Chapter 18
Measuring Predictor Importance

Often, we desire to quantify the strength of the relationship between the predictors and the outcome. As the number of attributes becomes large, exploratory analysis of the all the predictors may be infeasible and concentrating on those with strong relationships with the outcome may be an effective triaging strategy. Ranking predictors in this manner can be very useful when sifting through large amounts of data.

One of the primary reasons to measure the strength or relevance of the predictors is to filter which should be used as inputs in a model. This *supervised feature selection* can be data driven based on the existing data. The results of the filtering process, along with subject matter expertise, can be a critical step in creating an effective predictive model. As will be seen in the next chapter, many feature selection algorithms rely on a quantitative relevance score for filtering.

Many predictive models have built-in or *intrinsic* measurements of predictor importance and have been discussed in previous chapters. For example, MARS and many tree-based models monitor the increase in performance that occurs when adding each predictor to the model. Others, such as linear regression or logistic regression can use quantifications based on the model coefficients or statistical measures (such as t-statistics). The methodologies discussed in this chapter are not specific to any predictive model. If an effective model has been created, the scores derived from that model are likely to be more reliable than the methodologies discussed in this chapter because they are directly connected to the model.

In this chapter, the notion of variable importance is taken to mean an overall quantification of the relationship between the predictor and outcome. Most of the methodologies discussed cannot inform the modeler as to the *nature* of the relationship, such as "increasing the predictor results in a decrease in the outcome." Such detailed characterizations can only result from models having precise parametric forms such as linear or logistic regression, multivariate adaptive regression splines, and a few others. For example, Sect. 8.5 discussed variable importance scores for random forest. In

M. Kuhn and K. Johnson, *Applied Predictive Modeling*,
DOI 10.1007/978-1-4614-6849-3_18,

essence, this measurement of predictor relevance is derived by permuting each predictor individually and assessing the loss in performance when the effect of the predictor is negated. In this case, a substantial drop in performance is indicative of an important predictor. While this can be an effective approach, it does not enlighten the modeler as to the exact form of the relationship. Despite this, these measurements can be useful for guiding the user to focus more closely on specific predictors via visualizations and other means.

Many variable importance scores are specific to the type of data. For example, techniques for numeric outcomes may not be appropriate for categorical outcomes. This chapter is divided based on the nature of the outcome. Section 18.1 discusses techniques for numeric outcomes and Sect. 18.2 focuses on categorical outcomes. Finally, Sect. 18.3 discussed more general approaches to the problem. Several examples from previous chapters are used to illustrate the techniques.

18.1 Numeric Outcomes

For numeric predictors, the classic approach to quantifying each relationship with the outcome uses the sample correlation statistic. This quantity measures *linear* associations; if the relationship is nearly linear or curvilinear, then Spearman's correlation coefficient (Sect. 5.1) may be more effective. These metrics should be considered rough estimates of the relationship and may not be effective for more complex relationships.

For example, the QSAR data used in the previous regression chapters contained a number of numeric predictors. Figure 18.1 shows scatter plots of two predictors and the outcome. The relationship between solubility and number of carbon atoms is relatively linear and straightforward. The simple correlation was estimated to be -0.61 and the rank correlation was -0.67. For the surface area predictor, the situation is more complicated. There is a group of compounds with low surface area and, for these compounds, solubility is also low. The remainder of the compounds have higher solubility up to a point, after which there is a marginal trend. This could be a valuable piece of information to the modeler and would not be captured by the correlation statistic. For this predictor, simple correlation was estimated to be 0.27 and the rank correlation was 0.14. These values are relatively low and would rank accordingly low (17th out of 20 predictors).

An alternative is to use more flexible methods that may be capable of modeling general nonlinear relationships. One such technique is the locally weighted regression model (known more commonly as LOESS) of Cleveland and Devlin (1988). This technique is based on a series polynomial regressions that model the data in small neighborhoods (similar to computing a moving average). The approach can be effective at creating smooth regression trends that are extremely adaptive. The red lines in Fig. 18.1 are the LOESS

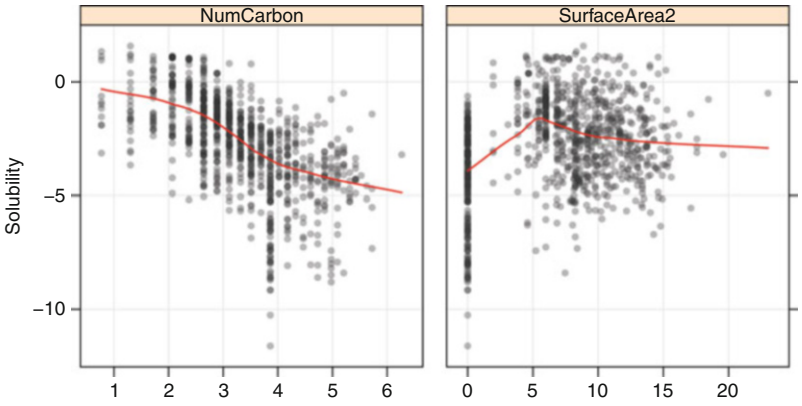

Fig. 18.1: Scatter plots for two numeric predictors (on transformed scales) for the solubility data. The *red lines* are smoother model fits

smoother fits. From the model fit, a pseudo-R^2 statistic can be calculated derived from the residuals. For the solubility data shown in Fig. 18.1, the LOESS pseudo-R^2 for the surface area was 0.22, which ranks higher (7th out of 20) than the correlation scores.

Figure 18.2 shows the relationship between importance scores for all of the continuous predictors. The third metric, the maximal information coefficient (MIC), is discussed in Sect. 18.3. The rank correlation and LOESS pseudo-R^2 tend to give the same results, with the exception of two surface area predictors. Both methods reflect three distinct clusters of predictors, based on their ranks: a set of four high scoring predictors (two significantly overlap in the plot), a set of moderately important predictors, and another set of very low importance. The only inconsistency between the methods is that the two surface area predictors score higher using LOESS than the other methods. These predictors are shown as red squares in the image.

Each of these techniques evaluates each predictor without considering the others. This can be potentially misleading in two ways. First, if two predictors are highly correlated with the response and with each other, then the univariate approach will identify both as important. As we have seen, some models will be negatively impacted by including this redundant information. Pre-processing approaches such as removing highly correlated predictors can alleviate this problem. Second, the univariate importance approach will fail to identify groups of predictors that together have a strong relationship with the response. For example, two predictors may not be highly correlated with the response; however, their interaction may be. Univariate correlations will not capture this predictive relationship. A sensible strategy would be to further explore these aspects of the predictors instead of using the rankings as the sole method for understanding the underlying trends. Knowing which relationship to explore often requires expert knowledge about the data.

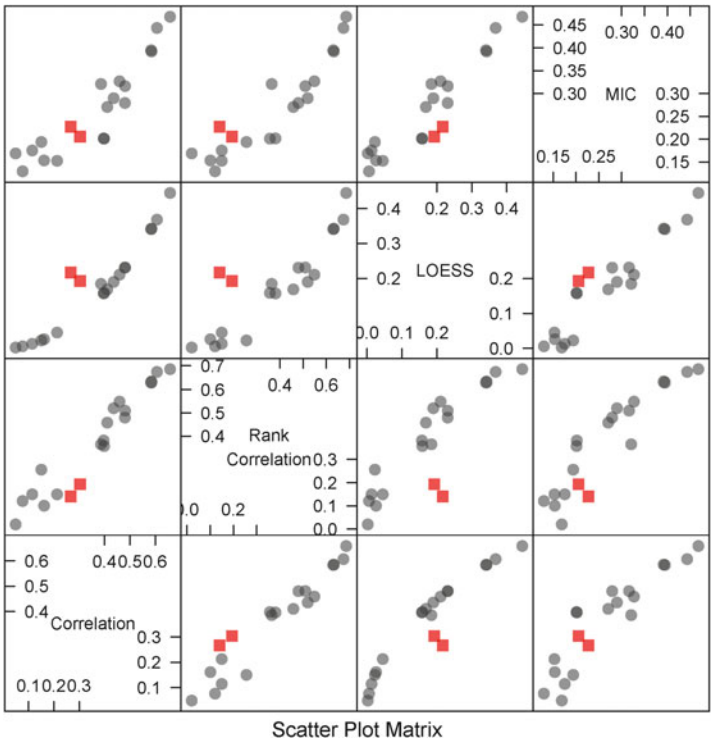

Fig. 18.2: A scatter plot matrix importance metrics for the 20 continuous predictors in the solubility data. The two surface area predictors are shown as *red squares*

When the predictors are categorical, different methodologies are required. In the solubility data, there were 208 fingerprint predictors, which are indicator variables representing particular atomic structures in the compound. The most straightforward method for determining the relevance of each binary predictor is to evaluate whether the average outcome in each category is different. Consider fingerprint `FP175`; the difference between the mean solubility value with and without the structure is −0.002. Given that solubility ranges from −11.6 to 1.6, this is likely to be negligible. On the other hand, fingerprint `FP044` has a difference of −4.1 log units. This is larger but on its own may not be informative. The variability in the data should also be considered.

The most natural method for comparing the mean of two groups is the standard t-statistic, which is essentially a signal-to-noise ratio (the difference in means is divided by a function of the variabilities in the groups). A p-value can be produced by this procedure where the null hypothesis is that there is no difference between the groups. The assumption for the statistic

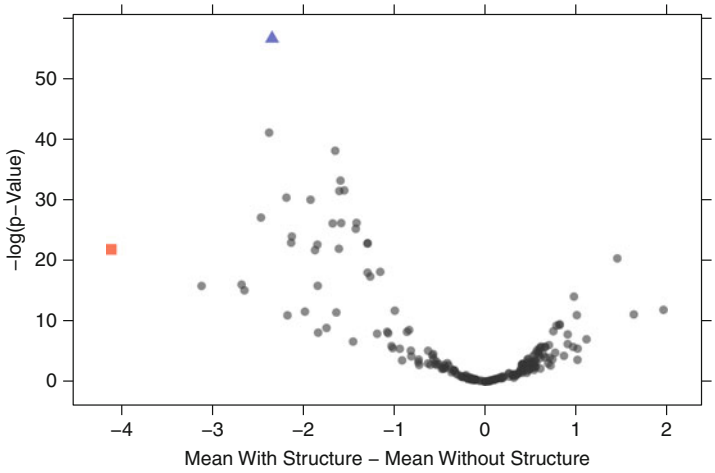

Fig. 18.3: A volcano plot of the t-test results applied to the fingerprint predictors. The *red square* is fingerprint `FP044` (which has the largest difference in means) while the *blue triangle* is fingerprint `FP076` and is the most statistically significant

is that the data are normally distributed. If this assumption is unlikely to be true, other methods (such as the Wilcoxon rank sum test) may be more appropriate.

For each of the 208 fingerprint predictors, a t-test was used. Instead of solely relying on the p-value, Fig. 18.3 shows a *volcano plot* where a transformed version of the p-value is shown on the y-axis and the difference in solubility means is shown on the x-axis. Higher values of the y-axis are indicative of strong statistical significance (i.e., low p-value). Most of the mean differences are negative, indicating that solubility tends to be higher without the structure. `FP044` has the largest difference (and is shown as a red square). However, this difference is not the most statistically significant. Fingerprint `FP076` is the most statistically significant (shown as a blue triangle) and has a difference in means of -2.3 log units. As previously mentioned, the t-statistic is a signal-to-noise ratio. While fingerprint `FP044` has the largest signal, it also has appreciable variability which diminishes its significance. This is reflected in the 95 % confidence intervals for the mean differences: (3.6, 4.7) for `FP044` and (2.1, 2.6) for fingerprint `FP076`. Which predictor is more important depends on the context and the particular data set. If the signal is large enough, the model may be able to overcome the noise. In other cases, smaller but more precise differences may generate better performance.

When the predictor has more than two values, an analysis of variance (ANOVA) model can be used to characterize the statistical significance of the predictors. However, if the means of the categories are found to be different, the natural next step is to discover which are different. For this reason, it may

be helpful to decompose the categories into multiple binary dummy variables and apply the procedure outlined above to determine the relevance of each category.

18.2 Categorical Outcomes

With categorical outcomes and numeric predictors, there are several approaches to quantifying the importance of the predictor. The image segmentation data from Chap. 3 will be used to illustrate the techniques. Figure 18.4 shows histograms of two predictors (the fiber width for channel 1 and the spot fiber count for channel 4) for each class. Fiber width shows a shift in the average values between the classes while the spot fiber count distribution appears very similar between classes.

One approach when there are two classes is to use the area under the ROC curve to quantify predictor relevance. Sections 11.3 and 16.4 utilized the ROC curve with predicted class probabilities as inputs. Here, we use the predictor data as inputs into the ROC curve. If the predictor could perfectly separate the classes, there would be a cutoff for the predictor that would achieve a sensitivity and specificity of 1 and the area under the curve would be one. As before, a completely irrelevant predictor would have an area under the curve of approximately 0.5. Figure 18.5 shows the area under the curve for the fiber width and spot fiber count predictors. The area under the ROC

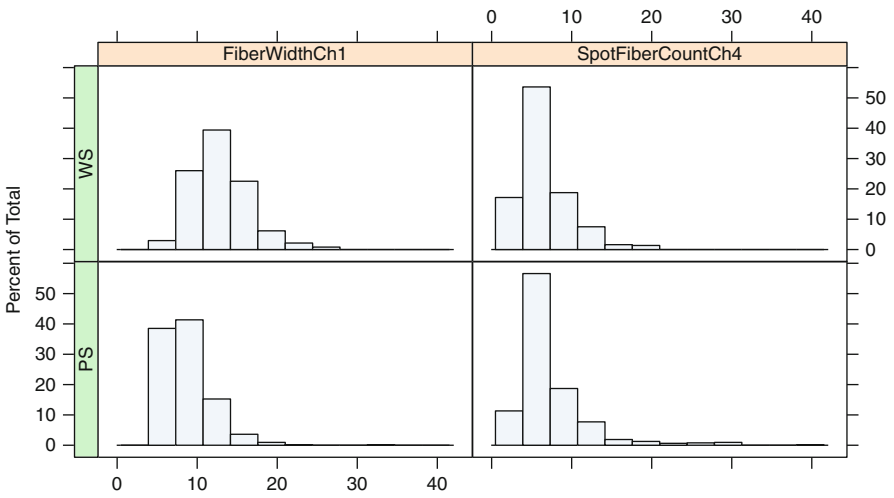

Fig. 18.4: Histograms for two predictors in the cell segmentation data, separated for each class

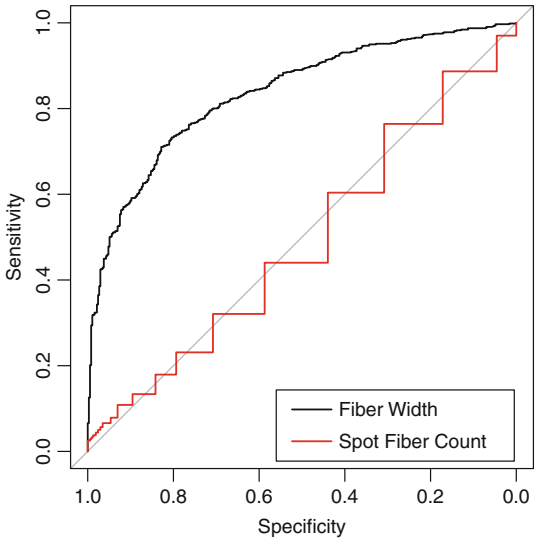

Fig. 18.5: ROC curves for two predictors in the cell segmentation data

curve for the fiber width data was 0.836 while the value for the spot fiber count was 0.538. This echoes the trends seen in Fig. 18.4 where the fiber width showed more separation between classes.

When there are multiple classes, the extensions of ROC curves described by Hanley and McNeil (1982), DeLong et al. (1988), Venkatraman (2000), and Pepe et al. (2009) can also be used. Alternatively, a set of "one versus all" ROC curves can be created by lumping all but one of the classes together. In this way, there would be a separate AUC for each class and the overall relevance can be quantified using the average or maximum AUC across the classes.

Another, more simplistic approach is to test if the mean values of predictors within each class are different. This is similar to the technique described in the previous section, but, in this context, the predictor is being treated as the outcome. Volcano plots cannot be used since the each of the differences may be on different scales. However, the t-statistics can be used to compare the predictors on different scales. For the fiber width, the t-statistic value was -19, indicating a clear signal above the noise. The signal was much weaker for the spot fiber count, with a value of 3.

For the segmentation data, the 58 continuous predictors were evaluated using each of these methods along with the random forest variable importance score and two methods that are described in Sect. 18.3 (MIC and Relief). The methods are roughly concordant (Fig. 18.6). The area under the ROC curve shows a tight, curvilinear relationship with the random forest score and a strong linear relationship with the t-statistic. The t-statistic and random

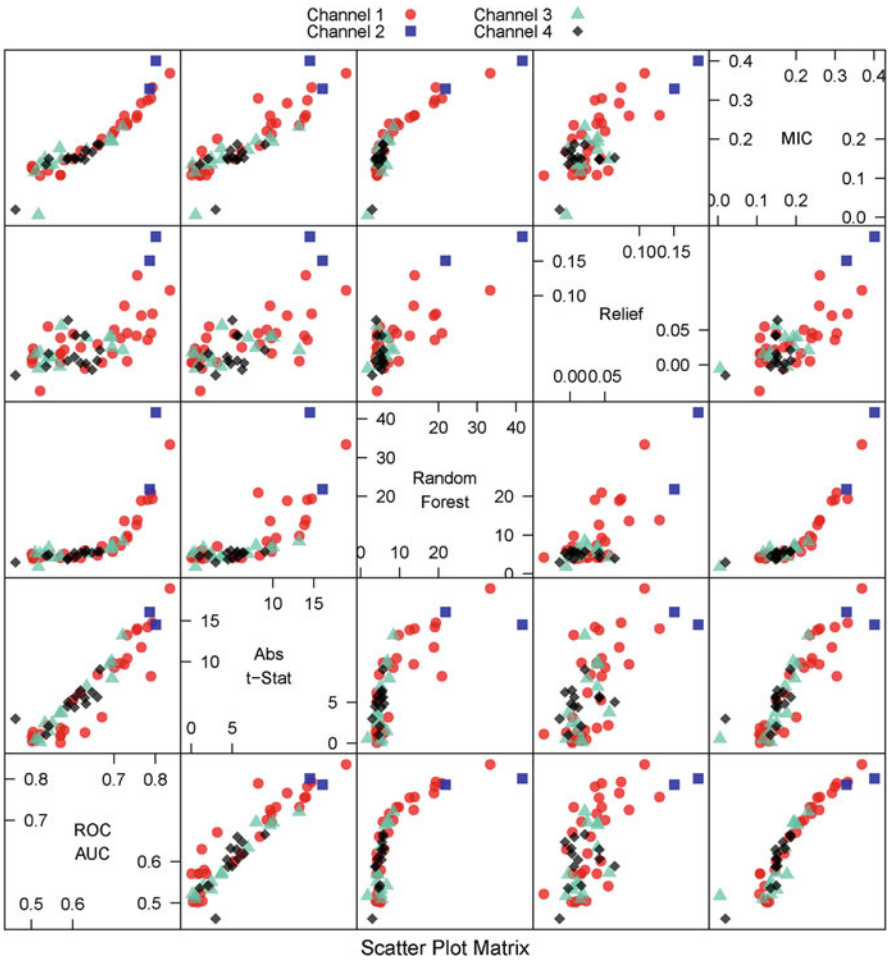

Fig. 18.6: A scatter plot matrix of several different metrics for ranking predictors with the image segmentation data set

forest appear to rank the same predictors as less important, but there are differences in the scores for those quantified as most important. Three predictors tend to be ranked very highly across the methods: average intensity (channel 2), fiber width (channel 1), and total intensity (channel 2).

When the predictor is categorical, there are several metrics that may be appropriate. For binary predictors and two classes, one effective method for measuring the relevance is the odds ratio. Recall that the odds of a probability are $p/(1-p)$. The probability of an event can be calculated for both levels of the predictor. If these are denoted as p_1 and p_2, the odds ratio (Bland and

Table 18.1: Statistics for three binary categorical predictors in the grant data

	Grant success					
	Yes	No	%	OR	p-value	Gain ratio
Sponsor						
62B	7	44	14			
Other	3226	3356	49	6.0	$2.6e^{-07}$	0.04726
CVB						
Band unknown	644	2075	24			
Band known	2589	1325	66	6.3	$1.7e^{-263}$	0.13408
RFCD code						
240302	13	15	46			
Other code	3220	3385	49	1.1	$8.5e^{-01}$	0.00017

Altman 2000; Agresti 2002) is

$$\text{OR} = \frac{p_1(1-p_2)}{p_2(1-p_1)}$$

and represents the increase in the odds of the event when going from the first level of the predictor to the other.

To illustrate this approach, the grant application data from previous chapters is used. For these data there are 226 binary predictors with at least 25 entries in each cell of the 2×2 table between the predictor and outcome. Table 18.1 shows cross tabulations for three predictors in the grant data. For contract value band, the probability of grant success is determined when the band is known ($p_1 = 0.661$) and unknown ($p_2 = 0.237$). The corresponding odds ratio is 6.3, meaning that there is more than a sixfold increase in the odds of success when the band is known. As with the discussion on the differences in means, the odds ratio only reflects the signal, not the noise. Although there are formulas for calculating the confidence interval for the odds ratio, a more common approach is to conduct a statistical hypothesis test that the odds ratio is equal to one (i.e., equal odds between the predictor levels). When there are two categories and two values for the predictor, Fisher's exact test (Agresti 2002) can be used to evaluate this hypothesis and can be summarized by the resulting p-value. For this predictor, the p-value was effectively zero, which indicates that levels of contract value band are related to grant success. Table 18.1 shows another predictor for Sponsor 62B that has a similar odds ratio (OR = 6), but the number of grants with this sponsor is very low. Here, the p-value was still small but would be ranked much lower using statistical significance. The left-hand panel of Fig. 18.7 shows a volcano

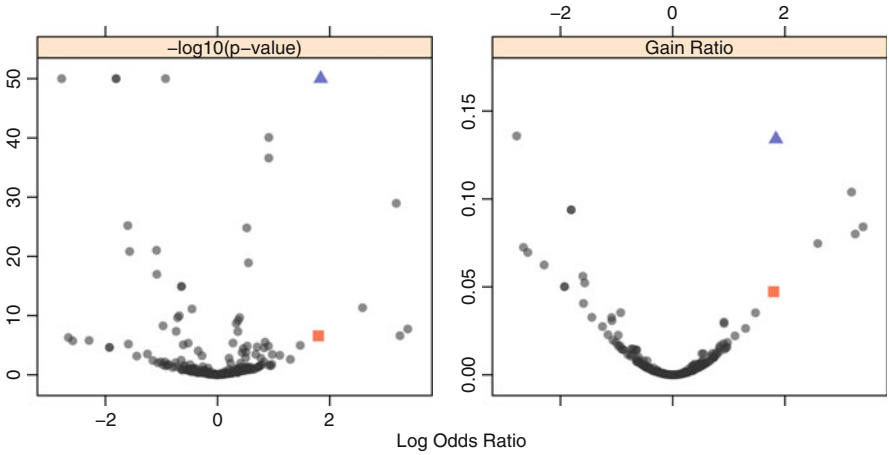

Fig. 18.7: The log odds ratio plotted against two other statistics. *Left*: a function of the p-values from Fisher's exact test is shown. The largest value in the data is 263; this image truncates at 50 to preserve the perspective. *Right*: The gain ratio for each predictor. The *red square* is the predictor for sponsor 62B and the *blue triangle* is for unknown contract value band

plot for the odds ratio and the corresponding p-value for 226 binary grant predictors.

When there are more than two classes or the predictors have more than two levels, other methods can be applied. Fisher's exact test can still be used to measure the association between the predictor and the classes. Alternatively, the gain ratio used for C4.5 can be applied to quantify relationship between two variables, where larger is better. Recall that the gain ratio adjusts for the number of levels in the predictor to remove the bias against attributes with few levels. For this reason, the information gain can be applied to all categorical predictors, irrespective of their characteristics. Table 18.1 shows the information gain for three of the grant predictors and Fig. 18.7 contrasts the gain statistics versus the odds ratio. Like the p-value for Fisher's exact test, the gain statistics favor the contract value band predictor over the one for Sponsor 62B, despite the similarities in their odds ratio.

18.3 Other Approaches

The Relief algorithm (Kira and Rendell 1992) is a generic method for quantifying predictor importance. It was originally developed for classification problems with two classes but has been extended to work across a

```
Initialize the predictor scores S_j to zero
for i = 1...m randomly selected training set samples (R_i) do
    Find the nearest miss and hit in the training set
    for j = 1...p predictor variables do
        Adjust the score for each predictor based on the proximity of
        R_j to the nearest miss and hit:
        S_j = S_j − diff_j(R_j, Hit)^2/m + diff_j(R_j, Miss)^2/m
    end
end
```

Algorithm 18.1: The original Relief algorithm for ranking predictor variables in classification models with two classes

wider range of problems. It can accommodate continuous predictors as well as dummy variables and can recognize nonlinear relationships between the predictors and the outcome. It uses random selected points and their nearest neighbors to evaluate each predictor in isolation.

For a particular predictor, the score attempts to characterize the separation between the classes in isolated sections of the data. Algorithm 18.1 shows the procedure.

For a randomly selected training set sample, the algorithm finds the nearest samples from both classes (called the "hits" and "misses"). For each predictor, a measure of difference in the predictor's values is calculated between the random data point and the hits and misses. For continuous predictors, Kira and Rendell (1992) suggested the distance between the two points be divided by the overall range of the predictor:

$$\text{diff}(x, y) = (x - y)/C$$

where C is a constant for the predictor that scales the difference to be between 0 and 1. For binary (i.e., 0/1) data, a simple indicator for equivalence can be used:

$$\text{diff}(x, y) = |x - y|$$

so that these values are also between 0 and 1.

The overall score (S_j) is an accumulation of these differences such that the score is decreased if the hit is far away from the randomly selected value but is increased if the miss is far away. The idea is that a predictor that shows a separation between the classes should have hits nearby and missed far away. Given this, larger scores are indicative of important predictors.

For example, Fig. 18.8 presents a set of data for several predictors. The top panel shows a predictor that completely separates the classes. Suppose the sampling procedure selects the second to last sample on the left-hand side

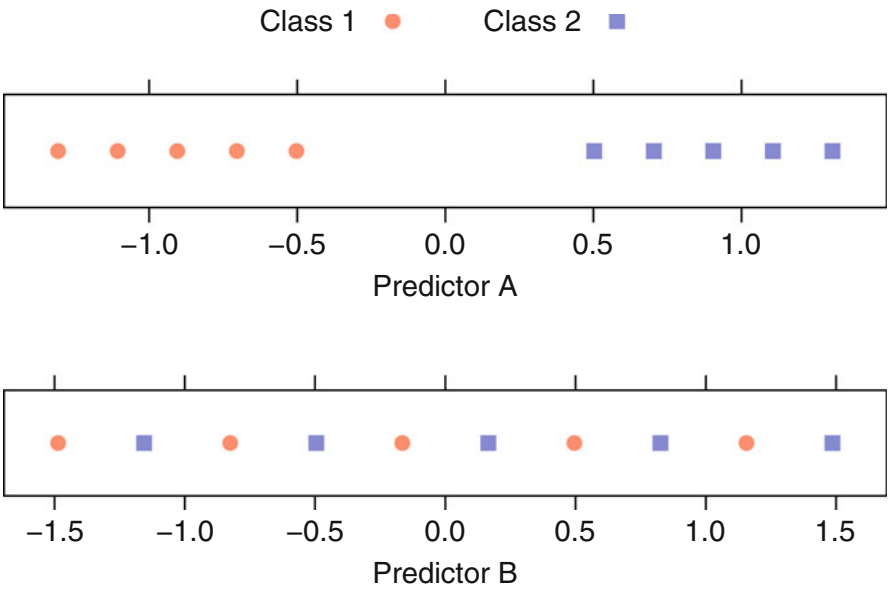

Fig. 18.8: Three data sets that illustrate the Relief algorithm. In the *top panel*, predictor A shows complete separation with a ReliefF score of 0.19. The *middle panel* shows a completely non-informative predictor with a corresponding ReliefF score of 0

(predictor $A = -1.1$). The nearest hit is on either side of this data point, with a difference of approximately 0.2 (we'll forgo the constant here). The nearest miss is far away ($A = 0.5$) with a corresponding difference of -1.61. If this were the first randomly selected training set point, the score would be

$$S_A = 0 - 0.2^2 + -1.61^2 = 2.55$$

This value would be divided by m and new values would be accumulated. As a counter example, predictor B in Fig. 18.8 is completely non-informative. Each hit and miss is adjacent to any randomly selected sample, so the differences will always offset each other and the score for this predictor is $S_B = 0$.

This procedure was subsequently improved by Kononenko (1994). The modified algorithm, called ReliefF, uses more than a single nearest neighbor, uses a modified difference metric, and allows for more than two classes as well as missing predictor values. Additionally, Robnik-Sikonja and Kononenko (1997) adapted the algorithm for regression (i.e., numeric outcomes).

The bottom panel of Fig. 18.9 illustrates an hypothetical data set with two classes and two correlated predictors (denoted as C and D). The class boundary was created using a simple logistic regression model equation with two strong main effects and a moderate interaction effect:

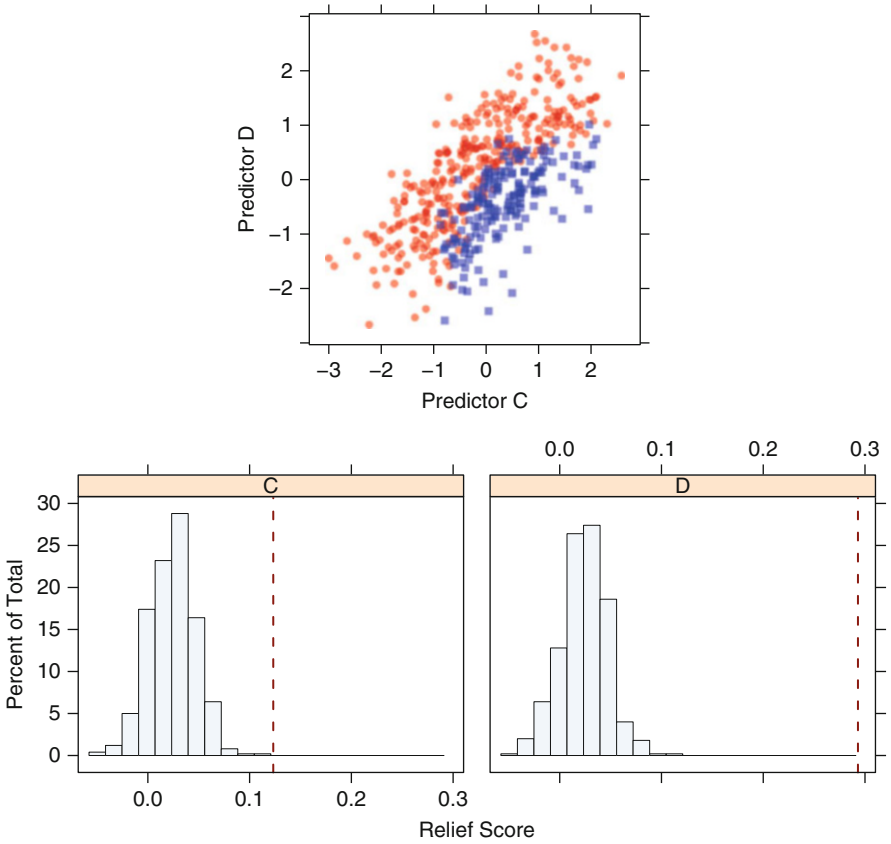

Fig. 18.9: *Top*: two predictors with distinct class boundary. In this example, the ReliefF scores for predictors C and D are 0.17 and 0.25, respectively. *Bottom*: the permutation distribution of the scores when there is no relationship between the predictors and the classes. The *dashed lines* indicate the observed ReliefF scores

$$\log\left(\frac{\pi}{1-\pi}\right) = -4C - 4D + 2CD$$

The data shown in Fig. 18.9 were simulated with this class boundary. For these data, a logistic regression model was able to achieve a cross-validated accuracy rate of 90.6 %. Knowing the true model equation, both predictors should be quantified as being important. Since there is an interaction effect in the model equation, one would expect the ReliefF algorithm to show a larger signal than other methods that only consider a single predictor at a time. For these data, the ReliefF algorithm was used with $m = 50$ random samples and $k = 10$-nearest neighbors. The two predictors were centered and scaled

prior to running the algorithm. The scores for predictors C and D were 0.17 and 0.25, respectively. The areas under the ROC curves for predictors C and D were moderate, with values of 0.65 and 0.69, respectively.

What cutoff for these scores should be used? Kira and Rendell (1992) suggest that a threshold "can be much smaller than $1/\sqrt{\alpha m}$," where α is the desired false-positive rate (i.e., the percent of time that an irrelevant predictor is judged to be important). Using this yardstick with a 5 % false positive rate, values greater than 0.63 may be considered important. Another alternative is to use a well-established statistical method called a *permutation test* (Good 2000) to evaluate the scores. Here, the true class labels are randomly shuffled and the ReliefF scores are recalculated many times. This should provide some sense of the distribution of the scores when the predictor has no relevance. From this, the observed score can be contrasted against this distribution to characterize how extreme it is relative to the assumption that the predictor is not relevant. Using 500 random permutations of the data, the distributions of the scores for predictors C and D resembled normally distributed data (Fig. 18.9). For predictor C, the score values ranged from -0.05 to 0.11, while for predictor D, the range was -0.05 to 0.11. In this context, the two observed values for the data in Fig. 18.9 are indicative of very important predictors. Given the symmetric, normally distributed shapes of the random permutations, it can be determined that the scores for predictors C and D are 4.5 and 11.6 standard deviations away from what should be expected by chance. Given how extreme the scores are, the conclusion would be that both predictors are important factors for separating the classes.

For the cell segmentation data, Fig. 18.6 shows the ReliefF scores for each predictor. These values are moderately correlated with the other metrics but show larger variability in the relationships. For example, the area under the ROC curve shows less noise in the relationships with the other metrics. This may be due to the metrics measuring different aspects of the data or may be due to the random sampling used by ReliefF.

Reshef et al. (2011) describe a new metric to quantify the relationship between two variables called the MIC. Their method partitions the two-dimensional area defined by the predictor and outcome into sets of two-dimensional grids. For example, Fig. 18.10 shows the scatter plot of the solubility data where four random 4×3 grids are created for the number of carbon atoms predictor. Within each grid, the number of data points is calculated and used to compute the *mutual information* statistic (Brillinger 2004), which is related to the information criteria described in Chap. 14 for C4.5 and C5.0 decision trees. Many different configurations of the same grid size are evaluated and the largest mutual information value is determined. This process is repeated for many different grid sizes (e.g., 2×2, 10×3). The compendium of mutual information values across all the bins are normalized and the largest value across all the bin configurations is used as the strength of association between the predictor and the outcome.

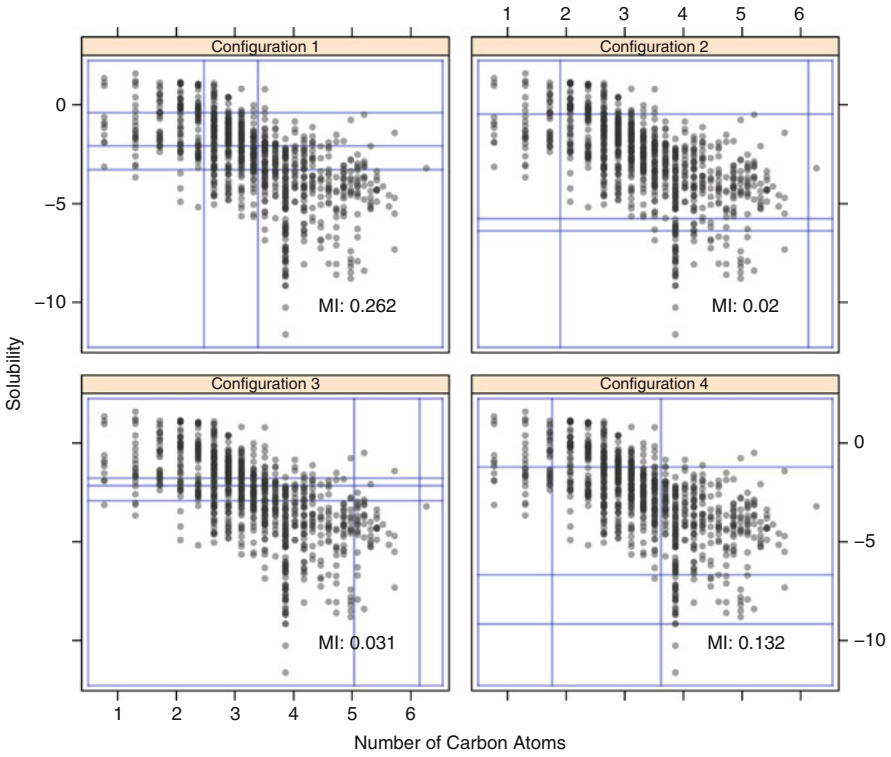

Fig. 18.10: Examples of how the maximal information criterion (MIC) is created. Each panel shows a random configuration of 4×3 grids for one of the solubility predictors with the corresponding the mutual information statistic

The authors demonstrate that this method can detect many types of relationships, such as sin waves, ellipses, and other highly nonlinear patterns. The metric has a similar scale to the simple correlation statistic where a value of zero indicates no relationship, whereas a value of one is indicative of an extremely strong relationship. One potential issue with such a general technique is that it may not do as well as others under some circumstances. For example, if the true relationship between the predictor and outcome were linear, the simple correlation statistic may perform better since it is specific to linear trends.

The MIC statistic was previously shown in Fig. 18.1 for the continuous solubility predictors. Here, the MIC values have a strong relationship with the other metrics. For the cell segmentation data shown in Fig. 18.6, the MIC statistic has a strong correlation with the absolute t-statistics, the area under the ROC curve, and random forest scores. There is smaller correlation with ReliefF.

18.4 Computing

This section uses functions from the following R packages: AppliedPredictiveModeling, caret, CORElearn, minerva, pROC, and randomForest.

The cell segmentation data from Chap. 3 are used and can be found in the caret package. The solubility data from Chaps. 6 through 9 are also used and can be found in the AppliedPredictiveModeling package. Finally, the grant application data from Chaps. 12 through 15 are also used (see Sect. 12.7 for reading and processing these data).

Numeric Outcomes

To estimate the correlations between the predictors and the outcome, the `cor` function is used. For example,

```
> library(AppliedPredictiveModeling)
> data(solubility)
> cor(solTrainXtrans$NumCarbon, solTrainY)
```

```
[1] -0.6067917
```

To get results for all of the numeric predictors, the `apply` function can be used to make the same calculations across many columns

```
> ## Determine which columns have the string "FP" in the name and
> ## exclude these to get the numeric predictors
> fpCols<- grepl("FP", names(solTrainXtrans))
> ## Exclude these to get the numeric predictor names
> numericPreds <- names(solTrainXtrans)[!fpCols]


> corrValues <- apply(solTrainXtrans[, numericPreds],
+                     MARGIN = 2,
+                     FUN = function(x, y) cor(x, y),
+                     y = solTrainY)
> head(corrValues)
```

```
     MolWeight      NumAtoms NumNonHAtoms      NumBonds NumNonHBonds
    -0.6585284    -0.4358113    -0.5836236    -0.4590395    -0.5851968
  NumMultBonds
    -0.4804159
```

The obtain the rank correlation, the `corr` function has an option `method = "spearman"`.

The LOESS smoother can be accessed with the `loess` function in the stats library. The formula method is used to specify the model:

```
> smoother <- loess(solTrainY ~ solTrainXtrans$NumCarbon)
> smoother
```

```
Call:
loess(formula = solTrainY ~ solTrainXtrans$NumCarbon)

Number of Observations: 951
Equivalent Number of Parameters: 5.3
Residual Standard Error: 1.548
```

The lattice function `xyplot` is convenient for displaying the LOESS fit:

```
> xyplot(solTrainY ~ solTrainXtrans$NumCarbon,
+        type = c("p", "smooth"),
+        xlab = "# Carbons",
+        ylab = "Solubility")
```

The caret function `filterVarImp` with the `nonpara = TRUE` option (for nonparametric regression) creates a LOESS model for each predictor and quantifies the relationship with the outcome:

```
> loessResults <- filterVarImp(x = solTrainXtrans[, numericPreds],
+                              y = solTrainY,
+                              nonpara = TRUE)
> head(loessResults)
```

```
                Overall
MolWeight    0.4443931
NumAtoms     0.1899315
NumNonHAtoms 0.3406166
NumBonds     0.2107173
NumNonHBonds 0.3424552
NumMultBonds 0.2307995
```

The minerva package can be used to calculate the MIC statistics between the predictors and outcomes. The `mine` function computes several quantities including the MIC value:

```
> library(minerva)
> micValues <-  mine(solTrainXtrans[, numericPreds], solTrainY)
> ## Several statistics are calculated
> names(micValues)
```

```
[1] "MIC"   "MAS"   "MEV"   "MCN"   "MICR2"
```

```
> head(micValues$MIC)
```

```
                     Y
MolWeight    0.4679277
NumAtoms     0.2896815
NumNonHAtoms 0.3947092
NumBonds     0.3268683
NumNonHBonds 0.3919627
NumMultBonds 0.2792600
```

For categorical predictors, the simple `t.test` function computes the difference in means and the p-value. For one predictor:

```
> t.test(solTrainY ~ solTrainXtrans$FP044)

         Welch Two Sample t-test

  data:  solTrainY by solTrainXtrans$FP044
  t = 15.1984, df = 61.891, p-value < 2.2e-16
  alternative hypothesis: true difference in means is not equal to 0
  95 percent confidence interval:
   3.569300 4.650437
  sample estimates:
  mean in group 0 mean in group 1
        -2.472237       -6.582105
```

This approach can be extended to all predictors using `apply` in a manner similar to the one shown above for correlations.

```
> getTstats <- function(x, y)
+   {
+     tTest <- t.test(y~x)
+     out <- c(tStat = tTest$statistic, p = tTest$p.value)
+     out
+   }
> tVals <- apply(solTrainXtrans[, fpCols],
+                MARGIN = 2,
+                FUN = getTstats,
+                y = solTrainY)
> ## switch the dimensions
> tVals <- t(tVals)
> head(tVals)
          tStat.t            p
  FP001 -4.022040 6.287404e-05
  FP002 10.286727 1.351580e-23
  FP003 -2.036442 4.198619e-02
  FP004 -4.948958 9.551772e-07
  FP005 10.282475 1.576549e-23
  FP006 -7.875838 9.287835e-15
```

Categorical Outcomes

The `filterVarImp` function also calculates the area under the ROC curve when the outcome variable is an R factor variable:

```
> library(caret)
> data(segmentationData)
> cellData <- subset(segmentationData, Case == "Train")
> cellData$Case <- cellData$Cell <- NULL
> ## The class is in the first column
> head(names(cellData))
  [1] "Class"       "AngleCh1"    "AreaCh1"     "AvgIntenCh1"
  [5] "AvgIntenCh2" "AvgIntenCh3"
```

```
> rocValues <- filterVarImp(x = cellData[,-1],
+                           y = cellData$Class)
> ## Column is created for each class
> head(rocValues)
                    PS        WS
AngleCh1    0.5025967 0.5025967
AreaCh1     0.5709170 0.5709170
AvgIntenCh1 0.7662375 0.7662375
AvgIntenCh2 0.7866146 0.7866146
AvgIntenCh3 0.5214098 0.5214098
AvgIntenCh4 0.6473814 0.6473814
```

This is a simple wrapper for the functions `roc` and `auc` in the pROC package. When there are three or more classes, `filterVarImp` will compute ROC curves for each class versus the others and then returns the largest area under the curve.

The Relief statistics can be calculated using the CORElearn package. The function `attrEval` will calculate several versions of Relief (using the `estimator` option):

```
> library(CORElearn)
> reliefValues <- attrEval(Class ~ ., data = cellData,
+                          ## There are many Relief methods
+                          ## available. See ?attrEval
+                          estimator = "ReliefFequalK",
+                          ## The number of instances tested:
+                          ReliefIterations = 50)
> head(reliefValues)
    AngleCh1     AreaCh1 AvgIntenCh1 AvgIntenCh2 AvgIntenCh3 AvgIntenCh4
 0.01631332  0.02004060  0.09402596  0.17200400  0.09268398  0.02672168
```

This function can also be used to calculate the gain ratio, Gini index, and other scores. To use a permutation approach to investigate the observed values of the ReliefF statistic, the AppliedPredictiveModeling package has a function `permuteRelief`:

```
> perm <- permuteRelief(x = cellData[,-1],
+                       y = cellData$Class,
+                       nperm = 500,
+                       estimator = "ReliefFequalK",
+                       ReliefIterations = 50)
```

The permuted ReliefF scores are contained in a sub-object called `permutations`:

```
> head(perm$permutations)
    Predictor        value
  1  AngleCh1 -0.009364024
  2  AngleCh1  0.011170669
  3  AngleCh1 -0.020425694
  4  AngleCh1 -0.037133238
  5  AngleCh1  0.005334315
  6  AngleCh1  0.010394028
```

The permutation distributions for the ReliefF scores can be helpful. Histograms, such as those show in Fig. 18.9, can be created with the syntax:

```
> histogram(~ value|Predictor,
+           data = perm$permutations)
```

Also, the standardized versions of the scores are in the sub-object called `standardized` and represent the number of standard deviations that the observed ReliefF values (i.e., without permuting) are from the center of the permuted distribution:

```
> head(perm$standardized)
     AngleCh1     AreaCh1 AvgIntenCh1 AvgIntenCh2 AvgIntenCh3 AvgIntenCh4
    -1.232653    3.257958    3.765691    8.300906    4.054288    1.603847
```

The MIC statistic can be computed as before but with a binary dummy variable encoding of the classes:

```
> micValues <- mine(x = cellData[,-1],
+                   y = ifelse(cellData$Class == "PS", 1, 0))
>

> head(micValues$MIC)
                     Y
  AngleCh1    0.1310570
  AreaCh1     0.1080839
  AvgIntenCh1 0.2920461
  AvgIntenCh2 0.3294846
  AvgIntenCh3 0.1354438
  AvgIntenCh4 0.1665450
```

To compute the odds ratio and a statistical test of association, the `fisher.test` function in the stats library can be applied. For example, to calculate these statistics for the grant objects created in Sect. 12.7:

```
> Sp62BTable <- table(training[pre2008, "Sponsor62B"],
+                     training[pre2008, "Class"])
> Sp62BTable
     successful unsuccessful
  0       3226         3356
  1          7           44
> fisher.test(Sp62BTable)

        Fisher's Exact Test for Count Data

  data:  Sp62BTable
  p-value = 2.644e-07
  alternative hypothesis: true odds ratio is not equal to 1
  95 percent confidence interval:
    2.694138 15.917729
  sample estimates:
  odds ratio
    6.040826
```

When the predictor has more than two classes, a single odds ratio cannot be computed, but the p-value for association can still be utilized:

```
> ciTable <- table(training[pre2008, "CI.1950"],
+                  training[pre2008, "Class"])
> ciTable
      successful unsuccessful
   0       2704         2899
   1        476          455
   2         45           39
   3          7            7
   4          1            0
> fisher.test(ciTable)

        Fisher's Exact Test for Count Data

data:  ciTable
p-value = 0.3143
alternative hypothesis: two.sided
```

In some cases, Fisher's exact test may be computationally prohibitive. In these cases, the χ^2 test for association can be computed:

```
> DayTable <- table(training[pre2008, "Weekday"],
+                   training[pre2008, "Class"])
> DayTable
         successful unsuccessful
   Fri          542          880
   Mon          634          455
   Sat          615          861
   Sun          223            0
   Thurs        321          246
   Tues         377          309
   Wed          521          649
> chisq.test(DayTable)

        Pearson's Chi-squared test

data:  DayTable
X-squared = 400.4766, df = 6, p-value < 2.2e-16
```

Model-Based Importance Scores

As described in the previous chapters, many models have built-in approaches for measuring the aggregate effect of the predictors on the model. The caret package contains a general class for calculating or returning these values. As of this writing, there are methods for 27 R classes, including: `C5.0`, `JRip`, `PART`, `RRF`, `RandomForest`, `bagEarth`, `classbagg`, `cubist`, `dsa`, `earth`, `fda`, `gam`, `gbm`, `glm`, `glmnet`, `lm`, `multinom`, `mvr`, `nnet`, `pamrtrained`, `plsda`, `randomForest`, `regbagg`, `rfe`, `rpart`, `sbf`, and `train`.

To illustrate, a random forest model was fit to the cell segmentation data:

```
> library(randomForest)
> set.seed(791)
> rfImp <- randomForest(Class ~ ., data = segTrain,
+                       ntree = 2000,
+                       importance = TRUE)
```

The randomForest package contains a function called `importance` that returns the relevant metric. The `varImp` function standardizes across models:

```
> head(varImp(rfImp))
```

```
                    PS        WS
AngleCh1    -1.002852 -1.002852
AreaCh1      8.769884  8.769884
AvgIntenCh1 21.460666 21.460666
AvgIntenCh2 22.377451 22.377451
AvgIntenCh3  7.690371  7.690371
AvgIntenCh4  9.108741  9.108741
```

Note that some models return a separate score for each class while others do not.

When using the `train` function, the `varImp` function executes the appropriate code based on the value of the `method` argument. When the model does not have a built-in function for measuring importances, `train` employs a more general approach (as described above).

Exercises

18.1. For the churn data described in Exercise 12.3:

(a) Calculate the correlations between predictors. Are there strong relationships between these variables? How does this compare to what one would expect with these attributes?
(b) Assess the importance of the categorical predictors (i.e., area code, voice mail plan, etc) individually using the training set.
(c) Also estimate the importance scores of the continuous predictors individually.
(d) Now use ReliefF to jointly estimate the importance of the predictors. Is there a difference in rankings? Why or why not?

18.2. For the oil type data described in Exercise 4.4, estimate variable importance scores. Does the large number of classes affect the process of quantifying the importances?

18.3. The UCI Abalone data (http://archive.ics.uci.edu/ml/datasets/Abalone) consist of data from 4,177 abalones. The data contain measurements of the type (male, female, and infant), the longest shell measurement, the diameter,

height, and several weights (whole, shucked, viscera, and shell). The outcome is the number of rings. The age of the abalone is the number of rings plus 1.5.

The data are contained in the AppliedPredictiveModeling package:

```
> library(AppliedPredictiveModeling)
> data(abalone)
> str(abalone)
'data.frame':        4177 obs. of  9 variables:
 $ Type         : Factor w/ 3 levels "F","I","M": 3 3 1 3 2 2 1 1 3 1 ...
 $ LongestShell : num  0.455 0.35 0.53 0.44 0.33 0.425 0.53 0.545 ...
 $ Diameter     : num  0.365 0.265 0.42 0.365 0.255 0.3 0.415 0.425 ...
 $ Height       : num  0.095 0.09 0.135 0.125 0.08 0.095 0.15 0.125  ...
 $ WholeWeight  : num  0.514 0.226 0.677 0.516 0.205 ...
 $ ShuckedWeight: num  0.2245 0.0995 0.2565 0.2155 0.0895 ...
 $ VisceraWeight: num  0.101 0.0485 0.1415 0.114 0.0395 ...
 $ ShellWeight  : num  0.15 0.07 0.21 0.155 0.055 0.12 0.33 0.26 ...
 $ Rings        : int  15 7 9 10 7 8 20 16 9 19 ...
> head(abalone)
  Type LongestShell Diameter Height WholeWeight ShuckedWeight
1    M        0.455    0.365  0.095      0.5140        0.2245
2    M        0.350    0.265  0.090      0.2255        0.0995
3    F        0.530    0.420  0.135      0.6770        0.2565
4    M        0.440    0.365  0.125      0.5160        0.2155
5    I        0.330    0.255  0.080      0.2050        0.0895
6    I        0.425    0.300  0.095      0.3515        0.1410
  VisceraWeight ShellWeight Rings
1        0.1010       0.150    15
2        0.0485       0.070     7
3        0.1415       0.210     9
4        0.1140       0.155    10
5        0.0395       0.055     7
6        0.0775       0.120     8
```

(a) Plot the data to assess the functional relationships between the predictors and the outcome.
(b) Use scatter plots and correlation plots to understand how the predictors relate to one another.
(c) Estimate variable importance scores for each predictor. Develop an approach to determining a reduced set of nonredundant predictors.
(d) Apply principal component analysis to the continuous predictors to determine how many distinct underlying pieces of information are in the data. Would feature extraction help these data?