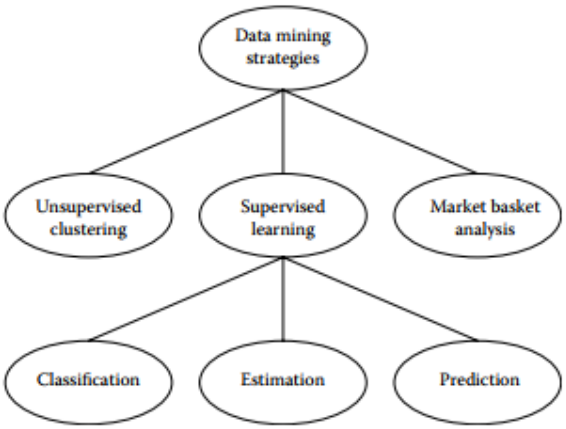
Data Mining Strategies

Data Mining Strategies

Data mining strategies can be broadly classified as either supervised or unsupervised. Supervised learning builds models by using input attributes to predict output attribute values. Many supervised data mining algorithms only permit a single output attribute. Other supervised learning tools allow us to specify one or several output attributes. Output attributes are also known as *dependent variables* as their outcome depends on the values of one or more input attributes. Input attributes are referred to as *independent variables*. When learning is unsupervised, an output attribute does not exist. Therefore, all attributes used for model building are independent variables.

Supervised learning strategies can be further labeled according to whether output attributes are discrete or categorical, as well as by whether models are designed to determine a current condition or predict a future outcome. In this section, we examine three supervised learning strategies, take a closer look at unsupervised clustering, and introduce a strategy for discovering associations among retail items sold in catalogs and stores. The figure shows the five data mining strategies.

Hierarchy of Data Mining Strategies



Classification

Classification is probably the best understood of all data mining strategies. Classification tasks have three common characteristics:

- Learning is supervised.
- The dependent variable is categorical.
- The emphasis is on building models able to assign new instances to one of a set of well-defined classes.

Some example classification tasks include the following:

- Determine those characteristics that differentiate individuals who have suffered a heart attack from those who have not.
- Develop a profile of a “successful” person.
- Determine if a credit card purchase is fraudulent.
- Classify a car loan applicant as a good or a poor credit risk.
- Determine if an income tax submission has been falsified.

Notice that each example deals with current rather than future behavior. For example, we want the car loan application model to determine whether an applicant is a good credit risk at this time rather than in some future time period. Prediction models are designed to answer questions about future behavior. Prediction models are discussed later in this section.

Estimation

Like classification, the purpose of an estimation model is to determine a value for an unknown output attribute. However, unlike classification, the output attribute for an estimation problem is numeric rather than categorical. Here are four examples of estimation tasks:

- Estimate the number of minutes before a thunderstorm will reach a given location.
- Estimate the return on investment for an advertising campaign.
- Estimate the likelihood that a credit card has been stolen.
- Estimate the length of a gamma-ray burst.

Most supervised data mining techniques are able to solve classification or estimation problems, but not both. If our data mining tool supports one strategy but not the other, we can usually adapt a problem for solution by either strategy. To illustrate, suppose the output attribute for the original training data in the aforementioned example of the stolen credit card is a numeric value between 0 and 1, with 1 being a most likely case for a stolen card. We can make discrete categories for the output attribute values by replacing scores ranging between 0.0 and 0.3 with the value unlikely, scores between 0.3 and 0.7 with likely, and scores greater than 0.7 with highly likely. In this case, the transformation between numeric values and discrete categories is straightforward. Cases such as attempting to make monetary amounts discrete present more of a challenge.

Prediction

It is not easy to differentiate prediction from classification or estimation. However, unlike a classification or estimation model, the purpose of a predictive model is to determine future outcome rather than current behavior. The output attribute(s) of a predictive model can be categorical or numeric. Here are several examples of tasks appropriate for predictive data mining:

- Predict the total number of touchdowns an NFL running back will score during the 2017 NFL season
- Determine whether a credit card customer is likely to take advantage of a special offer made available with his/her credit card billing
- Predict next week's closing price for the Dow Jones Industrial Average
- Forecast which cell phone subscribers are likely to change providers during the next three months

Most supervised data mining techniques appropriate for classification or estimation problems can also build predictive models. Actually, it is the nature of the data that determines whether a model is suitable for classification, estimation, or prediction. To show this, let's consider the Cardiology Patient Dataset described in the highlighted box above. The data set contains 303 instances. Of these, 165 hold information about patients who are free of heart disease. The remaining 138 instances contain data about patients who have a known heart condition. The attributes and possible attribute values associated with this data set are shown in Table below. Two forms of the data set exist. One data set consists of all numeric attributes. The second data set has categorical conversions for seven of the original numeric attributes. The table column labeled mixed values shows the value numeric for attributes that were not converted to a categorical equivalent. For example, the values for attribute age are identical for both data sets. However, the attribute fasting blood sugar <120 has values true or false in the converted data set and values 0 or 1 in the original data.

The Table below lists four instances from the mixed form of the data set. Two of the instances represent typical examples from each respective class. The remaining two instances are atypical class members. Some differences between a typical healthy and a typical sick patient are easily anticipated, for example, resting ECG and induced angina. Surprisingly, we do not see expected differences in cholesterol and blood pressure readings between the typical and atypical healthy individuals.

Here are two rules generated from this data set by RapidMiner's Subgroup Discovery operator. Class is specified as the output attribute:

IF Maximum Heart Rate > = 158.333
THEN Class = Healthy
Rule precision: 75.60%
Rule coverage: 40.60%

Cardiology Patient Data

Attribute Name	Mixed Values	Numeric Values	Comments
Age	Numeric	Numeric	Age in years
Gender	Male, Female	1, 0	Patient gender
Chest pain type	Angina, abnormal angina, NoTang, asymptomatic	1–4	NoTang = nonanginal pain
Blood pressure	Numeric	Numeric	Resting blood pressure upon hospital admission
Cholesterol	Numeric	Numeric	Serum cholesterol
Fasting blood sugar <120	True, false	1, 0	Is fasting blood sugar less than 120?
Resting electrocardiogram (ECG)	Normal, abnormal, Hyp	0, 1, 2	Hyp = left ventricular hypertrophy
Maximum heart rate	Numeric	Numeric	Maximum heart rate achieved
Induced angina?	True, false	1, 0	Does the patient experience angina as a result of exercise?
Old peak	Numeric	Numeric	ST depression induced by exercise relative to rest
Slope	Up, flat, down	1–3	Slope of the peak exercise ST segment
Number of colored vessels	0, 1, 2, 3	0, 1, 2, 3	Number of major vessels colored by fluoroscopy
Thalassemia	Normal, fix, rev	3, 6, 7	Normal, fixed defect, reversible defect
Concept class	Healthy, sick	1, 0	Angiographic disease status

Typical and Atypical Instances from the Cardiology Domain

Attribute Name	Typical Healthy Class	Atypical Healthy Class	Typical Sick Class	Atypical Sick Class
Age	52	63	60	62
Gender	Male	Male	Male	Female
Chest pain type	NoTang	Angina	Asymptomatic	Asymptomatic
Blood pressure	138	145	125	160
Cholesterol	223	233	258	164
Fasting blood sugar <120	False	True	False	False
Resting ECG	Normal	Hyp	Hyp	Hyp
Maximum heart rate	169	150	141	145
Induced angina?	False	False	True	False
Old peak	0	2.3	2.8	6.2
Slope	Up	Down	Flat	Down
Number of colored vessels	0	0	1	3
Thalassemia	Normal	Fix	Rev	Rev

IF Thal = Rev
THEN Class = Sick
Rule precision: 76.30%
Rule coverage: 38.90%

Let's consider the first rule. Rule coverage gives us the percent of data instances that satisfy the rule's preconditions. Rule coverage reveals that of the 303 individuals represented in the data set, 40.6% or 123 patients have a maximum heart rate greater than 158.333. Rule precision tells us that of the 123 individuals with maximum heart rate > = 158.333, 75.6% or 93 are healthy patients. That is, if a patient in the data set has a maximum heart rate greater than 158.333, we will be correct more than 75 times out of 100 in identifying the patient as healthy. When we combine this knowledge with the maximum heart rate values shown in the Table, we are able to conclude that healthy patients are likely to have higher maximum heart rate values. Is this first rule appropriate for classification or prediction?

If the rule is predictive, we can use the rule to warn healthy folks with the following statement:

Warning 1: Have your maximum heart rate checked on a regular basis. If your maximum heart rate is low, you may be at risk of having a heart attack!

If the rule is appropriate for classification but not prediction, the scenario reads as follows:

Warning 2: If you have a heart attack, expect your maximum heart rate to decrease.

In any case, we cannot imply the following stronger statement:

Warning 3: A low maximum heart rate will cause you to have a heart attack!

That is, with data mining, we can state relationships between attributes, but we cannot say whether the relationships imply causality. Therefore, entertaining an exercise program to increase maximum heart rate may or may not be a good idea.

The question still remains as to whether either of the first two warnings is correct. This question is not easily answered. A data mining specialist can develop models to generate rules such as those just given. Beyond this, the specialist must have access to additional information—in this case, a medical expert—before determining how to use discovered knowledge.

Unsupervised Clustering

With unsupervised clustering, we are without a dependent variable to guide the learning process. Rather, the learning program builds a knowledge structure by using some measure of cluster quality to group instances into two or more classes. A primary goal of an unsupervised clustering strategy is to discover concept structures in data.

Common uses of unsupervised clustering include the following:

- Detect fraud in stock trading activity, insurance claims, or financial transactions
- Determine if meaningful relationships in the form of concepts can be found in data representing gamma-ray burst flashes outside of our solar system
- Determine if publicly available data about banks that have failed is useful for building a supervised model to predict bank failure
- Determine a best set of input attributes for building a supervised learner model to identify cell phone customers likely to churn

To illustrate this idea, let's suppose we have built a supervised learner model using the heart patient data with output attribute class. To evaluate the supervised model, we present the training instances to an unsupervised clustering system. The attribute class is flagged as unused. Next, we examine the output of the unsupervised model to determine if the instances from each concept class (healthy and sick) naturally cluster together. If the instances from the individual classes do not cluster together, we may conclude that the attributes are unable to distinguish healthy patients from those with a heart condition. This being the case, the supervised model is likely to perform poorly. One solution is to choose a best set of attributes for a

supervised learner model by repeatedly applying unsupervised clustering with alternative attribute choices. In this way, those attributes best able to differentiate the classes known to be present in the data can be determined. Unfortunately, even with a small number of attribute choices, the application of this technique can be computationally unmanageable.

Unsupervised clustering can also help detect any atypical instances present in the data, that is, instances that do not group naturally with the other instances. Atypical instances are referred to as outliers. Outliers can be of great importance and should be identified whenever possible. With data mining, the outliers might be just those instances we are trying to identify. For example, an application that checks credit card purchases would likely identify an outlier as a positive instance of credit card fraud.

Market Basket Analysis

The purpose of market basket analysis is to find interesting relationships among retail products. The results of a market basket analysis help retailers design promotions, arrange shelf or catalog items, and develop cross-marketing strategies. Association rule algorithms, described later in this chapter, are often used to apply a market basket analysis to a set of data.

Supervised Data Mining Techniques

Now that we have described five common data mining strategies, we turn our attention to the data mining techniques used to apply these strategies to a set of data. A specific data mining technique is defined by an algorithm and an associated knowledge structure such as a tree or a set of rules.

The Credit Card Promotion Database

We will use the fictitious data displayed in the Table below and summarized in the box titled The Credit Card Promotion Database to help explain the data mining strategies presented here. The table shows data extracted from a database containing information collected on individuals who hold credit cards issued by the Acme Credit Card Company. The first row of the Table contains the attribute names for each column of data. The first column is for instance identification. The second column gives the salary range for an individual credit card holder. Values in columns 3–5 tell us which cardholders have taken advantage of specified promotions sent with their monthly credit card bill. Column 6 tells us whether an individual has credit card insurance. Column 7 gives the gender of the cardholder, and column 8 offers the cardholder's age. The first cardholder shown in the table has a yearly salary between $40,000 and $50,000, is a 45-year-old male, has purchased one or several magazines advertised with one of his credit cards bills, did not take advantage of any other credit card promotions, and does not have credit card insurance. Several attributes likely to be relevant for data mining purposes are not included in the table. Some of these attributes are promotion dates, dollar amounts for purchases, average monthly credit card balance, and marital status. Let's turn our attention to the data mining techniques to see what they can find in the credit card promotion database.

ID	Income Range ($)	Magazine Promotion	Watch Promotion	Life Insurance Promotion	Credit Card Insurance	Gender	Age
1	40–50K	Yes	No	No	No	Male	45
2	30–40K	Yes	Yes	Yes	No	Female	40
3	40–50K	No	No	No	No	Male	42
4	30–40K	Yes	Yes	Yes	Yes	Male	43
5	50–60K	Yes	No	Yes	No	Female	38
6	20–30K	No	No	No	No	Female	55
7	30–40K	Yes	No	Yes	Yes	Male	35
8	20–30K	No	Yes	No	No	Male	27
9	30–40K	Yes	No	No	No	Male	43
10	30–40K	Yes	Yes	Yes	No	Female	41
11	40–50K	No	Yes	Yes	No	Female	43
12	20–30K	No	Yes	Yes	No	Male	29
13	50–60K	Yes	Yes	Yes	No	Female	39
14	40–50K	No	Yes	No	No	Male	55
15	20–30K	No	No	Yes	Yes	Female	19

Rule-Based Techniques

Any decision tree can be translated into a set of production rules. However, there are several other rule generation techniques. One class of rule generators uses what is referred to as a covering approach. Instances are covered by a rule if they satisfy the rule's preconditions. For each class, the goal is to create a set of rules that maximizes the total number of covered within-class instances while at the same time minimizing the number of covered nonclass instances.

Another class of rule generators focuses on finding interesting subgroups of instances within the data. RapidMiner's Subgroup Discovery operator falls into this category. Earlier, we showed you two rules generated by this operator for the heart patient data set. Here we take a closer look at the subgroup discovery method and concentrate on the covering rule approach.

Let's assume the Acme Credit Card Company has authorized a new life insurance promotion similar to the previous life insurance promotion specified in the Table. The promotion material will be sent as part of the credit card billing for all cardholders with a nonzero balance. We will use data mining to help us send billings to a select group of individuals who do not have a current credit card balance but are likely to take advantage of the promotion.

Our problem calls for a supervised approach using life insurance promotion as the out-put attribute. Our goal is to develop a profile for individuals likely to take advantage of a life insurance promotion advertised along with their next credit card statement. Here is a possible hypothesis:

A combination of one or more of the data set attributes differentiates between Acme Credit Card Company cardholders who have taken advantage of a life insurance promotion and those cardholders who have chosen not to participate in the promotional offer.

The hypothesis is stated in terms of current rather than predicted behavior. However, the nature of the created rules will tell us whether we can use the rules for classification or prediction.

For our experiment, we presented RapidMiner's Subgroup Discovery operator with the data in the Table . We obtained several rules of interest. Here are four such rules:

1. IF Gender=Female
   THEN Life Insurance Promotion=Yes
   Rule precision: 85.7%
   Rule coverage: 46.7%

2. IF Gender=Male and Income Range=40–50K
   THEN Life Insurance Promotion=No
   Rule precision: 100.00%
   Rule coverage: 20.00%

3. IFCredit Card Insurance=Yes
   THEN Life Insurance Promotion=Yes
   Rule precision: 100.00%
   Rule coverage: 20%

4. IF Credit Card Insurance=No and Watch Promotion=No
   THEN Life Insurance Promotion=No
   Rule precision: 80.00%
   Rule coverage: 33.33%

The first rule tells us that it would be advantageous to send a credit card bill containing the promotion to all female cardholders. The second rule indicates that males who make between $40,000 and $50,000 a year are not good candidates for the promotion. The 100.00% precision value tells us that our sample does not contain a single male within the $40,000–$50,000 income range who took advantage of the previous life insurance promotion.

The first and second rules are particularly helpful, as neither rule contains an antecedent condition involving a previous promotion. The rule preconditions are based purely on information obtained at the time of initial application. As credit card insurance is always initially offered upon a card approval, the third rule is also useful. However, the fourth rule will not be applicable to new cardholders who have not had a chance to take advantage of a previous promotion. For new cardholders,

we should consider the first three rules as predictive and the fourth rule as effective for classification but not predictive purposes. Furthermore, it is important to point out that the precision values shown with each rule are based solely on the training data. The true test of whether these rules are useful comes when they are applied to new data.

Before leaving our discussion about rule-based techniques, two added points are in order. First, we have yet to address the issue of how rules are generated for numeric data. The basic technique is to discretize numeric data into a specified number of equal-sized bins where each bin represents one categorical value for the given attribute. Several rule generators have one or several built-in numeric discretization routines. The subgroup operator discussed here requires discretization to take place during preprocessing.

Secondly, each of the aforementioned rules displays a value for precision and coverage. Where rules are concerned, rule precision and rule accuracy are often considered as interchangeable terms. That is, both terms are used to represent the ratio of the total number of correctly classified instances to the total number of instances covered by the rule antecedent. However, RapidMiner's subgroup operator makes a notable distinction between rule accuracy and rule precision. Specifically, whereas rule precision is only concerned with rule performance relative to the instances covered by the rule, rule accuracy is measured using all instances within the data set.

To illustrate this, consider a data set having *N* instances. Suppose class C1 has rule Recovering *P* of its *T*1 instances (*P* < = T1). We also have class C2 with *T*2 = (*N* - T1) instances. Suppose R's antecedent condition covers *Q* of the instances within C2. The computation for precision is then given as (*P*/(*P* + *Q*). That is, the precision of rule R is the total number of C 1 instances covered by R divided by the total number of data set instances covered by R's antecedent condition. However, in contrast, rule accuracy is measured by the value of (*P* + *T*2 - *Q*)/*N*. That is, accuracy is defined as the sum of the number of C1 instances covered by R and the number of C2 instances whose antecedent condition is not covered by R, divided by the total number of instances. The rationale is that a rule is accurate relative to an instance of the competing class if the instance does not satisfy the rule's preconditions.

As an example, consider rule 1 stating that female instances accepted the life insurance promotion. Upon examining Table 2.3, we see that six of the seven females in the data set accepted the life insurance promotion. Therefore, rule precision is 6/7, which approximates to 85.7%. To compute rule accuracy, we add the total number of instances from the competing class (male instances) who did not accept the life insurance promotion to the numerator and divide by 15. As instance numbers 1, 3, 8, 9, and 14 represent males not accepting the life insurance promotion, our computation becomes (6 + 5)/15, giving us a rule accuracy of 73.3%. We will return to this issue in Chapter 5 when we discuss rule generation with RapidMiner.

Neural Networks

A *neural network* is a set of interconnected nodes designed to imitate the functioning of the human brain. As the human brain contains billions of neurons and a typical neural network has fewer than 100 nodes, the comparison is somewhat superficial. However, neural networks have been successfully applied to problems across several disciplines and for this reason are quite popular in the data mining community.

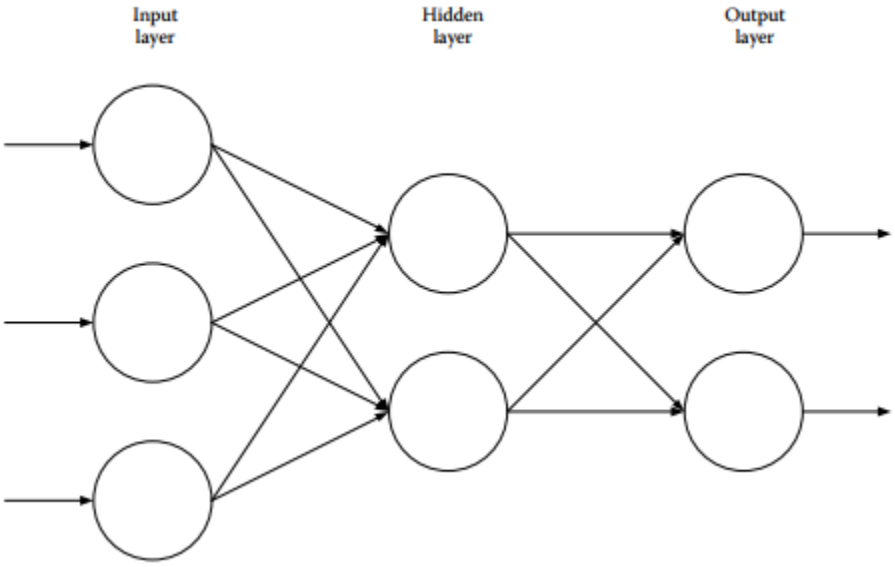
Neural networks come in many shapes and forms and can be constructed for supervised learning as well as unsupervised clustering. In all cases, the values input into a neural network must be numeric. The feed-forward network is a popular supervised learner model. Figure 2.2 shows a fully connected feed-forward neural network consisting of three layers. With a feed-forward network, the input attribute values for an individual instance enter at the input layer and pass directly through the output layer of the network structure. The output layer may contain one or several nodes. The output layer of the network shown in the Figure contains two nodes. Therefore, the output of the neural network will be an ordered pair of values. The network displayed in the Figure is fully connected, as all the nodes at one layer are connected to all nodes at the next layer. In addition, each network node connection has an associated weight (not shown in the diagram). Notice that nodes within the same layer of the network architecture are not connected to one another.

Neural networks operate in two phases. The first phase is called the learning phase. During network learning, the input values associated with each instance enter the network at the input layer. One input-layer node exists for each input attribute contained in the data. The actual output value for each instance is computed and compared with the desired network output. Any error between the desired and computed output is propagated back through the network by changing connection-weight values. Training terminates after a certain number of iterations or when the network converges to a predetermined minimum error rate. During the second phase of operation, the network weights are fixed, and the network is used to classify new instances.

We applied a supervised network model to the credit card promotion data to test the aforementioned hypothesis. Once again, *life insurance promotion* was designated as the output attribute. Because we wanted to construct a predictive model, the input attributes were limited to *income range, credit card insurance, gender,* and *age*. Therefore, the network architecture contained four input nodes and one output node. For our experiment, we chose five hidden-layer nodes. Because neural networks cannot accept categorical data, we transformed categorical attribute values by replacing *yes* and *no* with 1 and 0 respectively, *male* and *female* with 1 and 0, and income range values with the lower end of each range score.

Computed and actual values for the output attribute *life insurance promotion* are shown in Table. Notice that in most cases, a computed output value is within .03 of the actual value. To use the trained network to classify a new unknown instance, the attribute values for the unknown instance are passed through the network, and an output score is obtained.

Fully Connected Multilayer Neural Network



Neural Network Traning: Actual and Computed Output

Instance Number	Life Insurance Promotion	Computed Output
1	0	0.024
2	1	0.998
3	0	0.023
4	1	0.986
5	1	0.999
6	0	0.050
7	1	0.999
8	0	0.262
9	0	0.060
10	1	0.997
11	1	0.999
12	1	0.776
13	1	0.999
14	0	0.023
15	1	0.999

If the computed output value is closer to 0, we predict the instance to be an unlikely candidate for the life insurance promotion. A value closer to 1 shows the unknown instance as a good candidate for accepting the life insurance promotion. A major shortcoming of the neural network approach is a lack of explanation about what has been learned. Converting categorical data to numerical values can also be a challenge.

Statistical Regression

Statistical regression is a supervised learning technique that generalizes a set of numeric data by creating a mathematical equation relating one or more input attributes to a single numeric output attribute. A *linear regression* model is a type of statistical regression characterized by an output attribute whose value is determined by a linear sum of weighted input attribute values. Here is a linear regression equation for the data in the Table below:

$$\text{Life insurance promotion} = 0.5909 \times (\text{credit card insurance}) - 0.5455 \times (\text{gender}) + 0.7727$$

Notice that life insurance promotion is the attribute whose value is to be determined by a linear combination of attributes credit card insurance and gender. As with the neural network model, we transformed all categorical data by replacing yes and no with 1 and 0, male and female with 1 and 0, and income range values with the lower end of each range score.

To illustrate the use of the equation, suppose we wish to determine if a female who does not have credit card insurance is a likely candidate for the life insurance promotion. Using the equation, we have

$$\text{Life insurance promotion} = 0.5909(0) - 0.5455(0) + 0.7727 = 0.7727$$

Because the value 0.7727 is close to 1.0, we conclude that the individual is likely to take advantage of the promotional offer. Although regression can be nonlinear, the most popular use of regression is for linear modeling. Linear regression is appropriate provided that the data can be accurately modeled with a straight-line function. In Chapter 11, we experiment with Weka's and RapidMiner's linear regression models.

ASSOCIATION RULES

As the name implies, association rule mining techniques are used to discover interesting associations between attributes contained in a database. Unlike traditional production rules, association rules can have one or several output attributes. Also, an output attribute for one rule can be an input attribute for another rule. Association rules are a popular technique for market basket analysis because all possible combinations of potentially interesting product groupings can be explored. For this reason, a limited number of attributes are able to generate hundreds of association rules. We applied the Apriori association rule algorithm described by Agrawal et al. (1993) to the data in the Table below. The algorithm examines baskets of items and generates rules for those baskets containing a minimum number of items. The Apriori algorithm does not process numerical data. Therefore, before application of the algorithm, we transformed the attribute age to the following set of discrete categories: over15, over20, over30, over40, and over50. To illustrate, an individual with age = over40 is between the ages of 40 and 49 inclusive. Once again, we limited the choice of attributes to income range, credit card insurance, sex, and age. Here is a list of three association rules generated by the Apriori algorithm for the data in the Table below.

ID	Income Range ($)	Magazine Promotion	Watch Promotion	Life Insurance Promotion	Credit Card Insurance	Gender	Age
1	40–50K	Yes	No	No	No	Male	45
2	30–40K	Yes	Yes	Yes	No	Female	40
3	40–50K	No	No	No	No	Male	42
4	30–40K	Yes	Yes	Yes	Yes	Male	43
5	50–60K	Yes	No	Yes	No	Female	38
6	20–30K	No	No	No	No	Female	55
7	30–40K	Yes	No	Yes	Yes	Male	35
8	20–30K	No	Yes	No	No	Male	27
9	30–40K	Yes	No	No	No	Male	43
10	30–40K	Yes	Yes	Yes	No	Female	41
11	40–50K	No	Yes	Yes	No	Female	43
12	20–30K	No	Yes	Yes	No	Male	29
13	50–60K	Yes	Yes	Yes	No	Female	39
14	40–50K	No	Yes	No	No	Male	55
15	20–30K	No	No	Yes	Yes	Female	19

1. IF Gender = Female and Age = over40 and Credit Card Insurance = No
   THEN Life Insurance Promotion = Yes

2. IF Gender = Male and Age = over40 and Credit Card Insurance = No
   THEN Life Insurance Promotion = No

3. IF Gender = Female and Age = over40
   THEN Credit Card Insurance = No and Life Insurance Promotion = Yes

Each of these three rules has an accuracy of 100% and covers exactly 20% of all data instances. For rule 3, the 20% rule coverage tells us that one in every five individuals is a female over the age of 40 who does not have credit card insurance and has life insurance obtained through the life insurance promotional offer. Notice that in rule 3, credit card insurance and life insurance promotion are both output attributes. A problem with association rules is that along with potentially interesting rules, we are likely to see several rules of little value. In Chapter 3, we will explore this issue in more detail and describe the Apriori algorithm for generating association rules. Chapter 4 and Chapter 5 offer tutorials on using the association rule generators available with Weka and RapidMiner. The next section continues our discussion by exploring unsupervised clustering techniques.

CLUSTERING TECHNIQUES

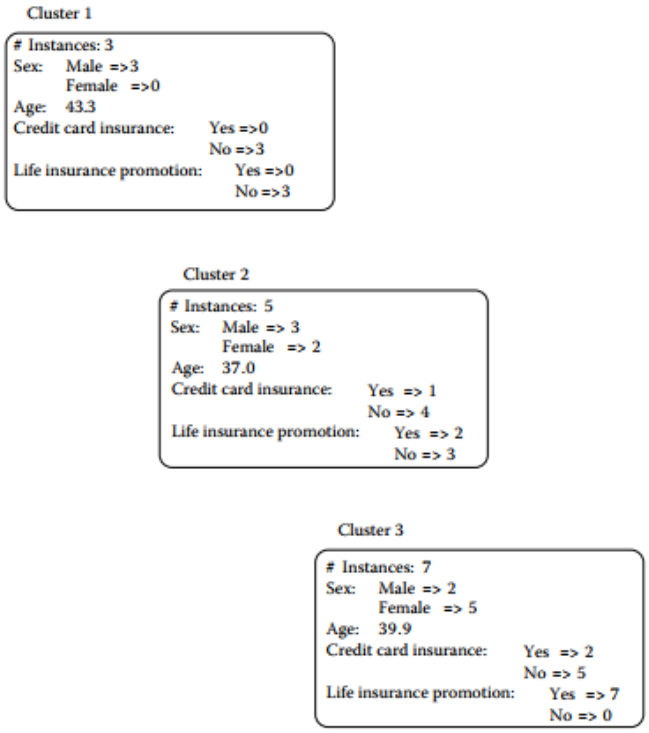
Several unsupervised clustering techniques are commonly used. One technique is to apply some measure of similarity to divide instances into disjoint partitions. The partitions are generalized by computing a group mean for each cluster or by listing a most typical subset of instances from each cluster.

We applied a hierarchical clustering model to the data in Table 2.3. Our choice for input attributes was again limited to income range, credit card insurance, gender, and age. We set the life insurance promotion as a label attribute, meaning that although the attribute is not used by the clustering system, it will appear as part of the summary statistics. As learning is unsupervised, our hypothesis needs to change. Here is a possible hypothesis that is consistent with our theme of determining likely candidates for the life insurance promotion:

By applying unsupervised clustering to the instances of the Acme Credit Card Company database, we will find a subset of input attributes that differentiate cardholders who have taken advantage of the life insurance promotion from those cardholders who have not accepted the promotional offer. As you can see, we are using unsupervised clustering to find a

best set of input attributes for differentiating current customers who have taken advantage of the special promotion from those who have not. Once we determine a best set of input attributes, we can use the attributes to develop a supervised model for predicting future outcomes. To test the hypothesis, we applied unsupervised clustering to the data several times until we found a set of input attributes that resulted in clusters that differentiate the two classes. The results of one such clustering are displayed in Figure 2.3. The figure indicates that three clusters were formed. As you can see, the three individuals represented in cluster 1 did not take advantage of the life insurance promotion. Two of the individuals in cluster 2 took

An Unsupervised Clustering of the Credit Card Database



advantage of the promotion, and three did not. Finally, all seven individuals in cluster 3 purchased the life insurance promotion. It is clear that two of the three clusters differentiate individuals who took advantage of the promotion from those who did not. This result offers positive evidence that the attributes used for the clustering are viable choices for building a predictive supervised learner model. In the next section, we lay the foundation for evaluating the performance of supervised and unsupervised learner models.

EVALUATING PERFORMANCE

Performance evaluation is probably the most critical of all the steps in the data mining process. In this section, we offer a commonsense approach to evaluating supervised and unsupervised learner models. In later chapters, we will concentrate on more formal evaluation techniques. As a starting point, we pose three general questions:

1. Will the benefits received from a data mining project more than offset the cost of the data mining process?
2. How do we interpret the results of a data mining session?
3. Can we use the results of a data mining process with confidence?

All three questions are difficult to answer. However, the first is more of a challenge because several factors come into play. Here is a minimal list of considerations for the first question:

1. Is there knowledge about previous projects similar to the proposed project? What are the success rates and costs of projects similar to the planned project?
2. What is the current form of the data to be analyzed? Do the data exist, or will they have to be collected? When a wealth of data exists and are not in a form amenable for data mining, the greatest project cost will fall under the category of data preparation. In fact, a larger question may be whether to develop a data warehouse for future data mining projects.

3. Who will be responsible for the data mining project? How many current employees will be involved? Will outside consultants be hired?
4. Is the necessary software currently available? If not, will the software be purchased or developed? If purchased or developed, how will the software be integrated into the current system?

As you can see, any answer to the first question requires knowledge about the business model, the current state of available data, and current resources. Therefore, we will turn our attention to providing evaluation tools for questions 2 and 3. We first consider the evaluation of supervised learner models.

Evaluating Supervised Learner Models

Supervised learner models are designed to classify, estimate, and/or predict future outcome. For some applications, the desire is to build models showing consistently high predictive accuracy. The following three applications focus on classification correctness:

- Develop a model to accept or reject credit card applicants
- Develop a model to accept or reject home mortgage applicants
- Develop a model to decide whether or not to drill for oil

Classification correctness is best calculated by presenting previously unseen data in the form of a test set to the model being evaluated. Test set model accuracy can be summarized in a table known as a confusion matrix. To illustrate, let's suppose we have three possible classes: C1, C2, and C3. A generic confusion matrix for the three-class case is shown in the Table below.

Values along the main diagonal give the total number of correct classifications for each class. For example, a value of 15 for C11 means that 15 class C1 test set instances were

Three-Class Confusion Matrix

	C_1	C_2	C_3
C_1	C_{11}	C_{12}	C_{13}
C_2	C_{21}	C_{22}	C_{23}
C_3	C_{31}	C_{32}	C_{33}

correctly classified. Values other than those on the main diagonal represent classification errors. To illustrate, suppose C12 has the value 4. This means that four class C1 instances were incorrectly classified as belonging to class C2. The following four observations may be helpful in analyzing the information in a confusion matrix:

1. For any given cell Cij, the subscript i represents the actual class, and j indicates the computed class.
2. Values along the main diagonal represent correct classifications. For the matrix in the Table above, the value C11 represents the total number of class C1 instances correctly classified by the model. A similar statement can be made for the values C22 and C33.
3. Values in row C i represent those instances that belong to class Ci. For example, with i = 2, the instances associated with cells C21, C22, and C23 are all actually members of C 2. To find the total number of C2 instances incorrectly classified as members of another class, we compute the sum of C21 and C23.
4. Values found in column C i indicate those instances that have been classified as members of C i. With i = 2, the instances associated with cells C12, C22, and C32 have been classified as members of class C. To find the total number of instances incorrectly classified as members of class C 2, we compute the sum of C12 and C32.

We can use the summary data displayed in the confusion matrix to compute model accuracy. To determine the accuracy of a model, we sum the values found on the main diagonal and divide the sum by the total number of test set instances. For example, if we apply a model to a test set of 100 instances and the values along the main diagonal of the resultant confusion matrix sum to 70, the test set accuracy of the model is 0.70 or 70%. As model accuracy is often given as an error rate, we

can compute model error rate by subtracting the model accuracy value from 1.0. For our example, the corresponding error rate is 0.30.

If ample test data are not available, we can apply a technique known as cross-validation. With this method, all available data are partitioned into n fixed-size units. n − 1 of the units are used for training, whereas the nth unit is the test set. This process is repeated until each of the fixed-size units has been used as test data. Model test set correctness is computed as the average accuracy realized from the n training/testing trials. Experimental results have shown a value of 10 for n to be maximal in most situations. Several applications of cross-validation to the data can help ensure an equal distribution of classes within the training and test data sets.

Bootstrapping is an alternative to cross-validation. With bootstrapping, we allow the training set selection process to choose the same training instance several times. This happens by placing each training instance back into the data pool after it has been selected for training. It can be shown mathematically that if a data set containing n instances is sampled n times using the bootstrap technique, the training set will contain approximately two-thirds of the n instances. This leaves one-third of the instances for testing.

Two-Class Error Analysis

The three applications listed at the beginning of this section represent two-class problems. For example, a credit card application is either accepted or rejected. We can use a simple two-class confusion matrix to help us analyze each of these applications.

Consider the confusion matrix displayed in the Table below. Cells showing True accept and True reject represent correctly classified test set instances. For the first and second applications presented in the previous section, the cell with False accept denotes accepted applicants that should have been rejected. The cell with False reject designates rejected applicants that should have been accepted. A similar analogy can be made for the third application. Let's use the confusion matrices shown in Table 2.7 to examine the first application in more detail.

Assume the confusion matrices shown in Table 2.7 represent the test set error rates of two supervised learner models built for the credit card application problem. The confusion matrices show that each model displays an error rate of 10%. As the error rates are identical, which model is better? To answer the question, we must compare the average cost of credit card payment default to the average potential loss in profit realized by rejecting individuals who are good approval candidates. Given that credit card purchases are unsecured, the cost of accepting credit card customers likely to default is more of a concern. In this case, we should choose model B because the confusion matrices tell us that this model is less likely to erroneously offer a credit card to an individual likely to default. Does the same reasoning apply for the home mortgage application? How about the application where the question is whether to drill for oil? As you can see, although test set error rate is a useful measure for model evaluation, other factors such as costs incurred for false inclusion as well as losses resulting from false omission must be considered.

A Simple Confusion Matrix

	Computed Accept	Computed Reject
Accept	True accept	False reject
Reject	False accept	True reject

TABLE 2.7 Two Confusion Matrices Each Showing a 10% Error Rate

Model A	Computed Accept	Computed Reject	Model B	Computed Accept	Computed Reject
Accept	600	25	Accept	600	75
Reject	75	300	Reject	25	300

Evaluating Numeric Output

A confusion matrix is of little use for evaluating supervised learner models offering numeric output. In addition, the concept of classification correctness takes on a new meaning with numeric output models because instances cannot be

directly categorized into one of several possible output classes. However, several useful measures of model accuracy have been defined for supervised models having numeric output. The most common numeric accuracy measures are mean absolute error and mean squared error.

The mean absolute error (mae) for a set of test data is computed by finding the average absolute difference between computed and desired outcome values. In a similar manner, the mean squared error is simply the average squared difference between computed and desired outcome. Finally, the root mean squared error (rms) is simply the square root of a mean squared error value. Rms is frequently used as a measure of test set accuracy with feed-forward neural networks. It is obvious that for a best test set accuracy, we wish to obtain the smallest possible value for each measure.

Comparing Models by Measuring Lift

Marketing applications that focus on response rates from mass mailings are less concerned with test set classification error and more interested in building models able to extract bias samples from large populations. Bias samples show higher response rates than the rates seen within the general population. Supervised learner models designed for extracting bias samples from a general population are often evaluated by a measure that comes directly from marketing known as lift. An example illustrates the idea.

Let's consider an expanded version of the credit card promotion database. Suppose the Acme Credit Card Company is about to launch a new promotional offer with next month's credit card statement. The company has determined that for a typical month, approximately 100,000 credit card holders show a zero balance on their credit card. The company has also determined that an average of 1% of all cardholders take advantage of promotional offers included with their card billings. Based on this information, approximately 1000 of the 100,000 zero-balance cardholders are likely to accept the new promotional offer. As zero-balance cardholders do not require a monthly billing statement, the problem is to send a zero-balance billing to exactly those customers who will accept the new promotion.

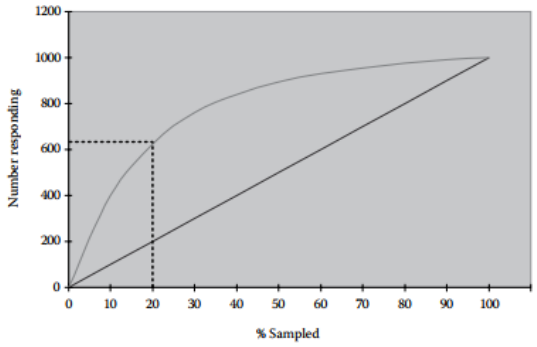
We can employ the concept of lift to help us choose a best solution. Lift measures the change in percent concentration of a desired class, Ci, taken from a biased sample relative to the concentration of C i within the entire population. We can formulate lift using conditional probabilities. Specifically,

$$\text{lift} = \frac{P(C_i|\text{sample})}{P(C_i|\text{population})}$$

where P(Ci|sample) is the portion of instances contained in class Ci relative to the biased sample population and P(Ci|population) is the fraction of class Ci instances relative to the entire population. For our problem, Ci is the class of all zero-balance customers who, given the opportunity, will take advantage of the promotional offer.

The Figure below offers a graphical representation of the credit card promotion problem. The graph is sometimes called a lift chart. The horizontal axis shows the percent of the total population sampled, and the vertical axis represents the number of likely respondents. The graph displays model performance as a function of sample size. The straight line represents the general population. This line tells us that if we randomly select 20% of the population for the mailing, we can expect a response from 200 of the 1000 likely respondents. Likewise, selecting 100% of the population will give us all respondents. The curved line shows the lift achieved by employing models of varying sample sizes. By examining the graph, you can see that an ideal model will show the greatest lift with the smallest sample size. This is represented in Figure 2.4 as the upper-left portion of the graph. Although Figure 2.4 is useful, the confusion matrix also offers us an explanation about how lift can be incorporated to solve problems.

The Table below shows two confusion matrices to help us understand the credit card promotion problem from the perspective of lift. The confusion matrix showing no model tells us that all zero-balance customers are sent a billing statement with the promotional offer. By definition, the lift for this scenario is 1.0 because the sample and the population are identical.



Two Confusion Matrices: No Model and Ideal Model

No Model	Computed Accept	Computed Reject	Ideal Model	Computed Accept	Computed Reject
Accept	1000	0	Accept	1000	0
Reject	99,000	0	Reject	0	99,000

Two Confusion Matrices for Alternative Models with lift Equal 2.25

No Model	Computed Accept	Computed Reject	Ideal Model	Computed Accept	Computed Reject
Accept	540	460	Accept	450	550
Reject	23,460	75,540	Reject	19,550	79,450

The lift for the matrix showing *ideal model* is 100 (100%/1%) because the biased sample contains only positive instances.

Consider the confusion matrices for the two models shown in Table 2.9. The lift for model X is computed as

$$\text{lift(model X)} = \frac{540/24{,}000}{1000/100{,}000}$$

which evaluates to 2.25. The lift for model Y is computed as

$$\text{lift(model Y)} = \frac{450/20{,}000}{1000/100{,}000}$$

which also evaluates to 2.25. As was the case with the previous example, to answer the question about which is a better model, we must have additional information about the relative costs of false-negative and false-positive selections. For our example, model Y is a better choice if the cost savings in mailing fees (4000 fewer mailings) more than offsets the loss in profits incurred from fewer sales (90 fewer sales).

Unsupervised Model Evaluation

Evaluating unsupervised data mining is, in general, a more difficult task than supervised evaluation. This is true because the goals of an unsupervised data mining session are frequently not as clear as the goals for supervised learning. Here we will introduce a general technique that employs supervised learning to evaluate an unsupervised clustering and leave a more detailed discussion of unsupervised evaluation for later chapters.

All unsupervised clustering techniques compute some measure of cluster quality. A common technique is to calculate the summation of squared error differences between the instances of each cluster and their corresponding cluster center. Smaller values for sums of squared error differences indicate clusters of higher quality. A second approach compares the

ratio of within-cluster to between-cluster similarity values. For cluster C, within cluster similarity is measured by computing the similarity of each instance within C to all other instances of C. The similarity scores are averaged to give the within-cluster similarity score for C. Between-cluster similarity for C is determined by computing the similarity of each instance of C to all other instances not in C. These similarity scores are averaged to give the between-cluster similarity score for class C. Larger values for the within-to between cluster ratio indicates that the instances of C form a meaningful cluster. However, for a more meaningful evaluation of unsupervised clustering, it is supervised learning that comes to the rescue. The technique is as follows:

1. Perform an unsupervised clustering. Designate each cluster as a class and assign each cluster an arbitrary name. For example, if the clustering technique outputs three clusters, the clusters could be given the class names C1, C2, and C3.
2. Choose a random sample of instances from each of the classes formed as a result of the instance clustering. Each class should be represented in the random sample in the same ratio as it is represented in the entire data set. The percentage of total instances to sample can vary, but a good initial choice is two-thirds of all instances.
3. Build a supervised learner model using the randomly sampled instances as training data. Employ the remaining instances to test the supervised model for classification correctness.

This evaluation method has at least two advantages. First, the unsupervised clustering can be viewed as a structure revealed by a supervised learner model. For example, the results of a clustering created by an unsupervised algorithm can be seen as a decision tree or a rule-based structure. A second advantage of the supervised evaluation is that test set classification correctness scores can provide additional insight into the quality of the formed clusters.

Finally, a common misconception in the business world is that data mining can be accomplished simply by choosing the right tool, turning it loose on some data, and waiting for answers to problems. This approach is doomed to failure. Machines are still machines. It is the analysis of results provided by the human element that ultimately dictates the success or failure of a data mining project.

CHAPTER SUMMARY

Data mining strategies include classification, estimation, prediction, unsupervised clustering, and market basket analysis. Classification and estimation strategies are similar in that each strategy is employed to build models able to generalize current outcome. However, the output of a classification strategy is categorical, whereas the output of an estimation strategy is numeric. A predictive strategy differs from a classification or estimation strategy in that it is used to design models for predicting future outcome rather than current behavior. Unsupervised clustering strategies are employed to discover hidden concept structures in data as well as to locate atypical data instances. The purpose of market basket analysis is to find interesting relationships among entities such as retail products. Here, discovered relationships can be used to design promotions, arrange shelf or catalog items, or develop cross-marketing strategies.

A data mining technique applies a data mining strategy to a set of data. A data mining technique is defined by an algorithm and a knowledge structure. Common features that distinguish the various techniques are whether learning is supervised or unsupervised and whether their output is categorical or numeric. Familiar supervised data mining methods include decision trees, rule generators, neural networks, and statistical methods. Association rules are a favorite technique for marketing applications. Clustering techniques divide the data set into sub-groups of instances that are similar to each other and dissimilar to instances in other sub-groups. Clustering methods are frequently used to help determine a best set of input attributes for building supervised learner models. Performance evaluation is probably the most critical of all the steps in the data mining process. Supervised model evaluation is often performed using a training/test set scenario. Supervised models with numeric output can be evaluated by computing average absolute, average squared error, or rms error differences between computed and desired outcome. Marketing applications that focus on mass mailings are interested in developing models for increasing response rates to promotions. A marketing application measures the goodness of a model by its ability to lift response rate thresholds to levels well above those achieved by naive (mass) mailing strategies. Unsupervised models support some measure of cluster quality that can be used for evaluative purposes. Supervised learning can also be employed to evaluate the quality of the clusters formed by an unsupervised model.