TABLE 1.1. *Average percentage of words or characters in an email message equal to the indicated word or character. We have chosen the words and characters showing the largest difference between* `spam` *and* `email`.

	george	you	your	hp	free	hpl	!	our	re	edu	remove
spam	0.00	2.26	1.38	0.02	0.52	0.01	0.51	0.51	0.13	0.01	0.28
email	1.27	1.27	0.44	0.90	0.07	0.43	0.11	0.18	0.42	0.29	0.01

measurements for a set of objects (such as people). Using this data we build a prediction model, or *learner*, which will enable us to predict the outcome for new unseen objects. A good learner is one that accurately predicts such an outcome.

The examples above describe what is called the *supervised learning* problem. It is called "supervised" because of the presence of the outcome variable to guide the learning process. In the *unsupervised learning problem*, we observe only the features and have no measurements of the outcome. Our task is rather to describe how the data are organized or clustered. We devote most of this book to supervised learning; the unsupervised problem is less developed in the literature, and is the focus of Chapter 14.

Here are some examples of real learning problems that are discussed in this book.

Example 1: Email Spam

The data for this example consists of information from 4601 email messages, in a study to try to predict whether the email was junk email, or "spam." The objective was to design an automatic spam detector that could filter out spam before clogging the users' mailboxes. For all 4601 email messages, the true outcome (email type) `email` or `spam` is available, along with the relative frequencies of 57 of the most commonly occurring words and punctuation marks in the email message. This is a supervised learning problem, with the outcome the class variable `email`/`spam`. It is also called a *classification* problem.

Table 1.1 lists the words and characters showing the largest average difference between `spam` and `email`.

Our learning method has to decide which features to use and how: for example, we might use a rule such as

$$\text{if } (\texttt{\%george} < 0.6) \;\&\; (\texttt{\%you} > 1.5) \quad \text{then } \texttt{spam} \text{ else } \texttt{email}.$$

Another form of a rule might be:

$$\text{if } (0.2 \cdot \texttt{\%you} \;-\; 0.3 \cdot \texttt{\%george}) > 0 \quad \text{then } \texttt{spam} \text{ else } \texttt{email}.$$

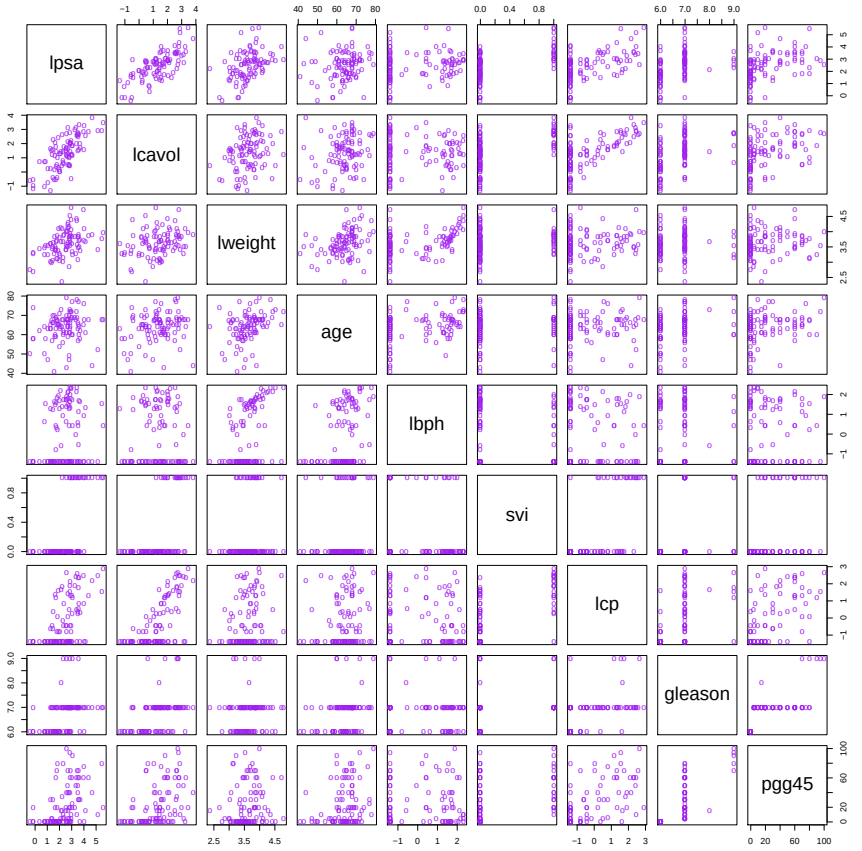

FIGURE 1.1. *Scatterplot matrix of the prostate cancer data. The first row shows the response against each of the predictors in turn. Two of the predictors,* `svi` *and* `gleason`*, are categorical.*

For this problem not all errors are equal; we want to avoid filtering out good email, while letting spam get through is not desirable but less serious in its consequences. We discuss a number of different methods for tackling this learning problem in the book.

Example 2: Prostate Cancer

The data for this example, displayed in Figure 1.1[1], come from a study by Stamey et al. (1989) that examined the correlation between the level of

[1]There was an error in these data in the first edition of this book. Subject 32 had a value of 6.1 for `lweight`, which translates to a 449 gm prostate! The correct value is 44.9 gm. We are grateful to Prof. Stephen W. Link for alerting us to this error.

Algorithm 9.2 *Local Scoring Algorithm for the Additive Logistic Regression Model.*

1. Compute starting values: $\hat{\alpha} = \log[\bar{y}/(1-\bar{y})]$, where $\bar{y} = \text{ave}(y_i)$, the sample proportion of ones, and set $\hat{f}_j \equiv 0\ \forall j$.

2. Define $\hat{\eta}_i = \hat{\alpha} + \sum_j \hat{f}_j(x_{ij})$ and $\hat{p}_i = 1/[1+\exp(-\hat{\eta}_i)]$.
   Iterate:

   (a) Construct the working target variable

   $$z_i = \hat{\eta}_i + \frac{(y_i - \hat{p}_i)}{\hat{p}_i(1-\hat{p}_i)}.$$

   (b) Construct weights $w_i = \hat{p}_i(1-\hat{p}_i)$

   (c) Fit an additive model to the targets z_i with weights w_i, using a weighted backfitting algorithm. This gives new estimates $\hat{\alpha}, \hat{f}_j, \forall j$

3. Continue step 2. until the change in the functions falls below a prespecified threshold.

The additive model fitting in step (2) of Algorithm 9.2 requires a weighted scatterplot smoother. Most smoothing procedures can accept observation weights (Exercise 5.12); see Chapter 3 of Hastie and Tibshirani (1990) for further details.

The additive logistic regression model can be generalized further to handle more than two classes, using the multilogit formulation as outlined in Section 4.4. While the formulation is a straightforward extension of (9.8), the algorithms for fitting such models are more complex. See Yee and Wild (1996) for details, and the `VGAM` software currently available from:

`http://www.stat.auckland.ac.nz/~yee.`

Example: Predicting Email Spam

We apply a generalized additive model to the `spam` data introduced in Chapter 1. The data consists of information from 4601 email messages, in a study to screen email for "spam" (i.e., junk email). The data is publicly available at `ftp.ics.uci.edu`, and was donated by George Forman from Hewlett-Packard laboratories, Palo Alto, California.

The response variable is binary, with values `email` or `spam`, and there are 57 predictors as described below:

- 48 quantitative predictors—the percentage of words in the email that match a given word. Examples include `business`, `address`, `internet`,

TABLE 9.1. *Test data confusion matrix for the additive logistic regression model fit to the spam training data. The overall test error rate is* 5.5%.

	Predicted Class	
True Class	`email` (0)	`spam` (1)
`email` (0)	58.3%	2.5%
`spam` (1)	3.0%	36.3%

`free`, and `george`. The idea was that these could be customized for individual users.

- 6 quantitative predictors—the percentage of characters in the email that match a given character. The characters are `ch;`, `ch(`, `ch[`, `ch!`, `ch$`, and `ch#`.
- The average length of uninterrupted sequences of capital letters: `CAPAVE`.
- The length of the longest uninterrupted sequence of capital letters: `CAPMAX`.
- The sum of the length of uninterrupted sequences of capital letters: `CAPTOT`.

We coded `spam` as 1 and `email` as zero. A test set of size 1536 was randomly chosen, leaving 3065 observations in the training set. A generalized additive model was fit, using a cubic smoothing spline with a nominal four degrees of freedom for each predictor. What this means is that for each predictor X_j, the smoothing-spline parameter λ_j was chosen so that $\text{trace}[\mathbf{S}_j(\lambda_j)]-1 = 4$, where $\mathbf{S}_j(\lambda)$ is the smoothing spline operator matrix constructed using the observed values x_{ij}, $i = 1, \ldots, N$. This is a convenient way of specifying the amount of smoothing in such a complex model.

Most of the `spam` predictors have a very long-tailed distribution. Before fitting the GAM model, we log-transformed each variable (actually $\log(x + 0.1)$), but the plots in Figure 9.1 are shown as a function of the original variables.

The test error rates are shown in Table 9.1; the overall error rate is 5.3%. By comparison, a linear logistic regression has a test error rate of 7.6%. Table 9.2 shows the predictors that are highly significant in the additive model.

For ease of interpretation, in Table 9.2 the contribution for each variable is decomposed into a linear component and the remaining nonlinear component. The top block of predictors are positively correlated with spam, while the bottom block is negatively correlated. The linear component is a weighted least squares linear fit of the fitted curve on the predictor, while the nonlinear part is the residual. The linear component of an estimated

TABLE 9.2. *Significant predictors from the additive model fit to the spam training data. The coefficients represent the linear part of $\hat{f}_j$, along with their standard errors and Z-score. The nonlinear P-value is for a test of nonlinearity of $\hat{f}_j$.*

Name	Num.	df	Coefficient	Std. Error	Z Score	Nonlinear P-value
Positive effects						
our	5	3.9	0.566	0.114	4.970	0.052
over	6	3.9	0.244	0.195	1.249	0.004
remove	7	4.0	0.949	0.183	5.201	0.093
internet	8	4.0	0.524	0.176	2.974	0.028
free	16	3.9	0.507	0.127	4.010	0.065
business	17	3.8	0.779	0.186	4.179	0.194
hpl	26	3.8	0.045	0.250	0.181	0.002
ch!	52	4.0	0.674	0.128	5.283	0.164
ch$	53	3.9	1.419	0.280	5.062	0.354
CAPMAX	56	3.8	0.247	0.228	1.080	0.000
CAPTOT	57	4.0	0.755	0.165	4.566	0.063
Negative effects						
hp	25	3.9	−1.404	0.224	−6.262	0.140
george	27	3.7	−5.003	0.744	−6.722	0.045
1999	37	3.8	−0.672	0.191	−3.512	0.011
re	45	3.9	−0.620	0.133	−4.649	0.597
edu	46	4.0	−1.183	0.209	−5.647	0.000

function is summarized by the coefficient, standard error and Z-score; the latter is the coefficient divided by its standard error, and is considered significant if it exceeds the appropriate quantile of a standard normal distribution. The column labeled *nonlinear P-value* is a test of nonlinearity of the estimated function. Note, however, that the effect of each predictor is fully adjusted for the entire effects of the other predictors, not just for their linear parts. The predictors shown in the table were judged significant by at least one of the tests (linear or nonlinear) at the $p = 0.01$ level (two-sided).

Figure 9.1 shows the estimated functions for the significant predictors appearing in Table 9.2. Many of the nonlinear effects appear to account for a strong discontinuity at zero. For example, the probability of `spam` drops significantly as the frequency of `george` increases from zero, but then does not change much after that. This suggests that one might replace each of the frequency predictors by an indicator variable for a zero count, and resort to a linear logistic model. This gave a test error rate of 7.4%; including the linear effects of the frequencies as well dropped the test error to 6.6%. It appears that the nonlinearities in the additive model have an additional predictive power.

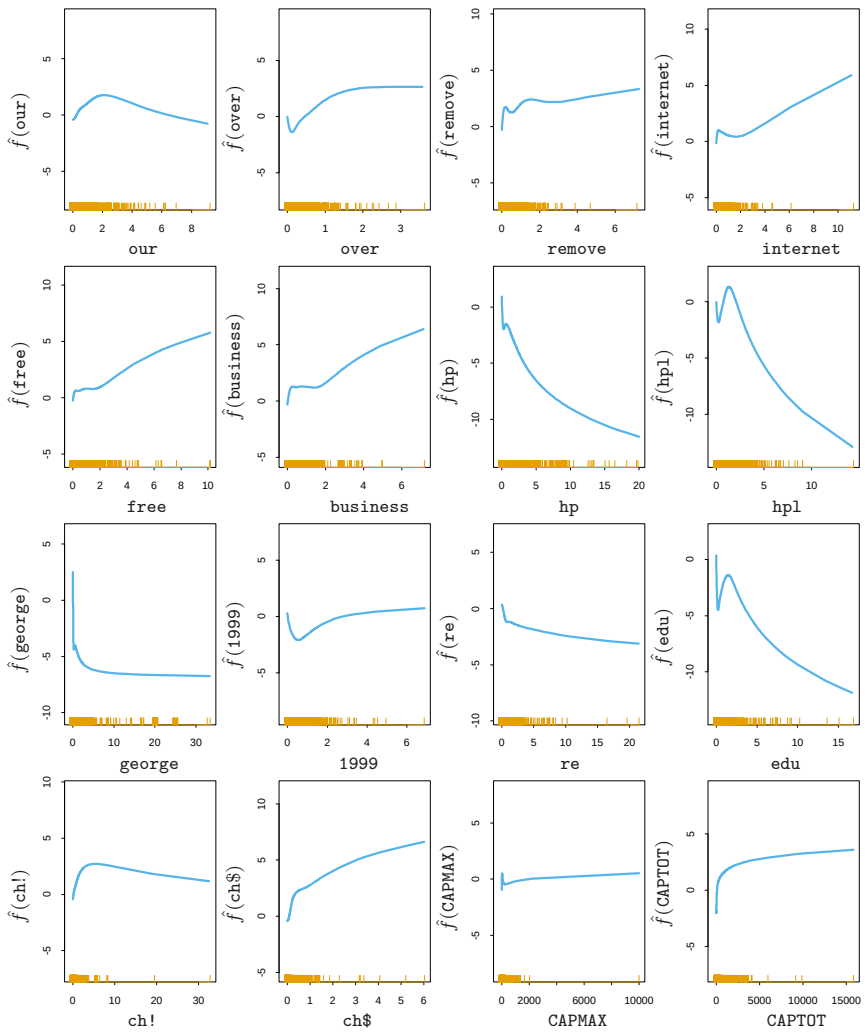

FIGURE 9.1. *Spam analysis: estimated functions for significant predictors. The* rug plot *along the bottom of each frame indicates the observed values of the corresponding predictor. For many of the predictors the nonlinearity picks up the discontinuity at zero.*

It is more serious to classify a genuine `email` message as `spam`, since then a good email would be filtered out and would not reach the user. We can alter the balance between the class error rates by changing the losses (see Section 2.4). If we assign a loss L_{01} for predicting a true class 0 as class 1, and L_{10} for predicting a true class 1 as class 0, then the estimated Bayes rule predicts class 1 if its probability is greater than $L_{01}/(L_{01} + L_{10})$. For example, if we take $L_{01} = 10, L_{10} = 1$ then the (true) class 0 and class 1 error rates change to 0.8% and 8.7%.

More ambitiously, we can encourage the model to fit better data in the class 0 by using weights L_{01} for the class 0 observations and L_{10} for the class 1 observations. As above, we then use the estimated Bayes rule to predict. This gave error rates of 1.2% and 8.0% in (true) class 0 and class 1, respectively. We discuss below the issue of unequal losses further, in the context of tree-based models.

After fitting an additive model, one should check whether the inclusion of some interactions can significantly improve the fit. This can be done "manually," by inserting products of some or all of the significant inputs, or automatically via the MARS procedure (Section 9.4).

This example uses the additive model in an automatic fashion. As a data analysis tool, additive models are often used in a more interactive fashion, adding and dropping terms to determine their effect. By calibrating the amount of smoothing in terms of df_j, one can move seamlessly between linear models ($\mathrm{df}_j = 1$) and partially linear models, where some terms are modeled more flexibly. See Hastie and Tibshirani (1990) for more details.

9.1.3 Summary

Additive models provide a useful extension of linear models, making them more flexible while still retaining much of their interpretability. The familiar tools for modeling and inference in linear models are also available for additive models, seen for example in Table 9.2. The backfitting procedure for fitting these models is simple and modular, allowing one to choose a fitting method appropriate for each input variable. As a result they have become widely used in the statistical community.

However additive models can have limitations for large data-mining applications. The backfitting algorithm fits all predictors, which is not feasible or desirable when a large number are available. The BRUTO procedure (Hastie and Tibshirani, 1990, Chapter 9) combines backfitting with selection of inputs, but is not designed for large data-mining problems. There has also been recent work using lasso-type penalties to estimate sparse additive models, for example the COSSO procedure of Lin and Zhang (2006) and the SpAM proposal of Ravikumar et al. (2008). For large problems a forward stagewise approach such as boosting (Chapter 10) is more effective, and also allows for interactions to be included in the model.

TABLE 9.3. *Spam data: confusion rates for the* 17*-node tree (chosen by cross–validation) on the test data. Overall error rate is* 9.3%*.*

	Predicted	
True	`email`	`spam`
`email`	57.3%	4.0%
`spam`	5.3%	33.4%

can be viewed as a modification of CART designed to alleviate this lack of smoothness.

Difficulty in Capturing Additive Structure

Another problem with trees is their difficulty in modeling additive structure. In regression, suppose, for example, that $Y = c_1 I(X_1 < t_1) + c_2 I(X_2 < t_2) + \varepsilon$ where ε is zero-mean noise. Then a binary tree might make its first split on X_1 near t_1. At the next level down it would have to split both nodes on X_2 at t_2 in order to capture the additive structure. This might happen with sufficient data, but the model is given no special encouragement to find such structure. If there were ten rather than two additive effects, it would take many fortuitous splits to recreate the structure, and the data analyst would be hard pressed to recognize it in the estimated tree. The "blame" here can again be attributed to the binary tree structure, which has both advantages and drawbacks. Again the MARS method (Section 9.4) gives up this tree structure in order to capture additive structure.

9.2.5 Spam Example (Continued)

We applied the classification tree methodology to the `spam` example introduced earlier. We used the deviance measure to grow the tree and misclassification rate to prune it. Figure 9.4 shows the 10-fold cross-validation error rate as a function of the size of the pruned tree, along with ± 2 standard errors of the mean, from the ten replications. The test error curve is shown in orange. Note that the cross-validation error rates are indexed by a sequence of values of α and *not* tree size; for trees grown in different folds, a value of α might imply different sizes. The sizes shown at the base of the plot refer to $|T_\alpha|$, the sizes of the pruned *original* tree.

The error flattens out at around 17 terminal nodes, giving the pruned tree in Figure 9.5. Of the 13 distinct features chosen by the tree, 11 overlap with the 16 significant features in the additive model (Table 9.2). The overall error rate shown in Table 9.3 is about 50% higher than for the additive model in Table 9.1.

Consider the rightmost branches of the tree. We branch to the right with a `spam` warning if more than 5.5% of the characters are the \$ sign.

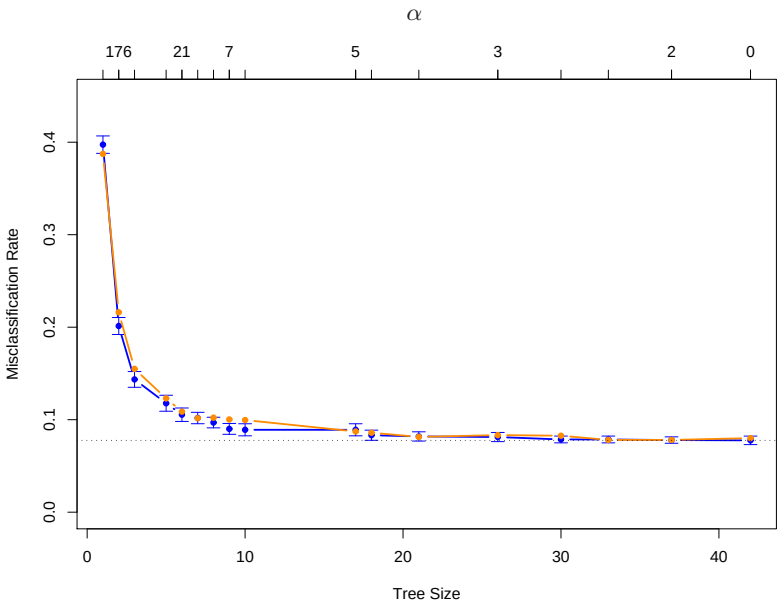

FIGURE 9.4. *Results for* `spam` *example. The blue curve is the* 10*-fold cross-validation estimate of misclassification rate as a function of tree size, with standard error bars. The minimum occurs at a tree size with about* 17 *terminal nodes (using the "one-standard-error" rule). The orange curve is the test error, which tracks the CV error quite closely. The cross-validation is indexed by values of* α*, shown above. The tree sizes shown below refer to* $|T_\alpha|$*, the size of the original tree indexed by* α*.*

However, if in addition the phrase `hp` occurs frequently, then this is likely to be company business and we classify as `email`. All of the 22 cases in the test set satisfying these criteria were correctly classified. If the second condition is not met, and in addition the average length of repeated capital letters `CAPAVE` is larger than 2.9, then we classify as `spam`. Of the 227 test cases, only seven were misclassified.

In medical classification problems, the terms *sensitivity* and *specificity* are used to characterize a rule. They are defined as follows:

Sensitivity: probability of predicting disease given true state is disease.

Specificity: probability of predicting non-disease given true state is non-disease.

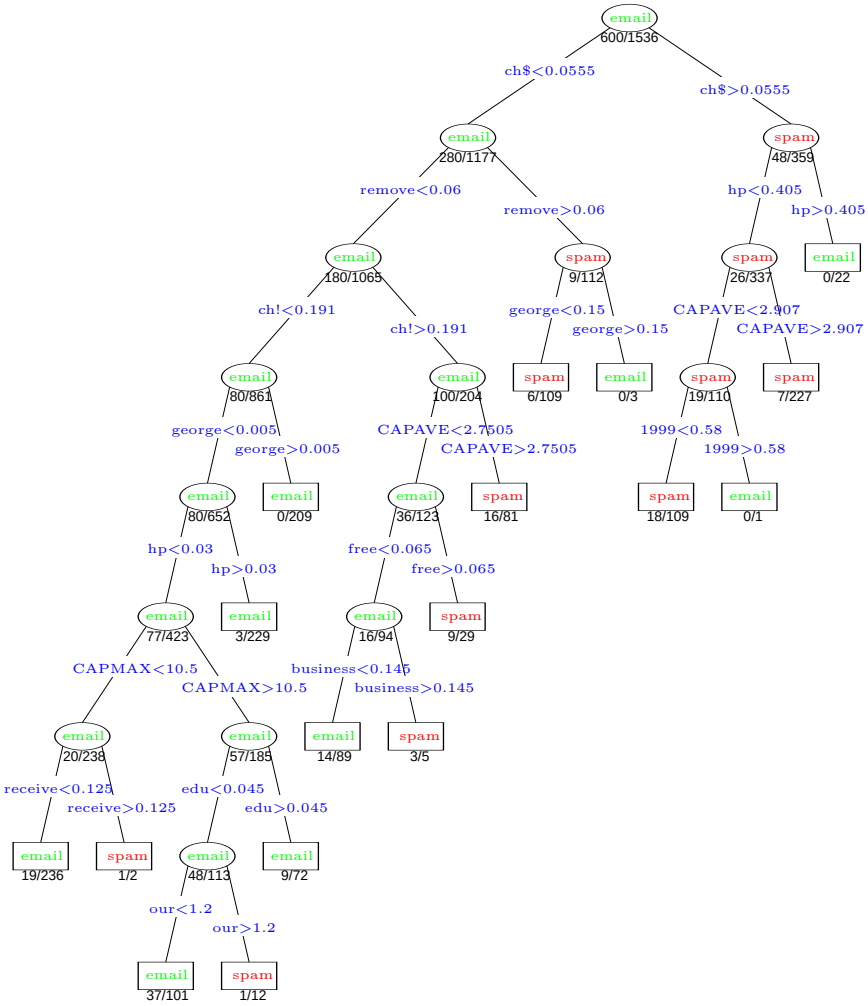

FIGURE 9.5. *The pruned tree for the* `spam` *example. The split variables are shown in blue on the branches, and the classification is shown in every node. The numbers under the terminal nodes indicate misclassification rates on the test data.*

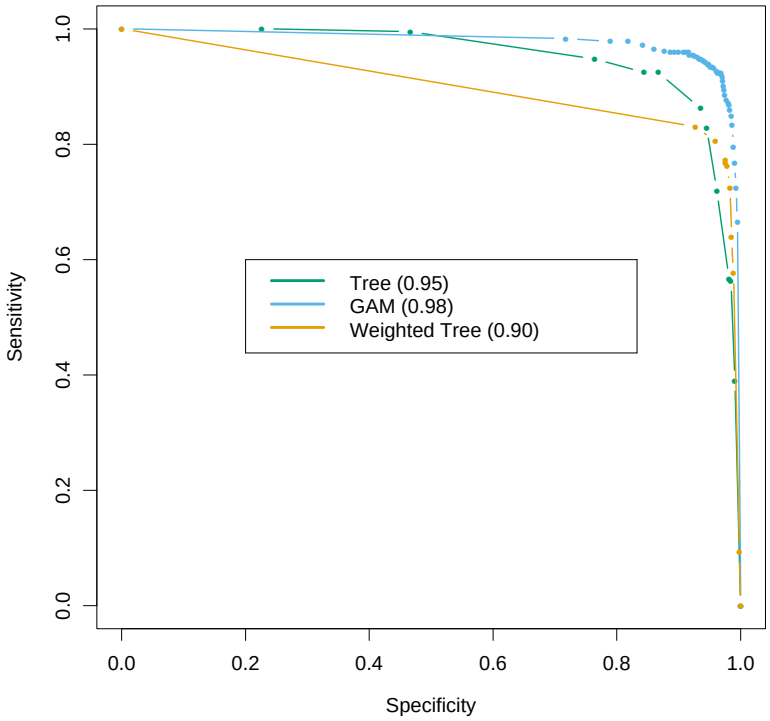

FIGURE 9.6. *ROC curves for the classification rules fit to the* `spam` *data. Curves that are closer to the northeast corner represent better classifiers. In this case the GAM classifier dominates the trees. The weighted tree achieves better sensitivity for higher specificity than the unweighted tree. The numbers in the legend represent the area under the curve.*

If we think of `spam` and `email` as the presence and absence of disease, respectively, then from Table 9.3 we have

$$\begin{aligned} \textit{Sensitivity} &= 100 \times \frac{33.4}{33.4 + 5.3} = 86.3\%, \\ \textit{Specificity} &= 100 \times \frac{57.3}{57.3 + 4.0} = 93.4\%. \end{aligned}$$

In this analysis we have used equal losses. As before let $L_{kk'}$ be the loss associated with predicting a class k object as class k'. By varying the relative sizes of the losses L_{01} and L_{10}, we increase the sensitivity and decrease the specificity of the rule, or vice versa. In this example, we want to avoid marking good `email` as `spam`, and thus we want the specificity to be very high. We can achieve this by setting $L_{01} > 1$ say, with $L_{10} = 1$. The Bayes' rule in each terminal node classifies to class 1 (`spam`) if the proportion of `spam` is $\geq L_{01}/(L_{10} + L_{01})$, and class zero otherwise. The

receiver operating characteristic curve (ROC) is a commonly used summary for assessing the tradeoff between sensitivity and specificity. It is a plot of the sensitivity versus specificity as we vary the parameters of a classification rule. Varying the loss L_{01} between 0.1 and 10, and applying Bayes' rule to the 17-node tree selected in Figure 9.4, produced the ROC curve shown in Figure 9.6. The standard error of each curve near 0.9 is approximately $\sqrt{0.9(1-0.9)/1536} = 0.008$, and hence the standard error of the difference is about 0.01. We see that in order to achieve a specificity of close to 100%, the sensitivity has to drop to about 50%. The area under the curve is a commonly used quantitative summary; extending the curve linearly in each direction so that it is defined over $[0, 100]$, the area is approximately 0.95. For comparison, we have included the ROC curve for the GAM model fit to these data in Section 9.2; it gives a better classification rule for any loss, with an area of 0.98.

Rather than just modifying the Bayes rule in the nodes, it is better to take full account of the unequal losses in growing the tree, as was done in Section 9.2. With just two classes 0 and 1, losses may be incorporated into the tree-growing process by using weight $L_{k,1-k}$ for an observation in class k. Here we chose $L_{01} = 5, L_{10} = 1$ and fit the same size tree as before ($|T_\alpha| = 17$). This tree has higher sensitivity at high values of the specificity than the original tree, but does more poorly at the other extreme. Its top few splits are the same as the original tree, and then it departs from it. For this application the tree grown using $L_{01} = 5$ is clearly better than the original tree.

The area under the ROC curve, used above, is sometimes called the *c-statistic*. Interestingly, it can be shown that the area under the ROC curve is equivalent to the Mann-Whitney U statistic (or Wilcoxon rank-sum test), for the median difference between the prediction scores in the two groups (Hanley and McNeil, 1982). For evaluating the contribution of an additional predictor when added to a standard model, the c-statistic may not be an informative measure. The new predictor can be very significant in terms of the change in model deviance, but show only a small increase in the c-statistic. For example, removal of the highly significant term `george` from the model of Table 9.2 results in a decrease in the c-statistic of less than 0.01. Instead, it is useful to examine how the additional predictor changes the classification on an individual sample basis. A good discussion of this point appears in Cook (2007).

9.3 PRIM: Bump Hunting

Tree-based methods (for regression) partition the feature space into box-shaped regions, to try to make the response averages in each box as differ-