Chapter 7

Data Mining Fundamentals

In chapter 1 you learned that the two core services provided by SSAS are OLAP and data mining. So far, we've been exploring UDM from the OLAP "dimension" only, completely ignoring its smaller, but no less important, cousin – data mining. Now, it's time to switch gears and give data mining its fair share of attention. In this chapter, we will see how the data mining technology complements OLAP to provide rich data analytics. You will learn:

- The fundamentals of data mining technology.
- The building blocks of SSAS data mining.
- The necessary steps to implement data mining models.
- How to construct data mining models from relational data sources.

We will put into practice what we've learned so far by implementing a data mining model for targeted campaigning. You can find the enhanced SOS OLAP UDM in the Ch07 solution file.

7.1 Understanding Data Mining

Data mining is a newcomer to the business intelligence arena. It is new because only in the past decade the computation power reached the necessary gains to support data warehousing and mining algorithms. Though a young technology, data mining has an exciting future. According to a market forecast study by IDC, data mining is the fastest growing business intelligence segment, surpassing OLAP, relational reporting, or any other business intelligence field. In 2005, the data mining market is expected to grow 32%!

Realizing that data mining is a logical next step for rich data analytics, Microsoft introduced data mining features in SQL Server 2000. Since the initial feature set was rather limited, data mining didn't enjoy broad acceptance. For example, data mining in SQL Server 2000 supported only two algorithms (Microsoft Decision Trees and Microsoft Clustering) and didn't provide adequate model building and tuning tools. In my opinion, all this is going to change with SQL Server 2005. After several years of intensive research and investment, the Microsoft data mining technology comes of age in SQL Server 2005. The data mining engine has been completely re-architected. Five new algorithms have been added to address various data mining scenarios. The data mining tools have undergone a major uplift and now support custom visualization for each algorithm. A great effort has been made to integrate data mining with other Microsoft BI technologies, including UDM, Integration Services and Reporting Services, and to enhance the programming interfaces to build intelligent applications.

7.1.1 What is Data Mining?

Data mining can be described as a business intelligence process that offers three main services -- *data exploration*, *pattern discovery*, and *prediction*. Let's find out more about these services and their practical usage scenarios.

Data mining services

Data mining could help organizations to *explore* large volumes of valuable data and derive knowledge from it. For example, you may have a huge Customer UDM dimension with thousands, if not millions, of members. Data mining can help you segment these customers to find out, for example, what is the typical profile (age, income, occupation, etc.) of customers buying a given product. Many organizations use data mining to find *patterns* in data, e.g. to recommend other items that the customer may be willing to buy together with a given product.

Finally, an organization can leverage data mining to *predict* business metrics based on existing data statistics, e.g. to find out what the next quarter sales could be given the sales statistical data for the past year. As you can imagine, market researchers and data analysts can leverage data mining as a potent and valuable tool to understand a company's business and its customers better.

Dimension	Hierarchy	Operator	Filter Expression
Customer	Customers by Geography	Equal	{ Canada }
<Select dimension>			

Drop Filter Fields Here

	Year				
	2001	2002	2003	2004	Grand Total
Customer	Internet Sales Amount	Internet Sales Amount	Internet Sales Amount	Internet Sales Amount	Internet Sales Amount
Katherine Gonzalez	$2,454.91			$1,566.33	$4,021.23
Paige Reed	$2,469.13		$1,546.93		$4,016.06
Isabella L. Bryant		$2,417.75	$1,594.24		$4,011.99
Xavier Martin	$2,467.94		$1,543.16		$4,011.10
Jordan C. Henderson		$2,431.22	$1,576.73		$4,007.95
Caitlin T. Richardson		$2,438.51		$1,569.30	$4,007.81
Paige Brooks	$2,467.09		$1,540.41		$4,007.50
Alexandra P. Barnes	$2,407.34			$1,598.80	$4,006.14
Madeline H. Parker	$2,446.18		$1,557.64		$4,003.82
Makayla Brooks	$2,397.66			$1,596.75	$3,994.41
Grand Total	$17,110.25	$7,287.48	$9,359.10	$6,331.17	$40,088.01

Figure 7.1 OLAP is a great technology for efficient data aggregation.

Data mining and OLAP

Considering the services that data mining provides, your first impression may be that it is a technology competing with OLAP. True, both technologies seek to provide rich data exploration and reporting. Also, both technologies are used typically in conjunction with data warehousing to process and analyze vast volumes of data. At the same time, however, data mining seeks to provide different services than OLAP and it should be viewed as a complementing, rather than competing technology.

To understand this better, consider the report shown in Figure 7.1. I generated this report from our sample SOS OLAP cube. It shows the top ten Canadian customers that have purchased Adventure Works products for four consecutive years. This report demonstrates one of the main strengths of the OLAP technology which is data *aggregation*. As you know by now, SSAS is designed to aggregate data across dimensions fast.

However, as useful as this report is, it doesn't tell us much about the customers themselves. For example, using just OLAP we have no easy way to find out data patterns, such as customer buying habits, perform basket analysis, or recommend products based on a customer's past purchase history. That's because once the data has been aggregated, hidden data patterns, data

relationships, and data associations are often no longer discernable. Moreover, OLAP doesn't provide prediction capabilities. For example, we can't forecast product sales for the fourth quarter of 2004.

Table 7.1 OLAP is suitable for model-driven analysis, while data mining provides data-driven analysis.

Characteristic	OLAP	Data mining
Pattern discovery	Limited	Core service
Prediction	No	Core service
Object model	Cubes, dimensions, and measure groups	Structrures and mining models
Business metrics	Measures	Dimensions (usually)
Analytics process	On-going, historical analytics	Done typically on as-needed, "ad hoc" basis

The above business requirements can be addressed easily by using data mining, as the examples in this and next chapters will demonstrate. Table 7.1 outlines other major differences between data mining and OLAP. They will be explained throughout the course of this chapter.

7.1.2 Data Mining Tasks and Algorithms

Instead of looking at a crystal ball, data mining practitioners perform their "magic" by leveraging well-known mathematical models that originate from three academic fields: statistics, machine learning, and database theory. Statistical models that are particularly related to data mining are those that are designed to find data correlations, such as Naïve Bayes and Clustering. Other models come from the machine learning (or Artificial Intelligence) research field, such as decision trees and neural networks models. Finally, database algorithms are used to process large data volumes efficiently.

Data mining can be applied to a number of different tasks. The most popular data mining tasks are association, classification, segmentation (clustering), regression, and forecasting. An essential coverage of these tasks is provided in the OLE DB for Data Mining Specification (OLE DB/DM) and the excellent webcasts from the SSAS data mining team (see Resources). Discussing them in this chapter will be redundant.

Note The OLE DB for Data Mining specification was created in 2000 by Microsoft with the assistance of other data mining partners to define an industry standard for creating and modifying data mining models, train these models, and then predict against them. I highly recommend that you read the specification (see Resources section). It is a great resource for understanding not only the Microsoft data mining implementation details, but also the data mining technology and algorithms in general. Currently, data mining is part of the XMLA specification.

The mining tasks are realized by well-known mathematical algorithms. SSAS 2005 implements seven data mining algorithms. Choosing an algorithm to perform a given task can be challenging. Table 7.2 should help you choose the right algorithm for the task at hand.

Table 7.2 Data mining tasks and algorithms to implement them.

Task	Decision Trees	Clustering	Association	Naïve Bayes	Sequence Clustering	Neural Network	Time Series
Classification	✔	✓		✓	✓	✓	
Segmentation		✔			✔	✓	
Association	✓		✓				
Regression	✓	✓			✓		
Forecasting							✓

Note that in some cases a given task can be performed by several algorithms. The ones that are most suitable are highlighted in Table 7.2. For example, the classification task can be realized by the Decision Trees, Clustering, Naïve Bayes, Sequence Clustering, and Neural Network algorithms, but the Decision Trees algorithm should be your best choice.

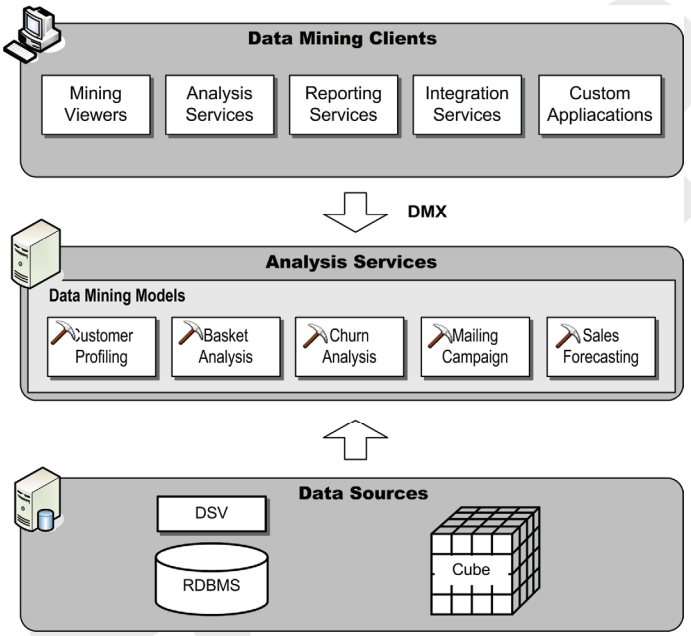

Figure 7.2 Data mining models can source data from relational databases or UDM cubes and can be integrated with different types of clients.

It is also important to note that the SSAS data mining architecture is extensible. Custom data mining algorithms can be plugged in (see Resources section for more information) and custom mining viewers can be implemented to replace the ones provided by Microsoft.

7.1.3 Data Mining Architectural View

Now, let's see how SSAS data mining works and how it fits in the Microsoft Business Intelligence Platform. Figure 7.2 shows a high level architectural view of SSAS data mining. The SSAS data mining architecture consists of data mining models (processed and executed on the server),

data sources that feed the models, and mining clients that retrieve and analyze the predicted results.

Data mining models

As shown in Figure 7.2, data mining technology in SSAS is exposed to clients as one or more *data mining models* hosted in an Analysis Services database. I will explain in more detail what a data mining model is in section 7.2.1. For now, know that a data mining model is just a UDM object that takes some input data and outputs the predicted results using the selected mathematical algorithm. A data mining model can be designed and managed using BI Studio or programmatic interfaces, such as Analysis Management Objects (AMO).

Data sources

Data mining models can be fed with data from two types of data sources.

- *Relational data sources* – A data mining model can source its data from any data source that has an OLE DB driver, including SQL Server, Oracle, Teradata, etc. If data is available as text files, Integration Services can be used to extract, transform, and load the source data into a relational data source or directly in the model. Similar to SSAS cubes, a data mining model doesn't access the data source directly. Instead, it uses a data source view to isolate itself from the relational data source schema and optionally enhance it.
- *UDM* – A data mining model can be built on top of an SSAS 2005 cube. In this case, the cube and the data mining models must be hosted in the same database. A DSV is not needed when a data mining model draws data from a cube.

In this chapter, I will show you how to build a data mining model from a relational data sources. The next chapter demonstrates how you can use UDM as a data source of mining models.

Data mining clients

In the simplest scenario, the end user could use Microsoft-provided or third-party data mining viewers to browse the data mining models inside the BI Studio IDE. At the other end of the spectrum, a custom application front end could be implemented to query the mining models and display the predicted results.

Data mining is well integrated with the other products of the Microsoft Business Intelligence Platform. For example, the results of the data mining model can be used to create a dimension to enrich an SSAS cube. Integration Services include tasks specifically tailored for data mining. For example, in the next chapter, we will implement an SSIS package that uses the Data Mining Query Task to classify customers on the fly. Finally, Reporting Services can be used to deliver the data mining results to the end users in the form of a standard or ad-hoc report. An SSRS integration example is demonstrated in chapter 18.

7.1.4 Data Mining Extensions (DMX)

Clients can create and query data mining models and obtain predictions by sending DMX (**D**ata **M**ining E**X**tensions) statements. To facilitate a wide spread adoption of data mining and minimize the learning curve, the SSAS team was set to provide a query language that is familiar to database developers and easy to use. Since almost everyone knows SQL, the SSAS team decided to adopt it as a foundation for mining queries and extend with data mining-specific features (documented by the OLE DB for Data Mining Specification). Similar to SQL, DMX

provides both Data Definition Language (DDL) and query constructs. For example, here is what the basic form of a DMX SELECT query looks like:

```
SELECT <expression list> FROM <mining model>
[NATURAL] PREDICTION JOIN
<source data> AS <alias> ON <column mappings>
```

Let's briefly explain the DMX-specific constructs in the SELECT statement.

Expression list

As an SQL SELECT statement specifies which columns from a relational table will be returned, the expression list in a DMX SELECT statement enumerates the predictable columns from the model that will be retrieved. For example, if I have a cluster mining model for customer profiling, the expression list can bring some customer-related columns, such as Age, Occupation, Education, etc.

In addition, DMX supports a set of functions (about thirty) that can be used in prediction queries. For example, you can use the *PredictCaseLikeliHood* function to determine the possibility for a customer to belong to a cluster. Again, similar to SQL joins, the PREDICTION JOIN...ON clause links the mining model with the source data. If the names of the mining model match the source data columns NATURAL PREDICATION JOIN can be used and the ON clause can be omitted. In this case, the columns relationships are automatically inferred through a naming convention.

Source Data

The source data specifies the input dataset that will be predicted against the mining model and it could be:

- *Database OPENQUERY or OPENROWSET query* – Use a database query to feed the mining model with data fro a relational table.
- *Singleton query* – In this case, the data values are embedded in the SELECT statement.
- *Another DMX query* – DMX queries can be nested.
- *Rowset parameter* – A client can pass an application rowset, such as an ADO.NET dataset or a data reader, as an input to the mining model.

Examples of the first two options are provided in this chapter. Chapter 17 includes more examples that demonstrate how custom applications can create and query data mining models.

7.2 Data Mining Objects

Now, let's see how the data mining technology is realized in SSAS. UDM defines two main data mining objects -- data mining *models* and data mining *structures*. Let's explain structures and models in more details. We will start with the data mining model since it is the cornerstone of the SSAS data mining technology.

7.2.1 Data Mining Models

The OLE DB for Data Mining Specification describes a data mining model as a virtual object that is similar to a relational database table. A mining model defines how data should be analyzed

and predicted. For example, suppose that you need to implement a data mining model to promote the latest bicycle product as part of a mailing campaign effort. To identify which customers are most likely to respond to the campaign, you may design your model as the one shown in Figure 7.3.

Usage: *Key*	Usage: *Ignore*	Usage: *Input*	Usage: *Input*	Usage: *Input*	Usage: *Input*	Usage: *Predict*
Customer ID	**Name**	**Age**	**Gender**	**Education**	**Commute Distance**	**Bike Buyer**
1	John	25	M	Graduate Degree	1-5 Miles	Y
2	Alice	30	F	Partial College	5-10 Miles	N
3	Bob	35	M	High School	1-5 Miles	Y

Figure 7.3 A data mining model consists of set of columns with different usage types.

Model definition

Just like a relational table, the mining model definition consists of columns. Each column can have different usage inside the model. If a column identifies a row in the model (also called a *case*), the column usage type needs to be set to *Key*. For example, in our mailing campaign model, the role of the key is fulfilled by the Customer ID column because it identifies uniquely a row in the model. A column with a usage type of *Input* is used as an input parameter to the data mining algorithm. In our case, all demographics-related columns could be used as input columns. For example, we may wish to find the correlation between the Commute Distance column and the probability that a customer will purchase a bicycle. To do so, you set the Usage type of the column to Input.

A data mining model can have one or more columns with Predict or PreductOnly Usage types. A column with a *Predict* Usage type is used both for input and predictions. In comparison, as its name suggests, a *PredictOnly* column is used only for predictions. In the mailing campaign scenario, we need to predict what category of customers is likely to become bike buyers based on past purchases. Therefore, we need to set the Bike Buyer Usage type to *Predict*. Finally, we may decide to ignore a given column. For example, the Name column is not useful to find patterns. That's why we set its Usage to *Ignore*.

Does the column usage type bring any recollection? Indeed, we can loosely relate a data model to a UDM dimension. Just like a UDM dimension, a data model consists of columns (attributes) which can have different usage types within the containing model. I will be quick to point out that one noticeable difference between a UDM dimension and a data mining model is that a mining model doesn't store the input dataset. Instead, the input dataset is used only to train the model. The server stores only the results of the prediction process in the form of rules or patterns.

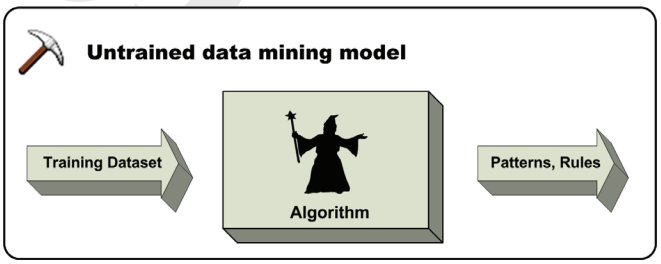

Figure 7.4 A data mining model must be trained before it can be used for predictions.

Training the model

Before a mining model can be used, it must be *trained* by loading the model with data. SSAS trains a model by executing the mathematical algorithm associated with the model to derive useful patterns or rules from input data (Figure 7.4). You can train a model by either processing it (e.g. using BI Studio or an SSIS package), or by submitting a DMX INSERT INTO statement. The latter option is typically used when creating the model programmatically, e.g. by using ADOMD.NET. For example, you can use the following statements to create and train a hypothetical CustomerProfilingMC mining model that uses the Microsoft Clustering algorithm:

```
CREATE MINING MODEL [CustomerProfilingMC]
(
    Customer LONG KEY,
    Age LONG CONTINUOUS,
    [Yearly Income] DOUBLE CONTINUOUS,
    [Occupation] TEXT DISCRETE
)
Using Microsoft_Clusterting

INSERT INTO  CustomerProfiling (Customer, Age, [Yearly Income], Occupation)
OPENQUERY(MyLinkedServer, 'SELECT CustomerID, Age, Income, Occupation FROM Customers')
```

In this case, an OPENQUERY clause is used to load the model with data from a relational table Customers. Note that we cannot query the table by using a SELECT statement because the model resides in an SSAS database (thus the OPENQUERY statement). The time needed for training a data mining model depends on the amount of input data and the complexity of the algorithm. The end result of training a data model is patterns or rules that are saved on the server.

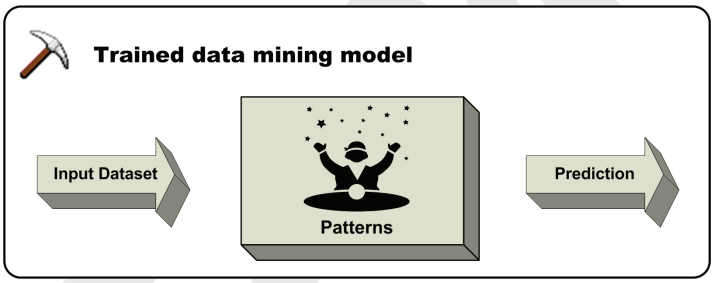

Figure 7.5 A trained mining model takes an input dataset, applies the learned patterns, and returns predications.

Performing predictions

Finding patterns in historical data is just half of the work in a typical data mining project. The next logical step is to query the mining model and obtain predictions against *new* data. For example, after training a customer profiling mining model with historical data, the model could identify (predict) a particular class of customers that are likely to purchase a product. Given the predicted results, we may need to know how likely it is that a new customer will become a potential buyer. To understand this better, it may be helpful to visualize a data mining model as a black box, which takes the data to be predicted as an input, applies the learned patterns, and outputs the predicted results (Figure 7.5).

Here, the term *prediction* is used rather loosely to describe the result of the mathematical calculation that the mining algorithm performs against the new dataset. The result may not even have a time dimension. For example, the output of the data mining model could be classifying a

customer to a given cluster if the data mining task is customer profiling. Considering the CustomerProfilingMC model example, this is how an intelligent application can send a DMX SELECT query to find out how likely it is that a given customer with the supplied demographics criteria may belong to a given cluster.

```
SELECT ClusterProbability('Cluster 8')
FROM [CustomerProfilingMC]
NATURAL PREDICTION JOIN
(SELECT 35 AS [Age],
  'Professional' AS [Occupation],
  80000 AS [Yearly Income]
) AS t
```

In this case, the application uses a *singleton* query to pass the data values directly to the model. Alternatively, the second SELECT statement can retrieve the data to be predicted from a data source.

Data mining tables

When a model is trained, input data must be passed to the model as a single table which is called a *case table*. In the simplest case, each row column in this table will have only one value, as with the mailing campaign scenario shown in Figure 7.3. Sometimes, you may need to find correlations between related datasets. For example, suppose that you need to perform customer basket analysis to recommend related products to customers (customers who bought A product also bought B product). To implement the basket analysis mining model, you will need two tables, as shown in Figure 7.6.

Customer ID	Name	Age	Gender	Education	Products Bought
1	Bob	35	M	College	

Product ID	Name	Price
1	Beer	$12
2	Chips	$3
3	Milk	$2.50

Figure 7.6 A correlated dataset can be exposed as a nested table.

This model has two tables, Customer and Product, linked with a one-to-many relationship. Since a mining model can have only one input table, this schema presents an issue because we need to "flatten" the dataset to a single table. SSAS solves this dilemma by allowing us to "embed" the child table (the one on the many side of the relationship) into a column of the parent table in the form of a *nested table*. For example, in our case, the child table (Product) could be defined as a nested table embedded inside the Products Bought column of the Customer parent table. SSAS supports a single level of nesting only. In other words, a child table cannot have another nested table. The OLE DB for Data Mining Specification also defines the term *case* and *case set*.

Definition A case is a collection of data associated with a single row in the input table. A case set is the collection of all cases (all records of the parent table plus all records of the nested tables).

For example, in our fictitious basket analysis model, a case represents a single customer and its associated products. In the mailing campaign example, a case simply corresponds to an individual customer because the Customer table doesn't have nested tables.

7.2.2 Data Mining Structures

As you've seen, a data mining model could simply be described as a collection of table columns and a mining algorithm that acts upon the data contained in these columns to analyze it and perform predictions. At the same time, we've learned that a particular data mining task may be performed by using different algorithms. Naturally, you may be willing to try a few algorithms with a given mining task to select the most accurate algorithm. Since, in this case, all task algorithms will use the same data schema, it may make sense to re-factor the database schema definition in its own object, just like a data source view may be shared by several cubes. In SSAS, a data mining structure gives you this level of abstraction.

What is a data mining structure?

You can think of a data mining structure as a blueprint of the database schema which is shared by all mining models inside the structure. In this respect, developers familiar with object-oriented programming may relate a data mining structure to a class, while the actual data mining models that share the same structure can be described as concrete instances (objects) of the class. The relationship between mining structures and models is shown in Figure 7.7.

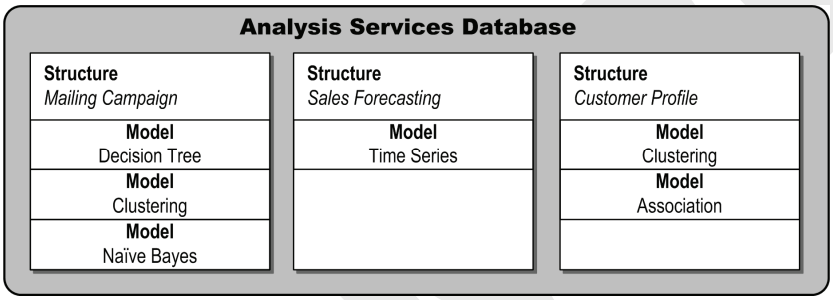

Figure 7.7 Data mining structures are containers of data mining models.

An Analysis Services Database may have many data mining structures. Each structure may contain one or more data mining models.

Why have data mining structures?

Why do we need separate data mining structures instead of uniting them into a coarser model, e.g. a cube? As I've mentioned, the main reason for this is to reduce complexity. Data mining models may be rather involved and may require experimenting with several algorithms. A structure encapsulates the logical mining domain at hand and the associated prediction algorithms. For example, a mailing campaign structure may have three data mining models associated with it that use three different algorithms, e.g. a Decision Tree, Clustering, and Naïve Bayes algorithms. Second, once the structure is processed, it is loaded with data and this data can be shared by all models within this structure. In other words, all mining algorithms contained in a structure can be trained with the same dataset.

Note With the introduction of structures in SSAS 2005, one potential source of confusion could be the naming convention that Microsoft has chosen for data mining objects. A mining model definition (a file with extension *.dmm) in fact represents a mining structure, while the mining models inside the structure actually represent one or more mathematical algorithms which perform the task. To make things even more confusing, the term *data mining model* is used to describe both a mining domain (e.g. sales forecasting), and a specific algorithm (model) inside the structure.

Finally, there are certain performance advantages of using structures as a method of data model isolation. For example, you can make changes to a data model inside a structure and even add new models, without having to re-process the structure.

The "structure" of a data mining structure

Just as a dimension definition in UDM describes the dimension attributes and their bindings, a data mining structure definition contains columns and column bindings. In this respect, a column in a mining structure can be related to a dimension attribute. As such, a structure column has a Type which denotes the column data type of the underlying table. The column type could be one of the five data types supported by the OLE DB/DM specification (Text, Long, Boolean, Double, and Date).

Column bindings

Again, similar to a dimension attribute, a structure column has KeyColumns and NameColumn properties that specify how the structure column is bound to the underlying data source column. Similar to a dimension key attribute, the KeyColumns property is used to identify the mining cases.

Warning Usually, the structure key coincides with the primary key of the relational case table (relational data source) or the dimension key of the case dimension (OLAP source). That's because all other columns (attributes) are related to the table (dimension) key. Therefore, you can choose any column as an input column. However, there is nothing stopping you from choosing another column as a structure key. For example, if you source data from the Customer dimension (SOS OLAP cube), you may choose to set the structure key to the Country attribute. In this case you will have as many input cases as the number of unique country members. When you do so, however, you may automatically disqualify other attributes, e.g. customer's age, gender, because they are not related to the key.

The NameColumn property could be used to supply alternative names of the attribute members if they need to be different than the key column values. Finally, when a SSAS 2005 cube is used as a data source, the Source property specifies which dimension attribute the structure column is bound to.

Column content

To predict data properly, a data mining algorithm needs to know in advance the content type of the column. The model designer uses the Content column property to specify the column content type. SSAS data mining supports several column content types which are explained thoroughly in the OLE DB for Data Mining specification. The most common ones are *Discrete* and *Continuous*. An example of a discrete content type is a customer gender column because it would contain only a few distinct values, e.g. *male* and *female*. An example of a continuous column is a customer income column because it may contain arbitrary values.

Some algorithms may not support all content types. For example, the Naïve Bayes algorithm supports only discrete columns as an input. If you need to use a continuous column, you can use the Data Mining Designer to discretize it into buckets by setting its *Content* property to *Discretized* and specifying the discretization bucket count and method. This is very similar to the process of discretizing attribute members of a UDM dimension, as we discussed back in chapter 4.

Processing mining structures and models

Just as a cube needs to be processed before it is first used, or when the cube structure changes, mining objects (structures and models) need to be processed occasionally.

Processing structures

Some common reasons for processing a mining structure include changes to the structure definition (e.g. a column is added or deleted), re-processing the source OLAP dimension is re-processed (OLAP data mining only), and loading the structure with new data. During the structure processing, the server retrieves the distinct values of each structure column (similar to dimension processing). To optimize the performance of the mining models contained in the structure, the structure data is compressed and stored on the server using the same storage primitives as UDM dimensions and measures.

Tip You don't need to process a structure if you've only made changes to its mining model(s). For example, the structure need not be processed if you add a new data model.

An interesting processing detail is that, behind the scenes, the server creates private cubes and dimensions to store the structure data. Specifically, the server generates a private dimension for each discrete structure column and a private measure for each continuous structure column. Each nested structure table is mapped to a measure group.

Processing mining models

As the data evolves (e.g. records are added, changed, or deleted), the mining model needs to be re-trained to keep its prediction capability on a par with the changes in the historical data. The terms *processing* and *training* are used interchangeably to refer to the process of loading data into a mining model to train the model. A model can be processed as part of processing the containing structure. For example, when you press the *Process the mining structure and its related models* toolbar button of the Data Mining Designer, the server initiates the processing task in two passes. During the first pass, the server processes the structure to load it with data. Next, the server processes the model(s) contained within the structure.

A model can also be processed (trained) independently of its containing structure (recall that a model doesn't store the trained dataset but just the predicted patterns). For example, after processing a structure, you may decide to train only one of the structure models with the new data while the rest should stay unaffected. To do so, you can use an SSIS package to process mining models selectively or you can train the model programmatically. Refer to the product documentation of the DMX INSERT INTO statement for more information about how structure and model processing affect each other.

Processing options

SSAS provides a number of processing options to process a structure and its mining models, including ProcessFull, ProcessStructure, and ProcessDefault. ProcessFull processes both the structure and its models in one pass. ProcessStructure processes only the structure. Specifically, ProcessStructure generates the structure metadata, loads the structure with data, and caches it on the server. Once the structure data is cached, you can process the containing mining models by choosing the ProcessDefault option.

7.2.3 The Data Mining Design Process

As with other IT projects, you may benefit from a guided methodology when implementing data mining models. One such methodology may encompass the steps shown in Figure 7.8. It is loosely based on the CRoss Industry Standard Process for Data Mining (CRISP-DM) process

model developed in Germany (see Resources). The tools you can use to perform the step are shown below the task name.

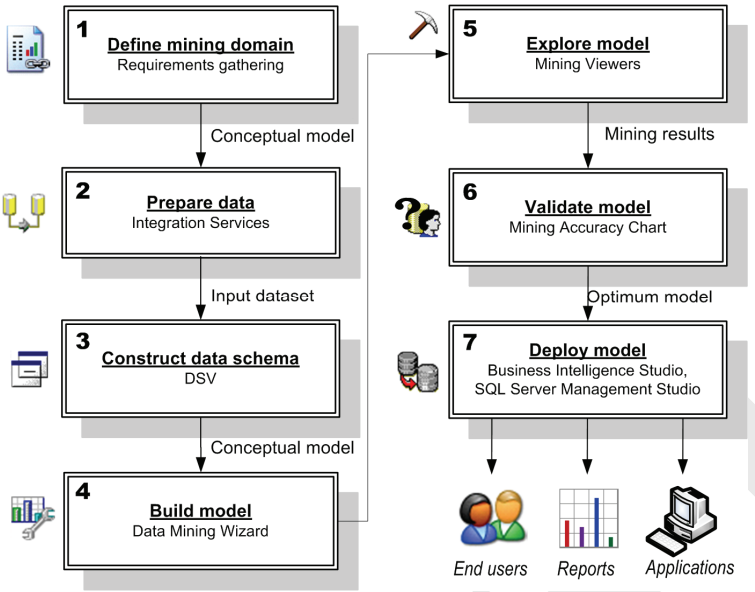

Figure 7.8 A typical data mining process consists of seven steps.

Although the flow depicted in Figure 7.8 is linear, in real life, you may find that you need several iterations to finalize the data model. Data mining models can be complex and there are really no magic rules to create an optimum mining model. For example, after validating the model, you may be surprised to find out that certain input columns have a stronger impact on the predicted results than the obvious candidates, e.g. the customer income is a stronger factor in buying a bicycle than the commute distance.

To refine the model, you may need to go back to step 2, e.g. to add more demographics-related columns, e.g. a column that indicates if the customer is a homeowner. This shouldn't discourage you (unless you need to collect a fine when going back, perhaps). Remember, data mining is more of an art than a science. Let's explain briefly the purpose of each step in the design process.

Step 1: Define mining domain

Start your data mining project from the drawing board by gathering the business requirements. What business problem are you trying to solve? Sales forecasting? Customer profiling? Basket analysis? End of the world, as we know it? As with every project, it is essential to start with a clear objective. This objective may be formulated as "Perform a data mining analysis to profile customers and identify those who are most likely to respond to a product X mailing campaign based on customer demographics". Once you've defined the mining domain, you need to conduct data availability study to identify the dataset that will support the model. This may translate into questions, such as:

- *What input columns do I need?* – For the mailing campaign scenario, these could be customer gender, income, age, etc?

- *Where is the data that will represent the input dataset stored?* – You need to identify which data source captures the data you need.
- *How can data be extracted?* – As noted, SSAS data mining can source data from a relational data source or a UDM cube. To prepare the input dataset, you may need to extract data from its original data source, which may require an ETL project of its own (now you know where data *mining* derives its name from).

Step 2: Prepare data

As noted, the source data may not be directly available and may require ETL processes to load the data into the desired input format. For example, if the data is kept in a mainframe database, data may need to be extracted to flat files, transformed, and loaded into a relational database, as we've demonstrated in the preceding chapter. Consider SQL Server Integration Services for the data preparation step.

The result of this project step is a relational data schema (in the form of a table or a view) or a UDM cube that will feed all input columns and nested tables of the data mining structure. You should have a clear understanding about the semantics of each input column and its business purpose. People new to data mining may have unreasonable (clairvoyant) expectations by believing that data mining can do everything by just running the Mining Wizard against a database. In reality, however, you will find that the more you know your data, the more effective your prediction results will be.

For example, instead of seeking correlations between all columns of the customer table and the probability that a customer will purchase a given product, it may make sense to identify a subset of the most likely candidates that may impact the customer decision, e.g. commute distance, home owner, education, etc. A less focused approach may require more time to process the data mining models and interpret the results.

Step 3: Construct data schema

As you know by now, a data source view isolates UDM from the underlying relational data schema. If you source your input dataset from a relational database, you need to construct a data source view. This means that you can take the relational schema as it is and adjust it to meet your structure requirements. For example, if security policies rule out direct changes to the data source schema, you can define named queries and calculations at a DSV level. Or, you can create logical relationships among tables in the absence of referential joins.

The end result of this step is the definition of the data mining structure that will serve as foundation of the data mining model(s).

Step 4: Build model

Once the structure is in place, you proceed to create the actual data mining models. Start by consulting with Table 7.2 to identify one or more algorithms that can be used to perform the data mining task at hand. Next, use the BI Studio Data Mining Designer to implement the model(s). For example, the data mining task that you will be performing in the mailing campaign scenario shortly is classification. This task can be performed by the Decision Trees, Clustering, Sequence Clustering, Naïve Bayes, and Neural Network algorithms.

If multiple algorithms can be used to implement the data mining task, a good approach is to start with a quick and simple algorithm, e.g. Naïve Bayes. However, as a best practice, I recommend you build additional mining models that use different algorithms to select the optimum

algorithm for the task at hand. The Data Mining Designer makes it very easy to create corresponding data mining models since they share the same structure.

Step 5: Explore model

Once the model is built and trained, you are ready to explore it and analyze its predictive results. The Microsoft-provided algorithms come with excellent graphical viewers. All viewers have multiple tabs to see data from different angles.

Step 6: Validate model

In addition, you need to validate the accuracy of the data mining model(s) you've built. This is especially important if you have a structure with several models that can perform the same data mining task. One practical approach to validate a data mining model is to prepare a smaller input dataset that you can easily train and evaluate. Once the test dataset is ready, you can use the Mining Accuracy Chart tab of the Data Mining Designer to compare the model accuracy. You can create a *lift chart* or *a classification matrix* to do so.

Step 7: Deploy model

Your data mining model is ready. As a final step, you need to deploy and configure the model to the production server. As part of the configuration process, don't forget to secure the model by specifying which users and Windows groups will have access to it. You may need to process the model occasionally if the data changes. Finally, once the model is deployed, you can use a variety of technologies to query the model and deliver the prediction results to the end users, including standard reports, or custom applications. To learn more about the data mining process, read the excellent book *Preparing and Mining Data with Microsoft SQL Server 2000 and Analysis Services* (see Resources section).

Now that we've covered the essentials, let's put data mining in action. We will demonstrate the seven-point framework we've just introduced by building a targeted campaigning mining model to identify the most likely buyers of Adventure Works bicycle products.

7.3 Targeted Campaigning Example

Suppose that Adventure Works has introduced a new bicycle product. The Adventure Works marketing department is planning a campaign to promote the new bicycle. As you can imagine, promotion campaigns are not cheap. In the case of a mailing campaign, promotion materials need to be designed, printed, and mailed to customers. Radio or TV campaigns are even more expensive. It certainly makes sense to focus the marketing effort on those groups of customers who are most likely to purchase the product. In addition, a mining model could help us identify the promotion channel where these customers are likely to be found. For example, if the data mining results indicate that predominantly teenagers purchase a given product, the management could decide to run an advertisement on MTV.

7.3.1 Defining the Mining Domain

Instead of campaigning to its entire customer base (more that 14,000 customers), the Adventure Works management has asked you to identify a subset of customers who are most likely to purchase the new bicycle. The data mining domain you need to solve is a classic example of the

classification mining task. Consulting with Table 7.2, you realize that there are several algorithms that you can use to perform it. As part of the process of building the data mining model, we will try a couple of them to identify the most accurate algorithm.

7.3.2 Preparing the Data

Unlike dimensional modeling, the business metrics that need to be passed as in input to a mining model typically originate from dimension-type tables (not fact tables). Since the focus of our campaigning task is customer-oriented, we need to load the data mining model with data from a customer profile table. The DimCustomer dimension table is the natural choice since it captures the profile details of the customers who purchase Adventure Products online.

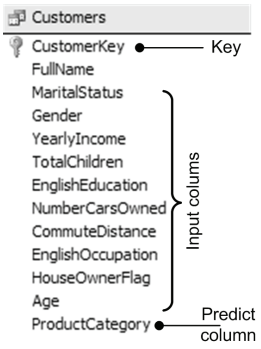

Figure 7.9 When determining the schema of the input dataset, identify which attributes could be used as input columns to influence the data mining prediction.

How do we determine the input dataset schema? Unfortunately, there are no fixed rules and you may find that knowing your data beforehand could be helpful. For example, we know that the Adventure Works OLTP sales application captures the customer demographics data. Therefore, it could certainly be useful to seek a correlation between these attributes and the customer decision to purchase a new bike. One practical approach to examine data and evaluate the usefulness of a given column is to use the data exploration feature of the DSV Designer. You can use the Pivot Table, Chart, and Pivot Char tabs to find the distribution of the column values, the content type (discrete or continuous), and possible correlation link among columns. After a few iterations, our input dataset schema may look like the one shown in Figure 7.9.

For the targeted campaign scenario, all input columns are derived from the DimCustomer table (no nested tables are needed). The most important column is the ProductCategory column which will be used both as an input and predict column. That's because we want to use the historical sales statistics to predict future behavior (the likelihood of a group of customers to purchase products from a given category). Breaking down the data mining results per product category makes our model more universal. Our data mining model will find data correlation patterns in the customer profile data based on the product category selected. When Adventure Works introduces a new bicycle, the marketing department can use the same model to find potential customers for this product by simply filtering the results.

7.3.3 Constructing the Data Schema

The only trick when constructing the input dataset is querying the customer order history to get the product category of the purchased bicycle. To accomplish this, we need to look up data in

table FactInternetSales in an attempt to find an order for a bicycle product (*ProductSubcategoryKey = 1*). However, a customer may have purchased more than one bicycle. Obviously, in this case, we cannot assign a single product category to the customer since there isn't a one-to-one relationship between the customer and the product category. For the sake of simplicity, we will ignore the customers who have purchased more than one bicycle.

Since we will be using a relational data source, we need to build a data source view (Bike Buyers.dsv) on which the mining structure will be built. If you can get the database administrator to grant you permissions to create new objects, by all means, consider using a SQL view for the query statement needed to join the required tables. As noted in chapter 2, SQL views have certain performance and functional advantages over data source views. Chances are, though, that the larger the enterprise and the more people involved, the less likely it is to finish your project on time. Let's take the data source schema as it is and build a DSV named query called Customers on top of it, which is based on the following SQL statement (abbreviated):

```
SELECT  C.CustomerKey, C.MaritalStatus, … ,
     DATEDIFF(yy, C.BirthDate, GETDATE()) AS Age,
     CustomerFilter.Subcategory AS ProductCategory
FROM DimCustomer AS C INNER JOIN
        (SELECT C.CustomerKey, PS.EnglishProductSubcategoryName AS Subcategory
        FROM DimCustomer AS C
     INNER JOIN FactInternetSales AS S …
     INNER JOIN DimProduct AS P ON S.ProductKey = P.ProductKey
     INNER JOIN DimProductSubcategory AS PS
     WHERE (PS.ProductCategoryKey = 1)
GROUP BY C.CustomerKey, PS.EnglishProductSubcategoryName
HAVING(COUNT(PS.ProductSubcategoryKey)=1)) AS CustomerFilter
ON C.CustomerKey = CustomerFilter.CustomerKey
```

The subquery statement is used to filter out the customers that have purchased multiple bicycles. The first dozen rows of the dataset derived from executing this query are shown in Figure 7.10. The ProductCategory column contains the product category of the item bought by the customer. In addition, I set the CustomerKey column as a logical primary key because it uniquely identifies each customer (a mining case represents a customer).

CustomerKey	FullName	MaritalStatus	Gender	YearlyIncome	TotalChildren	EnglishEducat	NumberCarsO	CommuteDist	EnglishOccup	HouseOwne	Age	ProductCategory
11000	Jon V. Yang	M	M	90000	2	Bachelors	0	1-2 Miles	Professional	1	39	Touring Bikes
11001	Eugene L. Hu	S	M	60000	3	Bachelors	1	0-1 Miles	Professional	0	40	Road Bikes
11002	Ruben Torres	M	M	60000	3	Bachelors	1	2-5 Miles	Professional	1	40	Touring Bikes
11003	Christy Zhu	S	F	70000	0	Bachelors	1	5-10 Miles	Professional	0	37	Touring Bikes
11004	Elizabeth Joh	S	F	80000	5	Bachelors	4	1-2 Miles	Professional	1	37	Touring Bikes
11005	Julio Ruiz	S	M	70000	0	Bachelors	1	5-10 Miles	Professional	1	40	Touring Bikes
11006	Janet G. Alvar	S	F	70000	0	Bachelors	1	5-10 Miles	Professional	1	40	Touring Bikes
11007	Marco Mehta	M	M	60000	3	Bachelors	2	0-1 Miles	Professional	1	41	Touring Bikes
11008	Rob Verhoff	S	F	60000	4	Bachelors	3	10+ Miles	Professional	1	41	Touring Bikes
11009	Shannon C. C	S	M	70000	0	Bachelors	1	5-10 Miles	Professional	0	41	Touring Bikes
11010	Jacquelyn C.	S	F	70000	0	Bachelors	1	5-10 Miles	Professional	0	41	Touring Bikes
11011	Curtis Lu	M	M	60000	4	Bachelors	4	10+ Miles	Professional	1	42	Touring Bikes
11015	Chloe Young	S	F	30000	0	Partial Colleg	1	5-10 Miles	Skilled Manua	0	26	Mountain Bikes

Figure 7.10 Consider using a named query to construct the input dataset

7.3.4 Building the Model

The next step involves constructing the data mining structure and a data mining model(s). The easiest way to perform this task is to use the handy Data Mining Wizard by following these steps.

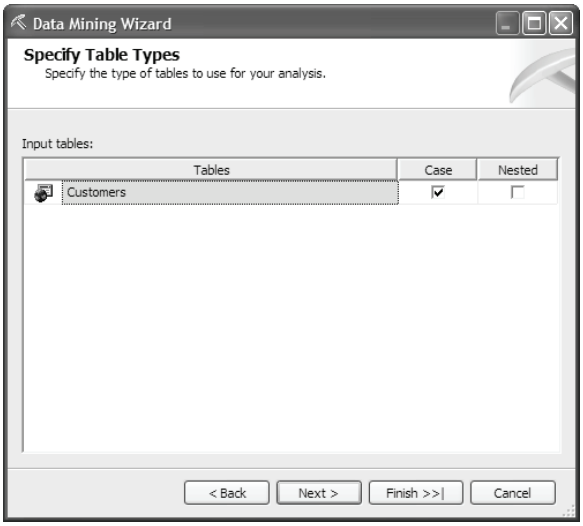

Figure 7.11 A data mining model can have only one case table.

Constructing the mining structure

1. Open the Ch07 solution file. Right-click on the Mining Structures folder and choose *New Mining Structure*. BI Studio launches the Data Mining Wizard. Click Next.
2. In the Select the Definition Method step, select the *From existing relational database or data warehouse* option since our structure will be loaded from the AdventureWorksDW relational database. Click Next.
3. In the *Select the Data Mining Technique* step, you need to pick an algorithm to perform the data mining task at hand. Glancing at Table 7.2, we determine that the Microsoft Decision Tree algorithm is especially suited for classification tasks, such as targeted campaigns. Accept the default *Microsoft Decision Tree* algorithm and click Next.

   Note By default, the *Algorithm name* dropdown lists only the algorithms provided by Microsoft. If you have developed and configured a custom algorithm, it will appear in the dropdown as well.

4. In the Select the Data Source View step, select the Bike Buyers data source view we discussed in the *Construct data schema* step. Click Next to advance to the Specify Table Type step (Figure 7.11).
5. In our scenario, the entire input dataset is contained in a single DSV table. The Data Mining Wizard has pre-selected the *Customer* table as a case table. Click Next.
6. In the *Specify the Training Data* step (Figure 7.12), you need to identify the input and predict columns. In our case, we need to predict the classes of customers that will buy a bike. Therefore, we only need one predict column. Select the *ProductCategory* column as both an Input and Predict column (*Predict* Usage type).
7. You can check the input columns manually (hold Shift for continuous selection), or you can ask the wizard to suggest indicative input columns by sampling the source data. Try the latter technique by clicking on the Suggest button. The Suggest Related Columns dialog appears and

starts sampling the case table in an attempt to find useful correlations between the predict column and the rest of the columns.

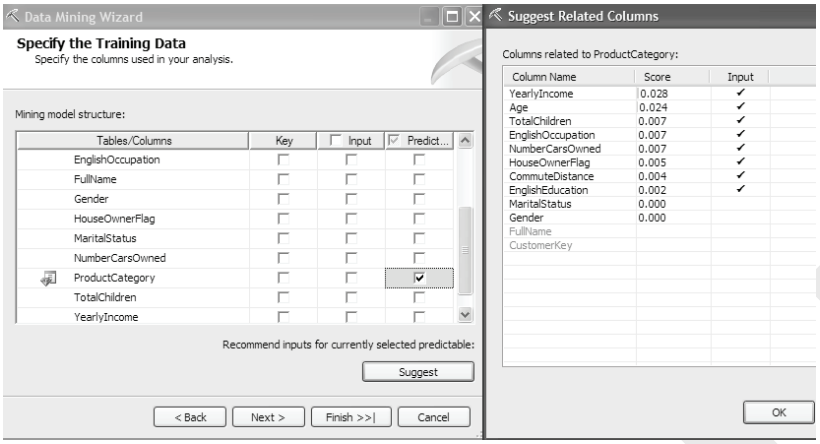

Figure 7.12 The Suggest Related Columns dialog samples case data and suggests input columns.

8. In the Suggest Related Columns dialog, select all columns with a score greater than zero and click OK to return to the Specify the Training Data step. Click Next to advance to the Specify Columns' Content and Data Type step (Figure 7.13).

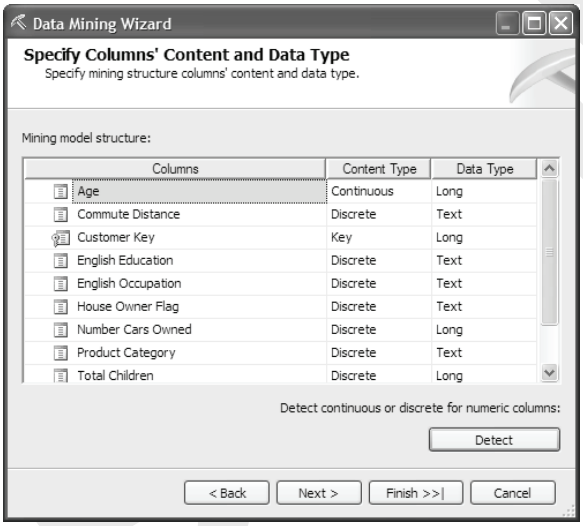

Figure 7.13 Click the Detect button to let the Data Mining Wizard probe the database schema and detect the columns' content type and data type.

9. As noted, the data mining model needs to know the content type and data type of the columns. You can specify the content type manually or ask the wizard to detect it for you by probing the database schema. We will gladly take advantage of the second approach. Click on the Detect button to let the wizard discover the column's content type and data type. Click Next.

10. In the Completing the Wizard step (Figure 7.14), name the data mining model **Targeted Campaign**. Name the Mining model **TargetedCampaignDT** to denote the fact that the model uses the Microsoft Decision Trees algorithm (recall that one structure may include several data mining models). Check the **Allow drill through** checkbox. We will find out what it does in a

moment. Click Finish to let the Data Mining Wizard create the model and open it in the Data Mining Designer.

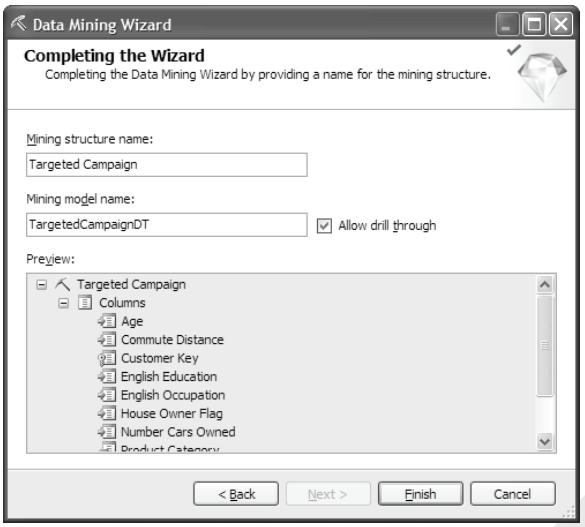

Figure 7.14 Enable drillthrough to let the end user see the source data for the model.

Introducing the Data Mining Designer

Undoubtedly, you will find the Data Mining Designer (Figure 7.15) very similar to the Dimension Designer. It has a tabbed user interface consisting of five tabs. As you would use the Dimension Structure tab to make changes to dimension attributes, use the Mining Structure tab to make changes to the mining structure. Examples of typical structure tasks you can perform are adding columns or nested tables, renaming columns, deleting columns, etc. Remember that any changes of the mining structure will require re-processing the structure.

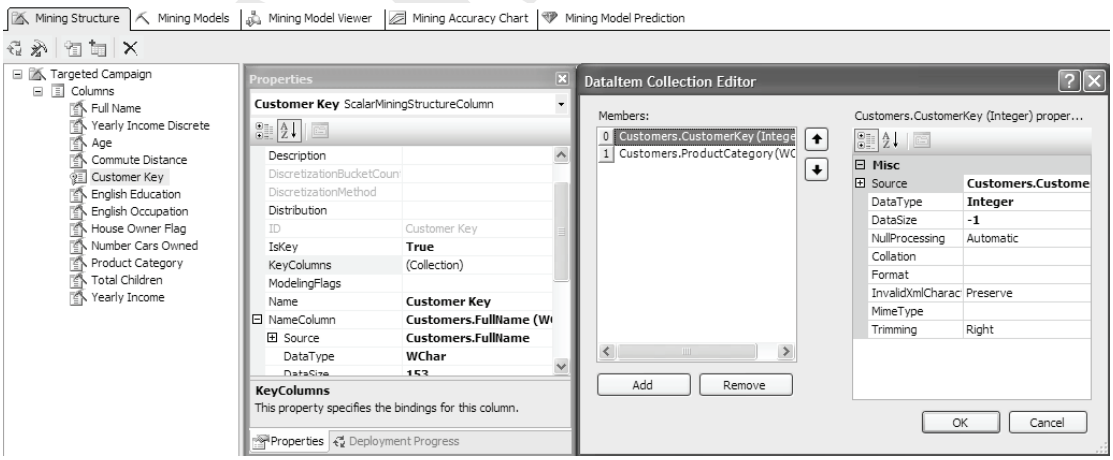

Figure 7.15 Use the Properties window to set the column properties and bindings.

One outstanding task we need to address is changing the structure key to qualify the input cases correctly. As it stands, the structure key uses the CustomerKey column only. As a result, if the case dataset has duplicated values in the CustomerKey column, these values will be ignored. This will be the case when a customer has purchased more than one bike. Indeed, if you run the Customers named query that feeds the structure, you will find that it brings 10,066 rows.

However, if you process the structure and examine the results in the Mining Model Viewer, you will find that the total number of cases contained in the structure is 7,273 only. As you know by now, the server ignores the duplicated keys when it processes the structure. We could keep this behavior and ignore the repeating customers but this will likely skew our results. Instead, let's use a key collection for the structure key. The net result of doing this is that a repeating customer will be treated as a new case and we will have 10,066 input cases.

1. In the Data Mining Designer, switch to the Mining Structure tab and select the CustomerKey column.
2. Click on the **...** button inside the KeyColumns property to open the DataItem Collection Editor.
3. Add the *ProductCategory* column to create a composite key consisting of the CustomerKey and ProductCategory columns (see again Figure 7.15).
4. Suppose you would like to see the customer name instead of the key when drilling through the customer data. Select the Customer Key column and create a new binding to bind the NameColumn property to the *FullName* column.

Adding a new mining model

The Mining Models tab (Figure 7.16) displays the mining models that are contained within the structure. Since we've just created the structure, it contains only the TargetCampaignDT model that uses the Microsoft Decision Trees algorithm. You can change the way a given column is used by the model. For example, if you decide not to use the Product Category column as an input to the Decision Trees algorithm, we can change its Usage from *Predict* to *PredictOnly*. You can also choose to ignore a column. For example, the customer's name is certainly not useful for mining purposes and we should ignore it to make our model more efficient. An algorithm may support various parameters that you can use to fine-tune the algorithm calculations. In most cases, the default settings are just fine. If you need to change the parameters, right-click on the model and choose *Set Algorithm Parameters*.

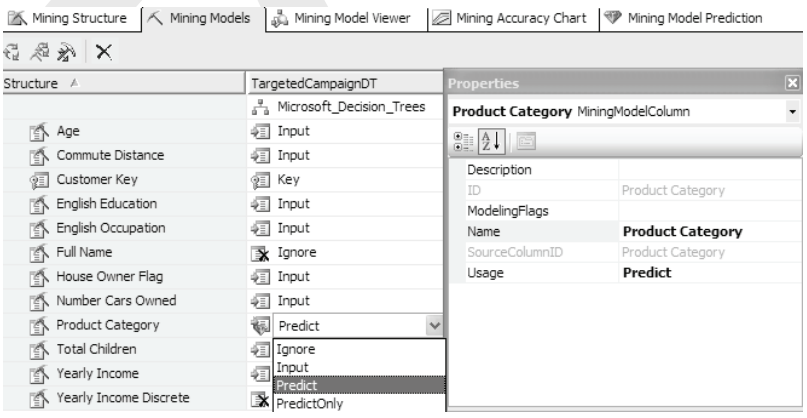

Figure 7.16 The columns of a structure may have different usage types in the containing model(s).

You may need to add additional models if you wish to compare the accuracy of different algorithms in order to choose the best fit for the task at hand. You may find the structure-model approach similar to the Controller-Viewer design pattern. You use the Mining Structure tab to define the schema definition and handle data storage and processing tasks (Controller) and then create multiple models (Viewer) on top of it to predict and show data in various ways. For example, a classification data mining task can be performed by the Microsoft Naïve Bayes algorithm as well. Follow these steps to add a new model to the Target Campaign mining structure that uses the Naïve Bayes algorithm.

1. Right-click anywhere inside the grid and choose *New Mining Model*. Alternatively, click the *Create a related mining model* toolbar button.
2. In the New Mining Model dialog, name the new model **TargetedCampaignNB** and select the *Microsoft Naïve Bayes* algorithm in the *Algorithm name* dropdown. Click OK to close the dialog.
3. The Data Mining Designer complains that the Naïve Bayes algorithm doesn't support the content type of the Age, Number Cars Owned, Total Children, and Yearly Income columns. That's because the content type of these column is continuous, while the Naïve Bayes algorithm supports only discrete columns. Accept the confirmation dialog to ignore these columns. The TargetedCampaignNB model is added as a new column in the grid. The usage type of the above four columns is set to *Ignore* and won't be used by the Naïve Bayes algorithm.

Creating a discrete column

But what if you need to use these columns? For example, there is probably a strong correlation between customer income and the type of bicycle a customer would buy. It is possible to use a continuous column with algorithms that don't support this content type provided that you discretize the column. To leave the Decision Tree model unaffected, we will add a new structure column (Yearly Income Discrete) to the Targeted Campaign mining structure and use it as an input to the TargetedCampaignNB model.

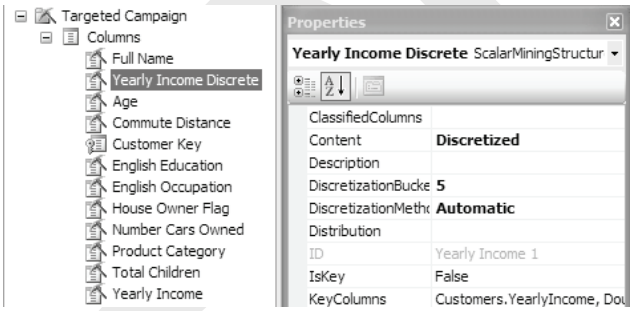

Figure 7.17 You can discretize structure columns with algorithms that don't support the continuous content type.

1. Switch to the Mining Structure tab. Right-click inside the Targeted Campaign column tree and choose *Add a Column*.
2. In the Select a Column dialog, choose the *YearlyIncome* column and click OK. A new column called *Yearly Income 1* is added to the structure.
3. Rename the new column **Yearly Income Discrete**. Click *Yes* in the confirmation dialog that follows to propagate the new name to the data mining models that use the column.

4. In the Properties window, change the column content to *Discretized*, the DiscretizationBucketCount property to 5 and leave the DiscretizationMethod to *Automatic* (see Figure 7.17). As a result of these changes, the server will group the customers into five income buckets.

5. Flip to the Mining Models tab and change the Usage type of the Yearly Income Discrete column to *Input*.

6. Deploy the project to process the mining structure and the two models. Alternatively, click on the Process button (the leftmost toolbar button) to process the structure. Processing the structure will load it with data from the Customer table and train the models.

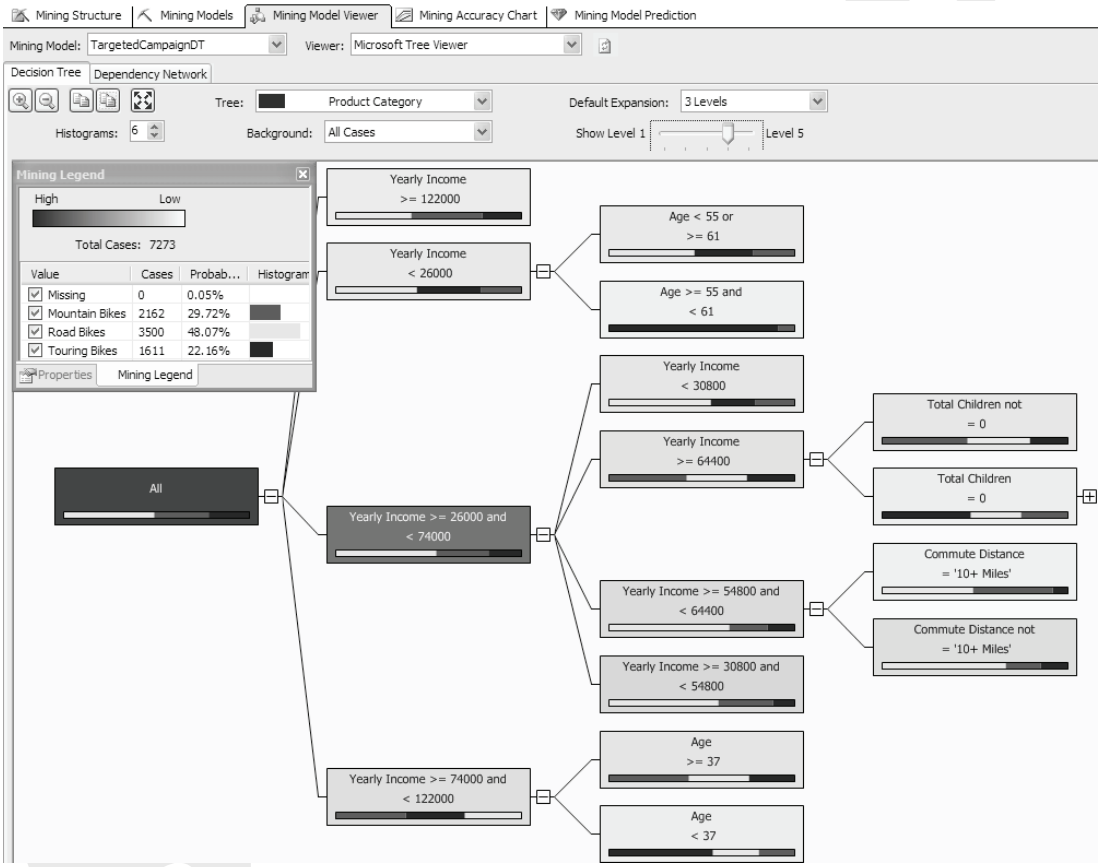

Figure 7.18 Use the Decision Tree Viewer to find the distribution of the customer population.

7.3.5 Exploring the Model

This is where the fun begins! We are ready to explore the predicted patterns found by our data mining model. Each Microsoft algorithm comes with its own viewer that is specifically designed to display the algorithm results optimally. We will start by examining the Decision Trees Viewer followed by the Naïve Bayes Viewer.

Interpreting the Decision Trees results

Switch to the Mining Model Viewer tab. The TargetedCampaignDT mining model should be selected by default in the Mining Model dropdown. The Decision Trees Viewer is displayed (Figure 7.18). Each predicted column generates a separate decision tree. Since, in our case, we have only one predicted column, there is only one entry in the Tree dropdown (Product Category). By default, the viewer displays the distribution of the customer population for all product categories (the Background dropdown is set to *All cases*).

Tip You can copy the graph as a picture by right-clicking on the graph and choosing the *Copy Graph View* or *Copy Entire Graph* context menus. If the viewer uses histogram charts, they will be copied as HTML.

Understanding the Decision Tree Graph

Let's see what educated conclusions we can make by inspecting the decision tree graph. As noted, the Microsoft Decision Trees algorithm is typically used for classification tasks. Each node of the tree corresponds to a customer class. When the background filter is set to *All Cases*, you can use the Decision Trees Viewer to find which factors may influence the customer decision to purchase a bike. For example, the model has determined that the most significant factor is customer income. That's why, the first split (after the root All node) is done on the customer age attribute.

The background color of each tree node is the most important visual indicator. The darker the color is, the larger the customer population. The root node of the tree is always the darkest. Hovering on top of the root node, or examining the mining legend, reveals that we have 10,066 cases (customers). From them, 32% have purchased mountain bikes, 47% road bikes, and 21% touring bikes. The same information can be derived approximately by glancing at the color bar inside each node. The bar has three colors because we have three distinct values in the predicted column; red for Mountain Bikes, yellow for Road Bikes, and blue for Touring Bikes).

Tracing the tree graph, we discover that the second darkest node represents 6,557customers with yearly income between $26,000 and $74,000. Therefore, this group of customers is most likely to purchase a bike. By clicking on this node and looking at the legend, we discover that more than half of these customers (56%) have purchased road bikes. Further, the Decision Trees algorithm has made another split on the customer income because it has discovered that, out of 6,557 customers, more than 5,000 customers are actually within the $26,000-$64,400 bracket.

Finally, the second most important factor after customer income is the number of cars a customer owns. The algorithm has concluded that customers with less than four cars are most likely to purchase a bicycle. Therefore, if the objective of our mining study is to identify potential bicycle buyers, this could be a good group to target.

What if the marketing department is interested in narrowing down the customers to those that are most likely to purchase a mountain bike? No problem. Change the Background dropdown to Mountain Bikes and voila! Interestingly, now we have two potential groups of customers we can target. First, we have customers with income greater than $74,000 and older than 37 years. Actually, this makes sense considering that mountain bikes tend to be more expensive. Second, we have a much smaller cluster of people with income between $64,400 and $74,000 who are 61 years old or older.

Drilling through

How can we see the individual customers that belong to a given group, so the marketing department can contact them to rekindle their interest in AW products (or spam them)? This is where the drillthrough feature comes in. As you would recall, we enabled this feature in the last

step of the Data Mining Wizard. Alternatively, since drilling through is a model-level feature, you can use the *AllowDrillThrough* property of a data mining model to enable or disable drilling through the model.

Note Similar to dimension or measure group drillthrough, model drillthrough is not enabled by default for security reasons because it allows end users to browse the source data for the model. The model designer has to make a conscious decision to enable this feature.

Once AllowDrillThrough is enabled, you can simply right-click on a tree node and choose the Drill Trough context menu to browse the source data forming the corresponding group. This action will pop up the Drill Through window with the cases displayed in a grid format. The classification criterion is shown on top. You can copy all records and paste them in a text file or an Excel spreadsheet.

Understanding the Dependency Network

In real life, your decision tree may have many splits and it may be difficult to find the most significant factors by using only the Decision Tree tab. The Dependency Network tab (Figure 7.19) is designed to help you find these attributes quickly. A Decision Trees model may have more than one predicted attributes. To filter the links that are correlated to a given predicted attribute, simply select that attribute in the diagram. In our case, we have four attributes (Yearly Income, Age, Commute Distance, and Number Cars Owned) that may influence a customer's decision to purchase a bike.

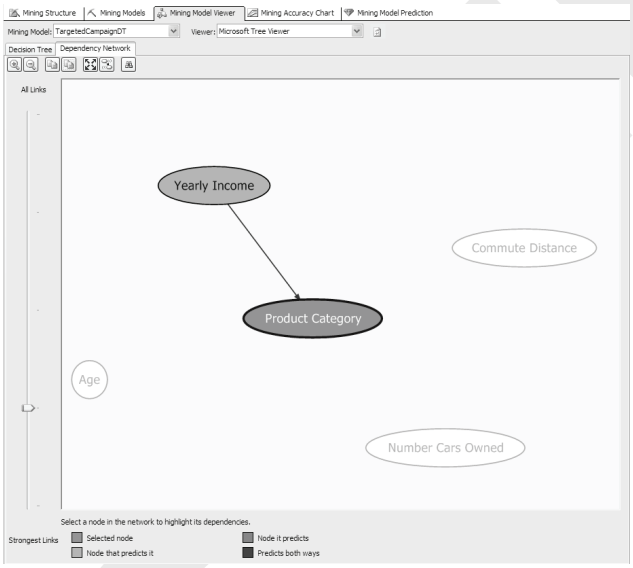

Figure 7.19 Use the Dependency Network graph to find quickly the links that have the strongest correlation factor.

To find the strongest links, slide down the slider on the left of the diagram. The slider scale will have as many nodes as the number of the distinct values of the predicted column. By sliding it down, you are in essence filtering out the less significant links. In the Targeted Campaign scenario, the Decision Trees algorithm finds that the most significant factor influencing AW customers to purchase a bike is the customer's yearly income (we don't need a mining model to figure this out, do we?). Backing up one nudge, we determine that the second important factor is customer's age.

Interpreting the Naïve Bayes results

Now, let's explore the results of the Naïve Bayes model by selecting the TargetedCampaignNB model in the *Mining model* dropdown.

Note The Naïve Bayes algorithm doesn't assume relationships among the input attributes. It is a pair-wise algorithm, meaning that it simply calculates the correlation between an input and a predict attribute, e.g. Age vs. Product Category. This is why, the algorithm is called **Naïve** Bayes. If correlating input columns is significant (e.g. Age vs. Income), you need to use more sophisticated algorithms, such as Microsoft Decision Trees. In addition, the Naïve Bayes algorithm doesn't support the drillthrough feature. As an upside, the Naïve Bayes algorithm is less computationally intensive and processes very fast.

The first tab is the Dependency Network tab algorithm:dwhich is the same as with the Decision Trees model. You can use it to get a good overall picture of the relative importance of the input columns. Since we decided to ignore the Age column when building the model, we have only three columns (attributes) with the Yearly Income Discretized being the most significant again.

Understanding the Attribute Profiles tab:attribute profiles

The Attribute Profiles tab (Figure 7.20) gives you a breakdown of the population of data in the input columns (attributes). The histogram has as many columns as the number of the discrete values of the predicted column, plus a column for *All* and missing values. For example, by just looking at the columns, we can deduce that there are 10,066 input cases (customers) from which 4,726 have bought Road Bikes, 2,125 Touring Bikes, and 3,215 Mountain Bikes.

The histogram shows the input attributes in rows and their discrete values (states) in the States column. You can roughly estimate the breakdown of the state population by inspecting the histogram columns. To find out the exact numbers, hover on top of the histogram. To determine the strongest correlation factors, examine their population distribution by comparing the Population (All) column and the predicted valued of interest.

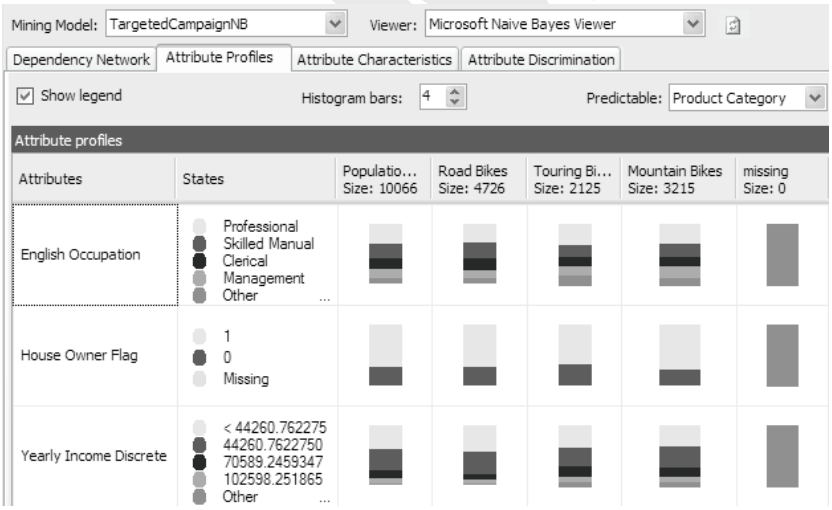

Figure 7.20 Use the Attribute Profiles chart to find the population distribution of the input attributes.

For example, if you inspect the Population column of the Yearly Income Discrete row, you can deduce that, overall, most customers have an income less than $44,000. By hovering on top of the histogram, or examining the mining legend (not shown), you can further find that this

segment contributes to about 40% of the overall population. However, if you now inspect the Mountain Bikes histogram, you will find that the majority of the customers purchasing mountain bikes are within the $44,000 – $70,000 income bracket (the second stacked bar from the top) because its bar is wider. This means that this customer segment could be potentially targeted during a mountain bike promotion campaign.

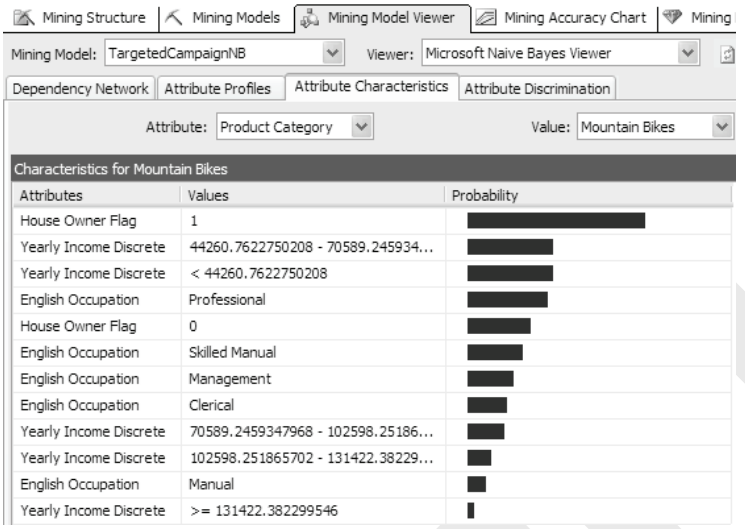

Figure 7.21 Use the Attribute Characteristics chart to determine the strongest correlation factors for a given class.

Understanding the Attribute Characteristics tabalgorithm:attribute characteristics

A more accurate picture of a particular customer class could be obtained by switching to the Attribute Characteristics tab (Figure 7.21). By examining the chart, we can deduce that most customers who purchase mountain bikes are home owners with an income between $44,000 and $70,000 and are professionals.

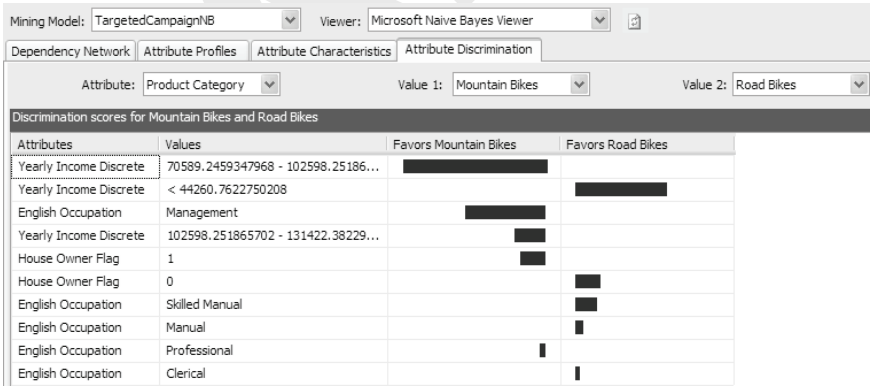

Figure 7.22 Use the Attribute Discrimination tab to compare profiles side by side.

Understanding the Attribute Discrimination tabalgorithm:d

Finally, you may be interested to compare two groups side by side. For example, you may need to find out how the Mountain Bikes customer profile compares against that of Road Bikes. You

can do this by using the Attribute Discrimination tab (Figure 7.22). Now we can conclude that customers who purchase mountain bikes tend to have higher income than those buying road bikes. In addition, the mountain bike buyers tend to have managerial or professional jobs, while the second group of customers tends to have manual or skilled manual occupations.

7.3.6 Evaluating the Model

Now that we've implemented two mining models for the targeted campaign scenario, how do we know which one is better? More importantly, how do we know if our data mining model predicts efficiently? We can use the Mining Accuracy Chart tab to evaluate the accuracy of our model(s) by following these steps:

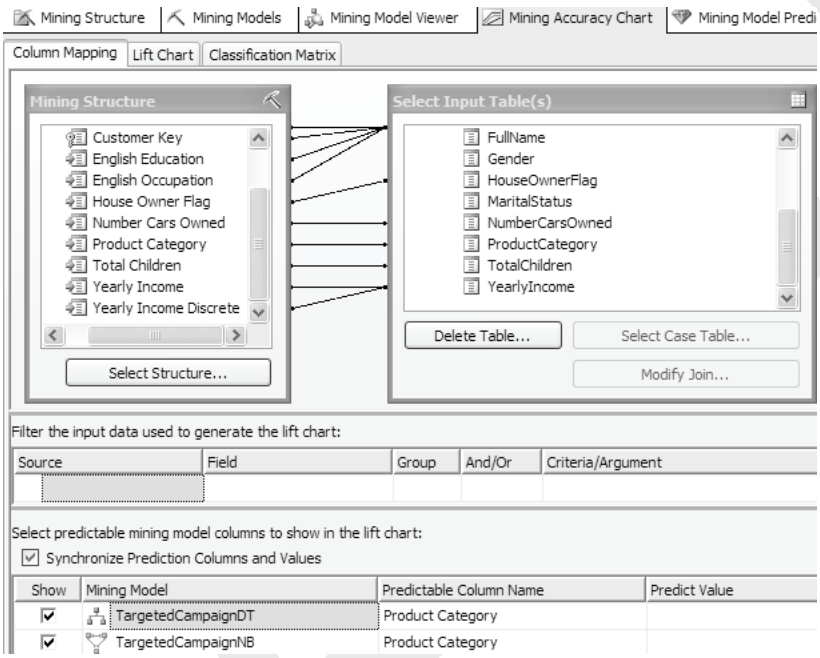

Figure 7.23 Use the Column Mapping tab to bind the columns of the mining structure to the columns of the test dataset.

Specifying column mappings

1. Click on the Mining Accuracy Chart tab. The Mining Accuracy Chart opens and its Column Mapping tab is selected (Figure 7.23). You can use the Column Mapping tab to specify an input table that holds the representative dataset that you want to test.
2. Click the Select Case Table button in the Select Input Table(s) pane and select the Customers table from the Bike Buyers DSV. For demo purposes, we will use the same input table that we used to train the model. In real life, it may make sense to prepare a smaller dataset that you can easily verify.
3. The Data Mining Designer tries to map automatically the columns of the mining structure and the input table by naming convention. If the columns have different names, you can link them manually by dragging a column from the Mining Structure pane and dropping it over the corresponding column in the Select Input Table(s) pane. Drag the Yearly Income Discrete

column which we created for the Naïve Bayes algorithm from the Mining Structure pane and drop it on the Yearly Income column in the Select Input Table(s) pane.

Optionally, you can filter the test dataset (e.g. Customers who are 25 years or older) by creating a filter criteria in the Filter grid. If you need to test the model accuracy for a particular value of the predicted column, you can specify this value in the Predict Value column of the *Select predictable columns*… grid. Leave the Predict Column empty for now to test the model accuracy irrespective of product category.

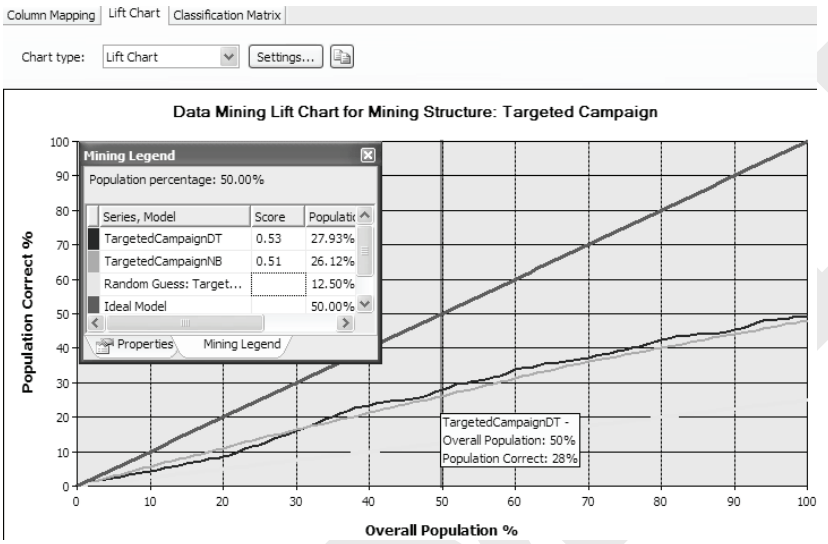

Figure 7.24 Create a lift chart to evaluate the model effectiveness.

Understanding lift charts

In our case, the customer dataset has about 10,000 customers. Suppose that, due to budget constraints, only 50% of the customer population will be targeted (5,000 customers). Naturally, not all customers will be interested in buying the new product. How can we identify the most likely buyers? Here is where a lift chart could help.

Flip the Lift Chart tab and notice that it has four lines, as shown in Figure 7.24 (the line closest to the X-axis is not easily discernable). In a perfect world, each targeted customer will respond to the campaign (100% penetration). This is what the topmost line represents. The bottom line shows the customer penetration rate if the customers are chosen at random. For example, if we are to pick 5,000 customers at random out of the entire 10,000 customer population, we will get only about 12% customer penetration.

However, if the marketing department is willing to try our model, they will get a much better response rate, as the middle two lines show. Any improvement over the random line is called a *lift*, and the more lift a model demonstrates, the more effective the model is. In our case, the darker line corresponds to the Decision Trees model, while the lighter one represents the Naïve Bayes model. As you can see, the Naïve Bayes model gives a better penetration results in a population of about 30% (3,000 customers). After that, the Decision Trees model outperforms the Naïve Bayes model.

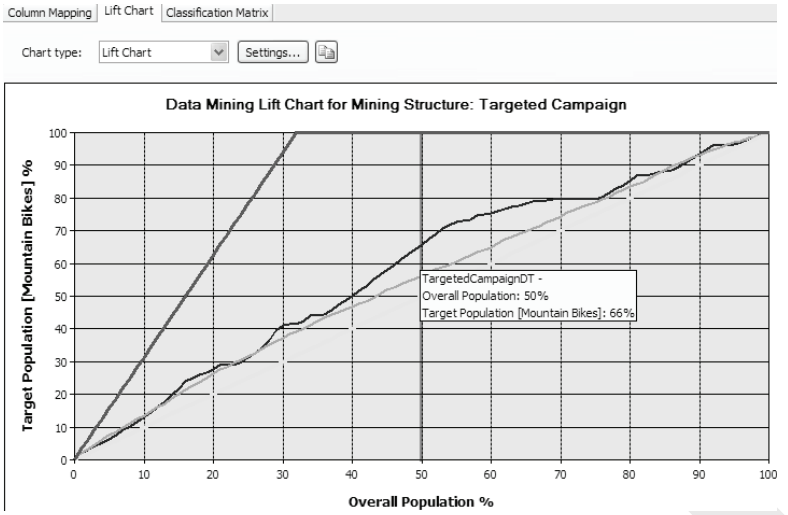

Figure 7.25 You can change the lift chart to evaluate a given predict value.

Now, hover on top of the TargetedCampaignDT line where it intersects with the 50% overall population vertical line. It turns out, that if we target 50% of the customers, the Decision Trees algorithm will give us 28% customer penetration rate, while if we select customers at random, we will get only 13% response rate.

Tip You can re-position the population line by clicking on a vertical grid line.

Now, suppose that the marketing campaign promotes mountain bikes only. Switch to the Column Mapping tab and change Chart Type dropdown to *Lift Chart*. Select *Mountain Bikes* in the Predict Value column of the bottom grid. Flip back to the Lift Chart tab and notice that its Y-axis caption has changed to *Target Population [Mountain Bikes] %* to reflect the fact that now we evaluate the model accuracy for the Mountain Bikes product category only (Figure 7.25).

The ideal model line reveals that, in a perfect world, we need to contact about 30% of the customers to identify all mountain bike buyers. Again, the Decision Trees algorithm outperforms Naïve Bayes. At 50% overall population, the Decision Trees model will identify 66% of the potential customers interested in mountain bikes. If you are not satisfied with the accuracy results, you have several options. Start by verifying the data in the input dataset. Make sure that it is a good representative of the task you need to model. Next, re-validate the structure and mining model definitions. Finally, consider using a different algorithm.

Creating a profit chart

Now, let's add a financial dimension to our model evaluation study and find how much profit each model could generate by creating a profit chart. Change the *Chart type* dropdown to *Profit Chart*. Click on the Settings button and enter the settings shown in Figure 7.26. In our hypothetical example, we would like to target 5,000 customers. The marketing department has estimated that the fixed cost for running the campaign will be $10,000, e.g. for designing, printing, and mailing the promotion materials. In addition, there is $5 individual overhead cost per customer, e.g. for customer representatives responding to customer questions. Finally, based on past

campaigns, the marketing department expects $50 as expected revenue per customer from the campaign.

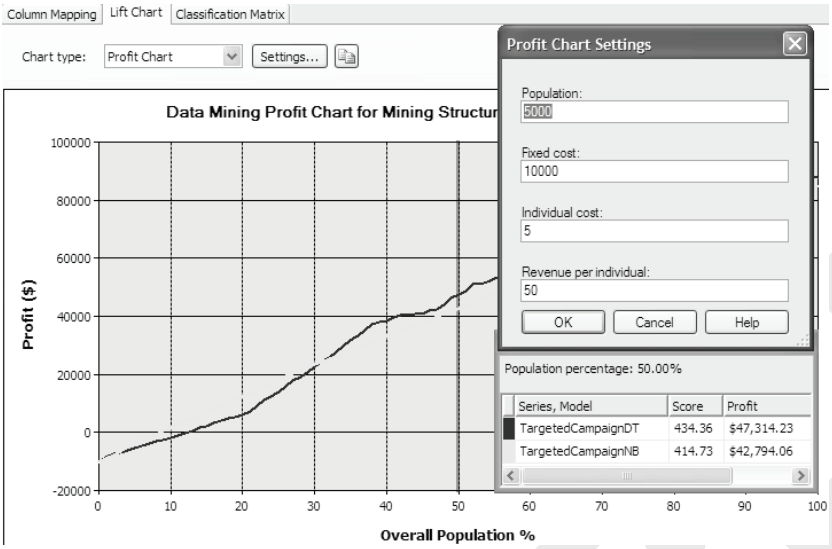

Figure 7.26 Create a profit chart to determine the profit generated by the campaign.

Based on these numbers, by examining the profit chart we could determine that we can actually lose money if we target only 10% of the customer population. Between 10% and 30%, the Naïve Bayes model will generate us more profit. After that, the Decision Trees model is the undisputed winner. If we target 50% of the customer population, we can expect our campaign to produce a profit of $47,314 with the Decision Trees algorithm. Using the profit chart, you can run what-if scenarios to find out the maximum return for your campaign investment. At this point, our models are trained and evaluated. Next, we need to deploy our model to production so it can be used for actual predictions against new customers.

Getting the predicted results

In the preceding steps, we used the Mining Accuracy Chart to compare the accuracy of the Decision Trees and Naïve Bayes models. We've identified that the Decision Trees model performs better for the classification task at hand. Specifically, the Decision Trees model could identify about 60% of the potential mountain bike buyers from a total population of 5,000 customers.

Suppose that Adventure Works Marketing department has a list of new customers. Let's run this list by the Decision Trees model to find which customers are likely to purchase a mountain bike based on the learned patterns. For the purposes of our demo, we will use the same input table (Customers) that we used to train the model. To get the model predictions, we need to query the model by submitting a DMX SELECT query. Creating the query is a child's play with the excellent DMX Query Builder provided by the Mining Model Prediction tab. Follow these steps to create a mining query to predict the mountain bike buyers.

1. Click on the Mining Model Prediction tab (Figure 7.27).

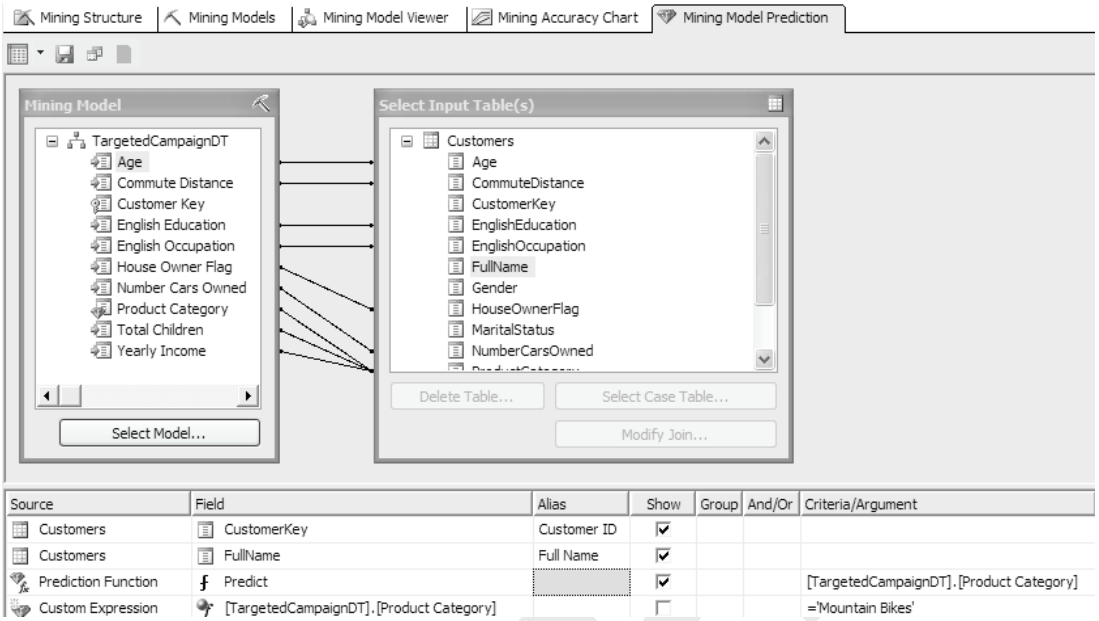

Figure 7.27 Use the Mining Model Prediction tab to build and execute a prediction query.

2. Click the Select Model button in the Mining Model pane. In the Select Mining Model dialog, expand the Targeted Campaign structure and select the *TargetedCampaignDT* model to use the Decision Tree model.

3. Next, we need to select the table that contains the input dataset. In the Select Input Table(s) pane, click the Select Case Table button and select the *Customers* table from the Bike Buyers DSV. Note that the Data Mining Designer automatically correlates the mining model and input table columns by naming convention.

   You can think of the Predict Mining Model tab as an ad hoc DMX query designer. As with the query designers included in the SQL Server Management Studio, it supports Design, Query, and Result modes. Let's use the Design mode to build our DMX query interactively.

4. First, let's select the columns we want to see in the predicted dataset. Drag the *CustomerKey* column from the input table (Select Input Table pane) and drop it onto the Source column of the grid. Alias the column as **Customer ID**.

5. Drag the FullName column and alias it as **Full Name**.

6. DMX supports various prediction functions. The function that we need is *Predict*. It allows you to predict a column from a given mining model. On the third row of the grid, expand the Source dropdown and select the *Prediction Function* item. Expand the Field column and select the first *Predict* function (there are two overloaded versions). Drag the *Product Category* column from the TargetedCampaign mining model to the Criteria/Argument column to replace the default criteria of the Predict function.

7. Since we are interested in predicting only potential mountain bike buyers, we need to filter the Product Category column. In the fourth row of the grid, expand the Source dropdown and select *Custom Expression*. Drag the Product Category column from the TargetedCampaignDT mining model to Field column. In the Criteria/Argument column, enter **= 'Mountain Bikes'**. Clear the *Show* checkbox since we don't want this column to show up in the results (it is just used for filtering).

8. Optionally, expand the left-most toolbar button and choose *Query* (or select Mining Model ⇨ Query menu) to view the underlying DMX statement. Note the query statement is actually an extended version of the T-SQL grammar and uses a PREDICTION JOIN construct to join the mining model and the input table.

9. Expand the left-most toolbar button and choose *Result* to submit the query to the server and retrieve the results in a tabular format. In my case, the query returns 3,595 customers that may be interested in purchasing mountain bikes. You can click the Save toolbar button to insert the results into a relational table.

Tip You can find my version of the DMX query saved as TargetedCampaign.dmx in the Code\Ch07\DMX folder. The DMX query builder is also available in the SQL Server Management Studio. To use it, connect to the Analysis Services database, expand the Mining Structures node, right-click on a given model, and choose Build Prediction Query. Alternatively, instead of using the query builder or the Mining Model Prediction tab, you can execute this query as a DMX query. To do so, right-click on the SOS OLAP database in the Object Explorer and choose New Query ⇨ DMX. Make sure that the TargetedCampaignDT model is selected in the *Mining model* dropdown in the Metadata pane. Next, paste the query text and execute the query by clicking the Execute button.

What if you want to specify the input values explicitly? For example, suppose you have a custom application that allows the end user to select a customer in order to find out if the customer is a likely buyer. In this case, you can use a singleton query. You can design this query by pressing the *Singleton query* button. The DMX Query Designer renames the right table to *Singleton Query Input* to denote the fact that now the values will be entered explicitly instead of retrieved from a relational table. Once the query syntax is ready, you can embed it in your application to pass the input criteria on the fly. We will see how this could be done in chapter 17.

7.4 Summary

Data mining is an exciting business intelligence technology. When properly used, data mining can help you derive valuable knowledge from mountains of data. It is a complementing, rather than competing, technology to OLAP. Use OLAP to aggregate data. Use data mining to discover patterns and trends.

Follow the seven-step data mining process to design and implement a data mining model. A data mining structure may contain more than one data mining model. Once the model is ready, train it with input data. The Mining Model Designer gives you the necessary tools to explore and evaluate a trained mining model. Once the model is in production, retrieve the predicted results by querying the model.

7.5 Resources

OLE DB for Data Mining Specification
(http://shrinkster.com/52t) – The goal of the OLE DB for Data Mining Specification is to provide an industry standard for data mining so that different mining algorithms from various data mining ISVs can be plugged easily into user applications.

Data mining models and structures and *Data mining algorithms* (part 1 and 2) webcasts
(SQL Server Resource Kit) – Your essential resource for introducing you to the SSAS data mining

CRoss Industry Standard Process for Data Mining (CRISP-DM)
(http://shrinkster.com/62h) – The CRISP-DM methodology was conceived in late 1996. It contains the corresponding phases of a project, their respective tasks, and relationships between these tasks.

SQL Server Data Mining: Plug-In Algorithms article by Raman Iyer and Bogdan Crivat
(http://shrinkster.com/53b) – Describes how SQL Server 2005 Data Mining allows aggregation directly at the algorithm level.

Preparing and Mining Data with Microsoft SQL Server 2000 and Analysis Services book
(http://shrinkster.com/8c4) – This book demonstrates how to apply data mining to a real-world situation using Microsoft SQL Server 2000, Microsoft SQL Server 2000 Analysis Services, and Microsoft Visual Basic 6.0.