methods.

RF (%)
95.43
85.80
84.39
96.14
82.18
85.8
75.13
74.5
78.04
83.33
83.23
85.33
93.45
93.33
84.21
73.06
82.43
77.60
85.58
95.82
74.94
84.04
97.75
97.03

ls

ıe perfor-
ndom for-
y decision
ined from
rally out-

many real
monitors
the num-

ber of defective products is significantly fewer than that of non-defective products. Similarly, in credit card fraud detection, fraudulent transactions are outnumbered by legitimate transactions. In both of these examples, there is a disproportionate number of instances that belong to different classes. The degree of imbalance varies from one application to another—a manufacturing plant operating under the six sigma principle may discover four defects in a million products shipped to their customers, while the amount of credit card fraud may be of the order of 1 in 100. Despite their infrequent occurrences, a correct classification of the rare class in these applications often has greater value than a correct classification of the majority class. However, because the class distribution is imbalanced, this presents a number of problems to existing classification algorithms.

The accuracy measure, which is used extensively to compare the performance of classifiers, may not be well suited for evaluating models derived from imbalanced data sets. For example, if 1% of the credit card transactions are fraudulent, then a model that predicts every transaction as legitimate has an accuracy of 99% even though it fails to detect any of the fraudulent activities. Additionally, measures that are used to guide the learning algorithm (e.g., information gain for decision tree induction) may need to be modified to focus on the rare class.

Detecting instances of the rare class is akin to finding a needle in a haystack. Because their instances occur infrequently, models that describe the rare class tend to be highly specialized. For example, in a rule-based classifier, the rules extracted for the rare class typically involve a large number of attributes and cannot be easily simplified into more general rules with broader coverage (unlike the rules for the majority class). Such models are also susceptible to the presence of noise in training data. As a result, many of the existing classification algorithms may not effectively detect instances of the rare class.

This section presents some of the methods developed for handling the class imbalance problem. First, alternative metrics besides accuracy are introduced, along with a graphical method called ROC analysis. We then describe how cost-sensitive learning and sampling-based methods may be used to improve the detection of rare classes.

5.7.1 Alternative Metrics

Since the accuracy measure treats every class as equally important, it may not be suitable for analyzing imbalanced data sets, where the rare class is considered more interesting than the majority class. For binary classification, the rare class is often denoted as the positive class, while the majority class is

Table 5.6. A confusion matrix for a binary classification problem in which the classes are not equally important.

		Predicted Class	
		+	−
Actual Class	+	f_{++} (TP)	f_{+-} (FN)
	−	f_{-+} (FP)	f_{--} (TN)

denoted as the negative class. A confusion matrix that summarizes the number of instances predicted correctly or incorrectly by a classification model is shown in Table 5.6.

The following terminology is often used when referring to the counts tabulated in a confusion matrix:

- True positive (TP) or f_{++}, which corresponds to the number of positive examples correctly predicted by the classification model.
- False negative (FN) or f_{+-}, which corresponds to the number of positive examples wrongly predicted as negative by the classification model.
- False positive (FP) or f_{-+}, which corresponds to the number of negative examples wrongly predicted as positive by the classification model.
- True negative (TN) or f_{--}, which corresponds to the number of negative examples correctly predicted by the classification model.

The counts in a confusion matrix can also be expressed in terms of percentages. The **true positive rate** (TPR) or **sensitivity** is defined as the fraction of positive examples predicted correctly by the model, i.e.,

$$TPR = TP/(TP + FN).$$

Similarly, the **true negative rate** (TNR) or **specificity** is defined as the fraction of negative examples predicted correctly by the model, i.e.,

$$TNR = TN/(TN + FP).$$

Finally, the **false positive rate** (FPR) is the fraction of negative examples predicted as a positive class, i.e.,

$$FPR = FP/(TN + FP),$$

while the **false negative rate** (FNR) is the fraction of positive examples predicted as a negative class, i.e.,

$$FNR = FN/(TP + FN).$$

Recall and **precision** are two widely used metrics employed in applications where successful detection of one of the classes is considered more significant than detection of the other classes. A formal definition of these metrics is given below.

$$\text{Precision, } p = \frac{TP}{TP + FP} \tag{5.74}$$

$$\text{Recall, } r = \frac{TP}{TP + FN} \tag{5.75}$$

Precision determines the fraction of records that actually turns out to be positive in the group the classifier has declared as a positive class. The higher the precision is, the lower the number of false positive errors committed by the classifier. Recall measures the fraction of positive examples correctly predicted by the classifier. Classifiers with large recall have very few positive examples misclassified as the negative class. In fact, the value of recall is equivalent to the true positive rate.

It is often possible to construct baseline models that maximize one metric but not the other. For example, a model that declares every record to be the positive class will have a perfect recall, but very poor precision. Conversely, a model that assigns a positive class to every test record that matches one of the positive records in the training set has very high precision, but low recall. Building a model that maximizes both precision and recall is the key challenge of classification algorithms.

Precision and recall can be summarized into another metric known as the F_1 measure.

$$F_1 = \frac{2rp}{r + p} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{5.76}$$

In principle, F_1 represents a harmonic mean between recall and precision, i.e.,

$$F_1 = \frac{2}{\frac{1}{r} + \frac{1}{p}}.$$

The harmonic mean of two numbers x and y tends to be closer to the smaller of the two numbers. Hence, a high value of F_1-measure ensures that both

precision and recall are reasonably high. A comparison among harmonic, geometric, and arithmetic means is given in the next example.

Example 5.8. Consider two positive numbers $a = 1$ and $b = 5$. Their arithmetic mean is $\mu_a = (a + b)/2 = 3$ and their geometric mean is $\mu_g = \sqrt{ab} = 2.236$. Their harmonic mean is $\mu_h = (2\times1\times5)/6 = 1.667$, which is closer to the smaller value between a and b than the arithmetic and geometric means. ■

More generally, the F_β measure can be used to examine the tradeoff between recall and precision:

$$F_\beta = \frac{(\beta^2 + 1)rp}{r + \beta^2 p} = \frac{(\beta^2 + 1) \times TP}{(\beta^2 + 1)TP + \beta^2 FP + FN}. \tag{5.77}$$

Both precision and recall are special cases of F_β by setting $\beta = 0$ and $\beta = \infty$, respectively. Low values of β make F_β closer to precision, and high values make it closer to recall.

A more general metric that captures F_β as well as accuracy is the weighted accuracy measure, which is defined by the following equation:

$$\text{Weighted accuracy} = \frac{w_1 TP + w_4 TN}{w_1 TP + w_2 FP + w_3 FN + w_4 TN}. \tag{5.78}$$

The relationship between weighted accuracy and other performance metrics is summarized in the following table:

Measure	w_1	w_2	w_3	w_4
Recall	1	1	0	0
Precision	1	0	1	0
F_β	$\beta^2 + 1$	β^2	1	0
Accuracy	1	1	1	1

5.7.2 The Receiver Operating Characteristic Curve

A receiver operating characteristic (ROC) curve is a graphical approach for displaying the tradeoff between true positive rate and false positive rate of a classifier. In an ROC curve, the true positive rate (TPR) is plotted along the y axis and the false positive rate (FPR) is shown on the x axis. Each point along the curve corresponds to one of the models induced by the classifier. Figure 5.41 shows the ROC curves for a pair of classifiers, M_1 and M_2.

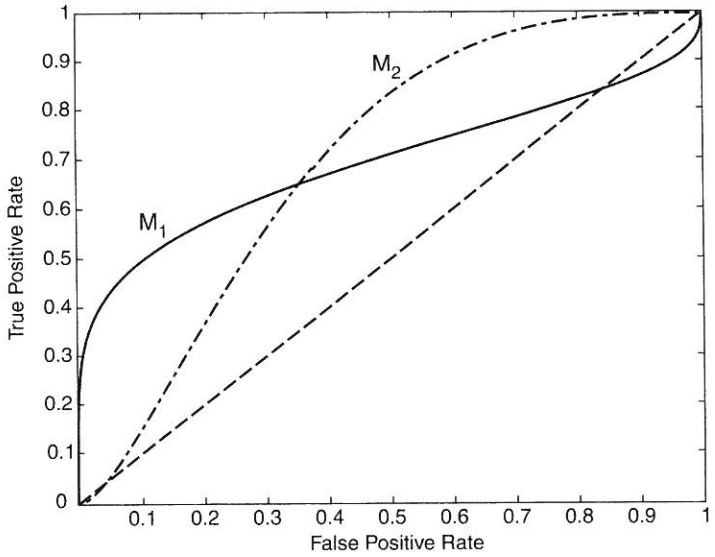


Figure 5.41. ROC curves for two different classifiers.

There are several critical points along an ROC curve that have well-known interpretations:

(TPR=0, FPR=0): Model predicts every instance to be a negative class.
(TPR=1, FPR=1): Model predicts every instance to be a positive class.
(TPR=1, FPR=0): The ideal model.

A good classification model should be located as close as possible to the upper left corner of the diagram, while a model that makes random guesses should reside along the main diagonal, connecting the points $(TPR = 0, FPR = 0)$ and $(TPR = 1, FPR = 1)$. Random guessing means that a record is classified as a positive class with a fixed probability p, irrespective of its attribute set. For example, consider a data set that contains n_+ positive instances and n_- negative instances. The random classifier is expected to correctly classify pn_+ of the positive instances and to misclassify pn_- of the negative instances. Therefore, the TPR of the classifier is $(pn_+)/n_+ = p$, while its FPR is $(pn_-)/p = p$. Since the TPR and FPR are identical, the ROC curve for a random classifier always reside along the main diagonal.

An ROC curve is useful for comparing the relative performance among different classifiers. In Figure 5.41, M_1 is better than M_2 when FPR is less

than 0.36, while M_2 is superior when FPR is greater than 0.36. Clearly, neither of these two classifiers dominates the other.

The area under the ROC curve (AUC) provides another approach for evaluating which model is better on average. If the model is perfect, then its area under the ROC curve would equal 1. If the model simply performs random guessing, then its area under the ROC curve would equal 0.5. A model that is strictly better than another would have a larger area under the ROC curve.

Generating an ROC curve

To draw an ROC curve, the classifier should be able to produce a continuous-valued output that can be used to rank its predictions, from the most likely record to be classified as a positive class to the least likely record. These outputs may correspond to the posterior probabilities generated by a Bayesian classifier or the numeric-valued outputs produced by an artificial neural network. The following procedure can then be used to generate an ROC curve:

1. Assuming that the continuous-valued outputs are defined for the positive class, sort the test records in increasing order of their output values.

2. Select the lowest ranked test record (i.e., the record with lowest output value). Assign the selected record and those ranked above it to the positive class. This approach is equivalent to classifying all the test records as positive class. Because all the positive examples are classified correctly and the negative examples are misclassified, $TPR = FPR = 1$.

3. Select the next test record from the sorted list. Classify the selected record and those ranked above it as positive, while those ranked below it as negative. Update the counts of TP and FP by examining the actual class label of the previously selected record. If the previously selected record is a positive class, the TP count is decremented and the FP count remains the same as before. If the previously selected record is a negative class, the FP count is decremented and TP count remains the same as before.

4. Repeat Step 3 and update the TP and FP counts accordingly until the highest ranked test record is selected.

5. Plot the TPR against FPR of the classifier.

Figure 5.42 shows an example of how to compute the ROC curve. There are five positive examples and five negative examples in the test set. The class

Class	+	−	+	−	−	−	+	−	+	+	
	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0

Figure 5.42. Constructing an ROC curve.

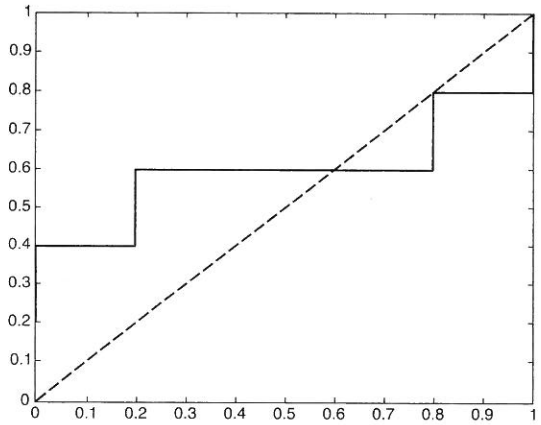


Figure 5.43. ROC curve for the data shown in Figure 5.42.

labels of the test records are shown in the first row of the table. The second row corresponds to the sorted output values for each record. For example, they may correspond to the posterior probabilities $P(+|\mathbf{x})$ generated by a naïve Bayes classifier. The next six rows contain the counts of TP, FP, TN, and FN, along with their corresponding TPR and FPR. The table is then filled from left to right. Initially, all the records are predicted to be positive. Thus, $TP = FP = 5$ and $TPR = FPR = 1$. Next, we assign the test record with the lowest output value as the negative class. Because the selected record is actually a positive example, the TP count reduces from 5 to 4 and the FP count is the same as before. The FPR and TPR are updated accordingly. This process is repeated until we reach the end of the list, where $TPR = 0$ and $FPR = 0$. The ROC curve for this example is shown in Figure 5.43.

5.7.3 Cost-Sensitive Learning

A cost matrix encodes the penalty of classifying records from one class as another. Let $C(i, j)$ denote the cost of predicting a record from class i as class j. With this notation, $C(+, -)$ is the cost of committing a false negative error, while $C(-, +)$ is the cost of generating a false alarm. A negative entry in the cost matrix represents the reward for making correct classification. Given a collection of N test records, the overall cost of a model M is

$$C_t(M) = TP \times C(+,+) + FP \times C(-,+) + FN \times C(+,-) + TN \times C(-,-). \quad (5.79)$$

Under the 0/1 cost matrix, i.e., $C(+,+) = C(-,-) = 0$ and $C(+,-) = C(-,+) = 1$, it can be shown that the overall cost is equivalent to the number of misclassification errors.

$$C_t(M) = 0 \times (TP + TN) + 1 \times (FP + FN) = N \times Err, \quad (5.80)$$

where Err is the error rate of the classifier.

Example 5.9. Consider the cost matrix shown in Table 5.7: The cost of committing a false negative error is a hundred times larger than the cost of committing a false alarm. In other words, failure to detect any positive example is just as bad as committing a hundred false alarms. Given the classification models with the confusion matrices shown in Table 5.8, the total cost for each model is

$$C_t(M_1) = 150 \times (-1) + 60 \times 1 + 40 \times 100 = 3910,$$
$$C_t(M_2) = 250 \times (-1) + 5 \times 1 + 45 \times 100 = 4255.$$

Table 5.7. Cost matrix for Example 5.9.

		Predicted Class	
		Class = +	Class = −
Actual Class	Class = +	−1	100
	Class = −	1	0

Table 5.8. Confusion matrix for two classification models.

Model M_1		Predicted Class	
		Class +	Class −
Actual Class	Class +	150	40
	Class −	60	250

Model M_2		Predicted Class	
		Class +	Class −
Actual Class	Class +	250	45
	Class −	5	200

Notice that despite improving both of its true positive and false positive counts, model M_2 is still inferior since the improvement comes at the expense of increasing the more costly false negative errors. A standard accuracy measure would have preferred model M_2 over M_1. ■

A cost-sensitive classification technique takes the cost matrix into consideration during model building and generates a model that has the lowest cost. For example, if false negative errors are the most costly, the learning algorithm will try to reduce these errors by extending its decision boundary toward the negative class, as shown in Figure 5.44. In this way, the generated model can cover more positive examples, although at the expense of generating additional false alarms.

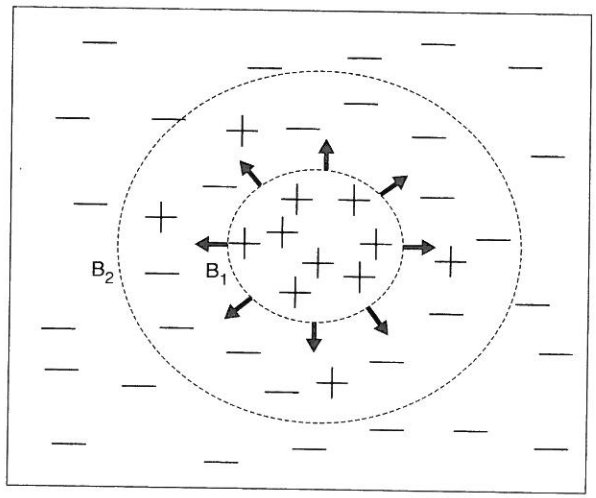


Figure 5.44. Modifying the decision boundary (from B_1 to B_2) to reduce the false negative errors of a classifier.

There are various ways to incorporate cost information into classification algorithms. For example, in the context of decision tree induction, the cost

information can be used to: (1) choose the best attribute to use for splitting the data, (2) determine whether a subtree should be pruned, (3) manipulate the weights of the training records so that the learning algorithm converges to a decision tree that has the lowest cost, and (4) modify the decision rule at each leaf node. To illustrate the last approach, let $p(i|t)$ denote the fraction of training records from class i that belong to the leaf node t. A typical decision rule for a binary classification problem assigns the positive class to node t if the following condition holds.

$$\begin{aligned} & p(+|t) > p(-|t) \\ \Longrightarrow \quad & p(+|t) > (1 - p(+|t)) \\ \Longrightarrow \quad & 2p(+|t) > 1 \\ \Longrightarrow \quad & p(+|t) > 0.5. \end{aligned} \tag{5.81}$$

The preceding decision rule suggests that the class label of a leaf node depends on the majority class of the training records that reach the particular node. Note that this rule assumes that the misclassification costs are identical for both positive and negative examples. This decision rule is equivalent to the expression given in Equation 4.8 on page 165.

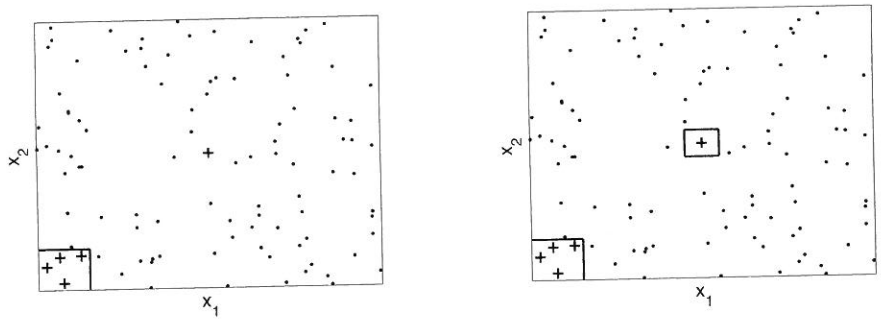
Instead of taking a majority vote, a cost-sensitive algorithm assigns the class label i to node t if it minimizes the following expression:

$$C(i|t) = \sum_j p(j|t)C(j,i). \tag{5.82}$$

In the case where $C(+,+) = C(-,-) = 0$, a leaf node t is assigned to the positive class if:

$$\begin{aligned} & p(+|t)C(+,-) > p(-|t)C(-,+) \\ \Longrightarrow \quad & p(+|t)C(+,-) > (1 - p(+|t))C(-,+) \\ \Longrightarrow \quad & p(+|t) > \frac{C(-,+)}{C(-,+) + C(+,-)}. \end{aligned} \tag{5.83}$$

This expression suggests that we can modify the threshold of the decision rule from 0.5 to $C(-,+)/(C(-,+) + C(+,-))$ to obtain a cost-sensitive classifier. If $C(-,+) < C(+,-)$, then the threshold will be less than 0.5. This result makes sense because the cost of making a false negative error is more expensive than that for generating a false alarm. Lowering the threshold will expand the decision boundary toward the negative class, as shown in Figure 5.44.



(a) Without oversampling

(b) With oversampling

Figure 5.45. Illustrating the effect of oversampling of the rare class.

5.7.4 Sampling-Based Approaches

Sampling is another widely used approach for handling the class imbalance problem. The idea of sampling is to modify the distribution of instances so that the rare class is well represented in the training set. Some of the available techniques for sampling include undersampling, oversampling, and a hybrid of both approaches. To illustrate these techniques, consider a data set that contains 100 positive examples and 1000 negative examples.

In the case of undersampling, a random sample of 100 negative examples is chosen to form the training set along with all the positive examples. One potential problem with this approach is that some of the useful negative examples may not be chosen for training, therefore, resulting in a less than optimal model. A potential method to overcome this problem is to perform undersampling multiple times and to induce multiple classifiers similar to the ensemble learning approach. Focused undersampling methods may also be used, where the sampling procedure makes an informed choice with regard to the negative examples that should be eliminated, e.g., those located far away from the decision boundary.

Oversampling replicates the positive examples until the training set has an equal number of positive and negative examples. Figure 5.45 illustrates the effect of oversampling on the construction of a decision boundary using a classifier such as a decision tree. Without oversampling, only the positive examples at the bottom right-hand side of Figure 5.45(a) are classified correctly. The positive example in the middle of the diagram is misclassified because there