- Generate hypotheses. For a scientist trying to figure out the causes of diabetes, it can be useful to construct a predictive model, then look to see what variables turn out to be related to the risk of developing this disorder. For instance, you might find that diet, age, and blood pressure are risk factors. Socioeconomic status is not a direct cause of diabetes, but it might be that there an association through factors related to the accessibility of health care. That "might be" is a hypothesis, and one that you probably would not have thought of before finding a function relating risk of diabetes to those inputs.

- Understand how a system works. For instance, a reasonable function relating hours of daylight to day-of-the-year and latitude reveals that the northern and southern hemisphere have reversed patterns: Long days in the southern hemisphere will be short days in the northern hemisphere.

Depending on your motivation, the kind of model and the input variables may differ. In understanding how a system works, the variables you use should be related to the actual, causal mechanisms involved, e.g., the genetics of diabetes. For predicting an output, it hardly matters what the causal mechanisms are. Instead, all that's required is that the inputs are known at a time *before* the prediction is to be made.

8.2 Classifiers

A logistic regression model (see Appendix E) takes a set of explanatory variables and converts them into a probability. In such a model the analyst specifies the form of the relationship and what variables are included. If $\mathbf{X}$ is the *matrix* of our p explanatory variables, we can think of this as a function $f : \mathbb{R}^p \to (0,1)$ that returns a value $\pi \in (0,1)$. However, since the actual values of the response variable y are binary (i.e., in $\{0,1\}$), we can implement rules $g : (0,1) \to \{0,1\}$ that round values of p to either 0 or 1. Thus, our rounded logistic regression models are essentially functions $h : \mathbb{R}^k \to \{0,1\}$, such that $h(\mathbf{X}) = g(f(\mathbf{X}))$ is always either 0 or 1. Such models are known as *classifiers*. More generally, whereas regression models for quantitative response variables return real numbers, models for categorical response variables are called classifiers.

Classifiers are an important complement to regression models in the fields of machine learning and predictive modeling. Whereas regression models have a quantitative response variable (and can thus be visualized as a geometric surface), classification models have a categorical response (and are often visualized as a discrete surface (i.e., a tree)). In the next section, we will discuss a particular type of *classifier* called a *decision tree*. *Regression trees* are analogous to decision trees, but with a quantitative response variable.[1]

8.2.1 Decision trees

A decision tree is a tree-like flowchart that assigns class labels to individual observations. Each branch of the tree separates the records in the data set into increasingly "pure" (i.e., homogeneous) subsets, in the sense that they are more likely to share the same class label.

How do we construct these trees? First, note that the number of possible decision trees grows exponentially with respect to the number of variables p. In fact, it has been proven that an efficient algorithm to determine the optimal decision tree almost certainly does not

[1]The oft-used acronym CART stands for "classification and regression trees."

exist [115].[2] The lack of a globally optimal algorithm means that there are several competing heuristics for building decision trees that employ greedy (i.e., locally optimal) strategies. While the differences among these algorithms can mean that they will return different results (even on the same data set), we will simplify our presentation by restricting our discussion to *recursive partitioning* decision trees. The R package that builds these decision trees is accordingly called `rpart`.

The partitioning in a decision tree follows Hunt's algorithm, which is itself recursive. Suppose that we are somewhere in the decision tree, and that $D_t = (y_t, \mathbf{X}_t)$ is the set of records that are associated with node t and that $\{y_1, y_2\}$ are the available class labels for the response variable.[3] Then:

- If all records in D_t belong to a single class, say, y_1, then t is a leaf node labeled as y_1.
- Otherwise, *split* the records into at least two child nodes, in such a way that the *purity* of the new set of nodes exceeds some threshold. That is, the records are separated more distinctly into groups corresponding to the response class. In practice, there are several competitive methods for optimizing the purity of the candidate child nodes, and—as noted above—we don't know the optimal way of doing this.

A decision tree works by running Hunt's algorithm on the full training data set.

What does it mean to say that a set of records is "purer" than another set? Two popular methods for measuring the purity of a set of candidate child nodes are the *Gini coefficient* and the *information* gain. Both are implemented in `rpart()`, which uses the Gini measurement by default. If $w_i(t)$ is the fraction of records belonging to class i at node t, then

$$Gini(t) = 1 - \sum_{i=1}^{2} (w_i(t))^2 , \qquad Entropy(t) = - \sum_{i=1}^{2} w_i(t) \cdot \log_2 w_i(t)$$

The information gain is the change in entropy. The following example should help to clarify how this works in practice.

8.2.2 Example: High-earners in the 1994 United States Census

A marketing analyst might be interested in finding factors that can be used to predict whether a potential customer is a high-earner. The 1994 United States Census provides information that can inform such a model, with records from 32,561 adults that include a binary variable indicating whether each person makes greater or less than \$50,000 (more than \$80,000 today after accounting for inflation). This is our response variable.

```
library(mdsr)
census <- read.csv(
"http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data",
  header = FALSE)
names(census) <- c("age", "workclass", "fnlwgt", "education",
  "education.num", "marital.status", "occupation", "relationship",
  "race", "sex", "capital.gain", "capital.loss", "hours.per.week",
  "native.country", "income")
glimpse(census)
```

[2]Specifically, the problem of determining the optimal decision tree is NP-complete, meaning that it does not have a polynomial-time solution unless $P = NP$, which would be the most life-altering scientific discovery in the history of human civilization.

[3]For simplicity, we focus on a binary outcome in this chapter, but classifiers can generalize to any number of discrete response values.

```
Observations: 32,561
Variables: 15
$ age            <int> 39, 50, 38, 53, 28, 37, 49, 52, 31, 42, 37, 30,...
$ workclass      <fctr>  State-gov,  Self-emp-not-inc,  Private,  Priv...
$ fnlwgt         <int> 77516, 83311, 215646, 234721, 338409, 284582, 1...
$ education      <fctr>  Bachelors,  Bachelors,  HS-grad,  11th,  Bach...
$ education.num  <int> 13, 13, 9, 7, 13, 14, 5, 9, 14, 13, 10, 13, 13,...
$ marital.status <fctr>  Never-married,  Married-civ-spouse,  Divorced...
$ occupation     <fctr>  Adm-clerical,  Exec-managerial,  Handlers-cle...
$ relationship   <fctr>  Not-in-family,  Husband,  Not-in-family,  Hus...
$ race           <fctr>  White,  White,  White,  Black,  Black,  White...
$ sex            <fctr>  Male,  Male,  Male,  Male,  Female,  Female, ...
$ capital.gain   <int> 2174, 0, 0, 0, 0, 0, 0, 0, 14084, 5178, 0, 0, 0...
$ capital.loss   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
$ hours.per.week <int> 40, 13, 40, 40, 40, 40, 16, 45, 50, 40, 80, 40,...
$ native.country <fctr>  United-States,  United-States,  United-States...
$ income         <fctr>  <=50K,  <=50K,  <=50K,  <=50K,  <=50K,  <=50K...
```

For reasons that we will discuss later, we will first separate our data set into two pieces by separating the rows at random. A sample of 80% of the rows will become the training data set, with the remaining 20% set aside as the testing (or "hold-out") data set.

```
set.seed(364)
n <- nrow(census)
test_idx <- sample.int(n, size = round(0.2 * n))
train <- census[-test_idx, ]
nrow(train)

[1] 26049

test <- census[test_idx, ]
nrow(test)

[1] 6512
```

Note that only about 24% of those in the sample make more than $50k. Thus, the *accuracy* of the *null model* is about 76%, since we can get that many right by just predicting that everyone makes less than $50k.

```
tally(~income, data = train, format = "percent")

income
 <=50K   >50K
  75.7   24.3
```

Pro Tip: Always benchmark your predictive models against a reasonable null model.

Let's consider the optimal split for income using only the variable capital.gain, which measures the amount each person paid in capital gains taxes. According to our tree, the optimal split occurs for those paying more than $5095.5 in capital gains:

```
library(rpart)
rpart(income ~ capital.gain, data = train)

n= 26049

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 26049 6320  <=50K (0.7575 0.2425)
  2) capital.gain< 5.1e+03 24784 5120  <=50K (0.7936 0.2064) *
  3) capital.gain>=5.1e+03 1265   63  >50K (0.0498 0.9502) *
```

Although nearly 80% of those who paid less than $5095.5 in capital gains tax made less than $50k, about 95% of those who paid more than $5095.5 in capital gains tax made *more* than $50k. Thus, splitting (partitioning) the records according to this criterion helps to divide them into relatively purer subsets. We can see this distinction geometrically as we divide the training records in Figure 8.1.

```
split <- 5095.5
train <- train %>% mutate(hi_cap_gains = capital.gain >= split)

ggplot(data = train, aes(x = capital.gain, y = income)) +
  geom_count(aes(color = hi_cap_gains),
    position = position_jitter(width = 0, height = 0.1), alpha = 0.5) +
  geom_vline(xintercept = split, color = "dodgerblue", lty = 2) +
  scale_x_log10(labels = scales::dollar)
```

Thus, this decision tree uses a single variable (`capital.gains`) to partition the data set into two parts: those who paid more than $5095.5 in capital gains, and those who did not. For the former—who make up 0.951 of all observations—we get 79.4% right by predicting that they made less than $50k. For the latter, we get 95% right by predicting that they made more than $50k. Thus, our overall accuracy jumps to 80.1%, easily besting the 75.7% in the null model.

How did the algorithm know to pick $5095.5 as the threshold value? It tried all of the sensible values, and this was the one that lowered the Gini coefficient the most. This can be done efficiently, since thresholds will always be between actual values of the splitting variable, and thus there are only $O(n)$ possible splits to consider.

So far, we have only used one variable, but we can build a decision tree for `income` in terms of all of the other variables in the data set. (We have left out `native.country` because it is a categorical variable with many levels, which can make some learning models computationally infeasible.)

```
form <- as.formula("income ~ age + workclass + education + marital.status +
  occupation + relationship + race + sex + capital.gain + capital.loss +
  hours.per.week")
mod_tree <- rpart(form, data = train)
mod_tree

n= 26049

node), split, n, loss, yval, (yprob)
```

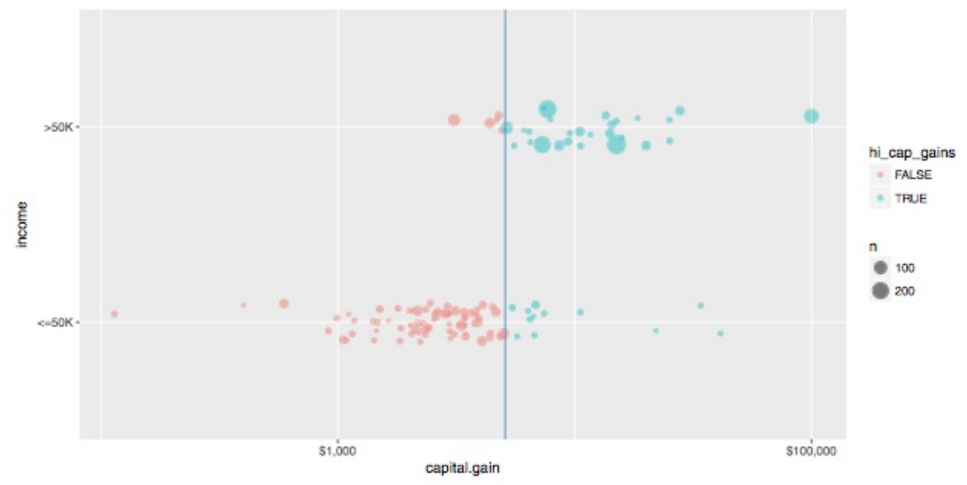

Figure 8.1: A single partition of the census data set using the capital.gain variable to determine the split. Color, and the vertical line at $5,095.50 in capital gains tax indicate the split. If one paid more than this amount, one almost certainly made more than $50,000 in income. On the other hand, if one paid less than this amount in capital gains, one almost certainly made less than $50,000.

```
      * denotes terminal node

 1) root 26049 6320  <=50K (0.7575 0.2425)
   2) relationship= Not-in-family, Other-relative, Own-child, Unmarried
      14196  947  <=50K (0.9333 0.0667)
     4) capital.gain< 7.07e+03 13946  706  <=50K (0.9494 0.0506) *
     5) capital.gain>=7.07e+03 250    9  >50K (0.0360 0.9640) *
   3) relationship= Husband, Wife 11853 5370  <=50K (0.5470 0.4530)
     6) education= 10th, 11th, 12th, 1st-4th, 5th-6th, 7th-8th, 9th,
        Assoc-acdm, Assoc-voc, HS-grad, Preschool, Some-college
        8280 2770  <=50K (0.6656 0.3344)
      12) capital.gain< 5.1e+03 7857 2360  <=50K (0.7003 0.2997) *
      13) capital.gain>=5.1e+03 423    9  >50K (0.0213 0.9787) *
     7) education= Bachelors, Doctorate, Masters, Prof-school
        3573  972  >50K (0.2720 0.7280) *
```

In this more complicated tree, the optimal first split now does not involve capital.gain, but rather relationship. A basic visualization of the tree can be created using the plot() function from the rpart package.

```
plot(mod_tree)
text(mod_tree, use.n = TRUE, all = TRUE, cex = 0.7)
```

A much nicer-looking plot (shown in Figure 8.2) is available through the partykit package, which contains a series of functions for working with decision trees.

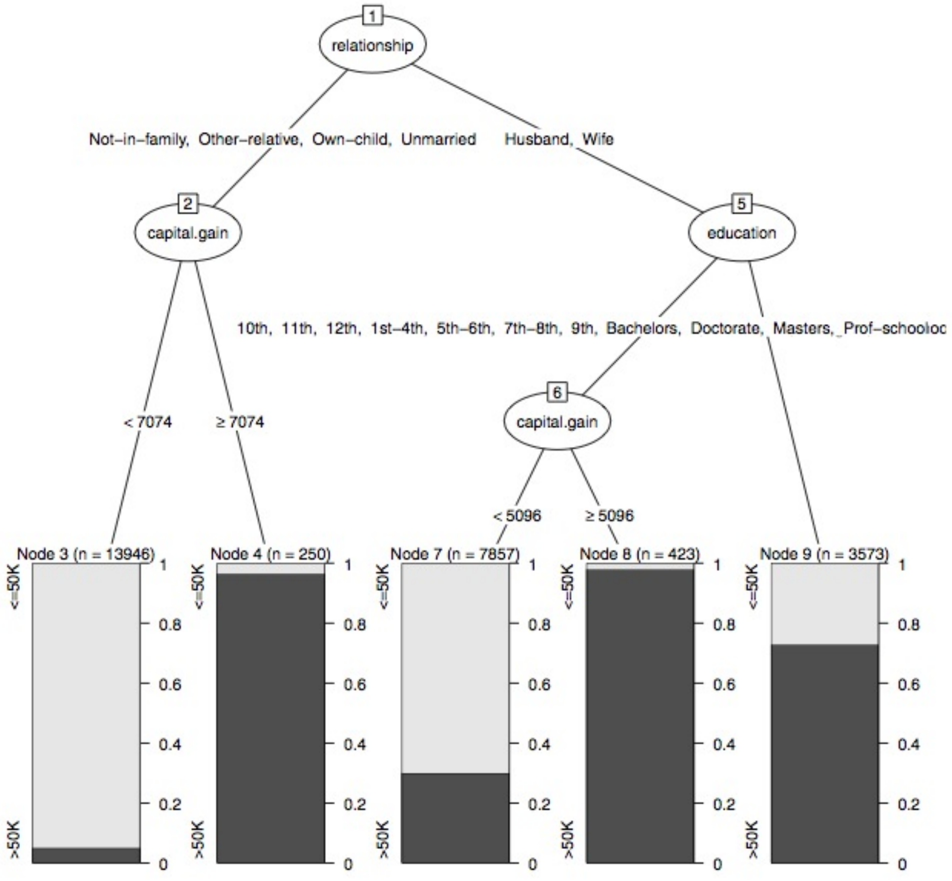

Figure 8.2: Decision tree for income using the `census` data.

```r
library(partykit)
plot(as.party(mod_tree))
```

Figure 8.2 shows the decision tree itself, while Figure 8.3 shows how the tree recursively partitions the original data. Here, the first question is whether `relationship` status is `Husband` or `Wife`. If not, then a capital gains threshold of $7,073.50 is used to determine one's income. 96.4% of those who paid more than the threshold earned more than $50k, but 94.9% of those who paid less than the threshold did not. For those whose `relationship` status was `Husband` or `Wife`, the next question was whether you had a college degree. If so, then the model predicts with 72.8% accuracy that you made more than $50k. If not, then again we ask about capital gains tax paid, but this time the threshold is $5,095.50. 97.9% of those who were neither a husband nor a wife, and had no college degree, but paid more than that amount in capital gains tax, made more than $50k. On the other hand, 70% of those who paid below the threshold made less than $50k.

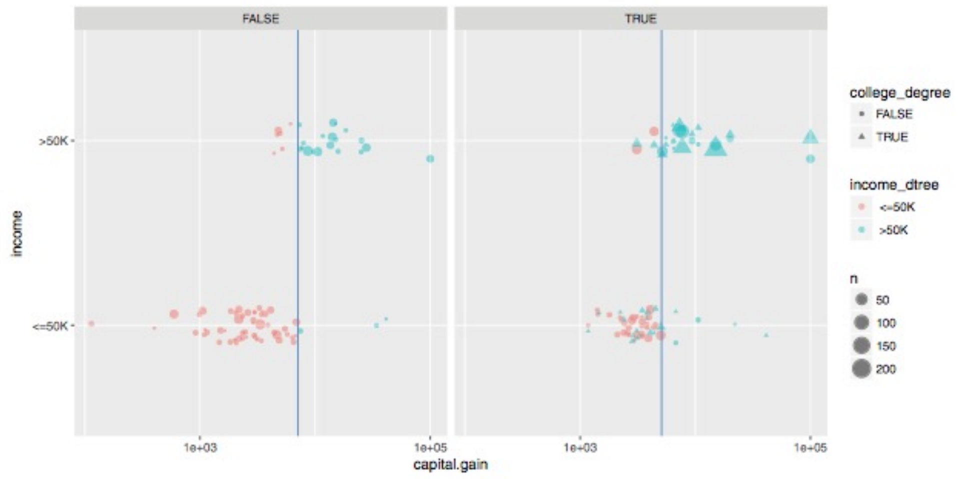

Figure 8.3: Graphical depiction of the full recursive partitioning decision tree classifier. On the left, those whose relationship status is neither "Husband" nor "Wife" are classified based on their capital gains paid. On the right, not only is the capital gains threshold different, but the decision is also predicated on whether the person has a college degree.

```r
train <- train %>%
  mutate(husband_or_wife = relationship %in% c(" Husband", " Wife"),
         college_degree = husband_or_wife & education %in%
           c(" Bachelors", " Doctorate", " Masters", " Prof-school"),
         income_dtree = predict(mod_tree, type = "class"))

cg_splits <- data.frame(husband_or_wife = c(TRUE, FALSE),
                        vals = c(5095.5, 7073.5))

ggplot(data = train, aes(x = capital.gain, y = income)) +
  geom_count(aes(color = income_dtree, shape = college_degree),
             position = position_jitter(width = 0, height = 0.1),
             alpha = 0.5) +
  facet_wrap(~ husband_or_wife) +
  geom_vline(data = cg_splits, aes(xintercept = vals),
             color = "dodgerblue", lty = 2) +
  scale_x_log10()
```

Since there are exponentially many trees, how did the algorithm know to pick this one? The *complexity parameter* controls whether to keep or prune possible splits. That is, the algorithm considers many possible splits (i.e., new branches on the tree), but prunes them if they do not sufficiently improve the predictive power of the model (i.e., bear fruit). By default, each split has to decrease the error by a factor of 1%. This will help to avoid *overfitting* (more on that later). Note that as we add more splits to our model, the relative error decreases.

```
printcp(mod_tree)


Classification tree:
rpart(formula = form, data = train)

Variables actually used in tree construction:
[1] capital.gain education    relationship

Root node error: 6317/26049 = 0.243

n= 26049

      CP nsplit rel error xerror    xstd
1 0.1289      0     1.000  1.000 0.01095
2 0.0641      2     0.742  0.742 0.00982
3 0.0367      3     0.678  0.678 0.00947
4 0.0100      4     0.641  0.641 0.00926

# plotcp(mod_tree)
```

An important tool in verifying a model's accuracy is called the *confusion matrix* (really). Simply put, this is a two-way table that counts how often our model made the correct prediction. Note that there are two different types of mistakes that our model can make: predicting a high income when the income was in fact low, and predicting a low income when the income was in fact high.

```
train <- train %>%
  mutate(income_dtree = predict(mod_tree, type = "class"))
confusion <- tally(income_dtree ~ income, data = train, format = "count")
confusion

            income
income_dtree  <=50K  >50K
       <=50K  18742  3061
       >50K     990  3256

sum(diag(confusion)) / nrow(train)

[1] 0.84449
```

In this case, the accuracy of the decision tree classifier is now 84.4%, a considerable improvement over the null model.

8.2.3 Tuning parameters

The decision tree that we built above was based on the default parameters. Most notably, our tree was pruned so that only splits that decreased the overall lack of fit by 1% were retained. If we lower this threshold to 0.2%, then we get a more complex tree.