Chapter 11
Measuring Performance in Classification Models

In the previous part of this book we focused on building and evaluating models for a continuous response. We now turn our focus to building and evaluating models for a categorical response. Although many of the regression modeling techniques can also be used for classification, the way we evaluate model performance is necessarily very different since metrics like RMSE and R^2 are not appropriate in the context of classification. We begin this part of the book by discussing metrics for evaluating classification model performance. In the first section of this chapter we take an in-depth look at the different aspects of classification model predictions and how these relate to the question of interest. The two subsequent sections explore strategies for evaluating classification models using statistics and visualizations.

11.1 Class Predictions

Classification models usually generate two types of predictions. Like regression models, classification models produce a continuous valued prediction, which is usually in the form of a probability (i.e., the predicted values of class membership for any individual sample are between 0 and 1 and sum to 1). In addition to a continuous prediction, classification models generate a predicted class, which comes in the form of a discrete category. For most practical applications, a discrete category prediction is required in order to make a decision. Automated spam filtering, for example, requires a definitive judgement for each e-mail.

Although classification models produce both of these types of predictions, often the focus is on the discrete prediction rather than the continuous prediction. However, the probability estimates for each class can be very useful for gauging the model's confidence about the predicted classification. Returning to the spam e-mail filter example, an e-mail message with a predicted probability of being spam of 0.51 would be classified the same as a message with

M. Kuhn and K. Johnson, *Applied Predictive Modeling*,
DOI 10.1007/978-1-4614-6849-3_11,

a predicted probability of being spam of 0.99. While both messages would be treated the same by the filter, we would have more confidence that the second message was, in fact, truly spam. As a second example, consider building a model to classify molecules by their *in-vivo* safety status (i.e., non-toxic, weakly toxic, and strongly toxic; e.g., Piersma et al. 2004). A molecule with predicted probabilities in each respective toxicity category of 0.34, 0.33, and 0.33, would be classified the same as a molecule with respective predicted probabilities of 0.98, 0.01, and 0.01. However in this case, we are much more confident that the second molecule is non-toxic as compared to the first.

In some applications, the desired outcome is the predicted class probabilities which are then used as inputs for other calculations. Consider an insurance company that wants to uncover and prosecute fraudulent claims. Using historical claims data, a classification model could be built to predict the probability of claim fraud. This probability would then be combined with the company's investigation costs and potential monetary loss to determine if pursuing the investigation is in the best financial interest of the insurance company. As another example of classification probabilities as inputs to a subsequent model, consider the customer lifetime value (CLV) calculation which is defined as the amount of profit associated with a customer over a period of time (Gupta et al. 2006). To estimate the CLV, several quantities are required, including the amount paid by a consumer over a given time frame, the cost of servicing the consumer, and the probability that the consumer will make a purchase in the time frame.

As mentioned above, most classification models generate predicted class probabilities. However, when some models are used for classification, like neural networks and partial least squares, they produce continuous predictions that do not follow the definition of a probability-the predicted values are not necessarily between 0 and 1 and do not sum to 1. For example, a partial least squares classification model (described in more detail in Sect. 12.4) would create 0/1 dummy variables for each class and simultaneously model these values as a function of the predictors. When samples are predicted, the model predictions are not guaranteed to be within 0 and 1. For classification models like these, a transformation must be used to coerce the predictions into "probability-like" values so that they can be interpreted and used for classification. One such method is the *softmax transformation* (Bridle 1990) which is defined as

$$\hat{p}_\ell^* = \frac{e^{\hat{y}_\ell}}{\sum_{l=1}^{C} e^{\hat{y}_l}}$$

where $\hat{y}_\ell$ is the numeric model prediction for the ℓ^{th} class and $\hat{p}_\ell^*$ is the transformed value between 0 and 1. Suppose that an outcome has three classes and that a PLS model predicts values of $\hat{y}_1 = 0.25$, $\hat{y}_2 = 0.76$, and $\hat{y}_3 =$-0.1. The softmax function would transform these values to $\hat{p}_1^* = 0.30$, $\hat{p}_2^* = 0.49$, and $\hat{p}_3^* = 0.21$. To be clear, no probability statement is being created by this transformation; it merely ensures that the predictions have the same mathematical qualities as probabilities.

Well-Calibrated Probabilities

Whether a classification model is used to predict spam e-mail, a molecule's toxicity status, or as inputs to insurance fraud or customer lifetime value calculations, we desire that the estimated class probabilities are reflective of the true underlying probability of the sample. That is, the predicted class probability (or probability-like value) needs to be well-calibrated. To be well-calibrated, the probabilities must effectively reflect the true likelihood of the event of interest. Returning to the spam filter illustration, if a model produces a probability or probability-like value of 20 % for the likelihood of a particular e-mail to be spam, then this value would be well-calibrated if similar types of messages would truly be from that class on average in 1 of 5 samples.

One way to assess the quality of the class probabilities is using a *calibration plot*. For a given set of data, this plot shows some measure of the observed probability of an event versus the predicted class probability. One approach for creating this visualization is to score a collection of samples with known outcomes (preferably a test set) using a classification model. The next step is to bin the data into groups based on their class probabilities. For example, a set of bins might be [0, 10 %], (10 %, 20 %], ..., (90 %, 100 %]. For each bin, determine the observed event rate. Suppose that 50 samples fell into the bin for class probabilities less than 10 % and there was a single event. The midpoint of the bin is 5 % and the observed event rate would be 2 %. The calibration plot would display the midpoint of the bin on the x-axis and the observed event rate on the y-axis. If the points fall along a 45° line, the model has produced well-calibrated probabilities.

As an illustration, a data set was simulated in a way that the true event probabilities are known. For two classes (classes 1 and 2) and two predictors (A and B), the true probability (p) of the event is generated from the equation:

$$\log\left(\frac{p}{1-p}\right) = -1 - 2A - .2A^2 + 2B^2$$

Figure 11.1 shows a simulated test set along with the a contour line for a $p = 0.50$ event probability. Two models were fit to the training set: quadratic discriminant analysis (QDA, Sect. 13.1) and a random forest model (Sect. 14.4). A test set of $n = 1000$ samples was used to score the model and create the calibration plot also shown in Fig. 11.1. Both classification models have similar accuracy for the test set (about 87.1 % for either model). The calibration plot shows that the QDA class probabilities tend to perform poorly compared to the random forest model. For example, in the bin with class probabilities ranging from 20 to 30 %, the observed percentage of events for QDA was 4.6 %, far lower than the percentage in the random forest model (35.4 %).

The class probabilities can be calibrated to more closely reflect the likelihood of the event (or, at least the likelihood seen in the actual data).

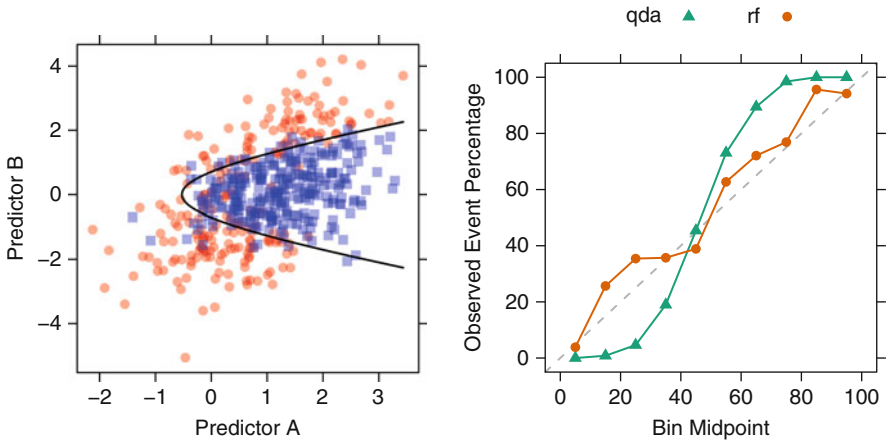

Fig. 11.1: *Left*: A simulated two-class data set with two predictors. The *solid black line* denotes the 50 % probability contour. *Right*: A calibration plot of the test set probabilities for random forest and quadratic discriminant analysis models

For example, Fig. 11.1 shows a sigmoidal pattern such that the QDA model under-predicts the event probability when the true likelihood is moderately high or low. An additional model could be created to adjust for this pattern. One equation that is consistent with this sigmoidal pattern is the logistic regression model (described in Sect. 12.2). The class predictions and true outcome values from the training set can be used to post-process the probably estimates with the following formula (Platt 2000):

$$\hat{p}^* = \frac{1}{1 + \exp\left(-\beta_0 - \beta_1 \hat{p}\right)} \tag{11.1}$$

where the β parameters are estimated by predicting the true classes as a function of the uncalibrated class probabilities ($\hat{p}$). For the QDA model, this process resulted in estimates $\hat{\beta}_0 = -5.7$ and $\hat{\beta}_1 = 11.7$. Figure 11.2 shows the results for the test set samples using this correction method. The results show improved calibration with the test set data. Alternatively, an application of Bayes' Rule (described model is Sect. 13.6) can be similarly applied to recalibrate the predictions. The Bayesian approach also improves the predictions (Fig. 11.2). Note that, after calibration, the samples must be reclassified to ensure consistency between the new probabilities and the predicted classes.

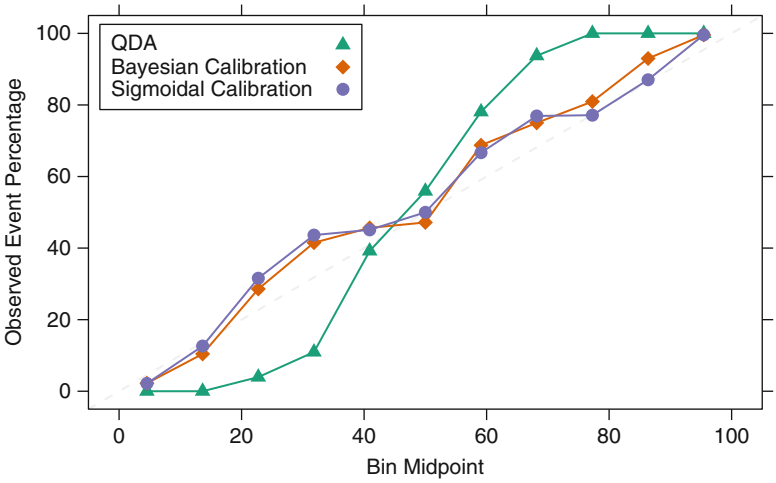

Fig. 11.2: The original QDA class probabilities and recalibrated versions using two different methodologies

Presenting Class Probabilities

Visualizations of the class probabilities are an effective method of communicating model results. For two classes, histograms of the predicted classes for each of the true outcomes illustrate the strengths and weaknesses of a model. In Chap. 4 we introduced the credit scoring example. Two classification models were created to predict the quality of a customer's credit: a support vector machine (SVM) and logistic regression. Since the performance of the two models were roughly equivalent, the logistic regression model was favored due to its simplicity. The top panel of Fig. 11.3 shows histograms of the test set probabilities for the logistic regression model (the panels indicate the true credit status). The probability of bad credit for the customers with good credit shows a skewed distribution where most customers' probabilities are quite low. In contrast, the probabilities for the customers with bad credit are flat (or uniformly distributed), reflecting the model's inability to distinguish bad credit cases.

This figure also presents a calibration plot for these data. The accuracy of the probability of bad credit degrades as it becomes larger to the point where no samples with bad credit were predicted with a probability above 82.7 %. This pattern is indicative of a model that has both poor calibration and poor performance.

When there are three or more classes, a *heat map* of the class probabilities can help gauge the confidence in the predictions. Figure 11.4 shows the test set results with eight classes (denotes `A` through `I`) and 48 samples. The

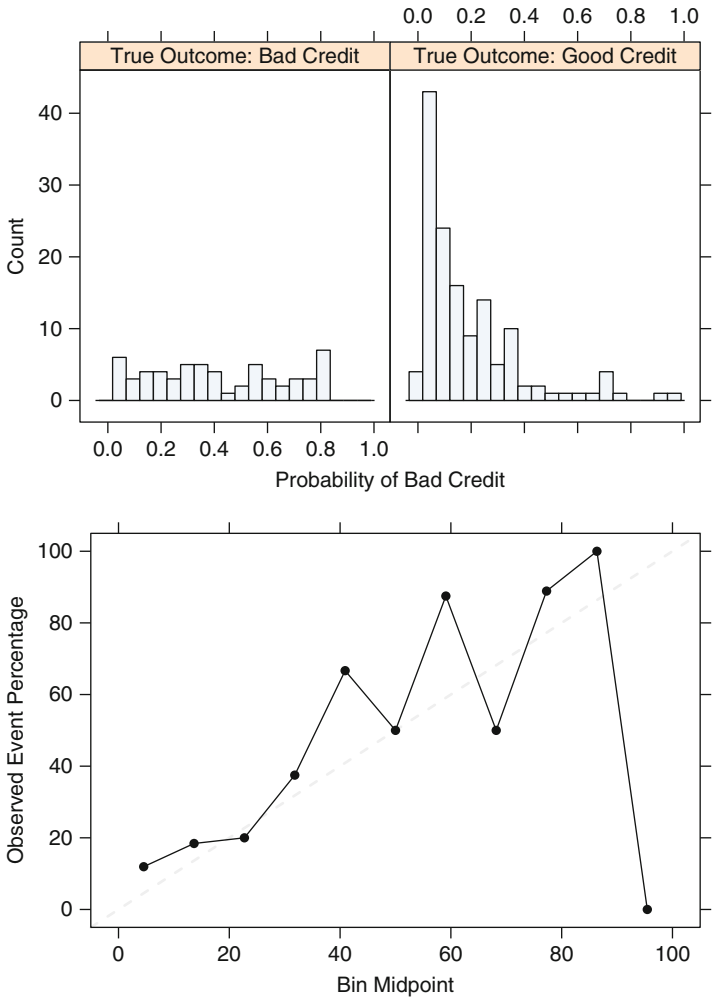

Fig. 11.3: *Top*: Histograms for a set of probabilities associated with bad credit. The two panels split the customers by their true class. *Bottom*: A calibration plot for these probabilities

true classes are shown in the rows (along with the sample identifiers) and the columns reflect the class probabilities. In some cases, such as Sample 20, there was a clear signal associated with the predicted class (the class `C` probability was 78.5 %), while in other cases, the situation is murky. Consider Sample 7. The four largest probabilities (and associated classes) were 19.6 % (`B`), 19.3 % (`C`), 17.7 % (`A`), and 15 % (`E`). While the model places the highest individual probability for this sample in the correct class, it is uncertain that it could also be of class `C`, `A`, or `E`.

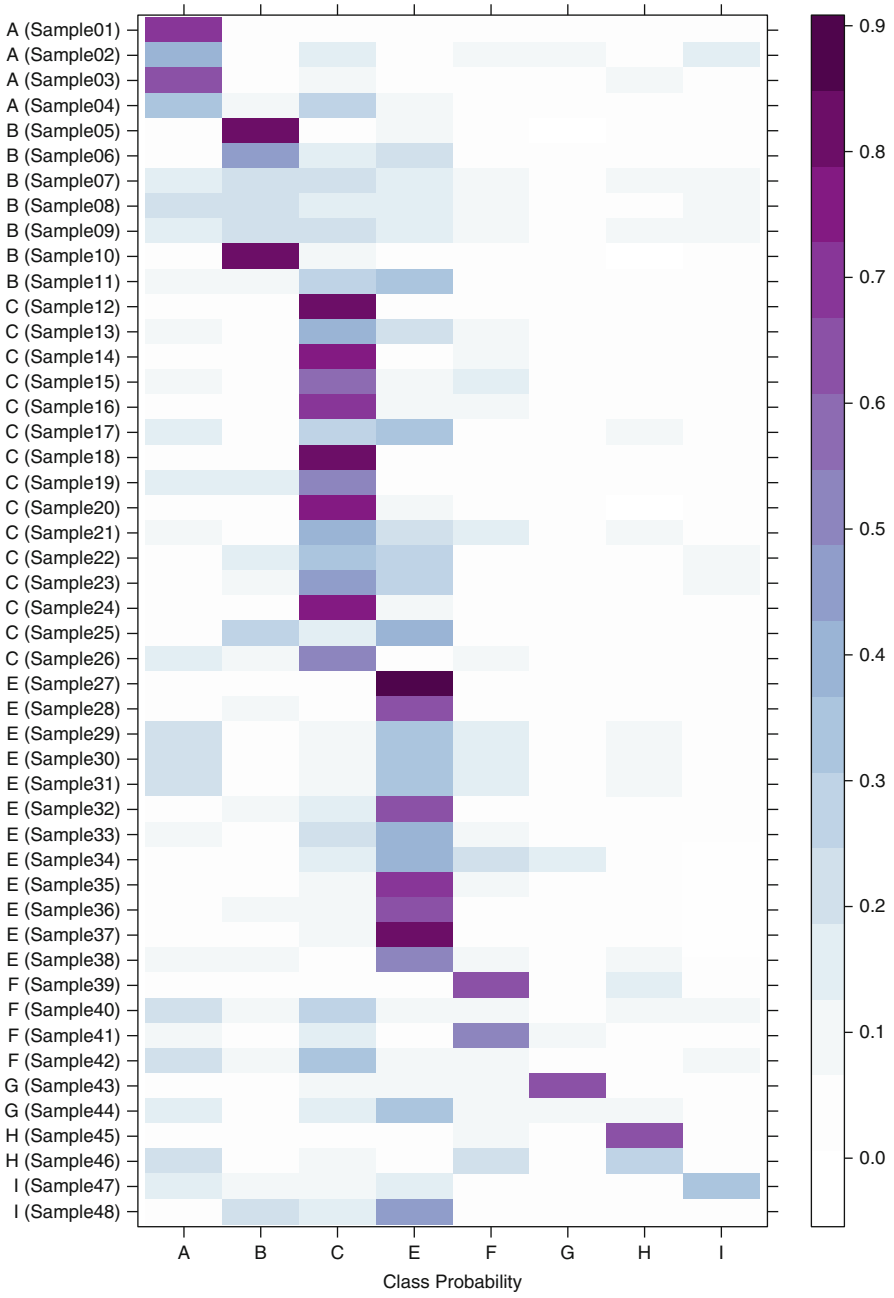

Fig. 11.4: A heat map of a test set with eight classes. The true classes are shown in the row labels while columns quantify the probabilities for each category (labeled as `A` through `I`)

Table 11.1: The confusion matrix for the two-class problem ("events" and "nonevents." The table cells indicate number of the true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN)

Predicted	Observed	
	Event	Nonevent
Event	TP	FP
Nonevent	FN	TN

Equivocal Zones

An approach to improving classification performance is to create an *equivocal* or *indeterminate zone* where the class is not formally predicted when the confidence is not high. For a two-class problem that is nearly balanced in the response, the equivocal zone could be defined as $0.50 \pm z$. If z were 0.10, then samples with prediction probabilities between 0.40 and 0.60 would be called "equivocal." In this case, model performance would be calculated excluding the samples in the indeterminate zone. The equivocal rate should also be reported with the performance so that the rate of unpredicted results is well understood. For data sets with more than 2 classes ($C > 2$), similar thresholds can be applied where the largest class probability must be larger than $(1/C) + z$ to make a definitive prediction. For the data shown in Fig. 11.4, if $(1/C) + z$ is set to 30 %, then 5 samples would be designated as equivocal.

11.2 Evaluating Predicted Classes

A common method for describing the performance of a classification model is the *confusion matrix*. This is a simple cross-tabulation of the observed and predicted classes for the data. Table 11.1 shows an example when the outcome has two classes. Diagonal cells denote cases where the classes are correctly predicted while the off-diagonals illustrate the number of errors for each possible case.

The simplest metric is the overall accuracy rate (or, for pessimists, the error rate). This reflects the agreement between the observed and predicted classes and has the most straightforward interpretation. However, there are a few disadvantages to using this statistic. First, overall accuracy counts make no distinction about the *type* of errors being made. In spam filtering, the cost of erroneous deleting an important email is likely to be higher than incorrectly allowing a spam email past a filter. In situations where the costs are different,

accuracy may not measure the important model characteristics. Provost et al. (1998) provide a comprehensive discussion of this issue, which is examined further below.

Second, one must consider the natural frequencies of each class. For example, in the USA, pregnant women routinely have blood drawn for alpha-fetoprotein testing, which attempts to detect genetic problems such as Down syndrome. Suppose the rate of this disorder[1] in fetuses is approximately 1 in 800 or about one-tenth of one percent. A predictive model can achieve almost perfect accuracy by predicting all samples to be negative for Down syndrome.

What benchmark accuracy rate should be used to determine whether a model is performing adequately? The no-information rate is the accuracy rate that can be achieved without a model. There are various ways to define this rate. For a data set with C classes, the simplest definition, based on pure randomness, is $1/C$. However, this does not take into account the relative frequencies of the classes in the training set. For the Down syndrome example, if 1,000 random samples are collected from the population who would receive the test, the expected number of positive samples would be small (perhaps 1 or 2). A model that simply predicted all samples to be negative for Down syndrome would easily surpass the no-information rate based on random guessing (50 %). An alternate definition of the no-information rate is the percentage of the largest class in the training set. Models with accuracy greater than this rate might be considered reasonable. The effect of severe class imbalances and some possible remedies are discussed in Chap. 16.

Rather than calculate the overall accuracy and compare it to the no-information rate, other metrics can be used that take into account the class distributions of the training set samples. The *Kappa statistic* (also known as Cohen's Kappa) was originally designed to assess the agreement between two raters (Cohen 1960). Kappa takes into account the accuracy that would be generated simply by chance. The form of the statistic is

$$Kappa = \frac{O - E}{1 - E}$$

where O is the observed accuracy and E is the expected accuracy based on the marginal totals of the confusion matrix. The statistic can take on values between -1 and 1; a value of 0 means there is no agreement between the observed and predicted classes, while a value of 1 indicates perfect concordance of the model prediction and the observed classes. Negative values indicate that the prediction is in the *opposite* direction of the truth, but large negative values seldom occur, if ever, when working with predictive models.[2]

[1] In medical terminology, this rate is referred to as the prevalence of a disease while in Bayesian statistics it would be the prior distribution of the event.

[2] This is true since predictive models seek to find a concordant relationship with the truth. A large negative Kappa would imply that there is relationship between the predictors and the response and the predictive model would seek to find the relationship in the correct direction.

When the class distributions are equivalent, overall accuracy and Kappa are proportional. Depending on the context, Kappa values within 0.30 to 0.50 indicate reasonable agreement. Suppose the accuracy for a model is high (90 %) but the expected accuracy is also high (85 %), the Kappa statistic would show moderate agreement (Kappa = 1/3) between the observed and predicted classes.

The Kappa statistic can also be extended to evaluate concordance in problems with more than two classes. When there is a natural ordering to the classes (e.g., "low," "medium," and "high"), an alternate form of the statistic called *weighted Kappa* can be used to enact more substantial penalties on errors that are further away from the true result. For example, a "low" sample erroneously predicted as "high" would reduce the Kappa statistic more than an error were "low" was predicted to be "medium." See (Agresti 2002) for more details.

Two-Class Problems

Consider the case where there are two classes. Table 11.1 shows the confusion matrix for generic classes "event" and "nonevent." The top row of the table corresponds to samples predicted to be events. Some are predicted correctly (the true positives, or TP) while others are inaccurately classified (false positives or FP). Similarly, the second row contains the predicted negatives with true negatives (TN) and false negatives (FN).

For two classes, there are additional statistics that may be relevant when one class is interpreted as the event of interest (such as Down syndrome in the previous example). The *sensitivity* of the model is the rate that the event of interest is predicted correctly for all samples having the event, or

$$Sensitivity = \frac{\text{\# samples with the event } \textit{and} \text{ predicted to have the event}}{\text{\# samples having the event}}$$

The sensitivity is sometimes considered the *true positive rate* since it measures the accuracy in the event population. Conversely, the *specificity* is defined as the rate that nonevent samples are predicted as nonevents, or

$$Specificity = \frac{\text{\# samples without the event } \textit{and} \text{ predicted as nonevents}}{\text{\# samples without the event}}$$

The *false-positive rate* is defined as one minus the specificity. Assuming a fixed level of accuracy for the model, there is typically a trade-off to be made between the sensitivity and specificity. Intuitively, increasing the sensitivity of a model is likely to incur a loss of specificity, since more samples are being predicted as events. Potential trade-offs between sensitivity and specificity may be appropriate when there are different penalties associated with each

Table 11.2: Test set confusion matrix for the logistic regression model training with the credit scoring data from Sect. 4.5

Predicted	Observed	
	Bad	Good
Bad	24	10
Good	36	130

type of error. In spam filtering, there is usually a focus on specificity; most people are willing to accept seeing some spam if emails from family members or coworkers are not deleted. The *receiver operating characteristic (ROC) curve* is one technique for evaluating this trade-off and is discussed in the next section.

In Chap. 4 we introduced the credit scoring example. Two classification models were created to predict the quality of a customer's credit: a SVM and logistic regression. Since the performance of the two models were roughly equivalent, the logistic regression model was favored due to its simplicity. Using the previously chosen test set of 200 customers, Table 11.2 shows the confusion matrix associated with the logistic regression model. The overall accuracy was 77 %, which is slightly better than the no-information rate of 70 %. The test set had a Kappa value of 0.375, which suggests moderate agreement. If we choose the event of interest to be a customer with bad credit, the sensitivity from this model would be estimated to be 40 % and the specificity to be 92.9 %. Clearly, the model has trouble predicting when customers have bad credit. This is likely due to the imbalance of the classes and a lack of a strong predictor for bad credit.

Often, there is interest in having a single measure that reflects the false-positive and false-negative rates. Youden's J Index (Youden 1950), which is

$$J = Sensitivity + Specificity - 1$$

measures the proportions of correctly predicted samples for both the event and nonevent groups. In some contexts, this may be an appropriate method for summarizing the magnitude of both types of errors. The most common method for combining sensitivity and specificity into a single value uses the receiver operating characteristic (ROC) curve, discussed below.

One often overlooked aspect of sensitivity and specificity is that they are *conditional* measures. Sensitivity is the accuracy rate for only the event population (and specificity for the nonevents). Using the sensitivity and specificity, the obstetrician can make statements such as "assuming that the fetus does not have Down syndrome, the test has an accuracy of 95 %." However, these statements might not be helpful to a patient since, for new samples, all that

is known is the prediction. The person using the model prediction is typically interested in *unconditional* queries such as "what are the chances that the fetus has the genetic disorder?" This depends on three values: the sensitivity and specificity of the diagnostic test and the prevalence of the event in the population. Intuitively, if the event is rare, this should be reflected in the answer. Taking the prevalence into account, the analog to sensitivity is the *positive predicted value*, and the analog to specificity is the *negative predicted value*. These values make unconditional evaluations of the data.[3] The positive predicted value answers the question "what is the probability that this sample is an event?" The formulas are

$$PPV = \frac{Sensitivity \times Prevalence}{(Sensitivity \times Prevalence) + ((1 - Specificity) \times (1 - Prevalence))}$$

$$NPV = \frac{Specificity \times (1 - Prevalence)}{(Prevalence \times (1 - Sensitivity)) + (Specificity \times (1 - Prevalence))}$$

Clearly, the predictive values are nontrivial combinations of performance and the rate of events. The top panel in Fig. 11.5 shows the effect of prevalence on the predictive values when the model has a specificity of 95 % and a sensitivity of either 90 % or 99 %. Large negative predictive values can be achieved when the prevalence is low. However, as the event rate becomes high, the negative predictive value becomes very small. The opposite is true for the positive predictive values. This figure also shows that a sizable difference in sensitivity (90 % versus 99 %) has little effect on the positive predictive values.

The lower panel of Fig. 11.5 shows the positive predictive value as a function of sensitivity and specificity when the event rate is balanced (50 %). In this case, the positive predicted value would be

$$PPV = \frac{Sensitivity}{Sensitivity(1 - Specificity)} = \frac{TP}{TP + FP}$$

This figure also shows that the value of the sensitivity has a smaller effect than specificity. For example, if specificity is high, say $\geq$90 %, a large positive predicted value can be achieved across a wide range of sensitivities.

Predictive values are not often used to characterize the model. There are several reasons why, most of which are related to prevalence. First, prevalence is hard to quantify. Our experience is that very few people, even experts, are willing to propose an estimate of this quantity based on prior knowledge. Also, the prevalence is dynamic. For example, the rate of spam emails increases when new schemes are invented but later fall off to baseline levels. For medical diagnoses, the prevalence of diseases can vary greatly depend-

[3] In relation to Bayesian statistics, the sensitivity and specificity are the conditional probabilities, the prevalence is the prior, and the positive/negative predicted values are the posterior probabilities.

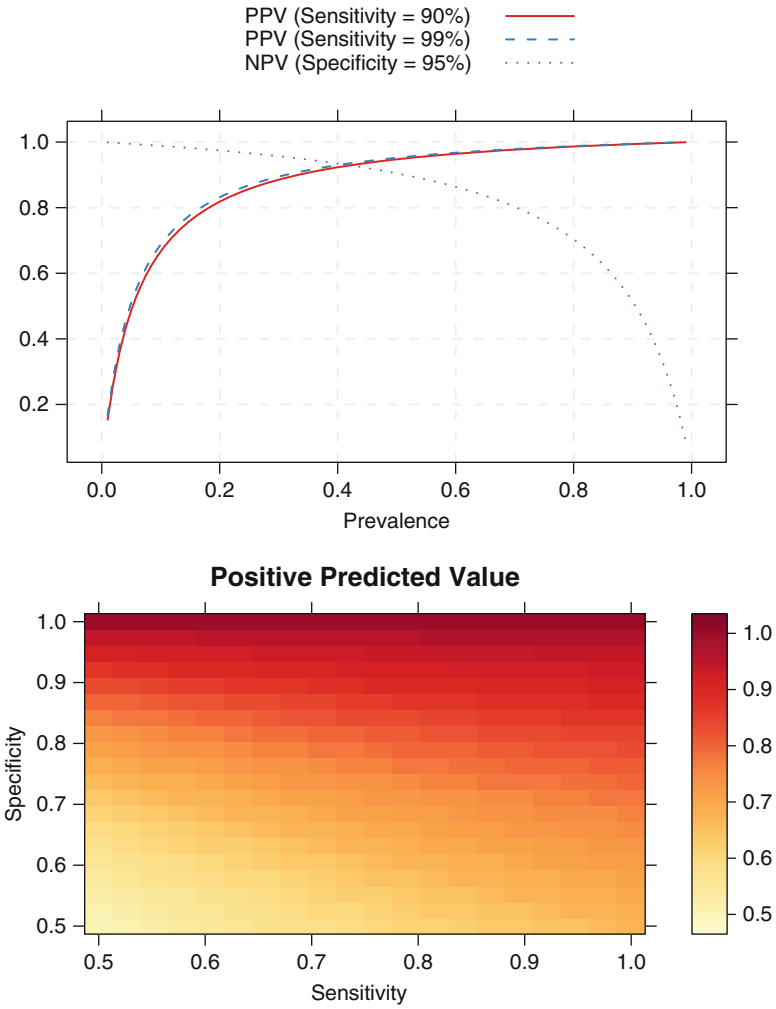

Fig. 11.5: *Top*: The effect of prevalence on the positive and negative predictive values. The PPV was computed using a specificity of 95 % and two values of sensitivity. The NPV was computed with 90 % sensitivity and 95 % specificity. *Bottom*: For a fixed prevalence of 50 %, positive predictive values are shown as a function of sensitivity and specificity

ing on the geographic location (e.g., urban versus rural). For example, in a multicenter clinical trial of a diagnostic test for *Neisseria gonorrhoeae*, the prevalence within the patient population varied from 0 % to 42.9 % across nine clinical sites (Becton Dickinson and Company 1991).

Table 11.3: The confusion matrix and profit costs/benefits for the direct mailing example of Larose (2006)

Predicted	Observed		Observed	
	Response	Nonresponse	Response	Nonresponse
Response	TP	FP	\$26.40	−\$2.00
Nonresponse	FN	TN	−\$28.40	–

Non-Accuracy-Based Criteria

For many commercial applications of predictive models, accuracy is not the primary goal for the model. Often, the purpose of the model might be to:

- Predict investment opportunities that maximize return
- Improve customer satisfaction by market segmentation
- Lower inventory costs by improving product demand forecasts or
- Reduce costs associated with fraudulent transactions

While accuracy is important, it only describes how well the model predicts the data. If the model is *fit for purpose*, other more direct metrics of performance should be considered. These metrics quantify the consequences of correct and incorrect predictions (i.e., the benefits and costs). For example, in fraud detection, a model might be used to quantify the likelihood that a transaction is fraudulent. Suppose that fraud is the event of interest. Any model predictions of fraud (correct or not) have an associated cost for a more in-depth review of the case. For true positives, there is also a quantifiable benefit to catching bad transactions. Likewise, a false negative results in a loss of income.

Consider the direct marketing application in Larose (2006, Chap. 7) where a clothing company is interested in offering promotions by mail to its customers. Using existing customer data on shopping habits, they wish to predict who would respond (i.e., the two classes and "responders" and "nonresponders"). The 2×2 table of possible outcomes is shown in Table 11.3 where the type of decisions is presented on the left and the revenue or cost per decision is on the right. For example, if the model were to accurately predict a responder, the average profit when the customer responds to the promotion is estimated to be \$28.40. There is a small \$2.00 cost for mailing the promotion, so the net profit of a correct decision is \$26.40. If we inaccurately predict that a customer will respond (a false positive), the only loss is the cost of the promotion (\$2.00).

Table 11.4: Left: A hypothetical test confusion matrix for a predictive model with a sensitivity of 75 % and a specificity of 94.4 %. Right: The confusion matrix when a mass mailing is used for all customers

Predicted	Observed		Observed	
	Response	Nonresponse	Response	Nonresponse
Response	1,500	1,000	2,000	18,000
Nonresponse	500	17,000	0	0

If the model accurately predicts a nonresponse, there is no gain or loss since they would not have made a purchase and the mailer was not sent.[4] However, incorrectly predicting that a true responder would not respond means that a potential $28.40 was lost, so this is the cost of a false-negative. The total profit for a particular model is then

$$profit = \$26.40TP - \$2.00FP - \$28.40FN \tag{11.2}$$

However, the prevalence of the classes should be taken into account. The response rate in direct marketing is often very low (Ling and Li 1998) so the expected profit for a given marketing application may be driven by the false-negative costs since this value is likely to be larger than the other two in Eq. 11.2.

Table 11.4 shows hypothetical confusion matrices for 20,000 customers with a 10 % response rate. The table on the left is the result of a predicted model with a sensitivity of 75 % and a specificity of 94.4 %. The total profit would be $23,400 or $1.17 per customer. Suppose another model had the same sensitivity but 100 % specificity. In this case, the total profit would increase to $25,400, a marginal gain given a significant increase in model performance (mostly due to the low cost of mailing the promotion).

The right side of Table 11.4 shows the results when a mass mailing for all the customers is used. This approach has perfect sensitivity and the worst possible specificity. Here, due to the low costs, the profit is $16,800 or $0.84 per customer. This should be considered the baseline performance for any predictive model to beat. The models could alternatively be characterized using the profit *gain* or *lift*, estimated as the model profit above and beyond the profit from a mass mailing.

With two classes, a general outline for incorporating unequal costs with performance measures is given by Drummond and Holte (2000). They define the probability-cost function (PCF) as

[4] This depends on a few assumptions which may or may not be true. Section 20.1 discusses this aspect of the example in more detail in the context of *net lift modeling*.

$$PCF = \frac{P \times C(+|-)}{P \times C(-|+) + (1-P) \times C(+|-)}$$

where P is the (prior) probability of the event, $C(-|+)$ is the cost associated with incorrectly predicting an event $(+)$ as a nonevent, and $C(+|-)$ is the cost of incorrectly predicting a nonevent. The PCF is the proportion of the total costs associated with a false-positive sample. They suggest using the normalized expected cost (NEC) function to characterize the model

$$NEC = PCF \times (1 - TP) + (1 - PCF) \times FP$$

for a specific set of costs. Essentially, the NEC takes into account the prevalence of the event, model performance, and the costs and scales the total cost to be between 0 and 1. Note that this approach only assigns costs to the two types of errors and might not be appropriate for problems where there are other cost or benefits (such as the direct marketing costs shown in Table 11.3).

11.3 Evaluating Class Probabilities

Class probabilities potentially offer more information about model predictions than the simple class value. This section discussed several approaches to using the probabilities to compare models.

Receiver Operating Characteristic (ROC) Curves

ROC curves (Altman and Bland 1994; Brown and Davis 2006; Fawcett 2006) were designed as a general method that, given a collection of continuous data points, determine an effective threshold such that values above the threshold are indicative of a specific event. This tool will be examined in this context in Chap. 19, but here, we describe how the ROC curve can be used for determining alternate cutoffs for class probabilities.

For the credit model test set previously discussed, the sensitivity was poor for the logistic regression model (40 %), while the specificity was fairly high (92.9 %). These values were calculated from classes that were determined with the default 50 % probability threshold. Can we improve the sensitivity by lowering the threshold[5] to capture more true positives? Lowering the threshold for classifying bad credit to 30 % results in a model with improved sensi-

[5] In this analysis, we have used the test set to investigate the effects of alternative thresholds. Generally, a new threshold should be derived from a separate data set than those used to train the model or evaluate performance.

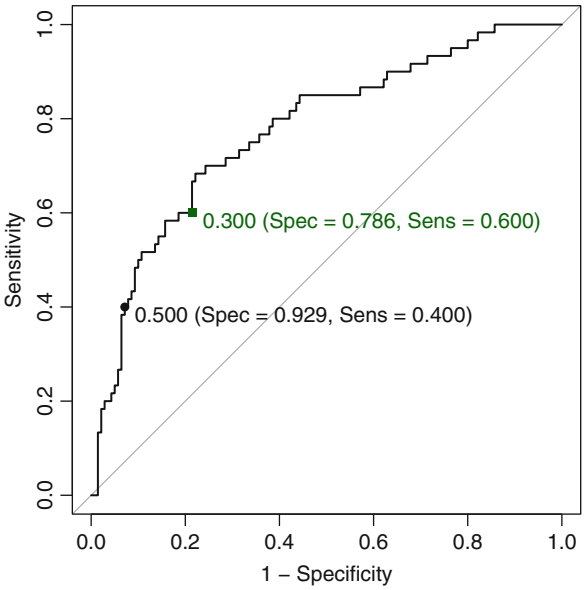

Fig. 11.6: A receiver operator characteristic (ROC) curve for the logistic regression model results for the credit model. The dot indicates the value corresponding to a cutoff of 50 % while the green square corresponds to a cutoff of 30 % (i.e., probabilities greater than 0.30 are called events)

tivity (60 %) but decrease specificity (79.3 %). Referring to Fig. 11.3, we see that decreasing the threshold begins to capture more of the customers with bad credit but also begins to encroach on the bulk of the customers with good credit.

The ROC curve is created by evaluating the class probabilities for the model across a continuum of thresholds. For each candidate threshold, the resulting true-positive rate (i.e., the sensitivity) and the false-positive rate (one minus the specificity) are plotted against each other. Figure 11.6 shows the results of this process for the credit data. The solid black point is the default 50 % threshold while the green square corresponds to the performance characteristics for a threshold of 30 %. In this figure, the numbers in parentheses are (*specificity*, *sensitivity*). Note that the trajectory of the curve between (0, 0) and the 50 % threshold is steep, indicating that the sensitivity is increasing at a greater rate than the decrease in specificity. However, when the sensitivity is greater than 70 %, there is a more significant decrease in specificity than the gain in sensitivity.

This plot is a helpful tool for choosing a threshold that appropriately maximizes the trade-off between sensitivity and specificity. However, altering the threshold only has the effect of making samples more positive (or negative

as the case may be). In the confusion matrix, it cannot move samples out of *both* off-diagonal table cells. There is almost always a decrease in either sensitivity or specificity as 1 is increased.

The ROC curve can also be used for a quantitative assessment of the model. A perfect model that completely separates the two classes would have 100 % sensitivity and specificity. Graphically, the ROC curve would be a single step between (0, 0) and (0, 1) and remain constant from (0, 1) to (1, 1). The area under the ROC curve for such a model would be one. A completely ineffective model would result in an ROC curve that closely follows the 45° diagonal line and would have an area under the ROC curve of approximately 0.50. To visually compare different models, their ROC curves can be superimposed on the same graph. Comparing ROC curves can be useful in contrasting two or more models with different predictor sets (for the same model), different tuning parameters (i.e., within model comparisons), or complete different classifiers (i.e., between models).

The optimal model should be shifted towards the upper left corner of the plot. Alternatively, the model with the largest area under the ROC curve would be the most effective. For the credit data, the logistic model had an estimated area under the ROC curve of 0.78 with a 95 % confidence interval of (0.7, 0.85) determined using the bootstrap confidence interval method (Hall et al. 2004). There is a considerable amount of research on methods to formally compare multiple ROC curves. See Hanley and McNeil (1982), DeLong et al. (1988), Venkatraman (2000), and Pepe et al. (2009) for more information.

One advantage of using ROC curves to characterize models is that, since it is a function of sensitivity and specificity, the curve is insensitive to disparities in the class proportions (Provost et al. 1998; Fawcett 2006). A disadvantage of using the area under the curve to evaluate models is that it obscures information. For example, when comparing models, it is common that no individual ROC curve is uniformly better than another (i.e., the curves cross). By summarizing these curves, there is a loss of information, especially if one particular area of the curve is of interest. For example, one model may produce a steep ROC curve slope on the left but have a lower AUC than another model. If the lower end of the ROC curve was of primary interest, then AUC would not identify the best model. The partial area under the ROC curve (McClish 1989) is an alternative that focuses on specific parts of the curve.

The ROC curve is only defined for two-class problems but has been extended to handle three or more classes. Hand and Till (2001), Lachiche and Flach (2003), and Li and Fine (2008) use different approaches extending the definition of the ROC curve with more than two classes.

Lift Charts

Lift charts (Ling and Li 1998) are a visualization tool for assessing the ability of a model to detect events in a data set with two classes. Suppose a group of samples with M events is *scored* using the event class probability. When ordered by the class probability, one would hope that the events are ranked higher than the nonevents. Lift charts do just this: rank the samples by their scores and determine the cumulative event rate as more samples are evaluated. In the optimal case, the M highest-ranked samples would contain all M events. When the model is non-informative, the highest-ranked $X\,\%$ of the data would contain, on average, X events. The *lift* is the number of samples detected by a model above a completely random selection of samples.

To construct the *lift chart* we would take the following steps:

1. Predict a set of samples that were not used in the model building process but have known outcomes.
2. Determine the *baseline* event rate, i.e., the percent of true events in the entire data set.
3. Order the data by the classification probability of the event of interest.
4. For each unique class probability value, calculate the percent of true events in all samples below the probability value.
5. Divide the percent of true events for each probability threshold by the baseline event rate.

The lift chart plots the cumulative gain/lift against the cumulative percentage of samples that have been screened. Figure 11.7 shows the best and worse case lift curves for a data set with a 50 % event rate. The non-informative model has a curve that is close to the 45° reference line, meaning that the model has no benefit for ranking samples. The other curve is indicative of a model that can perfectly separate two classes. At the 50 % point on the x-axis, all of the events have been captured by the model.

Like ROC curves, the lift curves for different models can be compared to find the most appropriate model and the area under the curve can be used as a quantitative measure of performance. Also like ROC curves, some parts of the lift curve are of more interest than others. For example, the section of the curve associated with the highest-ranked samples should have an enriched true-positive rate and is likely to be the most important part of the curve.

Consider the direct marketing application. Using this curve, a quasi-threshold can be determined for a model. Again, suppose there is a 10 % response rate and that most of the responders are found in the top 7 % of model predictions. Sending the promotions to this subset of customers effectively imposes a new threshold for customer response since samples below the threshold will not be acted on.

In this application, recall that a predictive model would have to generate more profit than the baseline profit associated with sending the promotion

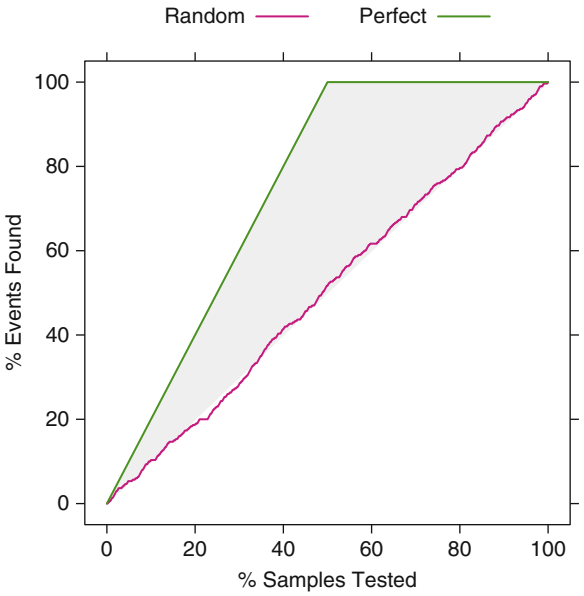

Fig. 11.7: An example lift plot with two models: one that perfectly separates two classes and another that is completely non-informative

to all customers. Using the lift plot, the expected profit can be calculated for each point on the curve to determine if the lift is sufficient to beat the baseline profit.

11.4 Computing

The R packages AppliedPredictiveModeling, caret, klaR, MASS, pROC, and randomForest will be utilized in this section.

For illustration, the simulated data set shown in Fig. 11.1 will be used in this section. To create these data, the `quadBoundaryFunc` function in the AppliedPredictiveModeling package is used to generate the predictors and outcomes:

```
> library(AppliedPredictiveModeling)
> set.seed(975)
> simulatedTrain <- quadBoundaryFunc(500)
> simulatedTest <- quadBoundaryFunc(1000)
> head(simulatedTrain)
           X1          X2      prob  class
1  2.4685709  2.28742015 0.9647251 Class1
2 -0.1889407 -1.63949455 0.9913938 Class1
```

```
  3 -1.9101460 -2.89194964 1.0000000 Class1
  4  0.3481279  0.06707434 0.1529697 Class1
  5  0.1401153  0.86900555 0.5563062 Class1
  6  0.7717148 -0.91504835 0.2713248 Class2
```

The random forest and quadratic discriminant models will be fit to the data:

```
> library(randomForest)
> rfModel <- randomForest(class ~ X1 + X2,
+                         data = simulatedTrain,
+                         ntree = 2000)
> library(MASS) ## for the qda() function
> qdaModel <- qda(class ~ X1 + X2, data = simulatedTrain)
```

The output of the `predict` function for `qda` objects includes both the predicted classes (in a slot called `class`) and the associated probabilities are in a matrix called `posterior`. For the QDA model, predictions will be created for the training and test sets. Later in this section, the training set probabilities will be used in an additional model to calibrate the class probabilities. The calibration will then be applied to the test set probabilities:

```
> qdaTrainPred <- predict(qdaModel, simulatedTrain)
> names(qdaTrainPred)
  [1] "class"     "posterior"
> head(qdaTrainPred$class)
  [1] Class1 Class1 Class1 Class2 Class1 Class2
  Levels: Class1 Class2
> head(qdaTrainPred$posterior)
       Class1      Class2
  1 0.7313136 0.268686374
  2 0.8083861 0.191613899
  3 0.9985019 0.001498068
  4 0.3549247 0.645075330
  5 0.5264952 0.473504846
  6 0.3604055 0.639594534
> qdaTestPred <- predict(qdaModel, simulatedTest)
> simulatedTrain$QDAprob <- qdaTrainPred$posterior[,"Class1"]
> simulatedTest$QDAprob <- qdaTestPred$posterior[,"Class1"]
```

The random forest model requires two calls to the `predict` function to get the predicted classes and the class probabilities:

```
> rfTestPred <- predict(rfModel, simulatedTest, type = "prob")
> head(rfTestPred)
    Class1 Class2
  1 0.4300 0.5700
  2 0.5185 0.4815
  3 0.9970 0.0030
  4 0.9395 0.0605
  5 0.0205 0.9795
  6 0.2840 0.7160
> simulatedTest$RFprob <- rfTestPred[,"Class1"]
> simulatedTest$RFclass <- predict(rfModel, simulatedTest)
```

Sensitivity and Specificity

caret has functions for computing sensitivity and specificity. These functions require the user to indicate the role of each of the classes:

```
> # Class 1 will be used as the event of interest
> sensitivity(data = simulatedTest$RFclass,
+             reference = simulatedTest$class,
+             positive = "Class1")
  [1] 0.8278867
> specificity(data = simulatedTest$RFclass,
+             reference = simulatedTest$class,
+             negative = "Class2")
  [1] 0.8946396
```

Predictive values can also be computed either by using the prevalence found in the data set (46 %) or by using prior judgement:

```
> posPredValue(data = simulatedTest$RFclass,
+              reference = simulatedTest$class,
+              positive = "Class1")
  [1] 0.8695652
> negPredValue(data = simulatedTest$RFclass,
+              reference = simulatedTest$class,
+              positive = "Class2")
  [1] 0.8596803
> # Change the prevalence manually
> posPredValue(data = simulatedTest$RFclass,
+              reference = simulatedTest$class,
+              positive = "Class1",
+              prevalence = .9)
  [1] 0.9860567
```

Confusion Matrix

There are several functions in R to create the confusion matrix. The `confusionMatrix` function in the caret package produces the table and associated statistics:

```
> confusionMatrix(data = simulatedTest$RFclass,
+                 reference = simulatedTest$class,
+                 positive = "Class1")

  Confusion Matrix and Statistics

            Reference
  Prediction Class1 Class2
      Class1    380     57
      Class2     79    484
```

```
               Accuracy : 0.864
                 95% CI : (0.8412, 0.8846)
    No Information Rate : 0.541
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.7252
 Mcnemar's Test P-Value : 0.07174

            Sensitivity : 0.8279
            Specificity : 0.8946
         Pos Pred Value : 0.8696
         Neg Pred Value : 0.8597
             Prevalence : 0.4590
         Detection Rate : 0.3800
   Detection Prevalence : 0.4370

       'Positive' Class : Class1
```

There is also an option in this function to manually set the prevalence. If there were more than two classes, the sensitivity, specificity, and similar statistics are calculated on a "one-versus-all" basis (e.g., the first class versus a pool of classes two and three).

Receiver Operating Characteristic Curves

The pROC package (Robin et al. 2011) can create the curve and derive various statistics.[6] First, an R object must be created that contains the relevant information using the pROC function `roc`. The resulting object is then used to generate the ROC curve or calculate the area under the curve. For example,

```
> library(pROC)
> rocCurve <-  roc(response = simulatedTest$class,
+                  predictor = simulatedTest$RFprob,
+                  ## This function assumes that the second
+                  ## class is the event of interest, so we
+                  ## reverse the labels.
+                  levels = rev(levels(simulatedTest$class)))
```

From this object, we can produce statistics (such as the area under the ROC curve and its confidence interval):

```
> auc(rocCurve)
  Area under the curve: 0.9328
> ci.roc(rocCurve)
  95% CI: 0.9176-0.948 (DeLong)
```

[6] R has a number of packages that can compute the ROC curve, including ROCR, caTools, PresenceAbsence, and others.

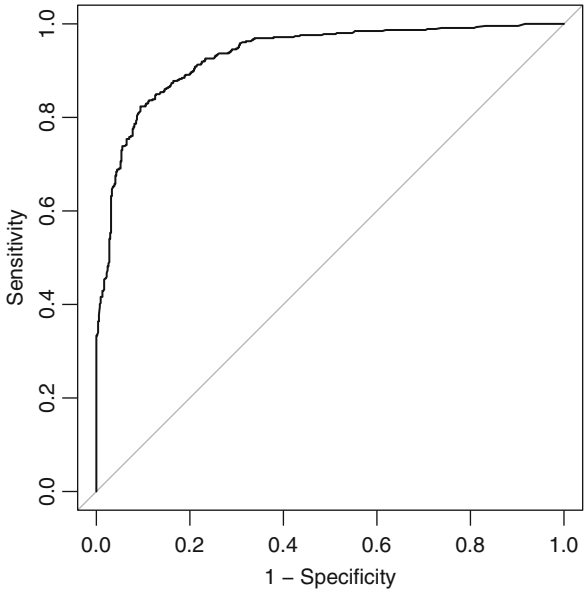

Fig. 11.8: An example of an ROC curve produced using the `roc` and `plot.roc` functions in the pROC package

We can also use the `plot` function to produce the ROC curve itself:

```
> plot(rocCurve, legacy.axes = TRUE)
> ## By default, the x-axis goes backwards, used
> ## the option legacy.axes = TRUE to get 1-spec
> ## on the x-axis moving from 0 to 1
>
> ## Also, another curve can be added using
> ## add = TRUE the next time plot.auc is used.
```

Figure 11.8 shows the results of this function call.

Lift Charts

The lift curve can be created using the `lift` function in the caret package. It takes a formula as the input where the true class is on the left-hand side of the formula, and one or more columns for model class probabilities are on the right. For example, to produce a lift plot for the random forest and QDA test set probabilities,

```
> labs <- c(RFprob = "Random Forest",
+           QDAprob = "Quadratic Discriminant Analysis")
> liftCurve <- lift(class ~ RFprob + QDAprob, data = simulatedTest,
+                   labels = labs)
> liftCurve

  Call:
  lift.formula(x = class ~ RFprob + QDAprob, data = simulatedTest, labels
   = labs)

  Models: Random Forest, Quadratic Discriminant Analysis
  Event: Class1 (45.9%)
```

To plot two lift curves, the `xyplot` function is used to create a lattice plot:

```
> ## Add lattice options to produce a legend on top
> xyplot(liftCurve,
+        auto.key = list(columns = 2,
+                        lines = TRUE,
+                        points = FALSE))
```

See Fig. 11.9.

Calibrating Probabilities

Calibration plots as described above are available in the `calibration.plot` function in the PresenceAbsence package and in the caret function `calibration` (details below). The syntax for the `calibration` function is similar to the `lift` function:

```
> calCurve <- calibration(class ~ RFprob + QDAprob, data = simulatedTest)
> calCurve

  Call:
  calibration.formula(x = class ~ RFprob + QDAprob, data = simulatedTest)

  Models: RFprob, QDAprob
  Event:  Class1
  Cuts: 11
> xyplot(calCurve, auto.key = list(columns = 2))
```

Figure 11.9 also shows this plot. An entirely different approach to calibration plots that model the observed event rate as a function of the class probabilities can be found in the `calibrate.plot` function of the gbm package.

To recalibrate the QDA probabilities, a post-processing model is created that models the true outcome as a function of the class probability. To fit a sigmoidal function, a logistic regression model is used (see Sect. 12.2 for more details) via the `glm` function in base R. This function is an interface to a broad set of methods called generalized linear models (Dobson 2002), which includes logistic regression. To fit the model, the function requires the

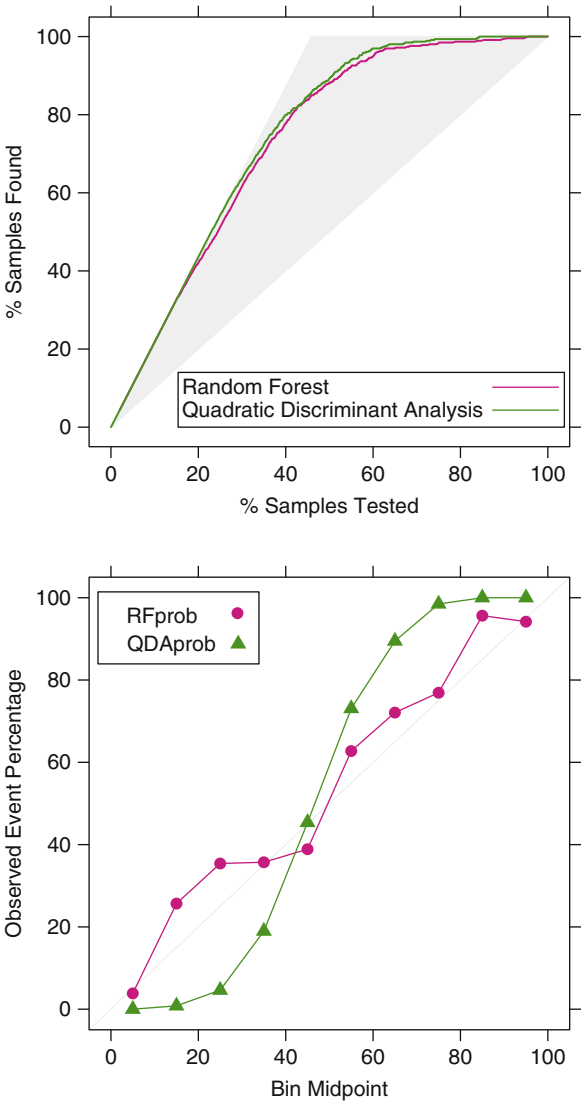

Fig. 11.9: Examples of lift and calibration curves for the random forest and QDA models

`family` argument to specify the type of outcome data being modeled. Since our outcome is a discrete category, the binomial distribution is selected:

```
> ## The glm() function models the probability of the second factor
> ## level, so the function relevel() is used to temporarily reverse the
> ## factors levels.
```

```
> sigmoidalCal <- glm(relevel(class, ref = "Class2") ~ QDAprob,
+                     data = simulatedTrain,
+                     family = binomial)
> coef(summary(sigmoidalCal))
              Estimate Std. Error   z value     Pr(>|z|)
  (Intercept) -5.701055  0.5005652 -11.38924 4.731132e-30
  QDAprob     11.717292  1.0705197  10.94542 6.989017e-28
```

The corrected probabilities are created by taking the original model and applying Eq. 11.1 with the estimated slope and intercept. In R, the `predict` function can be used:

```
> sigmoidProbs <- predict(sigmoidalCal,
+                         newdata = simulatedTest[,"QDAprob", drop = FALSE],
+                         type = "response")
> simulatedTest$QDAsigmoid <- sigmoidProbs
```

The Bayesian approach for calibration is to treat the training set class probabilities to estimate the probabilities $Pr[X]$ and $Pr[X|Y = C_\ell]$ (see Eq. 13.5 on page 354). In R, the naïve Bayes model function `NaiveBayes` in the klaR package can be used for the computations:

```
> BayesCal <- NaiveBayes(class ~ QDAprob, data = simulatedTrain,
+                        usekernel = TRUE)
> ## Like qda(), the predict function for this model creates
> ## both the classes and the probabilities
> BayesProbs <- predict(BayesCal,
+                       newdata = simulatedTest[, "QDAprob", drop = FALSE])
> simulatedTest$QDABayes <- BayesProbs$posterior[, "Class1"]
> ## The probability values before and after calibration
> head(simulatedTest[, c(5:6, 8, 9)])
      QDAprob RFprob QDAsigmoid  QDABayes
  1 0.3830767 0.4300 0.22927068 0.2515696
  2 0.5440393 0.5185 0.66231139 0.6383383
  3 0.9846107 0.9970 0.99708776 0.9995061
  4 0.5463540 0.9395 0.66835048 0.6430232
  5 0.2426705 0.0205 0.05428903 0.0566883
  6 0.4823296 0.2840 0.48763794 0.5109129
```

The option `usekernel = TRUE` allows a flexible function to model the probability distribution of the class probabilities.

These new probabilities are evaluated using another plot:

```
> calCurve2 <- calibration(class ~ QDAprob + QDABayes + QDAsigmoid,
+                          data = simulatedTest)
> xyplot(calCurve2)
```