The MiningMart Approach to Knowledge Discovery in Databases

Katharina MORIK, Martin SCHOLZ
Department of Computer Science, Chair VIII,
Baroper Str. 301, D-44221 University of Dortmund
{morik,scholz}@ls8.cs.uni-dortmund.de

Abstract. Although preprocessing is one of the key issues in data analysis, it is still common practice to address this task by manually entering SQL statements and using a variety of stand-alone tools. The results are not properly documented and hardly re-usable. The MiningMart system presented in this chapter focusses on setting up and re-using best-practice cases of preprocessing data stored in very large databases. A meta-data model named M4 is used to declaratively define and document both, all steps of such a preprocessing chain and all the data involved. For data and applied operators there is an abstract level, understandable by human users, and an executable level, used by the meta-data compiler to run cases for given data sets. An integrated environment allows for a rapid development of preprocessing chains. Case adaptation to different environments is supported by just specifying all involved database entities in the target DBMS. This allows to re-use best-practice cases published on the Internet.

1 Acquiring Knowledge from Existing Databases

The use of very large databases has enhanced in the last years from supporting transactions to additionally reporting business trends. The interest in analyzing the data has increased. One important topic is customer relationship management with the particular tasks of customer segmentation, customer profitability, customer retention, and customer acquisition (e.g. by direct mailing). Other tasks are the prediction of sales in order to minimize stocks, the prediction of electricity consumption or telecommunication services at particular day times in order to minimize the use of external services or optimize network routing, respectively. The health sector demands several analysis tasks for resource management, quality control, and decision making. Existing databases which were designed for transactions, such as billing and booking, are now considered a mine of information, and digging knowledge from the already gathered data is considered a tool for building up an organizational memory. Managers of an institution want to be informed about states and trends of their business. Hence, they demand concise reports from the database department.

On-line Analytical Processing (OLAP) offers interactive data analysis by aggregating data and counting the frequencies. This already answers questions like the following:

- What are the attributes of my most frequent customers?
- Which are the frequently sold products?
- How many returns did I receive after my last direct mailing action?
- What is the average duration of stay in my hospital?

Reports that support managers in decision making need more detailed information. Questions are more specific, for instance:

- Which customers are most likely to sell their insurance contract back to the insurance company before it ends?
- How many sales of a certain item do I have to expect in order to not offer empty shelfs to customers and at the same time minimize my stock?
- Which group of customers best answers to direct mailing advertising a particular product?
- Who are the most cost-intensive patients in my hospital?

Knowledge Discovery in Databases (KDD) can be considered a high-level query language for relational databases that aims at generating sensible reports such that a company may enhance its performance. The high-level question is answered by a *data mining* step. Several data mining algorithms exist. However, their application is still a cumbersome process. Several reasons explain, why KDD has not yet become a standard procedure. We list here the three obstacles that – in our view – are the most important ones and then discuss one after the other.

- Most tools for data mining need to handle the data internally and cannot access the database directly. Sampling the data and converting them into the desired format enhances the effort for data analysis.
- Preprocessing of the given data is decisive for the success of the data mining step. Aggregation, discretization, data cleaning, the treatment of null values, and the selection of relevant attributes are steps that still have to be programmed (usually in SQL) without any high-level support.
- The selection of the appropriate algorithm for the data mining step as well as for preprocessing is not yet well understood, but remains the result of a trial and error process.

The conversion of given data into the formats of diverse data mining tools is eased by toolboxes which use a common representation language for all the tools. Then, the given data need to be transformed only once and can be input into diverse tools. A first approach to such a toolbox was the development of a Common Knowledge Representation Language (CKRL), from which translators to several learning algorithms were implemented in the European project *Machine Learning Toolbox* [3, 11]. Today, the *weka* collection of learning algorithms implemented in JAVA with a common input format offers the opportunity to apply several distinct algorithms on a data set [15]. However, these toolboxes do not scale up to real-world databases naturally[1]. In contrast, database management systems offer basic statistical or OLAP procedures on the given data, but do not yet provide users with more sophisticated data mining algorithms. Building upon the database facilities and integrating data mining algorithms into the database environment will be the synergy of both developments. We expect the first obstacle for KDD applications to be overcome very soon.

The second obstacle is the most important one. If we inspect real-world applications of knowledge discovery, we realize that up to 80 percent of the efforts are spent on the clever preprocessing of the data. Preprocessing has long been underestimated, both, in its relevance and in its complexity. If the data conversion problem is solved, the preprocessing is not at all done. Feature generation and selection[2] (in databases this means to construct additional columns and select the relevant attributes for further

[1]Specialized on multi-relational learning algorithms, the *ILP toolbox* from Stefan Wrobel (to be published in the network ILPnet2) allows to try several logic learning programs on a database.

[2]Specialized on feature generation and selection, the toolbox YALE offers the opportunity to try and test diverse feature sets for learning with the support vector machine [6]. However, the YALE environment does not access a database.

learning) is a major challenge for KDD [9]. Machine learning is not restricted to the data mining step, but is also applicable in preprocessing. This view offers a variety of learning tasks that are not as well investigated as is learning classifiers. For instance, an important task is to acquire events and their duration (i.e. a time interval) on the basis of time series (i.e. measurements at time points). Another example is the replacement of null values in the database by the results of a learning algorithm. Given attributes A_i without null values, we may train our algorithm to predict the values of attribute A_j on those records, which do have a value for A_j. The learning result can then be applied in order to replace null values in A_j. Records without null values are a prerequisite for the application of some algorithms. These algorithms become applicable as the data mining step because of the learning in the preprocessing. With respect to preprocessing, we are just beginning to explore our opportunities. It is a field of greatest potential.

The third obstacle, the selection of the appropriate algorithm for a data mining task has long been on the research agenda of machine learning. The main problem is, that nobody has yet been able to identify reliable rules predicting when one algorithm should be superior to others. Beginning with the *Mlt-Consultant* [13] there was the idea of having a knowledge-based system support the selection of a machine learning method for an application. The *Mlt-Consultant* succeeded in differentiating the nine learning methods of the Machine Learning Toolbox with respect to specific syntactic properties of the input and output languages of the methods. However, there was little success in describing and differentiating the methods on an application level that went beyond the well known classification of machine learning systems into classification learning, rule learning, and clustering. Also, the European *Statlog*-Project [10], which systematically applied classification learning systems to various domains, did not succeed in establishing criteria for the selection of the best classification learning system. It was concluded that some systems have generally acceptable performance. In order to select the best system for a certain purpose, they must each be applied to the task and the best selected through a test-method such as cross-validation. Theusinger and Lindner [14] are in the process of re-applying this idea of searching for statistical dataset characteristics necessary for the successful applications of tools. An even more demanding approach was started by Engels [4]. This approach not only attempts to support the selection of data mining tools, but to build a knowledge-based process planning support for the entire knowledge discovery process. To date this work has not led to a usable system [5]. The European project *MetaL* now aims at learning how to combine learning algorithms and datasets [2]. Although successful in many respects, there is not enough knowledge available in order to propose the correct combination of preprocessing operations for a given dataset and task. The *IDEA* system now tries the bottom-up exploration of the space of preprocessing chains [1]. Ideally, the system would evaluate all possible transformations in parallel, and propose the most successful sequence of preprocessing steps to the user. For short sequences and few algorithms, this approach is feasible. Problems like the collection of all data concerning one customer (or patient) from several tables, or the generation of most suitable features enlarge the preprocessing sequences considerably. Moreover, considering learning algorithms as preprocessing steps enlarges the set of algorithms per step. For long sequences and many algorithms this principled approach of *IDEA* becomes computationally infeasible.

If the pairing of data and algorithms is all that difficult, can we support an application developer at all? The difficulty of the principled approaches to algorithm selection is that they all start from scratch. They apply rules that pair data and algorithm characteristics, or plan a sequence of steps, or try and evaluate possible sequences for each application anew. However, there are similar applications where somebody has already done the cumbersome exploration. Why not using these efforts to ease the new application development? Normally, it is much easier to solve a task if we are informed about the solution of a similar task. This is the basic assumption of case-based reasoning and it is the basis of the MiningMart approach. A successful case of a full KDD process is described at the meta-level. This description at the meta-level can be used as a blueprint for other, similar cases. In this way, the MiningMart project[3] eases preprocessing and algorithm selection in order to make KDD an actual high-level query language accessing real world databases.

2 The MiningMart Approach

Now that we have stated our goal of easing the KDD process, we may ask: What is MiningMart's path to reaching the goal? A first step is to implement operators that perform data transformations such as, e.g., discretization, handling null values, aggregation of attributes into a new one, or collecting sequences from time-stamped data. The operators directly access the database and are capable of handling large masses of data.

Given database oriented operators for preprocessing, the second step is to develop and collect successful cases of knowledge discovery. Since most of the time is used to find chains of operator applications that lead to good answers to complex questions, it is cumbersome to develop such chains over and over again for very similar discovery tasks and data. Currently, in practice even the same task on data of the same format is implemented anew every time new data are to be analyzed. Therefore, the re-use of successful cases speeds up the process considerably. The particular approach of the MiningMart project is to allow the re-use of cases by means of meta-data, also called *ontologies*. Meta-data describe the data as well as the operator chains. A compiler generates the SQL code according to the meta-data.

Several KDD applications have been considered when developing the operators, the method, and the meta-model. In the remaining part of this chapter, we shall first present the meta-data together with their editors and the compiler. We then describe the case base. We conclude the chapter by summarizing the MiningMart approach and relating it to other approaches.

2.1 The Meta-Model of Meta-Data M4

Ontologies or meta-data have been a key to success in several areas. For our purposes, the advantages of meta-data driven software generation are:

Abstraction: Meta-data are given at different levels of abstraction, a conceptual (abstract) and a relational (executable) level. This makes an abstract case understandable and re-usable.

Data documentation: All attributes together with the database tables and views, which are input to a preprocessing chain are explicitly listed at both, the conceptual and relational part of the meta-data level. An ontology allows to organize all data by means of inheritance and relationships between concepts. For all entities involved, there is a text field for documentation. This makes the data much more understandable, e.g. by human domain experts, than just referring to the names of specific database objects. Furthermore, statistics and important features for data mining (e.g., presence of null values) are accessible as well. This extends the meta-data as are usual in relational databases and gives a good impression of the data sets at hand.

Case documentation: The chain of preprocessing operators is documented, as well. First of all the declarative definition of an executable case in the M4 model can already be considered to be documentation. Furthermore, apart from the opportunity to use "speaking names" for steps and data objects, there are text fields to document all steps of a case together with their parameter settings. This helps to quickly figure out the relevance of all steps and makes cases reproducable.

[3]The MiningMart project is supported by the European Union under the contract IST-11993.

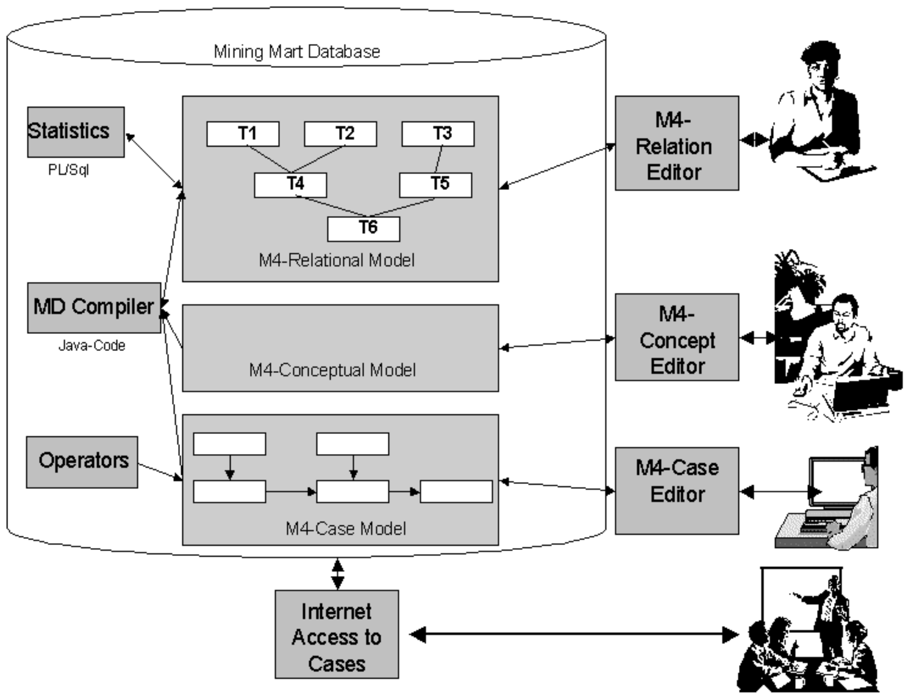

Figure 1: Overview of the MiningMart system

In contrast, the current state of documentation is most often the memory of the particular scientist who developed the case.

Ease of case adaptation: In order to run a given sequence of operators on a new database, only the relational meta-data and their mapping to the conceptual meta-data has to be written. A sales prediction case can for instance be applied for different kinds of shops, or a standard sequence of steps for preparing time series for a specific learner might even be applied as a template in very different mining contexts. The same effect eases the maintanance of cases, when the database schema changes over time. The user just needs to update the corresponding links from the conceptual to the relational level. This is especially easy, having all abstract M4 entities documented.

The MiningMart project has developed a model for meta-data together with its compiler, and has implemented human-computer interfaces that allow database managers and case designers to fill in their application-specific meta-data. The system will support preprocessing and can be used stand-alone or in combination with a toolbox for the data mining step.

This section gives an overview of how a case is represented at the meta-level, how it is practically applied to a database, and which steps need to be performed, when developing a new case or adapting a given one.

The form in which meta-data are to be written is specified in the meta-model of meta-data, M4. It is structured along two dimensions, topic and abstraction. The *topic* is either the data or the case. The data are the ones to be analyzed. The case is a sequence of (preprocessing) steps. The *abstraction* is either conceptual or relational. Where the conceptual level is expected to be the same for various applications, the relational level actually refers to the particular database at hand. The conceptual data model describes concepts like Customer and Product and relationships between them like Buys. The relational data model describes the business data that are analyzed. Most often it already exists in the database system in the form of the database schema. The meta-data written in the form as specified by M4 are stored in a relational

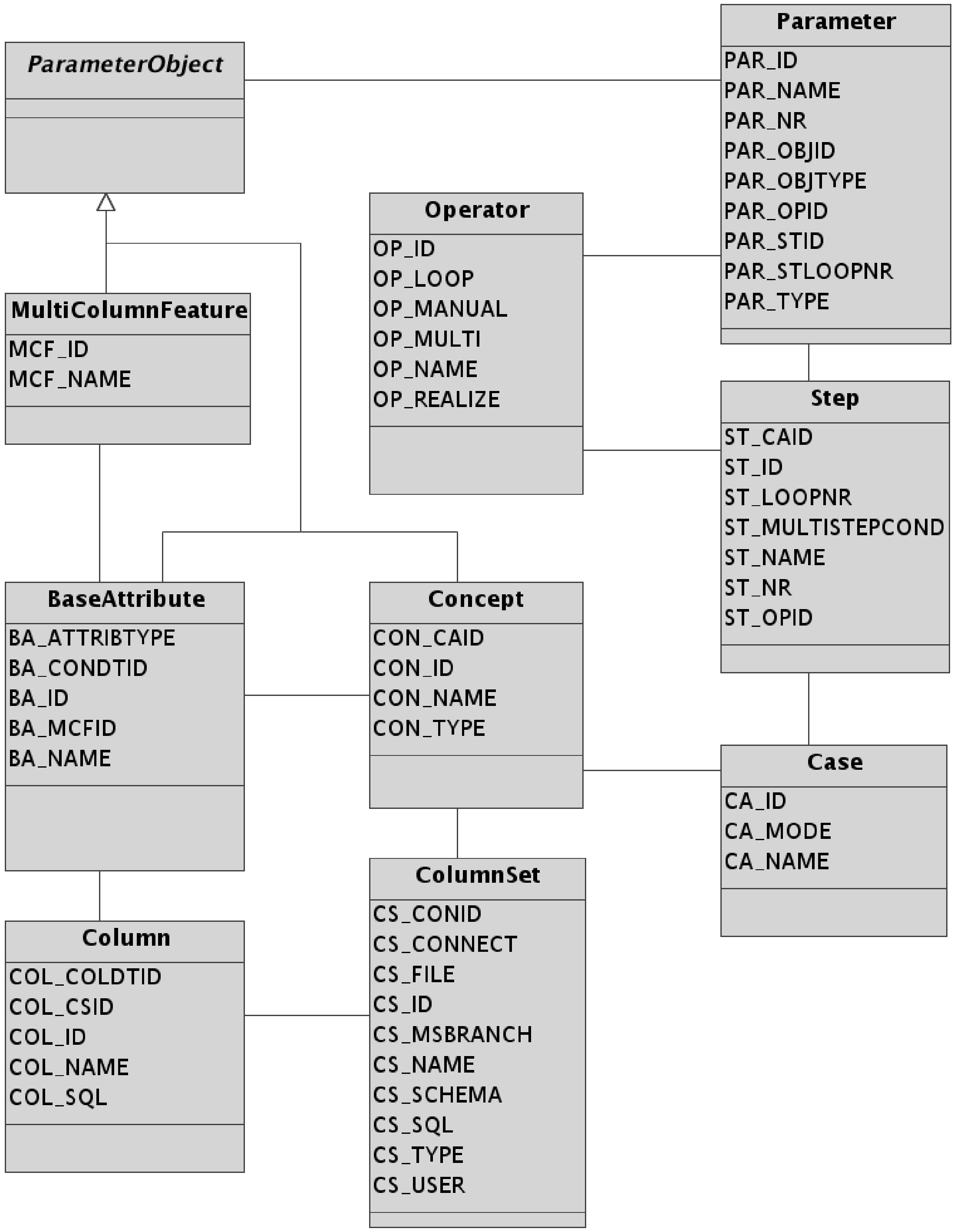

Figure 2: Simplified UML diagram of the MiningMart Meta Model (M4)

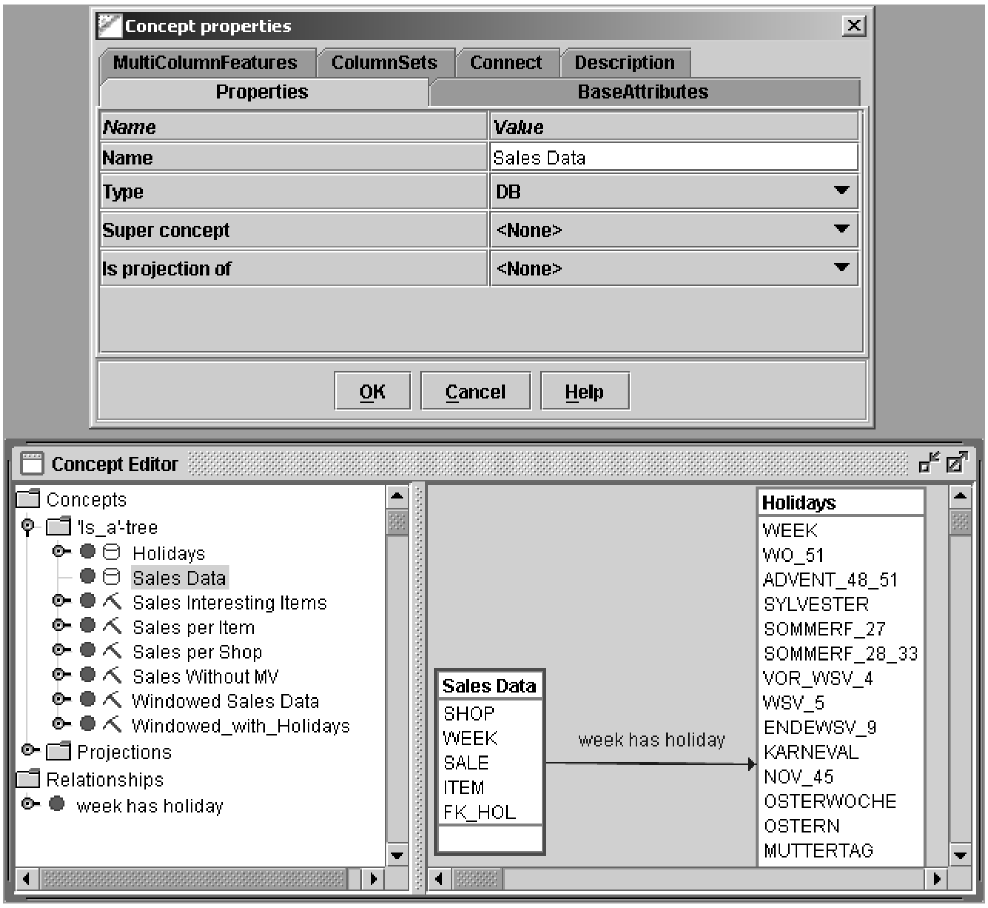

Figure 3: The Concept Editor

database themselves.

Figure 2 shows a simplified UML diagram of the M4 model. Each case contains steps, each of which embeds an operator an parameters. Apart from values, not shown here, parameters may be concepts, base attributes, or a multi column feature, a feature containing multiple base attributes. This part is a subset of the conceptual part of M4. The relational part contains columnsets and columns. Columnsets either refer to database tables, or to virtual (meta-data only) or database views. Each columnset consists of a set of columns, each of which refers to a database attribute. On the other hand columns are the relational counterpart of base attributes. For columns and base attributes there is a predefined set of data types, which is also omitted in Figure 2.

2.2 *Editing the Conceptual Data Model*

As depicted in Figure 1, there are different kinds of experts working at different ends of a knowledge discovery process. First of all a domain expert will define a conceptual data model, using a concept editor. The entities involved in data mining are made explicit by this expert. The conceptual model of M4 is about *concepts* having *features*, and *relationships* between these concepts.

Examples for concepts are Customer and Product. Although at the current stage of development concepts refer to either database views or tables, they should rather be considered as part of a more abstract model of the domain. Concepts consist of features, either base attributes or multi column fetures. A base attribute corresponds to a single database attribute, e.g. the name of a customer. A multi column feature

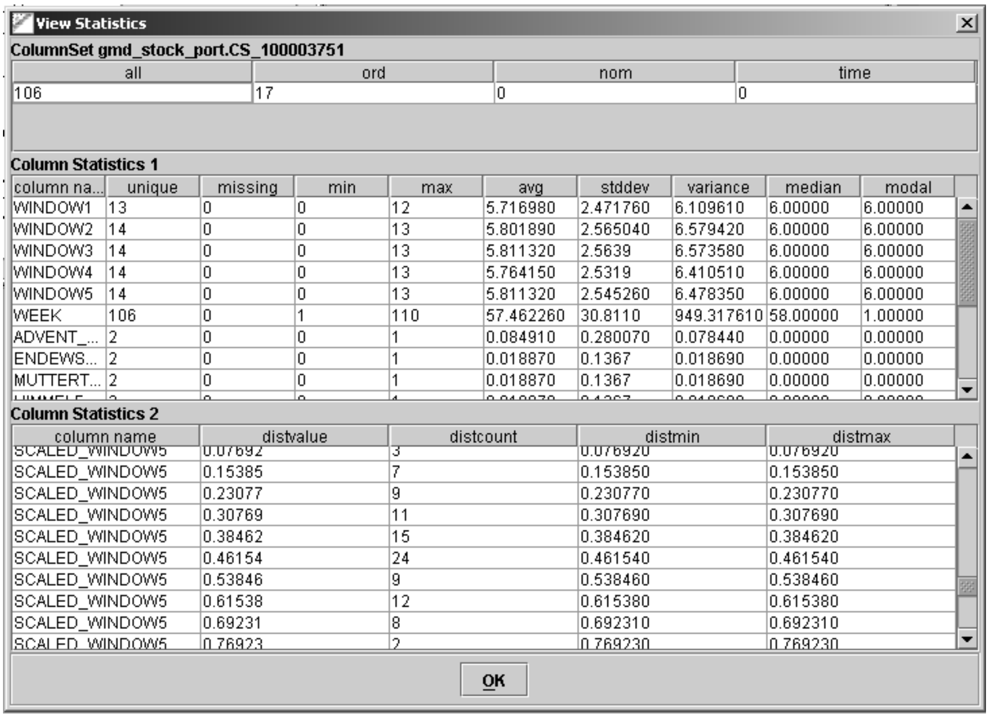

Figure 4: Statistics of a database view

is a feature containing a fixed set of base attributes. This kind of feature should be used, when information is split over multiple base attributes. An example is to define a single multi column fetaure for the amount and the currency of a bank transfer, which are both represented by base attributes.

Relationships are connections between concepts. There could be a relationship named Buys between the concepts Customer and Product, for example. At the database level one-to-many relationships are represented by foreign key references, many-to-many relationships make use of cross tables. However, these details are hidden from the user at the abstract conceptual level.

To organize concepts and relationships the M4 model offers the opportunity to use inheritance. Modelling the domain in this fashion, the concept Customer could have *subconcepts* like Private Customer and Business Customer. Subconcepts inherit all features of their superconcept. The relationship Buys could for instance have a subrelationship Purchases on credit.

Figure 3 shows a screenshot of the concept editor, while it is used to list and edit base attributes. The right part of the lower window states, that the selected concept Sales Data is connected to another concept Holidays by a relationship week has holiday.

2.3 Editing the Relational Model

Given a conceptual data model, a database administrator maps the involved entities to the corresponding database objects. The relational data model of M4 is capable of representing all the relevant properties of a relational database. The most simple mapping from the conceptual to the relational level is given, if concepts directly correspond to database tables or views. This can always be achieved manually by inspecting the database and creating a view for each concept. However, more sophisticated ways of graphically selecting features in the database and aggregating them to concepts increase the acceptance by end users and ease the adaptation of cases to other environ-

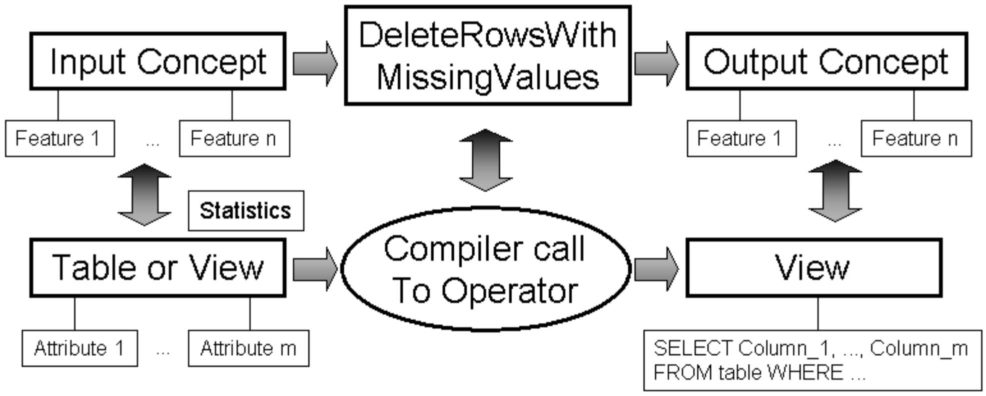

Figure 5: An illustration of the coupling of the abstract conceptual and executable level.

ments. In the MiningMart project, the relational editor is intended to support this kind of activity. In general it should be possible to map all reasonable representations of entities to reasonable conceptual definitions. A simple mapping of the concept Customer, containing the features Customer ID, Name, Address to the database would be to state that the table CUSTOMER holds all the necessary attributes, e.g. CUSTOM_ID, CUST_NAME and CUST_ADDR. Having the information about name and address distributed over different tables (e.g. sharing the key attribute CUSTOM_ID) is an example for more complex mappings. In this case the relation editor should be able to use a join operation.

Apart from connecting conceptual to database entities, the relation editor offers a data viewer and is capable of displaying statistics of connected views or tables. Figure 4 shows an example of the statistics displayed. For each view or table the number of tuples and the numbers of nominal, ordinal and time attributes are counted. For numerical attributes the number of different and missing values is displayed, the minimum, maximum, average, median and modal value are calculated together with the standard deviation and variance. For ordinal and time attributes the most reasonable subset of this information is given. Finally we have information on the distribution of the values for all attributes.

2.4 The Case and Its Compiler

All the information about the conceptual descriptions and about the according database objects involved are represented within the M4 model and stored within relational tables. M4 *cases* denote a collection of steps, basically performed sequentially, each of which changes or augments one or more concepts. Each step is related to exactly one M4 *operator*, and holds all of its input arguments. The M4 compiler reads the specifications of steps and executes the according operator, passing all the necessary inputs to it. This process requires the compiler to translate the conceptual entities, like input concepts of a step, to the corresponding relational entities, like database table name, the name of a view or the SQL definition of a virtual view, which is only defined as relational meta-data in the M4 model.

Two kinds of operators are distinguished, manual and machine learning operators. Manual operators just read the M4 meta-data of their input and add an SQL-definition to the meta-data for their output, establishing a virtual table. Currently, the MiningMart system offers 20 manual operators for selecting rows, selecting columns, handling time data, and generating new columns for the purposes of, e.g., handling null values, discretization, moving windows over time series, gathering information concerning an individual (e.g.,customer, patient, shop).

External machine learning operators on the other hand are invoked by using a wrapper approach. Currently, the MiningMart system offers learning of decision trees, k-means, and the support vector machine as learning preprocessing operators[4]. The necessary business data are read from the relational database tables, converted to the required format and passed to the algorithm. After execution the result is read by the wrapper, parsed, and either stored as an SQL-function, or materialized as additional business data.

In any case the M4 meta-data will have to be updated by the compiler. A complex machine learning tool to replace missing values is an example for operators altering the database. In contrast, for operators like a join it is sufficient to *virtually* add the resulting view together with its corresponding SQL-statement to the meta-data.

Figure 5 illustrates, how the abstract and the executable or relational level interact. First of all just the upper sequence is given, an input concept, a step, and an output concept. The concept definitions contain features, the step contains an operator together with its parameter settings. Apart from operator specific parameters, the input and output concept are parameters of the step, as well. The compiler needs the inputs, e.g. the input concept and its features to be mapped to relational objects before execution. The mapping may either be defined manually, using the relation editor, or it may be a result of executing the preceeding step. If there is a corresponding relational database object for each input, then the compiler executes the embedded operator. In the example this is a simple operator named “DeleteRowsWithMissingValues”. The corresponding executable part of this operator generates a view definition in the database and in the relational meta-data of M4. The latter is connected to the conceptual level, so that afterwards there is a mapping from the output concept to a view definition. The generated views may be used as inputs to subsequent steps, or they may be used by other tools for the data mining step.

Following the overall idea of declarative knowledge representation of the project, known pre-conditions and assertions of operators are formalized in the M4 schema. Conditions are checked at runtime, before an operator is applied. Assertions help to decrease the number of necessary database accesses, because necessary properties of the data can be derived from formalized knowledge, saving expensive database scans. A step replacing missing values might be skipped, for instance, if the preceding operator is known not not produce any missing values. If a user applies linear scaling to an attribute, then all values are known to lie in a specific interval. If the succeeding operator requires all values to be positive, then this pre-condition can be derived from the formalized knowledge about the linear scaling operator, rather than to recalculate this property by another database scan.

The task of a case designer, ideally a data mining expert, is to find sequences of steps resulting in a representation well suited for the given data mining task. This work is supported by a special tool, the *case editor*. Figure 6 shows a screenshot of a rather small example case edited by this tool. Typically a preprocessing chain consists of many different steps, usually organized as a directed acyclic graph, rather than as a linear sequence as the example case shown in Figure 6. To support the case designer a list of available operators and their overall categories, e.g. feature construction, clustering or sampling is part of the conceptual case model M4. The idea is to offer a fixed set of powerful pre-processing operators, in order to offer a comfortable way of setting up cases on the one hand, and ensuring re-usability of cases on the other. By modeling real world cases in the scope of the project further useful operators will be identified, implemented and added to the repository.

For each step the case designer chooses an applicable operator from the collection, sets all of its parameters, assigns the input concepts, input attributes and/or input relations and specifies the output. To ease the process of editing cases, applicability constraints on the basis of meta-data are provided as formalized knowledge and are automatically checked by the human computer interface. This way only valid sequences of steps can be produced by a case designer. Furthermore, the case editor supports

[4]Of course, the algorithms may also be used in the classical way, as data mining step operators.

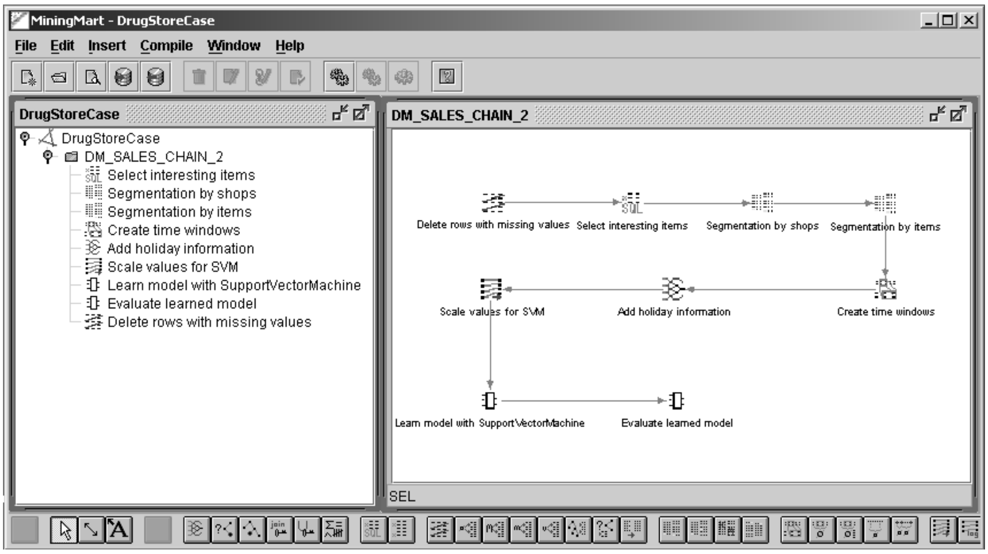

Figure 6: A small example case in the case editor.

the user by automatically defining output concepts of steps according to the meta-data constraints, and by offering property windows tailored to the demands of chosen operators.

A sequence of many steps, namely a *case* in M4 terminology, transforms the original database into another representation. Each step and their ordering is formalized within M4, so the system is automatically keeping track of the performed activities. This enables the user to interactively edit and replay a case or parts of it.

As soon as an efficient chain of preprocessing has been found, it can easily be exported and added to an Internet repository of best-practice MiningMart cases. Only the conceptual meta-data is submitted, so even if a case handles sensitive information, as given for most medical or business applications, it is still possible to distribute the valuable meta-data for re-use, while hiding all the sensitive data and even the local database schema.

3 The Case Base

One of the project's objectives is to set up a case-base of successful cases on the Internet. The shared knowledge allows all Internet users to benefit from a new case. Submitting a new case of best practice is a safe advertisement for KDD specialists or service providers, since the relational data model is kept private. To support users in finding the most relevant cases, their inherent structure will be exploited. An according Internet interface will be accessible, visualizing the conceptual meta-data. It will be possible to navigate through the case-base and to investigate single steps, i.e., which operators were used on which kind of concepts. The Internet interface is supposed to read the data directly from the M4 tables in the database, avoiding additional efforts and redundancies. Figure 7 shows a screenshot of a case's business level description. Additionally to the data explicitly represented in M4, a business level has been added. This level aims at relating the case to business goals and to give several kinds of additional descriptions, like which success criteria were important for the case. For instance, the sales prediction answers the question "How many sales of a particular item do I have to expect?" where the business goal is that it must not happen that the item is sold out, but the stock should be minimized. A particular application need is that the forecast can only be used if it predicts the sales 4 weeks ahead because of delivery times. Especially the more informal descriptions should help decision makers

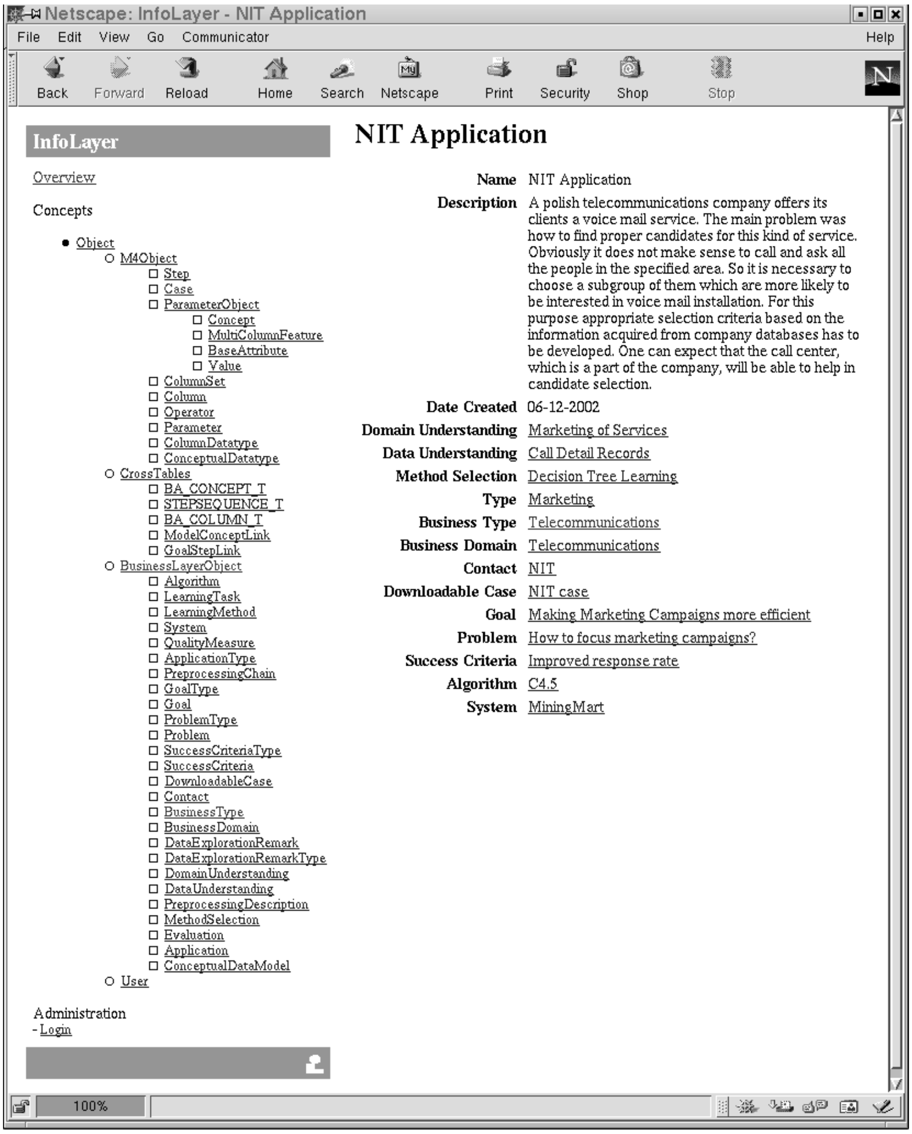

Figure 7: The Internet interface to the case base visualizes all cases, their steps, embedded operators, and parameters in HTML format. Entities related in the M4 schema are connected by hyperlinks. Additionally, a business level is part of the interface. It describe the available cases in terms like the addressed business goals of the data analysis. After deciding for a case with the help of conceptual M4 and business layer descriptions, the user can simply download the one addressing the most similar problem. The case adaption facilities of The MiningMart system helps to quickly adjust the case to the user's environment.

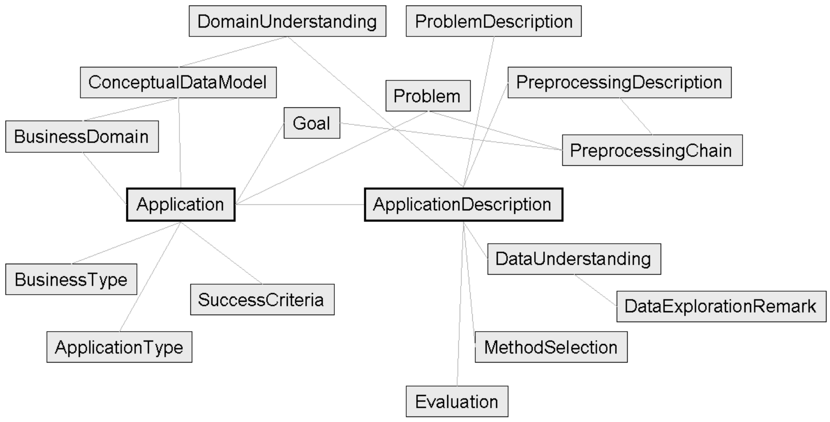

Figure 8: The ontology of the business layer, used to describe M4 cases in business terms.

to find a case tailored for their specific domain and problem. The additional information is stored in an XML-representation, directly connected to the M4 entities. On the Internet these connections are represented by hyperlinks. Figure 8 shows the ontology of the business level.

It is possible to start the search for a case at each category of the business level or conceptual level. In this sense the cases are indexed by all the categories part of the conceptual M4 model and the business model. If a user considers a case useful, then its conceptual data can be downloaded from the server. The downloadable case itself is a category in the XML framework. The locally installed MiningMart system offers an import facility, installing the meta-data into the user's M4 tables. If problems arise, or further help is necessary, the business level holds a category for the case designer or the company providing service.

The project has developed four cases:

- analysis of insurance data for direct mailing [8, 7],
- call center analyisis for marketing,
- analysis of data about calls and contracts for fraud detection in telecommunication, and
- analysis of sales data for sales prediction [12].

4 Related Work and Conclusion

The relevance of supporting not only single steps of data analysis but sequences of steps has long been underestimated. Where a large variety of excellent tools exist which offer algorithms for a data mining step, only very few approaches exist which tackle the task of making clever choices during preprocessing and combining these choices to an effective and efficient sequence. The *Clementine* system offers processing chains to users. However, the focus lies on the data mining step, not the preprocessing chain. The common data format in tool boxes such as, e.g. *Spss* or *weka* provides users with the prerequisites to formulate their own sequences [15]. However, the user is programming the sequence and has to do this anew for very similar tasks.

Zhong and colleagues have proposed an agent system, *GLS*, which supports the overall KDD process, i.e. preprocessing, knowledge elicitation, and refinement of the result [17, 16]. In some aspects, this system is similar to the MiningMart. Its agents are our operators, its controller corresponds to our compiler, both systems describe data and operators at the meta-level. Where in MiningMart the operator description entails applicability conditions and pointers to the resulting table, in *GLS* the pre- and post-conditions for the application of an agent are stated. The hierarchy of agents in *GLS* corresponds to the inheritance hierarchy of operators as exploited in MiningMart. In addition, MiningMart offers an even more abstract level for the description of a case in business terms. The planning approach of *GLS* is also similar to the use of applicability constraints as done in MiningMart. In contrast to *IDEA*, no comparison of quality is performed for alternative chains of operators. Hence, both, MiningMart and *GLS* produce valid sequences of steps, and none of them performs experiments – as does *IDEA* – in order to decide between several algorithms or agents. In addition to the similarities, the two systems do, of course, also differ. First, the set of algorithms (operators or agents, respectively) is different. Feature generation and selection – a focus of MiningMart – is less developed within *GLS*. Data mining algorithms are less complete in the MiningMart. This is no principled point, since both systems allow for easily enhancing the set of operators. Second, the relation to the database is different. The interaction between *GLS* and the database is not the primary focus the research in [17, 16]. In contrast, the MiningMart resides to a large degree within the database, compiles meta-data into SQL code, and many of its operators are directly integrated into the database. This allows to work on very large databases. Third, the use of human expertise is different. In *GLS*, some user interaction is required in order to optimize the automatically generated valid sequences. However, the notion of a complete case at the meta-level is not part of the meta-model. This means, that the diverse trials to establish an optimal sequence of agent activities are not documented. Hence, experience of failed selections, groupings, parameter settings cannot prevent users from doing so, again. Experience of successful cases is not stored at the meta-level. There is no mechanism to apply a successful chain to similar but different databases. In contrast, MiningMart compiles a successful case together with a meta-model of new data into a running new KDD case. We believe that the re-use of best-practice cases and the case documentation is extremely important.

The recent *Idea* system is also similar to the MiningMart approach [1]. Chains of operators are composed according to a ranking of algorithms in order to detect the best choice of an algorithm given data characteristics. Meta-data describing the data as well as the algorithms are used in order to check the validity of operator sequences or incorporate an additional step which allows to apply the best operator. The difference lies first in MiningMart's orientation towards very large databases. *Idea* uses the *weka* data format and, hence, is restricted to smaller files. The data transformations and aggregations incorporated as manual operators in the MiningMart system are not taken into account in *Idea*, because they are not needed in the single table small sample representation of *weka* tools. The second distinction is the approach to determining the best sequence of preprocessing. Although the MiningMart system exploits applicability conditions of operators in order to check the validity of sequences, it does not aim at planning the best sequence or perform a ranking of possible algorithms at each step of an operator chain, as IDEA can do. Instead, MiningMart exploits the findings of expert case designers. Real-world application of knowledge discovery comprise hundreds of steps in a KDD run (including manual operators) and ranking every algorithm at each of the steps would exhaust computing capacity. We feel that the adaptation of excellently solved KDD problems best combines human expertise and computational power.

We can now summarize the characteristics of the MiningMart:

Very large databases: It is a database oriented approach which easily interacts with all SQL-databases and scales up to real-world databases without any problems. Several operators have been re-implemented in order to make them ready for very large data sets.

Sophisticated operators for preprocessing: Preprocessing can make good use of learning operators as does the data mining step. For instance, a learning result can be used to replace missing values by the learned (predicted) values. Feature generation and selection in the course of preprocessing enhances the quality of data that are the input to the data minig step.

Meta-data driven code generation: The MiningMart approach relies on meta-driven software generation. Meta-data about operators and data are used by the compiler in order to generate a running KDD application.

Case documentation: Meta-data about a case document the overall KDD process with all operator selections and their parameter settings. In addition, a business layer offers the case description in less technical terms so that end-users of the KDD process are kept informed.

Case adaptation: The notion of a complete case in the meta-model allows to apply a given expert solution to a new database. The user only needs to provide the system with a new data model and the compiler generates the new case. For fine-tuning the new application, the human-computer interface offers easy access to the meta-model with all operators.

References

[1] Abraham Bernstein, Shawndra Hill, and Foster Provost. An Intelligent Assistant for the Knowledge Discovery Process. Technical Report IS02-02, New York University, Leonard Stern School of Business, 2002.

[2] Pavel Brazdil. Data Transformation and Model Selection by Experimentation and Meta-Learning. In C.Giraud-Carrier and M. Hilario, editors, *Workshop Notes – Upgrading Learning to the Meta-Level: Model Selection and Data Transformation*, number CSR-98-02 in Technical Report, pages 11–17. Technical University Chemnitz, April 1998.

[3] Karine Causse, Marc Csernel, Katharina Morik, and Céline Rouveirol. MLT Deliverable 2.2: Specification of the Common Knowledge Representation Language of the MLToolbox. GMD (German Natl. Research Center for Computer Science), P.O.Box 1240, W-5205 St. Augustin 1, Germany, September 1990.

[4] Robert Engels. Planning Tasks for Knowledge Discovery in Databases; Performing Task–Oriented User–Guidance. In *Proc. of th 2nd Int. Conf. on Knowledge Discovery in Databases*, August 1996.

[5] Robert Engels, Guido Lindner, and Rudi Studer. A Guided Tour through the Data Mining Jungle. In *Proceedings of the 3nd International Conference on Knowledge Discovery in Databases (KDD-97)*, August 14–17 1997.

[6] Simon Fischer, Ralf Klinkenberg, Ingo Mierswa, and Oliver Ritthoff. YALE: Yet Another Learning Environment – *tutorial*. Technical Report CI-136/02, Collaborative Research Center 531, University of Dortmund, Dortmund, Germany, 2002. ISSN 1433-3325.

[7] Joerg-Uwe Kietz, Regina Zuecker, Anna Fiammengo, and Giuseppe Beccari. Data Sets, Meta-data and Preprocessing Operators at Swiss Life and CSELT. Deliverable D6.2, IST Project MiningMart, IST-11993, 2000.

[8] Jörg-Uwe Kietz, Regina Zücker, and Anca Vaduva. Mining Mart: Combining Case-Based-Reasoning and Multi-Strategy Learning into a Framework to reuse KDD-Application. In R.S. Michalski and P. Brazdil, editors, *Proceedings of the fifth International Workshop on Multistrategy Learning (MSL2000)*, Guimares, Portugal, May 2000.

[9] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, 1998.

[10] D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, New York u.a., 1994.

[11] K. Morik, K. Causse, and R. Boswell. A Common Knowledge Representation Integrating Learning Tools. In *Proc. of the 1st International Workshop on Multistrategy Learning*, Harpers Ferry, 1991.

[12] Stefan Rueping. Zeitreihenprognose für Warenwirtschaftssysteme unter Berücksichtigung asymmetrischer Kostenfunktionen. Master's thesis, Universität Dortmund, 1999.

[13] D. Sleeman, R. Oehlman, and R. Davidge. Specification of Consultant–0 and a Comparision of Several Learning Algorithms. Deliverable D5.1, Esprit Project P2154, 1989.

[14] C. Theusinger and G. Lindner. Benutzerunterstützung eines KDD-Prozesses anhand von Datencharackteristiken. In F. Wysotzki, P. Geibel, and K. Schädler, editors, *Beiträge zum Treffen der GI-Fachgruppe 1.1.3 Machinelles Lernen (FGML-98)*, volume 98/11 of *Technical Report*. Technical University Berlin, 1998.

[15] Ian Witten and Eibe Frank. *Data Mining – Practical Machine Learning Tools and Techniques with JAVA Implementations*. Morgan Kaufmann, 2000.

[16] N. Zhong, C. Liu, and S. Ohsuga. A Way of Increasing both Autonomy and Versatility of a KDD System. In Z.W. Ras and A. Skowron, editors, *Foundations of Intelligent Systems*, pages 94–105. Springer, 1997.

[17] N. Zhong, C. Liu, and S. Ohsuga. Dynamically Organizing KDD Processes. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(3):451–473, 2001.