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.

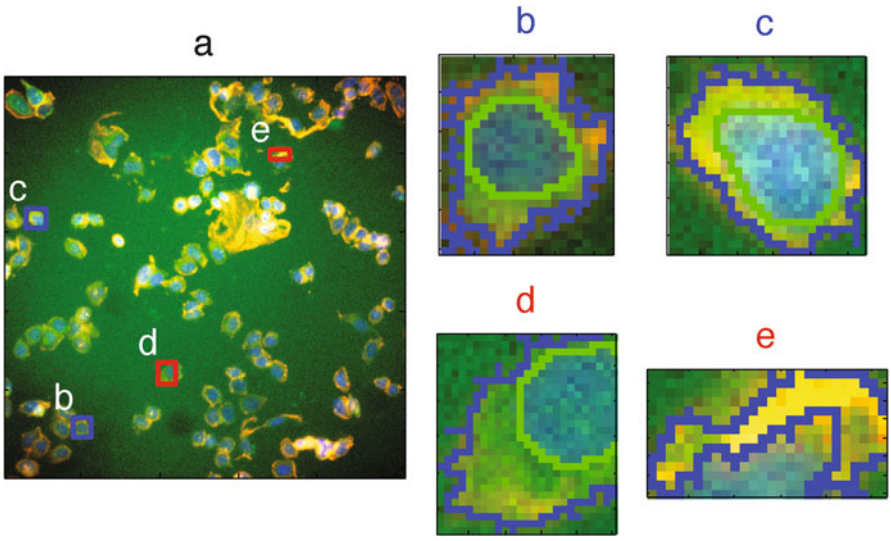

Fig. 3.1: An image showing cell segmentation from Hill et al. (2007). The *red boxes* [panels (**d**) and (**e**)] show poorly segmented cells while the cells in the *blue boxes* are examples of proper segmentation

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.

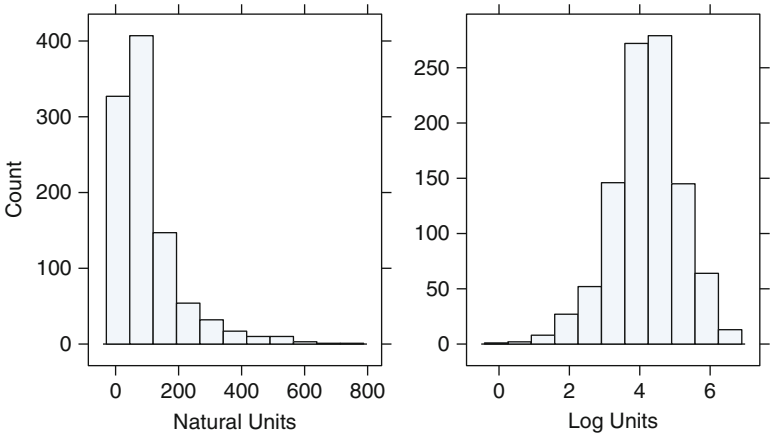

Fig. 3.2: *Left*: a histogram of the standard deviation of the intensity of the pixels in actin filaments. This predictor has a strong right skewness with a concentration of points with low values. For this variable, the ratio of the smallest to largest value is 870 and a skewness value of 2.39. *Right*: 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.

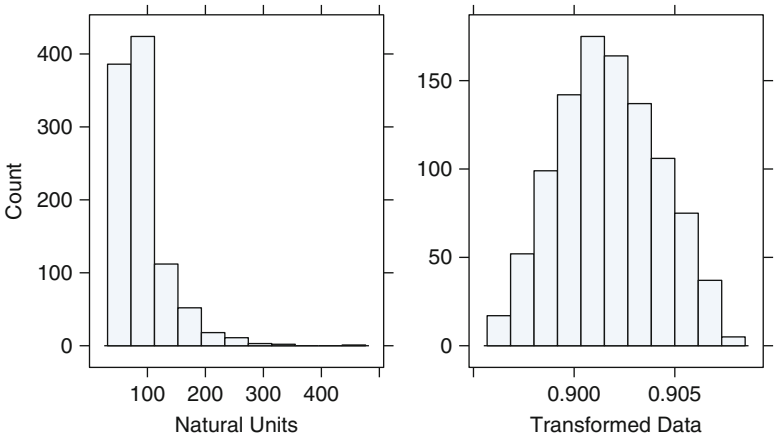

Fig. 3.3: *Left*: a histogram of the cell perimeter predictor. *Right*: the same data after a Box–Cox transformation with λ estimated to be -1.1

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:

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

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.

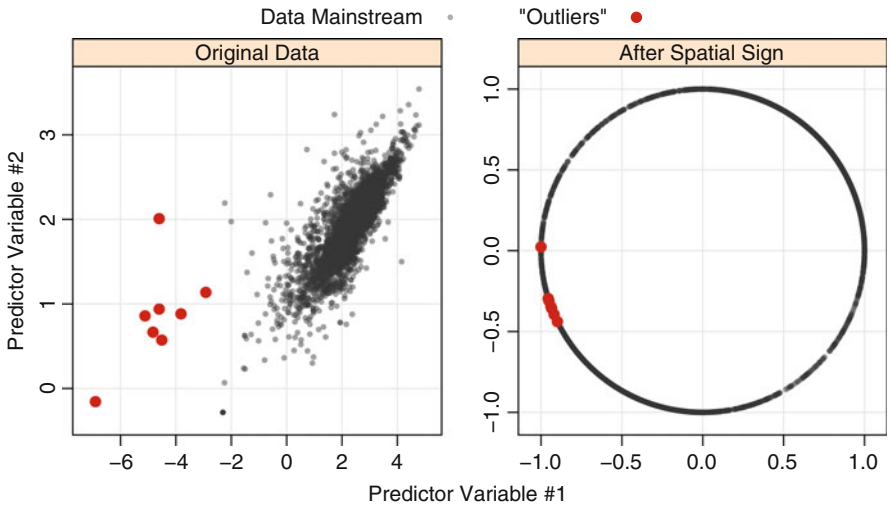

Fig. 3.4: *Left*: An illustrative example with a group of outlying data points. *Right*: When the original data are transformed, the results bring the outliers towards the majority of the data

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.

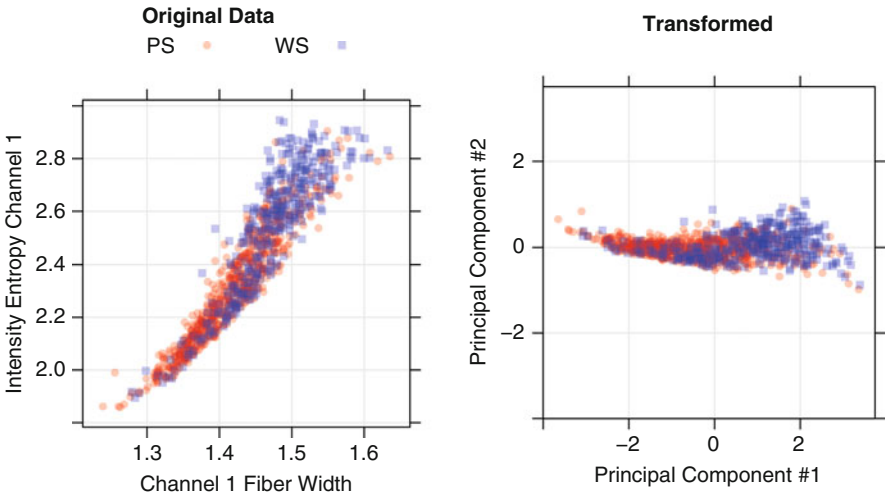

Fig. 3.5: An example of the principal component transformation for the cell segmentation data. The *shapes* and *colors* indicate which cells were poorly segmented or well segmented

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).

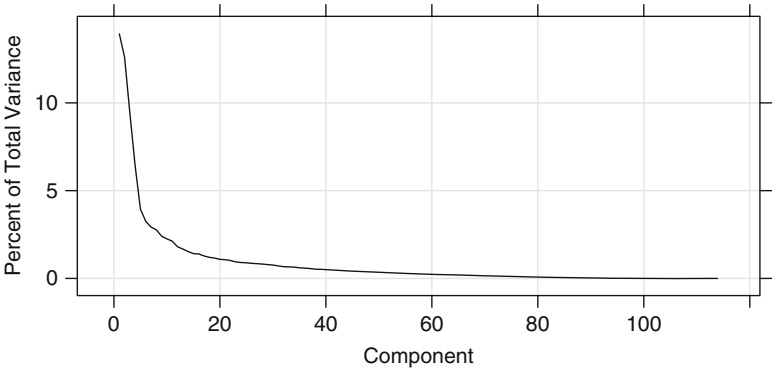

Fig. 3.6: A "scree plot" where the percentage of the total variance explained by each component is shown

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,

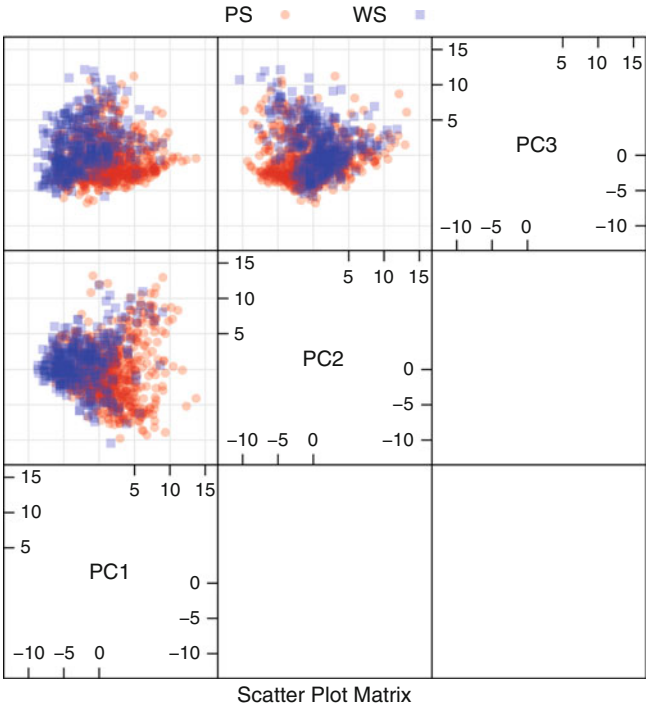

Fig. 3.7: A plot of the first three principal components for the cell segmentation data, colored by cell type

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

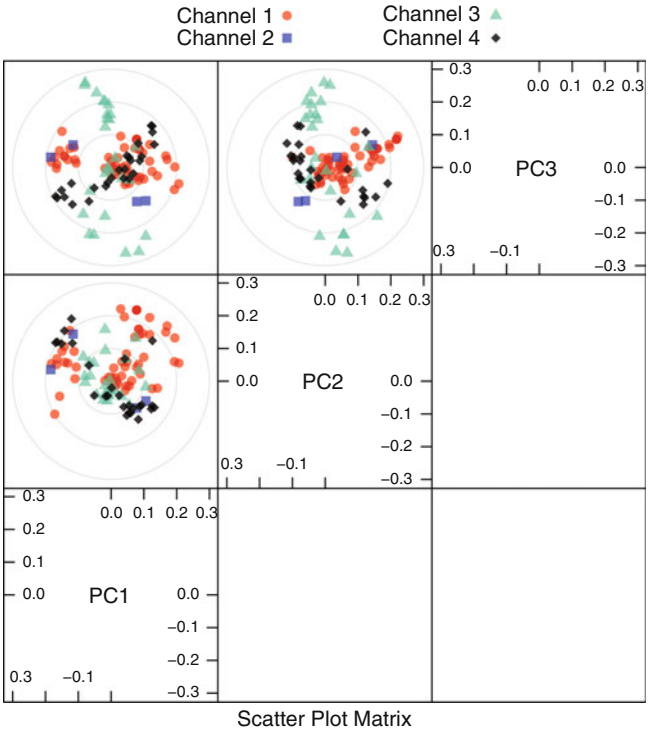

Fig. 3.8: A plot of the loadings of the first three principal components for the cell segmentation data, colored by optical channel. Recall that channel one was associated with the cell body, channel two with the cell nucleus, channel three with actin, and channel four with tubulin

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-

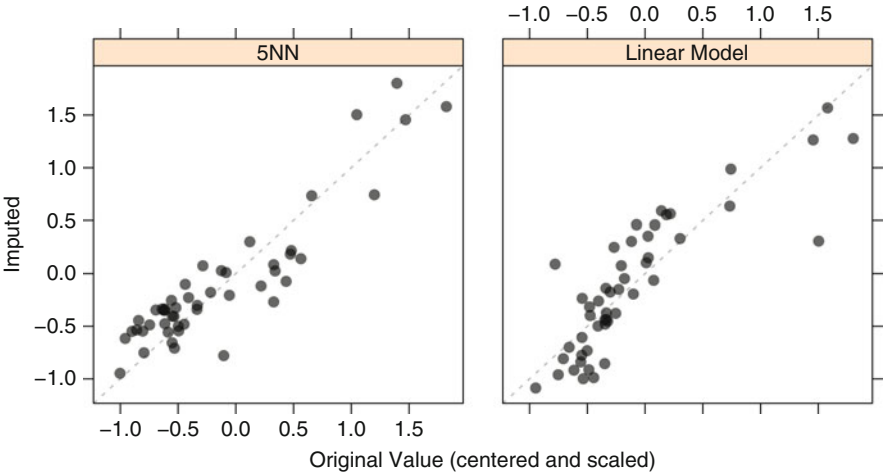

Fig. 3.9: After simulating 50 missing test set values at random for the cell perimeter data, two different imputation models were built with the training set and applied to the missing test set values. This plot shows the centered and scaled values before and after imputation

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

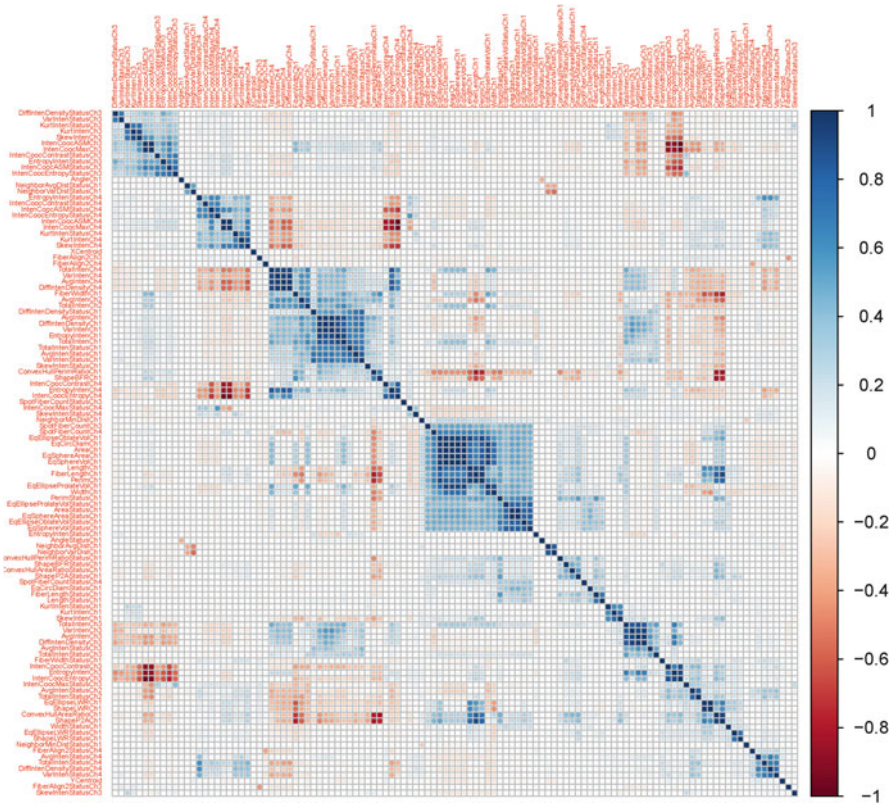

Fig. 3.10: A visualization of the cell segmentation correlation matrix. The order of the variables is based on a clustering algorithm

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.

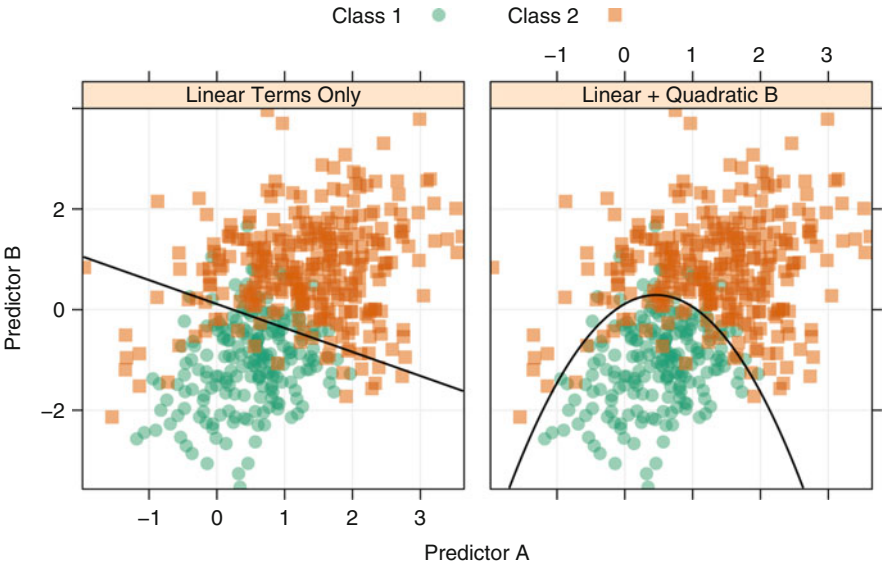

Fig. 3.11: Classification boundaries from two logistic regression models. The *left panel* has linear terms for the two predictors while the *right panel* has an additional quadratic term for predictor B. This model is discussed in more detail in Chap. 12

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.

3.8 Computing

This section uses data from the AppliedPredictiveModeling package and functions from the caret, corrplot, e1071, and lattice packages.

There are two locations where relevant R code can be found:

- The `chapters` directory of the AppliedPredictiveModeling package contains specific code to reproduce the specific models used in the chapter. This is intended to allow the reader to see exactly how the models used here were created.
- Many chapters in this book contain sections at the end of the chapter that detail how of the computations can be performed in R more generally. For example, there are individual functions that correspond to the data pre-processing methods shown in this chapter. While the computing section provides these details, the individual functions might not be used directly in practice. For example, when using the `train` function, the pre-processing steps are specified in a single argument and the individual functions are not utilized. These sections do relate to the models created in each chapter, but as discussion points for the functions.

As such, the Computing sections in each chapter explains how to generally do the computations while the code in the `chapters` directory of the AppliedPredictiveModeling package is the best source for the calculations for the specific models in each chapter.

As discussed in Appendix B, there are a few useful R functions that can be used to find existing functions or classes of interest. The function `apropos` will search any loaded R packages for a given term. For example, to find functions for creating a confusion matrix within the currently loaded packages:

```
> apropos("confusion")
  [1] "confusionMatrix"       "confusionMatrix.train"
```

To find such a function in any package, the `RSiteSearch` function can help. Running the command:

```
> RSiteSearch("confusion", restrict = "functions")
```

will search online to find matches and will open a web browser to display the results.

The raw segmentation data set is contained in the AppliedPredictiveModeling package.[5] To load the data set into R:

```
> library(AppliedPredictiveModeling)
> data(segmentationOriginal)
```

There were fields that identified each cell (called `Cell`) and a factor vector that indicated which cells were well segmented (`Class`). The variable `Case`

[5] A preprocessed version of these data can also be found in the caret package and is used in later chapters.

indicated which cells were originally used for the training and test sets. The analysis in this chapter focused on the training set samples, so the data are filtered for these cells:

```
> segData <- subset(segmentationOriginal, Case == "Train")
```

The `Class` and `Cell` fields will be saved into separate vectors, then removed from the main object:

```
> cellID <- segData$Cell
> class <- segData$Class
> case <- segData$Case
> # Now remove the columns
> segData <- segData[, -(1:3)]
```

The original data contained several "status" columns which were binary versions of the predictors. To remove these, we find the column names containing `"Status"` and remove them:

```
> statusColNum <- grep("Status", names(segData))
> statusColNum
  [1]   2   4   9  10  11  12  14  16  20  21  22  26  27  28  30  32  34
 [18]  36  38  40  43  44  46  48  51  52  55  56  59  60  63  64  68  69
 [35]  70  72  73  74  76  78  80  82  84  86  88  92  93  94  97  98 103
 [52] 104 105 106 110 111 112 114
> segData <- segData[, -statusColNum]
```

Transformations

As previously discussed, some features exhibited significantly skewness. The `skewness` function in the e1071 package calculates the sample skewness statistic for each predictor:

```
> library(e1071)
> # For one predictor:
> skewness(segData$AngleCh1)
[1] -0.0243
> # Since all the predictors are numeric columns, the apply function can
> # be used to compute the skewness across columns.
> skewValues <- apply(segData, 2, skewness)
> head(skewValues)
     AngleCh1      AreaCh1 AvgIntenCh1 AvgIntenCh2 AvgIntenCh3 AvgIntenCh4
      -0.0243       3.5251       2.9592       0.8482       2.2023       1.9005
```

Using these values as a guide, the variables can be prioritized for visualizing the distribution. The basic R function `hist` or the `histogram` function in the lattice can be used to assess the shape of the distribution.

To determine which type of transformation should be used, the MASS package contains the `boxcox` function. Although this function estimates λ, it

does not create the transformed variable(s). A caret function, `BoxCoxTrans`, can find the appropriate transformation and apply them to the new data:

```
> library(caret)
> Ch1AreaTrans <- BoxCoxTrans(segData$AreaCh1)
> Ch1AreaTrans
  Box-Cox Transformation

  1009 data points used to estimate Lambda

  Input data summary:
     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      150     194     256     325     376    2190

  Largest/Smallest: 14.6
  Sample Skewness: 3.53

  Estimated Lambda: -0.9
> # The original data
> head(segData$AreaCh1)
  [1] 819 431 298 256 258 358
> # After transformation
> predict(Ch1AreaTrans, head(segData$AreaCh1))
  [1] 1.1085 1.1064 1.1045 1.1036 1.1036 1.1055
> (819^(-.9) - 1)/(-.9)
  [1] 1.1085
```

Another caret function, `preProcess`, applies this transformation to a set of predictors. This function is discussed below. The base R function `prcomp` can be used for PCA. In the code below, the data are centered and scaled prior to PCA.

```
> pcaObject <- prcomp(segData,
+                     center = TRUE, scale. = TRUE)
> # Calculate the cumulative percentage of variance which each component
> # accounts for.
> percentVariance <- pcaObject$sd^2/sum(pcaObject$sd^2)*100
> percentVariance[1:3]
  [1] 20.9 17.0 11.9
```

The transformed values are stored in `pcaObject` as a sub-object called `x`:

```
> head(pcaObject$x[, 1:5])
       PC1    PC2     PC3   PC4    PC5
  2   5.099  4.551 -0.0335 -2.64  1.278
  3  -0.255  1.198 -1.0206 -3.73  0.999
  4   1.293 -1.864 -1.2511 -2.41 -1.491
  12 -1.465 -1.566  0.4696 -3.39 -0.330
  15 -0.876 -1.279 -1.3379 -3.52  0.394
  16 -0.862 -0.329 -0.1555 -2.21  1.473
```

The another sub-object called `rotation` stores the variable loadings, where rows correspond to predictor variables and columns are associated with the components:

```
> head(pcaObject$rotation[, 1:3])
                    PC1     PC2      PC3
  AngleCh1      0.00121 -0.0128  0.00682
  AreaCh1       0.22917  0.1606  0.08981
  AvgIntenCh1 -0.10271  0.1797  0.06770
  AvgIntenCh2 -0.15483  0.1638  0.07353
  AvgIntenCh3 -0.05804  0.1120 -0.18547
  AvgIntenCh4 -0.11734  0.2104 -0.10506
```

The caret package class `spatialSign` contains functionality for the spatial sign transformation. Although we will not apply this technique to these data, the basic syntax would be `spatialSign(segData)`.

Also, these data do not have missing values for imputation. To impute missing values, the impute package has a function, `impute.knn`, that uses K-nearest neighbors to estimate the missing data. The previously mentioned `preProcess` function applies imputation methods based on K-nearest neighbors or bagged trees.

To administer a series of transformations to multiple data sets, the caret class `preProcess` has the ability to transform, center, scale, or impute values, as well as apply the spatial sign transformation and feature extraction. The function calculates the required quantities for the transformation. After calling the `preProcess` function, the `predict` method applies the results to a set of data. For example, to Box–Cox transform, center, and scale the data, then execute PCA for signal extraction, the syntax would be:

```
> trans <- preProcess(segData,
+                     method = c("BoxCox", "center", "scale", "pca"))
> trans
  Call:
  preProcess.default(x = segData, method = c("BoxCox", "center",
   "scale", "pca"))

  Created from 1009 samples and 58 variables
  Pre-processing: Box-Cox transformation, centered, scaled,
   principal component signal extraction

  Lambda estimates for Box-Cox transformation:
     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
    -2.00   -0.50   -0.10    0.05    0.30    2.00      11

  PCA needed 19 components to capture 95 percent of the variance
> # Apply the transformations:
> transformed <- predict(trans, segData)
> # These values are different than the previous PCA components since
> # they were transformed prior to PCA
> head(transformed[, 1:5])
```

```
       PC1    PC2    PC3   PC4    PC5
2    1.568  6.291 -0.333 -3.06 -1.342
3   -0.666  2.046 -1.442 -4.70 -1.742
4    3.750 -0.392 -0.669 -4.02  1.793
12   0.377 -2.190  1.438 -5.33 -0.407
15   1.064 -1.465 -0.990 -5.63 -0.865
16  -0.380  0.217  0.439 -2.07 -1.936
```

The order in which the possible transformation are applied is transformation, centering, scaling, imputation, feature extraction, and then spatial sign.

Many of the modeling functions have options to center and scale prior to modeling. For example, when using the `train` function (discussed in later chapters), there is an option to use `preProcess` prior to modeling within the resampling iterations.

Filtering

To filter for near-zero variance predictors, the caret package function `nearZeroVar` will return the column numbers of any predictors that fulfill the conditions outlined in Sect. 3.5. For the cell segmentation data, there are no problematic predictors:

```
> nearZeroVar(segData)
  integer(0)
> # When predictors should be removed, a vector of integers is
> # returned that indicates which columns should be removed.
```

Similarly, to filter on between-predictor correlations, the `cor` function can calculate the correlations between predictor variables:

```
> correlations <- cor(segData)
> dim(correlations)
  [1] 58 58
> correlations[1:4, 1:4]
              AngleCh1  AreaCh1 AvgIntenCh1 AvgIntenCh2
  AngleCh1     1.00000 -0.00263     -0.0430     -0.0194
  AreaCh1     -0.00263  1.00000     -0.0253     -0.1533
  AvgIntenCh1 -0.04301 -0.02530      1.0000      0.5252
  AvgIntenCh2 -0.01945 -0.15330      0.5252      1.0000
```

To visually examine the correlation structure of the data, the corrplot package contains an excellent function of the same name. The function has many options including one that will reorder the variables in a way that reveals clusters of highly correlated predictors. The following command was used to produce Fig. 3.10:

```
> library(corrplot)
> corrplot(correlations, order = "hclust")
```

The size and color of the points are associated with the strength of correlation between two predictor variables.

To filter based on correlations, the `findCorrelation` function will apply the algorithm in Sect. 3.5. For a given threshold of pairwise correlations, the function returns column numbers denoting the predictors that are recommended for deletion:

```
> highCorr <- findCorrelation(correlations, cutoff = .75)
> length(highCorr)
  [1] 33
> head(highCorr)
  [1] 23 40 43 36  7 15
> filteredSegData <- segData[, -highCorr]
```

There are also several functions in the subselect package that can accomplish the same goal.

Creating Dummy Variables

Several methods exist for creating dummy variables based on a particular model. Section 4.9 discusses different methods for specifying how the predictors enter into the model. One approach, the formula method, allows great flexibility to create the model function. Using formulas in model functions parameterizes the predictors such that not all categories have dummy variables. This approach will be shown in greater detail for linear regression.

As previously mentioned, there are occasions when a complete set of dummy variables is useful. For example, the splits in a tree-based model are more interpretable when the dummy variables encode all the information for that predictor. We recommend using the full set if dummy variables when working with tree-based models.

To illustrate the code, we will take a subset of the `cars` data set in the caret package. For 2005, Kelly Blue Book resale data for 804 GM cars were collected (Kuiper 2008). The object of the model was to predict the price of the car based on known characteristics. This demonstration will focus on the price, mileage, and car type (e.g., sedan) for a subset of vehicles:

```
> head(carSubset)
     Price Mileage  Type
214 19981   24323 sedan
299 21757    1853 sedan
460 15047   12305 sedan
728 15327    4318 sedan
162 20628   20770 sedan
718 16714   26328 sedan
> levels(carSubset$Type)
  [1] "convertible" "coupe"       "hatchback"   "sedan"       "wagon"
```

To model the price as a function of mileage and type of car, we can use the function `dummyVars` to determine encodings for the predictors. Suppose our first model assumes that the price can be modeled as a simple additive function of the mileage and type:

```
> simpleMod <- dummyVars(~Mileage + Type,
+                        data = carSubset,
+                        ## Remove the variable name from the
+                        ## column name
+                        levelsOnly = TRUE)
> simpleMod
Dummy Variable Object

Formula: ~Mileage + Type
2 variables, 1 factors
Factor variable names will be removed
```

To generate the dummy variables for the training set or any new samples, the `predict` method is used in conjunction with the `dummyVars` object:

```
> predict(simpleMod, head(carSubset))
    Mileage convertible coupe hatchback sedan wagon
214   24323           0     0         0     1     0
299    1853           0     0         0     1     0
460   12305           0     0         0     1     0
728    4318           0     0         0     1     0
162   20770           0     0         0     1     0
718   26328           0     0         0     1     0
```

The `type` field was expanded into five variables for five factor levels. The model is simple because it assumes that effect of the mileage is the same for every type of car. To fit a more advance model, we could assume that there is a *joint* effect of mileage and car type. This type of effect is referred to as an interaction. In the model formula, a colon between factors indicates that an interaction should be generated. For these data, this adds another five predictors to the data frame:

```
> withInteraction <- dummyVars(~Mileage + Type + Mileage:Type,
+                              data = carSubset,
+                              levelsOnly = TRUE)
> withInteraction
Dummy Variable Object

Formula: ~Mileage + Type + Mileage:Type
2 variables, 1 factors
Factor variable names will be removed
> predict(withInteraction, head(carSubset))
    Mileage convertible coupe hatchback sedan wagon Mileage:convertible
214   24323           0     0         0     1     0                   0
299    1853           0     0         0     1     0                   0
460   12305           0     0         0     1     0                   0
728    4318           0     0         0     1     0                   0
```

```
162   20770           0     0         0     1     0                  0
718   26328           0     0         0     1     0                  0
    Mileage:coupe Mileage:hatchback Mileage:sedan Mileage:wagon
214             0                 0         24323             0
299             0                 0          1853             0
460             0                 0         12305             0
728             0                 0          4318             0
162             0                 0         20770             0
718             0                 0         26328             0
```

Exercises

3.1. The UC Irvine Machine Learning Repository[6] contains a data set related to glass identification. The data consist of 214 glass samples labeled as one of seven class categories. There are nine predictors, including the refractive index and percentages of eight elements: Na, Mg, Al, Si, K, Ca, Ba, and Fe.

The data can be accessed via:

```
> library(mlbench)
> data(Glass)
> str(Glass)
'data.frame':        214 obs. of  10 variables:
 $ RI  : num  1.52 1.52 1.52 1.52 1.52 ...
 $ Na  : num  13.6 13.9 13.5 13.2 13.3 ...
 $ Mg  : num  4.49 3.6 3.55 3.69 3.62 3.61 3.6 3.61 3.58 3.6 ...
 $ Al  : num  1.1 1.36 1.54 1.29 1.24 1.62 1.14 1.05 1.37 1.36 ...
 $ Si  : num  71.8 72.7 73 72.6 73.1 ...
 $ K   : num  0.06 0.48 0.39 0.57 0.55 0.64 0.58 0.57 0.56 0.57 ...
 $ Ca  : num  8.75 7.83 7.78 8.22 8.07 8.07 8.17 8.24 8.3 8.4 ...
 $ Ba  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Fe  : num  0 0 0 0 0 0.26 0 0 0 0.11 ...
 $ Type: Factor w/ 6 levels "1","2","3","5",..: 1 1 1 1 1 1 1 1 1 1 ...
```

(a) Using visualizations, explore the predictor variables to understand their distributions as well as the relationships between predictors.
(b) Do there appear to be any outliers in the data? Are any predictors skewed?
(c) Are there any relevant transformations of one or more predictors that might improve the classification model?

3.2. The soybean data can also be found at the UC Irvine Machine Learning Repository. Data were collected to predict disease in 683 soybeans. The 35 predictors are mostly categorical and include information on the environmental conditions (e.g., temperature, precipitation) and plant conditions (e.g., left spots, mold growth). The outcome labels consist of 19 distinct classes.

[6] http://archive.ics.uci.edu/ml/index.html.

The data can be loaded via:

```
> library(mlbench)
> data(Soybean)
> ## See ?Soybean for details
```

(a) Investigate the frequency distributions for the categorical predictors. Are any of the distributions degenerate in the ways discussed earlier in this chapter?
(b) Roughly 18 % of the data are missing. Are there particular predictors that are more likely to be missing? Is the pattern of missing data related to the classes?
(c) Develop a strategy for handling missing data, either by eliminating predictors or imputation.

3.3. Chapter 5 introduces Quantitative Structure-Activity Relationship (QSAR) modeling where the characteristics of a chemical compound are used to predict other chemical properties. The caret package contains a QSAR data set from Mente and Lombardo (2005). Here, the ability of a chemical to permeate the blood-brain barrier was experimentally determined for 208 compounds. 134 descriptors were measured for each compound.

(a) Start R and use these commands to load the data:

```
> library(caret)
> data(BloodBrain)
> # use ?BloodBrain to see more details
```

The numeric outcome is contained in the vector `logBBB` while the predictors are in the data frame `bbbDescr`.
(b) Do any of the individual predictors have degenerate distributions?
(c) Generally speaking, are there strong relationships between the predictor data? If so, how could correlations in the predictor set be reduced? Does this have a dramatic effect on the number of predictors available for modeling?