Chapter 20
Factors That Can Affect Model Performance

Several of the preceding chapters have focused on technical pitfalls of predictive models, such as over-fitting and class imbalances. Often, true success may depend on aspects of the problem that are not directly related to the model itself. For example, there may be limitations to what the data can support or perhaps there may be subtle obstacles related to the goal of the modeling effort. One issue not discussed here is related to the acceptance of modeling, especially in areas where these techniques are viewed as novel or disruptive. Ayres (2007) offers a broad discussion of this issue. Another important aspect of modeling not discussed in this chapter is *feature engineering*; that is, methods to encode one or more predictors for the model.

This chapter discusses several important aspects of creating and maintaining predictive models. The first section looks at "Type III" errors: developing a model that answers the wrong question. An illustrative example is used to show that, due to sampling issues with the training set, the model gives predictions that answer a different question than the one of interest.

Noise, or error, has varying degrees of impact on models' predictive performance and occurs in three general forms in most data sets:

- Since many predictors are measured, they contain some level of systematic noise associated with the measurement system. Any extraneous noise in the predictors is likely to be propagated directly through the model prediction equation and results in poor performance.
- A second way noise can be introduced into the data is by the inclusion of non-informative predictors (e.g., predictors that have no relationship with the response). Some models have the ability to filter out irrelevant information, and hence their predictive performance is relatively unaffected.
- A third way noise enters the modeling process is through the response variable. As with predictors, some outcomes can be measured with a degree of systematic, unwanted noise. This type of error gives rise to an upper bound on model performance for which no pre-processing, model complexity, or tuning can overcome. For example, if a measured categorical outcome is mislabeled in the training data 10 % of the time, it is unlikely

M. Kuhn and K. Johnson, *Applied Predictive Modeling*,
DOI 10.1007/978-1-4614-6849-3_20,

that any model could truly achieve more than a 90 % accuracy rate. Of course, the modeler will not be aware of this and may expend considerable time chasing noise.

These aspects of modeling are explored in Sects. 20.2 and 20.3. Section 20.4 discusses the effect of discretizing outcomes on predictive performance while Sect. 20.5 examines the consequences of model *extrapolation*. We wrap up this chapter with an overview of how large data can impact model performance.

20.1 Type III Errors

One of the most common mistakes in modeling is to develop a model that answers the wrong question, otherwise known as a "Type III" error (Kimball 1957). Often, there can be a tendency to focus on the technical details and inadvertently overlook true nature of the problem. In other words, it is very important to focus on the overall strategy of the problem at hand and not just the technical tactics of the potential solution. For example, in business applications, the goal is almost always to maximize profit. When the observed outcome is categorical (e.g., purchase/no-purchase or churn/retention), it is key to tie the model performance and class predictions back to the expected profit.

A more subtle example of a problematic modeling strategy is related to *response modeling* in marketing. Recall the direct marketing example discussed in Chap. 11 where a group of customers for a clothing store were contacted with a promotion. For each customer, the *response* was recorded (i.e., whether a purchase was made).

The true goal of the clothing store is to increase profits, but this particular campaign did not sample from all of the appropriate populations. It only utilized customers who had been contacted and made the assumption that *all customers* would mimic the behavior demonstrated in this population.[1] Any model built from these data is limited to predicting the probability of a purchase *only if the customer was contacted*. This conditional statement is contrary to the goal of increasing overall profit. In fact, offering a promotion to customers who would always respond reduces profit.

For example, there is a subpopulation of customers who would make a purchase regardless of a promotional offer. These responders are likely to be in the observed data and the model would be unable to distinguish the reasons for the response. The goal of the model building exercise is to increase profits by making promotional offers to only those customers who would not have response without one.

[1] In marketing, Rzepakowski and Jaroszewicz (2012) defined the following classes of marketing models: "In the propensity models, historical information about purchases (or other success measures like visits) is used, while in the *response* models, all customers have been subject to a pilot campaign."

Siegel (2011) outlines four possible cases:

		No contact	
		Response	Non response
Contact	Response	A	B
	Non response	C	D

The cells in the table are:

- A: customers who would respond regardless of contact
- B: customers who would respond solely because of the promotion
- C: customers who have a negative response to the promotion and would have responded if they would not have been contacted
- D: customers who have absolutely no interest in responding

To increase profits, a model that accurately predicts which customers are in cell B is the most useful as this is the population associated with *new profit*.

The potential negative consequences for the simple response model are:

- The response rate is overestimated since it contains customers who always respond. The overall estimated profit is not net profit since the baseline profit is embedded. For example, a lift curve generated from a response model might indicate that by contacting 30 % of the customers, a response rate of 70 % could be achieved. There is a good chance that the customers who always respond would be scored confidently by the model and consume some percentage of the 30 % designated for contact.
- Costs are increased by sending promotions to customers in cells C or D. For the cost structure described by Larose (2006, Chap. 7), the cost of the promotion was relatively low. However, in some situations, the costs can be much higher. For example, exposing customers to unwanted promotions can have a detrimental effect on their sentiment towards the business. Also, where response models are used for customer retention, there is the possibility that the contact will *trigger* churn since it is a reminder that they might find a better deal with another company.

These issues cannot be observed until the promotion is put into action.

Techniques that attempt to understand the impacts of customer response are called *uplift modeling* (Siegel 2011; Radcliffe and Surry 2011; Rzepakowski and Jaroszewicz 2012), *true lift models* (Lo 2002), *net lift modes*, *incremental lift models*, or *true response modeling*. Lo (2002) suggests that a control group of customers who are not contacted can be used to develop a separate response model to differentiate customers in cells A and C from those in cells B and D (although B and D cannot be differentiated by the model). In conjunction with a traditional response model that is created using customers who were contacted, a scoring strategy would be to find customers with large values of

$$Pr[\text{response}|\text{contact}] - Pr[\text{response}|\text{no contact}].$$

By subtracting off the probability of those customers who respond without contact, a more refined instrument is being used to find the appropriate customer set. The drawback with this approach is that it is indirect. Also, there is a likelihood that the model predictions would be highly correlated since they are modeling very similar events. If this is the case, there may be very few customers for which the two probabilities are not similar. Radcliffe and Surry (2011) describe tree-based models where the uplift is modeled directly with a control group of customers who have not been contacted.

Another approach could be to use more sophisticated sampling techniques to create an appropriate training set. For the table above, it is impossible to contact and to not contact the same customer. However, in medical research, this problem is often faced when evaluating a new treatment against an existing therapy. Here, clinical trials sometimes use *matched samples*. Two subjects are found that are nearly identical and are randomized into treatment groups. The idea is that the only differentiating factor is the treatment, and the patient response can be estimated more accurately than without matching. The important idea here is that the subjects are no longer the experimental unit. The matched pair itself becomes the primary data point in the analysis.

The same strategy can be applied to this situation. Suppose an initial sample of customers can be matched with others who have the same attributes such as demographic factors and income levels. Within each matched pair, a promotion could be randomly assigned to one customer in the pair. Now, for each matched pair, the above 2×2 table could be created. If the results are aggregated across all matched pairs, a classification model for the four different outcomes (A though D) can be created and customers can be scored on their probability of being in class B, which is the group of interest. This approach directly models the population of interest in a single model.

20.2 Measurement Error in the Outcome

What is the minimum error rate that a model could achieve? Recall the linear regression model shown in Eq. 6.1:

$$y_i = b_0 + b_1 x_{i1} + b_2 x_{i2} + \cdots + b_p x_{ip} + e_i.$$

The residuals e_i were assume, to have some variance denoted as σ^2. If we knew the true model structure (i.e., the exact predictors and their relationship with the outcome), then σ^2 would represent the lowest possible error achievable or the *irreducible error*. However, we do not usually know the true model structure, so this value becomes inflated to include *model error* (i.e., error

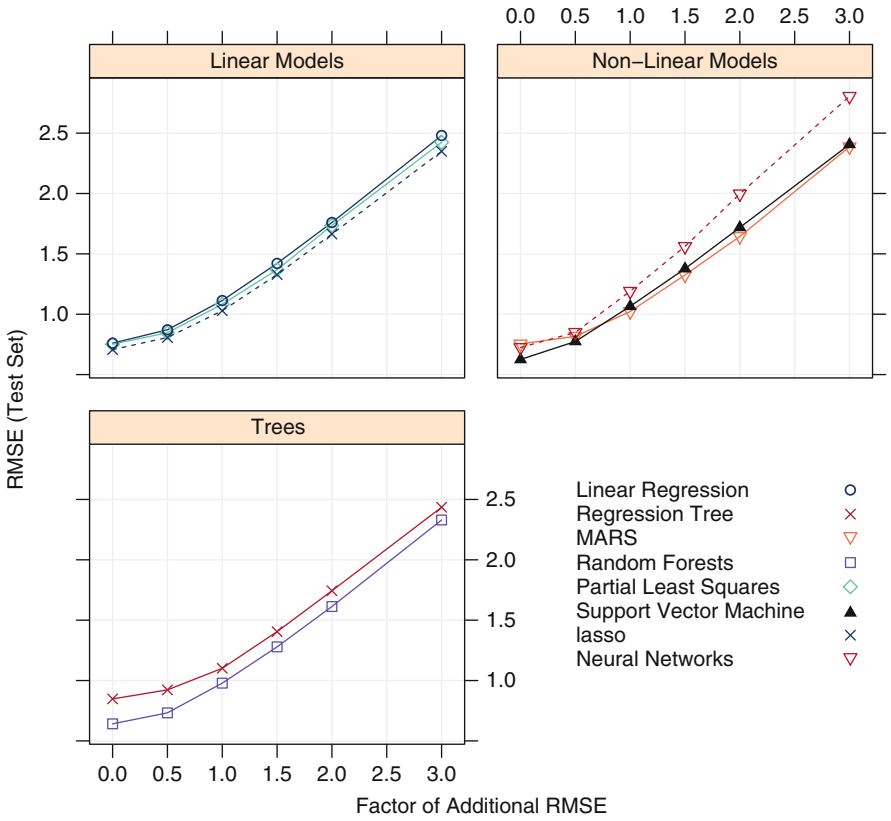

Fig. 20.1: Test set RMSE profiles for solubility models when measurement system noise increases

related to lack of fit). During the modeling process, the goal is to eliminate the model error.

However, there is another component that contributes to σ^2 that cannot be eliminated through the modeling process. If the outcome contains significant *measurement noise*, the irreducible error is increased in magnitude. The root mean squared error and R^2 then have respective lower and upper bounds due to this error. Therefore, the error term, in addition to containing the variation in the response that is not explained by the model, collects measurement system error. The better we understand the measurement system and its limits as well as the relationship between predictors and the response, the better we can foresee the limits of model performance. As mentioned in the introduction to this chapter, a similar problem occurs in classification.

To illustrate the impact of the degree of error in the response on model performance we will revisit the solubility QSAR data (Chaps. 6 through 9). The response for these data is the log of the solubility measurement (Fig. 6.2),

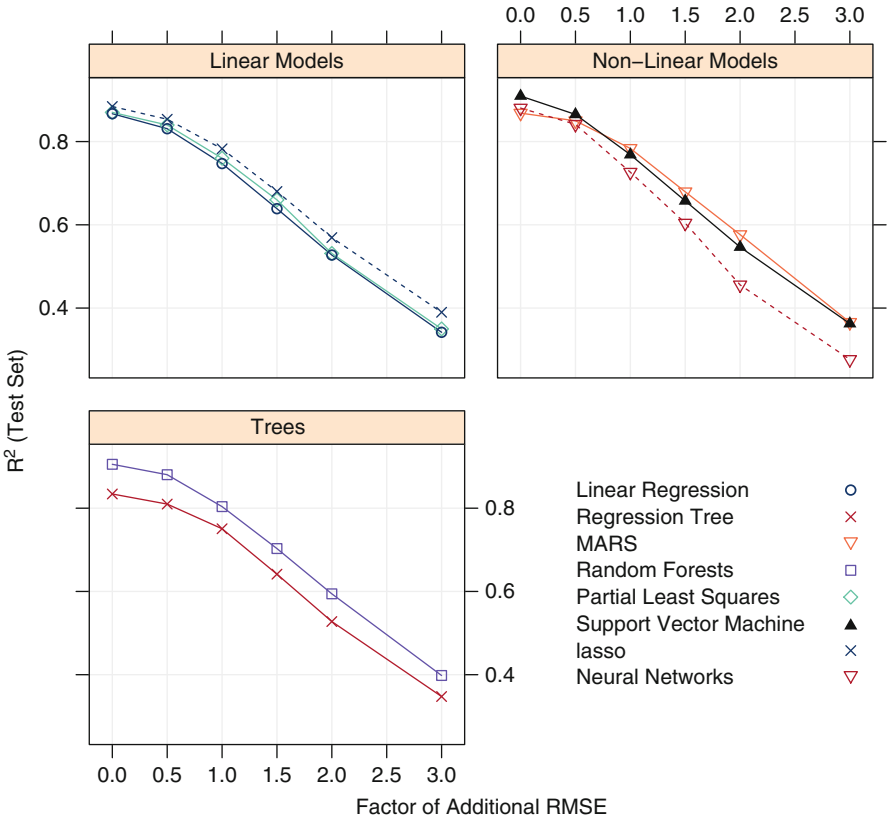

Fig. 20.2: Test set R^2 profiles for solubility models when measurement system noise increases

and the predictors were either continuous or binary (Sect. 6.1). The best linear regression model for these data had an RMSE of 0.7, which comprises the measurement system error, lack of fit error, and errors due to relevant predictors that are not included in the model. For our illustration, we will use this value as a base level for error and increase the error in the response proportionally. To do this, we have added noise to each compound's log solubility value that comes from a normal distribution with a mean of zero and a standard deviation of the linear regression model RMSE multiplied by a factor ranging from 0 to 3. For each level of additional noise, we trained and tuned the following models: linear regression, partial least squares, single regression trees, multivariate adaptive regression splines, random forests, neural networks, and radial basis function support vector machines. Performance for each of these models is evaluated based on RMSE and R^2 for the test set and the results are shown in Figs. 20.1 and 20.2. Clearly performance gets worse regardless of model type (linear models, nonlinear models, or trees)

and worsens proportionally to the degree of additional error incorporated into the response.

There are two important take-aways from this simple illustration. First, this type of noise is, noise that no model can predict—we're simply stuck with it and we cannot break through the RMSE floor or R^2 ceiling. Thus, the more the modeler knows about the measurement system, the better one can understand expectations about model performance. Second, as noise increases, the models become virtually indistinguishable in terms of their predictive performance. This means that the advantages that some of the more complex models, like ensembles, bring are only advantageous when the measurement system error is relatively low. This makes sense because the complex underlying structure that a model (such as an ensemble) can find will become more fuzzy as the noise increases. Therefore, we will likely be able to perform just as well with a simple, computationally efficient model when measurement system noise is high.

20.3 Measurement Error in the Predictors

As shown in several of the data sets, many of the predictors are calculated. For example, the solubility data contained fingerprints that indicated the presence or absence of a particular chemical structure. In text mining, predictors may be the frequency of certain important words or phrases contained in the text. In other cases, the predictors are observed or may be the product of some external process. For example:

- The cell segmentation data measured different aspects of cells, such as the area of the nucleus.
- The grant data collected information about the number of successful and unsuccessful grants.

Traditional statistical models typically assume that the predictors are measured without error, but this is not always the case. The effect of randomness in the predictors can be drastic, depending on several factors: the amount of randomness, the importances of the predictors, the type of model being used, as well as others. In some cases, if the initial data that are in the training set are generated in very controlled conditions, then the randomness can be hidden. For example, suppose data are generated by a person manually measuring an object (such as a rating). There may be differences in how people perceive the object, resulting in rater-to-rater noise. If a single rater is used for the training set data but another rate is used for new data, the bias between raters is likely to cause issues.

As another example, consider the concrete mixture data discussed previously. Even though the exact proportions and amounts of each mixture ingredient are known, there is some deviation in these values when the mixtures

are created. If the process of creating the concrete is complicated, the actual amount used may be different from the formula (although this difference is not measured or observed).

To illustrate the effect of random predictor noise on models, a simple *sin* wave was simulated with random error. Figure 20.3 shows the original data in the panel labeled "SD = 0". Here, the data on the x-axis are evenly spaced values between 2 and 10. The outcome was created by adding a small amount of normally distributed noise to the data. In each panel, the true relationship between the predictor and the response is shown as a solid black line. Two different regression models were used to fit the data. The left-hand column of panels shows an ordinary linear regression model where the true model form (i.e., $sin(x)$) is fit to the data. The right-hand column of panels corresponds to fits from a CART regression tree. The fitted curves from these models are shown as thick red lines.

The rows in Fig. 20.3 are the same data with no noise (in the top row) and when random error is incrementally added to the predictor values. Specifically, random normal values are added to the true values with different standard deviations. The small blue bell-shaped curves in this figure show the probability distribution of a data point with a mean of six and the corresponding standard deviation. The y-axis values are the same across all panels.

Linear regression is able to effectively model the true relationship with no additional noise. As noise is added to the predictors, the linear regression model begins to become poor at the apex and nadir points of the curve. The effect becomes very pronounced with a standard deviation of 0.75. However, the illustration is a somewhat optimistic assessment since it assumes that the modeler already knows the true relationship between x and y and has specified the model with this knowledge.

With no additional noise, the regression tree also approximates the pattern within the range of the observed data fairly well. The regression tree results are more problematic since the algorithm is having to determine the pattern in the data empirically. Additionally, this particular model is unstable (i.e., low bias but high variance). In this case, the difference between the predicted and true relationship begins to become more pronounced with smaller amounts of random error.

Measurement errors in the predictors can cause considerable issues when building models, especially in terms of reproducibility of the results on future data sets (similar in effect to over-fitting). In this case, future results may be poor because the underlying predictor data are different than the values used in the training set.

Case Study: Predicting Unwanted Side Effects

Pharmaceutical companies have departments for *derisking* compounds, that is, trying to detect if the candidate drug will have harmful side-effects or

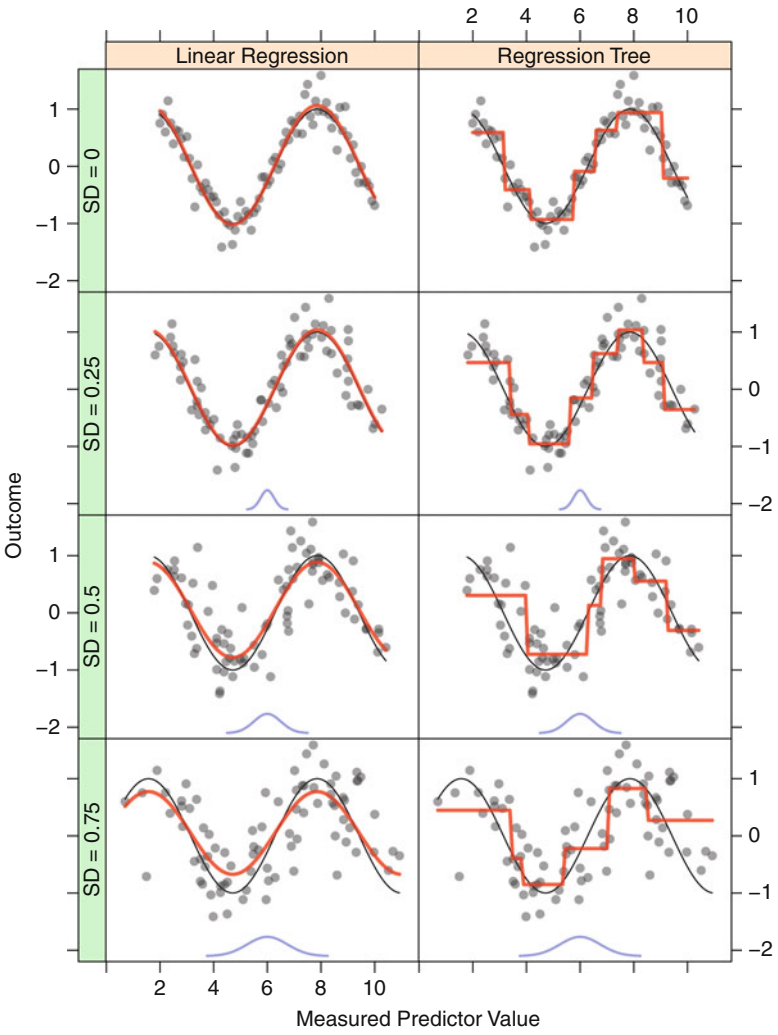

Fig. 20.3: A simulated *sin* wave and two model fits when different amounts of noise are added to the predictor values

toxicities in humans. One technique for detecting these issues is to create cell-based assays that signal if the compound is potentially dangerous. In conjunction with these lab results, a set of compounds with known issues are identified as well as a set of "clean" compounds with no known issues. Biological assays are created and used as inputs to predictive models. Much like the earlier example where solubility was predicted from chemical structures, these models use measurements of how cell lines react to the compounds to predict potential safety issues.

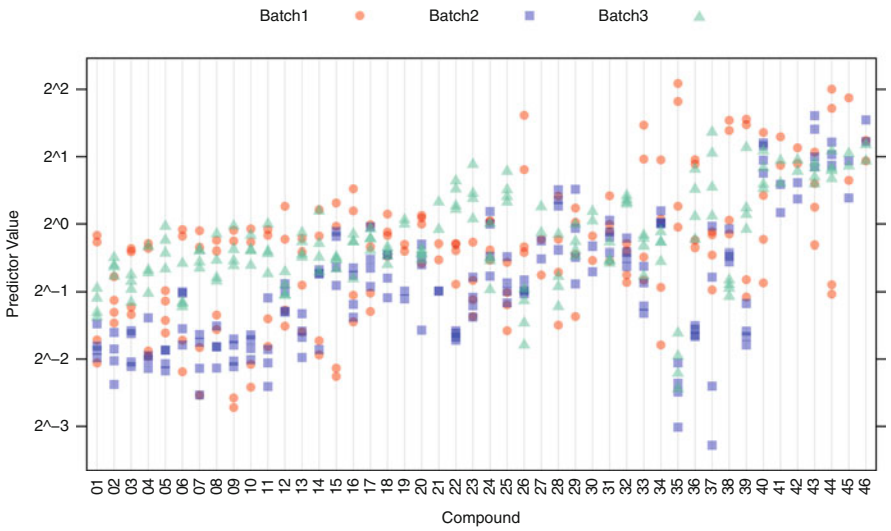

Fig. 20.4: The results of an experiment used to characterize the sources of unwanted noise in a set of biological assays

In the example presented here, the goal was to develop a predictive model for a specific toxicity. A set of about ten assays were created that measured the amount of RNA transcription (otherwise known as gene expression) in a specific type of cell. Around 250 compounds (either toxic or clean) were assayed and these results were used to train a set of models. The models resulted in sensitivities and specificities around 80 %.

However, when a new set of compounds was tested (along with several controls from the original training set), the results were no better than a random guessing. After reviewing the model building process for methodological errors, the quality of the assays was investigated.

An experiment was run where subset of 46 compounds were measured in three distinct batches (run over non-consecutive days). Within each batch, there were several replicates of each compound. A visualization of the individual data points for one of the assays is shown in Fig. 20.4, where the compounds are ordered by their average value. Clearly, there are batch-to-batch effects in the data, especially for some compounds. The trends are not uniform; in many cases, batch three had the largest values but not for every compound. Additionally, within a batch, the predictor values can range dramatically—sometimes across two logs. For example, compound 35 spans the entire range of the predictor data.

Statistical methods called variance component models (sometimes called *gauge reproducibility and repeatability* methods (Montgomery and Runger 1993)) quantify the source of the noise in the data. In this experimental design, the best possible case would be that the compound-to-compound vari-

ation in the data would account for 100 % of the noise (i.e., there would be no batch-to-batch or within-batch noise). Using a variance component analysis, the compounds accounted for about 38 % of the noise in the data. The batch-to-batch variation was 13 % of the total, and the within-batch variation was 49 %. Clearly, this is not an optimal situation since a large majority of the variation in the data is unwanted noise.[2] Since some of these compounds are toxic and others are not, it is unlikely that these measurements will help differentiate toxic and clean compounds. The other assays had similar results. It should be noted that these assays were chosen based on the data from other experiments that produced a sound biological rational for using them in this context. Therefore, we believe that there is some signal in these predictors, but it is being drowned out by the noise of the measurement system.

20.4 Discretizing Continuous Outcomes

In many fields, it may be desirable to work with a categorical response even if the original response is on a continuous scale.[3] This could be due to the fact that the underlying distribution of the response is truly bimodal. Consider Fig. 20.5 which illustrates two histograms of numeric outcomes from different data sets. The top histogram is the solubility data discussed thus far in the chapter while the histogram in the bottom of the figure represents the distribution of a measurement for another data set. While the solubility distribution is symmetric, the bottom distribution is clearly bimodal where most of the data are found at either end of the response, with relatively few in the midrange. Trying to categorize the data in the top distribution will be difficult, since there is no natural categorical distinction. Categorizing data in the bottom distribution, however, is more natural.

In other situations, the desire to work with the response on a categorical scale may be due to practical reasons. For the data that we have been examining thus far, solubility may be one of many characteristics that scientists evaluate in order to move forward in the drug discovery process. To simplify this multidimensional optimization problem, decision makers may prefer to know whether or not a compound is predicted to be *soluble enough* rather than the compound's predicted log solubility value. The selection of an optimal set of compounds can then be simplified into a checklist across the conditions of interest where the preferred compounds are the ones that satisfy the most properties of interest. While this process can grossly identify the most promising compounds for further research, more nuanced distinc-

[2] Note that, in this case, the noise in the predictors is *systematic* and random. In other words, we can attribute the source of variation to a specific cause.

[3] An example of this is the job scheduling data where the execution time of a job was binned into four groups. In this case, the queuing system cannot utilize estimates of job length but can use binned versions of this outcome.

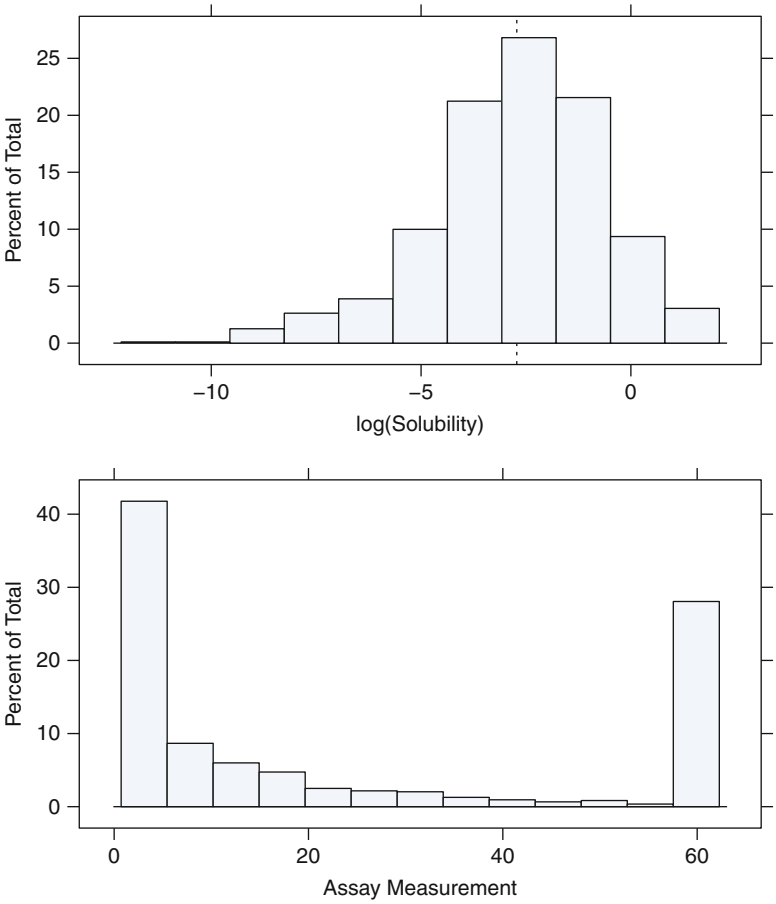

Fig. 20.5: Comparison of distributions between two data sets. *Top*: Solubility measurements on compounds previously discussed throughout this chapter. The dotted vertical line is the mean log solubility value which will be used to categorize these data. *Bottom*: Outcome values for another data set. This distribution is clearly bimodal and a categorization of data into two categories would be more natural

tions among compounds are lost because information in the outcome is being discarded.

When the response is bimodal (or multimodal), categorizing the response is appropriate. However, if the response follows a continuous distribution, as the solubility data do, then categorizing the response prior to modeling induces a loss of information, which weakens the overall utility of the model.

To illustrate this loss of information, we will continue to work with the solubility data. For this illustration, we have categorized the response as being

above or below the mean (-2.72 log units) and tuned each classification model with a couple of necessary modifications. Linear regression has been replaced with logistic regression, MARS has been replaced with FDA, and the LASSO has been replaced with glmnet, where each replacement is the parallel classification technique. After training each model, we predict the probability of each compound in the test set in addition to the Kappa value for the test set. For each test set compound we also have the predicted continuous response which we can align with the predicted probability of a compound being soluble. We can then compare the scatter plots of the continuous prediction with the probability prediction across compounds.

Figure 20.6 illustrates the test set results for the regression and classification approaches for PLS, SVM, and random forests (results from other models are similar and are presented in Exercise 20.1). For each of the regression models, the observed and predicted log(solubility) values follow a line of agreement with predicted values falling within approximately four log(solubility) units of the actual values across the range of response. Looking at the center of the distribution, a predicted log(solubility) value of -4 traces back to observed values ranging between approximately -6 and -2 log units across the models. On the other hand, a predicted probability of 0.5 for the classification models traces back to actual values ranging between approximately -6 and 0 for PLS and -4 and 0 for SVM and random forests. The range of predictions at the extremes of the classification models is even wider. For example, a predicted probability near zero for SVM corresponds to actual log(solubility) values in the range of -10 to -2, and predicted probabilities near one correspond to actual values between -3.5 and 2 (similarly for random forests). In this example, working with the data in on the original scale provides more accurate predictions across the range of response for all models. Further comparisons of model results for these data are presented in Exercise 20.1.

A second common reason for wanting to categorize a continuous response is that the scientist may believe that the continuous response contains a high degree of error, so much so that only the response values in either extreme of the distribution are likely to be correctly categorized. If this is the case, then the data can be partitioned into three categories, where data in either extreme are classified generically as positive and negative, while the data in the midrange are classified as unknown or indeterminate. The middle category can be included as such in a model (or specifically excluded from the model tuning process) to help the model more easily discriminant between the two categories.

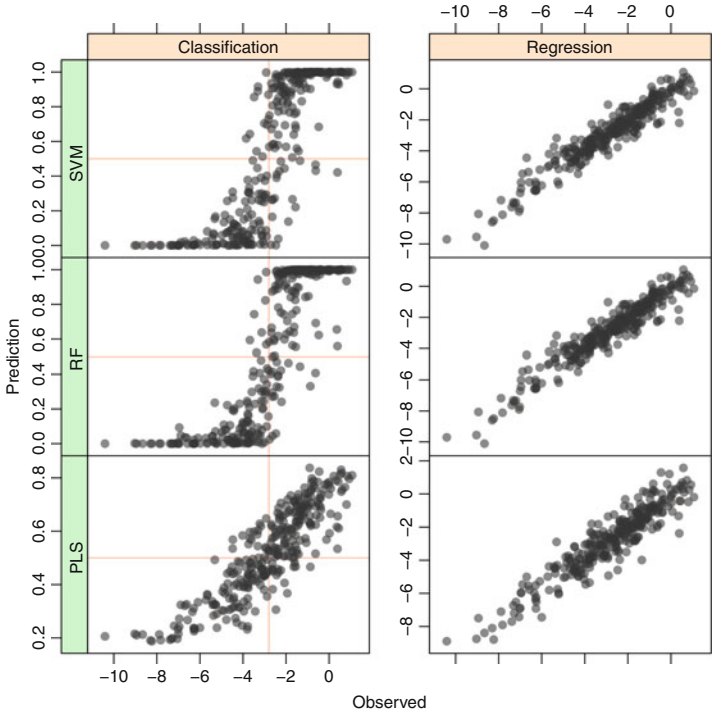

Fig. 20.6: Test set performance comparison of solubility models when response is modeled as a continuous and categorical value

20.5 When Should You Trust Your Model's Prediction?

We could view this section as "when should you *not* trust your model's prediction?" The predictive modeling process assumes that the underlying mechanism that generated the current, existing data for both the predictors and the response will continue to generate data from the same mechanism. For a simple example of a data-generating mechanism, consider the commercial food manufacturing business. Commercial food companies strive to create the same product over time so that customers can consistently have the same food tasting experience. Therefore companies keep the recipe, ingredients, and preparation process the same over time. If we were modeling moisture content of chocolate chip cookies from a commercial bakery, then we would expect that the predictors measured on a new batch of cookies would likely fall in the range as the predictors of the training set (collected at some past points in time). This means that new data will have similar characteristics and will occupy similar parts of the predictor space as the data on which the model was built.

For the examples used throughout this book, we have taken appropriate steps to create test sets that had similar properties across the predictor space as the training set. Principles underlying this process were described in Sect. 4.3. If new data are generated by the same data-generating mechanism as the training set, then we can have the confidence that the model will make sensible predictions for the new data. However, if the new data are not generated by the same mechanism, or if the training set was too small or sparse to adequately cover the range of space typically covered by the data-generating mechanism, then predictions from the model may not be trustworthy. *Extrapolation* is commonly defined as using a model to predict samples that are outside the range of the training data (Armitage and Berry 1994). It is important to recognize for high-dimensional data, however, that there may be regions within the predictors' range where no training data exist. Therefore, we need to extend our notion of extrapolation to include these vacuous interior regions of predictor space. Extrapolated predictions, whether outside the range of the predictors or within vacant regions of space, may not be trustworthy and can lead to poor decision making.

Understanding when a model is producing an extrapolated prediction can be difficult to diagnose, and the practitioner's first defense is a deep understanding of the mechanism used to generate both the training set and the new set of samples for which a prediction will be generated. Consider again the commercial chocolate chip cookie manufacturing process, and suppose that the manufacturer had changed the recipe, ingredients, and baking process. Using a previously developed model to predict moisture content for cookies from the current process may yield inaccurate predictions since the predictors for the new data are likely in a different part of space as those in the training data. If the practitioner knows this information about the data at hand, then she can use appropriate caution in applying the model and interpreting the resulting predictions.

Many times though, the practitioner does not know if the underlying data-generating mechanism is the same for the new data as the training data. Data sets with few predictors are much easier to understand and relationships between predictors in the training set and new set of samples can be examined via simple scatter plots and a comparison of distributions of each predictor. But as the dimension of the space increases, examining scatter plots and distributions is very inefficient and may not lead to a correct understanding of the predictor space between the training data and new data. In these circumstances there are a few tools that can be employed to better understand the similarity of the new data to the training data.

The *applicability domain* of a model is the region of predictor space where "the model makes predictions with a given reliability" (Netzeva et al. 2005). Different variations of this definition exist, but the most simplistic is to define the domain in terms of similarity to the training set data. If the new data being predicted are similar enough to the training set, the assumption would be that these points would, on average, have reliability that is characterized

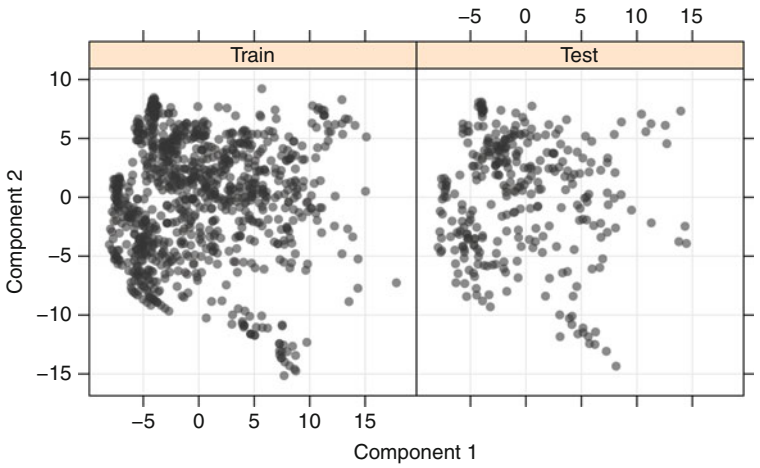

Fig. 20.7: PCA plots for the solubility data

by the model performance estimates (e.g., the accuracy rate derived from a test set).

A gross comparison of the space covered by the predictors from the training set and the new set can be made using routine dimension reduction techniques such as principal components analysis (Sect. 3.3) or multidimensional scaling (Davison 1983). If the training data and new data are generated from the same mechanism, then the projection of these data will overlap in the scatter plot. However, if the training data and new data occupy different parts of the scatter plot, then the data may not be generated by the same mechanism and predictions for the new data should be used with caution. Figure 20.7 displays the projection of the training and testing sets for the solubility data onto the first two principal components of the training data. In this case, the training and testing data appear to occupy the same space as determined by these components. Of course, this result is expected since the training set and test set were randomly partitioned from the original data set. But a further examination of similarity between training and test set samples is developed in Sect. 20.7 and Exercise 20.3.

When projecting many predictors into two dimensions, intricate predictor relationships as well as sparse and dense pockets of space can be masked. This means that while the training data and new data may appear to overlap in the projection plot, there may be regions within the predictor space where the model will inadequately predict new samples.

To address this problem, Hastie et al. (2008) describe an approach for quantifying the likelihood that a new sample is a member of the training data. In this approach, the training predictors are selected. Then a random multivariate uniform sample based on the ranges of the training predictors is

1. Compute the variable importance for the original model and identify the top 20 predictors
2. Randomly permute these predictors from the training set
3. Row-wise concatenate the original training set's top predictors and the randomly permuted version of these predictors
4. Create a classification vector that identifies the rows of the original training set and the rows of the permuted training set
5. Train a classification model on the newly created data
6. Use the classification model to predict the probability of new data being in the class of the training set.

Algorithm 20.1: Algorithm for determining similarity to the training set

generated such that it has the same number of samples as the training data. The response vector is categorical with the original training data identified as the training set and the random uniform sample identified as the random set. Any classification model can then be built on this set, and the resulting model can be used to predict the probability that a new sample is a member of the training set.

We propose two slight alterations to this method, to better address real data. First, given that many data sets contain different types of predictors, we recommend randomly permuting predictors rather than sampling from a uniform distribution across the range of predictors. By doing this, we achieve a random distribution across the entire space, while keeping within the permissible values of each predictor. Therefore categorical predictors will only receive appropriate categorical values. Second, given that data are large both in terms of predictors and dimensions, we recommend building the categorical model on a subset of the original predictors. Specifically, we suggest selecting the top 20 (or some reasonable fraction based on the context of the problem and model) important predictors from the original model of the training data. This will greatly reduce model building time, while still generating a model that can assist in assessing if the model is producing an extrapolated prediction. The steps of this process are listed in Algorithm 20.1.

To illustrate this method, we return to the two-dimensional example originally displayed in Fig. 4.1. Of course, here we omit variable selection and proceed to step 2. We augmented the original data with randomly permuted values of the original predictors (Fig. 20.8) and built a bagged classification tree. We then generated 50 random samples and moved them further away from the original data in sequential steps and used the bagged classification tree to predict the probability that the samples were from the training data.

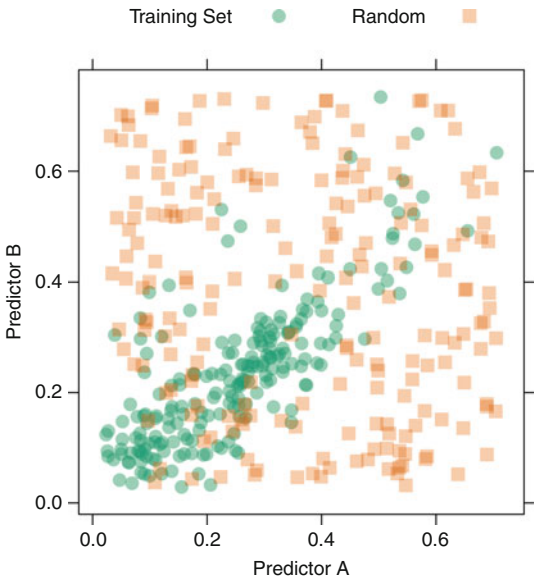

Fig. 20.8: Random uniform data have been added to the example data originally displayed in Fig. 4.1 in order to cover the range of space of the two predictors

The placement of the new samples as well as the probability of the samples being members of the training set are illustrated in Fig. 20.9. Clearly, as samples move farther from the original training data, the likelihood of training set membership decreases.

20.6 The Impact of a Large Sample

Our focus on performance thus far has been through the lens of the predictors and the response. Now we turn to the impact of the number of samples on model performance. An underlying presumption is that the more samples we have, the better model we can produce. This presumption is certainly fueled by our ability to now easily attain as many samples as we desire. As an example of the ease at which data can be obtained, consider Ayres (2007), who described the process he followed for naming his book *Super Crunchers*. To understand the reader population's opinion on his choices of titles, he used several variations of targeted Google Ads, each with a different candidate name for the book. After a short period of time, he collected a quarter of a million samples related to which ad was clicked on most. Since the ads were

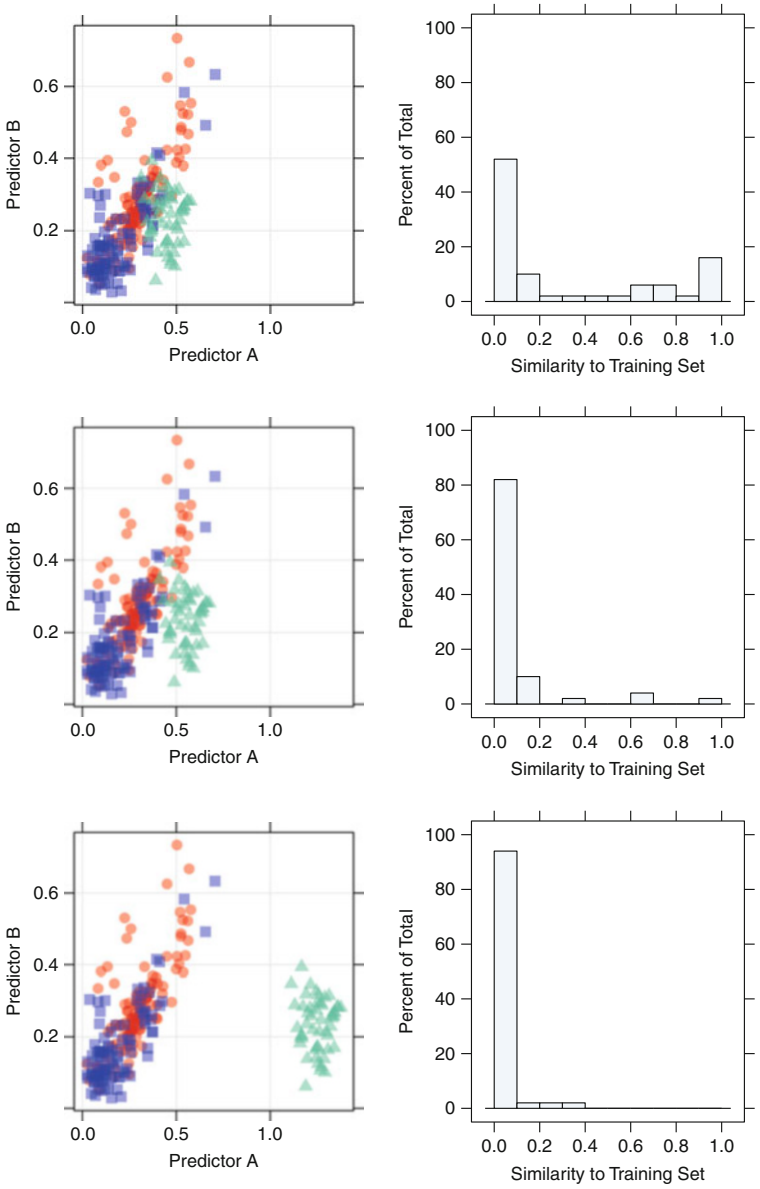

Fig. 20.9: An illustration for detecting model extrapolation. The *left column* of figures depicts three test set scenarios with the proximity ranging from near the training data (*top*) to far from the training data (*bottom*). The *right column* of figures present the probability that the test set samples are members of the original training data

served at random, this large-scale test provided strong evidence of which book name the reader population liked best.

This straightforward and clever approach to understanding a population provides a positive example of how a large number of samples can be beneficially used. But as we have seen from previous sections, noise in the predictors or the response can minimize any advantages that may be brought by an increase in the number of samples. Furthermore, an increase in the number of samples can have less positive consequences. First, many of the predictive models have significant computational burdens as the number of samples (and predictors) grows. For example, a single classification tree does many exhaustive searches across the samples and predictors to find optimal splits of the data. As the dimensions of the data increase, computation time likewise increases. Moreover the computational burden for ensembles of trees is even greater, often requiring more expensive hardware and/or special implementations of models that make the computations feasible. Second, there are diminishing returns on adding more *of the same data* from the same population. Since models stabilize with a sufficiently large number of samples, garnering more samples is less likely to change the model fit.

For example, web search technology initially used the content of a web site to rank search results. If one searched on a term such as "`predictive modeling`", the search algorithm would focus on terms in the collection of web sites in their collection and use these to predict which web pages were relevant. For this particular search term, web sites related to "`machine learning`", "`pattern recognition`", "`data mining`", and others may also be relevant and the algorithm would need to understand. For these algorithms, adding more web sites to their collection is unlikely to substantially improve the search results, but adding different data may. A web search portal like Google can also track user interaction. This leverages more direct, and higher quality, information for each search. For example, if a large proportion of users who searched for "`predictive modeling`" clicked on web site A, this should raise the likelihood that it is relevant. In this example, adding *different* data is more effective than adding more realizations of the same data attributes. In short, "Big P" usually helps more than "Big n". There are cases where adding more samples may materially improve the quality of predictions, but one should remember that *big data* may not mean *better data*.

In summary, a large number of samples can be beneficial, especially if the samples contain information throughout the predictor space, the noise in the predictors and the response can be minimized, or new content is being added. At the same time, the cost of these samples increases computational burden.

20.7 Computing

Computing details for training models discussed in this chapter can be found in the earlier sections of the book. One new computing thread presented here addresses the implementation of Algorithm 20.1. To illustrate this method, the R caret package will be referenced.

To illustrate the implementation of the similarity algorithm, first load the solubility data and define the control structure for training. Here we will use the training structure used throughout the text for these data.

```
> library(AppliedPredictiveModeling)
> data(solubility)
> set.seed(100)
> indx <- createFolds(solTrainY, returnTrain = TRUE)
> ctrl <- trainControl(method = "cv", index = indx)
```

Next, tune the desired model and compute variable importance, since the similarity algorithm can be made more efficient by working with the most important predictors. Here we tune a random forests model and create a subset of the training and test data using the top 20 predictors (step 1) for inclusion in the similarity algorithm:

```
> set.seed(100)
> mtryVals <- floor(seq(10, ncol(solTrainXtrans), length = 10))
> mtryGrid <- data.frame(.mtry = mtryVals)
> rfTune <- train(x = solTrainXtrans, y = solTrainY,
+                 method = "rf",
+                 tuneGrid = mtryGrid,
+                 ntree = 1000,
+                 importance = TRUE,
+                 trControl = ctrl)
> ImportanceOrder <- order(rfTune$finalModel$importance[,1],
+                          decreasing = TRUE)
> top20 <- rownames(rfTune$finalModel$importance[ImportanceOrder,])[1:20]
> solTrainXimp <- subset(solTrainX, select = top20)
> solTestXimp <- subset(solTestX, select = top20)
```

The subset of predictors are then permuted to create the random set. There are many ways to permute data in R; a simple and direct way is by using the `apply` and `sample` functions together. The original subset of data and permuted set are then combined and a new classification variable is created to identify each row's membership. This defines steps 2–4 of the algorithm which can be implemented as follows:

```
> permutesolTrainXimp <- apply(solTrainXimp, 2, function(x) sample(x))
> solSimX <- rbind(solTrainXimp, permutesolTrainXimp)
> groupVals <- c("Training", "Random")
> groupY <- factor(rep(groupVals, each = nrow(solTrainX)))
```

Finally, we tune a model on the newly created classification data and use the model to predict the training set membership probability.

```
> rfSolClass <- train(x = solSimX, y = groupY,
+                     method = "rf",
+                     tuneLength = 5,
+                     ntree = 1000,
+                     control = trainControl(method = "LGOCV"))
> solTestGroupProbs <- predict(rfSolClass, solTestXimp, type = "prob")
```

Exercises

20.1. Figure 20.10 provides a comparison across several models when the response is modeled as both a continuous and categorical value. The x-axis represents the observed test set solubility value, and the y-axis represents the predicted solubility value or predicted solubility probability.

(a) Based on the figure, which continuous model performs best (and worst) on the test set? Which categorical model performs best (and worst)?

(b) Examine the results from the neural net models. If the predicted probability for a compound is close to zero, what is the range of actual solubility values for the test set? If the predicted solubility value (from the continuous model) is close to -10, what is the range of actual solubility values? Which model (categorical or continuous) gives more precise results at this extreme?

(c) Are any of the categorical models better than the corresponding regression models? If yes, then explain.

20.2. As discussed in Sect. 20.4, categorizing a continuous outcome can have detrimental impacts on model performance especially when the distribution does not have distinct groupings. Sometimes, however, there may be a plausible rationale for binning continuous data. If this is the case, but the data have a distribution like the solubility data (Fig. 20.5), then a possible option for the modeler is to construct a response that partitions data into three groups for which there is higher confidence in the data at the extremes. For example, the solubility data could be partitioned into groups such as "Insoluble," "Soluble," and "Indeterminate," where the indeterminate group is the data centered around the mean of the response.

Load the solubility data and create two training sets. For the first set, split the data at the mean of the response. For the second set, define compounds to be indeterminate if they are within one standard deviation of the mean of the response. This can be done with the following code:

```
> library(AppliedPredictiveModeling)
> data(solubility)
```

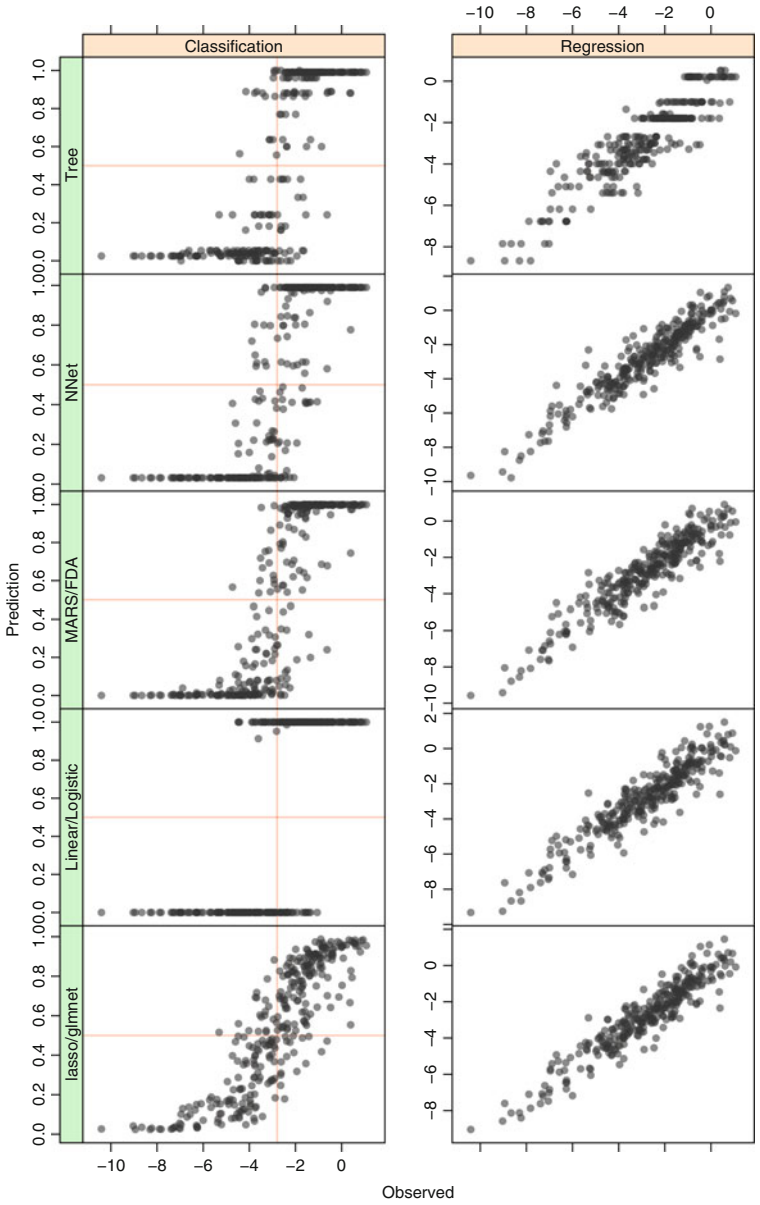

Fig. 20.10: Test set performance comparison of solubility models when the response is modeled as a continuous and categorical value

```
> trainData <- solTrainXtrans
> lowcut <- mean(solTrainY) - sd(solTrainY)
> highcut <- mean(solTrainY) + sd(solTrainY)
> breakpoints <- c(min(solTrainY), lowcut, highcut, max(solTrainY))
> groupNames <- c("Insoluble", "MidRange", "Soluble")
> solTrainY3bin <- cut(solTrainY,
+                      breaks = breakpoints,
+                      include.lowest = TRUE,
+                      labels = groupNames)
> solTestY3bin <- cut(solTestY,
+                     breaks = breakpoints,
+                     include.lowest = TRUE,
+                     labels = groupNames)
```

(a) Fit a linear model, a nonlinear model, and a tree-based model to the three-bin data. For example, the following code could be used to generate a recursive partitioning model:

```
> set.seed(100)
> indx3bin <- createFolds(solTrainY3bin, returnTrain = TRUE)
> ctrl3bin <- trainControl(method = "cv",
+                          index = indx3bin,
+                          classProbs = TRUE,
+                          savePredictions = TRUE)
> Rpart3bin <- train(x = trainXfiltered, y = solTrainY3bin,
+                    method = "rpart",
+                    metric = "Kappa",
+                    tuneLength = 30,
+                    trControl = ctrl3bin)
>
```

(b) Predict test set performance using the performance measure of your choice for each of the models in (a). Which model performs best for the three-bin data?

(c) Now exclude the "MidRange" data from the training set, rebuild each model, and predict the test set. This can be done with recursive partitioning as follows:

```
> trainXfiltered2bin <- trainXfiltered[solTrainY3bin != "MidRange",]
> solTrainY2bin <- solTrainY3bin[solTrainY3bin != "MidRange"]
> testXfiltered2bin <- testXfiltered[solTestY3bin != "MidRange",]
> solTestY2bin <- solTestY3bin[solTestY3bin != "MidRange"]
> set.seed(100)
> indx2bin <- createFolds(solTrainY2bin, returnTrain = TRUE)
> ctrl2bin <- trainControl(method = "cv",
+                          index = indx2bin,
+                          classProbs = TRUE,
+                          savePredictions = TRUE)
> Rpart2bin <- train(x = trainXfiltered2bin, y = solTrainY2bin,
+                    method = "rpart",
```

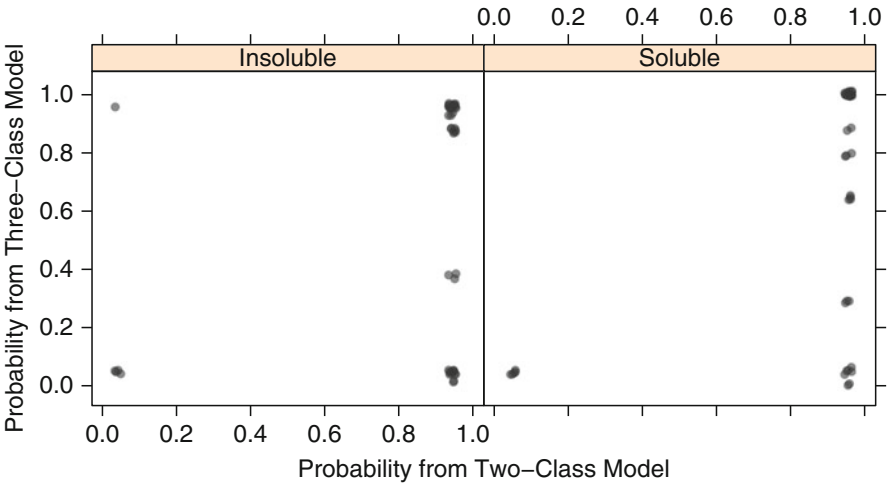

Fig. 20.11: Predicted probabilities from the two- and three-class models for the test set from the recursive partitioning model. Probabilities have been slightly jittered in order to see relative numbers of samples

```
+                    metric = "Kappa",
+                    tuneLength = 30,
+                    trControl = ctrl2bin)
> Rpart2binPred <- predict(Rpart2bin, newdata = testXfiltered)
> Rpart2binCM <- confusionMatrix(Rpart2binPred, solTestY3bin)
```

(d) How do sensitivity and specificity compare for the insoluble and soluble classes for the binning approaches in (b) and (c) within each model and between models?

(e) Figure 20.11 compares the predicted class probabilities from the two-bin and three-bin models for both the insoluble and soluble test set compounds. Based on class probabilities, does the two- or three-class model provide any distinct prediction advantages for the insoluble and/or soluble compounds? How do the predicted class probabilities within the insoluble and soluble groups compare for the other models you have developed?

20.3. Computing details for Algorithm 20.1 applied to the solubility data are presented in Sect. 20.7. Run this code and plot the distribution of the test set samples' probability of training set membership. How many test set samples are unlikely to have been part of the training set distribution?

20.4. Exercise 4.4 describes a data set in which food oils were analyzed for the content of seven types of fatty acids. Load the data and create a training and testing set as follows:

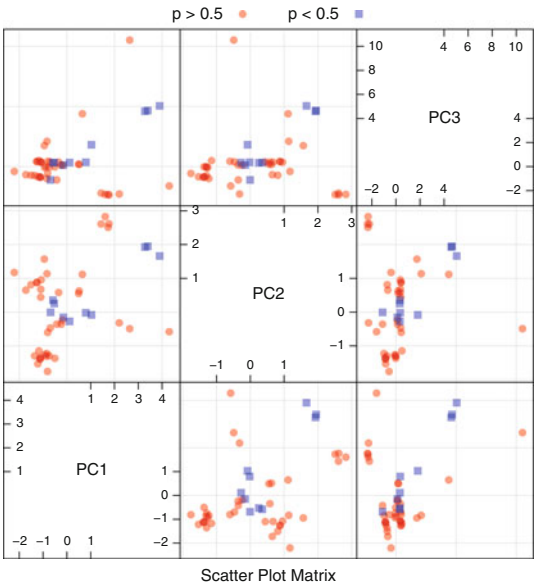

Fig. 20.12: The projection of the oil test set onto the first three principal components. Samples are colored and shaped by their probability of membership in the training set

```
> data(oil)
> set.seed(314)
> sampleRows <- sample.int(nrow(fattyAcids), size = 0.5*nrow(fattyAcids))
> fattyAcidsTrain <- fattyAcids[sampleRows,]
> fattyAcidsTest <- fattyAcids[-sampleRows,]
```

(a) Run Algorithm 20.1 using the training set. Which test set samples are not likely to be members of the training set? Why are these samples not likely to be members of the training set?

(b) Figure 20.12 presents the projections of the test set data onto the first three PCA components as determined by the training set. Samples are colored and shaped by their probability of membership in the training set. What does this plot reveal about the location of the samples that are not likely to be members of the training set?

(c) What steps could be taken to better ensure that the training and test sets cover the same region of predictor space?