Publishing Operational Models of Data Mining Case Studies

Timm Euler
University of Dortmund, Germany
Computer Science VIII
euler@ls8.cs.uni-dortmund.de

Abstract

This paper presents a method to publish executable models of data mining case studies in a so-called Case Base, where they can be inspected in detail by anyone using a common web browser. The Case Base serves to inspire and educate other developers of data mining applications, in particular if they are not yet experts in the field. A case study can be directly downloaded and executed on new data if it is found to provide a useful template. The approach is validated and exemplified using two data mining case studies from an Italian and a Polish telecommunications company. These case studies are interesting in their own right in that they involve complex data preparation issues. The paper studies these issues and relates them to the knowledge transfer opportunities that the Case Base offers.

1. Introduction

Software tools supporting common data mining or knowledge discovery tasks have matured greatly in the past decade, offering now basic graphical and conceptual support for all phases of the knowledge discovery process. This eases the daily work of data mining experts and allows a growing number of non-experts to try and start knowledge discovery projects. Though both experts and inexperienced users may find guidelines for their work in the CRISP-DM model [5], they are still faced with two essential problems, those of finding a suitable data representation and of choosing and tuning a learning algorithm to give acceptable results. Data preparation, as the subprocess that leads to the desired data representation, still consumes the largest part of the overall work, according to [11] and a 2003 KDnuggets poll, despite the existence of graphical, data flow oriented user interfaces for this task in modern software tools. The likely reason is that what is a good data representation depends on the mining task and data at hand, which poses a challenging problem, especially for inexperienced users. Such users would benefit greatly from sources of knowledge about how experts have solved past KDD (Knowledge Discovery in Databases) problems, especially from exemplary, executable KDD solutions. Even the experts might find inspirations in solutions from other business domains if these were available to them. The need for an environment to exchange and reuse KDD processes has long been recognised in the KDD community, see section 3.

This paper uses a framework in which successful KDD processes can be modelled, executed, and published to different KDD users, to present two data mining case studies. A web platform (the Case Base) to publicly display the models in a structured way, together with descriptions about their business domains, goals, methods and results, is described. The models are downloadable from the web platform and can be imported into the system which executes them (on a relational database).

The two case studies have been published in the Case Base, and are presented in detail in this paper. The descriptions here can be compared to the models in the Case Base. Issues that were identified as relevant by the authors of the two studies when evaluating the framework are discussed. In particular, the first study allowed a direct comparison of the effectiveness of graphical process modelling as compared to manual, low-level programming. Further, the second study indicated that some advantages of graphical modelling, as in the presented framework, can be disadvantageous when used by non-experts in a naïve way. The second study also allowed to compare the processing performance of a relational database with that of a SAS installation.

The paper is organised as follows: section 2 describes the MiningMart framework which provides the metamodel in which the models of the case studies are expressed. Section 3 gives related work. Sections 4

and 5 present the two case studies, each with some discussion of relevant problems and lessons learned. Finally, section 6 concludes the paper.

2. The MiningMart framework

MiningMart is an environment for the development of KDD applications that makes use of a formal metamodel (called M4) to model KDD processes. The central ideas of MiningMart have been published in [10]. They are summarised in this section and extended by a more detailed discussion of the Case Base and its technology, since this concerns publishing the case studies which are presented in later sections of this paper.

2.1. Overview

In MiningMart, both the data and the processing/mining operations on the data are modelled declaratively using M4, and translated to operational SQL by a compiler module. A data model and a process model together describe an instance of a KDD process and are called a *case*.

The metamodel M4 can be expressed in various ways, for example a relational database schema or an XML DTD. MiningMart currently uses a database to store the case models while working with them, but uses XML for import and export of the models. A database has the advantage that good support for the consistency of the case models is given, as the dependencies between model objects can be expressed in database constraints such as foreign key links.

In order to facilitate the reuse of cases, a data model in M4 consists of two levels. On the higher level, data is modeled by *concepts*, which contain *features*, and *relationships*. Every step of the KDD process is described, in terms of input and output, on this level. The lower level uses *tables* and *columns* to which the higher-level elements are mapped. It can model both database schemas and flat file data. This two-level approach allows to reuse the higher level elements on new data by simply changing the mapping. For the mapping, each concept corresponds to one table or view, a feature can correspond to one or more columns, and relationships correspond to foreign key links between tables.

The mapping between the levels is provided by the user, if the case is developed for the first time; in the MiningMart system, a graphical editor supports the creation and manipulation of higher level elements and their mapping to given data. However, if an existing case is reused, a simple schema-matching algorithm can be employed to find at least a partial mapping. The matcher algorithm is based on comparing the names and datatypes of the concepts and features (higher level) of the existing case, and the tables and columns (lower level) of the given data (compare [12]). Once the mapping is done, all user work on the KDD process continues using the higher data level. This provides a more abstract, task-oriented view of the KDD process than low-level programming would.

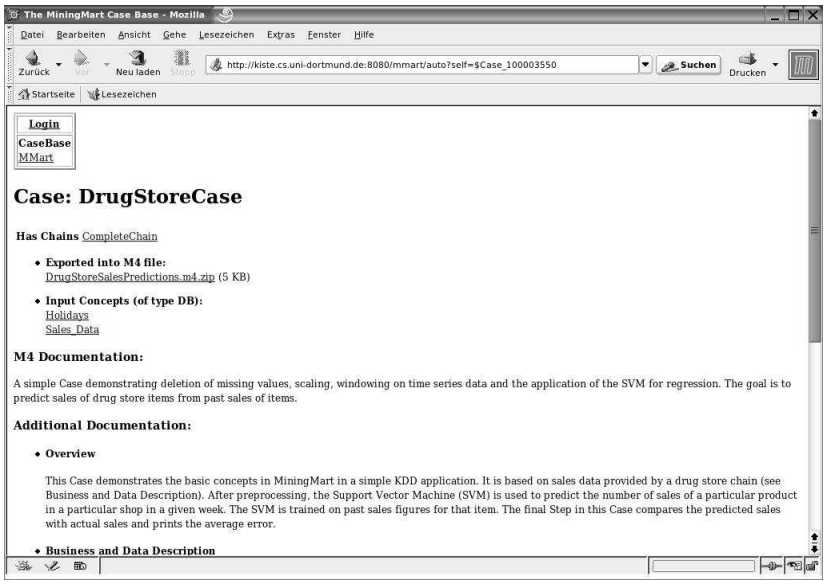
To model the data processing operations, the metamodel allows to define characteristics of basic processing operations by specifying *operators*. The definition of an operator in the metamodel includes its name, its input and output parameters and their types (concept, feature, or simple values like strings or numbers), and constraints on the parameters that must be fulfilled. A typical constraint might specify, for example, that a certain input feature must have a certain conceptual datatype. The actual processing behaviour of the operator is not specified in the metamodel but in the system that interprets it. This is the *compiler* functionality of the system. The output of an operator can be used as the input to another operator, so that the data flow induces a directed acyclic graph of operators.

To ensure that a wide range of KDD processes can be modeled, new operators can easily be added declaratively to M4 and will then automatically be available in the system; only the compiler has to be extended by a new module for each new operator (a Java API is available for this task).

2.2. The case base

This section describes the knowledge portal, called Case Base (`http://mmart.cs.uni-dortmund.de`), that serves to distribute successful KDD models (cases) publicly. The core of this portal is a software called InfoLayer [8] that translates structured information, according to a given ontology, to HTML files. It can also generate RDF files which can be read by software agents. In MiningMart, the ontology is the metamodel M4, and a collection of instances of this ontology forms the central repository of KDD cases. Only the higher level of the data model is published for confidentiality reasons. These higher-level parts are represented in UML, which is read by the InfoLayer software. The UML classes are linked to a database that contains the M4 schema. Whenever a web client requests information about an M4 object (via HTTP), the InfoLayer creates an HTML file for it, disregarding caching for this discussion (M4 objects are *operators*, *concepts* etc.). The HTML files are generated using

templates that provide the layout for the information to be displayed. There can be zero or one layout template for each type of M4 object. If no template is given, the contents of an HTML file for an M4 object are automatically determined by the InfoLayer software from the UML model. A template can be used to provide only parts of the default contents, or to arrange them in a particular way, for example by using HTML tables. By default, the M4 object is displayed with its name, its properties, and the names of M4 objects it is directly linked to. The linked M4 objects appear as HTML links so that a web user can browse through a case model according to the structure of M4. For instance, an operator is displayed together with its name and its parameters, and a click on any parameter shows the realisation of that parameter, which is in turn an M4 object, for example a concept used as an input parameter for the operator. The following is a screenshot showing the case base as it displays an example case.



When setting up a case with the MiningMart system, every object from the case itself to operators, parameters, concepts and features can be documented using free text. These comments serve users for their own orientation in complex models. They are stored in M4 and appear on the web pages when a case is published, so that other users browsing the case have a good orientation as to the purpose of each step in the KDD model and the use of their parameters. If such comments are missing, they can be added by the operators of the case base.

However, users who search for a case which they might use as an inspiration for their own KDD problem, or even as a blueprint of a solution, need some additional, more general information about each case. The most important types of information are (i) the business domain, (ii) the business problem that was attempted to solve, (iii) the kind of data that was used, (iv) the mining task and other KDD methods that were employed, and (v) the results, both in terms of KDD and the original business problem. Hence, exactly this information is provided together with every case that is presented in the case base. To this end there is a template with five slots for free text, corresponding to the five types of information above, which is to be filled by every case publisher (a sixth slot with contact information enables further inquiries by interested users). The filled template is displayed in the case base as the first page of information about each case. From there users who are motivated by the descriptions can start to browse the case model, beginning with the chains of operators or the input data. In this way, the case model is related to the context in which it was set up, which allows to judge its suitability for a new business problem. Finally, each case model is linked to a file that can be imported into a MiningMart client.

2.3. Case retrieval

This section briefly discusses a few ideas for case retrieval, that is, how to find a MiningMart case that can serve as a template for an own solution from the case base. A suitable starting point is the additional documentation published in the case base for every case. Assuming a low number of published cases, this information can be searched manually, but as the case base grows, automatic search methods should be added to allow at least keyword search. Another useful way of approaching the case base can be offered by sorting the cases according to various topics extracted from the additional case documentation. The five slots of the documentation template provide five useful topics for indexing the case base. Further topics (such as type of business/institution where the application was realised) can be added by extracting this information from the free text descriptions in the slot.

The business-related information will often not be enough to determine whether a published solution is suitable for adaptation to own data sets. A second

method of approaching the case base is by looking for data models in it, called *target models* hereafter, that are similar to the own (local) data sets. The automatic schema matcher included in MiningMart can be used for this. It searches among all data models in the case base for models similar to the local data.

This online method has an important advantage. All cases use a particular data model as input, then preparation operations are applied to the data. Each preparation operation produces intermediate data models. These intermediate models can be included into the search for target models, so that the most suitable *entry point* into a case can be found. Since preparation is actually a method to adapt data representations, it would make no sense to restrict the search for target data models to the initial data that the original KDD process started out on. Schema matching is a useful tool in this setting as the number of target data models is high, making manual search for the best entry point a cumbersome task.

A unique option that the case base offers is to search it for common subtasks that have been solved using identical processing structures. A simple subgraph detection algorithm can be used for this (since the nodes of the graphs are typed, efficient algorithms exist). More cases are needed before this will lead to interesting results, however.

3. Related work

MiningMart was mainly described in [10]; see also related work cited there. The technology of the case base was updated recently; this and case retrieval issues are a contribution of this paper. The idea of collecting and publishing KDD solutions was mentioned (though not realised) early in [15] and [9]. The importance of the reusability of KDD models is also stressed in [16] and [2].

To document and store KDD processes requires a modeling language, or metamodel. A well-known but informal standard to model the KDD process is Crisp-Dm [5]. The new PMML version 3.0, a standard to describe machine-learned models in XML [13], includes facilities to model the data set and data transformations executed on it before mining. However, it is not process-oriented, thus it does not allow to model a data flow through a complex KDD process, and the data model is restricted to one table. Other standards around data mining are Java Data Mining and SQL/MM Data Mining. Though extensible, they currently provide interfaces to modeling algorithms rather than to complete KDD processes. Similarly, in [3] a data mining ontology is presented to enable grid-based services, but is currently restricted to the modeling phase of the KDD process.

Recently, some new research attempts to employ grid infrastructures for knowledge discovery; a good overview is given in [4]. To enable the execution of KDD processes on a grid, these processes have to be modeled independently from the machines that execute them, and heterogenous data schemas and sources have to be modeled. In [1], a Discovery Process Markup Language (DPML) is used, based on XML, to model the complete KDD process. Unfortunately, from the available publications it is not clear how comprehensive and detailed DPML is.

4. Case study 1: Churn prediction

This section describes a data mining application that was developed in an Italian telecommunications institute. An overview of it was given in [7] and [14]; the present paper adds important details as regards the data preparation and the lessons learned.

A major concern in customer relationship management in telecommunications companies is the ease with which customers can move to a competitor, a process called “churning”. Churning is a costly process for the company, as it is much cheaper to retain a customer than to acquire a new one [14]. Churn prediction is the task of predicting which types of customers are likely to churn, and more challenging, when they will churn. These business problems can be translated to data mining or KDD problems in various ways. One successful translation to a classification task that predicts a class of customers likely to churn within a given month in the near future is described in this paper. The task was solved using decision trees which achieved a predictive accuracy of 82%. This good result was only possible due to the introduction of relevant derived features for prediction which were not available in the original data, and due to a re-representation of the data so that temporal aspects could be included. Thus data preprocessing was a key success factor in this application.

One interesting aspect of this case study is that it was implemented twice, based on manual programming on the one hand, and on graphical modelling on the other. This allowed to compare the amounts of work spent by highly paid KDD experts on the application in both scenarios (see section 4.6).

4.1. Overview

As said above, the objectives of the application to be presented here were to find out which types of

customers of a telecommunications company are likely to churn, and when. To this end, the available data tables were transformed so that a classification algorithm could be applied. In the resulting data set, each row (that is, each example for classification) corresponded to one customer of the company, and contained many features describing their telecommunication behaviour for each of five consecutive months. Whether or not the customer left the company in the sixth month determined the classification label or target. Thus a binary classification problem was formed that could directly be addressed using several classification algorithms. Once a learned classifier is available it can be applied every month on data from the current and past four months, to predict churn for the following month. A longer prediction horizon (to predict churn not for the following month but, say, the second or third month) can be realised easily by changing a few parameters in the graphical model of the application.

4.2. The data

The available data sets were: (i) call detail records, recording for each phone call a customer made the time and date, called number, tariff, type of call etc.; (ii) billing data from the accounts department, containing revenues generated by each customer in a given month; (iii) and customer services data from the customer registry, containing the gender and address of a customer as well as the dates of entering and leaving their contract with the company. Those customers still with the company serve as negative examples for churn, while for those who have left the company, the data from the last five months they stayed with the company is used to form positive examples.

4.3. Data preparation

The first table to be prepared is the call detail records table. The transformation of the original data starts by extracting an Id for each month from the date of each call, because monthwise statistics are needed. This month Id has to be the same as the one used in the billing data. A new column with the month Id is added to the call detail records. Additionally, the type of phonecall (internet provider, local, distance, abroad, mobile phone etc.) is derived from the number called, with the aim of creating a telecommunication profile for each customer.

Next, the time span to be used for mining (the five consecutive months) must be selected from the complete table. Those customers who happened to have left the company at the end of the time span are positive examples for churning. However, selecting only one particular time span does not deliver a sufficient number of positive examples. Also it might introduce a learning bias. For these reasons, six different spans were selected. Notice that the six resulting data sets are likely to contain overlapping customer sets, since many customers (who have not left the company) participate in all time spans.

Now the six subsets must be mapped to the same time index (e.g. 1 to 5 for the five months), so that the six time spans can be matched. After creating the common time index, the six data sets are further processed in exactly the same way. Rather than setting up the same transformation process six times, one might set it up once and use it on six different inputs. However, the overall mining process should be executable automatically every month to predict new groups of churners. Not every KDD system supports automatic execution of a modelled process on different input tables. In this application, a different approach was taken that simplifies the automatic execution by exploiting a *Segmentation* operator available in MiningMart. One additional, nominal column is added to each of the six data sets that contains only one value which is different for each data set. Then the data sets are unified (using a union operation like in SQL). Now the segmentation operator takes the additional column as the segmentation index, and ensures that all following operators are applied to each segment in parallel. This means that the process can be described hereafter as if it was applied to only one input table, though there are six segments to be processed. This input table now contains one row per phonecall made, and four columns: the customer Id; the month Id; the calllength; and the type of call. Using aggregation as in SQL (forming a data cube), the sum of calllengths spent by every customer in each month for each type of phonecall can be computed. The resulting table contains the data that is needed for mining; however, the format is not suitable yet: it is not a table with a single row for every customer, but with 35 rows per customer: the number of months, five, times the number of different call types, seven. What is needed now are 35 new attributes: one per month and per call type. Each new attribute will contain the calllengths spent by every customer in that month making that type of phonecall.

These attributes can be created using 35 derivation operations. However, exploiting the special MiningMart operators *Segmentation* and *Pivotisation*, the process becomes much simpler. Segmentation is applied a second time, this time using the call types as

the segmenting attribute. Now again the further process can be described and set up as if there was only one input table, although in reality there are 42 tables with the same data format: seven, the number of call types, times six, the number of time spans.

At this point the operator *pivotisation* can be used to gain the final representation of this part of the data in one single step. Conceptually, pivotisation creates a table with six columns, one per month plus one customer Id, so that each row corresponds to exactly one customer. Behind this conceptual view are 42 data tables, six time spans for each of the seven call types. In the next step, the union of all data tables corresponding to the same call type can be formed, leaving seven tables. Finally, the seven tables (each with five non-key attributes) are joined, resulting in one real data table with the customer Id column and 35 telecommunication profile columns, where each row contains all of the information for exactly one customer.

All of the above concerned only one of the three original data tables, the call detail records table. The second table contains the revenues per month and per customer. This table is transformed in a similar way, using selection of the six time spans and one pivotisation so that the resulting table contains one column per each of the five months, indicating the revenue generated by every customer during that month. The third table with the individual customer information contributes the target attribute: those customers who left the company in one of the six end months of the six time spans are positive examples for churning, all others are negative examples. All three preprocessing results can then be joined to form a final table with 41 columns to be used for prediction, plus the target and the key column.

4.4. Data mining

By following the rather complex preparation process above, it was possible to transform the data from a "transactional" format, containing information for every single customer transaction (phonecall), to an aggregated format from which the time-related information was still available, but which provided each customer as a single learning example.

On this table a decision tree was trained to predict the binary target. However, the first results were not satisfactory. A possible reason was presumed to be the fact that the five months that form a time span were not related to each other from the view of the learning algorithm. It was felt that *changes* in telecommunication behaviour might be a good indicator for churning, but that these changes could obviously not be found by the decision tree. Therefore additional columns were derived. As an indicator of change, the differences in the calllengths between consecutive months were tested as well as the slope of a line connecting the first and fifth month's calllengths on a graph (in this case, the sum of calllengths of all call types). This latter indicator in particular helped to increase the predictive accuracy to a satisfactory level. Note that it was only possible to use this indicator based on the complex data preprocessing phase described above.

Another factor that increased the predictive accuracy was the differentiation of customer groups according to the overall revenue that the company generated from them. More precisely, four different decision trees were trained on four groups of customers ranging over low, medium, high and very high profitability, where profitability was indicated by the sum of revenues in the five months considered. This turned out to be a successful differentiation, in that the average predictive accuracy of the four trees was 2% higher than that of one global tree trained on all customers.

4.5. The published case study

The reader is invited to compare the above descriptions to the browsable model of the case study, which is available in the MiningMart Case Base (URL see section 2.2) under "Model Case Telecom".

4.6. Lessons learned

An interesting aspect of the case study above is that the application was implemented twice, once manually in SQL and once using MiningMart. Thus it was possible to quantify the amount of work saved by using a high-level modelling software with a GUI, compared to low-level programming. While programming the application required 12 person days, modelling it graphically could be achieved in 2 person days of work. Especially the availability of rather advanced preprocessing opertors for segmentation and pivotisation eased the task greatly in the graphical approach. Further advantages of graphical modelling that were attested are simplified documentation, especially for in-house education, simplified changing and testing of parameters (for example to change the prediction horizon, by no means a trivial change given the complex preprocessing phase), and versioning of the developing process model.

It was also confirmed in experiments that the overhead caused by parsing and translating the declarative model is negligible for real-world data sets (in this application, two million records were processed). Since MiningMart translates the process model to SQL, this approach scales as far as the underlying database scales. However, an interesting counterpart to this situation is encountered in the second case study, compare section 5.3.

This data mining application led the company that commissioned it to execute a trial campaign on a sample of the customers, to reduce churn. The results justified the investment in the project. The application was therefore integrated into other measures for CRM in the front-back office automation of that company.

5. Case study 2: Targeting a marketing campaign

This section describes a marketing application in the Polish telecommunications institute NIT. A technical report about it is available [6].

5.1. Overview

The task that was solved in the application was customer profiling, for the purpose of adapting a marketing strategy to introduce a new voice mail product. Three sources of data were available: call detail records, listing information about each phone call each customer has made; contract data, listing information about the type of contract each customer has signed with the company; and data from a call center that contacted a number of customers initially to see if they would buy the new product.

From this data, after a long process involving dozens of atomic data transformations, the input to the mining algorithm was constructed. Here only the key points of the process are described.

The biggest part of the data preparation was consumed by processing the call detail records, rather like in the first case study (section 4). This table contains the start time, date, length in minutes, number of tariff units, the number called and some other information about each phonecall of each customer. This large amount of detailed data had to be aggregated to meaningful single statistics for each customer. This was done in a similar fashion as in the other case study. However, the first attempt to do so involved segmenting the data such that each customer corresponded to a single segment. Conceptually, in the graphical model that MiningMart provides, this is a neat way of setting up a short processing chain to solve the given problem. Technically, however, this means to create as many SQL views on the original table as there are customers stored in it. This approach does not scale so well to larger amounts of data. In particular, the overhead caused by compiling the declarative process model into SQL, which was found to be negligible in the first case study (section 4.6), was very high in this study under this approach, due to the high number of views that had to be created and, in progress, evaluated. Changing the conceptual setting such that first some aggregation tasks were performed, and only then the segmentation took place, was therefore beneficial.

The customer profiles built in this way were used to predict the customers' response to the new product, based on the call center data. This data provided the target attribute for prediction: whether the customers responded positively or negatively to the product offer. Since only a small sample of customers could be contacted by the call center, mining was used to generalise from the sample to the whole group of customers, in order to save marketing costs. Detailed results are unfortunately kept confidential.

5.2. The published case study

The reader is invited to compare the above descriptions to the browsable model of the case study, which is available in the MiningMart Case Base (URL see section 2.2) under "Call Center Case – NIT".

5.3. Lessons learned

Two interesting aspects of this case study were identified. The first one is described above, and concerns the scalability problem encountered using the naïve segmentation approach. It shows that although many tools provide rather high-quality, high-level support for data processing, still one needs experts who know the underlying procedures well enough to develop efficient models. This emphasises the need for a public repository of successful KDD solutions, such as the Case Base presented in section 2, to provide templates or blueprints that help new developers of KDD applications to avoid traps that others have already discovered. As powerful KDD tools are becoming increasingly available and easy to use, knowledge about good KDD solutions must not stay behind, but needs an efficient means of distribution such as the public Case Base.

The second interesting aspect of this study is that it was implemented also both in MiningMart and in another system, namely SAS. Since MiningMart

translates its models to SQL, this allowed to compare the data processing performance of the underlying relational database (Oracle) with that of SAS. Sound claims about the relative performances of these two environments cannot be made because they were installed on different hardware. However, data processing was about two times faster in the SAS system, which is not surprising given that relational database management systems must perform overhead tasks such as consistency and rollback management. On the other hand, SAS required manual programming of many tasks that could be realised easily in the GUI using MiningMart. Further, having intermediate processing results available in the database allows simpler examination of such results by querying and searching. These contrasting issues have to be prioritised based on the intended application when choosing a suitable data mining workbench.

6. Conclusions

This paper has presented two case studies in data mining from the area of telecommunications. The focus was on data preparation, where usually the bulk of efforts in a mining project is spent. In both studies it was possible to reduce these efforts greatly using a powerful, graphical preprocessing environment. However, the second study showed that such environments do not render experts in the field superfluous, but that detailed knowledge of underlying processes is necessary to develop a successful application.

This, in turn, motivates the introduction of a declarative metamodel for modelling such successful applications; using the metamodel, the (annotated) application models can be published to be inspected in detail by anyone. This serves to distribute knowledge about successful case studies to less experienced users, and helps them to avoid hidden traps. Of course, also unsuccessful applications can be documented in the Case Base.

The two studies also show that a choice of the most suitable environment to conduct an application in is dependent on several criteria, which may be prioritised differently in different applications. Scalability of the underlying processing software is important, but the possibility to use graphical modelling in a suitable user interface can help to save developing time (compared to low-level programming), and developing time is usually more expensive than computing time.

7. References

[1] S. AlSairafi, F. Emmanouil, M. Ghanem, N. Giannadakis, Y. Guo, D. Kalaitzopoulos, M. Osmond, A. Rowe, J. Syed, and P. Wendel, "The Design of Discovery Net: Towards Open Grid Services for Knowledge Discovery", *High-Performance Computing Applications*, 17(3), pp. 297-315, 2003.

[2] A. Bernstein, S. Hill, and F. Provost, "Toward Intelligent Assistance for a Data Mining Process: An Ontology-Based Approach for Cost-Sensitive Classification", *IEEE Transactions on Knowledge and Data Engineering*, 17(4), pp. 503-518, 2005.

[3] M. Cannataro and C. Comito, "A Data Mining Ontology for Grid Programming", *1st Workshop on Semantics in Peer-to-Peer and Grid Computing at the 12th International World Wide Web Conference*, 2003.

[4] M. Cannataro, A. Congiusta, C. Mastroianni, A. Pugliese, T. Domenico, and P. Trunfio, "Grid-Based Data Mining and Knowledge Discovery", in N. Zhong and J. Liu (eds.), *Intelligent Technologies for Information Analysis*, Springer, 2004.

[5] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth, "CRISP-DM 1.0", Technical Report, The CRISP-DM Consortium, 2000.

[6] C. Chudzian, J. Granat, and W. Tracyk, "Call Center Case", Deliverable D17.2b, IST Project MiningMart, IST-11993, 2003.

[7] T. Euler, "Churn Prediction in Telecommunications Using MiningMart", *Proceedings of the Workshop on Data Mining and Business (DMBiz) at the 9th European Conference on Principles and Practice in Knowledge Discovery in Databases (PKDD)*, 2005.

[8] S. Haustein and J. Pleumann, "Easing Participation in the Semantic Web", *Proceedings of the International Workshop on the Semantic Web at WWW2002*, 2002.

[9] R. Kerber, H. Beck, T. Anand, and B. Smart, "Active Templates: Comprehensive Support for the Knowledge Discovery Process", in R. Agrawal, P. Stolorz, and G. Piatetsky-Shapiro (eds.), *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, 1998.

[10] K. Morik and M. Scholz, "The MiningMart Approach to Knowledge Discovery in Databases", in N. Zhong and J. Liu (eds.), *Intelligent Technologies for Information Analysis*, Springer, 2004.

[11] D. Pyle, *Data Preparation for Data Mining*, Morgan Kaufmann Publishers, 1999.

[12] E.Rahm and P. Bernstein, “A Survey of Approaches to Automatic Schema Matching”, *The VLDB Journal*, 10, pp. 334-350, 2001.

[13] S.Raspl , “PMML Version 3.0 – Overview and Status”, in R. Grossman (ed.), *Proceedings of the Workshop on Data Mining Standards, Services and Platforms at the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2004.

[14] M. Richeldi and A. Perrucci, “Churn Analysis Case Study”, Deliverable D17.2, IST Project MiningMart, IST-11993, 2002.

[15] R. Wirth, C. Shearer, U. Grimmer, T. Reinartz, J. Schlösser, C. Breitner, R. Engels, and G. Lindner, “Towards Process-Oriented Tool Support for KDD“, Proceedings of the 1st European Symposium on Principles of Data Mining and Knowledge Discovery, 1997.

[16] N. Zhong, C. Liu, and S. Ohsuga, “Dynamically Organizing KDD Processes”, *International Journal of Pattern Recognition and Artificial Intelligence*, 15(3), pp.451-473, 2001.