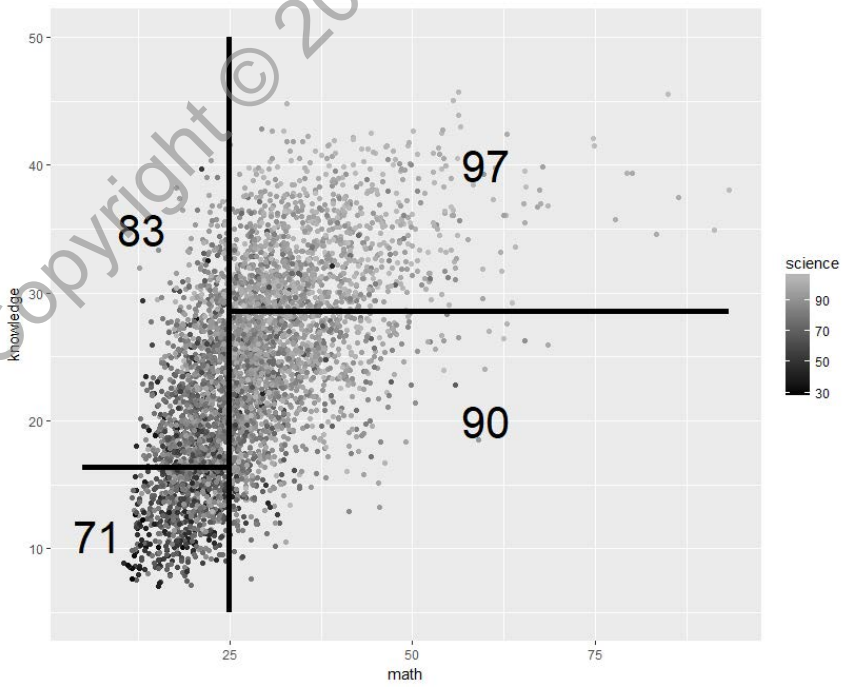
We can see that, in general, as both `math` and `knowledge` increase, so does science. Although the relationship between `knowledge` and `math` looks linear, we would have to plot the marginal relationships between both math and knowledge with science to understand how a linear model would work. Eschewing the linearity assumption using decision trees, with `math` and `knowledge` as predictors of science scores, results in the structure displayed in Figure 5.8.

FIGURE 5.8. ECLS-K decision tree.



The next step in understanding what a tree structure depicts is to see the binning as applied to the scatterplot in Figure 5.9. This is displayed in Figure 5.7.

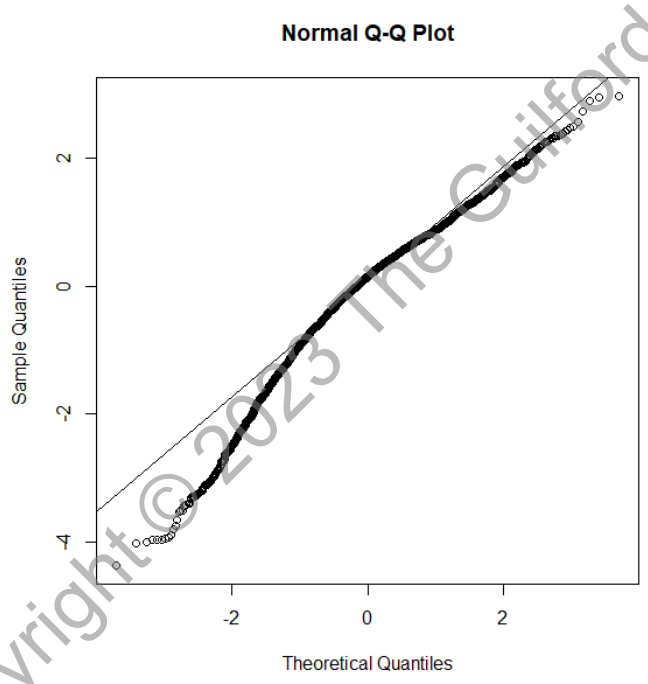
FIGURE 5.9. Decision boundaries.



In this, we can directly map the binning of observations by their predictor values and predicted responses. For instance, the group with cut points of less than 25 on math and less than 16 on knowledge falls into Terminal 1, which occupies the lower left corner of Figure 5.9 and has a predicted science score of 71.

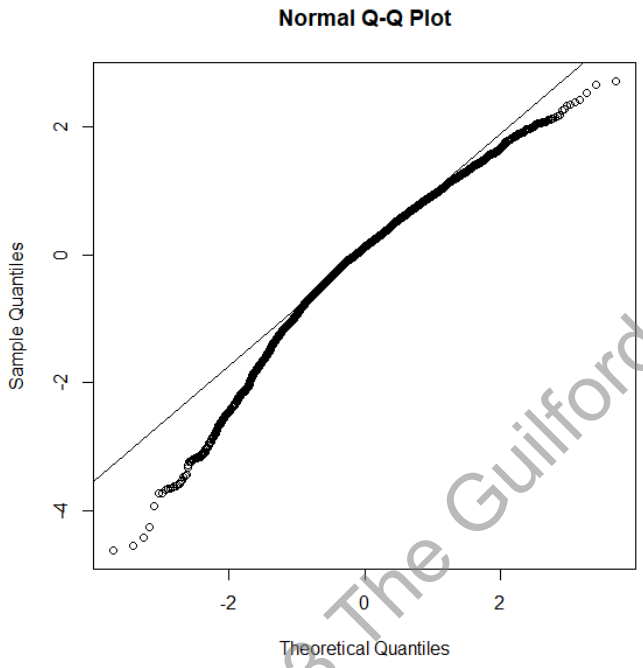
To better understand how well this model fits the data, we also can examine the residuals. The quantile–quantile plot of the standardized residuals is displayed in Figure 5.10.

FIGURE 5.10. Decision tree Q–Q plot.



In this, we can see that for the most part, the model captures the data well except for at high values of science. This can also be seen in Figure 5.10, where the top right partition has a large amount of variability to the predictor values. Given this, we may need additional splits among those with high values in both math and general knowledge. In comparison to using a linear model, which results in the quantile–quantile plot in Figure 5.11, both models seem insufficient to capture the nonlinearity, indicating that a larger tree may be necessary.

FIGURE 5.11. Linear regression Q–Q plot



The linear model has difficulty in capturing observations at both the high and low end of science responses. In examining the R^2 for both the linear model and decision trees, the decision trees model performs only slightly worse, explaining 32.6% of the variance in comparison to 36.4% for the linear model. To better understand how decision trees capture nonlinear relationships, we first reran the analyses only using knowledge as a predictor of science. The resultant tree, depicted in Figure 5.12, can then be translated to the nonlinear function in Figure 5.13.

FIGURE 5.12. Decision trees result only using knowledge.

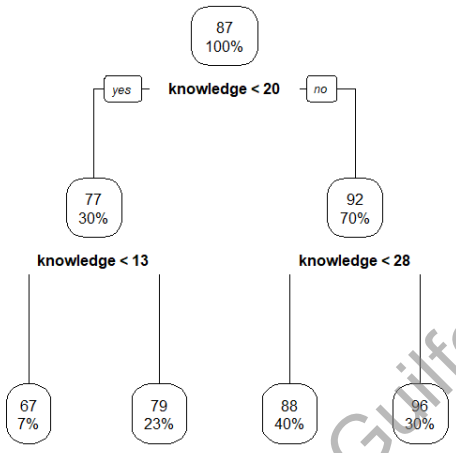
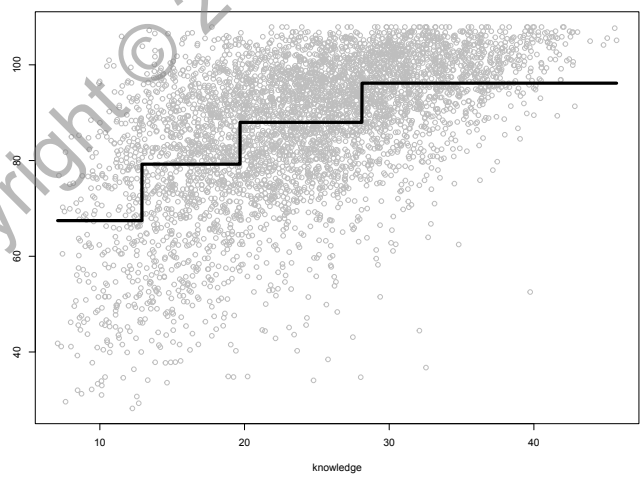


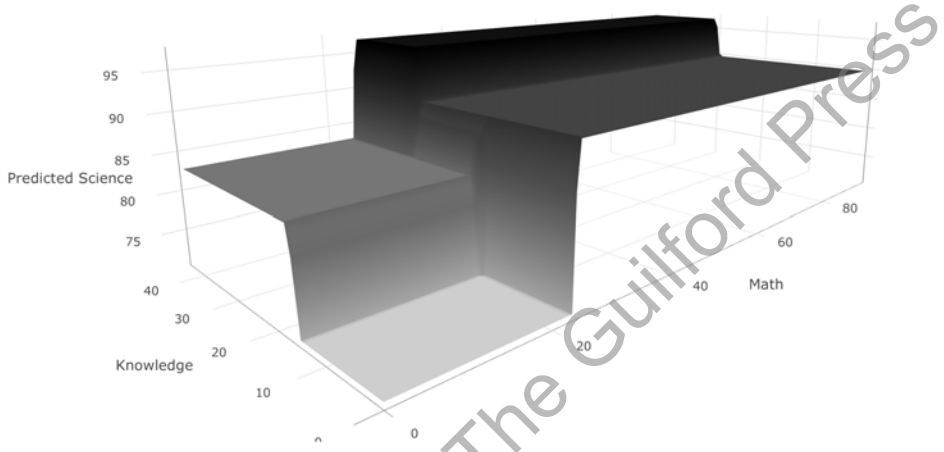
FIGURE 5.13. Nonlinear decision trees prediction line.



In this figure, we see that the group with the lowest predictions for science act as a form of intercept for the model. From here, as knowledge increases, predicted science scores increase according to a step function, with steps located at the cutoffs in the actual tree structure. With more than

one predictor, it becomes more difficult to visualize this relationship. Going back to our original tree, using both knowledge and math as predictors, we can view this step function in three dimensions, as displayed in Figure 5.14.

FIGURE 5.14. Three-dimensional decision surface.



For this predicted surface, we see abrupt changes at coordinates that correspond to the cutoffs in the tree structure. If we grew an even larger tree with both of these predictors, we could image a much more uneven surface.

5.4 Decision Tree Algorithms

Under the umbrella of decision trees fall a number of different algorithms that create tree structures, albeit in different fashions. Although this could in and of itself, be a book, we only provide a general introduction of two algorithms that are available in R.

5.4.1 CART

The term decision trees often refers to the use of classification and regression trees, denoted as the overarching methodology, classification and regression trees (CART; Breiman et al., 1984), and refers to a specific algorithm that falls under the umbrella of decision trees. CART creates binary splits between ordered or unordered predictor categories in order to reduce impurity (heterogeneity). This creates more homogenous groups of observations with respect to the predicted classes. The CART algorithm creates

trees in a greedy fashion, where at each level of the tree, the split that reduces impurity the most is chosen, with no consideration to splits occuring farther down the tree. After the first split (root node), the covariate space is recursively split further until there is no longer an improvement greater than some threshold in the model fit.

Oftentimes this tree structure fits the data too well (overfit), meaning that parts of the tree structure are unlikely to generalize well to alternative samples. One strategy to overcome this propensity is to *prune* the initial tree back to a smaller size. CART is one of the most popular implementations and inspired a host of future implementations that were variants of the original methodology. One of these is the `rpart` package (Therneau, Atkinson, & Ripley, 2015).

5.4.2 Pruning

In choosing a final model, it is common to build a large tree, and then prune back the tree to select a subtree, or a smaller version of the tree that minimizes the cross-validation error. This is done in order to prevent "missing" an important additional split (see Breiman et al., 1984) Note that the largest tree will always have the lowest within-sample error. However, we generally want to choose the tree structure that will generalize the best. This can be accomplished by choosing the model with the lowest average error using k-fold cross-validation (or bootstrapping).

Oftentimes when creating a tree, the tree structure can be larger than is practically interpretable, that is, the size of the tree compromises the generalizability. To test this, we first create a tree without attempting to control the size of it, then proceed to *prune* back the leaves to create a series of submodels (subtree). In pruning, the initial tree is pruned back based on the performance of a complexity parameter (α) that controls the size of the tree. The cost-complexity measure is

$$R_\alpha(T_p) = R(T_p) + \alpha s_p, \tag{5.1}$$

where $R(T_p)$ is the error, and s_p is the number of leaves (terminal nodes) for tree T_p (Breiman et al., 1984). When $\alpha = 0$, then we have the original tree structure. In testing a sequence of increasing values of α, the models incur larger and larger penalties, thus creating successively smaller subtrees. Although the fit of each of the subtrees will be worse on the training data in comparison to the original tree T_0, we would expect better fit according to cross-validation, or with assessing the fit on a test set.

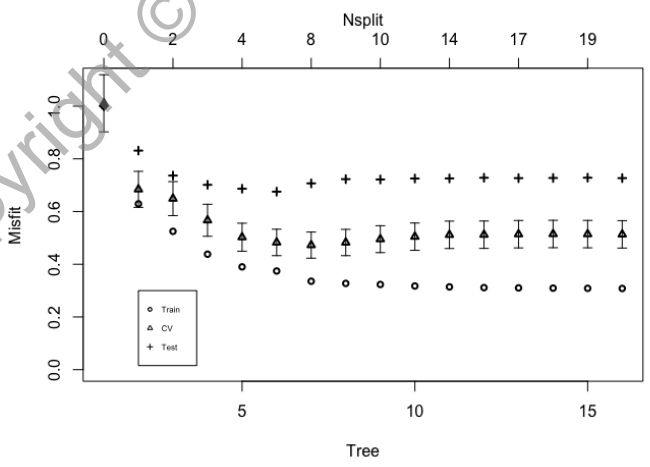
Going back to Example 1 and running the CART algorithm separate from the process described, we get the following tree sizes and errors according to different metrics depicted in Table 5.2.

TABLE 5.2. Table of Complexity Parameters and Fit From Example 1

	CP	Splits	Sample Error	Avg. CV Error	CV Error Std.
1	0.17	0.00	1.00	1.00	0.03
2	0.03	1.00	0.83	0.83	0.03
3	0.02	3.00	0.77	0.84	0.03
4	0.01	5.00	0.74	0.82	0.03
5	0.01	6.00	0.73	0.83	0.03

Although the Sample Error decreases with increasingly larger trees, the Average CV Error decreases dramatically, at first going from a tree with no splits to one split, but then remains consistently in the range of 0.82–0.84 with additional splits. An additional mechanism for understanding cross-validation and the effect of pruning trees is to examine the performance on a separate holdout sample. To demonstrate this process we used the ECLS-K data, this time predicting reading achievement in grade 8. The sample was split into both a train and test set. For each tree, we recorded the fit on the train set, the average fit across the 10 CV folds, and on the test set. This is displayed in Figure 5.15.

FIGURE 5.15. Comparing three forms of assessing fit.



First we examine the fit on the training sample, and we see that as the tree gets larger (Nsplit), the amount of misfit declines monotonically. In contrast, assessing misfit with CV and on the test sample results in a

different conclusion. For both, the misfit declines until a tree size of three splits, and then either only increases in the case of CV or finds another low point at six splits for the test sample, before increasing at larger tree sizes. Using CV we get additional information in the form of the standard deviation of the calculated misfit across the 10 folds. We can use this information to examine *how much* improvement in fit occurs by comparing trees. For instance, although the lowest misfit was achieved at three splits, we can see considerable overlap of the error bars between the two adjacent tree sizes. Similar to the use of the standard deviation of fit using CV in regularized regression, we could also choose the smallest tree within one standard deviation (standard error) of the minimum misfit. This would lead us to choosing the tree with only one split.

5.4.3 Conditional Inference Trees

Conditional inference trees (CTree; Hothorn, Hornik, & Zeileis, 2006) are based on a general theory of permutation tests, by performing a hypothesis test at each node resulting in a *p*-value criterion to help determine whether the tree should stop or keep growing. Using a permutation test to calculate a *p*-value entails comparing a split on the original sample to one using that same split on randomly shuffled response values (e.g. swapping observation 1s and 2s responses). Once completed many times, a *p*-value for the conditional distribution of tests statistics is calculated. This allows for unbiased variable selection as each *p*-value was calculated based on a partial hypothesis. This in effect controls for the scale of each covariate, overcoming the propensity for CART to select variables with larger numbers of response options. As an additional feature that is different than CART, by using a *p*-value to test each split, it negates the use of pruning, as it attempts to control for false positives (splits on noise variables) during the tree construction process.

This algorithm is implemented as the `ctree()` function in the `partykit` package (Hothorn & Zeileis, 2015). The default for the *p*-value criterion is 0.05 (expressed as 0.95, or 1-*p*), although this can manually be altered or tested as a tuning parameter using the `train` function in the `caret` package (Kuhn, 2008). In practice, the trees created by CTree tend to be overly large, making interpretation difficult. Although one could change the *p*-value threshold to something smaller (e.g., 0.01), a new algorithm was proposed that attempts to control the size of the tree without missing important effects, particularly interactions. This algorithm, termed CTreePrune (Alvarez-Iglesias, Hinde, Ferguson, & Newell, 2016), proceeds by first growing a large, saturated tree using CTree by setting a large (0.999) *p*-value criterion. Once a saturated tree is created, the algorithm proceeds

bottom up, recalculating each *p*-value based on a false discovery rate procedure (Benjamini & Hochberg, 1995), an alternative method for controlling familywise error rate. This newly proposed algorithm is implemented in the `dtree` package (Jacobucci, 2017).

5.5 Miscellaneous Topics

5.5.1 Interactions

In linear regression (or other types of regression), it is common to manually enter two-way interactions if it is hypothesized that the effect of one variable may depend on the values of another variable. In Chapter 4, we discussed atheoretical approaches to identify linear interactions in the presence of no a priori hypotheses. As already mentioned, decision trees *automatically* searches for and includes interactions into the resultant tree structure. Despite this, some confusion exists as to what exactly constitutes both a main effect and interaction in a tree structure (see Strobl, Malley, & Tutz, 2009).

An example that uses two variables for splitting and represents two main effects is displayed in Figure 5.16. In the left-hand panel, the first split occurs between males and females. Then within each gender, the same split occurs on depression, resulting in the same predicted increase in the terminal nodes. Examining each internal and terminal node shows that each split results in a four-point increase in the resulting subgroup in comparison to the combined observations. In contrast to this, the right hand side of Figure 5.16 represents an interaction between gender and depression. Notably, the split on depression *depends* on the value of gender.

We can visualize this further with three-dimensional plots of the relationship between both predictors and anxiety. In the left panel of Figure 5.17, the slope of the prediction surface stays constant across values of depression and gender. In contrast, the right side of Figure 5.17 represents an interaction, denoting different effects across the values of both depression and gender. Notably, there is less slope at moderate values of depression, with stronger slopes at both lower and higher values.

In most applications of decision trees, the resulting tree structure will reflect interactions between variables. Particularly with continuous predictors, it is highly unlikely in real data to see the exact same cut-point reflected multiple times in a tree. Additionally, as compared to linear regression, where researchers must manually enter interaction effects, decision trees place no constraints on the interaction effect, while automatically including

FIGURE 5.16. Comparison of tree structures with main effects and an interaction.

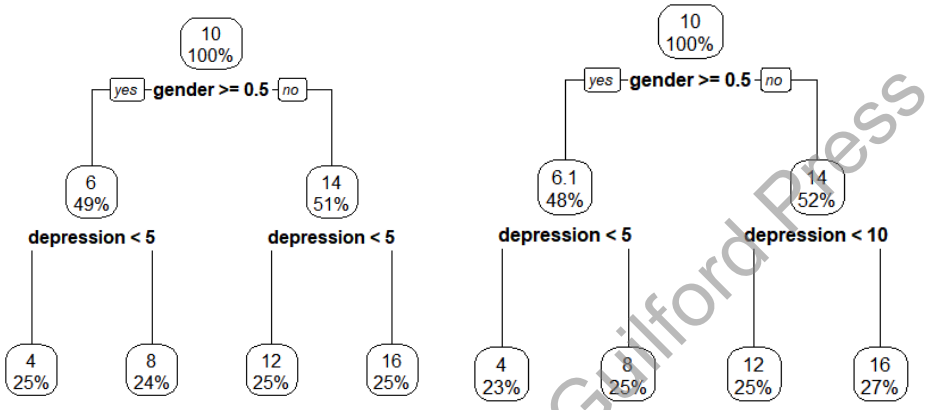
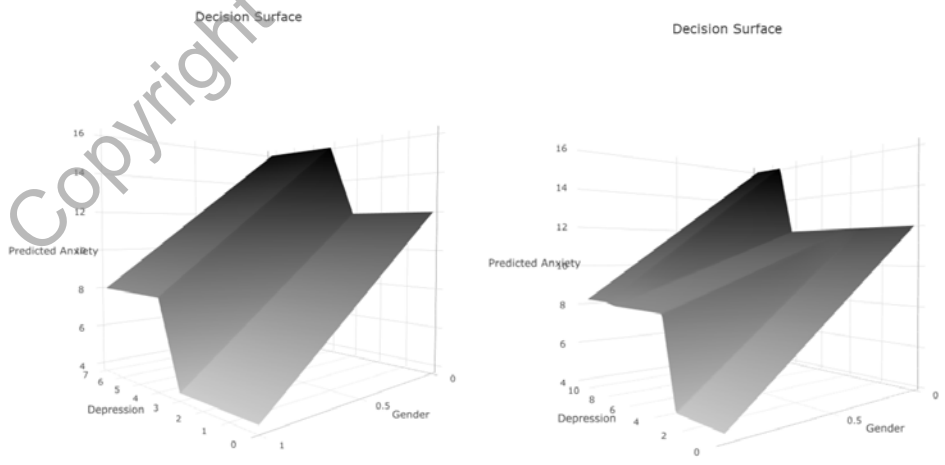


FIGURE 5.17. Comparison of tree structures with main effects (left pane) and an interaction (right pane).



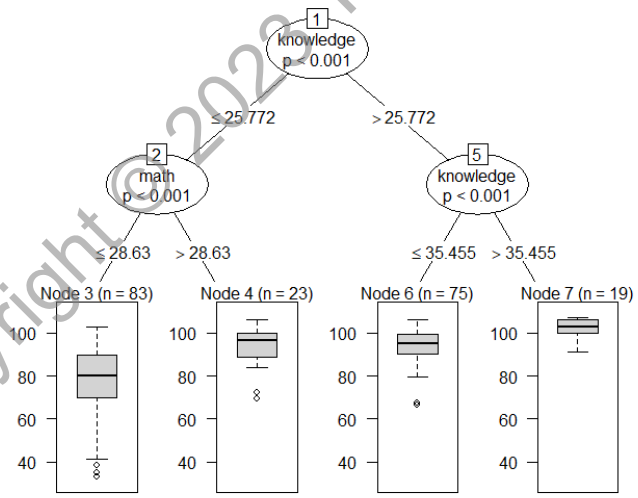
the possibility of interacting effects among predictors. Thus, in line with Berk (2008) most splits represent interactions in decision trees.

5.5.2 Pathways

Beyond just identifying main and interaction effects present in the trees, trees have also been used to describe the various pathways that cases can take to end up receiving similar predictions. More formally, we can describe this as *equifinality* and *multifinality*, following Scott, Whitehead, Bergeman, and Pitzer (2013). In the context of data analysis and tree models, *equifinality* refers to cases having different responses on predictor variables, but similar values on the outcome. On the other hand, *multifinality* refers to cases having similar values on predictor variables, but different outcome values.

As an example of how both of these concepts can be represented in a tree structure, see the tree displayed in Figure 5.18.

FIGURE 5.18. A tree structure demonstrating equifinality. Notice that the interaction between `math` and `knowledge` results in Nodes 4 and 6 with similar distributions on the outcome, despite different pathways.



Note that for this we used the CTree algorithm, which results in a different form of output. In this example, we used a subset of the ECLS-K data, with `math` and `knowledge` scores as predictors of `science` scores. In this, examine Node 4 and 6. Node 4 contains cases that had lower `knowledge` scores (< 25.7), but higher `math` scores (> 28.63) relative to those that ended up in Node 3. Node 6 contains cases that had higher `knowledge`

scores, but not the highest ($\leq$ 35.5). Even though the observations that ended up in Nodes 4 and 6 had different *paths*, as in splits on both `math` and `knowledge`, both groups had very similar expected `science` scores as evidenced by the boxplot in each node. This could be attributed to a higher math score having a compensatory effect that makes up for slightly lower knowledge scores.

Multifinality is more difficult to define in terms of a tree structure. Decision tree algorithms explicitly attempt to identify group differences on the outcome, with respect to the predictors in the model. Instead, one can identify subgroups that have very similar values on multiple predictors, but a split on an additional variable results in a discrepancy in outcome predictions. An example of this can also be seen in Figure 5.18, through examining Node 3. Notice the variability within this subgroup of cases, where the whiskers of the boxplot reach above values of 100 and down to values of 40. Part of this could be attributed to the fact that the splits on both knowledge and math do not fully determine, or cause, science values, hence the degree of uncertainty or variability in each terminal node.

One caveat with regard to describing the tree as resulting in *pathways* of observations is what we discuss later in the section on stability. Use of this level of description of the tree structure should be accompanied by a healthy dose of skepticism regarding this structure as "optimal." Additionally, just because a decision trees algorithm results in splits does not denote a magnitude of effect. To understand this more broadly, one needs to pair explanation with prediction, or describing model performance. This will be touched on in more depth later in the chapter, however, with the tree in Figure 5.18, the predictions from the four resultant subgroups explained 37% of the variability in science. Although far from explaining most of the variance, this might represent an important contribution, depending on performance relative to other methods (linear or lasso regression) or based on results reported in the research area of application.

5.5.3 Stability

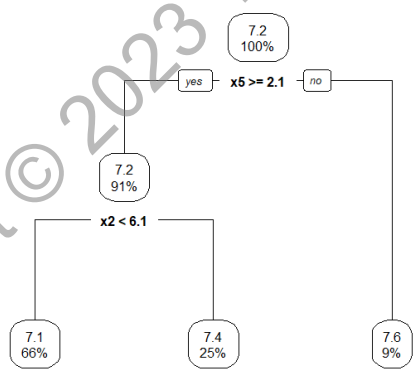
The principle criticism of decision trees is that they are unstable (Breiman, 1996b). This instability occurs when small changes in the dataset produce large changes in the fit of the model (Breiman, 1996b). Instability makes the choice of the final model more difficult, while also imparting doubt into the generalizability of the results, particularly in comparison to a method that produces more stable results. Much of the cause for concern for the instability of decision trees can be attributed to their reliance on binary splits. Problems caused by binary classification is not unique to decision trees. For instance, in mental illness diagnosis, low reliability in rater

disagreement on whether an individual has a disorder or not can be mostly attributed to difficulties in applying categorical cutoffs to disorders that are dimensional in nature (e.g., Brown, Di Nardo, Lehman, & Campbell, 2001). In contrast to comparing whether an individual is diagnosed with a form of schizophrenia or not, low stability (reliability) with decision trees manifests itself as disagreement in the tree structure across datasets that use the same variables.

Example

To make the concept of (in)stability more concrete, we will use the `Holzinger-Swineford` dataset from the `lavaan` package (Rosseel, 2012). With this dataset, we created three trees using the `rpart` package, one on the first 100 respondents, one on the second 100 respondents, and finally on the entire sample (N=301). The resulting tree structures are displayed in Figures 5.19 to 5.21.

FIGURE 5.19. Tree created using first 100 HS observations.



In Figures 5.19 to 5.21, we capture vastly different ideas about what predictors are most important, as well as the functional relationships between the selected predictors and the outcome. The first tree used two variables, $x5$ and $x2$, while the second tree also used two variables, $x2$ and $x3$, and split twice on $x3$. Finally, in the tree created on the entire sample, a new variable was used, $x4$, along with $x2$ and $x3$. Note that although $x2$ was used in each of the three trees, each time was a different cutoff (5.875, 6.125, 7.375). Part of this problem can be attributed to the use of continuous variables as

FIGURE 5.20. Tree created using second 100 HS observations.

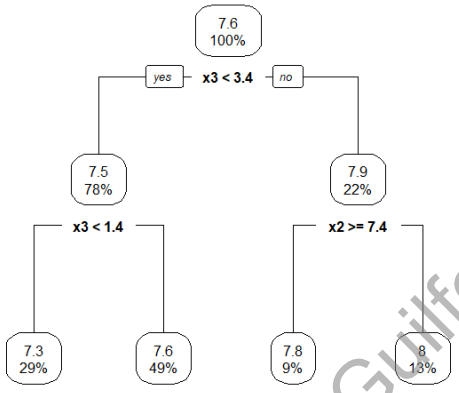
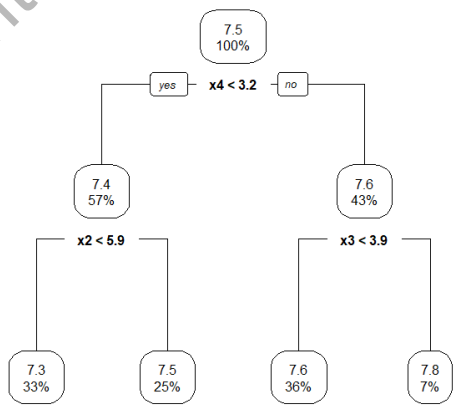


FIGURE 5.21. Tree created using the entire HS sample (N=301).



predictors, as there are more possible splits in comparison to an ordinal or nominal variable in most cases.

It is important to note that variability in model estimates is not unique to decision trees, as a similar process can occur with linear regression, although not to the same extent. In Table 5.3 are the results from applying linear regression to the same three datasets.

TABLE 5.3. Linear Regression Coefficients across the Subsamples of the Holzinger-Swineford Dataset

	First 100		Second 100		Full Sample	
Variable	β	p	β	p	β	p
(Intercept)	7.15	.001	7.27	.001	6.83	.001
$x1$	0.01	.83	.02	.74	.03	.36
$x2$	0.02	.53	-.03	.48	.03	.22
$x3$	-0.02	.60	.19	.001	.03	.31
$x4$	0.09	.15	.09	.12	.06	.11
$x5$	-0.09	.09	-.02	.75	.02	.50
$x6$	0.04	.61	-.11	.09	-.02	.63

Although there is some variability to the parameter estimates across the three models, notably with $x4$ denoted as a significant predictor only in the second sample of 100, for the most part, our conclusions stay the same. However, when we add variable selection to the process, our conclusions can vary greatly. Adding backward selection to choose which variables should be in the model (using the Akaike information criterion to choose a final model), we get additional uncertainty. In the first sample, variables $x4$ and $x5$ are chosen, in the second sample $x3$, $x4$, and $x6$, while in the full sample $x2$ and $x4$. As we can see here, it is the act of variable selection and dichotomous cutpoints that induce instability into the resulting models, not just the use of binary splits for decision trees.

Although there have been many suggestions for overcoming the instability of decision trees, the method that has generated the most research is that of bootstrap aggregating (bagging; Breiman, 1996a). The general idea is that instead of creating one tree, many (hundreds or thousands) are created on bootstrapped samples taken from the original dataset. This topic will be the focus of the following chapter, along with extensions of this concept. The main drawback is that although the creation of a host of trees makes the results more stable, we no longer have a single tree structure to interpret. Because the goal of many research projects is the creation of a single, interpretable tree structure, our focus for the rest of the chapter is only on methods that address stability, while resulting in a single tree.

As a more practical solution, when entering variables as predictors in decision trees, researchers should first determine the level of granularity

of interest in splitting these variables. In some research domains, creating a cutoff between values of 7.345 and 7.346 may be of interest, in others, much less so. If three decimal places is not of interest, then rounding the predictor values will increase the potential for creating stable trees, along with the benefit of decreasing computational time.

To understand the degree of (in)stability to a tree structure, we recommend the use of repeated bootstrap or subsample sampling to determine possible structures. This has been implemented in the `dtree` package with the `stable()` function. Additionally, we recommend pairing decision trees analyses with those from either boosting or random forests, methods that will be covered in the following chapter. Both of these methods create hundreds or thousands of trees, allowing researchers to further examine whether variables used in a singular tree structure are also consistently used when this process is repeated.

As an example of repeatedly creating tree structures, we used the `stable` function on the ECLS-K example with both `math` and `knowledge` as predictors of science scores. In this, we compared the use of linear regression to the CART and CTree algorithms. Linear regression was used to compare performance using CV, while both CART and CTree can be compared with respect to stability and other metrics.

TABLE 5.4. Stability Results from the ECLS-K Example

	nodes	nvar	nsplits	RMSE CV	R^2 CV
lm	-	-	-	11.75	0.37
rpart	15.42 (4.4)	2.00	14.42	11.51	0.39
ctree	22.67 (9.4)	2.00	21.67	11.48	0.40

Some of these results are displayed in Table 5.4. Using 100 bootstrap samples, and 10-fold cross-validation to assess performance on each bootstrap sample, we can see that both decision trees algorithms outperformed linear regression. This gives us some degree of confidence in the utility of nonlinear relationships in examining this set of predictors and outcome. Next, each tree structure was compared to every other across both algorithms to assess stability. Even with rounding cutpoints to the nearest decimal place, each tree structure was unique. This can partly be attributed to the large trees, with an average number of nodes equal to 15.4 and 22.7 for CART and CTree, respectively.

The fact that both predictors are continuous makes the uniqueness of each tree structure more likely. Going back to Figure 5.13 and examining the predictions for just the `knowledge` variable, one can imagine why this level of instability occurs. With random sampling fluctuations, the predic-

tions for each decision trees structure will vary slightly based on how the bootstrap sample varies.

The purpose of this demonstration is not necessarily to dissuade researchers from the use of decision trees, but instead to impart a degree of skepticism with regards to the optimality of a single tree. Instability is not necessarily problematic for inferences within the sample, but instead for the assumption that a tree structure derived on this sample is likely to be the same or highly similar to a tree structure created on an alternative sample. Instead, each tree structure should be interpreted with some apprehension, as alternative structures exist that may represent the data just as well.

5.5.4 Missing Data

Significant research exists that evaluates various strategies for handling missing data in decision trees (e.g., Ding & Simonoff, 2010; He, 2006). Some decision tree methods have options for using surrogate splits in the presence of missing data. Surrogate splits work in the following way. After a primary split is found for a given node, surrogate splits can be found by reapplying the partitioning algorithm to predict the binary primary split. For example, if the primary split on education is between ≤ 12 and > 12 years and greater than 12, then this new binary variable becomes the outcome, with the remaining variables used as predictors. Those variables that perform best in predicting the primary split are retained (default is five in `rpart`) and used for those cases that had missing values on the primary split variable. In the example detailed earlier, if cognitive score is the first surrogate variable, with splits between high (predicted > 12 years of education) and low values (< 12 years of education), then those with high values on cognitive score (and missing on education) would be given predicted values of > 12 years of education in the tree.

The use of surrogate splits is implemented in both the `rpart` and `partykit` packages, however, in `partykit`, surrogate splits is only implemented for cases when both variables are ordered. If researchers wish to use the CART algorithm, then we recommend the use of surrogate splits. However, with other decision trees algorithms, multiple imputation may be the only option for handling missingness. Unfortunately this does not result in a single tree structure, however the K trees, where K is the number of imputations, could be used for high-level inferences, such as which variables were split on and where, as well as the stability of the results.

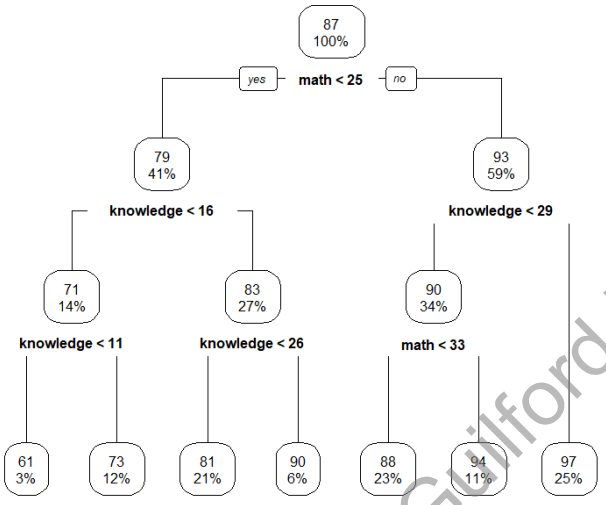
5.5.5 Variable Importance

Similar to linear regression, high correlations among covariates presents problems for decision trees. In the tree building algorithm, at a given split two collinear variables may produce almost identical improvements in fit, but only one can be chosen for a given split. This is analogous to the idea of masking, but in the case of decision trees, it results in one of the variables either not being split on at all, or lower in the tree. To quantify how influential a variable is in predicting the outcome variable, one can also calculate the *variable importance* metric for a given tree. These alternative splits can be used to calculate how much improvement in fit would have occurred for a given variable if the split was chosen. This allows us to quantify the importance of a variable even if it was not split on in the tree. Although there are various ways to calculate this, which is not limited to decision trees, the `rpart` package creates an overall measure of variable importance that is the sum of the fit improvement for all splits of which it was the primary variable (i.e., in the tree), plus for the improvement in fit (adjusted for improvement above baseline) for all splits in which it was a surrogate (Therneau & Atkinson, 1997).

As an example, we used the ECLS-K dataset to create an additional tree. Using both general knowledge and math as predictors of eighth-grade science, we added a third variable, which we created by duplicating each observation's math score and adding a small amount of noise. This simulated math score had a correlation of 0.95 with the original math score. We expected that this would result in only one of the math variables being selected in the tree, but that the variable importance should show roughly similar values for each math score. The resultant tree is displayed in Figure 5.22.

As expected, one of the math scores was not used in the tree (the simulated math score). However, the variable importance metrics were as follows: 39 for math, 33 for knowledge, and 28 for the simulated math score. Note that the variable importance metric is scaled to add up to 100. Despite not showing up in Figure 5.22, the simulated math score receives an importance score that is nonzero. This metric should not be interpreted to too high of precision, as there are only so many splits in a tree, making it difficult to fully account for each variable's importance. In general, only the relative ranking of variables should be interpreted with respect to importance metrics (Strobl, Malley, & Tutz, 2009). In the next chapter, we will build variable importance metrics that are much more robust, as we are using hundreds, if not thousands, of trees to average each variable's effect.

FIGURE 5.22. Tree structure with collinear variables.



5.6 Summary

Decision trees require researchers to make a fundamental shift in the way they interpret results, namely, in that the mapping between predictors and an outcome can be visualized and not reliant on understanding slope coefficients in regression models. Instead, interactions are automatically tested and can be visualized in a number of ways. Although decision trees can in many cases produce more theoretically informative results in comparison to generalized linear models, there are a number of drawbacks to decision trees. Below we detail some of the main points that characterize the advantages and disadvantages of decision trees.

Advantages:

- Are robust to outliers—single misclassifications don't greatly influence the splitting.
- Perform variable selection.
- Generally result in easy-to-interpret tree structures.
- Automatically include interaction effects.

Disadvantages:

- Instability.
- Collinearity presents problems.
- Relatively lower predictive performance.

Just as researchers need to understand the assumptions of linear models, and in what situations these models may not be appropriate, we urge them to not just turn a blind eye and believe that decision trees represent a panacea. As we have noted, there are some research contexts for which tree structures are not appropriate, and there are underlying assumptions that can be violated. Decision trees should instead be viewed as complementary to the use of linear models, and in most cases we recommend using both methods to compare and contrast the different theoretical conclusions that the results represent.

5.6.1 Further Reading

- For further discussion on the instability of decision trees and how to assess them, see Philipp, Rusch, Hornik, and Strobl (2018).
- A number of tutorials have been written on the use of decision trees for various disciplines. Two that we recommend include King and Resick (2014) and McArdle (2012).
- One topic that was not discussed was the use of decision trees in a number of applications as a model for imputing data and for creating propensity scores. See Lee, Lessler, and Stuart (2010) for an evaluation of the use of trees for creating propensity scores, and Carrig et al. (2015) for imputing when datasets are to be integrated.

5.6.2 Computational Time and Resources

Relative to the algorithms that comprise the following chapter, decision trees are relatively quick to fit, even when paired with cross-validation or the assessment of stability. Similarly to regularized regression, decision trees often take on the order of seconds or a few minutes to fit. The runtime is mainly influenced by the number of predictors, and number of possible cutpoints on each predictor. Predictors that are continuous and contain a large number of unique values, along with predictors coded as nominal (requiring one vs. all comparisons), can drastically increase the runtime.

In this chapter, we covered two of the most commonly used R packages for decision trees: `rpart` and `partykit`. Many additional packages are available, either ones that implement quite similar algorithms, such as the `tree` package (Ripley, 2023), or ones that were developed with novel splitting procedures, such as the `evtree` package (Grubinger, Zeileis, & Pfeiffer, 2014). Further, while the `rpart` package is restricted to binary or continuous outcomes, a number of packages have been developed to extend the fundamental tree algorithms to cover alternative outcome types, such as the `rpartOrdinal` package (Archer, 2010). Finally, while most packages also contain functions for plotting the resultant trees, a number of additional packages were built specifically to improve the visualization of trees. The `partykit` package allows for plotting `rpart` trees in the same manner as `ctree()` trees, while `rpart.plot` package (Milborrow, 2022) drastically improves the quality of plots from `rpart` trees.