Mining Mart: Metadata-Driven Preprocessing

Regina Zücker[1], Jörg-Uwe Kietz[1], and Anca Vaduva[2]

[1] Swiss Life, Corporate Center, IT Research & Development, CH-8022 Zurich, Switzerland {regina.zuecker,uwe.kietz}@swisslife.ch
[2] University of Zurich, Dept. of Information Technology, CH-8057 Zurich, Switzerland vaduva@ifi.unizh.ch

Abstract. The Mining Mart project[1] (Enabling End-User Data Warehouse Mining) proposes a case-based reasoning system for maximum support of end users during data preprocessing. Our approach 1) uses a case base for efficient support for data preprocessing, respectively the persistent storage of cases as metadata, 2) machine learning tools for preprocessing, and 3) generates views as results of preprocessing operators. The stored metadata will be used by all system components during execution. This ensures reusability and automation of a KDD-process and conceals deep data mining knowledge from end users. The execution of cases takes place within a database server and guarantees proper handling of large amounts of data. Therefore, preprocessing operators must be implemented within the database server. In the Mining Mart project we did implement and apply such "database" operators and achieved good results on real business data.
In this paper, we will demonstrate how to receive a high level of automation and reusability of a KDD-process, based on metadata-driven preprocessing. End users will gain increased flexibility and transparency. The resulting time and cost savings will make the KDD-process applicable in industrial companies.

1 Introduction

Practical experiences [8] have shown that 50-80% of the efforts for knowledge discovery are spent on data preprocessing. It requires profound business, data mining and database know-how. The support of existing KDD-systems for data preprocessing is still not satisfying for end users.

Mining Mart provides a user-friendly environment to perform preprocessing for data mining on the basis of a case-based reasoning framework[5]. The framework provides cases that are already successfully applied, as well as system components to develop new cases. Considering all individual tasks of a KDD-process, we can identify user types with different background knowledge. The framework should offer appropriate support for any user type. Within this paper, a business case is the technical solution of a business task that is stored in the framework of

[1] http://www-ai.cs.uni-dortmund.de/FORSCHUNG/PROJEKTE/MININGMART/index.eng.html

the case base. It can either be re-applied without any changes or re-used within the specification phase of a new case.

The development process of a business case consists of three parts: specification, compilation and execution. The specification phase is most time consuming and requires profound knowledge in business data and data mining techniques. To guarantee reusability of a case, the specification must be persistently stored within the framework. Mining Mart system components automatically compile the specification and create code to be executed within a database server. The execution within the database server allows to manage large amounts of data, i.e. of a data warehouse.

In the Mining Mart project we have developed a Meta Model for storing all kinds of metadata within a KDD-process. Metadata exist about 1) the business case, 2) the preprocessing steps and their operators, and 3) the business data.

The remainder of this paper is organized as follows: the next section explains the KDD-process regarding reusability and easy adaptation of a case. In section 3 we consider data mining specific requirements for metadata. Section 4 introduces the Mining Mart architecture and the Meta Model. In section 5 we will present a short application example. We will show the realization-process for a business task within the framework and concrete possibilities for reuse. Section 6 shows the related work and section 7 concludes the paper.

2 Reusability of a KDD-Process

Nowadays business data of companies is reaching the amount of giga- and terabytes. The force to efficiently use information out of this huge amount of data is growing, too. But getting this information should be as quick and cheap as possible. Typically, companies consult external data mining experts and their experience to achieve good results. The goal of Mining Mart is to enable in-house people without profound data mining knowledge to gather high-quality information.

2.1 Requirements

If we want to broaden the usage of data mining as analysis method for business data in companies, we have to offer reusability of business domain knowledge as well as technical knowledge about a KDD-process. The reusability effort should be automatically supported by a system. A knowledge exchange should take place between experts and end users (implicitly by using the system) and also between experts. Such an exchange will motivate experts to publish their knowledge. Therefore, reusability of a KDD-process should fulfill two main goals:

First, the support data mining experts: Experts can gain domain knowledge of a business case from other experts by using an existing case during the specification phase. The system should support experts in giving advices for proper applicability of machine learning tools. Sharing this expert knowledge could result in time and cost savings when designing a new case and, more important, should enrich the knowledge of every single expert.

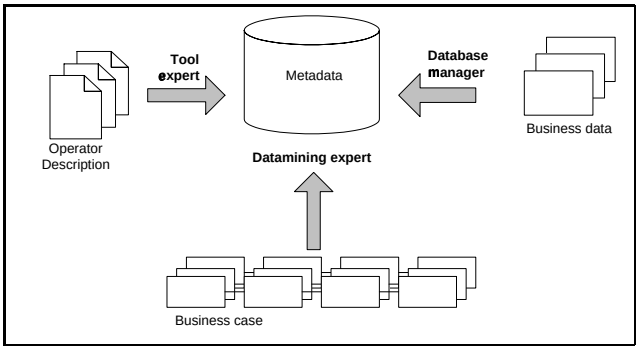

Fig. 1. Who stores Metadata in the System

Second, reusability respectively re-application of an existing case: Changes in the business data, either the underlying data schema or data contents, should be handled automatically by the system whenever possible. At that point massive cost savings could be achieved and data mining could become a real cost effective method. No data mining expert should be necessary any more as far as data mining results are understandable to end users[2] and the re-application-process should be much faster. While the proper design of a case will take up to several months, the pure re-application should take hours or days, depending on the amount of data and the efficiency of the underlying database server.

2.2 The Role of Metadata

Figure 1 shows, who delivers what kind of knowledge.

- Business data metadata: Information about business data is often available as data objects and their attributes, e.g. in data warehouses or metadata repositories. It can be automatically collected by the system. Otherwise, a database manager, who is specialized in describing business data, has to manually add metadata.
- Operator metadata: Knowledge about machine learning tools only exist as individual tool expert knowledge or tool specifications. This knowledge cannot be transferred automatically into the system. It has to be described as metadata by tool experts. Within the Mining Mart project, metadata about a set of the most important tools will be developed. Later on, tool experts have to use a defined interface to bring new tools and their metadata into the system.
- Business case metadata: Information about the KDD-process solving a specific business task only exists as individual knowledge of data mining experts being nowadays data mining consultants. To get this knowledge as metadata, experts have to specify new business cases.

[2] In Mining Mart we focus on preprocessing and not on handling mining tools and results. For post-processing and presentation see [3].

Who uses the metadata:

- Data mining experts implicitly use metadata of existing cases and of operators by getting system support within the specification phase of new cases.
- End users use metadata when identifying an existing case as THE proper case to solve their specific business task or to re-apply an existing case on changed data contents.
- Only the system uses metadata explicitly. By checking the applicability of a preprocessing operator, it supports the data mining expert. By compiling all preprocessing operator specifications into executable code whenever the chain is applied it enhances reusability on changed data contents.

3 Data Mining Specific Metadata

Within a KDD-process several data preprocessing steps are necessary until the actual data mining tool can be applied. In the Mining Mart project we have defined these steps as a preprocessing chain (Figure 3) that is a directed acyclic graph (DAG). This chain consists of several operators, which work on data and produce several intermediate results and the final result. Every operator transforms data for a special purpose, for example improve data quality by replacing missing values or reduce data amount by selecting a special data segment. Data mining experts need 1) knowledge about the data so that they can formulate the transformation purpose, 2) knowledge about the operator so that they can decide, which operator is applicable to reach the identified purpose, and 3) support if actual data does not fit the input requirements of an operator. Then, an intermediate operator must transform data into the proper input format for the next operator.

Operator Metadata about functional applicability and input requirements: All available operators within the system are grouped into classes with different functionality. In Mining Mart, we have identified the following classes: "Feature Construction", "Multi Relational Feature Construction", "Feature Selection", "Row Selection" and "Time Operator". Based on this classification the search for a proper operator is triggered by the intended data transformation purpose. A preprocessing operator can be seen as a transformation function with input and output parameters. Therefore, business data to be transformed are also parameters.

Data Metadata about operator applicability and data contents: Within the Mining Mart system, business data is stored in a database. Necessary metadata about these tables (basic business data, coming from e.g. a data warehouse) or views (results of preprocessing operators) are name and datatype of all attributes. The system uses this information for checking operator applicability. To support data mining experts in identifying the next data transformation purpose, statistic information for all data tables/views exist. It contains the number of data rows, number of different or missing data values and distribution information about every data attribute.

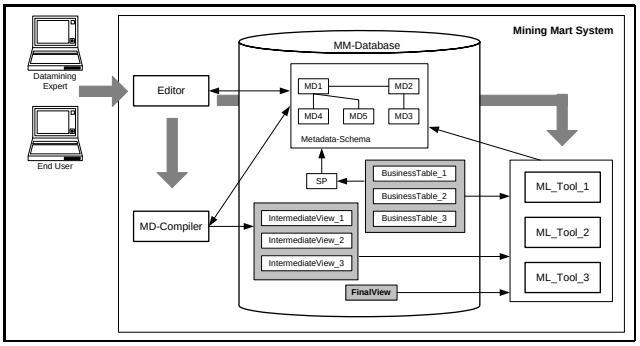

Fig. 2. Mining Mart Architecture

Business Case Metadata about a business case and the DAG of preprocessing operators: It conceals data mining techniques and domain knowledge of business data from end users. Based on this information end users are able to identify a case as THE proper mining solution for their business task. In Mining Mart, we supply a textual/graphical description that will be defined by data mining experts during the case specification phase. Information about the DAG enables system automation for re-application of a business case.

4 Mining Mart System

4.1 Mining Mart Architecture

Figure 2 shows the system components and how they use or create metadata.

Editor The graphical user interface for both data mining experts and end users is the editor. The system offers different functionality for these user types:

First, a data mining expert will use the editor for developing a new business case and iteratively defining the proper preprocessing chain. Only the "correct" chain will be stored as metadata. During the specification phase it may happen that the data mining expert defines a preprocessing operator but after execution the operator must be deleted again because of a dissatisfying result (Figure 3). Therefore, metadata and intermediate results must be easily editable. When deleting an operator step, all succeeding related operators and intermediate results will be automatically deleted.

Second, an end user will use the editor for visualizing a defined preprocessing chain. He cannot make changes within the chain. He is only able to re-apply the complete chain and look at chain descriptions and the final result. The system checks case applicability and marks all steps having invalidated conditions that the data mining expert can take over the case again and redesign the invalid parts.

A preprocessing chain has different states that are defined by data mining experts and used by the system:

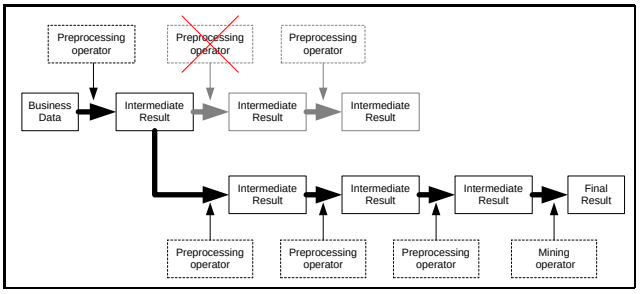

Fig. 3. Specification process of a Preprocessing Chain

During the specification phase, a chain has the state *DesignAndTest*. That means, the chain is not completely designed yet. Every operator can be executed separately. If the operator is of type machine learning, the editor starts the ML-Tool. Then the business data of this operator will mostly be a data sample.

When the specification phase is terminated, the data mining expert sets the state to *Application*. That means, the chain will be published in the case base. Therefore, every other user, end user or other data mining expert, will be able to use the chain for his purpose. Executing such a chain will apply all its steps without intermediate stop.

MetaData-Compiler The MD-Compiler will be started by the editor. It is a powerful tool to generate optimized SQL-code out of the stored operator metadata following the sequence of the DAG of a chain. For every operator that has as output an intermediate result, the compiler will generate a view within the database and store the metadata about this new view in the metadata-schema.

During *DesignAndTest*-state of a chain, SQL-code will be generated for all operators. When reaching the node of an operator of type machine learning a ML-Tool has to be applied. After the tool execution, parameters exist that are used for the appropriate manual operator in the next iteration of the design process. Only steps 1 to 3 of Figure 4 are applied. During *Application*-state, SQL-code will be generated for the complete operator chain. If an operator is of type machine learning, the re-execution of the ML-tool and afterwards the generation of the corresponding manual operator is necessary. All steps of Figure 4 illustrate this process.

ML-Tool A ML-Tool can be used as preprocessing operator and as mining tool. In this paper we will concentrate only on preprocessor operators. The tool must transform and store the learned results as metadata that the MD-Compiler is able to generate SQL-code for the corresponding manual operator.

4.2 Meta Model

The Meta Model[10] is an intermediate result of the Mining Mart project and currently enriched with all operator specifications by the Mining Mart project partners. It will be finished in June 2001 as public Deliverable 8 and 9.

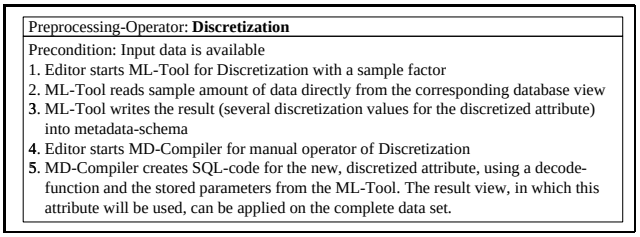

Fig. 4. ML-Tool Used as Preprocessing Operator

We have developed the Meta Model to efficiently store the different aspects of metadata. First metadata is grouped into a conceptual and implementational layer and second into a data and case layer. The user only works with the conceptual layer. The underlying implementational layer is almost completely used by the system. The separation in data and case layer results in a better understanding of the KDD-process. The Meta Model is the heart within the Mining Mart system to make it very flexible and scalable. Every system component uses it independently and could be easily exchanged by another component. A new component can be integrated by using the Meta Model, too.

Conceptual Case Layer In the conceptual case layer (right part of Figure 5), all operators within a preprocessing chain are stored. The class *step* defines the DAG of a chain. The functional purpose of operators is given by grouping them into different operator classes. The class *parameter* is the interface to the conceptual data layer specifying parameters of an operator. That means data types of operator parameters are implicitly defined. Because every operator has at least one output parameter, the *parameter* class also defines the result data-set of an operator. The relation to *executionElement* is the interface to the implementational case layer.

Conceptual Data Layer In the conceptual data layer (left part of Figure 5), business data is described as *concept*s with one or more *featureAttribute*s. For every feature attribute a data type is necessary. This specification allows later an automatic checking of operator input requirements by the system. An operator parameter can be of the type *concept*, *featureAttribute* or *relation* (interface to conceptual case layer). *Concept* and *featureAttribute* build also the interface to the implementational data layer. The interface to the implementational case layer is shown in the relation to *executionElement*.

Every concept specified by the user must be integrated into an ontology (see Figure 6). Three different ontology-levels exist: base level, database level and mining level. The base level reflects a general mining terminology and enables exchangeability of business cases between companies. Concepts of this type have no connection to the implementational layer. Concepts of the database level are represented within the database as tables, concepts of mining level as views. Both are connected to the implementational layer by a relation to the corresponding columnSets. Concepts of the database level need a connection to their superconcepts of type base level. Concepts of the type mining level are automatically

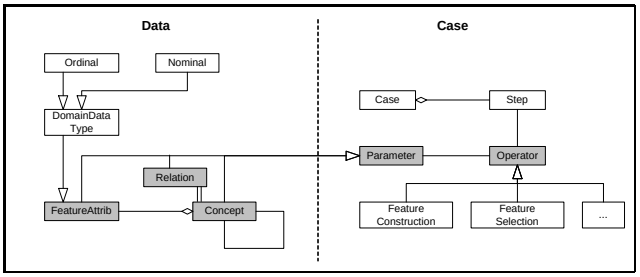

Fig. 5. Conceptual Layer

created by operators and are always sub-concepts of type database levels. When a chain of state *Application* is re-applied all concepts of type mining level will be re-generated by the MD-Compiler.

Implementational Data Layer In the implementational data layer (left part of Figure 7), metadata about business data is stored. Every *columnSet* has exactly one relation to class *concept* of type database level or mining level, every *column* to class *featureAttribute*. These classes are the interface to the conceptual data layer. For *columnSet* and *column* statistics are calculated.

Implementational Case Layer In the implementational case layer (right part of Figure 7), information about operator execution is stored. In this layer, the different operator types (manual or machine learning) are distinguished. The *executionElement* has a relation to *columnSet* and *operator* to build the interface to the implementational data layer and conceptual case layer.

5 From a Business Task to Mining-Relevant Business Data

In this example, we introduce a real business task from Swiss Life showing all necessary preprocessing steps and metadata that has to be defined by the data mining expert. We show possibilities how to re-use a completely developed preprocessing chain.

The business task is **"Potential Risk Groups for Customer Surrenders"**. In an insurance company, a customer buys an insurance but after several years he may sell the contract back to the company. Normally the reasons are not known. If the insurance company would have knowledge about potential risk groups it could better interact and maybe save customer losses.

Business Task

Identify target concept	Contracts
Identify target featureAttribute	hasSurrendered
Identify analysis period	Year 2000
Identify other necessary concepts	Person, Product, SalesRepresentative

As underlying datamodel for this business task, we consider all contracts that are valid at beginning of the analysis period. We take all relevant infor-

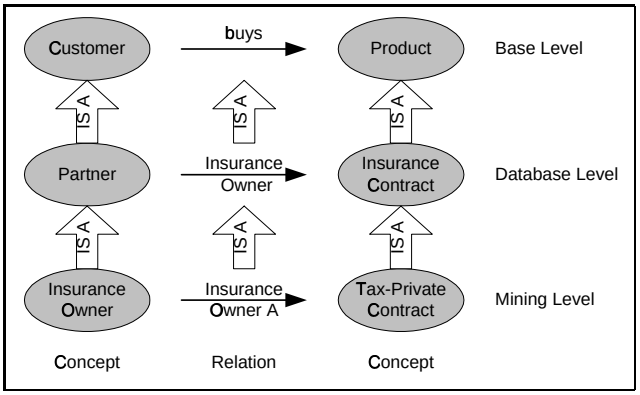

Fig. 6. Ontology of Concepts

mation about *contract*, *person*, *product* and *salesRepresentative* at that time. Next, we mark contracts with a surrender during analysis period. Then, we look at changes during the year by taking the same *contract*, *person*, *product* and *salesRepresentative* information at the end of the analysis period or at surrender date. Last, we can construct change-information by comparing the information at the beginning and end. Finally, we have a data basis with all relevant information describing possible reasons for customer surrender: marked contracts with actual surrender and customer-initiated changes that could be indicators for future surrenders.

Preprocessing Operators Following some operators of the business task are described as pseudo-code functions of preprocessing operators[3]:

1. Get all relevant contracts and their information at analysis begin
FeatureConstruction (I_C **Contract**, O_FA contractDuration(I_FA beginDate, I_FA endDate),
O_FA actualDuration(I_FA beginDate, I_FA sysDate), O_C Contract);
FeatureSelection (I_C Contract, O_FA contractDuration, O_FA actualDuration,
O_FA lifeInsuranceSum, O_FA premium, O_FA paymentType,
O_C ContractFeatures);
RowSelection (I_C ContractFeatures, O_C Contracts2000,
Constr (01012000 $\leq$ contractChangeDate $\leq$ 31122000));
2. Get all person information for these contracts
MultiRelFeatureConstr (I_C Contracts2000, I_R **Person**_Contract, O_FA age, O_FA gender,
O_FA placeOfResidence, O_C Contracts2000,
Constr (0101200 $\leq$ personChangeDate $\leq$ 31122000));
3. Mark surrendered contracts in analysis period
MultiRelFeatureConstr (I_C Contracts2000, I_R Contracts2000_Contract, O_FA isSurrendered,
O_FA surrenderDate, O_C Contracts2000,

[3] I_C = Input parameter of type Concept, I_FA = Input parameter of type Feature Attribute, I_R = Input parameter of type Relation, O_FA = Output parameter of type Feature Attribute, O_C = Output parameter of type Concept, Constr = Constraint

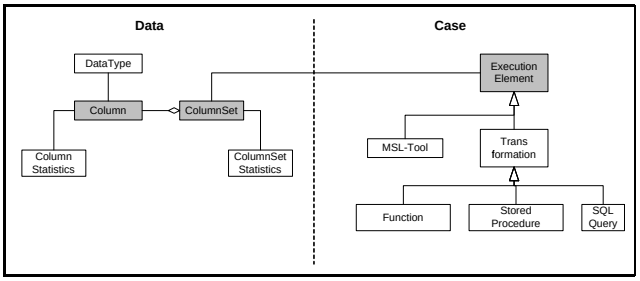

Fig. 7. Implementational Layer

Constr (01012000 $\leq$ surrenderDate $\leq$ 31122000));
4. Get contract information at end of analysis period or surrender date
MultiRelFeatureConstr (I_C Contracts2000, I_R Contracts2000_Contract,
O_FA endLifeInsuranceSum, O_FA endPaymentType, O_FA endPremium,
O_C Contracts2000, Constr (contractChangeDate $\leq$ 31122000));
5. Get change-information
FeatureConstruction (I_C Contracts2000, O_FA changeLifeInsuranceSum, O_FA changePremium,
O_FA changePaymentType, O_FA changePlaceOfResidence,
O_C Contracts2000
6. Discretize attribute lifeInsuranceSum
FeatureConstruction (I_C Contracts2000, O_FA lifeInsuranceSumDiscr,
O_C Contracts2000, Constr (10 groups));

The example shows that every preprocessing operator needs an input and an output concept. These parameters specify the data to be transformed and the result concept. Especially operator 1 is defined very precisely. The newly created feature attribute *contractDuration* is a function that calculates this attribute from the two input attributes *beginDate* and *endDate*. The other operators give an overview to complete the business task. After the preprocessing chain is specified, all mining relevant attributes are available. For the actual classification problem, a mining tool for subgroup discovery can be applied, e.g. MIDOS [11].

Just by storing this chain within the metadata-schema, an easy re-use and re-application is given.

The concepts *contract* and *person* (in the example implicitly used by the relation Person_Contract) are of type database level within the concept ontology (see Figure 6). When the data mining expert defines such concepts, he also has to map them to the base ontology level. If a chain is transferred to another company or data warehouse, all metadata from the conceptional case layer and only concepts of type base ontology level from the conceptual data layer have to be taken. To re-install it on the new place, a concept mapping from the base level to the new database level and to the implementational layer is necessary. All other metadata as well as the re-application itself is not affected by this transfer.

6 Related Work

The focus of Mining Mart is a user-friendly and efficient KDD-environment for data preprocessing that enables re-use and automation. To fulfill these goals, several aspects have to be considered:
Ensuring re-application of business cases means storing process steps as metadata. Within the data warehouse field several ETL-tools (Extract, Transform, Load) like Powermart (Informatica, http://www.informatica.com) or Datastage (Ardent, http://www.ardentsoftware.com) exist that use metadata for data transformation. But these tools can only perform general-purpose operators for data processing like filter, aggregation and join. Other transformations have to be programmed manually and are working on a low-level (data rows instead of data concepts). Operators specific for data mining preprocessing are not supported (neither manual operators nor those that use machine learning tools).

Applying preprocessing on large amounts of data requires execution within a database server. This approach takes the advantages of a database server for optimized handling of large amounts of data. As a result network traffic for data decreases. Many existing data mining environments generally support only internal preprocessing where data must be loaded into a KDDSE. That limits preprocessing on the available size of main memory.

Announcements from SPSS (SPSS, http://www.spss.com) Clementine-Server and Oracle Darwin (Oracle, http://otn.oracle.com/products/datamining) propose the execution of all preprocessing operators within a database server, too. In Mining Mart intermediate results consist only of views and not of tables. View specifications of the required data are stored whereas in tables the data itself is duplicated. That provides the user with a high flexibility during the specification phase of a preprocessing chain. Intermediate results can easily be deleted by just dropping the view specification. No physical data has to be deleted.

7 Conclusion

Making a preprocessing process time efficient and user-friendly requires a shared knowledge pool, where users can exchange and create KDD-experience in form of business cases. For this purpose metadata play THE important rule. We have described a Meta Model and metadata-driven system components that allow users to share their knowledge in a user-friendly way. The technical implementation should guarantee maximal performance by using a database server for handling large amounts of data. Furthermore, it should support the user in operator applicability. And last but not least, it should automate the re-use of an existing preprocessing chain and this results in huge time savings.

The heart of the system is the introduced Meta Model. It is the base for all described requirements like user-friendliness, applicability on large data bases and re-usability. Challenge of Mining Mart at this phase of the project will be the implementation of the different system components that are based on the Meta Model.

Acknowledgements Mining Mart is an European commission research project (IST-1999-11993) and is partly funded by of the Swiss federal office for education and science (project number BBW 99.0158). We thank all Mining Mart project partners for their ideas and work during specification of the Meta Model and system architecture, which is described in this paper. We also thank our colleagues for their helpful comments on drafts of this paper.

References

1. A. Bauer and H. Günzel. *Data Warehouse Systeme.* dpunkt.verlag, 2001.
2. S. Bell. *Entdeckung von Metadaten zur semantischen Anfrageoptimierung in relationalen Datenbanken.* Shaker Verlag Aachen, 1996.
3. Surajit Chaudhuri. Data mining and database systems: Where is the intersection? *Bulletin of the Technical Committee on Data Engineering*, 21.1:4–8, 1997.
4. J.-U. Kietz, R. Zücker, A. Fiammengo, and G. Beccari. Enabling end-user datawarehouse mining. In *Data Sets, Meta-data and Preprocessing Operators at Swiss Life and CSELT, Deliverable 6.2*, June 2000.
5. J.-U. Kietz, R. Zücker, and A. Vaduva. Mining Mart: Combining case-based-reasoning and multistrategy learning into a framework for reusing kdd-applications. In *Proc. of the Int. Conf. on Multi-Strategy Learning, MSL-2000*, 2000.
6. G. Koch and K. Loney. *ORACLE8: The Complete Reference.* Osborne McGraw-Hill, 1997.
7. K. Morik. The representation race - preprocessing for handling time phenomena. In *Proc. of the European Conference on Machine Learning, ECML-2000.* Springer Verlag, 2000.
8. D. Pyle. *Data Preparation for Data Mining.* Morgan Kaufmann Publishers, San Francisco, California, 1999.
9. T. Reinartz, R. Wirth, R. Clinton, T. Khabaza, J. Hejlesen, and P. Chapman. The current crisp-dm process modelling for data mining. In *Beitr. z. Treffen d. GI-Fachgruppe 1.1.3 Maschinelles Lernen (FGML-98). Techn. Report 98/11, Technical University Berlin*, 1998.
10. A. Vaduva, J.-U. Kietz, R. Zücker, and K.R. Dittrich. M4 - the mining mart metamodel. In *Technical Report 2001.02., Department of Information Technology, University of Zurich, ftp://ftp.ifi.unizh.ch/pub/techreports/TR-2001/ifi-2001.02.pdf*, 2001.
11. S. Wrobel. An algorithm for multi–relational discovery of subgroups. In J. Komorowski and J. Zytkow, editors, *Principles of Data Minig and Knowledge Discovery: First European Symposium (PKDD 97)*, pages 78–87, Berlin, New York, 1997. Springer.
12. R. Zücker and J.-U. Kietz. How to preprocess large databases. In *Proc of the Workshop on Data Mining, Decision Support, Meta-learning and ILP (DDMI2000), Lyon*, 2000.