Chapter 19
An Introduction to Feature Selection

Determining which predictors should be included in a model is becoming one of the most critical questions as data are becoming increasingly high-dimensional. For example:

- In business, companies are now more proficient at storing and accessing large amounts of information on their customers and products. Large databases are often mined to discover crucial relationships (Lo 2002).
- In pharmaceutical research, chemists can calculate thousands of predictors using quantitative structure-activity relationship (QSAR) methodology described in the regression chapters for numerically describing various aspects of molecules. As an example, one popular software suite calculates 17 flavors of a compound's surface area. These predictors can be categorical or continuous and can easily number in the tens of thousands.
- In biology, a vast array of biological predictors can be measured at one time on a sample of biological material such as blood. RNA expression profiling microarrays can measure thousands of RNA sequences at once. Also, DNA microarrays and sequencing technologies can comprehensively determine the genetic makeup of a sample, producing a wealth of numeric predictors. These technologies have rapidly advanced over time, offering ever larger quantities of information.

From a practical point of view, a model with less predictors may be more interpretable and less costly especially if there is a cost to measuring the predictors. Statistically, it is often more attractive to estimate fewer parameters. Also, as we will soon see, some models may be negatively affected by non-informative predictors.

Some models are naturally resistant to non-informative predictors. Tree- and rule-based models, MARS and the lasso, for example, intrinsically conduct feature selection. For example, if a predictor is not used in any split during the construction of a tree, the prediction equation is functionally independent of the predictor.

M. Kuhn and K. Johnson, *Applied Predictive Modeling*,
DOI 10.1007/978-1-4614-6849-3_19,

An important distinction to be made in feature selection is that of *supervised* and *unsupervised* methods. When the outcome is ignored during the elimination of predictors, the technique is *unsupervised*. Examples of these filters were described in Chap. 3 and included removing predictors that have high correlations with other predictors or that have very sparse and unbalanced distributions (i.e., near-zero variance predictors). In each case, the outcome is independent of the filtering calculations. For *supervised methods*, predictors are specifically selected for the purpose of increasing accuracy or to find a subset of predictors to reduce the complexity of the model. Here, the outcome is typically used to quantify the importance of the predictors (as illustrated s in Chap. 18).

The issues related to each type of feature selection are very different, and the literature for this topic is large. Subsequent sections highlight several critical topics including the need for feature selection, typical approaches, and common pitfalls.

19.1 Consequences of Using Non-informative Predictors

Feature selection is primarily focused on removing non-informative or redundant predictors from the model. As with many issues discussed in this text, the importance of feature selection depends on which model is being used. Many models, especially those based on regression slopes and intercepts, will estimate parameters for every term in the model. Because of this, the presence of non-informative variables can add uncertainty to the predictions and reduce the overall effectiveness of the model.

The solubility QSAR data which were analyzed in Chaps. 6–9 and 20 will again be used to demonstrate the effect of non-informative predictors on various models. The data most likely contain non-informative predictors already. Despite this, we will tune and rebuild several models after adding more irrelevant predictors. To do this, the original predictor data are altered by randomly shuffling their rows (independently for each column). As a result, there should be no connection between the new predictors and the solubility values. This procedure preserves the nature of the individual predictors (i.e., fingerprints with the same frequency distribution) but has the side effect of eliminating the between-predictor correlations. If the inclusion of correlated predictors would add additional injury to a particular model, that impact will not be reflected in this exercise.

To quantify the effect of additional predictors, models were created that used the original 228 predictors in the data and then supplemented with either 10, 50, 100, 200, 300, 400, or 500 additional, non-informative predictors. The models were tuned and finalized, and the test set RMSE values were calculated. The models assessed in this manner were linear regression, partial

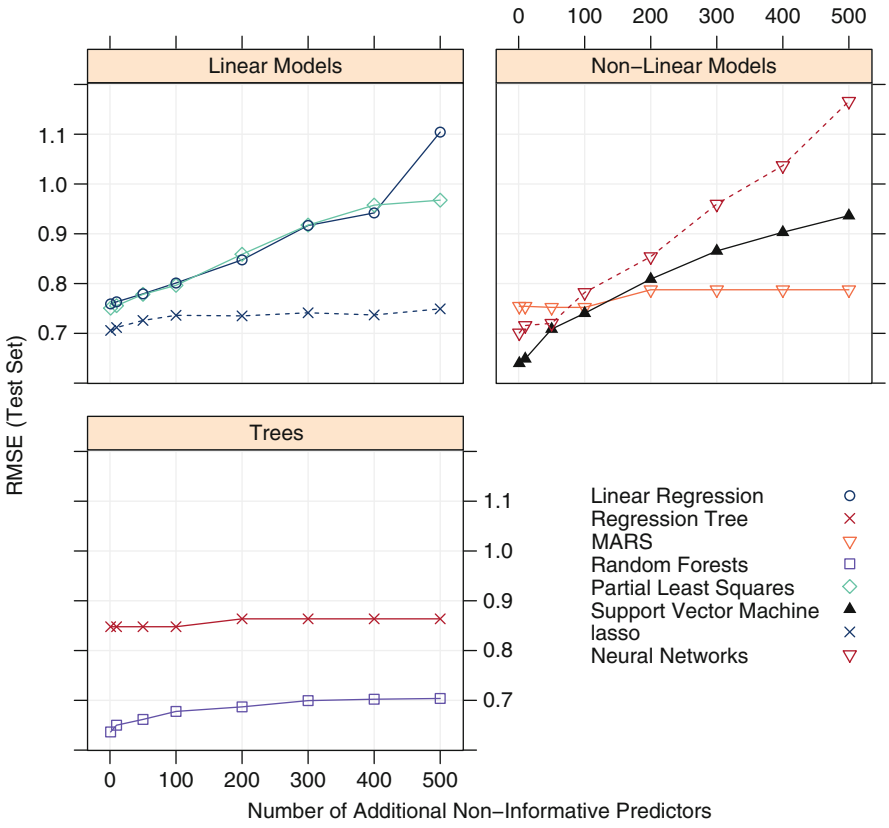

Fig. 19.1: Test set RMSE profiles for solubility models when non-informative predictors are added

least squares, single regression trees, multivariate adaptive regression splines, random forests, neural networks, and radial basis function support vector machines.

Figure 19.1 shows the test set results. As expected, regression trees and MARS models were not affected due to the built-in feature selection. Random forests showed a moderate degradation in performance. The issue here is that the random selection of predictors for splitting can coerce the model into including some unimportant predictors. However, their inclusion does not seriously impact the overall model. The parametrically structured models, such as linear regression, partial least squares, and neural networks, were most affected. Neural networks appear to have the most extensive issues, perhaps due to the excess number of parameters added to the model. For example, a network with 5 hidden units would originally have estimated 961 regression parameters. Adding 500 irrelevant predictors increases the number

of parameters to 3,461. Given that there are 951 data points in the training set, this may be too difficult of a problem to solve without over-fitting. Support vector machines also saw a substantial increase in the RMSE, which is consistent with comments made by Hastie et al. (2008, Chap. 12).

Given the potential negative impact, there is the need to find a smaller subset of predictors. Our basic goal is to reduce their number in a way that maximizes performance. This is similar to the previous discussions: how can we reduce complexity without negatively affecting model effectiveness?

19.2 Approaches for Reducing the Number of Predictors

Apart from models with built-in feature selection, most approaches for reducing the number of predictors can be placed into two main categories (John et al. 1994):

- *Wrapper* methods evaluate multiple models using procedures that add and/or remove predictors to find the optimal combination that maximizes model performance. In essence, wrapper methods are search algorithms that treat the predictors as the inputs and utilize model performance as the output to be optimized.
- *Filter methods* evaluate the relevance of the predictors outside of the predictive models and subsequently model only the predictors that pass some criterion. For example, for classification problems, each predictor could be individually evaluated to check if there is a plausible relationship between it and the observed classes. Only predictors with important relationships would then be included in a classification model. Saeys et al. (2007) survey filter methods.

Both approaches have advantages and drawbacks. Filter methods are usually more computationally efficient than wrapper methods, but the selection criterion is not directly related to the effectiveness of the model. Also, most filter methods evaluate each predictor separately, and, consequently, redundant (i.e., highly-correlated) predictors may be selected and important interactions between variables will not be able to be quantified. The downside of the wrapper method is that many models are evaluated (which may also require parameter tuning) and thus an increase in computation time. There is also an increased risk of over-fitting with wrappers.

The following two sections describe these methods in more detail, and a case study is used to illustrate their application.

19.3 Wrapper Methods

As previously stated, wrapper methods conduct a search of the predictors to determine which, when entered into the model, produce the best results. A simple example is classical forward selection for linear regression (Algorithm 19.1). Here, the predictors are evaluated (one at a time) in the current linear regression model. A statistical hypothesis test can be conducted to see if each of the newly added predictors is statistically significant (at some predefined threshold). If at least one predictor has a p-value below the threshold, the predictor associated with the smallest value is added to the model and the process starts again. The algorithm stops when none of the p-values for the remaining predictors are statistically significant. In this scheme, linear regression is the *base learner* and forward selection is the *search procedure*. The *objective function* is the quantity being optimized which, in this case, is statistical significance as represented by the p-value.

There are a few issues with this approach:

1. The forward search procedure is *greedy*, meaning that it does not reevaluate past solutions.
2. The use of repeated hypothesis tests in this manner invalidates many of their statistical properties since the same data are being evaluated numerous times. See Fig. 19.2 for a nontechnical illustration of this issue.
3. Maximizing statistical significance may not be the same as maximizing more relevant accuracy-based quantities.

```
Create an initial model containing only an intercept term.
repeat
    for each predictor not in the current model do
        Create a candidate model by adding the predictor to the current model
        Use a hypothesis test to estimate the statistical significance of the new model term
    end
    if the smallest p-value is less than the inclusion threshold then
        Update the current model to include a term corresponding to the most statistically significant predictor
    else
        Stop
    end
until no statistically significant predictors remain outside the model
```

Algorithm 19.1: Classical forward selection for linear regression models

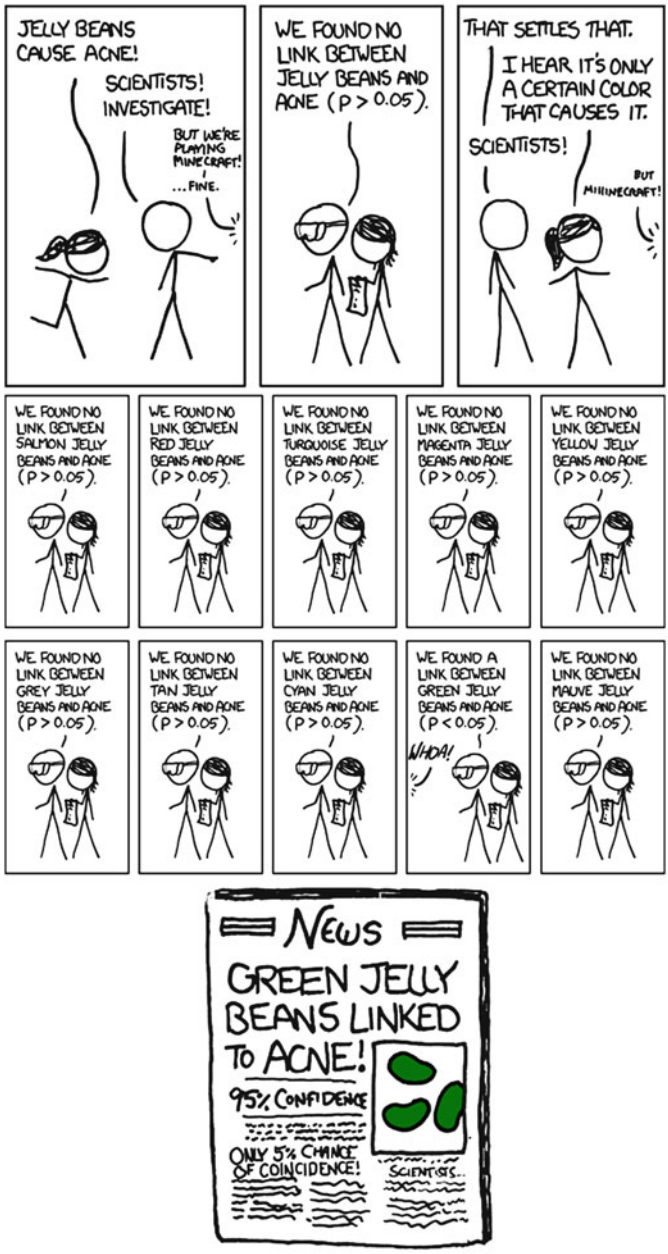

Fig. 19.2: The perils of repeated testing of the same data (Randall Munroe, http://xkcd.com/882, modified for space)

For the first issue, more complex search procedures may be more effective. Several such algorithms are discussed below. The second issue has been extensively studied (Derksen and Keselman 1992; Olden and Jackson 2000; Harrell 2001; Whittingham et al. 2006).[1] Harrell (2001) summarizes the use of p-values during automated model selection:

> ... if this procedure had just been proposed as a statistical method, it would most likely be rejected because it violates every principal of statistical estimation and hypothesis testing.

The second and third issues can be mitigated by using some measure of predictive performance such as RMSE, classification accuracy, or the area under the ROC curve.

Suppose that the RMSE was the objective function instead of statistical significance. Here, the algorithm would be the same but would add predictors to the model that results in the smallest model RMSE. The process would continue until some predefined number of predictors has been reached or the full model is used. From this process, the RMSE can be monitored to determine a point where the error began to increase. The subset size associated with the smallest RMSE would be chosen. As one would suspect, the main pitfall here is obtaining good estimates of the error rate that are not subject to over-fitting caused by the model or the feature selection process (see Sect. 19.5 below).

There are several other criteria that can penalize performance based on how many predictors are in the model. For example, when choosing between two models with the same RMSE, the simpler model with fewer predictors is preferable. For linear regression, a commonly used statistic is the *Akaike Information Criterion*, a penalized version of the sums of squares error:

$$\text{AIC} = n \log\left(\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2\right) + 2P$$

where P is the number of terms in the model. Here, no inferential statements are being made about this quantity, and, all other things being equal, simplicity is favored over complexity. Similar AIC statistics are also available for other models, such as logistic regression. Searching the predictor space for the model with the smallest AIC may help avoid over-fitting.[2]

Another approach is correlation-based feature selection (Hall and Smith 1997), which attempts to find the best subset of predictors that have strong

[1] The majority of the scholarship on this problem has revolved around *stepwise* model selection, a variant of forward selection. However, the results apply to any search procedure using hypothesis testing in this manner.

[2] It should be noted that the AIC statistic is designed for preplanned comparisons between models (as opposed to comparisons of many models during automated searches).

correlations with the outcome but weak between-predictor correlations. To do this, one possible metric is

$$G = \frac{P\, R_y}{\sqrt{P + P(P-1)\bar{R}_x}}$$

where R_y is a measure of correlation between the candidate predictor and the outcome, and $\bar{R}_x$ is the average correlation between the current predictor and the P predictors already included in the subset. At each stage of the search, this metric can rank predictors as a function of effectiveness and redundancy.

For models built *to predict*, rather than explain, there are two important overall points:

- Many of the criticisms of wrapper methods listed above[3] focus on the use of statistical hypothesis tests.
- As in many other cases presented in this text, methodologies based on dubious statistical principals may still lead to very accurate models. The key protection in these instances is a thorough, methodical validation process with independent data.

The following subsections describe different search methods to use with wrapper methods.

Forward, Backward, and Stepwise Selection

Classical forward selection was previously described in Algorithm 19.1. *Stepwise selection* is a popular modification where, after each candidate variable is added to the model, each term is reevaluated for removal from the model. In some cases, the p-value threshold for adding and removing predictors can be quite different (Derksen and Keselman 1992). Although this makes the search procedure less greedy, it exacerbates the problem of repeated hypothesis testing. In *backward selection*, the initial model contains all P predictors which are then iteratively removed to determine which are not significantly contributing to the model. These procedures can be improved using non-inferential criteria, such as the *AIC* statistic, to add or remove predictors from the model.

Guyon et al. (2002) described a backward selection algorithm (called *recursive feature elimination*) that avoids refitting many models at each step of the search. When the full model is created, a measure of variable importance is computed that ranks the predictors from most important to least. The importance calculations can be model based (e.g., the random forest importance criterion) or using a more general approach that is independent of the full model. At each stage of the search, the least important predictors are

[3] However, Derksen and Keselman (1992) and Henderson and Velleman (1981) make arguments against automated feature selection in general.

1 Tune/train the model on the training set using all P predictors
2 Calculate model performance
3 Calculate variable importance or rankings
4 **for** *each subset size* S_i, $i = 1 \ldots S$ **do**
5 Keep the S_i most important variables
6 [Optional] Pre-process the data
7 Tune/train the model on the training set using S_i predictors
8 Calculate model performance
9 [Optional] Recalculate the rankings for each predictor
10 **end**
11 Calculate the performance profile over the S_i
12 Determine the appropriate number of predictors (i.e. the S_i associated with the best performance)
13 Fit the final model based on the optimal S_i

Algorithm 19.2: Backward selection via the recursive feature elimination algorithm of Guyon et al. (2002)

iteratively eliminated prior to rebuilding the model. As before, once a new model is created, the objective function is estimated for that model. The process continues for some predefined sequence, and the subset size corresponding to the best value of the objective function is used as the final model. This process is outlined in more detail in Algorithm 19.2 and is illustrated in Sect. 19.6.

While it is easy to treat the RFE algorithm as a black box, there are some considerations that should be made. For example, when the outcome has more than two classes, some classes may have a large degree of separation from the rest of the training set. As such, it may be easier to achieve smaller error rates for these classes than the others. When the predictors are ranked for selection, the predictors associated with the "easy" classes may saturate the positions for the highest ranks. As a result, the difficult classes are neglected and maintain high error rates. In this case, class-specific importance scores can aid in selecting a more balanced set of predictors in an effort to balance the error rates across all the classes.

Simulated Annealing

A multitude of modern search procedures exist that can be applied to the feature selection problem. *Simulated annealing* (Bohachevsky et al. 1986) mimics the process of metal cooling. Algorithm 19.3 describes this process in detail. An initial subset of predictors is selected and is used to estimate

Generate an initial random subset of predictors
for *iterations* $i = 1 \ldots t$ **do**
  Randomly perturb the current best predictor set
  [Optional] Pre-process the data
  Tune/train the model using this predictor set
  Calculate model performance (E_i)
  if $E_i < E_{best}$ **then**
    Accept current predictor set as best
    Set $E_{best} = E_i$
  else
    Calculate the probability of accepting the current predictor set

$$p_i^a = \exp\left[(E_{best} - E_i)/T\right]$$

    Generate a random number U between $[0, 1]$
    if $p_i^a \leq U$ **then**
      Accept current predictor set as best
      Set $E_{best} = E_i$
    else
      Keep current best predictor set
    end
  end
end
Determine the predictor set associated with the smallest E_i across all iterations
Finalize the model with this predictor set

Algorithm 19.3: Simulated annealing for feature selection. E is a measure of performance where small values are best and T is a temperature value that changes over iterations

performance of the model (denoted here as E_1, for the initial error rate). The current predictor subset is slightly changed, and another model is created with an estimated error rate of E_2. If the new model is an improvement over the previous one (i.e., $E_2 < E_1$), the new feature set is accepted. However, if it is worse, it may still be accepted based on some probability p_i^a, where i is the iteration of the process. This probability is configured to decrease over time so that, as i becomes large, it becomes very unlikely that a suboptimal configuration will be accepted. The process continues for some pre-specified number of iterations and the best variable subset across all the iterations,

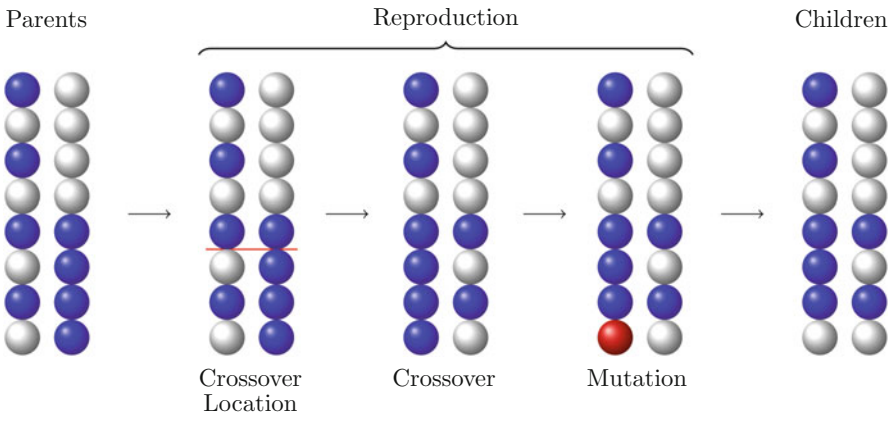

Fig. 19.3: A schematic of the reproduction phase of a genetic algorithm

is used. The idea is to avoid a local optimum (a solution that is currently best but is not best overall). By accepting "bad" solutions, the algorithm is able to continue the search in other spaces and therefore is less greedy.

Genetic Algorithms

A distant kin to simulated annealing are *genetic algorithms* (GAs) (Holland 1975; Goldberg 1989). This optimization tool is based on evolutionary principles of population biology and has been shown to be effective at finding optimal solutions of complex, multivariate functions. Specifically, GAs are constructed to imitate the evolutionary process by allowing the current population of solutions to reproduce, generating children which compete to survive. The most fit survivors are then allowed to reproduce, creating the next generation of children. Over time, generations converge to a fitness plateau (Holland 1992) and an optimal solution can be selected.

As we have seen thus far, the problem of feature selection is inherently a complex optimization problem, where we seek the combination of features that provides an optimal prediction of the response. To employ GAs towards this end, we must frame the feature selection problem in terms of the GA machinery. The subjects of this machinery are chromosomes, which consist of genes and are evaluated based on their fitness. To create the next generation of offspring, two chromosomes reproduce through the process of crossover and mutation, which is illustrated in Fig. 19.3. GAs have been shown to be effective feature selection tools in the fields of chemometrics (Lavine et al. 2002), image analysis (Bhanu and Lin 2003), and finance (Min et al. 2006; Huang et al. 2012).

```
Define the stopping criteria, number of children for each generation (GenSize), and probability of mutation (p_m)
Generate an initial random set of m binary chromosomes, each of length p
repeat
    for each chromosome do
        Tune and train a model and compute each chromosome's fitness
    end
    for reproduction k = 1 ... GenSize/2 do
        Select two chromosomes based on the fitness criterion
        Crossover: Randomly select a loci and exchange each chromosome's genes beyond the loci
        Mutation: Randomly change binary values of each gene in each new child chromosome with probability, p_m
    end
until stopping criteria is met
```

Algorithm 19.4: A genetic algorithm for feature selection

In the context of feature selection, the chromosome is a binary vector that has the same length as the number of predictors in the data set. Each binary entry of the chromosome, or gene, represents the presence or absence of each predictor in the data. The fitness of the chromosome is determined by the model using the predictors indicated by the binary vector. GAs are therefore tasked with finding optimal solutions from the 2^n possible combinations of predictor sets.

To begin the search process, GAs are often initiated with a random selection of chromosomes from the population of all possible chromosomes. Each chromosome's fitness is computed, which determines the likelihood of the chromosome's selection for the process of reproduction. Two chromosomes from the current population are then selected based on the fitness criterion and are allowed to reproduce. In the reproduction phase, the two parent chromosomes are split at a random position (also called *loci*), and the head of one chromosome is combined with the tail of the other chromosome and vice versa. After crossover, the individual entries of the new chromosomes can be randomly selected for mutation in which the current binary value is changed to the other value. Algorithm 19.4 lists these steps.

The crossover phase drives subsequent generations towards optimums in subspaces of similar genetic material. In other words, the search subspace will be narrowed to the space defined by the most fit chromosomes. This means that the algorithm could become trapped in a local optimum. In the context of feature selection, this means that the selected features may pro-

duce an optimal model, but other more optimal feature subsets may exist. The mutation phase enables the algorithm to escape local optimums by randomly perturbing the genetic material. Usually the probability of mutation is kept low (say, $p_m < 0.05$). However, if the practitioner is concerned about local optimums, then the mutation probability can be raised. The effect of raising the mutation probability is a slowing of the convergence to an optimal solution.

19.4 Filter Methods

As previously stated, filter methods evaluate the predictors prior to training the model, and, based on this evaluation, a subset of predictors are entered into the model. Since the scoring of predictors is disconnected from the model, many of the variable importance metrics discussed in Chap. 18 would be appropriate for filtering the variables. Most of these techniques are univariate, meaning that they evaluate each predictor in isolation. In this case, the existence of correlated predictors makes it possible to select important, but redundant, predictors. The obvious consequences of this issue are that too many predictors are chosen and, as a result, collinearity problems arise. Guyon and Elisseeff (2003) discuss several aspects of predictor redundancy during filtering.

Also, if hypothesis tests are used to determine which predictors have statistically significant relationships with the outcome (such as the t-test), the problem of *multiplicity* can occur (see Westfall and Young (1993) and Fig. 19.2). For example, if a confidence level of $\alpha = 0.05$ is used as a p-value threshold for significance, each individual test has a theoretical false-positive rate of 5 %. However, when a large number of simultaneous statistical tests are conducted, the overall false-positive probability increases exponentially. To account for this, p-value adjustment procedures can control the false positive rate. The *Bonferroni correction* (Bland and Altman 1995) is one such procedure. If a p-value cutoff of α is used to define statistical significance for each of M tests, using an alternative cutoff of α/M increases the stringency and will help control the probability of a false-positive results. However, this procedure can be very conservative and limit the number of true-positive results. Other approaches to dealing with multiplicity can be found in Westfall and Young (1993). Also, Ahdesmaki and Strimmer (2010) propose a modified t-test that accounts for the large number of tests being conducted as well as between-predictor correlations.

While filter methods tend to be simple and fast, there is a subjective nature to the procedure. Most scoring methods have no obvious cut point to declare which predictors are important enough to go into the model. Even in the case of statistical hypothesis tests, the user must still select the confidence level to apply to the results. In practice, finding an appropriate value for the confidence value α may require several evaluations until acceptable performance is achieved.

19.5 Selection Bias

While some filtering methods or search procedures are more effective than others, the more important question is related to how model performance is calculated (especially when the sample size is small). Over-fitting the predictors to the training data can occur and, without a proper validation, may go unnoticed. For example, Guyon et al. (2002) demonstrated recursive feature elimination with support vector machine classification models for a well-known colon cancer microarray data set. In these data, measurements on 2,000 unique RNA sequences were used as predictors of disease status (cancer or normal) in a set of 62 patients. No test set was used to verify the results. To monitor performance, leave-one-out cross-validation was used for each model (at Line 8 in Algorithm 19.2). Their analysis showed that the SVM model can achieve accuracy over 95 % using only 4 predictors and 100 % accuracy with models using 8–256 predictors.

The leave-one-out error rates were based on the SVM model *after* the features had been selected. One could imagine that if the feature selection were repeated with a slightly different data set, the results might change. It turns out that, in some cases, the uncertainty induced by feature selection can be much larger than the uncertainty of the model (once the features have been selected). To demonstrate this, Ambroise and McLachlan (2002) conducted the same RFE procedure with the same data set but scrambled the class labels (to coerce all the predictors into being non-informative). They showed that the leave-one-out cross-validation strategy used by Guyon et al. (2002) would achieve zero errors even when the predictors are completely non-informative.

The logical error in the original approach is clear. A model was created from the training set and, using these data, the predictors were evaluated and ranked. If the model is refit using only the important predictors, performance almost certainly improves on the same data set. In addition, the P to n ratio is extreme (2000:62) which appreciably increases the odds that a completely irrelevant predictor will be declared important by chance.

The methodological error occurred because the feature selection was not considered as part of the model-building process. As such, it should be included within the resampling procedure so that the variation of feature selection is captured in the results. The leave-one-out cross-validation procedure on Line 8 in Algorithm 19.2 is ignorant of the steps outside of the model training process that it measures. For example, even if the feature selection process arrives at the true model after evaluating a large number of candidate models, the performance estimates should reflect the process that led to the result.

To properly resample the feature selection process, an "outer" resampling loop is needed that encompasses the entire process. Algorithm 19.5 shows such an resampling scheme for recursive feature elimination. At Line 1, resampling is applied such that the entire feature selection process is within. For example,

if 10-fold cross-validation were in this initial loop, 90 % of the data would be used to conduct the feature selection and the heldout 10 % would be used to evaluate performance (e.g., Line 10) for each subset of predictors. The entire feature selection process would be conducted nine additional times with a different set of heldout samples. In the end, these ten holdout sets determine the optimal number of predictors in the final model (Line 14). Given this result, the entire training set is used to rank the predictors and train the final model (Lines 16 and 17, respectively). Note that an additional "inner" layer of resampling may be needed to optimize any tuning parameters in the model (Lines 3 and 9).

Ambroise and McLachlan (2002) also showed that when the bootstrap, 10-fold cross-validation or repeated test set resampling methods were used properly, the model results were correctly determined to be around the value of the no-information rate.

The additional resampling layer can have a significant negative impact on the computational efficiency of the feature selection process. However, especially with small training sets, this process will drastically reduce the chances of over-fitting the predictors.

The critical point is that it is sometimes easy to commit errors in validating the results of feature selection procedures. For example, Castaldi et al. (2011) conducted a survey of articles that used classification techniques with biological data (e.g., RNA microarrays, proteins) and found that 64 % of the analyses did not appropriately validate the feature selection process. They also showed that these analyses had a significant difference between the resampled performance estimates and those calculated from an independent test set (the test set results were more pessimistic).

The risk of over-fitting in this way is not confined to recursive feature selection or wrappers in general. When using other search procedures or filters for reducing the number of predictors, there is still a risk.

The following situations increase the likelihood of selection bias:

- The data set is small.
- The number of predictors is large (since the probability of a non-informative predictor being falsely declared to be important increases).
- The predictive model is powerful (e.g., black-box models), which is more likely to over-fit the data.
- No independent test set is available.

When the data set is large, we recommend separate data sets for selecting features, tuning models, and validating the final model (and feature set). For small training sets, proper resampling is critical. If the amount of data is not too small, we also recommend setting aside a small test set to double check that no gross errors have been committed.

```
for each resampling iteration do
    Partition data into training and test/hold-back set via resampling
    Tune/train the model on the training set using all P predictors
    Calculate model performance
    Calculate variable importance or rankings
    for Each subset size S_i, i = 1 ... S do
        Keep the S_i most important variables
        [Optional] Pre-process the data
        Tune/train the model on the training set using S_i predictors
        Calculate model performance using the held-back samples
        [Optional] Recalculate the rankings for each predictor
    end
end
Calculate the performance profile over the S_i using the held-back samples
Determine the appropriate number of predictors
Determine the final ranks of each predictor
Fit the final model based on the optimal S_i using the original training set
```

Algorithm 19.5: Recursive feature elimination with proper resampling

19.6 Case Study: Predicting Cognitive Impairment

Alzheimer's disease (AD) is a cognitive impairment disorder characterized by memory loss and a decrease in functional abilities above and beyond what is typical for a given age. It is the most common cause of dementia in the elderly. Biologically, Alzheimer's disease is associated with amyloid-β (Aβ) brain plaques as well as brain tangles associated with a form of the Tau protein.

Diagnosis of AD focuses on clinical indicators that, once manifested, indicate that the progression of the disease is severe and difficult to reverse. Early diagnosis of Alzheimer's disease could lead to a significant improvement in patient care. As such, there is an interest in identifying *biomarkers*, which are measurable quantities that do not involve a clinical evaluation.[4]

[4] There are different types of biomarkers. For example, blood cholesterol levels are believed to indicate cardiovascular fitness. As such, they can be monitored to understand patient health but are also used to characterize the effect of treatments. In the

de Leon and Klunk (2006) and Hampel et al. (2010) contain broad discussion of biomarkers for Alzheimer's disease.

Although medical imaging may be helpful in predicting the onset of the disease there is also an interest in potential low-cost fluid biomarkers that could be obtained from plasma or cerebrospinal fluid (CSF). There are currently several accepted non-imaging biomarkers: protein levels of particular forms of the Aβ and Tau proteins and the Apolipoprotein E genotype. For the latter, there are three main variants: E2, E3 and E4. E4 is the allele most associated with AD (Kim et al. 2009; Bu 2009). Prognostic accuracy may be improved by adding other biomarkers to this list.

Craig-Schapiro et al. (2011) describe a clinical study of 333 patients, including some with mild (but well characterized) cognitive impairment as well as healthy individuals. CSF samples were taken from all subjects. The goal of the study was to determine if subjects in the early states of impairment could be differentiated from cognitively healthy individuals. Data collected on each subject included:

- Demographic characteristics such as age and gender
- Apolipoprotein E genotype
- Protein measurements of Aβ, Tau, and a phosphorylated version of Tau (called pTau)
- Protein measurements of 124 exploratory biomarkers, and
- Clinical dementia scores

For these analyses, we have converted the scores to two classes: impaired and healthy. The goal of this analysis is to create classification models using the demographic and assay data to predict which patients have early stages of disease.

Given the relatively small sample size, one could argue that the best strategy for data splitting is to include all the subjects in the training set to maximize the amount of information for estimating parameters and selecting predictors. However, with the ratio of predictors to data points in these data, the possibility of selection bias is relatively high. For this reason, a small set of subjects will be held back as a test set to verify that no gross methodological errors were committed. The test set contained 18 impaired subjects and 48 cognitively healthy. Any measures of performance calculated from these data will have high uncertainty but will be adequate to detect over-fitting due to feature selection.

For the 267 subjects in the training set, five repeats of 10-fold cross-validation will be used to evaluate the feature selection routines (e.g., the "outer" resampling procedure). If models have additional tuning parameters, simple 10-fold cross-validation is used. Tuning these models will occur at every stage of the feature selection routine. During feature selection and model

latter case, the cholesterol information is treated as surrogate end points for the truly important attributes (e.g., mortality or morbidity).

Table 19.1: Training set frequencies for two encodings of the genotype data

	E2/E2	E2/E3	E2/E4	E3/E3	E3/E4	E4/E4	E2	E3	E4
Impaired	0	6	1	27	32	7	7	65	40
Control	2	24	6	107	51	4	32	182	61

tuning, the area under the ROC curve (for the predicted class probabilities) was optimized.

Figure 19.4 shows the 124×124 correlation matrix of the predictors. There are many strong between-predictor correlations, as indicated by the dark red and blue areas. The average pairwise correlation was 0.27. The minimum and maximum correlations were -0.93 and 0.99, respectively. Many of the correlations are contained within a large group of predictors (as shown by the large red block on the diagonal). This may have a negative effect on the modeling and feature selection process. The analysis shown below uses all the predictors to model the data. Applying an unsupervised filter to reduce the feature set prior to analysis may help improve the results (see Exercise 19.1).

Almost all of the predictors in the data are continuous. However, the Apolipoprotein E genotype is not. For the three genetic variants (E2, E3, and E4), there are six possible values as one copy of the gene is inherited from each parent. The breakdown of these values in the training set is shown in Table 19.1. When broken down into the maternal/paternal genetic combinations, some variants have very small frequencies (e.g., E2/E2). Some predictive models may have issues estimating some parameters, especially if the frequencies are reduced during resampling. An alternate encoding for these data is to create three binary indicators for each allele (e.g., E2, E3 and E4). This version of the information is shown in the three most right-hand columns in Table 19.1. Since these frequencies are not as sparse as the genotype pairs, this encoding is used for all the predictive models shown here.

To illustrate feature selection, recursive feature elimination was used for several models and 66 subset sizes ranging from 1 to 131. Models that require parameter tuning were tuned at each iteration of feature elimination. The following models were evaluated:

- Random forests: The default value of $m_{\text{try}} = \sqrt{p}$ was used at each iteration, and 1,000 trees were used in the forest.
- Linear discriminant analysis: The standard approach was used (i.e., no penalties or internal feature selection).
- Unregularized logistic regression: A model with only main effects was considered.
- K-nearest neighbors: The model was tuned over odd number of neighbors ranging from 5 to 43.

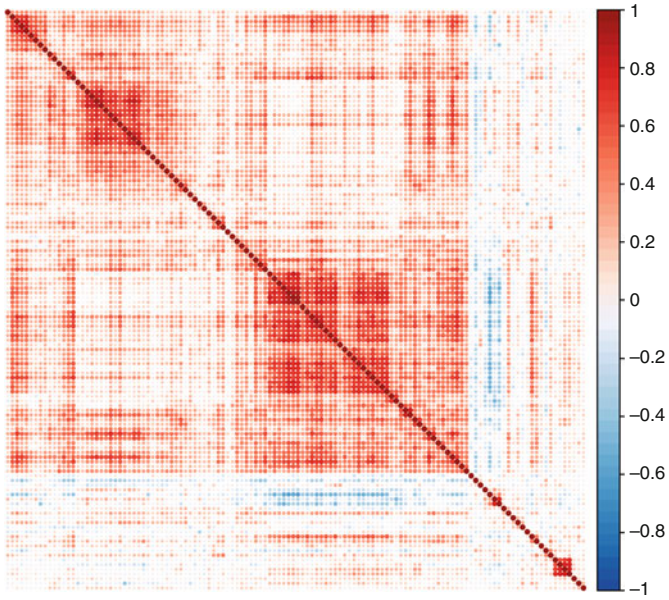

Fig. 19.4: Between predictor correlations for the AD data. Each *row* and *column* represents one of the predictors. The row and column orders have been sorted using clustering methods

- Naïve Bayes: Nonparametric kernel estimates were used for the continuous predictors
- Support Vector Machines: A radial basis function kernel was used. The analytical estimate of σ was used along with cost values ranging from 2^{-2} to 2^9.

The random forest variable importance scores (based on the initial first model) ranked predictors for that model, whereas logistic regression used the absolute value of the Z-statistic for each model parameter. The other models ranked predictors with the area under the ROC curve for each individual predictor.

Figure 19.5 shows the resampling profiles of the feature selection process for each model. Random forest showed very little change until important predictors were eventually removed from the model. This is somewhat expected since, even though random forests do minimal embedded filtering of predictors, non-informative predictors tend to have very little impact on the model predictions. The optimal number of predictors estimated for the model was 7, although there is considerable leeway in this number. LDA showed a large improvement and peaked at 35 predictors and the area under the ROC curve was estimated to be slightly better than the random forest model. Logistic regression showed very little change in performance until about 50 predic-

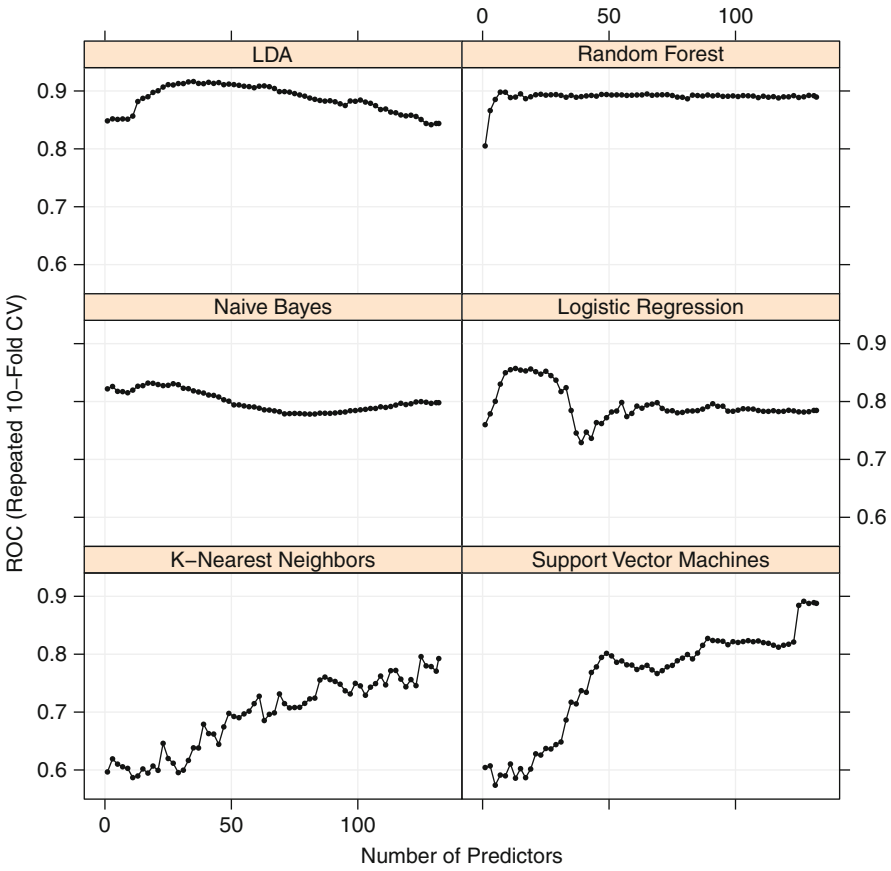

Fig. 19.5: The resampling profiles for the RFE procedure for several models

tors when, after an initial dip, performance improved considerably. The best subset size for this model was 13. However, there is a significant drop in performance for the model once important predictors are removed. Naïve Bayes showed a slight improvement as predictors were removed, culminating at approximately 17 predictors. Unlike the others, this model did not show a sharp downturn in performance when a small number of predictors were left. Both support vector machines and K-nearest neighbors suffered considerably when predictors were removed and were at their best with the full predictor set. In the case of support vector machines, the poor performance may be unrelated to selection bias. A closer examination of the SVM results indicates that the models have high specificity and low sensitivity (see the results in the computing section below). In other words, the poor results are likely a result of the class imbalance in the data.

Table 19.2: Cross-validation results for recursive feature selection

	Full set		Reduced set			
	ROC	C.I.	Size	ROC	C.I.	p-value
LDA	0.844	(0.82 – 0.87)	35	0.916	(0.90 – 0.94)	0
RF	0.891	(0.87 – 0.91)	7	0.898	(0.88 – 0.92)	0.1255
SVM	0.889	(0.87 – 0.91)	127	0.891	(0.87 – 0.91)	0.0192
Logistic reg.	0.785	(0.76 – 0.81)	13	0.857	(0.83 – 0.88)	0
N. Bayes	0.798	(0.77 – 0.83)	17	0.832	(0.81 – 0.86)	0.0002
K-NN	0.849	(0.83 – 0.87)	125	0.796	(0.77 – 0.82)	1.0000

The "C.I." column corresponds to 95 % confidence intervals while the p-value column corresponds to a statistical test that evaluates whether the ROC value for the reduced model was a larger than the curve associated with all of the predictors

Table 19.2 contains a summary of the resampling results for each model. The area under the ROC curve is shown for the full model (i.e., all of the predictors) versus the models resulting from the recursive feature elimination process. In Table 19.2, 95 % confidence intervals are also shown. These were calculated from the 50 resampled estimates produced by repeated cross-validation. Of the models evaluated, LDA has the best estimate of performance. However, based on the confidence intervals, this value is similar in performance to other models (at least within the experimental error reflected in the intervals), including random forest (post feature selection) and support vector machines (before or after feature selection). The p-value column is associated with a statistical test where the null hypothesis is that the reduced predictor set has a larger ROC value than the full set. This value was computed using the paired resampling result process described in Sect. 4.8. Although the support vector machine and K-nearest neighbors models were not substantially enhanced, there was considerable evidence that the other models were improved by recursive feature elimination.

Did the test set show the same trends as the resampling results? Figure 19.6 shows the estimates of the area under the ROC curve for the reduced models. The y-axis corresponds to the resampled ROC values (and confidence intervals) shown previously in Table 19.2. The x-axis has similar values across models, but these were calculated from ROC curves using predictions on the test set of 66 subjects. There was much more uncertainty in the test set results; the widths of the test set confidence intervals tended to be more than 3 times the width of the corresponding intervals from resampling. There is a moderate correlation between the ROC values calculated using the test set and resampling. The test set results were more optimistic than the resampling results for the naïve Bayes, logistic regression, and K-nearest neighbors

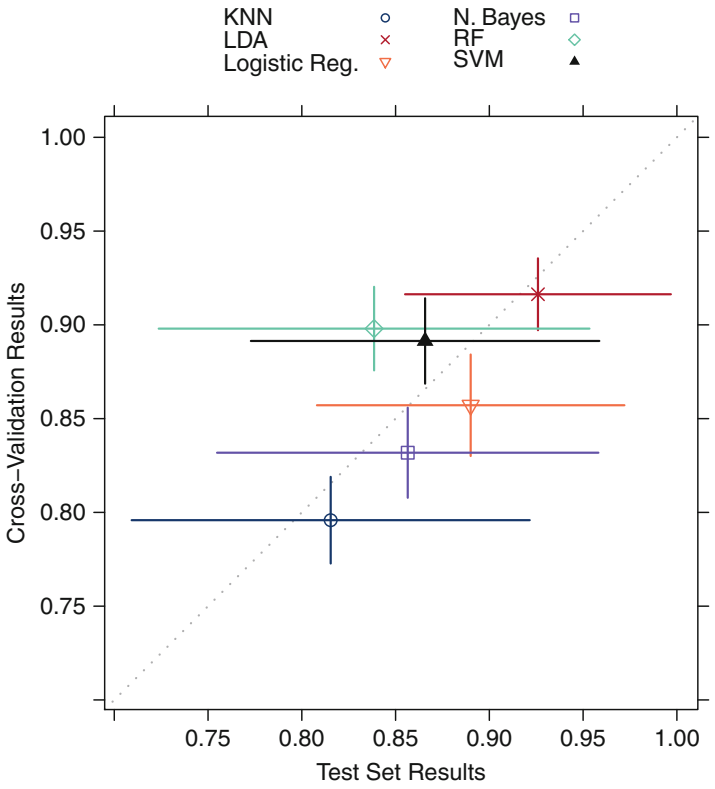

Fig. 19.6: Estimates of the area under the ROC curve for the reduced model using the test set and repeated cross-validation. The horizontal and vertical bars are the 95 % confidence intervals using each method

models. Also, the support vector machine model was ranked higher by cross-validation than the test set. Despite these inconsistencies, the test set results generally reinforce the results seen in Table 19.2; the feature selection process tended to show benefits for several models and did not result in severe selection bias.

The two leading models, LDA and random forests, are quite different. Did these models select the same predictors? This is a somewhat subjective question since, as shown in Fig. 19.5, neither of the profiles for these models had a single peak. Because of this, the optimal subset size is not clear. However, using the numerically optimal result, random forests used 7 and LDA selected 35. All of the random forest predictors are included in the LDA model, but 28 of the LDA predictors are not in the random forest model. Figure 19.7 shows the variable importance estimates for both models (recall that the area under the ROC curve was used for LDA). In this plot, the importance values are the averages of those found during the resampling process for the final

subset size. For example, there are points on the plot for predictors only contained in the LDA model. These predictors have random forest values since that predictor was selected in at least one of the 50 resamples with a subset size of 7 predictors. The rug plot on the y-axis corresponds to the area under the ROC curve for predictors that were not selected in any of the random forest models. There is a fair amount of concordance between the two models; the rank correlation between the common predictors is 0.44. The discordant predictors tend to have low importance scores for both models. Both models contained two predictors, the Aβ and Tau protein assays, with a large influence on the models. The third most important predictor for both models was the modified Tau assay. After this, the importance scores begin to disagree with LDA/ROC using MMP10 and random forest using VEGF.

This underscores the idea that feature selection is a poor method for determining the most significant variables in the *data*, as opposed to which predictors most influenced the *model*. Different models will score the predictors differently. To assess which predictors have individual associations with the outcome, common classical statistical methods are far more appropriate. For example, Craig-Schapiro et al. (2011) also used analysis of covariance and other inferential models to evaluate possible associations. These techniques are designed to yield higher-quality, probabilistic statements about the potential biomarkers than any variable importance method. Finally, the biological aspects of the analysis potentially trump all of the empirical evaluations. While the study is aimed at finding novel biomarkers, a certain degree of scientific legitimacy is needed, as well as prospective experimentation to further validate the results found in these data.

The use of filters with linear discriminant analysis was also explored. For the continuous predictors, a simple t-test was used to generate a p-value for the difference in means of the predictor between the two classes. For the categorical predictors (e.g., gender, Apolipoprotein E genotype), Fisher's exact test was used to calculate a p-value that tests the association between the predictor and the class. Once the p-values were calculated, two approaches were used for filtering:

1. Retain the predictors with raw p-values less than 0.05, i.e., accept a 5 % false-positive rate for each individual comparison.
2. Apply a Bonferroni correction so that predictors are retained when their p-values are less than 0.000379 or 0.05/132.

For the first approach, the process retained 47 predictors out of 132. This process was also resampled using the same repeated cross-validation process used for the wrapper analysis. Across the 50 resamples, the filter selected 46.06 predictors on average, although this number ranged from 38 to 57. Using the predictor subset, an LDA model was fit. The resulting test set area under the ROC curve for this model was 0.918. When the filtering and modeling activities were resampled, a similar area under the curve resulted (AUC: 0.917). The filtered set of variables consisted of members with relatively small

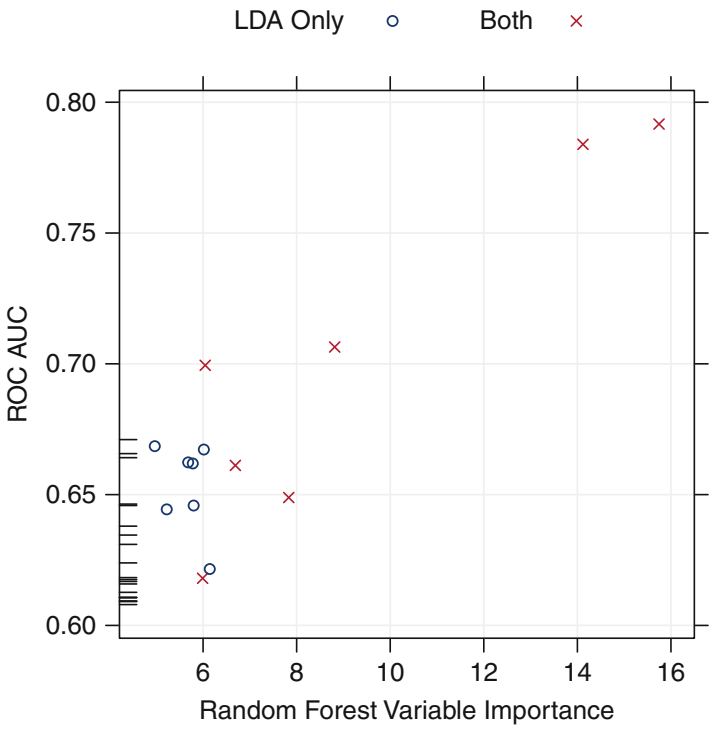

Fig. 19.7: A comparison of two methods for determining variable importance. The random forest values correspond to models with 7 predictors while the ROC values are based on the top 35 predictors (i.e., the best subset for LDA). The values are averaged across the 50 resamples. The "LDA Only" values are predictors that were in the final LDA model but not the random forest model. The hash marks, not the y-axis, are for LDA predictors that were not selected in any of the 50 resampled random forest models

between-correlations with one exception. The Tau and pTau predictors are strongly related (correlation: 0.91). Repeating this process without the pTau predictor resulted in no effective change in the area under the ROC curve.

When the p-value filter is modified to reduce the number of false-positive results, only 17 predictors were selected (13.46 were selected, on average, during cross-validation). Here, the impact on the model is not as clear. The test set area under the ROC curve was a slight improvement (AUC: 0.926), but the resampled estimate was considerably worse, with an area under the curve of 0.856. Given the sample size, there is considerable variability in the test set; the 95 % confidence interval for the area under the curve is (0.841, 1). While the cross-validation process only held out 27 patients on average, this process was repeated 50 times. For this reason, there may be more credence in the more pessimistic resampling estimates.

19.7 Computing

This section discusses data and/or functions from the following packages: AppliedPredictiveModeling, caret, klaR, leaps, MASS, pROC, rms, and stats.

The data are contained in the AppliedPredictiveModeling package. The data objects consist of a data frame of predictors called `predictors` and a factor vector of class values called `diagnosis` (with levels impaired and control). The following code was used to prepare the data for analysis:

```
> library(AppliedPredictiveModeling)
> data(AlzheimerDisease)
> ## Manually create new dummy variables
> predictors$E2 <- predictors$E3 <- predictors$E4 <- 0
> predictors$E2[grepl("2", predictors$Genotype)] <- 1
> predictors$E3[grepl("3", predictors$Genotype)] <- 1
> predictors$E4[grepl("4", predictors$Genotype)] <- 1
>

> ## Split the data using stratified sampling
> set.seed(730)
> split <- createDataPartition(diagnosis, p = .8, list = FALSE)
> ## Combine into one data frame
> adData <- predictors
> adData$Class <- diagnosis
> training <- adData[ split, ]
> testing  <- adData[-split, ]
> ## Save a vector of predictor variable names
> predVars <- names(adData)[!(names(adData) %in% c("Class",  "Genotype"))]

> ## Compute the area under the ROC curve, sensitivity, specificity,
> ## accuracy and Kappa
>
> fiveStats <- function(...) c(twoClassSummary(...),
+                              defaultSummary(...))
> ## Create resampling data sets to use for all models
> set.seed(104)
> index <- createMultiFolds(training$Class, times = 5)
> ## Create a vector of subset sizes to evaluate
> varSeq <- seq(1, length(predVars)-1, by = 2)
```

The code to reproduce the computations in this chapter is extensive and can be found in the AppliedPredictiveModeling package. This section demonstrates how feature selection can be conducted in R for a subset of the analyses.

Forward, Backward, and Stepwise Selection

There are several R functions for this class of wrappers:

- `step` in the stats package can be used to search for appropriate subsets for linear regression and generalized linear models (from the `lm` and `glm` functions, respectively). The `direction` argument controls the search method

(e.g. "both," "backward," or "forward"). A more general function is the `stepAIC` function in the MASS package, which can handle additional model types. In either case, the AIC statistic (or its variants) is used as the objective function.

- The `fastbw` function in the `rms` package conducts similar searches but has the optional but unrecommended choice of using p-values as the objective function.
- The `regsubsets` function in the leaps package has similar functionality.
- The klaR package contains the `stepclass` function than searches the predictor space for models that maximize cross-validated accuracy rates. The function has built-in methods for several models, such as `lda`, but can be more broadly generalized.

The caret package function `train` has wrappers for `leaps`, `stepAIC`, and `stepclass`, so that the entire feature selection process can be resampled and the risk of selection bias is reduced.

For example, to use `stepAIC` with logistic regression, the function takes an initial model as input. To illustrate the function, a small model is used:

```
> initial <- glm(Class ~ tau + VEGF + E4 + IL_3, data = training,
+                family = binomial)
> library(MASS)
> stepAIC(initial, direction = "both")
  Start:  AIC=189.46
  Class ~ tau + VEGF + E4 + IL_3

         Df Deviance    AIC
  - IL_3  1   179.69 187.69
  - E4    1   179.72 187.72
  <none>      179.46 189.46
  - VEGF  1   242.77 250.77
  - tau   1   288.61 296.61

  Step:  AIC=187.69
  Class ~ tau + VEGF + E4

         Df Deviance    AIC
  - E4    1   179.84 185.84
  <none>      179.69 187.69
  + IL_3  1   179.46 189.46
  - VEGF  1   248.30 254.30
  - tau   1   290.05 296.05

  Step:  AIC=185.84
  Class ~ tau + VEGF

         Df Deviance    AIC
  <none>      179.84 185.84
  + E4    1   179.69 187.69
  + IL_3  1   179.72 187.72
  - VEGF  1   255.07 259.07
```

```
- tau   1   300.69 304.69

Call:  glm(formula = Class ~ tau + VEGF, family = binomial, data = training)

Coefficients:
(Intercept)          tau         VEGF
     9.8075      -4.2779       0.9761

Degrees of Freedom: 266 Total (i.e. Null);  264 Residual
Null Deviance:            313.3
Residual Deviance: 179.8         AIC: 185.8
```

The function returns a `glm` object with the final predictor set. The other functions listed above use similar strategies.

Recursive Feature Elimination

The caret and varSelRF packages contain functions for recursive feature elimination. While the `varSelRF` function in the varSelRF is specific to random forests, the `rfe` function in caret is a general framework for any predictive model. For the latter, there are predefined functions for random forests, linear discriminant analysis, bagged trees, naïve Bayes, generalized linear models, linear regression models, and logistic regression. The random forest functions are in a list called `rfFuncs`:

```
> library(caret)
> ## The built-in random forest functions are in rfFuncs.
> str(rfFuncs)
List of 6
 $ summary   :function (data, lev = NULL, model = NULL)
 $ fit       :function (x, y, first, last, ...)
 $ pred      :function (object, x)
 $ rank      :function (object, x, y)
 $ selectSize:function (x, metric, maximize)
 $ selectVar :function (y, size)
```

Each of these function defines a step in Algorithm 19.2:

- The `summary` function defines how the predictions will be evaluated (Line 10 in Algorithm 19.2).
- The `fit` function allows the user to specify the model and conduct parameter tuning (Lines 19.2, 6, and 12).
- The `pred` function generates predictions for new samples.
- The `rank` function generates variable importance measures (Line 2).
- The `selectSize` function chooses the appropriate predictor subset size (Line 11).
- The `selectVar` function picks which variables are used in the final model.

These options can be changed. For example, to compute the expanded set of performance measures shown above,

```
> newRF <- rfFuncs
> newRF$summary <- fiveStats
```

To run the RFE procedure for random forests, the syntax is

```
> ## The control function is similar to trainControl():
> ctrl <- rfeControl(method = "repeatedcv",
+                    repeats = 5,
+                    verbose = TRUE,
+                    functions = newRF,
+                    index = index)


> set.seed(721)
> rfRFE <- rfe(x = training[, predVars],
+              y = training$Class,
+              sizes = varSeq,
+              metric = "ROC",
+              rfeControl = ctrl,
+              ## now pass options to randomForest()
+              ntree = 1000)
> rfRFE


  Recursive feature selection

  Outer resampling method: Cross-Validation (10-fold, repeated 5 times)

  Resampling performance over subset size:

   Variables    ROC   Sens   Spec Accuracy  Kappa Selected
           1 0.8051 0.5375 0.8806   0.7869 0.4316
           3 0.8661 0.6407 0.9167   0.8415 0.5801
           5 0.8854 0.6736 0.9365   0.8645 0.6386
           7 0.8980 0.6571 0.9414   0.8637 0.6300        *
           9 0.8978 0.6850 0.9506   0.8779 0.6679
          11 0.8886 0.6750 0.9609   0.8825 0.6756
          13 0.8895 0.6604 0.9609   0.8786 0.6636
          15 0.8950 0.6586 0.9629   0.8794 0.6628
          17 0.8867 0.6554 0.9621   0.8780 0.6576
          19 0.8900 0.6418 0.9642   0.8758 0.6514
           :      :      :      :        :      :       :
         129 0.8923 0.4439 0.9826   0.8351 0.4947
         131 0.8918 0.4439 0.9836   0.8360 0.4976
         132 0.8895 0.4439 0.9815   0.8345 0.4963

  The top 5 variables (out of 7):
     Ab_42, tau, p_tau, VEGF, FAS
```

(This output has been truncated, and the columns for the standard deviations were removed to fit on the page.)

The process for predicting new samples is straightforward:

```
> predict(rfRFE, head(testing))
        pred Impaired Control
  2   Control    0.291   0.709
  6  Impaired    0.695   0.305
  15  Control    0.189   0.811
  16 Impaired    0.794   0.206
  33  Control    0.302   0.698
  38 Impaired    0.930   0.070
```

The built-in functions predict the classes and probabilities for classification.

There are also built-in functions to do recursive feature selection for models that require retuning at each iteration. For example, to fit support vector machines:

```
> svmFuncs <- caretFuncs
> svmFuncs$summary <- fivestats

> ctrl <- rfeControl(method = "repeatedcv",
+                    repeats = 5,
+                    verbose = TRUE,
+                    functions = svmFuncs,
+                    index = index)

> set.seed(721)
> svmRFE <- rfe(x = training[, predVars],
+               y = training$Class,
+               sizes = varSeq,
+               metric = "ROC",
+               rfeControl = ctrl,
+               ## Now options to train()
+               method = "svmRadial",
+               tuneLength = 12,
+               preProc = c("center", "scale"),
+               ## Below specifies the inner resampling process
+               trControl = trainControl(method = "cv",
+                                        verboseIter = FALSE,
+                                        classProbs = TRUE))
> svmRFE

  Recursive feature selection

  Outer resampling method: Cross-Validation (10-fold, repeated 5 times)

  Resampling performance over subset size:

   Variables    ROC     Sens   Spec Accuracy     Kappa Selected
           1 0.6043 0.000000 0.9959   0.7237 -0.005400
           3 0.6071 0.005714 0.9858   0.7178 -0.010508
           5 0.5737 0.000000 0.9979   0.7252 -0.002718
           7 0.5912 0.005357 0.9969   0.7259  0.002849
           9 0.5899 0.000000 0.9979   0.7252 -0.002799
          11 0.6104 0.000000 0.9959   0.7237 -0.005625
          13 0.5858 0.000000 0.9979   0.7252 -0.002829
```

```
          :     :       :       :         :         :       :
        121 0.8172 0.513571 0.9241    0.8116   0.473426
        123 0.8210 0.514286 0.9199    0.8087   0.469536
        125 0.8844 0.608571 0.9559    0.8610   0.613538
        127 0.8914 0.671786 0.9548    0.8775   0.666157       *
        129 0.8877 0.647500 0.9445    0.8632   0.629154
        131 0.8891 0.644643 0.9487    0.8655   0.631925
        132 0.8879 0.647143 0.9455    0.8639   0.630313

  The top 5 variables (out of 127):
     Ab_42, tau, p_tau, MMP10, MIF
```

Here we can see that the poor performance is related to the class imbalance; the model is biased towards high specificity since most samples are controls.

The caret web page[5] contains more details and examples related to `rfe`.

Filter Methods

caret has a function called `sbf` (for Selection By Filter) that can be used to screen predictors for models and to estimate performance using resampling. Any function can be written to screen the predictors.

For example, to compute a p-value for each predictor, depending on the data type, the following approach could be used:

```
> pScore <- function(x, y)
+   {
+     numX <- length(unique(x))
+     if(numX > 2)
+       {
+         ## With many values in x, compute a t-test
+         out <- t.test(x ~ y)$p.value
+       } else {
+         ## For binary predictors, test the odds ratio == 1 via
+         ## Fisher's Exact Test
+         out <- fisher.test(factor(x), y)$p.value
+       }
+     out
+   }

> ## Apply the scores to each of the predictor columns
> scores <- apply(X = training[, predVars],
+                 MARGIN = 2,
+                 FUN = pScore,
+                 y = training$Class)
> tail(scores)
        p_tau        Ab_42         male           E4           E3
1.699064e-07 8.952405e-13 1.535628e-02 6.396309e-04 1.978571e-01
           E2
1.774673e-01
```

[5] http://caret.r-forge.r-project.org/.

A function can also be designed to apply a p-value correction, such as the Bonferroni procedure:

```
> pCorrection <- function (score, x, y)
+ {
+   ## The options x and y are required by the caret package
+   ## but are not used here
+   score <- p.adjust(score,  "bonferroni")
+   ## Return a logical vector to decide which predictors
+   ## to retain after the filter
+   keepers <- (score <= 0.05)
+   keepers
+ }
> tail(pCorrection(scores))
  p_tau Ab_42  male    E4    E3    E2
   TRUE  TRUE FALSE FALSE FALSE FALSE
```

As before, caret contains a number of built-in functions for filter methods: linear regression, random forests, bagged trees, linear discriminant analysis, and naïve Bayes (see `?rfSBF` for more details). For example, `ldaSBF` has the following functions:

```
> str(ldaSBF)
  List of 5
   $ summary:function (data, lev = NULL, model = NULL)
   $ fit    :function (x, y, ...)
   $ pred   :function (object, x)
   $ score  :function (x, y)
   $ filter :function (score, x, y)
```

These functions are similar to those shown for `rfe`. The `score` function computes some quantitative measure of importance (e.g., the p-values produced by the previous `pScore` function). The function `filter` takes these values (and the raw training set data) and determines which predictors pass the filter.

For the biomarker data, the filtered LDA model was fit using

```
> ldaWithPvalues <- ldaSBF
> ldaWithPvalues$score <- pScore
> ldaWithPvalues$summary <- fiveStats
> ldaWithPvalues$filter <- pCorrection
> sbfCtrl <- sbfControl(method = "repeatedcv",
+                       repeats = 5,
+                       verbose = TRUE,
+                       functions = ldaWithPvalues,
+                       index = index)
> ldaFilter <- sbf(training[, predVars],
+                  training$Class,
+                  tol = 1.0e-12,
+                  sbfControl = sbfCtrl)


> ldaFilter
```

```
Selection By Filter

Outer resampling method: Cross-Validation (10-fold, repeated 5 times)

Resampling performance:

    ROC   Sens   Spec Accuracy  Kappa   ROCSD SensSD  SpecSD AccuracySD
 0.9168 0.7439 0.9136    0.867 0.6588 0.06458 0.1778 0.05973     0.0567
 KappaSD
  0.1512

Using the training set, 47 variables were selected:
   Alpha_1_Antitrypsin, Apolipoprotein_D, B_Lymphocyte_Chemoattractant_BL,
   Complement_3, Cortisol...

During resampling, the top 5 selected variables (out of a possible 66):
   Ab_42 (100%), Cortisol (100%), Creatine_Kinase_MB (100%),
   Cystatin_C (100%), E4 (100%)

On average, 46.1 variables were selected (min = 38, max = 57)
```

Again, the caret package web site has additional detail regarding the `rfe` and `sbf` functions, including features not shown here.

Exercises

19.1. For the biomarker data, determine if the between-predictor correlations shown in Fig. 19.4 have an effect on the feature selection process. Specifically:

(a) Create an initial filter of the predictors that removes predictors to minimize the amount of multicollinearity in the data prior to modeling.
(b) Refit the recursive feature selection models.
(c) Did the RFE profiles shown in Fig. 19.4 change considerably? Which models would have been mostly likely affected by multicollinearity?

19.2. Use the same resampling process to evaluate a penalized LDA model. How does performance compare? Is the same variable selection pattern observed in both models?

19.3. Apply different combinations of filters and predictive models to the biomarker data. Are the same predictors retained across different filters? Do models react differently to the same predictor set?

19.4. Recall Exercise 7.2 where a simulation tool from Friedman (1991) utilized a nonlinear function of predictors:

$$y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + e$$

where the predictors x_1 through x_5 have uniform distributions and the error e is normally distributed with zero mean and a standard deviation that can be specified:

(a) Simulate a training and test set with $n = 500$ samples per data set. Plot the predictors against the outcome in the training set using scatter plots, correlation plots, table plots, and other visualizations.
(b) Use forward, backward, and stepwise algorithms for feature selection. Did the final models select the full set of predictors? Why or why not? If cross-validation was used with these search tools, was the performance similar to the test set?
(c) Use recursive feature selection with several different models? How did these methods perform? Were the informative predictors selected?
(d) Apply filter methods to the data using filters where each predictor is evaluated separately and others are evaluated simultaneously (e.g., the ReliefF algorithm). Were the two interacting predictors (x_1 and x_2) selected? Was one favored more than the other?
(e) Reduce the sample size of the training set and add a larger number of non-informative predictors. How do these search procedures perform under more extreme circumstances?

19.5. For the cell segmentation data:

(a) Use filter and wrapper methods to determine an optimal set of predictors.
(b) For linear discriminant analysis and logistic regression, use the alternative versions of these models with built-in feature selection (e.g., the glmnet and sparse LDA). How do the different approaches compare in terms of performance, the number of predictors required, and training time?