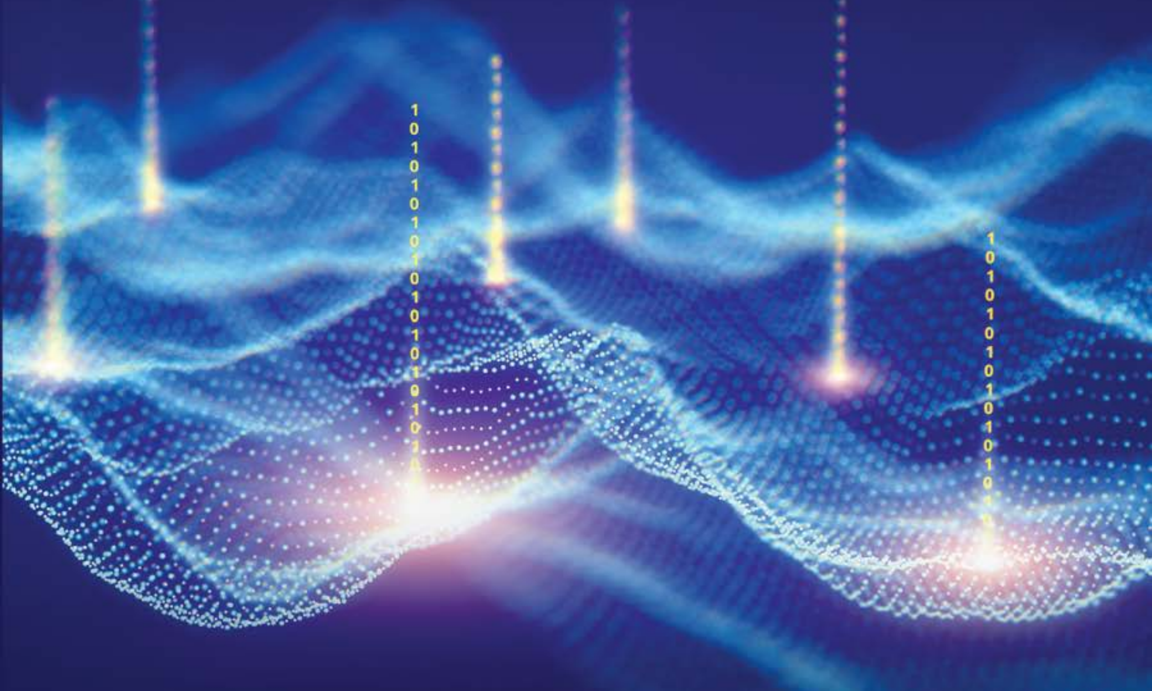
PEARSON BUSINESS ANALYTICS SERIES

SECOND EDITION

PREDICTIVE ANALYTICS

DATA MINING, MACHINE LEARNING AND DATA SCIENCE FOR PRACTITIONERS



DURSUN DELEN

Open for Innovation
KNIME

Predictive Analytics, Second Edition

Area Under the ROC Curve

The area under the ROC (receiver operating characters—a term borrowed from radar detection) curve is a graphical assessment technique for binary classification problems, in which the true positive rate (i.e., sensitivity) is plotted on the *y*-axis, and the false positive rate (1: specificity) is plotted on the *x*-axis. Because AUC (area under the ROC curve) is scale-agnostic/invariant, it can be used to objectively rank multiple prediction methods. Although it is a very good metric for judging how well each prediction model performs and helps rank order the model types, it does not provide an absolute value for the models' predictive accuracy. That is, the AUC value is not to be confused with the overall prediction accuracy of the model. (However, as you will deduce from the experimentations, the AUC value and overall accuracy values will be highly correlated: The model types that produce higher values of AUC will most likely also create higher overall accuracy values.) AUC would be a very good metric to use in a marketing campaign, for instance, because you could use the predictions ranked by probability to create an ordered list of users to whom to send the marketing promotions.

Another benefit of using AUC is that it is classification-threshold invariant; that is, it measures the quality of the model's prediction power, regardless of what classification threshold is chosen. In contrast, F1 score or overall accuracy both are prone to the choice of threshold value. Specifically, AUC determines the overall unbiased predictive power of a classifier, wherein the AUC value 1 indicates a perfect classifier, and the value 0.5 indicates no predictive power (i.e., no better than random chance); however, in reality, the AUC values would range between these two extreme cases, 0.5 and 1.0. For example, in Figure 4.7, ROC curve *A* has a better AUC and hence better classification performance than *B* (the AUC of which is illustrated with the blue shaded area), while *C* is the baseline, representing the ROC curve not any better than random chance. Although in Figure 4.7 the three lines are shown perfectly smooth for purposes of illustration, in real-world applications, the ROC curves look rather rugged and discrete.

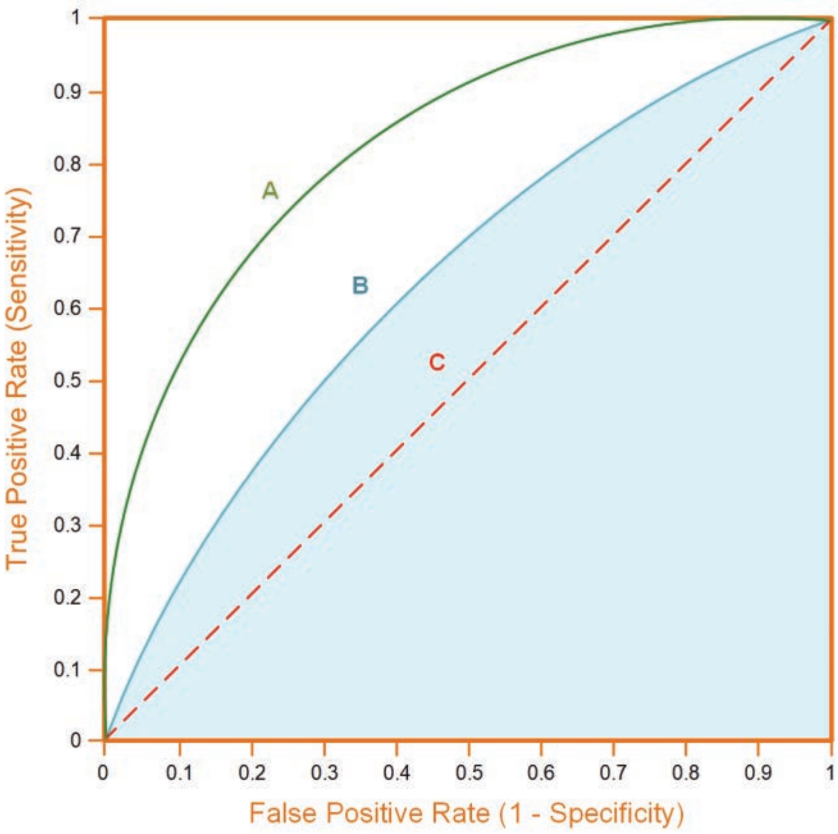


Figure 4.7 An Illustration of Three Areas Under the ROC Curves

Other Cross-Validation Methodologies

Although they are not as common as the techniques described so far in this chapter, some other assessment techniques can also be used for classification-type problems. The following are some of the exemplary ones:

- **Leave-one-out.** The leave-one-out method is similar to a *k*-fold cross-validation, where *k* takes the value 1; that is, every data point is used for testing once on as many models developed as there are data points. This is a time-consuming methodology, but it can be a viable option for small data sets.

- **Bootstrapping.** With bootstrapping, a fixed number of instances from the original data are sampled (with replacement) for training, and the rest of the data set is used for testing. This process is repeated as many times as desired.
- **Jackknifing.** This methodology is similar to the leave-one-out methodology; you can use it to calculate accuracy by leaving out one sample at each iteration of the estimation process.

Classification Methods

Many methods and algorithms are used for classification modeling, including the following:

- **Decision tree analysis.** Decision tree analysis (a machine learning technique) is arguably the most popular classification technique in the data mining arena. The following section provides a detailed description of this technique.
- **Statistical analysis.** For many years—until the emergence of machine learning techniques—statistical techniques were the primary classification algorithms. Statistical classification techniques include *logistic regression* and *discriminant analysis*, both of which make the assumptions that the relationships between the input and output variables are linear in nature, the data is normally distributed, and the variables are not correlated and are independent of each other. The questionable nature of these assumptions has led to the shift toward machine learning techniques.
- **Neural networks.** Using neural networks is among the most popular machine learning techniques that can be used for

classification-type problems. A detailed description of this technique is presented in Chapter 5.

- **Support vector machines.** Along with neural networks, support vector machines are becoming increasingly popular as powerful classification algorithms. A detailed description of this technique is presented in Chapter 5.
- **Ensemble models.** Either homogeneous or heterogeneous ensemble models can be used for better predictive ability and robust performance. Chapter 5 provides a detailed description of model ensembles.
- ***k*-nearest-neighbor algorithm.** This deceptively simple yet highly efficient algorithm uses similarity as the basis for its classification method. Chapter 5 provides a detailed description of this technique.
- **Case-based reasoning.** This approach, which is conceptually similar to the nearest-neighbor algorithm, uses historical cases to recognize commonalities in order to assign a new case to the most probable category.
- **Bayesian classifiers.** This approach uses probability theory to build classification models based on past occurrences that are capable of placing a new instance into a most probable class (or category).
- **Genetic algorithms.** Genetic algorithms use the analogy of natural evolution to build directed-search-based mechanisms to classify data samples.
- **Rough sets.** This method takes into account the partial membership of class labels to predefined categories in building models (collection of rules) for classification problems.

Complete coverage of all these classification techniques is beyond the scope of this book, but the following section describes the most popular one, decision trees, and Chapter 5 covers several of the others (the most notable ones).

Decision Trees

Before describing the details of decision trees, we need to discuss some simple terminology. First, decision trees include many input variables that may have an impact on the classification of different patterns. These input variables are usually called *attributes*. For example, if we were to build a model to classify loan risks on the basis of just two characteristics—income and credit rating—these two characteristics would be the attributes, and the resulting output would be the class label (e.g., low, medium, or high risk). Second, a tree consists of branches and nodes. A branch represents the outcome of a test to classify a pattern (on the basis of a test), using one of the attributes. A leaf node at the end represents the final class choice for a pattern (a chain of branches from the root node to the leaf node, which can be represented as a complex if–then statement).

The basic idea behind a decision tree is that it recursively divides a training set until each division consists entirely or primarily of examples from one class. Each nonleaf node of the tree contains a split point, which is a test on one or more attributes and determines how the data is to be divided further. Decision tree algorithms, in general, build an initial tree from the training data such that each leaf node is pure, and they then prune the tree to increase its generalization and, hence, the prediction accuracy of the test data.

In the growth phase, the tree is built by recursively dividing the data until each division is either pure (i.e., contains members of the same class) or relatively small. The basic idea is to ask questions whose answers provide the most information, as is done in the game "Twenty Questions."

The split used to partition the data depends on the type of attribute used in the split. For a continuous attribute A, splits are of the following form:

value(A) *x*

where *x* is some "optimal" split value of A. For example, the split based on income could be Income > 50000. For the categorical attribute A, splits are of the following form:

value(A) belongs to *x*

where *x* is a subset of A. For example, the split could be on the basis of education (whether the individual has a college degree).

A general algorithm for building a decision tree is as follows:

1. Create a root node and assign all the training data to it.
2. Select the best splitting attribute.
3. Add a branch to the root node for each value of the split. Split the data into mutually exclusive (i.e., non-overlapping) subsets along the lines of the specific split of the branches.
4. Repeat steps 2 and 3 for each leaf node until the stopping criteria is reached (e.g., the node is dominated by a single class label).

Many different algorithms have been proposed for creating decision trees. These algorithms differ primarily in terms of the way in which they determine the splitting attribute (and its split values), the order of splitting the attributes (splitting the same attribute only once or many times), the number of splits at each node (binary versus ternary), the stopping criteria, and the pruning of the tree (pre- versus postpruning). Some of the most well-known algorithms are ID3 (followed by C4.5 and C5 as the improved versions of ID3) from machine learning, Classification and Regression Trees (CART) from statistics, and the Chi-squared Automatic Interaction Detector (CHAID) from pattern recognition.