Lecture 2

Judging the Performance of Classifiers

Nitin R. Patel

In this note we will examine the question of how to judge the usefulness of a classifier and how to compare different classifiers. Not only do we have a wide choice of different types of classifiers to choose from but within each type of classifier we have many options such as how many nearest neighbors to use in a k-nearest neighbors classifier, the minimum number of cases we should require in a leaf node in a tree classifier, which subsets of predictors to use in a logistic regression model, and how many hidden layer neurons to use in a neural net.

A Two-class Classifier

Let us first look at a single classifier for two classes with options set at certain values. The two-class situation is certainly the most common and occurs very frequently in practice. We will extend our analysis to more than two classes later.

A natural criterion for judging the performance of a classifier is the probability that it makes a misclassification. A classifier that makes no errors would be perfect but we do not expect to be able to construct such classifiers in the real world due to “noise” and to not having all the information needed to precisely classify cases. Is there a minimum probability of misclassification we should require of a classifier?

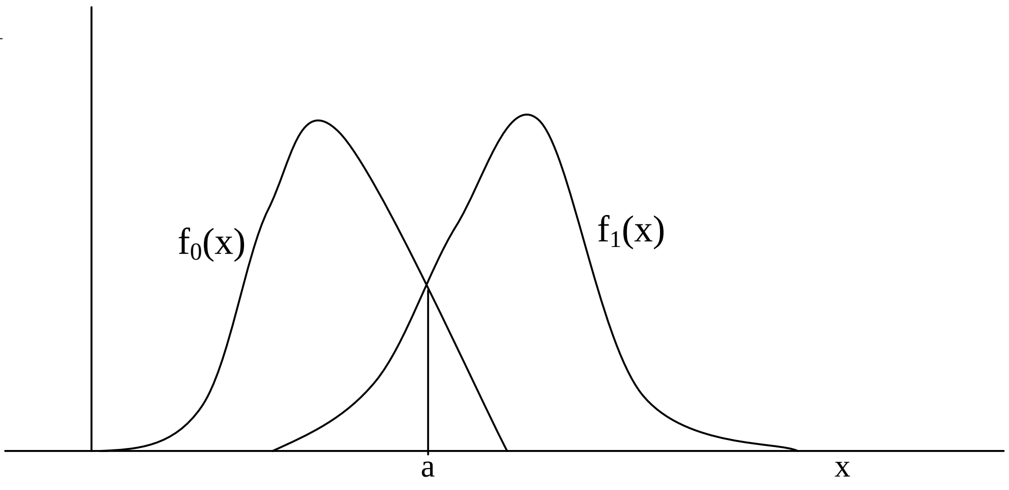
Suppose that the two classes are denoted by C_0 and C_1. Let $p(C_0)$ and $p(C_1)$ be the apriori probabilities that a case belongs to C_0 and C_1 respectively. The apriori probability is the probability that a case belongs to a class without any more knowledge about it than that it belongs to a population where the proportion of C_0’s is $p(C_0)$ and the proportion of C_1’s is $p(C_1)$. In this situation we will minimize the chance of a misclassification error by assigning class C_1 to the case if $p(C_1) > p(C_0)$ and to C_0 otherwise. The probability of making a misclassification would be the minimum of $p(C_0)$ and $p(C_1)$. If we are using misclassification rate as our criterion any classifier that uses predictor variables must have an error rate better than this.

What is the best performance we can expect from a classifier? Clearly the more training data available to a classifier the more accurate it will be. Suppose we had a huge amount of training data, would we then be able to build a classifier that makes no errors? The answer is no. The accuracy of a classifier depends critically on how separated the classes are with respect to the predictor variables that the classifier uses. We can use the well-known Bayes’ formula from probability theory to derive the best performance we can expect from a classifier for a given set of predictor variables if we had a very large amount of training data. Bayes' formula uses the distributions of the decision variables in the two classes to give us a classifier that will have the minimum error amongst all classifiers that use the same predictor variables. This classifier follows the Minimum Error Bayes Rule.

Bayes Rule for Minimum Error

Let us take a simple situation where we have just one continuous predictor variable for classification, say X. X is a random variable, since it's value depends on the individual case we sample from the population consisting of all possible cases of the class to which the case belongs. Suppose that we have a very large training data set. Then the relative frequency histogram of the variable X in each class would be almost identical to the probability density function (p.d.f.) of X for that class. Let us assume that we have a huge amount of training data and so we know the p.d.f.s accurately. These p.d.f.s are denoted $f_0(x)$ and $f_1(x)$ for classes C_0 and C_1 in Fig. 1 below.

Figure 1



Now suppose we wish to classify an object for which the value of X is x_0. Let us use Bayes' formula to predict the probability that the object belongs to class 1 conditional on the fact that it has an X value of x_0. Appling Bayes' formula, the probability, denoted by $p(C_1|X= x_0)$, is given by:

$$p(C_1|X = x_0) = \frac{p(X = x_0|C_1)\,p(C_1)}{p(X = x_0|C_0)\,p(C_0) + p(X = x_0|C_1)\,p(C_1)}$$

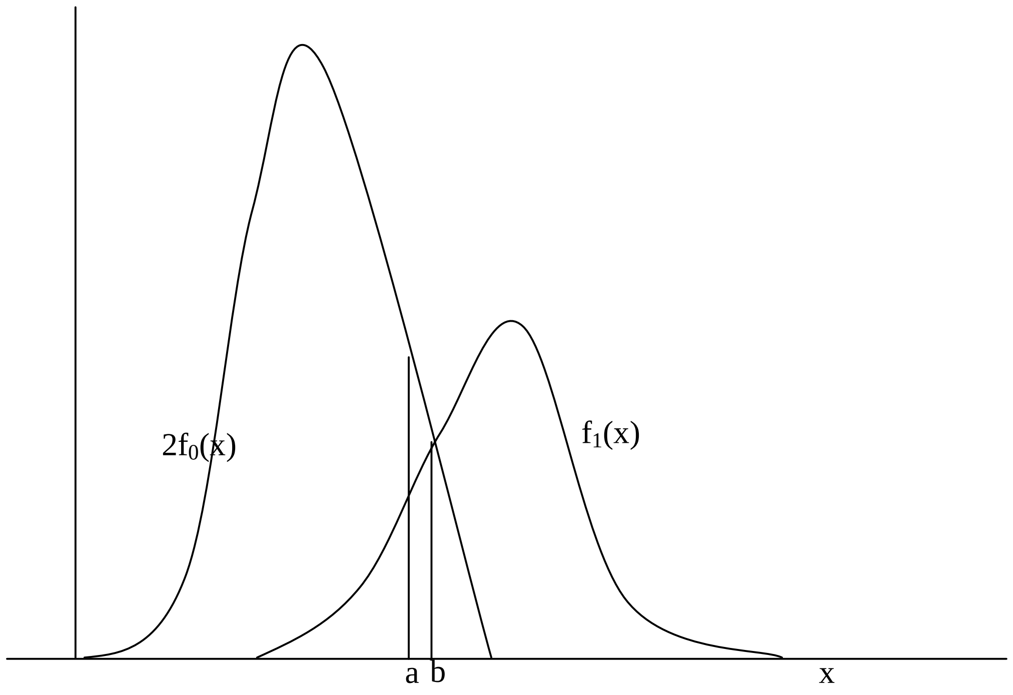
Writing this in terms of the density functions, we get

$$p(C_1|X = x_0) = \frac{f_1(x_0)\,p(C_1)}{f_0(x_0)\,p(C_0) + f_1(x_0)\,p(C_1)}$$

Notice that to calculate $p(C_1|X= x_0)$ we need to know the apriori probabilities $p(C_0)$ and $p(C_1)$. Since there are only two possible classes, if we know $p(C_1)$ we can always compute $p(C_0)$ because $p(C_0) = 1 - p(C_1)$. The apriori probability $p(C_1)$ is the probability that an object belongs to C_1 without any knowledge of the value of X associated with it. Bayes' formula enables us to update this apriori probability to the aposteriori probability, the probability of the object belonging to C_1 after knowing that its X value is x_0.

When $p(C_1) = p(C_0) = 0.5$, the formula shows that $p(C_1|X = x_0) > p(C_0|X = x_0)\ if\ \ f_1(x_0) > f_0(x_0)$. This means that if x_0 is greater than a, and we classify the object as belonging to C_1 we will make a smaller misclassification error than if we were to classify it as belonging to C_0 .Similarly if x_0 is less than a, and we classify the object as belonging to C_0 we will make a smaller misclassification error than if we were to classify it as belonging to $C_{1.}$ If x_0 is exactly equal to a we have a 50% chance of making an error for either classification.

Figure 2



What if the prior class probabilities were not the same? Suppose C_0 is twice as likely apriori as C_1. Then the formula says that $p(C_1 | X = x_0) > p(C_0 | X = x_0)\ if\ \ f_1(x_0) > 2 \times f_0(x_0)$. The new boundary value, b for classification will be to the right of a as shown in Fig.2. This is intuitively what we would expect. If a class is more likely we would expect the cut-off to move in a direction that would increase the range over which it is preferred.

In general we will minimize the misclassification error rate if we classify a case as belonging to $C_1\ if\ \ p(C_1) \times f_1(x_0) > p(C_0) \times f_0(x_0)$, and to C_0 , otherwise. This rule holds even when X is a vector consisting of several components, each of which is a random variable. In the remainder of this note we shall assume that X is a vector.

An important advantage of Bayes' Rule is that, as a by-product of classifying a case, we can compute the conditional probability that the case belongs to each class. This has two advantages.

First, we can use this probability as a "score" for each case that we are classifying. The score enables us to rank cases that we have predicted as belonging to a class in order of confidence that we have made a correct classification. This capability is important in developing a lift curve (explained later) that is important for many practical data mining applications.

Second, it enables us to compute the expected profit or loss for a given case. This gives us a better decision criterion than misclassification error when the loss due to error is different for the two classes.

Practical assessment of a classifier using misclassification error as the criterion

In practice, we can estimate $p(C_1)$ *and* $p(C_0)$ from the data we are using to build the classifier by simply computing the proportion of cases that belong to each class. Of course, these are estimates and they can be incorrect, but if we have a large enough data set and neither class is very rare our estimates will be reliable. Sometimes, we may be able to use public data such as census data to estimate these proportions. However, in most practical business settings we will not know $f_1(x)$ *and* $f_0(x)$. If we want to apply Bayes' Rule we will need to estimate these density functions in some way. Many classification methods can be interpreted as being methods for estimating such density functions[1]. In practice X will almost always be a vector. This makes the task difficult and subject to the curse of dimensionality we referred to when discussing the k-Nearest Neighbors technique.
To obtain an honest estimate of classification error, let us suppose that we have partitioned a data set into training and validation data sets by random selection of cases. Let us assume that we have constructed a classifier using the training data. When we apply it to the validation data, we will classify each case into C_0 or C_1. The resulting misclassification errors can be displayed in what is known as a confusion table, with rows and columns corresponding to the true and predicted classes respectively. We can summarize our results in a confusion table for training data in a similar fashion. The resulting confusion table will not give us an honest estimate of the misclassification rate due to over-fitting. However such a table will be useful to signal over-fitting when it has substantially lower misclassification rates than the confusion table for validation data.

Confusion Table (Validation Cases)	*Predicted Class*	
True Class	C_0	C_1
C_0	True Negatives (Number of correctly classified cases that belong to C_0)	False Positives (Number of cases incorrectly classified as C_1 that belong to C_0)
C_1	False Negatives (Number of cases incorrectly classified as C_0 that belong to C_1)	True Positives (Number of correctly classified cases that belong to C_1)

If we denote the number in the cell at row i and column j by N_{ij}, the estimated misclassification rate Err = $(N_{01} + N_{10}) / N_{val}$ *where* $N_{val} \equiv (N_{00} + N_{01} + N_{10} + N_{11})$, or the total number of cases in the validation data set. If N_{val} is reasonably large, our estimate of the misclassification

[1] There are classifiers that focus on simply finding the boundary between the regions to predict each class without being concerned with estimating the density of cases within each region. For example, Support Vector Machine classifiers have this characteristic.

rate is probably quite accurate. We can compute a confidence interval for Err using the standard formula for estimating a population proportion from a random sample.

The table below gives an idea of how the accuracy of the estimate varies with N_{val} . The column headings are values of the misclassification rate and the rows give the desired accuracy in estimating the misclassification rate as measured by the half-width of the confidence interval at the 99% confidence level. For example, if we think that the true misclassification rate is likely to be around 0.05 and we want to be 99% confident that Err is within ± 0.01 of the true misclassification rate, we need to have a validation data set with 3,152 cases.

	0.01	0.05	0.10	0.15	0.20	0.30	0.40	0.50
±0.025	250	504	956	1,354	1,699	2,230	2,548	2,654
±0.010	657	3,152	5,972	8,461	10,617	13,935	15,926	16,589
±0.005	2,628	12,608	23,889	33,842	42,469	55,741	63,703	66,358

Note that we are assuming that the cost (or benefit) of making correct classifications is zero. At first glance, this may seem incomplete. After all, the benefit (negative cost) of correctly classifying a buyer as a buyer would seem substantial. And, in other circumstances (e.g. calculating the expected profit from having a new mailing list), it will be appropriate to consider the actual net dollar impact of classifying each case on the list. Here, however, we are attempting to assess the value of a classifier in terms of misclassifications, so it greatly simplifies matters if we can capture all cost/benefit information in the misclassification cells. So, instead of recording the benefit of correctly classifying a buyer, we record the cost of failing to classify him as a buyer. It amounts to the same thing. In fact the costs we are using are the opportunity costs.

Asymmetric misclassification costs and Bayes’ Risk

Up to this point we have been using the misclassification error rate as the criterion for judging the efficacy of a classifier. However, there are circumstances when this measure is not appropriate. Sometimes the error of misclassifying a case belonging to one class is more serious than for the other class. For example, misclassifying a household as unlikely to respond to a sales offer when it belongs to the class that would respond incurs a greater opportunity cost than the converse error. In such a scenario using misclassification error as a criterion can be misleading. Consider the situation where the sales offer is accepted by 1% of the households on a list. If a classifier simply classifies every household as a non-responder it will have an error rate of only 1% but will be useless in practice. A classifier that misclassifies 30% of buying households as non-buyers and 2% of the non-buyers as buyers would have a higher error rate but would be better if the profit from a sale is substantially higher than the cost of sending out an offer. In these situations, if we have estimates of the cost of both types of misclassification, we can use the confusion table to compute the expected cost of misclassification for each case in the validation data. This enables us to compare different classifiers using opportunity cost as the criterion. This may suffice for some situations, but a better method would be to change the classification rules (and hence the misclassification rates) to reflect the asymmetric costs. In fact, there is a Bayes classifier for this situation which gives rules that are optimal for minimizing the expected opportunity loss from misclassification. This classifier is known as the Bayes’ Risk Classifier and the corresponding minimum expected opportunity cost of misclassification is known as the Bayes’ Risk. The Bayes’ Risk Classifier employs the following classification rule:

Classify a case as belonging to C_1 $if\ \ p(C_1)\times f_1(x_0)\times C(0|1) > p(C_0)\times f_0(x_0)\times C(1|0)$, and to C_0 , otherwise. Here C(0|1) is the opportunity cost of misclassifying a C_1 case as belonging to C_0 and C(1|0) is the opportunity cost of misclassifying a C_0 case as belonging to C_1. Note that the opportunity cost of correct classification for either class is zero. Notice also that this rule reduces to the Minimum Error Bayes Rule when C(0|1) = C(1|0).

Again, as we rarely know $f_1(x)\ and\ f_0(x)$, we cannot construct this classifier in practice. Nonetheless, it provides us with an ideal that the various classifiers we construct for minimizing expected opportunity cost attempt to emulate. The method used most often in practice is to use stratified sampling instead of random sampling so as to change the ratio of cases in the training set to reflect the relative costs of making the two types of misclassification errors.

Stratified sampling to make the classifier sensitive to asymmetric costs

The basic idea in using stratified sampling is to oversample the cases from a class to increase the weight given to errors made in classifying cases in that class. If we feel that the opportunity cost of misclassifying a class C_1 case as a C_0 case is ten times that of misclassifying a class C_0 case as a C_1 case, we randomly sample ten times as many C_1 cases as we randomly sample C_0 cases. By virtue of this oversampling, the training data will automatically tune the classifier to be more accurate in classifying C_1 cases than C_0 cases. Most of the time the class that has a higher misclassification cost will be the less frequently occurring class (for example fraudulent cases). In this situation rather than reduce the number of cases that are not fraudulent to a small fraction of the fraud cases and thus have a drastic reduction in the training data size, a good thumb rule often used in practice is to sample an equal number of cases from each class. This is a very commonly used option as it tends to produce rules that are quite efficient relative to the best over a wide range of misclassification cost ratios.

Generalization to more than two classes

All the comments made above about two-class classifiers extend readily to classification into more than two classes. Let us suppose we have k classes C_0, $C_{1,}$ C_2, … $C_{k-1.}$ Then Bayes formula gives us:

$$p(C_j | X = x_0) = \frac{f_j(x_0)\,p(C_j)}{\sum_{i=1}^{k-1} f_i(x_0)\,p(C_i)}$$

The Bayes Rule for Minimum Error is to classify a case as belonging to C_j
$if\ \ p(C_j)\times f_j(x_0) \geq \underset{i=0,1,\ldots k-1}{Max}\ p(C_i)\times f_i(x_0)$.

The confusion table has k rows and k columns. The opportunity cost associated with the diagonal cells is always zero. If the costs are asymmetric the Bayes Risk Classifier follows the rule:
Classify a case as belonging to C_1

$$if\ \ p(C_j)\times f_j(x_0)\times C(\sim j|\,j) \geq \underset{i\neq j}{Max}\ p(C_i)\times f_i(x_0)\times C(\sim i|i)\,.$$

where C(~j|j) is the cost of misclassifying a case that belongs to C_j to any other class $C_{i,}$ $i \neq j$.

Lift Charts for two-class classifiers

Often in practice, opportunity costs are not known accurately and decision makers would like to examine a range of possible opportunity costs. In such cases, when the classifier gives a probability of belonging to each class and not just a binary (or “hard”) classification to C_1 or C_0, we can use a very useful device known as the lift curve. The lift curve is a popular technique in direct marketing. The input required to construct a lift curve is a validation data set that has been “scored” by appending the probability predicted by a classifier to each case. In fact we can use classifiers that do not predict probabilities but give scores that enable us to rank cases in order of how likely the cases are to belong to one of the classes.

Example: Boston Housing (Two classes)

Let us fit a logistic regression model to the Boston Housing data. We fit a logistic regression model to the training data (304 randomly selected cases) with all the 13 variables available in the data set as predictor variables and with the binary variable HICLASS (high valued property neighborhood) as the dependent variable. The model coefficients are applied to the validation data (the remaining 202 cases in the data set). The first three columns of XLMiner output for the first 30 cases in the validation data are shown below.

	Predicted Log-odds of Success	Predicted Prob. of Success	Actual Value of HICLASS
1	3.5993	0.9734	1
2	-6.5073	0.0015	0
3	0.4061	0.6002	0
4	-14.2910	0.0000	0
5	4.5273	0.9893	1
6	-1.2916	0.2156	0
7	-37.6119	0.0000	0
8	-1.1157	0.2468	0
9	-4.3290	0.0130	0
10	-24.5364	0.0000	0
11	-21.6854	0.0000	0
12	-19.8654	0.0000	0
13	-13.1040	0.0000	0
14	4.4472	0.9884	1
15	3.5294	0.9715	1
16	3.6381	0.9744	1
17	-2.6806	0.0641	0
18	-0.0402	0.4900	0
19	-10.0750	0.0000	0
20	-10.2859	0.0000	0
21	-14.6084	0.0000	0
22	8.9016	0.9999	1
23	0.0874	0.5218	0
24	-6.0590	0.0023	1
25	-1.9183	0.1281	1
26	-13.2349	0.0000	0

27	-9.6509	0.0001	0
28	-13.4562	0.0000	0
29	-13.9340	0.0000	0
30	1.7257	0.8489	1

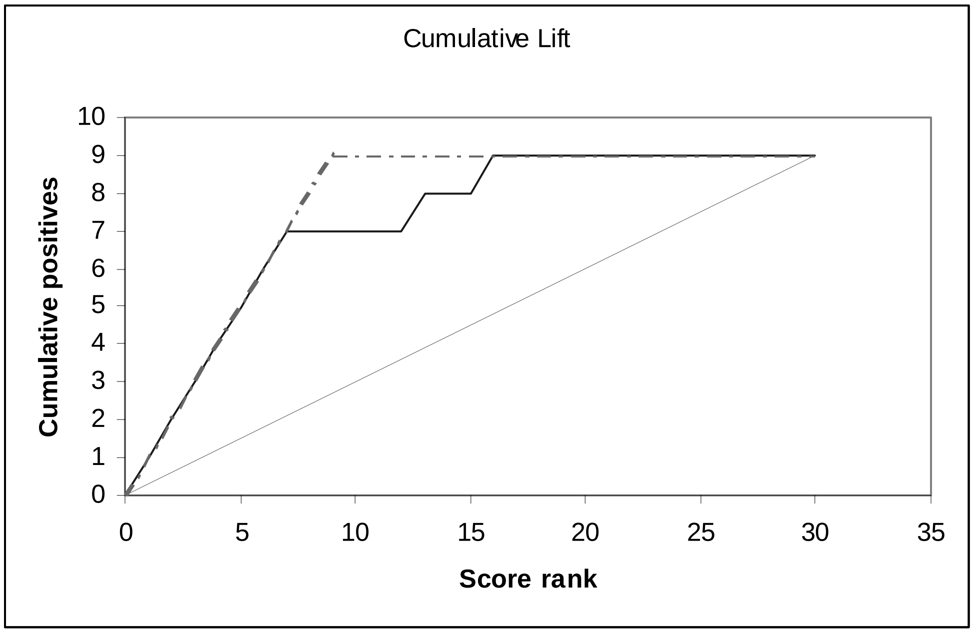
The same 30 cases are shown below sorted in descending order of the <u>predicted probability</u> of being a HCLASS=1 case.

	Predicted Log-odds of Success	Predicted Prob. of Success	Actual Value of HICLASS
22	8.9016	0.9999	1
5	4.5273	0.9893	1
14	4.4472	0.9884	1
16	3.6381	0.9744	1
1	3.5993	0.9734	1
15	3.5294	0.9715	1
30	1.7257	0.8489	1
3	0.4061	0.6002	0
23	0.0874	0.5218	0
18	-0.0402	0.4900	0
8	-1.1157	0.2468	0
6	-1.2916	0.2156	0
25	-1.9183	0.1281	1
17	-2.6806	0.0641	0
9	-4.3290	0.0130	0
24	-6.0590	0.0023	1
2	-6.5073	0.0015	0
27	-9.6509	0.0001	0
19	-10.0750	0.0000	0
20	-10.2859	0.0000	0
13	-13.1040	0.0000	0
26	-13.2349	0.0000	0
28	-13.4562	0.0000	0
29	-13.9340	0.0000	0
4	-14.2910	0.0000	0
21	-14.6084	0.0000	0
12	-19.8654	0.0000	0
11	-21.6854	0.0000	0
10	-24.5364	0.0000	0
7	-37.6119	0.0000	0

First, we need to set a cutoff probability value, above which we will consider a case to be a positive or "1," and below which we will consider a case to be a negative or "0." For any given cutoff level, we can use the sorted table to compute a confusion table for a given cut-off probability. For example, if we use a cut-off probability level of 0.400, we will predict 10 positives (7 true positives and 3 false positives); we will also predict 20 negatives (18 true negatives and 2 false negatives). For each cut-off level, we can calculate the appropriate confusion table. Instead of looking at a large number of confusion tables, it is much more convenient to look at the cumulative lift curve (sometimes called a gains chart) which summarizes all the information in these multiple confusion tables into a graph. The graph is constructed with the cumulative number of cases (in descending order of probability) on the x axis and the cumulative number of true positives on the y axis as shown below.

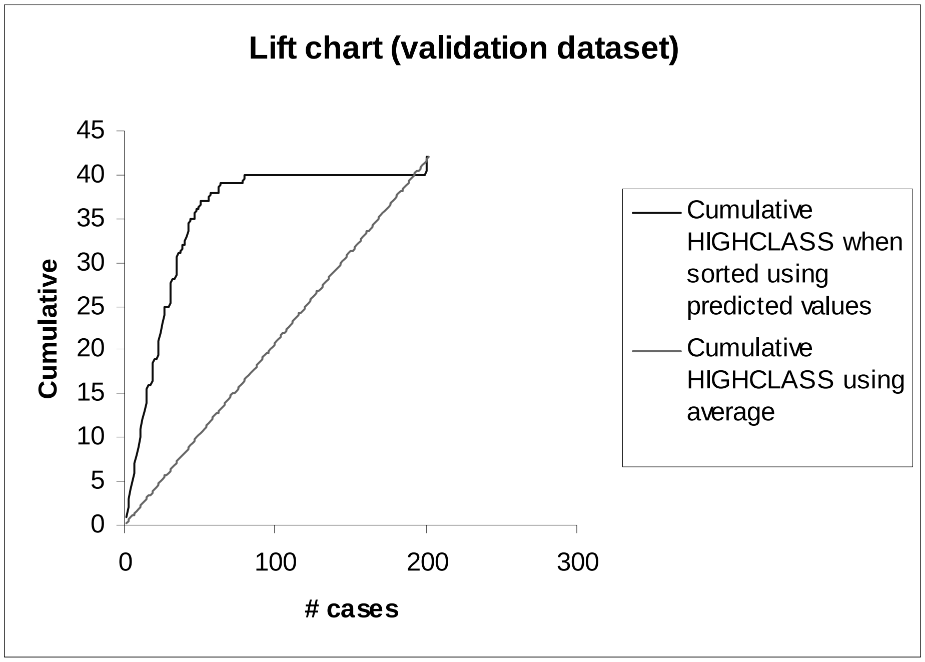
Probability Rank	Predicted Prob. of Success	Actual Value of HICLASS	cumulative Actual Value
1	0.9999	1	1
2	0.9893	1	2
3	0.9884	1	3
4	0.9744	1	4
5	0.9734	1	5
6	0.9715	1	6
7	0.8489	1	7
8	0.6002	0	7
9	0.5218	0	7
10	0.4900	0	7
11	0.2468	0	7
12	0.2156	0	7
13	0.1281	1	8
14	0.0641	0	8
15	0.0130	0	8
16	0.0023	1	9
17	0.0015	0	9
18	0.0001	0	9
19	0.0000	0	9
20	0.0000	0	9
21	0.0000	0	9
22	0.0000	0	9
23	0.0000	0	9
24	0.0000	0	9
25	0.0000	0	9
26	0.0000	0	9
27	0.0000	0	9
28	0.0000	0	9
29	0.0000	0	9
30	0.0000	0	9

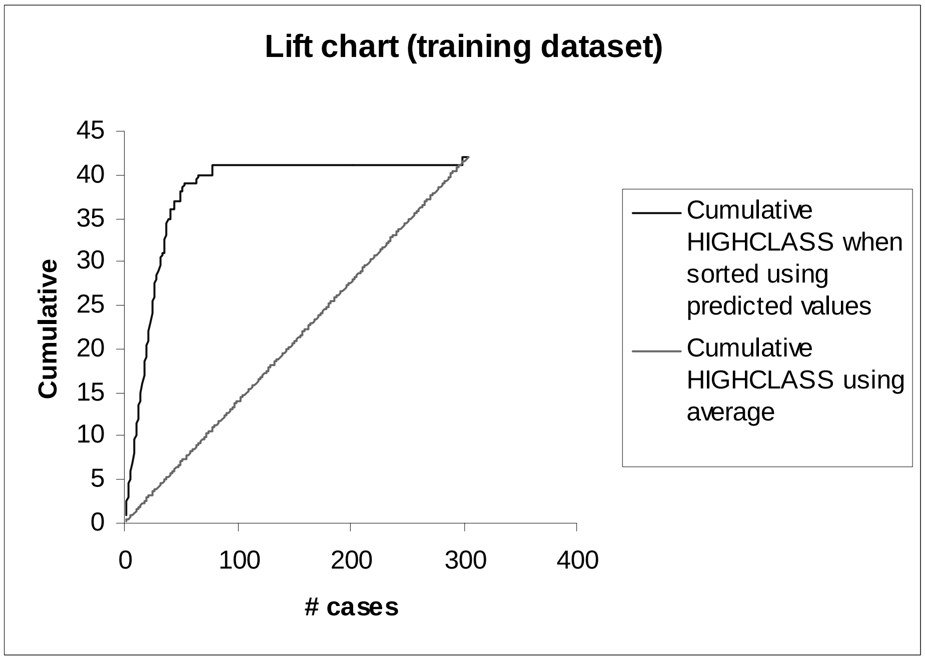
The cumulative lift chart is shown below.



The line joining the points (0,0) to (30,9) is a reference line. It represents the expected number of positives we would predict if we did not have a model but simply selected cases at random. It provides a benchmark against which we can see performance of the model. If we had to choose 10 neighborhoods as HICLASS=1 neighborhoods and used our model to pick the ones most likely to be "1's,", the lift curve tells us that we would be right about 7 of them. If we simply select 10 cases at random we expect to be right for $10 \times 9/30 = 3$ cases. The model gives us a "lift" in predicting HICLASS of $7/3 = 2.33$. The lift will vary with the number of cases we choose to act on. A good classifier will give us a high lift when we act on only a few cases (i.e. use the prediction for the ones at the top). As we include more cases the lift will decrease. The lift curve for the best possible classifier is shown as a broken line.

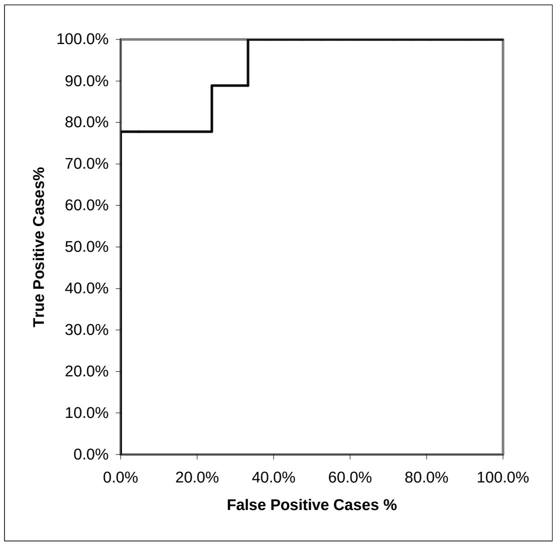
XLMiner automatically creates lift charts from probabilities predicted by logistic regression for both training and validation data. The charts created for the full Boston Housing data are shown below.





It is worth mentioning that a curve that captures the same information as the lift curve in a slightly different manner is also popular in data mining applications. This is the ROC (short for Receiver Operating Characteristic) curve. It uses the same variable on the y axis as the lift curve (but expressed as a percentage of the maximum) and on the x axis it shows the false positives (also expressed as a percentage of the maximum) for differing cut-off levels.

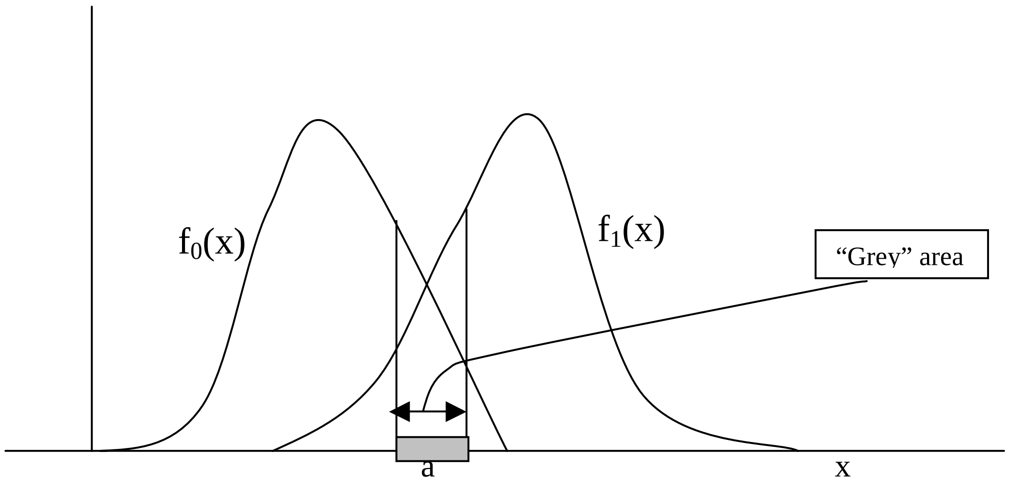
The ROC curve for our 30 cases example above is shown below.



Classification using a Triage strategy

In some cases it is useful to have a "can't say" option for the classifier. In a two-class situation this means that for a case we can make one of three predictions. The case belongs to C_0, or the case belongs to C_1, or we cannot make a prediction because there is not enough information to confidently pick C_0 or C_1. Cases that the classifier cannot classify are subjected to closer scrutiny either by using expert judgment or by enriching the set of predictor variables by gathering additional information that is perhaps more difficult or expensive to obtain. This is analogous to the strategy of triage that is often employed during retreat in battle. The wounded are classified into those who are well enough to retreat, those who are too ill to retreat even if medically treated under the prevailing conditions, and those who are likely to become well enough to retreat if given medical attention. An example is in processing credit card transactions where a classifier may be used to identify clearly legitimate cases and the obviously fraudulent ones while referring the remaining cases to a human decision-maker who may look up a database to form a judgment. Since the vast majority of transactions are legitimate, such a classifier would substantially reduce the burden on human experts.

To gain some insight into forming such a strategy let us revisit the simple two-class, one predictor variable, classifier that we examined at the beginning of this chapter.



Clearly the grey area of greatest doubt in classification is the area around **a**. At **a** the ratio of the conditional probabilities of belonging to the classes is one. A sensible rule way to define the grey area is the set of x values such that:

$$t > \frac{p(C_1) \times f_1(x_0)}{p(C_0) \times f_0(x_0)} > 1/t$$

where **t** is a threshold for the ratio. A typical value of **t** may in the range 1.05 or 1.2.