Chapter 2
A Short Tour of the Predictive Modeling Process

Before diving in to the formal components of model building, we present a simple example that illustrates the broad concepts of model building. Specifically, the following example demonstrates the concepts of data "spending," building candidate models, and selecting the optimal model.

2.1 Case Study: Predicting Fuel Economy

The `fueleconomy.gov` web site, run by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy and the U.S. Environmental Protection Agency, lists different estimates of fuel economy for passenger cars and trucks. For each vehicle, various characteristics are recorded such as the engine displacement or number of cylinders. Along with these values, laboratory measurements are made for the city and highway miles per gallon (MPG) of the car.

In practice, we would build a model on as many vehicle characteristics as possible in order to find the most predictive model. However, this introductory illustration will focus high-level concepts of model building by using a single predictor, engine displacement (the volume inside the engine cylinders), and a single response, unadjusted highway MPG for 2010–2011 model year cars.

The first step in any model building process is to understand the data, which can most easily be done through a graph. Since we have just one predictor and one response, these data can be visualized with a scatter plot (Fig. 2.1). This figure shows the relationship between engine displacement and fuel economy. The "2010 model year" panel contains all the 2010 data while the other panel shows the data only for new 2011 vehicles. Clearly, as engine displacement increases, the fuel efficiency drops regardless of year. The relationship is somewhat linear but does exhibit some curvature towards the extreme ends of the displacement axis.

M. Kuhn and K. Johnson, *Applied Predictive Modeling*,
DOI 10.1007/978-1-4614-6849-3_2,

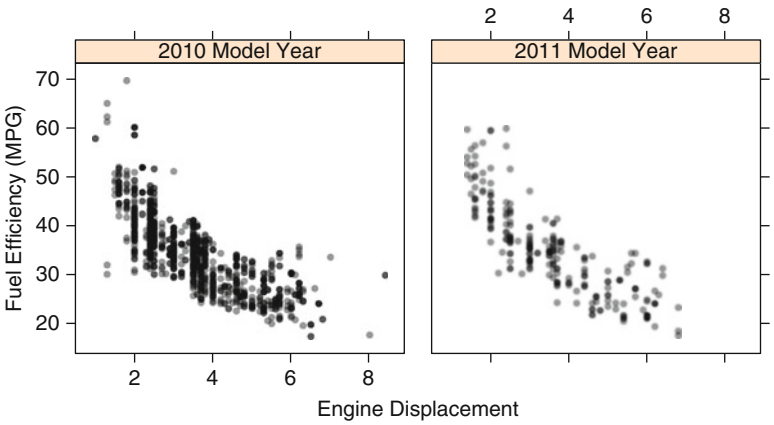

Fig. 2.1: The relationship between engine displacement and fuel efficiency of all 2010 model year vehicles and new 2011 car lines

If we had more than one predictor, we would need to further understand characteristics of the predictors and the relationships among the predictors. These characteristics may suggest important and necessary pre-processing steps that must be taken prior to building a model (Chap. 3).

After first understanding the data, the next step is to build and evaluate a model on the data. A standard approach is to take a random sample of the data for model building and use the rest to understand model performance. However, suppose we want to predict the MPG for a *new* car line. In this situation, models can be created using the 2010 data (containing 1,107 vehicles) and tested on the 245 new 2011 cars. The common terminology would be that the 2010 data are used as the model "training set" and the 2011 values are the "test" or "validation" set.

Now that we have defined the data used for model building and evaluation, we should decide how to measure performance of the model. For regression problems where we try to predict a numeric value, the residuals are important sources of information. Residuals are computed as the observed value minus the predicted value (i.e., $y_i - \widehat{y}_i$). When predicting numeric values, the root mean squared error (RMSE) is commonly used to evaluate models. Described in more detail in Chap. 7, RMSE is interpreted as how far, on average, the residuals are from zero.

At this point, the modeler will try various techniques to mathematically define the relationship between the predictor and outcome. To do this, the training set is used to estimate the various values needed by the model equations. The test set will be used only when a few strong candidate models have been finalized (repeatedly using the test set in the model build process negates its utility as a final arbitrator of the models).

Suppose a linear regression model was created where the predicted MPG is a basic slope and intercept model. Using the training data, we estimate the

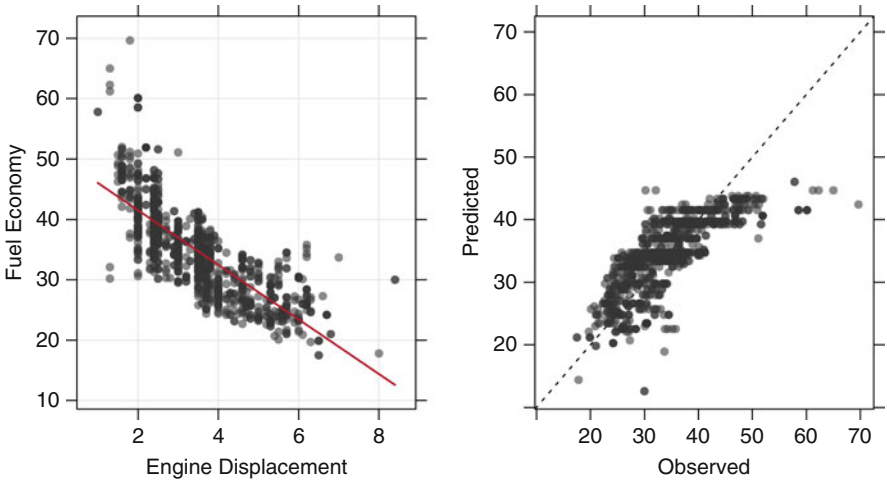

Fig. 2.2: Quality of fit diagnostics for the linear regression model. The training set data and its associated predictions are used to understand how well the model works

intercept to be 50.6 and the slope to be −4.5 MPG/liters using the method of least squares (Sect. 6.2). The model fit is shown in Fig. 2.2 for the training set data.[1] The left-hand panel shows the training set data with a linear model fit defined by the estimated slope and intercept. The right-hand panel plots the observed and predicted MPG. These plots demonstrate that this model misses some of the patterns in the data, such as under-predicting fuel efficiency when the displacement is less than 2 L or above 6 L.

When working with the training set, one must be careful not to simply evaluate model performance using the same data used to build the model. If we simply re-predict the training set data, there is the potential to produce overly optimistic estimates of how well the model works, especially if the model is highly adaptable. An alternative approach for quantifying how well the model operates is to use *resampling*, where different subversions of the training data set are used to fit the model. Resampling techniques are discussed in Chap. 4. For these data, we used a form of resampling called 10-fold cross-validation to estimate the model RMSE to be 4.6 MPG.

Looking at Fig. 2.2, it is conceivable that the problem might be solved by introducing some nonlinearity in the model. There are many ways to do this. The most basic approach is to supplement the previous linear regression model with additional complexity. Adding a squared term for engine displacement would mean estimating an additional slope parameter associated with the square of the predictor. In doing this, the model equation changes to

[1] One of our graduate professors once said "the only way to be comfortable with your data is to never look at it."

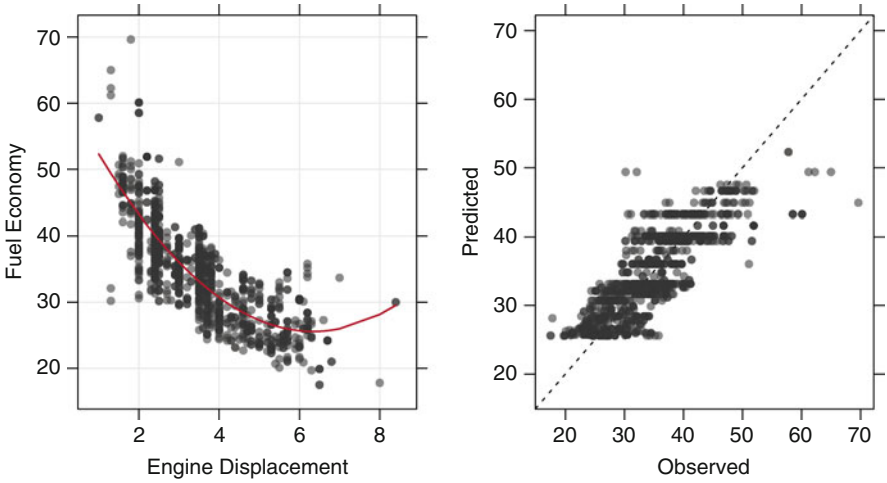

Fig. 2.3: Quality of fit diagnostics for the quadratic regression model (using the training set)

$$\text{efficiency} = 63.2 - 11.9 \times \text{displacement} + 0.94 \times \text{displacement}^2$$

This is referred to as a *quadratic model* since it includes a squared term; the model fit is shown in Fig. 2.3. Unquestionably, the addition of the quadratic term improves the model fit. The RMSE is now estimated to be 4.2 MPG using cross-validation. One issue with quadratic models is that they can perform poorly on the extremes of the predictor. In Fig. 2.3, there may be a hint of this for the vehicles with very high displacement values. The model appears to be bending upwards unrealistically. Predicting new vehicles with large displacement values may produce significantly inaccurate results.

Chapters 6–8 discuss many other techniques for creating sophisticated relationships between the predictors and outcome. One such approach is the multivariate adaptive regression spline (MARS) model (Friedman 1991). When used with a single predictor, MARS can fit separate linear regression lines for different ranges of engine displacement. The slopes and intercepts are estimated for this model, as well as the number and size of the separate regions for the linear models. Unlike the linear regression models, this technique has a *tuning parameter* which cannot be directly estimated from the data. There is no analytical equation that can be used to determine how many segments should be used to model the data. While the MARS model has internal algorithms for making this determination, the user can try different values and use resampling to determine the appropriate value. Once the value is found, a final MARS model would be fit using all the training set data and used for prediction.

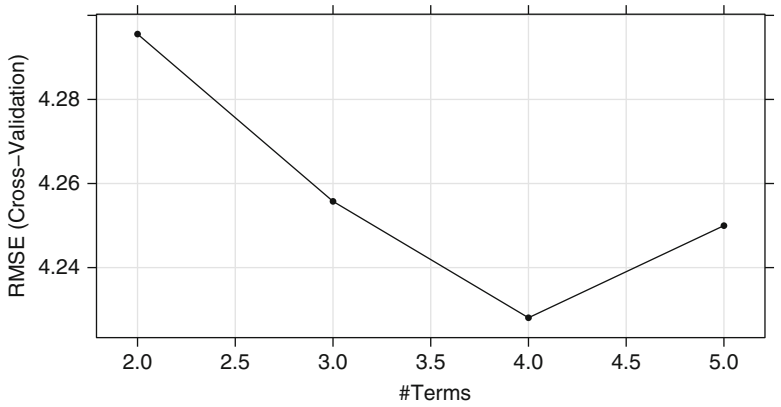

Fig. 2.4: The cross-validation profile for the MARS tuning parameter

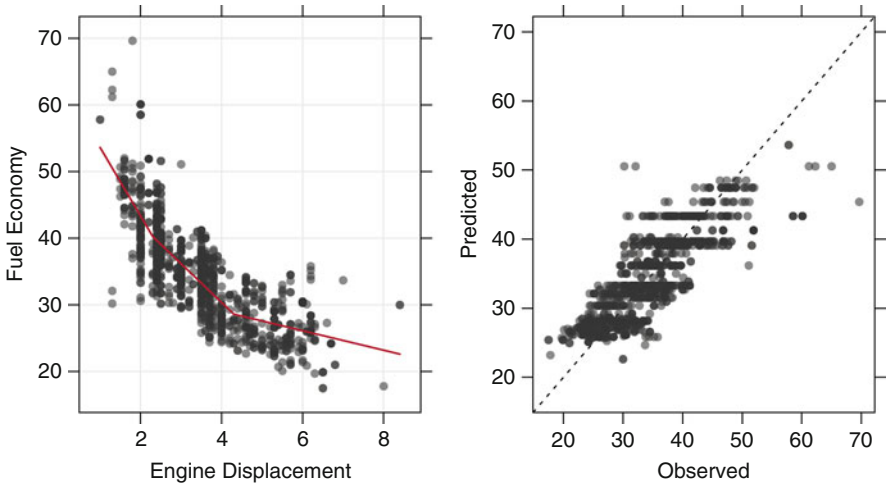

Fig. 2.5: Quality of fit diagnostics for the MARS model (using the training set). The MARS model creates several linear regression fits with change points at 2.3, 3.5, and 4.3 L

For a single predictor, MARS can allow for up to five model terms (similar to the previous slopes and intercepts). Using cross-validation, we evaluated four candidate values for this tuning parameter to create the resampling profile which is shown in Fig. 2.4. The lowest RMSE value is associated with four terms, although the scale of change in the RMSE values indicates that there is some insensitivity to this tuning parameter. The RMSE associated with the optimal model was 4.2 MPG. After fitting the final MARS model with four terms, the training set fit is shown in Fig. 2.5 where several linear segments were predicted.

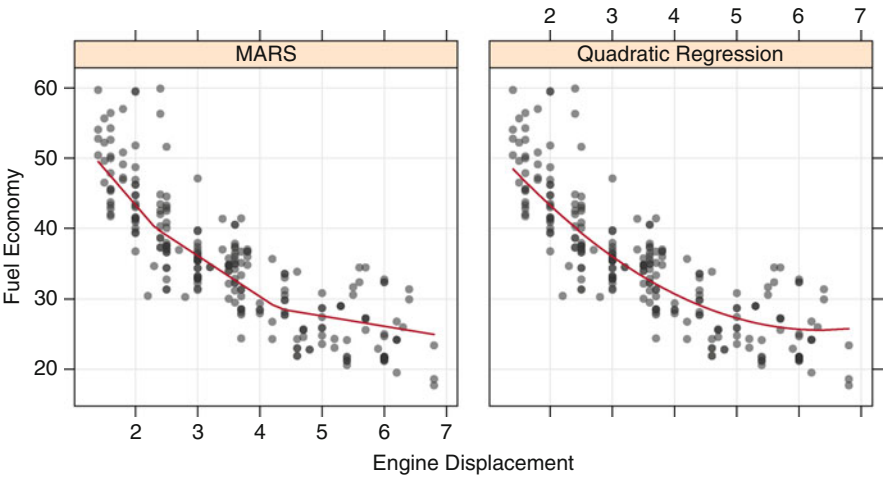

Fig. 2.6: The test set data and with model fits for two models

Based on these three models, the quadratic regression and MARS models were evaluated on the test set. Figure 2.6 shows these results. Both models fit very similarly. The test set RMSE values for the quadratic model was 4.72 MPG and the MARS model was 4.69 MPG. Based on this, either model would be appropriate for the prediction of new car lines.

2.2 Themes

There are several aspects of the model building process that are worth discussing further, especially for those who are new to predictive modeling.

Data Splitting

Although discussed in the next chapter, how we allocate data to certain tasks (e.g., model building, evaluating performance) is an important aspect of modeling. For this example, the primary interest is to predict the fuel economy of *new* vehicles, which is not the same population as the data used to build the model. This means that, to some degree, we are testing how well the model *extrapolates* to a different population. If we were interested in predicting from the same population of vehicles (i.e., *interpolation*), taking a simple random sample of the data would be more appropriate. How the training and test sets are determined should reflect how the model will be applied.

How much data should be allocated to the training and test sets? It generally depends on the situation. If the pool of data is small, the data splitting decisions can be critical. A small test would have limited utility as a judge of performance. In this case, a sole reliance on resampling techniques (i.e., no test set) might be more effective. Large data sets reduce the criticality of these decisions.

Predictor Data

This example has revolved around one of many predictors: the engine displacement. The original data contain many other factors, such as the number of cylinders, the type of transmission, and the manufacturer. An earnest attempt to predict the fuel economy would examine as many predictors as possible to improve performance. Using more predictors, it is likely that the RMSE for the new model cars can be driven down further. Some investigation into the data can also help. For example, none of the models were effective at predicting fuel economy when the engine displacement was small. Inclusion of predictors that target these types of vehicles would help improve performance.

An aspect of modeling that was not discussed here was feature selection: the process of determining the minimum set of relevant predictors needed by the model. This common task is discussed in Chap. 19.

Estimating Performance

Before using the test set, two techniques were employed to determine the effectiveness of the model. First, quantitative assessments of statistics (i.e., the RMSE) using resampling help the user understand how each technique would perform on new data. The other tool was to create simple visualizations of a model, such as plotting the observed and predicted values, to discover areas of the data where the model does particularly good or bad. This type of qualitative information is critical for improving models and is lost when the model is gauged only on summary statistics.

Evaluating Several Models

For these data, three different models were evaluated. It is our experience that some modeling practitioners have a favorite model that is relied on indiscriminately. The "No Free Lunch" Theorem (Wolpert 1996) argues that,

without having substantive information about the modeling problem, there is no single model that will always do better than any other model. Because of this, a strong case can be made to try a wide variety of techniques, then determine which model to focus on. In our example, a simple plot of the data shows that there is a nonlinear relationship between the outcome and the predictor. Given this knowledge, we might exclude linear models from consideration, but there is still a wide variety of techniques to evaluate. One might say that "model X is always the best performing model" but, for these data, a simple quadratic model is extremely competitive.

Model Selection

At some point in the process, a specific model must be chosen. This example demonstrated two types of model selection. First, we chose some models over others: the linear regression model did not fit well and was dropped. In this case, we chose *between models*. There was also a second type of model selection shown. For MARS, the tuning parameter was chosen using cross-validation. This was also model selection where we decided on the *type of MARS model* to use. In this case, we did the selection *within* different MARS models.

In either case, we relied on cross-validation and the test set to produce quantitative assessments of the models to help us make the choice. Because we focused on a single predictor, which will not often be the case, we also made visualizations of the model fit to help inform us. At the end of the process, the MARS and quadratic models appear to give equivalent performance. However, knowing that the quadratic model might not do well for vehicles with very large displacements, our intuition might tell us to favor the MARS model. One goal of this book is to help the user gain intuition regarding the strengths and weakness of different models to make informed decisions.

2.3 Summary

At face value, model building appears straightforward: pick a modeling technique, plug in data, and generate a prediction. While this approach will generate a predictive model, it will most likely *not* generate a reliable, trustworthy model for predicting new samples. To get this type of model, we must first understand the data *and* the objective of the modeling. Upon understanding the data and objectives, we then pre-process and split the data. Only after these steps do we finally proceed to building, evaluating, and selecting models.

Chapter 3
Data Pre-processing

Data pre-processing techniques generally refer to the addition, deletion, or transformation of training set data. Although this text is primarily concerned with modeling techniques, data preparation can make or break a model's predictive ability. Different models have different sensitivities to the type of predictors in the model; *how* the predictors enter the model is also important. Transformations of the data to reduce the impact of data skewness or outliers can lead to significant improvements in performance. Feature extraction, discussed in Sect. 3.3, is one empirical technique for creating surrogate variables that are combinations of multiple predictors. Additionally, simpler strategies such as removing predictors based on their lack of information content can also be effective.

The need for data pre-processing is determined by the type of model being used. Some procedures, such as tree-based models, are notably insensitive to the characteristics of the predictor data. Others, like linear regression, are not. In this chapter, a wide array of *possible* methodologies are discussed. For modeling techniques described in subsequent chapters, we will also discuss which, if any, pre-processing techniques can be useful.

This chapter outlines approaches to *unsupervised* data processing: the outcome variable is not considered by the pre-processing techniques. In other chapters, *supervised* methods, where the outcome is utilized to pre-process the data, are also discussed. For example, partial least squares (PLS) models are essentially supervised versions of principal component analysis (PCA). We also describe strategies for removing predictors without considering how those variables might be related to the outcome. Chapter 19 discusses techniques for finding subsets of predictors that optimize the ability of the model to predict the response.

How the predictors are encoded, called *feature engineering*, can have a significant impact on model performance. For example, using combinations of predictors can sometimes be more effective than using the individual values: the ratio of two predictors may be more effective than using two independent

M. Kuhn and K. Johnson, *Applied Predictive Modeling*,
DOI 10.1007/978-1-4614-6849-3_3,

predictors. Often the most effective encoding of the data is informed by the modeler's understanding of the problem and thus is not derived from any mathematical technique.

There are usually several different methods for encoding predictor data. For example, in Chaps. 12 through 15, an illustrative data set is described for predicting the success of academic grants. One piece of information in the data is the submission date of the grant. This date can be represented in myriad ways:

- The number of days since a reference date
- Isolating the month, year, and day of the week as separate predictors
- The numeric day of the year (ignoring the calendar year)
- Whether the date was within the school year (as opposed to holiday or summer sessions)

The "correct" feature engineering depends on several factors. First, some encodings may be optimal for some models and poor for others. For example, tree-based models will partition the data into two or more bins. Theoretically, if the month were important, the tree would split the numeric day of the year accordingly. Also, in some models, multiple encodings of the same data may cause problems. As will be illustrated several times in later chapters, some models contain built-in *feature selection*, meaning that the model will only include predictors that help maximize accuracy. In these cases, the model can pick and choose which representation of the data is best.

The relationship between the predictor and the outcome is a second factor. For example, if there were a seasonal component to these data, and it appears that there is, then the numeric day of the year would be best. Also, if some months showed higher success rates than others, then the encoding based on the month is preferable.

As with many questions of statistics, the answer to "which feature engineering methods are the best?" is that *it depends*. Specifically, it depends on the model being used and the true relationship with the outcome. A broad discussion regarding how the data were encoded for our analyses is given in Sect. 12.1.

Prior to delving into specific techniques, an illustrative data set that is used throughout the chapter is introduced.

3.1 Case Study: Cell Segmentation in High-Content Screening

Medical researchers often seek to understand the effects of medicines or diseases on the size, shape, development status, and number of cells in a living organism or plant. To do this, experts can examine the target serum or tissue under a microscope and manually assess the desired cell characteristics.

This work is tedious and requires expert knowledge of the cell type and characteristics.

Another way to measure the cell characteristics from these kinds of samples is by using high-content screening (Giuliano et al. 1997). Briefly, a sample is first dyed with a substance that will bind to the desired characteristic of the cells. For example, if a researcher wants to quantify the size or shape of cell nuclei, then a stain can be applied to the sample that attaches to the cells' DNA. The cells can be fixed in a substance that preserves the nature state of the cell. The sample is then interrogated by an instrument (such as a confocal microscope) where the dye deflects light and the detectors quantify the degree of scattering for that specific wavelength. If multiple characteristics of the cells are desired, then multiple dyes and multiple light frequencies can be used simultaneously. The light scattering measurements are then processed through imaging software to quantify the desired cell characteristics.

Using an automated, high-throughput approach to assess samples' cell characteristics can sometimes produce misleading results. Hill et al. (2007) describe a research project that used high-content screening to measure several aspects of cells. They observed that the imaging software used to determine the location and shape of the cell had difficulty segmenting cells (i.e., defining cells' boundaries). Consider Fig. 3.1, which depicts several example cells from this study. In these images, the bright green boundaries identify the cell nucleus, while the blue boundaries define the cell perimeter. Clearly some cells are *well segmented*, while others are not. Cells that are poorly segmented appear to be damaged, when in reality they are not. If cell size, shape, and/or quantity are the endpoints of interest in a study, then it is important that the instrument and imaging software can correctly segment cells.

For this research, Hill et al. (2007) assembled a data set consisting of 2,019 cells. Of these cells, 1,300 were judged to be poorly segmented (PS) and 719 were well segmented (WS); 1,009 cells were reserved for the training set.[1]

For a particular type of cell, the researchers used different stains that would be visible to different optical channels. Channel one was associated with the cell body and can be used to determine the cell perimeter, area, and other qualities. Channel two interrogated the cell nucleus by staining the nuclear DNA (shown in blue shading in Fig. 3.1). Channels three and four were stained to detect actin and tubulin, respectively. These are two types of filaments that transverse the cells in scaffolds and are part of the cell's cytoskeleton. For all cells, 116 features (e.g., cell area, spot fiber count) were measured and were used to predict the segmentation quality of cells.[2]

[1] The individual data points can be found on the journal web site or in the R AppliedPredictiveModeling package. See the Computing section at the end of this chapter.

[2] The original authors included several "status" features that are binary representations of other features in the data set. We excluded these from the analysis in this chapter.

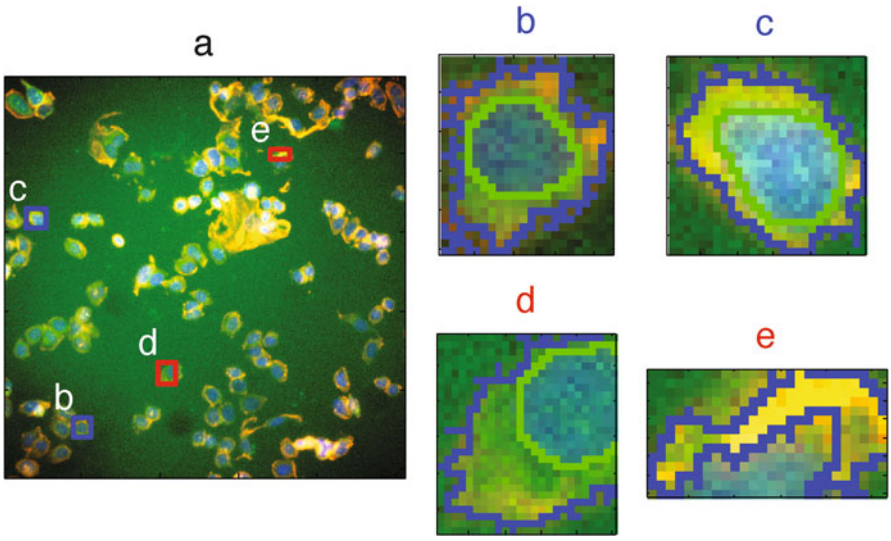

Fig. 3.1: An image showing cell segmentation from Hill et al. (2007). The *red boxes* [panels (**d**) and (**e**)] show poorly segmented cells while the cells in the *blue boxes* are examples of proper segmentation

This chapter will use the training set samples identified by the original authors to demonstrate data pre-processing techniques.

3.2 Data Transformations for Individual Predictors

Transformations of predictor variables may be needed for several reasons. Some modeling techniques may have strict requirements, such as the predictors having a common scale. In other cases, creating a good model may be difficult due to specific characteristics of the data (e.g., outliers). Here we discuss centering, scaling, and skewness transformations.

Centering and Scaling

The most straightforward and common data transformation is to center scale the predictor variables. To center a predictor variable, the average predictor value is subtracted from all the values. As a result of centering, the predictor has a zero mean. Similarly, to scale the data, each value of the predictor variable is divided by its standard deviation. Scaling the data coerce the values to have a common standard deviation of one. These manipulations are

generally used to improve the numerical stability of some calculations. Some models, such as PLS (Sects. 6.3 and 12.4), benefit from the predictors being on a common scale. The only real downside to these transformations is a loss of interpretability of the individual values since the data are no longer in the original units.

Transformations to Resolve Skewness

Another common reason for transformations is to remove distributional skewness. An un-skewed distribution is one that is roughly symmetric. This means that the probability of falling on either side of the distribution's mean is roughly equal. A right-skewed distribution has a large number of points on the left side of the distribution (smaller values) than on the right side (larger values). For example, the cell segmentation data contain a predictor that measures the standard deviation of the intensity of the pixels in the actin filaments. In the natural units, the data exhibit a strong right skewness; there is a greater concentration of data points at relatively small values and small number of large values. Figure 3.2 shows a histogram of the data in the natural units (left panel).

A general rule of thumb to consider is that skewed data whose ratio of the highest value to the lowest value is greater than 20 have significant skewness. Also, the skewness statistic can be used as a diagnostic. If the predictor distribution is roughly symmetric, the skewness values will be close to zero. As the distribution becomes more right skewed, the skewness statistic becomes larger. Similarly, as the distribution becomes more left skewed, the value becomes negative. The formula for the sample skewness statistic is

$$\text{skewness} = \frac{\sum (x_i - \overline{x})^3}{(n-1)v^{3/2}}$$
$$\text{where} \quad v = \frac{\sum (x_i - \overline{x})^2}{(n-1)},$$

where x is the predictor variable, n is the number of values, and $\overline{x}$ is the sample mean of the predictor. For the actin filament data shown in Fig. 3.2, the skewness statistic was calculated to be 2.39 while the ratio to the largest and smallest value was 870.

Replacing the data with the log, square root, or inverse may help to remove the skew. For the data in Fig. 3.2, the right panel shows the distribution of the data once a log transformation has been applied. After the transformation, the distribution is not entirely symmetric but these data are better behaved than when they were in the natural units.

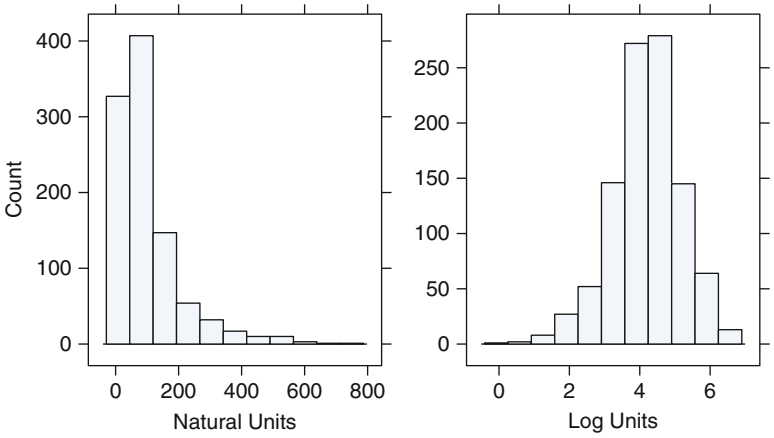

Fig. 3.2: *Left*: a histogram of the standard deviation of the intensity of the pixels in actin filaments. This predictor has a strong right skewness with a concentration of points with low values. For this variable, the ratio of the smallest to largest value is 870 and a skewness value of 2.39. *Right*: the same data after a log transformation. The skewness value for the logged data was −0.4

Alternatively, statistical methods can be used to empirically identify an appropriate transformation. Box and Cox (1964) propose a *family* of transformations[3] that are indexed by a parameter, denoted as λ:

$$x^* = \begin{cases} \frac{x^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log(x) & \text{if } \lambda = 0 \end{cases}$$

In addition to the log transformation, this family can identify square transformation ($\lambda = 2$), square root ($\lambda = 0.5$), inverse ($\lambda = -1$), and others in-between. Using the training data, λ can be estimated. Box and Cox (1964) show how to use maximum likelihood estimation to determine the transformation parameter. This procedure would be applied independently to each predictor data that contain values greater than zero.

For the segmentation data, 69 predictors were not transformed due to zero or negative values and 3 predictors had λ estimates within 1 ± 0.02, so no transformation was applied. The remaining 44 predictors had values estimated between -2 and 2. For example, the predictor data shown in Fig. 3.2 have an estimated transformation value of 0.1, indicating the log

[3] Some readers familiar with Box and Cox (1964) will know that this transformation was developed for *outcome* data while Box and Tidwell (1962) describe similar methods for transforming a set of predictors in a linear model. Our experience is that the Box–Cox transformation is more straightforward, less prone to numerical issues, and just as effective for transforming individual predictor variables.

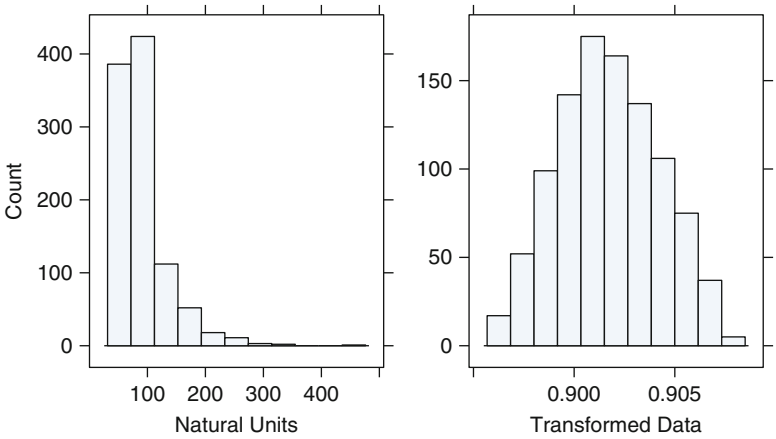

Fig. 3.3: *Left*: a histogram of the cell perimeter predictor. *Right*: the same data after a Box–Cox transformation with λ estimated to be -1.1

transformation is reasonable. Another predictor, the estimated cell perimeter, had a λ estimate of -1.1. For these data, the original and transformed values are shown in Fig. 3.3.

3.3 Data Transformations for Multiple Predictors

These transformations act on groups of predictors, typically the entire set under consideration. Of primary importance are methods to resolve outliers and reduce the dimension of the data.

Transformations to Resolve Outliers

We will generally define outliers as samples that are exceptionally far from the mainstream of the data. Under certain assumptions, there are formal statistical definitions of an outlier. Even with a thorough understanding of the data, outliers can be hard to define. However, we can often identify an unusual value by looking at a figure. When one or more samples are suspected to be outliers, the first step is to make sure that the values are scientifically valid (e.g., positive blood pressure) and that no data recording errors have occurred. Great care should be taken not to hastily remove or change values, especially if the sample size is small. With small sample sizes, apparent outliers might be a result of a skewed distribution where there are not yet

enough data to see the skewness. Also, the outlying data may be an indication of a special part of the population under study that is just starting to be sampled. Depending on how the data were collected, a "cluster" of valid points that reside outside the mainstream of the data might belong to a different population than the other samples.[4]

There are several predictive models that are resistant to outliers. Tree-based classification models create splits of the training data and the prediction equation is a set of logical statements such as "if predictor A is greater than X, predict the class to be Y," so the outlier does not usually have an exceptional influence on the model. Also, support vector machines for classification generally disregard a portion of the training set samples when creating a prediction equation. The excluded samples may be far away from the decision boundary and outside of the data mainstream.

If a model is considered to be sensitive to outliers, one data transformation that can minimize the problem is the *spatial sign* (Serneels et al. 2006). This procedure projects the predictor values onto a multidimensional sphere. This has the effect of making all the samples the same distance from the center of the sphere. Mathematically, each sample is divided by its squared norm:

$$x_{ij}^* = \frac{x_{ij}}{\sum_{j=1}^{P} x_{ij}^2}.$$

Since the denominator is intended to measure the squared distance to the center of the predictor's distribution, it is important to center and scale the predictor data prior to using this transformation. Note that, unlike centering or scaling, this manipulation of the predictors transforms them as a group. Removing predictor variables after applying the spatial sign transformation may be problematic.

Figure 3.4 shows another data set with two correlated predictors. In these data, at least eight samples cluster away from the majority of other data. These data points are likely a valid, but poorly sampled subpopulation of the data. The modeler would investigate why these points are different; perhaps they represent a group of interest, such as highly profitable customers. The spatial sign transformation is shown on the right-hand panel where all the data points are projected to be a common distance away from the origin. The outliers still reside in the Northwest section of the distribution but are contracted inwards. This mitigates the effect of the samples on model training.

[4] Section 20.5 discusses model *extrapolation*—where the model predicts samples outside of the mainstream of the training data. Another concept is the *applicability domain* of the model, which is the population of samples that can be effectively predicted by the model.

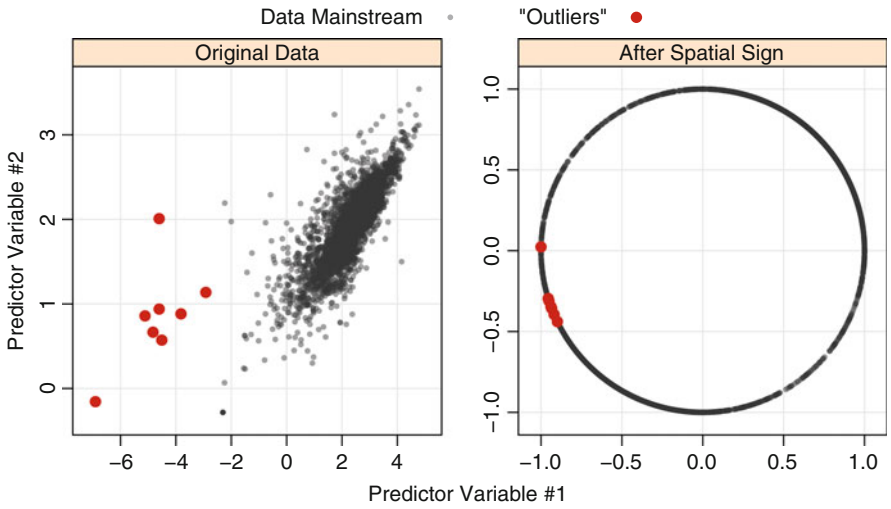

Fig. 3.4: *Left*: An illustrative example with a group of outlying data points. *Right*: When the original data are transformed, the results bring the outliers towards the majority of the data

Data Reduction and Feature Extraction

Data reduction techniques are another class of predictor transformations. These methods reduce the data by generating a smaller set of predictors that seek to capture a majority of the information in the original variables. In this way, fewer variables can be used that provide reasonable fidelity to the original data. For most data reduction techniques, the new predictors are functions of the original predictors; therefore, all the original predictors are still needed to create the surrogate variables. This class of methods is often called *signal extraction* or *feature extraction* techniques.

PCA is a commonly used data reduction technique (Abdi and Williams 2010). This method seeks to find linear combinations of the predictors, known as principal components (PCs), which capture the most possible variance. The first PC is defined as the linear combination of the predictors that captures the most variability of all possible linear combinations. Then, subsequent PCs are derived such that these linear combinations capture the most remaining variability while also being uncorrelated with all previous PCs. Mathematically, the jth PC can be written as:

$$\text{PC}_j = (a_{j1} \times \text{Predictor 1}) + (a_{j2} \times \text{Predictor 2}) + \cdots + (a_{jP} \times \text{Predictor } P).$$

P is the number of predictors. The coefficients $a_{j1}, a_{j2}, \ldots, a_{jP}$ are called component weights and help us understand which predictors are most important to each PC.

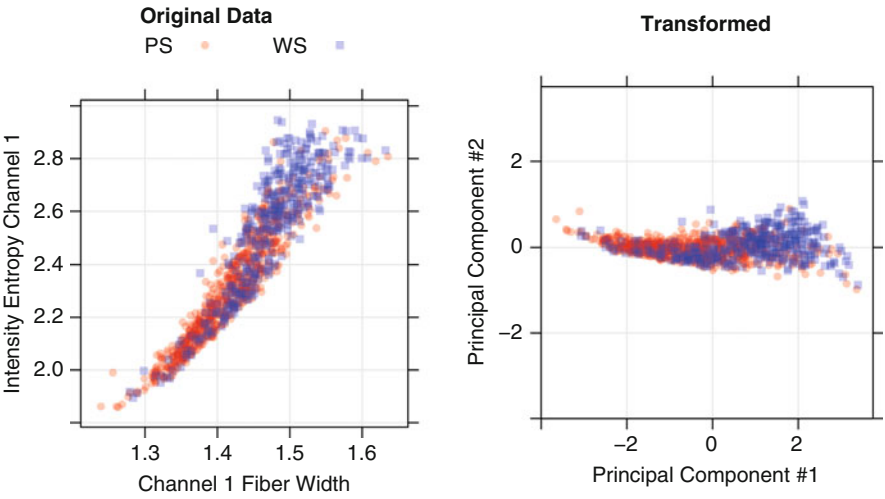

Fig. 3.5: An example of the principal component transformation for the cell segmentation data. The *shapes* and *colors* indicate which cells were poorly segmented or well segmented

To illustrate PCA, consider the data in Fig. 3.5. This set contains a subset of two correlated predictors, average pixel intensity of channel 1 and entropy of intensity values in the cell (a measure of cell shape), and a categorical response. Given the high correlation between the predictors (0.93), we could infer that average pixel intensity and entropy of intensity values measure redundant information about the cells and that either predictor or a linear combination of these predictors could be used in place of the original predictors. In this example, two PCs can be derived (right plot in Fig. 3.5); this transformation represents a rotation of the data about the axis of greatest variation. The first PC summarizes 97 % of the original variability, while the second summarizes 3 %. Hence, it is reasonable to use only the first PC for modeling since it accounts for the majority of information in the data.

The primary advantage of PCA, and the reason that it has retained its popularity as a data reduction method, is that it creates components that are uncorrelated. As mentioned earlier in this chapter, some predictive models prefer predictors to be uncorrelated (or at least low correlation) in order to find solutions and to improve the model's numerical stability. PCA preprocessing creates new predictors with desirable characteristics for these kinds of models.

While PCA delivers new predictors with desirable characteristics, it must be used with understanding and care. Notably, practitioners must understand that PCA seeks predictor-set variation without regard to any further understanding of the predictors (i.e., measurement scales or distributions)

or to knowledge of the modeling objectives (i.e., response variable). Hence, without proper guidance, PCA can generate components that summarize characteristics of the data that are irrelevant to the underlying structure of the data and also to the ultimate modeling objective.

Because PCA seeks linear combinations of predictors that maximize variability, it will naturally first be drawn to summarizing predictors that have more variation. If the original predictors are on measurement scales that differ in orders of magnitude [consider demographic predictors such as income level (in dollars) and height (in feet)], then the first few components will focus on summarizing the higher magnitude predictors (e.g., income), while latter components will summarize lower variance predictors (e.g., height). This means that the PC weights will be larger for the higher variability predictors on the first few components. In addition, it means that PCA will be focusing its efforts on identifying the data structure based on measurement scales rather than based on the important relationships within the data for the current problem.

For most data sets, predictors are on different scales. In addition, predictors may have skewed distributions. Hence, to help PCA avoid summarizing distributional differences and predictor scale information, it is best to first transform skewed predictors (Sect. 3.2) and then center and scale the predictors prior to performing PCA. Centering and scaling enables PCA to find the underlying relationships in the data without being influenced by the original measurement scales.

The second caveat of PCA is that it does not consider the modeling objective or response variable when summarizing variability. Because PCA is blind to the response, it is an *unsupervised technique*. If the predictive relationship between the predictors and response is not connected to the predictors' variability, then the derived PCs will not provide a suitable relationship with the response. In this case, a *supervised technique*, like PLS (Sects. 6.3 and 12.4), will derive components while simultaneously considering the corresponding response.

Once we have decided on the appropriate transformations of the predictor variables, we can then apply PCA. For data sets with many predictor variables, we must decide how many components to retain. A heuristic approach for determining the number of components to retain is to create a scree plot, which contains the ordered component number (x-axis) and the amount of summarized variability (y-axis) (Fig. 3.6). For most data sets, the first few PCs will summarize a majority of the variability, and the plot will show a steep descent; variation will then taper off for the remaining components. Generally, the component number prior to the tapering off of variation is the maximal component that is retained. In Fig. 3.6, the variation tapers off at component 5. Using this rule of thumb, four PCs would be retained. In an automated model building process, the optimal number of components can be determined by cross-validation (see Sect. 4.4).

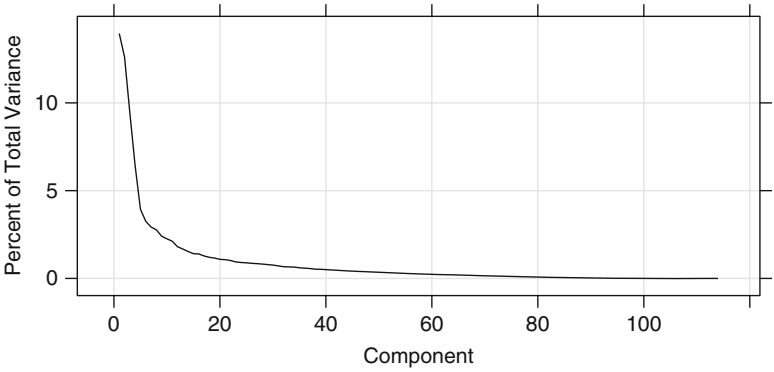

Fig. 3.6: A "scree plot" where the percentage of the total variance explained by each component is shown

Visually examining the principal components is a critical step for assessing data quality and gaining intuition for the problem. To do this, the first few principal components can be plotted against each other and the plot symbols can be colored by relevant characteristics, such as the class labels. If PCA has captured a sufficient amount of information in the data, this type of plot can demonstrate clusters of samples or outliers that may prompt a closer examination of the individual data points. For classification problems, the PCA plot can show potential separation of classes (if there is a separation). This can set the initial expectations of the modeler; if there is little clustering of the classes, the plot of the principal component values will show a significant overlap of the points for each class. Care should be taken when plotting the components; the scale of the components tend to become smaller as they account for less and less variation in the data. For example, in Fig. 3.5, the values of component one range from -3.7 to 3.4 while the component two ranges from -1 to 1.1. If the axes are displayed on separate scales, there is the potential to over-interpret any patterns that might be seen for components that account for small amounts of variation. See Geladi, Manley, and Lestander (2003) for other examples of this issue.

PCA was applied to the entire set of segmentation data predictors. As previously demonstrated, there are some predictors with significant skewness. Since skewed predictors can have an impact on PCA, there were 44 variables that were transformed using the Box–Cox procedure previously described. After the transformations, the predictors were centered and scaled prior to conducting PCA.

Figure 3.6 shows the percentage of the total variation in the data which was accounted for by each component. Notice that the percentages decrease as more components are added. The first three components accounted for 14 %, 12.6 %, and 9.4 % of the total variance, respectively. After four components,

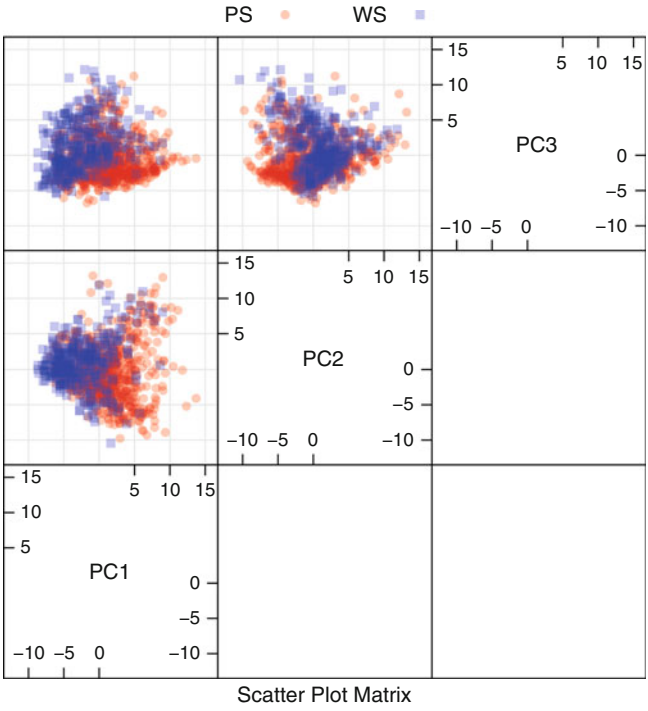

Fig. 3.7: A plot of the first three principal components for the cell segmentation data, colored by cell type

there is a sharp decline in the percentage of variation being explained, although these four components describe only 42.4 % of the information in the data set.

Figure 3.7 shows a scatter plot matrix for the first three principal components. The points are colored by class (segmentation quality). Since the percentages of variation explained are not large for the first three components, it is important not to over-interpret the resulting image. From this plot, there appears to be some separation between the classes when plotting the first and second components. However, the distribution of the well-segmented cells is roughly contained within the distribution of the poorly identified cells. One conclusion to infer from this image is that the cell types are not *easily* separated. However, this does not mean that other models, especially those which can accommodate highly nonlinear relationships, will reach the same conclusion. Also, while there are some cells in the data that are not completely within the data mainstream, there are no blatant outliers.

Another exploratory use of PCA is characterizing which predictors are associated with each component. Recall that each component is a linear combination of the predictors and the coefficient for each predictor is called the

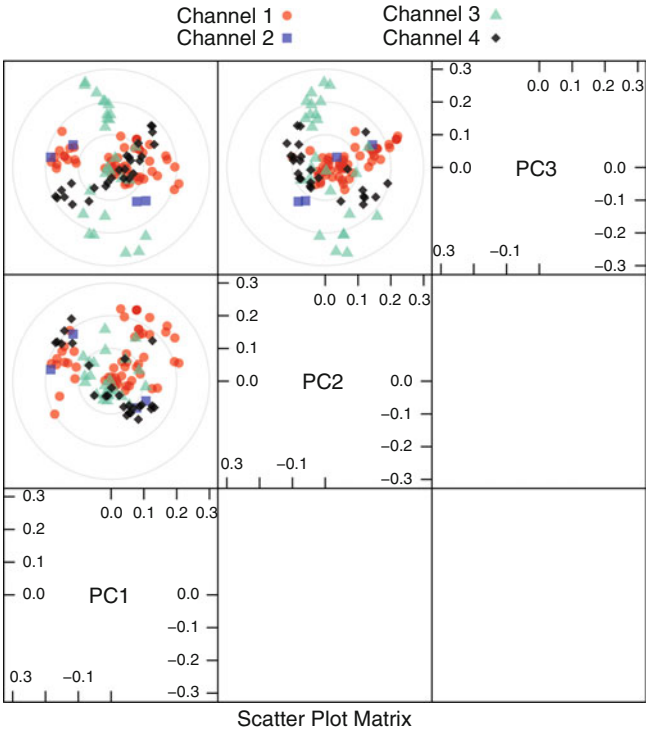

Fig. 3.8: A plot of the loadings of the first three principal components for the cell segmentation data, colored by optical channel. Recall that channel one was associated with the cell body, channel two with the cell nucleus, channel three with actin, and channel four with tubulin

loading. Loadings close to zero indicate that the predictor variable did not contribute much to that component. Figure 3.8 shows the loadings for the first three components in the cell segmentation data. Each point corresponds to a predictor variable and is colored by the optical channel used in the experiment. For the first principal component, the loadings for the first channel (associated with the cell body) are on the extremes. This indicates that cell body characteristics have the largest effect on the first principal component and by extension the predictor values. Also note that the majority of the loadings for the third channel (measuring actin and tubulin) are closer to zero for the first component. Conversely, the third principal component is mostly associated with the third channel while the cell body channel plays a minor role here. Even though the cell body measurements account for more variation in the data, this does not imply that these variables will be associated with predicting the segmentation quality.

3.4 Dealing with Missing Values

In many cases, some predictors have no values for a given sample. These missing data could be *structurally missing*, such as the number of children a man has given birth to. In other cases, the value cannot or was not determined at the time of model building.

It is important to understand *why* the values are missing. First and foremost, it is important to know if the pattern of missing data is related to the outcome. This is called "informative missingness" since the missing data pattern is instructional on its own. Informative missingness can induce significant bias in the model. In the introductory chapter, a short example was given regarding predicting a patient's response to a drug. Suppose the drug was extremely ineffective or had significant side effects. The patient may be likely to miss doctor visits or to drop out of the study. In this case, there clearly is a relationship between the probability of missing values and the treatment. Customer ratings can often have informative missingness; people are more compelled to rate products when they have strong opinions (good or bad). In this case, the data are more likely to be polarized by having few values in the middle of the rating scale. In the Netflix Prize machine learning competition to predict which movies people will like based on their previous ratings, the "Napoleon Dynamite Effect" confounded many of the contestants because people who did rate the movie *Napoleon Dynamite* either loved or hated it.

Missing data should not be confused with *censored* data where the exact value is missing but something is known about its value. For example, a company that rents movie disks by mail may use the duration that a customer has kept a movie in their models. If a customer has not yet returned a movie, we do not know the actual time span, only that it is as least as long as the current duration. Censored data can also be common when using laboratory measurements. Some assays cannot measure below their limit of detection. In such cases, we know that the value is smaller than the limit but was not precisely measured.

Are censored data treated differently than missing data? When building traditional statistical models focused on interpretation or inference, the censoring is usually taken into account in a formal manner by making assumptions about the censoring mechanism. For predictive models, it is more common to treat these data as simple missing data or use the censored value as the observed value. For example, when a sample has a value below the limit of detection, the actual limit can be used in place of the real value. For this situation, it is also common to use a random number between zero and the limit of detection.

In our experience, missing values are more often related to predictor variables than the sample. Because of this, amount of missing data may be concentrated in a subset of predictors rather than occurring randomly across all the predictors. In some cases, the percentage of missing data is substantial enough to remove this predictor from subsequent modeling activities.

There are cases where the missing values might be concentrated in specific samples. For large data sets, removal of samples based on missing values is not a problem, assuming that the missingness is not informative. In smaller data sets, there is a steep price in removing samples; some of the alternative approaches described below may be more appropriate.

If we do not remove the missing data, there are two general approaches. First, a few predictive models, especially tree-based techniques, can specifically account for missing data. These are discussed further in Chap. 8.

Alternatively, missing data can be imputed. In this case, we can use information in the training set predictors to, in essence, estimate the values of other predictors. This amounts to a predictive model within a predictive model.

Imputation has been extensively studied in the statistical literature, but in the context of generating correct hypothesis testing procedures in the presence of missing data. This is a separate problem; for predictive models we are concerned about accuracy of the predictions rather than making valid inferences. There is a small literature on imputation for predictive models. Saar-Tsechansky and Provost (2007b) examine the issue of missing values and delve into how specific models deal with the issue. Jerez et al. (2010) also look at a wide variety of imputation methods for a specific data set.

As previously mentioned, imputation is just another layer of modeling where we try to estimate values of the predictor variables based on other predictor variables. The most relevant scheme for accomplishing this is to use the training set to built an imputation model for each predictor in the data set. Prior to model training or the prediction of new samples, missing values are filled in using imputation. Note that this extra layer of models adds uncertainty. If we are using resampling to select tuning parameter values or to estimate performance, the imputation should be incorporated within the resampling. This will increase the computational time for building models, but it will also provide honest estimates of model performance.

If the number of predictors affected by missing values is small, an exploratory analysis of the relationships between the predictors is a good idea. For example, visualizations or methods like PCA can be used to determine if there are strong relationships between the predictors. If a variable with missing values is highly correlated with another predictor that has few missing values, a focused model can often be effective for imputation (see the example below).

One popular technique for imputation is a K-nearest neighbor model. A new sample is imputed by finding the samples in the training set "closest" to it and averages these nearby points to fill in the value. Troyanskaya et al. (2001) examine this approach for high-dimensional data with small sample sizes. One advantage of this approach is that the imputed data are confined to be within the range of the training set values. One disadvantage is that the entire training set is required every time a missing value needs to be imputed. Also, the number of neighbors is a tuning parameter, as is the method for de-

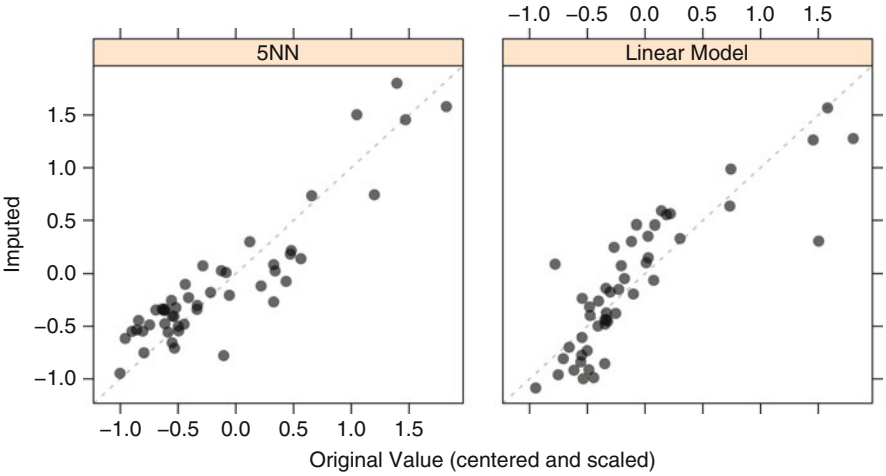

Fig. 3.9: After simulating 50 missing test set values at random for the cell perimeter data, two different imputation models were built with the training set and applied to the missing test set values. This plot shows the centered and scaled values before and after imputation

termining "closeness" of two points. However, Troyanskaya et al. (2001) found the nearest neighbor approach to be fairly robust to the tuning parameters, as well as the amount of missing data.

In Sect. 3.2, a predictor that measures the cell perimeter was used to illustrate skewness (see Fig. 3.3). As an illustration, a 5-nearest neighbor model was created using the training set values. In the test set, missing values were randomly induced in 50 test set cell perimeter values and then imputed using the model. Figure 3.9 shows a scatter plot of the samples set to missing. The left-hand panel shows the results of the 5-nearest neighbor approach. This imputation model does a good job predicting the absent samples; the correlation between the real and imputed values is 0.91.

Alternatively, a simpler approach can be used to impute the cell perimeter. The cell fiber length, another predictor associated with cell size, has a very high correlation (0.99) with the cell perimeter data. We can create a simple linear regression model using these data to predict the missing values. These results are in the right-hand panel of Fig. 3.9. For this approach, the correlation between the real and imputed values is 0.85.

3.5 Removing Predictors

There are potential advantages to removing predictors prior to modeling. First, fewer predictors means decreased computational time and complexity. Second, if two predictors are highly correlated, this implies that they are

measuring the same underlying information. Removing one should not compromise the performance of the model and might lead to a more parsimonious and interpretable model. Third, some models can be crippled by predictors with degenerate distributions. In these cases, there can be a significant improvement in model performance and/or stability without the problematic variables.

Consider a predictor variable that has a single unique value; we refer to this type of data as a zero variance predictor. For some models, such an uninformative variable may have little effect on the calculations. A tree-based model (Sects. 8.1 and 14.1) is impervious to this type of predictor since it would never be used in a split. However, a model such as linear regression would find these data problematic and is likely to cause an error in the computations. In either case, these data have no information and can easily be discarded. Similarly, some predictors might have only a handful of unique values that occur with very low frequencies. These "near-zero variance predictors" may have a single value for the vast majority of the samples.

Consider a text mining application where keyword counts are collected for a large set of documents. After filtering out commonly used "stop words," such as *the* and *of*, predictor variables can be created for interesting keywords. Suppose a keyword occurs in a small group of documents but is otherwise unused. A hypothetical distribution of such a word count distribution is given in Table 3.1. Of the 531 documents that were searched, there were only four unique counts. The majority of the documents (523) do not have the keyword; while six documents have two occurrences, one document has three and another has six occurrences. Since 98 % of the data have values of zero, a minority of documents might have an undue influence on the model. Also, if any resampling is used (Sect. 4.4), there is a strong possibility that one of the resampled data sets (Sect. 4.4) will only contain documents without the keyword, so this predictor would only have one unique value.

How can the user diagnose this mode of problematic data? First, the number of unique points in the data must be small relative to the number of samples. In the document example, there were 531 documents in the data set, but only four unique values, so the percentage of unique values is 0.8 %. A small percentage of unique values is, in itself, not a cause for concern as many "dummy variables" (Sect. 3.6 below) generated from categorical predictors would fit this description. The problem occurs when the frequency of these unique values is severely disproportionate. The ratio of the most common frequency to the second most common reflects the imbalance in the frequencies. Most of the documents in the data set ($n = 523$) do not have the keyword. After this, the most frequent case is documents with two occurrences ($n = 6$). The ratio of these frequencies, $523/6 = 87$, is rather high and is indicative of a strong imbalance.

Given this, a rule of thumb for detecting near-zero variance predictors is:

Table 3.1: A predictor describing the number of documents where a keyword occurred

	#Documents
Occurrences: 0	523
Occurrences: 2	6
Occurrences: 3	1
Occurrences: 6	1

- The fraction of unique values over the sample size is low (say 10 %).
- The ratio of the frequency of the most prevalent value to the frequency of the second most prevalent value is large (say around 20).

If both of these criteria are true and the model in question is susceptible to this type of predictor, it may be advantageous to remove the variable from the model.

Between-Predictor Correlations

Collinearity is the technical term for the situation where a pair of predictor variables have a substantial correlation with each other. It is also possible to have relationships between multiple predictors at once (called *multicollinearity*).

For example, the cell segmentation data have a number of predictors that reflect the size of the cell. There are measurements of the cell perimeter, width, and length as well as other, more complex calculations. There are also features that measure cell morphology (i.e., shape), such as the roughness of the cell.

Figure 3.10 shows a correlation matrix of the training set. Each pairwise correlation is computed from the training data and colored according to its magnitude. This visualization is symmetric: the top and bottom diagonals show identical information. Dark blue colors indicate strong positive correlations, dark red is used for strong negative correlations, and white implies no empirical relationship between the predictors. In this figure, the predictor variables have been grouped using a clustering technique (Everitt et al. 2011) so that collinear groups of predictors are adjacent to one another. Looking along the diagonal, there are blocks of strong positive correlations that indicate "clusters" of collinearity. Near the center of the diagonal is a large block of predictors from the first channel. These predictors are related to cell size, such as the width and length of the cell.

When the data set consists of too many predictors to examine visually, techniques such as PCA can be used to characterize the magnitude of the problem. For example, if the first principal component accounts for a large

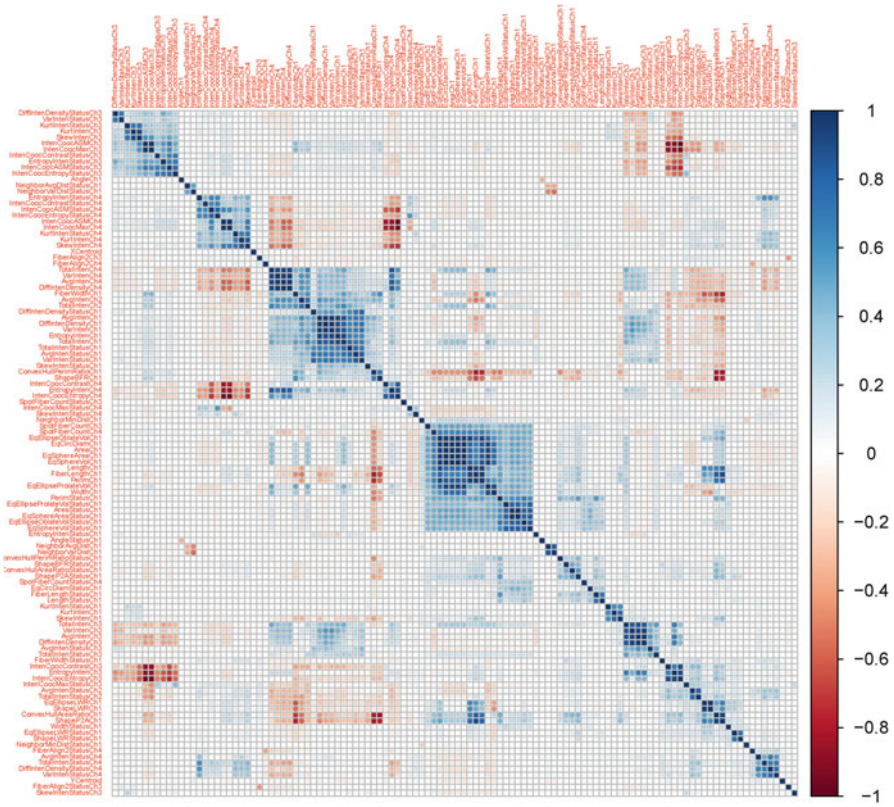

Fig. 3.10: A visualization of the cell segmentation correlation matrix. The order of the variables is based on a clustering algorithm

percentage of the variance, this implies that there is at least one group of predictors that represent the same information. For example, Fig. 3.6 indicates that the first 3–4 components have relative contributions to the total variance. This would indicate that there are at least 3–4 significant relationships between the predictors. The PCA loadings can be used to understand which predictors are associated with each component to tease out this relationships.

In general, there are good reasons to avoid data with highly correlated predictors. First, redundant predictors frequently add more complexity to the model than information they provide to the model. In situations where obtaining the predictor data is costly (either in time or money), fewer variables is obviously better. While this argument is mostly philosophical, there are mathematical disadvantages to having correlated predictor data. Using highly correlated predictors in techniques like linear regression can result in highly unstable models, numerical errors, and degraded predictive performance.

Classical regression analysis has several tools to diagnose multicollinearity for linear regression. Since collinear predictors can impact the variance of parameter estimates in this model, a statistic called the variance inflation factor (VIF) can be used to identify predictors that are impacted (Myers 1994). Beyond linear regression, this method may be inadequate for several reasons: it was developed for linear models, it requires more samples than predictor variables, and, while it does identify collinear predictors, it does not determine which should be removed to resolve the problem.

A less theoretical, more heuristic approach to dealing with this issue is to remove the minimum number of predictors to ensure that all pairwise correlations are below a certain threshold. While this method only identify collinearities in two dimensions, it can have a significantly positive effect on the performance of some models.

The algorithm is as follows:

1. Calculate the correlation matrix of the predictors.
2. Determine the two predictors associated with the largest absolute pairwise correlation (call them predictors A and B).
3. Determine the average correlation between A and the other variables. Do the same for predictor B.
4. If A has a larger average correlation, remove it; otherwise, remove predictor B.
5. Repeat Steps 2–4 until no absolute correlations are above the threshold.

The idea is to first remove the predictors that have the most correlated relationships.

Suppose we wanted to use a model that is particularly sensitive to between-predictor correlations, we might apply a threshold of 0.75. This means that we want to eliminate the minimum number of predictors to achieve all pairwise correlations less than 0.75. For the segmentation data, this algorithm would suggest removing 43 predictors.

As previously mentioned, feature extraction methods (e.g., principal components) are another technique for mitigating the effect of strong correlations between predictors. However, these techniques make the connection between the predictors and the outcome more complex. Additionally, since signal extraction methods are usually unsupervised, there is no guarantee that the resulting surrogate predictors have any relationship with the outcome.

3.6 Adding Predictors

When a predictor is categorical, such as gender or race, it is common to decompose the predictor into a set of more specific variables. For example, the credit scoring data discussed in Sect. 4.5 contains a predictor based on how much money was in the applicant's savings account. These data were

Table 3.2: A categorical predictor with five distinct groups from the credit scoring case study. The values are the amount in the savings account (in Deutsche Marks)

		Dummy variables				
Value	n	<100	100–500	500–1,000	>1,000	Unknown
<100 DM	103	1	0	0	0	0
100–500 DM	603	0	1	0	0	0
500–1,000 DM	48	0	0	1	0	0
>1,000 DM	63	0	0	0	1	0
Unknown	183	0	0	0	0	1

encoded into several groups, including a group for "unknown." Table 3.2 shows the values of this predictor and the number of applicants falling into each bin.

To use these data in models, the categories are re-encoded into smaller bits of information called "dummy variables." Usually, each category get its own dummy variable that is a zero/one indicator for that group. Table 3.2 shows the possible dummy variables for these data. Only four dummy variables are needed here; once you know the value of four of the dummy variables, the fifth can be inferred. However, the decision to include all of the dummy variables can depend on the choice of the model. Models that include an intercept term, such as simple linear regression (Sect. 6.2), would have numerical issues if each dummy variable was included in the model. The reason is that, for each sample, these variables all add up to one and this would provide the same information as the intercept. If the model is insensitive to this type of issue, using the complete set of dummy variables would help improve interpretation of the model.

Many of the models described in this text automatically generate highly complex, nonlinear relationships between the predictors and the outcome. More simplistic models do not unless the user manually specifies which predictors should be nonlinear and in what way. For example, logistic regression is a well-known classification model that, by default, generates linear classification boundaries. Figure 3.11 shows another illustrative example with two predictors and two classes. The left-hand panel shows the basic logistic regression classification boundaries when the predictors are added in the usual (linear) manner. The right-hand panel shows a logistic model with the basic linear terms and an additional term with the square of predictor B. Since logistic regression is a well-characterized and stable model, using this model with some additional nonlinear terms may be preferable to highly complex techniques (which may overfit).

Additionally, Forina et al. (2009) demonstrate one technique for augmenting the prediction data with addition of complex combinations of the data.

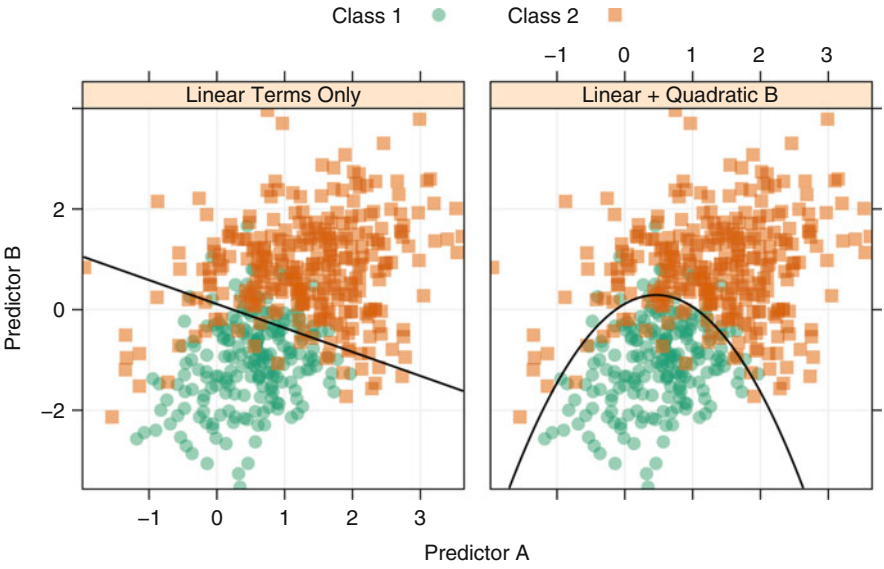

Fig. 3.11: Classification boundaries from two logistic regression models. The *left panel* has linear terms for the two predictors while the *right panel* has an additional quadratic term for predictor B. This model is discussed in more detail in Chap. 12

For classification models, they calculate the "class centroids," which are the centers of the predictor data for each class. Then for each predictor, the distance to each class centroid can be calculated and these distances can be added to the model.

3.7 Binning Predictors

While there are recommended techniques for pre-processing data, there are also methods to *avoid*. One common approach to simplifying a data set is to take a numeric predictor and pre-categorize or "bin" it into two or more groups prior to data analysis. For example, Bone et al. (1992) define a set of clinical symptoms to diagnose Systemic Inflammatory Response Syndrome (SIRS). SIRS can occur after a person is subjected to some sort of physical trauma (e.g., car crash). A simplified version of the clinical criteria for SIRS are:

- Temperature less than 36 °C or greater than 38 °C.
- Heart rate greater than 90 beats per minute.
- Respiratory rate greater than 20 breaths per minute.

- White blood cell count less than 4,000 cells/mm^3 or greater than 12,000 cells/mm^3.

A person who shows two or more of these criteria would be diagnosed as having SIRS.

The perceived advantages to this approach are:

- The ability to make seemingly simple statements, either for sake of having a simple decision rule (as in the SIRS example) or the belief that there will be a simple interpretation of the model.
- The modeler does not have to know the exact relationship between the predictors and the outcome.
- A higher response rate for survey questions where the choices are binned. For example, asking the date of a person's last tetanus shot is likely to have fewer responses than asking for a range (e.g., in the last 2 years, in the last 4 years).

There are many issues with the manual binning of continuous data. First, there can be a significant loss of performance in the model. Many of the modeling techniques discussed in this text are very good at determining complex relationships between the predictors and outcomes. Manually binning the predictors limits this potential. Second, there is a loss of precision in the predictions when the predictors are categorized. For example, if there are two binned predictors, only four combinations exist in the data set, so only simple predictions can be made. Third, research has shown (Austin and Brunner 2004) that categorizing predictors can lead to a high rate of false positives (i.e., noise predictors determined to be informative).

Unfortunately, the predictive models that are most powerful are usually the least interpretable. The bottom line is that the perceived improvement in interpretability gained by manual categorization is usually offset by a significant loss in performance. Since this book is concerned with predictive models (where interpretation is not the primary goal), loss of performance should be avoided. In fact, in some cases it may be unethical to arbitrarily categorize predictors. For example, there is a great deal of research on predicting aspects of disease (e.g., response to treatment, screening patients). If a medical diagnostic is used for such important determinations, patients desire the most accurate prediction possible. As long as complex models are properly validated, it may be improper to use a model that is built for interpretation rather than predictive performance.

Note that the argument here is related to the *manual* categorization of predictors prior to model building. There are several models, such as classification/regression trees and multivariate adaptive regression splines, that estimate cut points in the process of model building. The difference between these methodologies and manual binning is that the models use all the predictors to derive bins based on a single objective (such as maximizing accuracy). They evaluate many variables simultaneously and are usually based on statistically sound methodologies.

Chapter 4
Over-Fitting and Model Tuning

Many modern classification and regression models are highly adaptable; they are capable of modeling complex relationships. However, they can very easily overemphasize patterns that are not reproducible. Without a methodological approach to evaluating models, the modeler will not know about the problem until the next set of samples are predicted.

Over-fitting has been discussed in the fields of forecasting (Clark 2004), medical research (Simon et al. 2003; Steyerberg 2010), chemometrics (Gowen et al. 2010; Hawkins 2004; Defernez and Kemsley 1997), meteorology (Hsieh and Tang 1998), finance (Dwyer 2005), and marital research (Heyman and Slep 2001) to name a few. These references illustrate that over-fitting is a concern for any predictive model regardless of field of research. The aim of this chapter is to explain and illustrate key principles of laying a foundation onto which trustworthy models can be built and subsequently used for prediction. More specifically, we will describe strategies that enable us to have confidence that the model we build will predict new samples with a similar degree of accuracy on the set of data for which the model was evaluated. Without this confidence, the model's predictions are *useless*.

On a practical note, all model building efforts are constrained by the existing data. For many problems, the data may have a limited number of samples, may be of less-than-desirable quality, and/or may be unrepresentative of future samples. While there are ways to build predictive models on small data sets, which we will describe in this chapter, we will assume that data quality is sufficient and that it is representative of the entire sample population.

Working under these assumptions, we must use the data at hand to find the best predictive model. Almost all predictive modeling techniques have tuning parameters that enable the model to flex to find the structure in the data. Hence, we must use the existing data to identify settings for the model's parameters that yield the best and most realistic predictive performance (known as model tuning). Traditionally, this has been achieved by splitting the existing data into training and test sets. The training set is used to build and tune the model and the test set is used to estimate the model's

M. Kuhn and K. Johnson, *Applied Predictive Modeling*,
DOI 10.1007/978-1-4614-6849-3_4,

predictive performance. Modern approaches to model building split the data into multiple training and testing sets, which have been shown to often find more optimal tuning parameters and give a more accurate representation of the model's predictive performance.

To begin this chapter we will illustrate the concept of over-fitting through an easily visualized example. To avoid over-fitting, we propose a general model building approach that encompasses model tuning and model evaluation with the ultimate goal of finding the reproducible structure in the data. This approach entails splitting existing data into distinct sets for the purposes of tuning model parameters and evaluating model performance. The choice of data splitting method depends on characteristics of the existing data such as its size and structure. In Sect. 4.4, we define and explain the most versatile data splitting techniques and explore the advantages and disadvantages of each. Finally, we end the chapter with a computing section that provides code for implementing the general model building strategy.

4.1 The Problem of Over-Fitting

There now exist many techniques that can learn the structure of a set of data so well that when the model is applied to the data on which the model was built, it correctly predicts every sample. In addition to learning the general patterns in the data, the model has also learned the characteristics of each sample's unique noise. This type of model is said to be over-fit and will usually have poor accuracy when predicting a new sample. To illustrate over-fitting and other concepts in this chapter, consider the simple classification example in Fig. 4.1 that has two predictor variables (i.e., independent variables). These data contain 208 samples that are designated either as "Class 1" or "Class 2." The classes are fairly balanced; there are 111 samples in the first class and 97 in the second. Furthermore, there is a significant overlap between the classes which is often the case for most applied modeling problems.

One objective for a data set such as this would be to develop a model to classify new samples. In this two-dimensional example, the classification models or rules can be represented by boundary lines. Figure 4.2 shows example class boundaries from two distinct classification models. The lines envelop the area where each model predicts the data to be the second class (blue squares). The left-hand panel ("Model #1") shows a boundary that is complex and attempts to encircle every possible data point. The pattern in this panel is not likely to generalize to new data. The right-hand panel shows an alternative model fit where the boundary is fairly smooth and does not overextend itself to correctly classify every data point in the training set.

To gauge how well the model is classifying samples, one might use the training set. In doing so, the estimated error rate for the model in the left-hand panel would be overly optimistic. Estimating the utility of a model

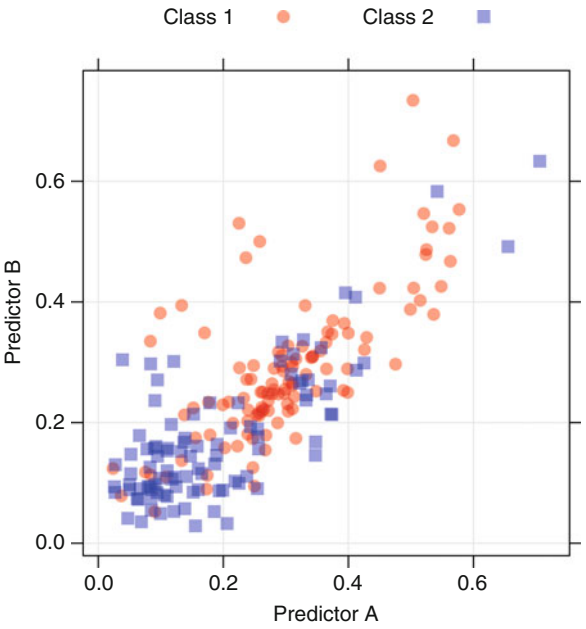

Fig. 4.1: An example of classification data that is used throughout the chapter

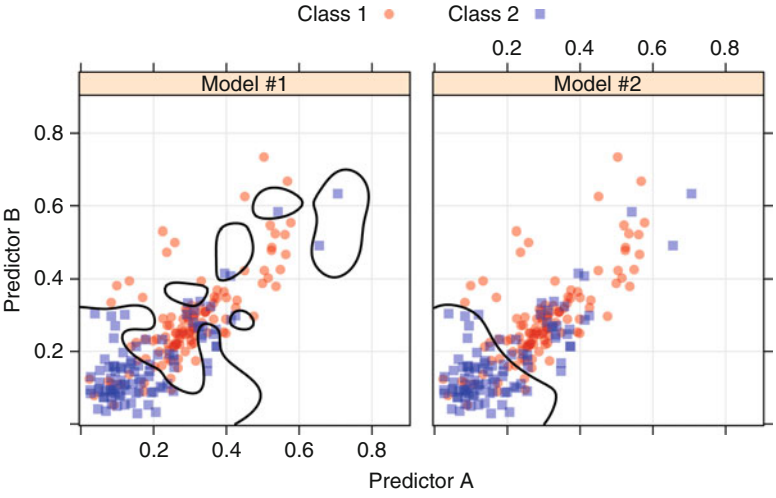

Fig. 4.2: An example of a training set with two classes and two predictors. The panels show two different classification models and their associated class boundaries

by re-predicting the training set is referred to *apparent performance* of the model (e.g., the apparent error rate). In two dimensions, it is not difficult to visualize that one model is over-fitting, but most modeling problems are in much higher dimensions. In these situations, it is very important to have a tool for characterizing how much a model is over-fitting the training data.

4.2 Model Tuning

Many models have important parameters which cannot be directly estimated from the data. For example, in the K-nearest neighbor classification model, a new sample is predicted based on the K-closest data points in the training set. An illustration of a 5-nearest neighbor model is shown in Fig. 4.3. Here, two new samples (denoted by the solid dot and filled triangle) are being predicted. One sample (•) is near a mixture of the two classes; three of the five neighbors indicate that the sample should be predicted as the first class. The other sample (▲) has all five points indicating the second class should be predicted. The question remains as to how many neighbors should be used. A choice of too few neighbors may over-fit the individual points of the training set while too many neighbors may not be sensitive enough to yield

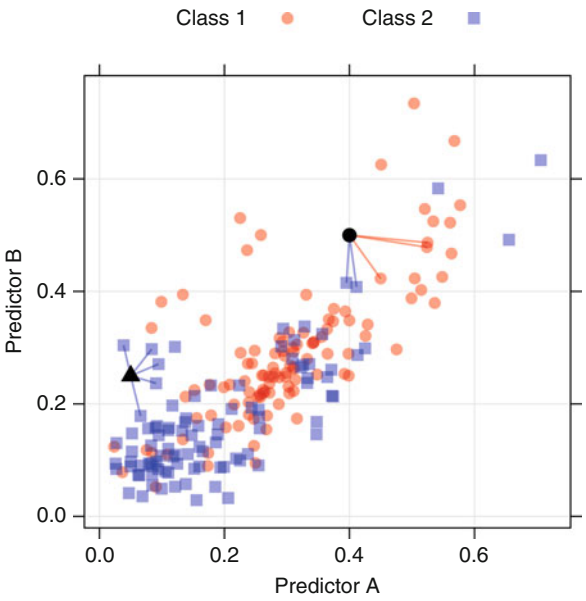

Fig. 4.3: The K-nearest neighbor classification model. Two new points, symbolized by *filled triangle* and *solid dot*, are predicted using the training set

reasonable performance. This type of model parameter is referred to as a *tuning parameter* because there is no analytical formula available to calculate an appropriate value.

Several models discussed in this text have at least one tuning parameter. Since many of these parameters control the complexity of the model, poor choices for the values can result in over-fitting. Figure 4.2 illustrates this point. A support vector machine (Sect. 13.4) was used to generate the class boundaries in each panel. One of the tuning parameters for this model sets the price for misclassified samples in the training set and is generally referred to as the "cost" parameter. When the cost is large, the model will go to great lengths to correctly label every point (as in the left panel) while smaller values produce models that are not as aggressive. The class boundary in the left panel was created by manually setting the cost parameter to a very high number. In the right panel, the cost value was determined using cross-validation (Sect. 4.4).

There are different approaches to searching for the best parameters. A general approach that can be applied to almost any model is to define a set of candidate values, generate reliable estimates of model utility across the candidates values, then choose the optimal settings. A flowchart of this process is shown in Fig. 4.4.

Once a candidate set of parameter values has been selected, then we must obtain trustworthy estimates of model performance. The performance on the hold-out samples is then aggregated into a performance profile which is then used to determine the final tuning parameters. We then build a final model with all of the training data using the selected tuning parameters. Using the K-nearest neighbor example to illustrate the procedure of Fig. 4.4, the candidate set might include all odd values of K between 1 and 9 (odd values are used in the two-class situation to avoid ties). The training data would then be resampled and evaluated many times for each tuning parameter value. These results would then be aggregated to find the optimal value of K.

The procedure defined in Fig. 4.4 uses a set of candidate models that are defined by the tuning parameters. Other approaches such as genetic algorithms (Mitchell 1998) or simplex search methods (Olsson and Nelson 1975) can also find optimal tuning parameters. These procedures algorithmically determine appropriate values for tuning parameters and iterate until they arrive at parameter settings with optimal performance. These techniques tend to evaluate a large number of candidate models and can be superior to a defined set of tuning parameters when model performance can be efficiently calculated. Cohen et al. (2005) provides a comparison of search routines for tuning a support vector machine model.

A more difficult problem is obtaining trustworthy estimates of model performance for these candidate models. As previously discussed, the apparent error rate can produce extremely optimistic performance estimates. A better approach is to test the model on samples that were not used for training.

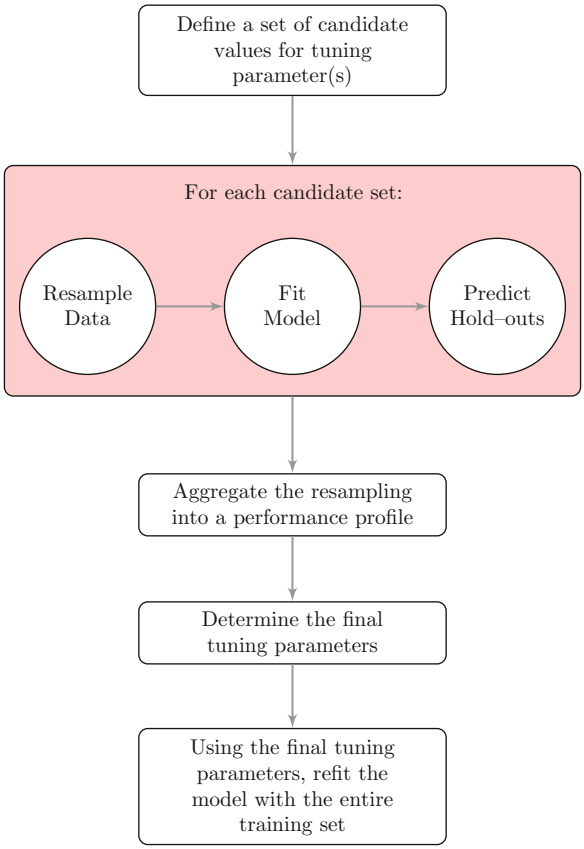

Fig. 4.4: A schematic of the parameter tuning process. An example of a candidate set of tuning parameter values for K-nearest neighbors might be odd numbers between 1 and 9. For each of these values, the data would be resampled multiple times to assess model performance for each value

Evaluating the model on a test set is the obvious choice, but, to get reasonable precision of the performance values, the size of the test set may need to be large.

An alternate approach to evaluating a model on a single test set is to *resample* the training set. This process uses several modified versions of the training set to build multiple models and then uses statistical methods to provide honest estimates of model performance (i.e., not overly optimistic). Section 4.4 illustrates several resampling techniques, and Sect. 4.6 discusses approaches to choose the final parameters using the resampling results.

4.3 Data Splitting

Now that we have outlined the general procedure for finding optimal tuning parameters, we turn to discussing the heart of the process: data splitting. A few of the common steps in model building are:

- Pre-processing the predictor data
- Estimating model parameters
- Selecting predictors for the model
- Evaluating model performance
- Fine tuning class prediction rules (via ROC curves, etc.)

Given a fixed amount of data, the modeler must decide how to "spend" their data points to accommodate these activities.

One of the first decisions to make when modeling is to decide which samples will be used to evaluate performance. Ideally, the model should be evaluated on samples that were not used to build or fine-tune the model, so that they provide an unbiased sense of model effectiveness. When a large amount of data is at hand, a set of samples can be set aside to evaluate the final model. The "training" data set is the general term for the samples used to create the model, while the "test" or "validation" data set is used to qualify performance.

However, when the number of samples is not large, a strong case can be made that a test set should be avoided because every sample may be needed for model building. Additionally, the size of the test set may not have sufficient power or precision to make reasonable judgements. Several researchers (Molinaro 2005; Martin and Hirschberg 1996; Hawkins et al. 2003) show that validation using a single test set can be a poor choice. Hawkins et al. (2003) concisely summarize this point: "holdout samples of tolerable size [...] do not match the cross-validation itself for reliability in assessing model fit and are hard to motivate." Resampling methods, such as cross-validation, can be used to produce appropriate estimates of model performance using the training set. These are discussed in length in Sect. 4.4. Although resampling techniques can be misapplied, such as the example shown in Ambroise and McLachlan (2002), they often produce performance estimates superior to a single test set because they evaluate many alternate versions of the data.

If a test set is deemed necessary, there are several methods for splitting the samples. Nonrandom approaches to splitting the data are sometimes appropriate. For example,

- If a model was being used to predict patient outcomes, the model may be created using certain patient sets (e.g., from the same clinical site or disease stage), and then tested on a different sample population to understand how well the model generalizes.
- In chemical modeling for drug discovery, new "chemical space" is constantly being explored. We are most interested in accurate predictions in the chemical space that is currently being investigated rather than the space that

was evaluated years prior. The same could be said for spam filtering; it is more important for the model to catch the new spamming techniques rather than prior spamming schemes.

However, in most cases, there is the desire to make the training and test sets as homogeneous as possible. Random sampling methods can be used to create similar data sets.

The simplest way to split the data into a training and test set is to take a simple random sample. This does not control for any of the data attributes, such as the percentage of data in the classes. When one class has a disproportionately small frequency compared to the others, there is a chance that the distribution of the outcomes may be substantially different between the training and test sets.

To account for the outcome when splitting the data, stratified random sampling applies random sampling within subgroups (such as the classes). In this way, there is a higher likelihood that the outcome distributions will match. When the outcome is a number, a similar strategy can be used; the numeric values are broken into similar groups (e.g., low, medium, and high) and the randomization is executed within these groups.

Alternatively, the data can be split on the basis of the predictor values. Willett (1999) and Clark (1997) propose data splitting based on *maximum dissimilarity sampling*. Dissimilarity between two samples can be measured in a number of ways. The simplest method is to use the distance between the predictor values for two samples. If the distance is small, the points are in close proximity. Larger distances between points are indicative of dissimilarity. To use dissimilarity as a tool for data splitting, suppose the test set is initialized with a single sample. The dissimilarity between this initial sample and the unallocated samples can be calculated. The unallocated sample that is most dissimilar would then be added to the test set. To allocate more samples to the test set, a method is needed to determine the dissimilarities between *groups* of points (i.e., the two in the test set and the unallocated points). One approach is to use the average or minimum of the dissimilarities. For example, to measure the dissimilarities between the two samples in the test set and a single unallocated point, we can determine the two dissimilarities and average them. The third point added to the test set would be chosen as having the maximum average dissimilarity to the existing set. This process would continue until the targeted test set size is achieved.

Figure 4.5 illustrates this process for the example classification data. Dissimilarity sampling was conducted separately within each class. First, a sample within each class was chosen to start the process (designated as ■ and ● in the figure). The dissimilarity of the initial sample to the unallocated samples within the class was computed and the most dissimilar point was added to the test set. For the first class, the most dissimilar point was in the extreme Southwest of the initial sample. On the second round, the dissimilarities were aggregated using the minimum (as opposed to the average). Again,

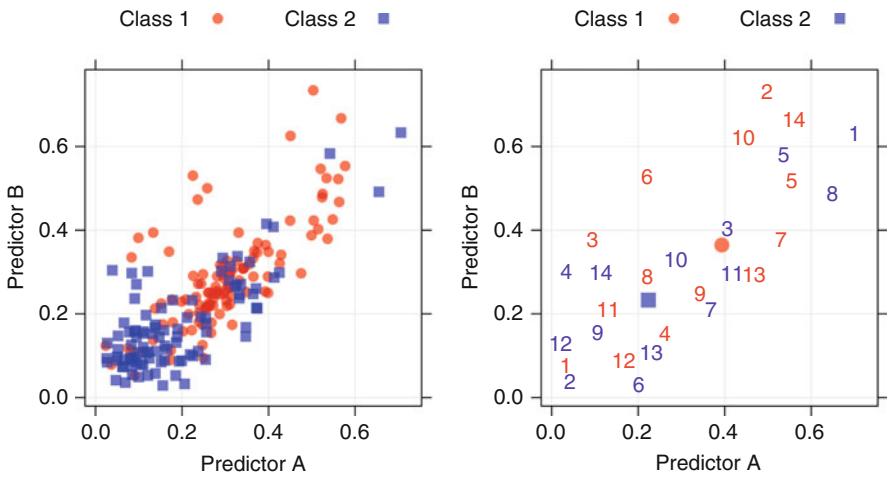

Fig. 4.5: An example of maximum dissimilarity sampling to create a test set. After choosing an initial sample within a class, 14 more samples were added

for the first class, the chosen point was far in the Northeast of the predictor space. As the sampling proceeds, samples were selected on the periphery of the data then work inward.

Martin et al. (2012) compares different methods of splitting data, including random sampling, dissimilarity sampling, and other methods.

4.4 Resampling Techniques

Generally, resampling techniques for estimating model performance operate similarly: a subset of samples are used to fit a model and the remaining samples are used to estimate the efficacy of the model. This process is repeated multiple times and the results are aggregated and summarized. The differences in techniques usually center around the method in which subsamples are chosen. We will consider the main flavors of resampling in the next few subsections.

k-Fold Cross-Validation

The samples are randomly partitioned into k sets of roughly equal size. A model is fit using the all samples except the first subset (called the first *fold*). The held-out samples are predicted by this model and used to estimate performance measures. The first subset is returned to the training set and

procedure repeats with the second subset held out, and so on. The k resampled estimates of performance are summarized (usually with the mean and standard error) and used to understand the relationship between the tuning parameter(s) and model utility. The cross-validation process with $k = 3$ is depicted in Fig. 4.6.

A slight variant of this method is to select the k partitions in a way that makes the folds balanced with respect to the outcome (Kohavi 1995). Stratified random sampling, previously discussed in Sect. 4.3, creates balance with respect to the outcome.

Another version, leave-one-out cross-validation (LOOCV), is the special case where k is the number of samples. In this case, since only one sample is held-out at a time, the final performance is calculated from the k individual held-out predictions. Additionally, repeated k-fold cross-validation replicates the procedure in Fig. 4.6 multiple times. For example, if 10-fold cross-validation was repeated five times, 50 different held-out sets would be used to estimate model efficacy.

The choice of k is usually 5 or 10, but there is no formal rule. As k gets larger, the difference in size between the training set and the resampling subsets gets smaller. As this difference decreases, the *bias* of the technique becomes smaller (i.e., the bias is smaller for $k = 10$ than $k = 5$). In this context, the bias is the difference between the estimated and true values of performance.

Another important aspect of a resampling technique is the uncertainty (i.e., variance or noise). An unbiased method may be estimating the correct value (e.g., the true theoretical performance) but may pay a high price in uncertainty. This means that repeating the resampling procedure may produce a very different value (but done enough times, it will estimate the true value). k-fold cross-validation generally has high variance compared to other methods and, for this reason, might not be attractive. It should be said that for large training sets, the potential issues with variance and bias become negligible.

From a practical viewpoint, larger values of k are more computationally burdensome. In the extreme, LOOCV is most computationally taxing because it requires as many model fits as data points and each model fit uses a subset that is nearly the same size of the training set. Molinaro (2005) found that leave-one-out and k =10-fold cross-validation yielded similar results, indicating that $k = 10$ is more attractive from the perspective of computational efficiency. Also, small values of k, say 2 or 3, have high bias but are very computationally efficient. However, the bias that comes with small values of k is about the same as the bias produced by the bootstrap (see below), but with much larger variance.

Research (Molinaro 2005; Kim 2009) indicates that repeating k-fold cross-validation can be used to effectively increase the precision of the estimates while still maintaining a small bias.

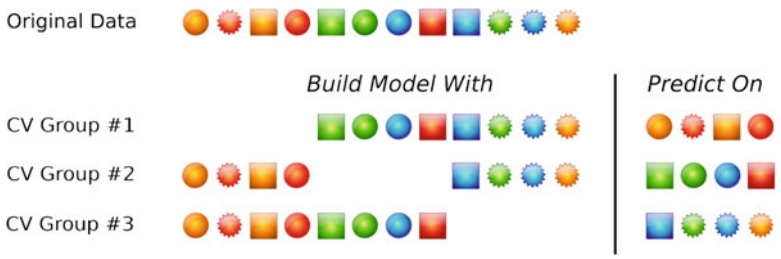

Fig. 4.6: A schematic of threefold cross-validation. Twelve training set samples are represented as symbols and are allocated to three groups. These groups are left out in turn as models are fit. Performance estimates, such as the error rate or R^2 are calculated from each set of held-out samples. The average of the three performance estimates would be the cross-validation estimate of model performance. In practice, the number of samples in the held-out subsets can vary but are roughly equal size

Generalized Cross-Validation

For linear regression models, there is a formula for approximating the leave-one-out error rate. The generalized cross-validation (GCV) statistic (Golub et al. 1979) does not require iterative refitting of the model to different data subsets. The formula for this statistic is the i_{th} training set outcome

$$\text{GCV} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - df/n}\right)^2,$$

where y_i is the i^{th} in the training set set outcome, $\hat{y}_i$ is the model prediction of that outcome, and df is the degrees of freedom of the model. The degrees of freedom are an accounting of how many parameters are estimated by the model and, by extension, a measure of complexity for linear regression models. Based on this equation, two models with the same sums of square errors (the numerator) would have different GCV values if the complexities of the models were different.

Repeated Training/Test Splits

Repeated training/test splits is also known as "leave-group-out cross-validation" or "Monte Carlo cross-validation." This technique simply creates multiple splits of the data into modeling and prediction sets (see Fig. 4.7). The proportion of the data going into each subset is controlled by the practitioner as is the number of repetitions. As previously discussed, the bias

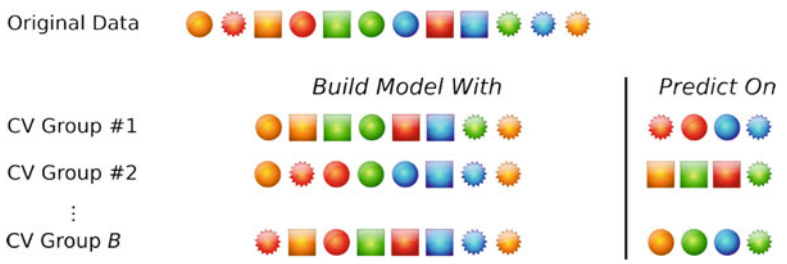

Fig. 4.7: A schematic of B repeated training and test set partitions. Twelve training set samples are represented as symbols and are allocated to B subsets that are 2/3 of the original training set. One difference between this procedure and k-fold cross-validation are that samples can be represented in multiple held-out subsets. Also, the number of repetitions is usually larger than in k-fold cross-validation

of the resampling technique decreases as the amount of data in the subset approaches the amount in the modeling set. A good rule of thumb is about 75–80 %. Higher proportions are a good idea if the number of repetitions is large.

The number of repetitions is important. Increasing the number of subsets has the effect of decreasing the uncertainty of the performance estimates. For example, to get a gross estimate of model performance, 25 repetitions will be adequate if the user is willing to accept some instability in the resulting values. However, to get stable estimates of performance, it is suggested to choose a larger number of repetitions (say 50–200). This is also a function of the proportion of samples being randomly allocated to the prediction set; the larger the percentage, the more repetitions are needed to reduce the uncertainty in the performance estimates.

The Bootstrap

A bootstrap sample is a random sample of the data taken *with replacement* (Efron and Tibshirani 1986). This means that, after a data point is selected for the subset, it is still available for further selection. The bootstrap sample is the same size as the original data set. As a result, some samples will be represented multiple times in the bootstrap sample while others will not be selected at all. The samples not selected are usually referred to as the "out-of-bag" samples. For a given iteration of bootstrap resampling, a model is built on the selected samples and is used to predict the out-of-bag samples (Fig. 4.8).

In general, bootstrap error rates tend to have less uncertainty than k-fold cross-validation (Efron 1983). However, on average, 63.2 % of the data points the bootstrap sample are represented at least once, so this technique has bias

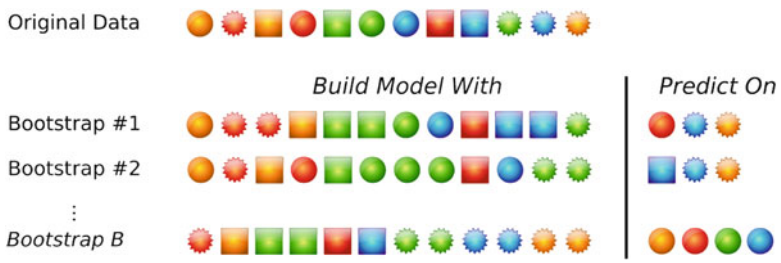

Fig. 4.8: A schematic of bootstrap resampling. Twelve training set samples are represented as symbols and are allocated to B subsets. Each subset is the same size as the original and can contain multiple instances of the same data point. Samples not selected by the bootstrap are predicted and used to estimate model performance

similar to k-fold cross-validation when $k \approx 2$. If the training set size is small, this bias may be problematic, but will decrease as the training set sample size becomes larger.

A few modifications of the simple bootstrap procedure have been devised to eliminate this bias. The "632 method" (Efron 1983) addresses this issue by creating a performance estimate that is a combination of the simple bootstrap estimate and the estimate from re-predicting the training set (e.g., the apparent error rate). For example, if a classification model was characterized by its error rate, the 632 method would use

$$(0.632 \times \text{simple bootstrap estimate}) + (0.368 \times \text{apparent error rate}).$$

The modified bootstrap estimate reduces the bias, but can be unstable with small samples sizes. This estimate can also result in unduly optimistic results when the model severely over-fits the data, since the apparent error rate will be close to zero. Efron and Tibshirani (1997) discuss another technique, called the "632+ method," for adjusting the bootstrap estimates.

4.5 Case Study: Credit Scoring

A straightforward application of predictive models is credit scoring. Existing data can be used to create a model to predict the probability that applicants have good credit. This information can be used to quantify the risk to the lender.

The German credit data set is a popular tool for benchmarking machine learning algorithms. It contains 1,000 samples that have been given labels of good and bad credit. In the data set, 70 % were rated as having good

credit. As discussed in Sect. 11.2, when evaluating the accuracy of a model, the baseline accuracy rate to beat would be 70 % (which we could achieve by simply predicting all samples to have good credit).

Along with these outcomes, data were collected related to credit history, employment, account status, and so on. Some predictors are numeric, such as the loan amount. However, most of the predictors are categorical in nature, such as the purpose of the loan, gender, or marital status. The categorical predictors were converted to "dummy variables" that related to a single category. For example, the applicant's residence information was categorized as either "rent," "own," or "free housing." This predictor would be converted to three yes/no bits of information for each category. For example, one predictor would have a value of one if the applicant rented and is zero otherwise. Creation of dummy variables is discussed at length in Sect. 3.6. In all, there were 41 predictors used to model the credit status of an individual.

We will use these data to demonstrate the process of tuning models using resampling, as defined in Fig. 4.4. For illustration, we took a stratified random sample of 800 customers to use for training models. The remaining samples will be used as a test set to verify performance when a final model is determined. Section 11.2 will discuss the results of the test set in more detail.

4.6 Choosing Final Tuning Parameters

Once model performance has been quantified across sets of tuning parameters, there are several philosophies on how to choose the final settings. The simplest approach is to pick the settings associated with the numerically best performance estimates.

For the credit scoring example, a nonlinear support vector machine model[1] was evaluated over cost values ranging from 2^{-2} to 2^{7}. Each model was evaluated using five repeats of 10-fold cross-validation. Figure 4.9 and Table 4.1 show the accuracy profile across the candidate values of the cost parameter. For each model, cross-validation generated 50 different estimates of the accuracy; the solid points in Fig. 4.9 are the average of these estimates. The bars reflect the average plus/minus two-standard errors of the mean. The profile shows an increase in accuracy until the cost value is one. Models with cost values between 1 and 16 are relatively constant; after which, the accuracy decreases (likely due to over-fitting). The numerically optimal value of the cost parameter is 8, with a corresponding accuracy rate of 75 %. Notice that the apparent accuracy rate, determined by re-predicting the training set samples, indicates that the model improves as the cost is increased, although more complex models over-fit the training set.

[1] This model uses a radial basis function kernel, defined in Sect. 13.4. Although not explored here, we used the analytical approach discussed later for determining the kernel parameter and fixed this value for all resampling techniques.

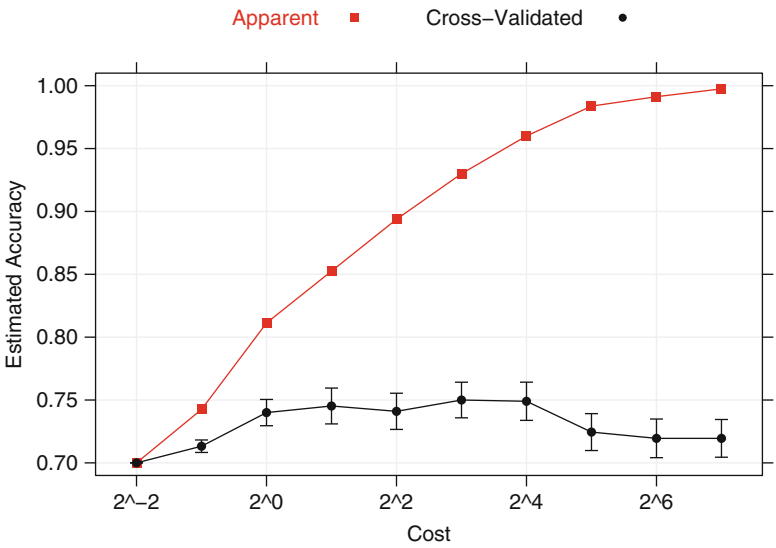

Fig. 4.9: The performance profile of a radial basis function support vector machine for the credit scoring example over different values of the cost parameter. The *vertical lines* indicate ± two-standard errors of the accuracy

In general, it may be a good idea to favor simpler models over more complex ones and choosing the tuning parameters based on the numerically optimal value may lead to models that are overly complicated. Other schemes for choosing less complex models should be investigated as they might lead to simpler models that provide acceptable performance (relative to the numerically optimal settings).

The "one-standard error" method for choosing simpler models finds the numerically optimal value and its corresponding standard error and then seeks the simplest model whose performance is within a single standard error of the numerically best value. This procedure originated with classification and regression trees (Breiman et al. (1984) and Sects. 8.1 and 14.1). In Fig. 4.10, the standard error of the accuracy values when the cost is 8 is about 0.7 %. This technique would find the simplest tuning parameter settings associated with accuracy no less than 74.3 % (75 %–0.7 %). This procedure would choose a value of 2 for the cost parameter.

Another approach is to choose a simpler model that is within a certain tolerance of the numerically best value. The percent decrease in performance could be quantified by $(X - O)/O$ where X is the performance value and O is the numerically optimal value. For example, in Fig. 4.9, the best accuracy value across the profile was 75 %. If a 4 % loss in accuracy was acceptable as a trade-off for a simpler model, accuracy values greater than 71.2 % would

Table 4.1: Repeated cross-validation accuracy results for the support vector machine model

	Resampled accuracy (%)		
Cost	Mean	Std. error	% Tolerance
0.25	70.0	0.0	−6.67
0.50	71.3	0.2	−4.90
1.00	74.0	0.5	−1.33
2.00	74.5	0.7	−0.63
4.00	74.1	0.7	−1.20
8.00	75.0	0.7	0.00
16.00	74.9	0.8	−0.13
32.00	72.5	0.7	−3.40
64.00	72.0	0.8	−4.07
128.00	72.0	0.8	−4.07

The one-standard error rule would select the simplest model with accuracy no less than 74.3 % (75 %–0.7 %). This corresponds to a cost value of 2. The "pick-the-best" solution is shown in bold

be acceptable. For the profile in Fig. 4.9, a cost value of 1 would be chosen using this approach.

As an illustration, additional resampling methods were applied to the same data: repeated 10-fold cross-validation, LOOCV, the bootstrap (with and without the 632 adjustment), and repeated training/test splits (with 20 % held-out). The latter two methods used 50 resamples to estimate performance.

The results are shown in Fig. 4.10. A common pattern within the cross-validation methods is seen where accuracy peaks at cost values between 4 and 16 and stays roughly constant within this window.

In each case, performance rapidly increases with the cost value and then, after the peak, decreases at a slower rate as over-fitting begins to occur. The cross-validation techniques estimate the accuracy to be between 74.5 % and 76.6 %. Compared to the other methods, the simple bootstrap is slightly pessimistic, estimating the accuracy to be 74.2 % while the 632 rule appears to overcompensate for the bias and estimates the accuracy to be 82.3 %. Note that the standard error bands of the simple 10-fold cross-validation technique are larger than the other methods, mostly because the standard error is a function of the number of resamples used (10 versus the 50 used by the bootstrap or repeated splitting).

The computational times varied considerably. The fastest was 10-fold cross-validation, which clocked in at 0.82 min. Repeated cross-validation, the bootstrap, and repeated training-test splits fit the same number of models and, on average, took about 5-fold more time to finish. LOOCV, which fits as many models as there are samples in the training set, took 86-fold longer and should only be considered when the number of samples is very small.

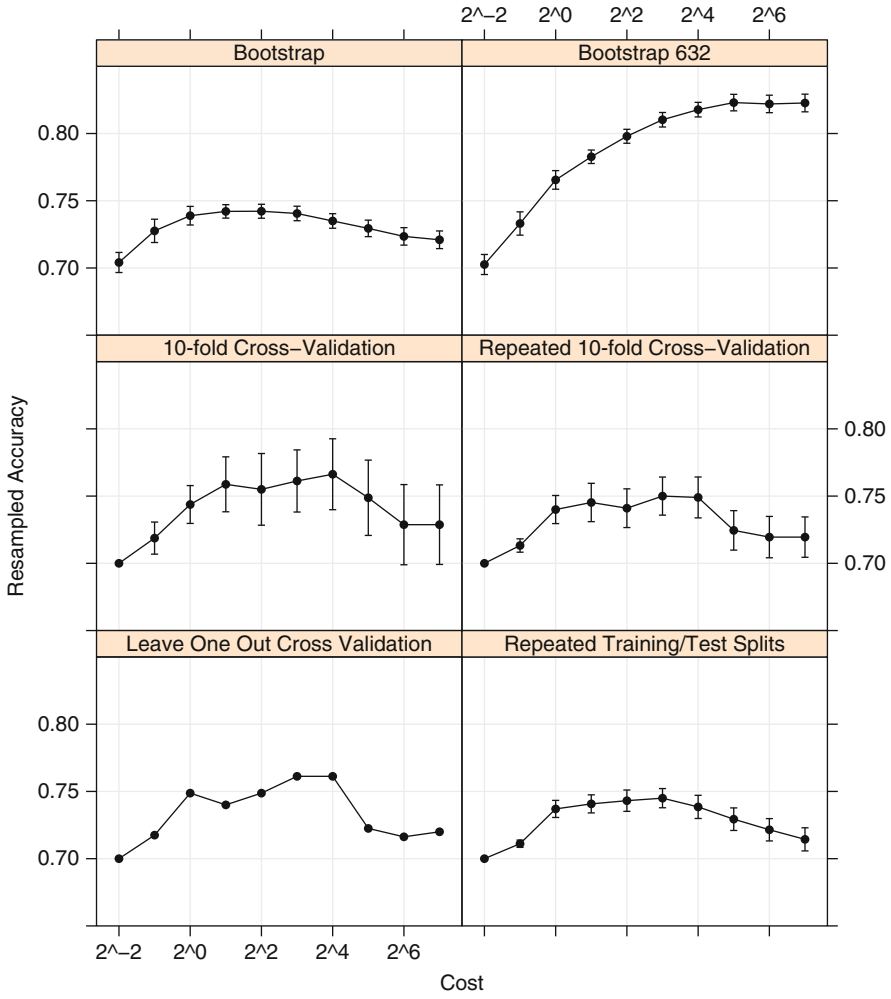

Fig. 4.10: The performance profile of nonlinear support vector machine over different values of the cost parameter for the credit scoring example using several different resampling procedures. The *vertical lines* indicate ± two-standard errors of the accuracy

4.7 Data Splitting Recommendations

As previously discussed, there is a strong technical case to be made against a single, independent test set:

- A test set is a single evaluation of the model and has limited ability to characterize the uncertainty in the results.

- Proportionally large test sets divide the data in a way that increases bias in the performance estimates.
- With small sample sizes:
  - The model may need every possible data point to adequately determine model values.
  - The uncertainty of the test set can be considerably large to the point where different test sets may produce very different results.
- Resampling methods can produce reasonable predictions of how well the model will perform on future samples.

No resampling method is uniformly better than another; the choice should be made while considering several factors. If the samples size is small, we recommend repeated 10-fold cross-validation for several reasons: the bias and variance properties are good and, given the sample size, the computational costs are not large. If the goal is to choose between models, as opposed to getting the best indicator of performance, a strong case can be made for using one of the bootstrap procedures since these have very low variance. For large sample sizes, the differences between resampling methods become less pronounced, and computational efficiency increases in importance. Here, simple 10-fold cross-validation should provide acceptable variance, low bias, and is relatively quick to compute.

Varma and Simon (2006) and Boulesteix and Strobl (2009) note that there is a potential bias that can occur when estimating model performance during parameter tuning. Suppose that the final model is chosen to correspond to the tuning parameter value associated with the smallest error rate. This error rate has the potential to be optimistic since it is a random quantity that is chosen from a potentially large set of tuning parameters. Their research is focused on scenarios with a small number of samples and a large number of predictors, which exacerbates the problem. However, for moderately large training sets, our experience is that this bias is small. In later sections, comparisons are made between resampled estimates of performance and those derived from a test set. For these particular data sets, the *optimization bias* is insubstantial.

4.8 Choosing Between Models

Once the settings for the tuning parameters have been determined for each model, the question remains: how do we choose between multiple models? Again, this largely depends on the characteristics of the data and the type of questions being answered. However, predicting which model is most fit for purpose can be difficult. Given this, we suggest the following scheme for finalizing the type of model:

1. Start with several models that are the least interpretable and most flexible, such as boosted trees or support vector machines. Across many problem domains, these models have a high likelihood of producing the empirically optimum results (i.e., most accurate).
2. Investigate simpler models that are less opaque (e.g., not complete black boxes), such as multivariate adaptive regression splines (MARS), partial least squares, generalized additive models, or naïve Bayes models.
3. Consider using the simplest model that reasonably approximates the performance of the more complex methods.

Using this methodology, the modeler can discover the "performance ceiling" for the data set before settling on a model. In many cases, a range of models will be equivalent in terms of performance so the practitioner can weight the benefits of different methodologies (e.g., computational complexity, easy of prediction, interpretability). For example, a nonlinear support vector machine or random forest model might have superior accuracy, but the complexity and scope of the prediction equation may prohibit exporting the prediction equation to a production system. However, if a more interpretable model, such as a MARS model, yielded similar accuracy, the implementation of the prediction equation would be trivial and would also have superior execution time.

Consider the credit scoring support vector machine classification model that was characterized using resampling in Sect. 4.6. Using repeated 10-fold cross-validation, the accuracy for this model was estimated to be 75 % with most of the resampling results between 66 % and 82 %.

Logistic regression (Sect. 12.2) is a more simplistic technique than the nonlinear support vector machine model for estimating a classification boundary. It has no tuning parameters and its prediction equation is simple and easy to implement using most software. Using the same cross-validation scheme, the estimated accuracy for this model was 74.9 % with most of the resampling results between 66 % and 82 %.

The same 50 resamples were used to evaluate each model. Figure 4.11 uses box plots to illustrate the distribution of the resampled accuracy estimates. Clearly, there is no performance loss by using a more straightforward model for these data.

Hothorn et al. (2005) and Eugster et al. (2008) describe statistical methods for comparing methodologies based on resampling results. Since the accuracies were measured using identically resampled data sets, statistical methods for *paired comparisons* can be used to determine if the differences between models are statistically significant. A paired t-test can be used to evaluate the hypothesis that the models have equivalent accuracies (on average) or, analogously, that the mean difference in accuracy for the resampled data sets is zero. For these two models, the average difference in model accuracy was 0.1 %, with the logistic regression supplying the better results. The 95 % confidence interval for this difference was (−1.2 %, 1 %), indicating that there

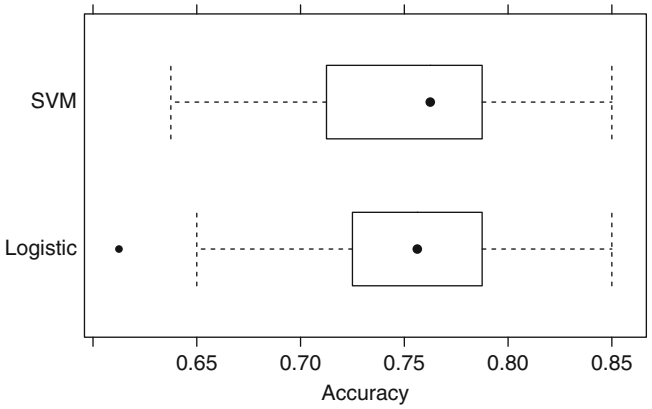

Fig. 4.11: A comparison of the cross-validated accuracy estimates from a support vector machine model and a logistic regression model for the credit scoring data described in Sect. 4.5

is no evidence to support the idea that the accuracy for either model is significantly better. This makes intuitive sense; the resampled accuracies in Fig. 4.11 range from 61.3 % to 85 %; given this amount of variation in the results, a 0.1 % improvement of accuracy is not meaningful.

When a model is characterized in multiple ways, there is a possibility that comparisons between models can lead to different conclusions. For example, if a model is created to predict two classes, sensitivity and specificity may be used to characterize the efficacy of models (see Chap. 11). If the data set includes more events than nonevents, the sensitivity can be estimated with greater precision than the specificity. With increased precision, there is a higher likelihood that models can be differentiated in terms of sensitivity than for specificity.

4.9 Computing

The R language is used to demonstrate modeling techniques. A concise review of R and its basic usage are found in Appendix B. Those new to R should review these materials prior to proceeding. The following sections will reference functions from the AppliedPredictiveModeling, caret, Design, e1071, ipred and MASS packages. Syntax will be demonstrated using the simple two-class example shown in Figs. 4.2 and 4.3 and the data from the credit scoring case study.