and Linoff (2000) give the example of health insurance underwriting, where the insurer is required to show that coverage denial is not based on discrimination. By showing rules that led to denial (e.g., income < $20K AND low credit history), the company can avoid law suits. Compared to the output of other classifiers, such as discriminant functions, tree-based classification rules are easily explained to managers and operating staff. Their logic is certainly far more transparent than that of weights in neural networks!

9.6 Classification Trees for More Than Two Classes

Classification trees can be used with an outcome variable that has more than two classes. In terms of measuring impurity, the two measures presented earlier (the Gini impurity index and the entropy measure) were defined for *m* classes and hence can be used for any number of classes. The tree itself would have the same structure, except that its terminal nodes would take one of the *m*–class labels.

9.7 Regression Trees

The tree method can also be used for a numerical outcome variable. Regression trees for prediction operate in much the same fashion as classification trees. The outcome variable (*Y*) is a numerical variable in this case, but both the principle and the procedure are the same: Many splits are attempted, and for each, we measure "impurity" in each branch of the resulting tree. The tree procedure then selects the split that minimizes the sum of such measures. To illustrate a regression tree, consider the example of predicting prices of Toyota Corolla automobiles (from Chapter 6). The dataset includes information on 1000 sold Toyota Corolla cars (We use the first 1000 cars from the dataset *ToyotoCorolla.csv*. The goal is to find a predictive model of price as a function of 10 predictors (including mileage, horsepower, number of doors, etc.). A regression tree for these data was built using a training set of 600 records. The best-pruned tree is shown in Figure 9.14.

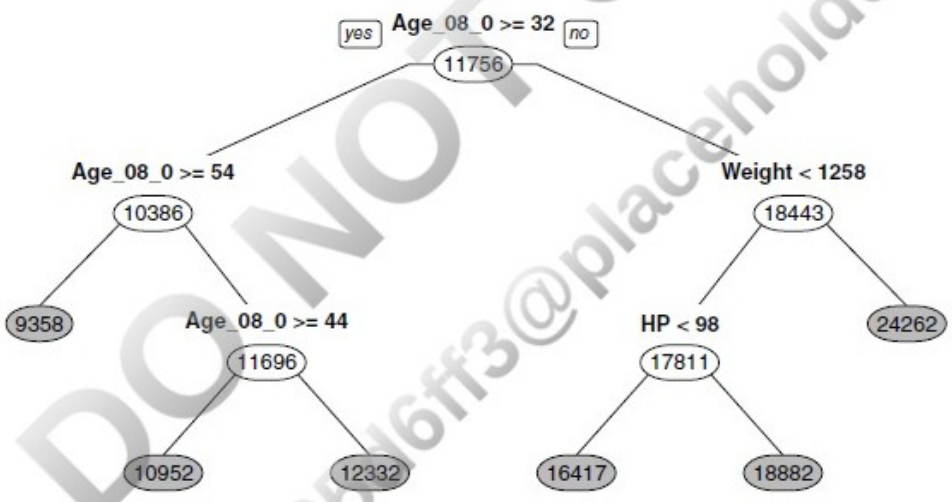


Figure 9.14 Best-pruned regression tree for Toyota Corolla prices

We see that from the 12 input variables (including dummies), only three predictors show up as useful for predicting price: the age of the car, its weight, and horsepower.

Three details differ between regression trees and classification trees: prediction, impurity measures, and evaluating performance. We describe these next.

Prediction

Predicting the outcome value for a record is performed in a fashion similar to the classification case: The predictor information is used for "dropping" the record down the tree until reaching a terminal node. For instance, to predict the price of a Toyota Corolla with Age = 60, Horse_Power = 100, and Weight = 1200, we drop it down the tree and reach the node that has the value $9358. This is the price prediction for this car according to the tree. In classification trees, the value of the terminal node (which is one of the categories) is determined by the "voting" of the training records that were in that terminal node. In regression trees, the value of the terminal node is determined by the

average outcome value of the training records that were in that terminal node. In the example above, the value $9358 is the average of the 279 cars in the training set that fall in the category of Age ≥ 54.

Measuring Impurity

We described two types of impurity measures for nodes in classification trees: the Gini index and the entropy-based measure. In both cases, the index is a function of the *ratio* between the categories of the records in that node. In regression trees, a typical impurity measure is the sum of the squared deviations from the mean of the terminal node. This is equivalent to the sum of the squared errors, because the mean of the terminal node is exactly the prediction. In the example above, the impurity of the node with the value $9358 is computed by subtracting 9358 from the price of each of the 279 cars in the training set that fell in that terminal node, then squaring these deviations and summing them up. The lowest impurity possible is zero, when all values in the node are equal.

Evaluating Performance

As stated above, predictions are obtained by averaging the outcome values in the nodes. We therefore have the usual definition of predictions and errors. The predictive performance of regression trees can be measured in the same way that other predictive methods are evaluated (e.g., linear regression), using summary measures such as RMSE.

9.8 Improving Prediction: Random Forests and Boosted Trees

Notwithstanding the transparency advantages of a single tree as described above, in a pure prediction application, where visualizing a set of rules does not matter, better performance is provided by several extensions to trees that combine results from multiple trees. These are examples of *ensembles* (see Chapter 13). One popular multitree approach is *random forests*, introduced by Breiman and Cutler.[1] Random forests are a special case of *bagging*, a method for improving predictive power by combining multiple classifiers or prediction algorithms. See Chapter 13 for further details on bagging.

Random Forests

The basic idea in random forests is to:

1. Draw multiple random samples, with replacement, from the data (this sampling approach is called the *bootstrap*).
2. Using a random subset of predictors at each stage, fit a classification (or regression) tree to each sample (and thus obtain a "forest").
3. Combine the predictions/classifications from the individual trees to obtain improved predictions. Use voting for classification and averaging for prediction.

The code and output in Figure 9.15 illustrates applying a random forest in R to the personal loan example. The accuracy of the random forest is slightly higher than the single default tree that we fit earlier (compare to the validation performance in Table 9.3).

 code for running a random forest, plotting variable importance plot, and computing accuracy

```
library(randomForest)
## random forest
rf <- randomForest(as.factor(Personal.Loan) ~ ., data = train.df, ntree = 500,
    mtry = 4, nodesize = 5, importance = TRUE)

## variable importance plot
varImpPlot(rf, type = 1)

## confusion matrix
rf.pred <- predict(rf, valid.df)
confusionMatrix(rf.pred, valid.df$Personal.Loan)

Partial Output
```

```
> confusionMatrix(rf.pred, valid.df$Personal.Loan)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1801   19
         1   10  170

               Accuracy : 0.986
```

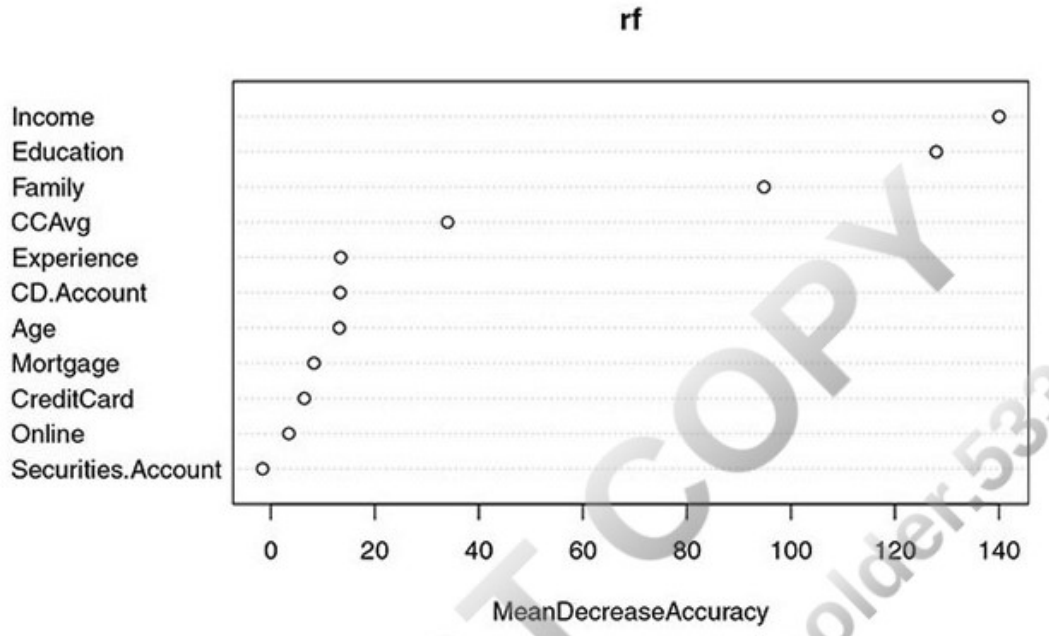


Figure 9.15 Variable importance plot from Random forest (Personal Loan Example)

Unlike a single tree, results from a random forest cannot be displayed in a tree-like diagram, thereby losing the interpretability that a single tree provides. However, random forests can produce "variable importance" scores, which measure the relative contribution of the different predictors. The importance score for a particular predictor is computed by summing up the decrease in the Gini index for that predictor over all the trees in the forest. Figure 9.15 shows the variable importance plots generated from the random forest model for the personal loan example. We see that Income and Education have the highest scores, with Family being third. Importance scores for the other predictors are considerably lower.

Boosted Trees

The second type of multitree improvement is *boosted trees*. Here a sequence of trees is fitted, so that each tree concentrates on misclassified records from the previous tree.

1. Fit a single tree.
2. Draw a sample that gives higher selection probabilities to misclassified records.
3. Fit a tree to the new sample.
4. Repeat Steps 2 and 3 multiple times.
5. Use weighted voting to classify records, with heavier weight for later trees.

Table 9.5 shows the result of running a boosted tree on the loan acceptance example that we saw earlier. We can see that compared to the performance of the single pruned tree (Table 9.3), the boosted tree has better performance on the validation data in terms of overall accuracy and especially in terms of correct classification of 1's—the rare class of special interest. Where does boosting's special talent for finding 1's come from? When one class is dominant (0's constitute over 90% of the data here), basic classifiers are tempted to classify cases as belonging to the dominant class, and the 1's in this case constitute most of the misclassifications with the single best-pruned tree. The boosting algorithm concentrates on the misclassifications (which are mostly 1's), so it is naturally going to do well in reducing the misclassification of 1's (from 18 in the single tree to 15 in the boosted tree, in the validation set).

the misclassification of 1's (from 18 in the single tree to 15 in the boosted tree, in the validation set).

Table 9.5 Boosted tree: confusion matrix for the validation set (loan data)

code for running boosted trees

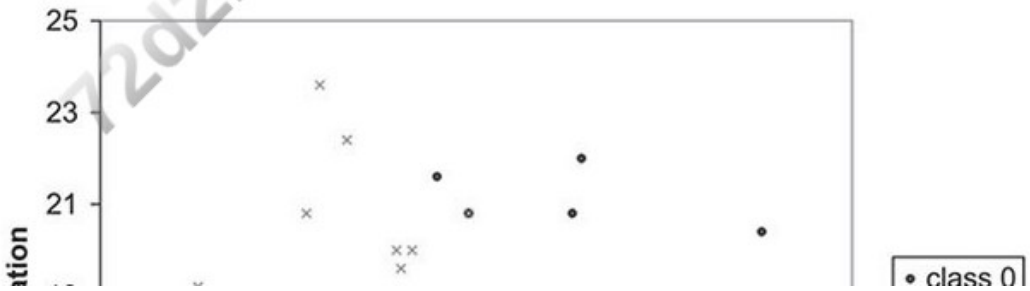
```
library(adabag)
library(rpart)
library(caret)

boost <- boosting(Personal.Loan ~ ., data = train.df)
pred <- predict(boost, valid.df)
confusionMatrix(pred$class, valid.df$Personal.Loan)

Output

Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1805   15
         1    6  174

               Accuracy : 0.9895
```

9.9 Advantages and Weaknesses of a Tree

Tree methods are good off-the-shelf classifiers and predictors. They are also useful for variable selection, with the most important predictors usually showing up at the top of the tree. Trees require relatively little effort from users in the following senses: First, there is no need for transformation of variables (any monotone transformation of the variables will give the same trees). Second, variable subset selection is automatic since it is part of the split selection. In the loan example, note that the best-pruned tree has automatically selected just three variables (Income, Education, and Family) out of the set of 14 variables available.

Trees are also intrinsically robust to outliers, since the choice of a split depends on the *ordering* of values and not on the absolute *magnitudes* of these values. However, they are sensitive to changes in the data, and even a slight change can cause very different splits!

Unlike models that assume a particular relationship between the outcome and predictors (e.g., a linear relationship such as in linear regression and linear discriminant analysis), classification and regression trees are nonlinear and nonparametric. This allows for a wide range of relationships between the predictors and the outcome variable. However, this can also be a weakness: Since the splits are done on one predictor at a time, rather than on combinations of predictors, the tree is likely to miss relationships between predictors, in particular linear structures like those in linear or logistic regression models. Classification trees are useful classifiers in cases where horizontal and vertical splitting of the predictor space adequately divides the classes. But consider, for instance, a dataset with two predictors and two classes, where separation between the two classes is most obviously achieved by using a diagonal line (as shown in Figure 9.16). In such cases, a classification tree is expected to have lower performance than methods such as discriminant analysis. One way to improve performance is to create new predictors that are derived from existing predictors, which can capture hypothesized relationships between predictors (similar to interactions in regression models). Random forests are another solution in such situations.



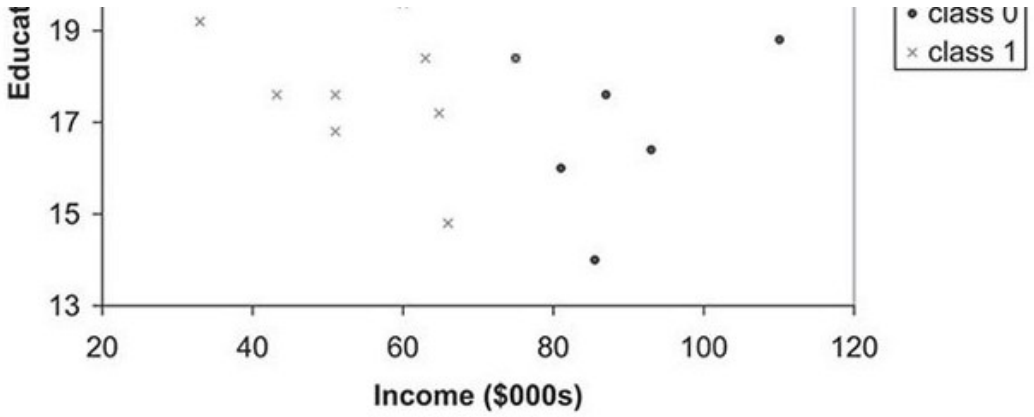


Figure 9.16 Scatter plot describing a two-predictor case with two classes. The best separation is achieved with a diagonal line, which classification trees cannot do

Another performance issue with classification trees is that they require a large dataset in order to construct a good classifier. From a computational aspect, trees can be relatively expensive to grow, because of the multiple sorting involved in computing all possible splits on every variable. Pruning the data using the validation sets adds further computation time.

Although trees are useful for variable selection, one challenge is that they "favor" predictors with many potential split points. This includes categorical predictors with many categories and numerical predictors with many different values. Such predictors have a higher chance of appearing in a tree. One simplistic solution is to combine multiple categories into a smaller set and bin numerical predictors with many values. Alternatively, some special algorithms avoid this problem by using a different splitting criterion [e.g., conditional inference trees in the R package party—see Hothorn et al. (2006)—and QUEST classification trees—see Loh and Shih (1997) and www.math.ccu.edu.tw/yshih/quest.html].

An appealing feature of trees is that they handle missing data without having to impute values or delete records with missing values. Finally, a very important practical advantage of trees is the transparent rules that they generate. Such transparency is often useful in managerial applications, though this advantage is lost in the ensemble versions of trees (random forests, boosted trees).

Problems

1. **Competitive Auctions on eBay.com.** The file *eBayAuctions.csv* contains information on 1972 auctions that transacted on eBay.com during May–June 2004. The goal is to use these data to build a model that will classify auctions as competitive or noncompetitive. A *competitive auction* is defined as an auction with at least two bids placed on the item auctioned. The data include variables that describe the item (auction category), the seller (his/her eBay rating), and the auction terms that the seller selected (auction duration, opening price, currency, day-of-week of auction close). In addition, we have the price at which the auction closed. The task is to predict whether or not the auction will be competitive.

   Data Preprocessing. Convert variable *Duration* into a categorical variable. Split the data into training (60%) and validation (40%) datasets.

   a. Fit a classification tree using all predictors, using the best-pruned tree. To avoid overfitting, set the minimum number of records in a terminal node to 50 (in R: *minbucket* = 50). Also, set the maximum number of levels to be displayed at seven (in R: *maxdepth = 7*). Write down the results in terms of rules. (*Note*: If you had to slightly reduce the number of predictors due to software limitations, or for clarity of presentation, which would be a good variable to choose?)

   b. Is this model practical for predicting the outcome of a new auction?

   c. Describe the interesting and uninteresting information that these rules provide.

   d. Fit another classification tree (using the best-pruned tree, with a minimum number of records per terminal node = 50 and maximum allowed number of displayed levels = 7), this time only with predictors that can be used for predicting the outcome of a new auction. Describe the resulting tree in terms of rules. Make sure to report the smallest set of rules required for classification.

e. Plot the resulting tree on a scatter plot: Use the two axes for the two best (quantitative) predictors. Each auction will appear as a point, with coordinates corresponding to its values on those two predictors. Use different colors or symbols to separate competitive and noncompetitive auctions. Draw lines (you can sketch these by hand or use R) at the values that create splits. Does this splitting seem reasonable with respect to the meaning of the two predictors? Does it seem to do a good job of separating the two classes?

f. Examine the lift chart and the confusion matrix for the tree. What can you say about the predictive performance of this model?

g. Based on this last tree, what can you conclude from these data about the chances of an auction obtaining at least two bids and its relationship to the auction settings set by the seller (duration, opening price, ending day, currency)? What would you recommend for a seller as the strategy that will most likely lead to a competitive auction?

2. **Predicting Delayed Flights.** The file *FlightDelays.csv* contains information on all commercial flights departing the Washington, DC area and arriving at New York during January 2004. For each flight, there is information on the departure and arrival airports, the distance of the route, the scheduled time and date of the flight, and so on. The variable that we are trying to predict is whether or not a flight is delayed. A delay is defined as an arrival that is at least 15 minutes later than scheduled.

   Data Preprocessing. Transform variable day of week (DAY_WEEK) info a categorical variable. Bin the scheduled departure time into eight bins (in R use function *cut()*). Use these and all other columns as predictors (excluding DAY_OF_MONTH). Partition the data into training and validation sets.

   a. Fit a classification tree to the flight delay variable using all the relevant predictors. Do not include DEP_TIME (actual departure time) in the model because it is unknown at the time of prediction (unless we are generating our predictions of delays after the plane takes off, which is unlikely). Use a pruned tree with maximum of 8 levels, setting *cp* = 0.001. Express the resulting tree as a set of rules.

   b. If you needed to fly between DCA and EWR on a Monday at 7:00 AM, would you be able to use this tree? What other information would you need? Is it available in practice? What information is redundant?

   c. Fit the same tree as in (a), this time excluding the Weather predictor. Display both the pruned and unpruned tree. You will find that the pruned tree contains a single terminal node.

      i. How is the pruned tree used for classification? (What is the rule for classifying?)

      ii. To what is this rule equivalent?

      iii. Examine the unpruned tree. What are the top three predictors according to this tree?

      iv. Why, technically, does the pruned tree result in a single node?

      v. What is the disadvantage of using the top levels of the unpruned tree as opposed to the pruned tree?

      vi. Compare this general result to that from logistic regression in the example in Chapter 10. What are possible reasons for the classification tree's failure to find a good predictive model?

3. **Predicting Prices of Used Cars (Regression Trees).** The file *ToyotaCorolla.csv* contains the data on used cars (Toyota Corolla) on sale during late summer of 2004 in the Netherlands. It has 1436 records containing details on 38 attributes, including Price, Age, Kilometers, HP, and other specifications. The goal is to predict the price of a used Toyota Corolla based on its specifications. (The example in Section 9.19 is a subset of this dataset).

   Data Preprocessing. Split the data into training (60%), and validation (40%) datasets.

   a. Run a regression tree (RT) with outcome variable Price and predictors Age_08_04, KM, Fuel_Type, HP, Automatic, Doors, Quarterly_Tax, Mfr_Guarantee, Guarantee_Period, Airco, Automatic_Airco, CD_Player, Powered_Windows, Sport_Model, and Tow_Bar. Keep the minimum number of records in a terminal node to 1, maximum number of tree levels to 30, and *cp* = 0.001, to make the run least restrictive.

      i. Which appear to be the three or four most important car specifications for predicting the car's price?

      ii. Compare the prediction errors of the training and validation sets by examining their RMS error and by plotting the two boxplots. How does the predictive performance of the validation set compare to the training set? Why does this occur?

      iii. How might we achieve better validation predictive performance at the expense of training performance?

      iv. Create a less deep tree by leaving the arguments cp, minbucket, and maxdepth at their defaults. Compared to the deeper tree, what is the predictive performance on the validation set?

to the deeper tree, what is the predictive performance on the validation set?

b. Let us see the effect of turning the price variable into a categorical variable. First, create a new variable that categorizes price into 20 bins. Now repartition the data keeping Binned_Price instead of Price. Run a classification tree with the same set of input variables as in the RT, and with Binned_Price as the output variable. As in the less deep regression tree, leave the arguments cp, minbucket, and maxdepth at their defaults.

   i. Compare the tree generated by the CT with the one generated by the less deep RT. Are they different? (Look at structure, the top predictors, size of tree, etc.) Why?

   ii. Predict the price, using the less deep RT and the CT, of a used Toyota Corolla with the specifications listed in Table 9.6.

   iii. Compare the predictions in terms of the predictors that were used, the magnitude of the difference between the two predictions, and the advantages and disadvantages of the two methods.

Table 9.6 Specifications For A Particular Toyota Corolla

Variable	Value
Age_-08_-04	77
KM	117,000
Fuel_Type	Petrol
HP	110
Automatic	No
Doors	5
Quarterly_Tax	100
Mfg_Guarantee	No
Guarantee_Period	3
Airco	Yes
Automatic_Airco	No
CD_Player	No
Powered_Windows	No
Sport_Model	No
Tow_Bar	Yes

Note

[1]For further details on random forests, see www.stat.berkeley.edu/users/breiman/RandomForests/cc_home.htm.