Machine Learning For Data Science

Unit I

Decision Analytic Thinking I: What Is a Good Model?

Recall from the beginning of Chapter 5: as a manager at MegaTelCo, you wanted to assess whether the model my consulting firm had produced was any good. Overfitting aside, how would you go about measuring that?

For data science to add value to an application, it is important for the data scientists and other stakeholders to consider carefully what they would like to achieve by mining data. This sounds obvious, so it is sometimes surprising how often it is ignored. Both data scientists themselves and the people who work with them often avoid—perhaps without even realizing it—connecting the results of mining data back to the goal of the undertaking. This may manifest itself in the reporting of a statistic without a clear understanding of why it is the right statistic, or in the failure to figuring out how to measure performance in a meaningful way.

We should be careful with such a criticism, though. Often it is not possible to measure perfectly one's ultimate goal, for example because the systems are inadequate, or because it is too costly to gather the right data, or because it is difficult to assess causality. So, we might conclude that we need to measure some surrogate for what we'd really like to measure. It is nonetheless crucial to think carefully about what we'd really like to measure. If we have to choose a surrogate, we should do it via careful, data-analytic thinking.

A challenge with writing a chapter on this topic is that every application is different. We cannot offer a single evaluation metric that is "right" for any classification problem, or regression problem, or whatever problem you may encounter. Nevertheless, there are various common issues and themes in evaluation, and frameworks and techniques for dealing with them.

We will work through a set of such frameworks and metrics for tasks of classification (in this chapter) and instance scoring (e.g., ordering consumers by their likelihood of responding to an offer), and class probability estimation (in the following chapter). The specific techniques should be seen as examples illustrating the general concept of thinking deeply about the needs of the application. Fortunately, these specific techniques do apply quite broadly. We also will describe a very general framework for thinking about evaluation, using expected value, that can cover a very wide variety of applications. As we will show in a later chapter, it also can be used as an organizational tool for data-analytic thinking generally, all the way back to problem formulation.

Evaluating Classifiers

Recall that a classification model takes an instance for which we do not know know the class and predicts its class. Let's consider binary classification, for which the classes often are simply called "positive" and "negative." How shall we evaluate how well such a model performs? In Chapter 5 we discussed how for evaluation we should use a holdout test set to assess the generalization performance of the model. But how should we measure generalization performance?

Sidebar: Bad Positives and Harmless Negatives

In discussing classifiers, we often refer to a bad outcome as a "positive" example, and a normal or good outcome as "negative." This may seem odd to you, given the everyday definitions of positive and negative. Why, for example, is a case of fraud considered positive and a legitimate case considered negative? This terminology is conventional in many fields, including machine learning and data mining, and we will use it throughout this book. Some explanation may be useful to avoid confusion.

It is useful to think of a positive example as one worthy of attention or alarm, and a negative example as uninteresting or benign. For example, a medical test (which is a type of classifier) performed on a biological sample tries to detect disease or an unusual condition by examining certain aspects of the sample. If the test comes back positive it means the disease or condition is present; if the test is negative there is no cause for alarm and usually no need for treatment. Similarly, if a fraud detector finds unusual activity on a customer account and decides there is cause for alarm, this is called a positive. On the other hand, negatives (accounts with only legitimate activity) are prof- itable but from a fraud detection perspective they are unworthy of attention.

There are advantages to maintaining this general orientation rather than redefining the meaning of positive and negative for every domain we introduce. You can think of a classifier as sifting through a large population consisting mostly of negative.

interesting cases—looking for a small number of positive instances. By convention, then, the positive class is often rare, or at least rarer than the negative class. In consequence, the number of mistakes made on negative examples (the false positive errors) may dom- inate, though the cost of each mistake made on a positive example (a false negative error) will be higher.

Plain Accuracy and Its Problems

Up to this point we have assumed that some simple metric, such as classifier error rate or accuracy, was being used to measure a model's performance.

Classification accuracy is a popular metric because it's very easy to measure. Unfortunately, it is usually too simplistic for applications of data mining techniques to real business problems. This section discusses it and some of the alternatives.

The term "classifier accuracy" is sometimes used informally to mean any general measure of classifier performance. Here we will reserve *accuracy* for its specific technical meaning as the proportion of correct decisions:

$$\text{accuracy} = \frac{\text{Number of correct decisions made}}{\text{Total number of decisions made}}$$

This is equal to 1–*error rate*. Accuracy is a common evaluation metric that is often used in data mining studies because it reduces classifier performance to a single number and it is very easy to measure. Unfortunately, it is simplistic and has some well-known problems (Provost, Fawcett, & Kohavi, 1998). To understand these problems we need a way to decompose and count the different types of correct and incorrect decisions made by a classifier. For this we use the confusion matrix.

The Confusion Matrix

To evaluate a classifier properly it is important to understand the notion of *class confusion* and the *confusion matrix*, which is one sort of contingency table. A confusion matrix for a problem involving *n* classes is an *n* × *n* matrix with the columns labeled with actual classes and the rows labeled with predicted classes. Each example in a test set has an actual class label as well as the class predicted by the classifier (the predicted class), whose combination determines which matrix cell the instance counts into. For simplicity we will deal with two-class problems having 2 × 2 confusion matrices.

A confusion matrix separates out the decisions made by the classifier, making explicit how one class is being confused for another. In this way different sorts of errors may be dealt with separately. Let's differentiate between the true classes and the classes predicted by the model by using different symbols. We will consider two-class problems, and will denote the true classes as **p**(ositive) and **n**(egative), and the classes predicted by the model (the "predicted" classes) as **Y**(es) and **N**(o), respectively (think: the model says "**Y**es, it is a positive" or "**N**o, it is not a positive").

Table 7-1. The layout of a 2 × 2 confusion matrix showing the names of the correct predictions (main diagonal) and errors (off-diagonal) entries.

	p	n
Y	True positives	False positives
N	False negatives	True negatives

In the confusion matrix, the main diagonal contains the counts of correct decisions. The errors of the classifier are the **false positives** (negative instances classified as positive) and **false negatives** (positives classified as negative).

Problems with Unbalanced Classes

As an example of how we need to think carefully about model evaluation, consider a classification problem where one class is rare. This is a common situation in applications, because classifiers often are used to sift through a large population of normal or uninteresting entities in order to find a relatively small number of unusual ones; for example, looking for defrauded customers, checking an assembly line for defective parts, or targeting consumers who actually would respond to an offer. Because the unusual or interesting class is rare among the general population, the class distribution is unbalanced or *skewed* (Ezawa, Singh, & Norton, 1996; Fawcett & Provost, 1996; Japkowicz & Stephen, 2002).

Unfortunately, as the class distribution becomes more skewed, evaluation based on accuracy breaks down. Consider a domain where the classes appear in a 999:1 ratio. A simple rule—always choose the most prevalent class—gives 99.9% accuracy. Presumably this is not satisfactory if a nontrivial solution is sought. Skews of 1:100 are common in fraud detection, and skews greater than 1:10^6 have been reported in other classifier learning applications (Clearwater & Stern, 1991; Attenberg & Provost, 2010). Chapter 5 mentioned the "base rate" of a class, which corresponds to how well a classifier would perform by simply choosing that class for every instance. With such skewed domains the base rate for the majority class could be very high, so a report of 99.9% accuracy may tell us little about what data mining has really accomplished.

Even when the skew is not so great, in domains where one class is more prevalent than another accuracy can be greatly misleading. Consider again our cellular-churn example. Let's say you are a manager at MegaTelCo and as an analyst I report that our churn-prediction model generates 80% accuracy. This sounds good, but is it? My coworker reports that her model generates an accuracy of 37%. That's pretty bad, isn't it?

You might say, wait—we need more information about the data. And you would be exactly right to do so (and would be engaging in data-analytic thinking). What do we need? Considering the line of discussion so far in this subsection, you might rightly say: we need to know what is the proportion of churn in the population we are considering. Let's say you know that in these data the baseline churn rate is approximately 10% per month. Let's consider a customer who churns to be a positive example, so within our population of customers we expect a positive to negative class ratio of 1:9. So if we simply classify everyone as negative we could achieve a base rate accuracy of 90%!

Digging deeper, you discover that my coworker and I evaluated on two different datasets. This would not be suprising if we had not coordinated our data analysis efforts. My coworker calculated the accuracy on a representative sample from the population, whereas I created artificially balanced datasets for training and testing (both common practices). Now my coworker's model looks really bad—she could have achieved 90% accuracy, but only got 37%. However, when she applies her model to my balanced data set, she also sees an accuracy of 80%. Now it's really confusing.

The bottom line is that accuracy simply is the wrong thing to measure. In this admittedly contrived example, my coworker's model (call it Model A) achieves 80% accuracy on the balanced sample by correctly identifying all positive examples but only 30% of the negative examples. My model (Model B) does this, conversely, by correctly identifying all the negative examples but only 30% of the positive examples.

Let's look at these two models more carefully, using confusion matrices as a conceptual tool. In a training population of 1,000 customers, the confusion matrices are as follows. Recall that a model's predicted classes are denoted **Y** and **N**.

Table 7-2. Confusion matrix of A

	churn	not churn

Y	500	200
N	0	300

Table 7-3. Confusion matrix of B

	churn	not churn
Y	300	0
N	200	500

Figure 7-1 illustrates these classifications on a balanced population and on a representative population. As mentioned, both models correctly classify 80% of the balanced population, but the confusion matrices and the figure show that they operate very differently. Classifier A often falsely predicts that customers will churn when they will not, while classifier B makes many opposite errors of predicting that customers will not churn when in fact they will. When applied to the original, unbalanced population of customers, model A's accuracy declines to 37% while model B's rises to 93%. This is a huge change. So which model is better?

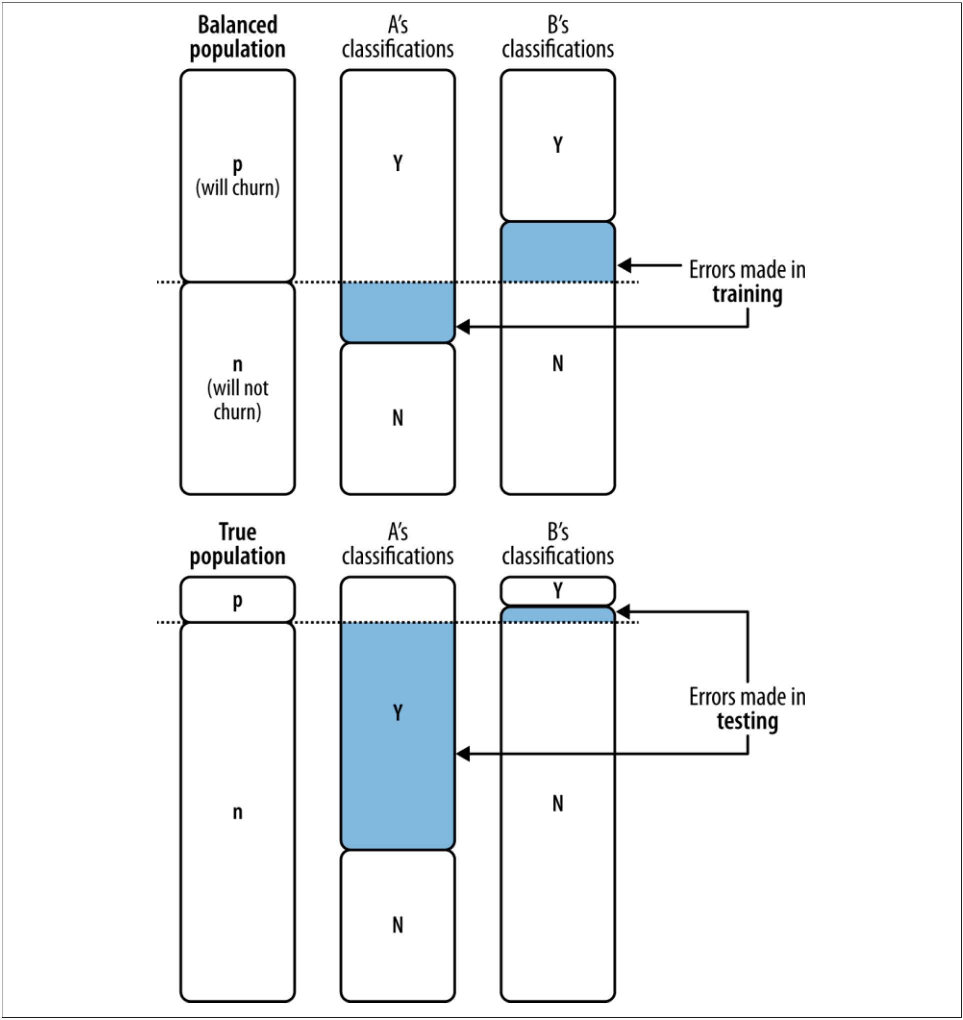

Figure 7-1. Two churn models, A and B, can make an equal number of errors on a balanced population used for training (top) but a very different number of errors when tested against the true population (bottom).

My model (B) now appears to be better than A because B seems to have greater performance on the population we care about—the 1:9 mix of customers we expect to see. But we still can't say for sure because of another problem with accuracy: we don't know how much we *care* about the different errors and correct decisions. This issue is the topic of the next section.

Problems with Unequal Costs and Benefits

Another problem with simple classification accuracy as a metric is that it makes no distinction between false positive and false negative errors. By counting them together, it makes the tacit assumption that both errors are equally important. With real-world domains this is rarely the case. These are typically very different kinds of errors with very different costs because the classifications have consequences of differing severity.

Consider a medical diagnosis domain where a patient is wrongly informed he has cancer when he does not. This is a false positive error. The result would likely be that the patient would be given further tests or a biopsy, which would eventually disconfirm the initial diagnosis of cancer. This mistake might be expensive, inconvenient, and stressful for the patient, but it would not be life threatening. Compare this with the opposite error: a patient who has cancer but she is wrongly told she does not. This is a false negative. This second type of error would mean a person with cancer would miss early detection, which could have far more serious consequences. These two errors are very different, should be counted separately, and should have different costs.

Returning to our cellular-churn example, consider the cost of giving a customer a retention incentive which still results in departure (a false positive error). Compare this with the cost of losing a customer because no incentive was offered (a false negative). Whatever costs you might decide for each, it is unlikely they would be equal; and the errors should be counted separately regardless.

Indeed, it is hard to imagine any domain in which a decision maker can safely be indifferent to whether she makes a false positive or a false negative error. Ideally, we should estimate the cost or benefit of each decision a classifier can make. Once aggregated, these will produce an *expected profit* (or *expected benefit* or *expected cost*) estimate for the classifier.

Generalizing Beyond Classification

We have been using classification modeling to illustrate many data science issues concretely. Most of these issues are applicable beyond classification.

The general principle we're developing here is that when we are applying data science to an actual application it is vital to return to the question: what is important in the application? What is the goal? Are we assessing the results of data mining appropriately given the actual goal?

As another example, let's apply this thinking to regression modeling rather than classification. Say our data science team has to build a movie recommendation model. It predicts how much a given customer will like a given movie, which we will use to help us provide personalized recommendations. Let's say each customer rates a movie by giving it one to five stars, and the recommendation model predicts how many stars a user will give an unseen movie. One of our analysts describes each model by reporting the mean-squared-error (or root-mean-squared-error, or R^2, or whatever metric) for the model. We should ask: mean-squared-error of what? The analyst replies: the value of the target variable, which is the number of stars that a user would give as a rating for the movie. Why is the mean-squared-error on the predicted number of stars an appropriate metric for our recommendation problem? Is it meaningful? Is there a better metric? Hopefully, the analyst has thought this through carefully. It is surprising how often one finds that an analyst has not, and is simply reporting some measure he learned about in a class in school.

A Key Analytical Framework: Expected Value

We now are ready to discuss a very broadly useful conceptual tool to aid data analytic thinking: expected value. The expected value computation provides a framework that is extremely useful in organizing thinking about data-analytic problems. Specifically, it decomposes data-analytic thinking into (i) the structure of the problem, (ii) the elements of the analysis that can be extracted from the data, and (iii) the elements of the analysis that need to be acquired from other sources (e.g., business knowledge of subject matter experts).

In an expected value calculation the possible outcomes of a situation are enumerated. The expected value is then the weighted average of the values of the different possible outcomes, where the weight given to each value is its probability of occurrence. For example, if the outcomes represent different possible levels of profit, an expected profit calculation weights heavily the highly likely levels of profit, while unlikely levels of profit are given little weight. For this book, we will assume that we are considering repeated tasks (like targeting a large number of consumers, or diagnosing a large number of problems) and we are interested in maximizing expected profit.[1]

The expected value framework provides structure to an analyst's thinking (i) via the general form shown in Equation 7-1 .

Equation 7-1. The general form of an expected value calculation

$$EV = p(o_1) \cdot v(o_1) + p(o_2) \cdot v(o_2) + p(o_3) \cdot v(o_3) \ldots$$

Each o_i is a possible decision outcome; $p(o_i)$ is its probability and $v(o_i)$ is its value. The probabilities often can be estimated from the data (ii), but the business values often need to be acquired from other sources (iii). As we will see in Chapter 11, data-driven mod-

1. A course in decision theory would lead you into a thicket of interesting related issues.

eling may help estimate business values, but usually the values must come from external domain knowledge.

We now will illustrate the use of expected value as an analytical framework with two different data science scenarios. The two scenarios are often confused but it is vital to be able to distinguish them. To do so, recall the difference from Chapter 2 between the *mining* (or induction) of a model, and the model's *use*.

Using Expected Value to Frame Classifier Use

In use, we have many individual cases for which we would like to predict a class, which may then lead to an action. In targeted marketing, for example, we may want to assign each consumer a class of *likely responder* versus *not likely responder*, then we could target the likely responders. Unfortunately, for targeted marketing often the probability of response for any individual consumer is very low—maybe one or two percent—so no consumer may seem like a likely responder. If we choose a "common sense" threshold of 50% for deciding what a likely responder is, we would probably not target anyone. Many inexperienced data miners are surprised when the application of data mining models results in everybody being classified as *not likely responder* (or a similar negative class).

However, with the expected value framework we can see the crux of the problem. Let's walk through a targeted marketing scenario.[2] Consider that we have an offer for a product that, for simplicity, is only available via this offer. If the offer is not made to a consumer, the consumer will not buy the product. We have a model, mined from historical data, that gives an estimated probability of response $p_R()$ for any consumer whose feature vector description **x** is given as input. The model could be a classification tree or a logistic regression model or some other model we haven't talked about yet. Now we would like to decide whether to target a particular consumer described by feature vector **x**.

Expected value provides a framework for carrying out the analysis. Specifically, let's calculate the expected benefit (or cost) of targeting consumer **x**:

$$\text{Expected benefit of targeting} = p_R() \cdot v_R + 1 - [p_R() \cdot v_{NR}]$$

where v_R is the value we get from a response and v_{NR} is the value we get from no response. Since everyone either responds or does not, our estimate of the probability of not responding is just $(1 - p_R())$. As mentioned, the probabilities came from the historical

2. We use targeted marketing here, rather than the churn example, because the expected value framework actually reveals an important complexity to the churn example that we're not ready to deal with. We will return to that later in Chapter 11 when we're ready to deal with it.

data, as summarized in our predictive model. The benefits v_R and v_{NR} need to be determined separately, as part of the Business Understanding step (recall Chapter 2). Since a customer can only purchase the product by responding to the offer (as discussed above), the expected benefit of not targeting her conveniently is zero.

To be concrete, let's say that a consumer buys the product for $200 and our product-related costs are $100. To target the consumer with the offer, we also incur a cost. Let's say that we mail some flashy marketing materials, and the overall cost including postage is $1, yielding a value (profit) of v_R = $99 if the consumer responds (buys the product). Now, what about v_{NR}, the value to us if the consumer does not respond? We still mailed the marketing materials, incurring a cost of $1 or equivalently a benefit of -$1.

Now we are ready to say precisely whether we want to target this consumer: do we expect to make a profit? Technically, is the expected value (profit) of targeting greater than zero? Mathematically, this is:

$$p_R() \cdot \$99 - 1 [p_R() \cdot \$1] > 0$$

A little rearranging of the equation gives us a decision rule: Target a given customer **x** only if:

$$p_R() \cdot \$99 > 1[- p_R() \cdot]\$1 \; p_R() >$$

0.01

With these example values, we should target the consumer as long as the estimated probability of responding is greater than 1%.

This shows how an expected value calculation can express how we will *use* the model. Making this explicit helps to organize problem formulation and analysis. We will return to this in Chapter 11. Now, let's move on to the other important application of the expected value framework, to organize our analysis of whether the model we have induced from the data is any good.

Using Expected Value to Frame Classifier Evaluation

At this point we want to shift our focus from individual decisions to collections of decisions. Specifically, we need to evaluate the set of decisions made by a model when applied to a set of examples. Such an evaluation is necessary in order to compare one model to another. For example, does our data-driven model perform better than the hand-crafted model suggested by the marketing group? Does a classification tree work better than a linear discriminant model for a particular problem? Do any of the models do substantially better than a baseline "model," such as randomly choosing consumers to target? It is likely that each model will make some decisions better than the other model. What we care about is, *in aggregate*, how well does each model do: what is its *expected* value.

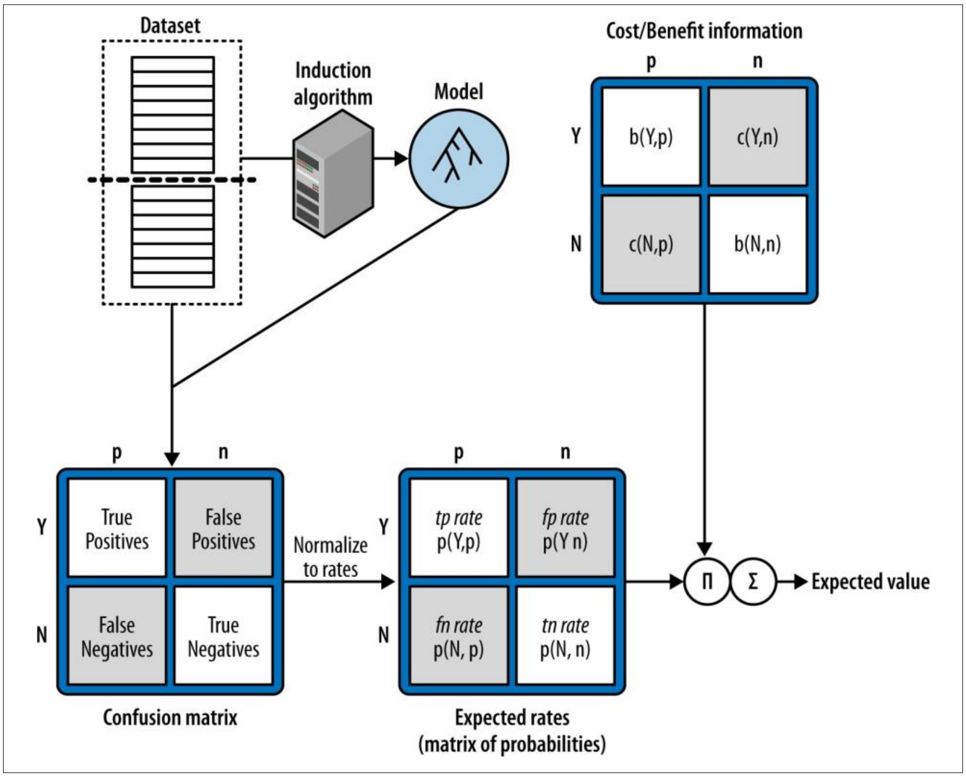

Figure 7-2. A diagram of the expected value calculation. The Π and Σ refer to the multiplication and summation in the expected value calculation.

We can use the expected value framework just described to determine the best decisions for each particular model, and then use the expected value in a different way to compare the models. If we are to calculate the expected profit for a model in aggregate, each o_i in Equation 7-1 corresponds to one of the possible combinations of the class we predict, and the actual class. We want to aggregate all the different possible cases: overall, when we decide to target consumers, what is the probability that they respond? What is the probability that they do not? What about when we do not target consumers, would they have responded? Fortunately, as you may recall, we already have the counts necessary to calculate all these—in the confusion matrix. Each o_i corresponds to one cell of the confusion matrix. For example, what is the probability associated with the particular combination of a consumer being *predicted to churn* and *actually does not churn*? That would be estimated by the number of test-set consumers who fell into the confusion matrix cell (**Y**,**n**), divided by the total number of test-set consumers.

Let's walk through an entire expected profit calculation at the aggregate (model) level, in the process computing these probabilities. Figure 7-2 shows a schematic diagram of the expected value calculation in the context of model induction and evaluation. At the top left of the diagram, a training portion of a dataset is taken as input by an induction algorithm, which produces the model that we will evaluate. That model is applied to a holdout (test) portion of the data, and the counts for the different cells of the confusion matrix are tallied. Let's consider a concrete example of a classifier confusion matrix in Table 7-4.

Table 7-4. A sample confusion matrix with counts.

	p	n
Y	56	7
N	5	42

Error rates

When calculating expected values for a business problem, the analyst is often faced with the question: where do these probabilities actually come from? When evaluating a model on testing data, the answer is straightforward: these probabilities (of errors and correct decisions) can be estimated from the tallies in the confusion matrix by computing the rates of the errors and correct decisions. Each cell of the confusion matrix contains a count of the number of decisions corresponding to the corresponding combination of (predicted, actual), which we will express as *count*(*h*,*a*) (we use h for "hypothesized" since p is already being used). For the expected value calculation we reduce these counts to rates or estimated probabilities, *p(h,a)*. We do this by dividing each count by the total number of instances:

$$p(h\,,\ a) = count(h\,,\ a)\,/\,T$$

Here are the calculations of the rates for each of the raw statistics in the confusion matrix. These rates are estimates of the probabilities that we will use in the expected value computation of Equation 7-1.

$T = 110$

$p(\mathbf{Y},\mathbf{p}) = 56/110 = 0.51 \quad p(\mathbf{Y},\mathbf{n}) = 7/110 = 0.06$

$p(\mathbf{N},\mathbf{p}) = 5/110 = 0.05 \quad p(\mathbf{N},\mathbf{n}) = 42/110 = 0.38$

Costs and benefits

To compute expected profit (recall Equation 7-1), we also need the cost and benefit values that go with each decision pair. These will form the entries of a cost-benefit matrix with the same dimensions (rows and columns) as the confusion matrix. However, the cost-benefit matrix specifies, for each (predicted,actual) pair, the cost or benefit of making such a decision (see Figure 7-3). Correct classifications (true positives and true negatives) correspond to the benefits *b*(**Y**,**p**) and *b*(**N**, **n**), respectively. Incorrect classifications (false positives and false negatives) correspond to the "benefit" *b*(**Y**,**n**) and *b*(**N**, **p**), respectively, which may well actually be a cost (a negative benefit), and often are explicitly referred to as costs *c*(**Y**, **n**) and *c*(**N**, **p**).

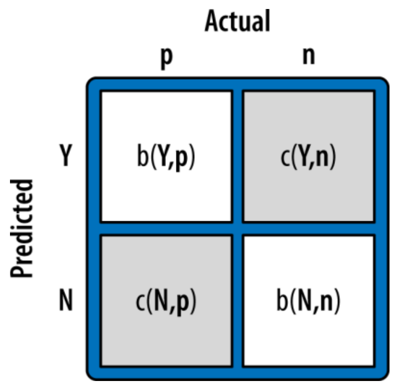

Figure 7-3. A cost-benefit matrix.

While the probabilities can be estimated from data, *the costs and benefits often cannot*. They generally depend on external information provided via analysis of the consequences of decisions in the context of the specific business problem. Indeed, specifying the costs and benefits may take a great deal of time and thought. In many cases they cannot be specified exactly but only as approximate ranges. Chapter 8 will return to address what we might do when these values are not known exactly. For example, in our churn problem, how much is it really worth us to retain a customer? The value depends on future cell phone usage and probably varies a great deal between customers. It may be that data on customers' prior usage can be helpful in this estimation. In many cases, average estimated costs and benefits are used rather than individual-specific costs and benefits, for simplicity of problem formulation and calculation. Therefore, we will ignore customer-specific cost/benefit calculations for the rest of our example, but will return to it in Chapter 11.

So, let's return to our targeted marketing example. What are the costs and benefits? We will express all values as *benefits*, with costs being negative benefits, so the function we're specifying is *b*(predicted, actual). For simplicity, all numbers will be expressed as dollars.

- A *false positive* occurs when we classify a consumer as a likely responder and therefore target her, but she does not respond. We've said that the cost of preparing and mailing the marketing materials is a fixed cost of \$1 per consumer. The benefit in this case is negative: $b(\mathbf{Y}, \mathbf{n}) = -1$.

- A *false negative* is a consumer who was predicted not to be a likely responder (so was not offered the product), but would have bought it if offered. In this case, no money was spent and nothing was gained, so $b(\mathbf{N}, \mathbf{p}) = 0$.
- A *true positive* is a consumer who is offered the product and buys it. The benefit in this case is the profit from the revenue (\$200) minus the product-related costs (\$100) and the mailing costs (\$1), so $b(\mathbf{Y}, \mathbf{p}) = 99$.
- A *true negative* is a consumer who was not offered a deal and who would not have bought it even if it had been offered. The benefit in this case is zero (no profit but no cost), so $b(\mathbf{N}, \mathbf{n}) = 0$.

These cost-benefit estimations can be summarized in a 2 × 2 cost-benefit matrix, as in Figure 7-4. Note that the rows and columns are the same as for our confusion matrix, which is exactly what we'll need to compute the overall expected value for the classification model.

$$\begin{array}{cc|cc} & & \multicolumn{2}{c}{\text{Actual}} \\ & & \mathbf{p} & \mathbf{n} \\ \hline \text{Predicted} & \mathbf{Y} & 99 & -1 \\ & \mathbf{N} & 0 & 0 \end{array}$$

Figure 7-4. A cost-benefit matrix for the targeted marketing example.

Given a matrix of costs and benefits, these are multiplied cell-wise against the matrix of probabilities, then summed into a final value representing the total expected profit. The result is:

$$\begin{aligned} \textit{Expected profit} = {} & p(,\ ) \cdot b(,\ ) + p(,\ ) \cdot b(,\ ) + \\ & p(,\ ) \cdot b(,\ ) + p(,\ ) \cdot b(,\ ) \end{aligned}$$

Using this equation, we can now compute and compare the expected profits for various models and other targeting strategies. All we need is to be able to compute the confusion matrices over a set of test instances, and to generate the cost-benefit matrix.

This equation is sufficient for comparing classifiers, but let's continue along this path a little further, because an alternative calculation of this equation is often used in practice. This alternative view is closely related to some techniques used to visualize classifier performance (see Chapter 8). Furthermore, by examining the alternative formulation we can see exactly how to deal with the model comparison problem we introduced at the beginning of the chapter—where one analyst had reported performance statistics over a representative (but unbalanced) population, and another had used a class-balanced population.

A common way of expressing expected profit is to factor out the probabilities of seeing each class, often referred to as the *class priors*. The class priors, *p*(**p**) and *p*(**n**), specify the likelihood of seeing positive and negative instances, respectively. Factoring these out allows us to separate the influence of class imbalance from the fundamental predictive power of the model, as we will discuss in more detail in Chapter 8.

A rule of basic probability is:

$$p(x,\ y) = p(y) \cdot p(x \mid y)$$

This says that the probability of two different events both occurring is equal to the probability of one of them occurring times the probability of the other occurring if we know that the first occurs. Using this rule we can re-express our expected profit as:

$$\begin{aligned}\text{Expected profit} \quad = \quad & p(\ |\ ) \cdot p() \cdot b(,\ ) + p(\ |\ ) \cdot p() \cdot b(,\ ) + \\ & p(\ |\ ) \cdot p() \cdot b(,\ ) + p(\ |\ ) \cdot p() \cdot b(,\ )\end{aligned}$$

Factoring out the class priors *p*(**y**) and *p*(**n**), we get the final equation:

*Equation 7-2. Expected profit equation with priors p(**p**) and p(**n**) factored.*

$$\begin{aligned}\text{Expected profit} \quad = \quad & p() \ \cdot \ \ p(\ |\ ) \cdot b(,\ ) + p(\ |\ ) \cdot c(,\ ) \ + \qquad] \\ & p() \ \cdot \ \ p(\ |\ ) \cdot b(,\ ) + p(\ |\ ) \cdot c(,\ ) \qquad]\end{aligned}$$

From this mess, notice that we now have one component (the first one) corresponding to the expected profit from the positive examples, and another (the second one) corresponding to the expected profit from the negative examples. Each of these is weighted by the probability that we see that sort of example. So, if positive examples are very rare, their contribution to the overall expected profit will be correspondingly small.[3] In this alternative formulation, the quantities *p*(**Y**|**p**), *p*(**Y**|**n**), etc. correspond directly to the true positive rate, the false positive rate, etc., that also can be calculated directly from the confusion matrix (see "Sidebar: Other Evaluation Metrics" on page 203).

3. This could be extended to any number of classes, though to keep things simple(r) we'll use two.Here again is our sample confusion matrix in Table 7-5.

Table 7-5. Our sample confusion matrix (raw counts)

	p	n
Y	56	7
N	5	42

Table 7-6 shows the class priors and various error rates we need.

Table 7-6. The class priors and the rates of true positives, false positives, and so on.

T = 110	
P = 61	N = 49
$p(\mathbf{p}) = 0.55$	$p(\mathbf{n}) = 0.45$
tp rate = 56/61 = 0.92	*fp rate* = 7/49 = 0.14
fn rate = 5/61 = 0.08	*tn rate* = 42/49 = 0.86

Returning to the targeted marketing example, what is the expected profit of the model learned? We can calculate it using Equation 7-2:

$$
\begin{aligned}
\text{expected profit} &= p() \cdot \; p(\,|\,) \cdot b(,\,) + p(\,|\,) \cdot c(,\,) \;\; + \qquad\qquad] \\
&\quad\; p() \cdot \; p(\,|\,) \cdot b(,\,) + p(\,|\,) \cdot c(,\,) \qquad\qquad] \\
&= 0.55 \cdot [0.92 \cdot b(,\,) + 0.08 \cdot b(,\,) \;\; + 0.45 \cdot 0.86 \\
&\quad\; \cdot b(,\,) + 0.14 \cdot p(,\,) \qquad] \\
&= 0.55 \cdot [0.92 \cdot 99 + 0.08 \cdot 0] + \\
&\quad\; 0.45 \cdot [0.86 \cdot 0 + 0.14 \cdot \; -1\;] \\
&= 50.1 - 0.063 \\
&\approx \mathbf{\$50.04}
\end{aligned}
$$

This expected value means that if we apply this model to a population of prospective customers and mail offers to those it classifies as positive, we can expect to make an average of about $50 profit per consumer.

We now can see one way to deal with our motivating example from the beginning of the chapter. Instead of computing accuracies for the competing models, we would compute expected values. Furthermore, using this alternative formulation, we can compare the two models even though one analyst tested using a representative distribution and the other tested using a class-balanced distribution. In each calculation, we simply can replace the priors. Using a balanced distribution corresponds to priors of $p(\mathbf{p}) = 0.5$ and $p(\mathbf{n}) = 0.5$. The mathematically savvy reader is encouraged to convince herself that the other factors in the equation will not change if the test-set priors change.

To close this section on estimated profit, we emphasize two pitfalls that are common when formulating cost-benefit matrices:

- It is important to make sure the signs of quantities in the cost-benefit matrix are consistent. In this book we take benefits to be positive and costs to be negative. In many data mining studies, the focus is on minimizing cost rather than maximizing profit, so the signs are reversed. Mathematically, there is no difference. However, it is important to pick one view and be consistent.
- An easy mistake in formulating cost-benefit matrices is to "double count" by putting a benefit in one cell and a negative cost *for the same thing* in another cell (or vice versa). A useful practical test is to compute the *benefit improvement* for changing the decision on an example test instance.

  For example, say you've built a model to predict which accounts have been defrauded. You've determined that a fraud case costs $1,000 on average. If you decide that the benefit of catching fraud is therefore +$1,000/case on average, *and* the cost of missing fraud is -$1,000/case, then what would be the *improvement in benefit* for catching a case of fraud? You would calculate:

  $b(\mathbf{Y},\mathbf{p}) - b(\mathbf{N},\mathbf{p}) = \$1000 - (-\$1000) = \$2000$

  But intuitively you know that this improvement should only be about $1,000, so this error indicates double counting. The solution is to specify either that the benefit of catching fraud is $1,000 *or* that the cost of missing fraud is -$1,000, but not both. One should be zero.+ *FP*), which is the accuracy over the cases predicted to be positive. The *F-measure* isthe harmonic mean of precision and recall at a given point, and is:

$$\text{F-measure} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Practitioners in many fields such as statistics, pattern recognition, and epidemiology speak of the sensitivity and specificity of a classifier:

$$\text{Sensitivity} = TN / (TN + FP) = \text{True negative rate} = 1 - \text{False positive rate}$$

$$\text{Specificity} = TP / (TP + FN) = \text{True positive rate}$$

You may also hear about the *positive predictive value*, which is the same as precision.

Accuracy, as mentioned before, is simply the count of correct decisions divided by the total number of decisions, or:

$$Accuracy = \frac{TP + TN}{P + N}$$

Swets (1996) lists many other evaluation metrics and their relationships to the confusion matrix.

Evaluation, Baseline Performance, and Implications for Investments in Data

Up to this point we have talked about model evaluation in isolation. In some cases just demonstrating that a model generates *some* (nonzero) profit, or a positive return on investment, will be informative by itself. Nevertheless, another fundamental notion in data science is: *it is important to consider carefully what would be a reasonable baseline against which to compare model performance.* This is important for the data science team in order to understand whether they indeed are improving performance, and is equally important for demonstrating to stakeholders that mining the data has added value. So, what is an appropriate baseline for comparison?

The answer of course depends on the actual application, and coming up with suitable baselines is one task for the business understanding phase of the data mining process. However, there are some general principles that can be very helpful.

For classification models it is easy to simulate a completely random model and measure its performance. The visualization frameworks we will discuss in Chapter 8 have natural baselines showing what random classification should achieve. This is useful for verydifficult problems or initial explorations. Comparison against a random model estab- lishes that there is some information to be extracted from the data.

However, beating a random model may be easy (or may seem easy), so demonstrating superiority to it may not be very interesting or informative. A data scientist will often need to implement an alternative model, usually one that is simple but not simplistic, in order to justify continuing the data mining effort.

In Nate Silver's book on prediction, *The Signal and the Noise* (2012), he mentions the baseline issue with respect to weather forecasting:

> There are two basic tests that any weather forecast must pass to demonstrate its merit: It must do better than what meteorologists call persistence: the assumption that the weather will be the same tomorrow (and the next day) as it was today. It must also beat climatology, the long-term historical average of conditions on a particular date in a particular area.

In other words, weather forecasters have two simple—but not simplistic—baseline models that they compare against. One (persistence) predicts that the weather tomorrow is going to be whatever it was today. The other (climatology) predicts whatever the average historical weather has been on this day from prior years. Each model performs considerably better than random guessing, and both are so easy to compute that they make natural baselines of comparison. Any new, more complex model must beat these.

What are some general guidelines for good baselines? For classification tasks, one good baseline is the *majority classifier*, a naive classifier that always chooses the majority class of the training dataset (see Note: Base rate in "Holdout Data and Fitting Graphs" on page 113). This may seem like advice so obvious it can be passed over quickly, but it is worth spending an extra moment here. There are many cases where smart, analytical people have been tripped up in skipping over this basic comparison. For example, an analyst may see a classification accuracy of 94% from her classifier and conclude that it is doing fairly well—when in fact only 6% of the instances are positive. So, the simple majority prediction classifier also would have an accuracy of 94%. Indeed, many beginning data science students are surprised to find that the models they build from the data simply predict everything to be of the majority class. Note that this may make sense if the modeling procedure is set up to build maximum accuracy models—it may be hard to beat 94% accuracy. The answer, of course, is to apply the central idea of this chapter: consider carefully what is desired from the data mining results. Maximizing simple prediction accuracy is usually not an appropriate goal. If that's what our algorithm is doing, we're using the wrong algorithm. For regression problems we have a directly analogous baseline: predict the average value over the population (usually the mean or median).

In some applications there are multiple simple averages that one may want to combine. For example, when evaluating recommender systems that internally predict how many "stars" a particular customer would give to a particular movie, we have the average number of stars a movie gets across the population (how well liked it is) and the average number of stars a particular customer gives to movies (what that customer's overall bias is). A simple prediction based on these two may do substantially better than using one or the other in isolation.

Moving beyond these simple baseline models, a slightly more complex alternative is a model that only considers a very small amount of feature information. For example, recall from Chapter 3 our very first example of a data mining procedure: finding informative variables. If we find the one variable that correlates best with the target, we can build a classification or regression model that uses just that variable, which gives another view of baseline performance: how well does a simple "conditional" model perform? "Conditional" here means that it predicts differently based on, or conditioned on, the value of the feature(s). The overall population average is therefore sometimes called the "unconditional" average.

One example of mining such single-feature predictive models from data is to use tree induction to build a "**decision stump**" — a decision tree with only one internal node, the root node. A tree limited to one internal node simply means that the tree induction selects the single most informative feature to make a decision. In a well-known paper in machine learning, Robert Holte (1993) showed that decision stumps often produce quite good baseline performance on many of the test datasets used in machine learning research. A decision stump is an example of the strategy of choosing the single most informative piece of information available (recall Chapter 3) and basing all decisions on it. In some cases most of the leverage may be coming from a single feature, and this method assesses whether and to what extent that is the case.

This idea can be extended to data *sources*, and relates to our fundamental principle from Chapter 1 that we should regard data as an asset to be invested in. If you are considering building models that integrate data from various sources, you should compare the result to models built from the individual sources. Often there are substantial costs to acquiring new sources of data. In some cases these are actual monetary costs; in other cases they involve commitments of personnel time for managing relationships with data providers and monitoring data feeds. To be thorough, for each data source the data science team should compare a model that uses the source to one that does not. Such comparisons help to justify the cost of each source by quantifying its value. If the contribution is negligible, the team may be able to reduce costs by eliminating it.

Beyond comparing simple models (and reduced-data models), it is often useful to implement simple, inexpensive models based on domain knowledge or "received wisdom" and evaluate their performance. For example, in one fraud detection application it was commonly believed that most defrauded accounts would experience a sudden increase in usage, and so checking accounts for sudden jumps in volume was sufficient for catching a large proportion of fraud. Implementing this idea was straightforward and it provided a useful baseline for demonstrating the benefit of data mining. (This essentially was a single-feature predictive model.) Similarly, an IBM team that used data mining to direct sales efforts chose to implement a simple sales model that prioritized existing customers by the size of previous revenue and other companies by annual sales. [4] They were able to demonstrate that their data mining added significant value beyond this simpler strategy. Whatever the data mining group chooses as a baseline for comparison, it should be something the stakeholders find informative, and hopefully persuasive.