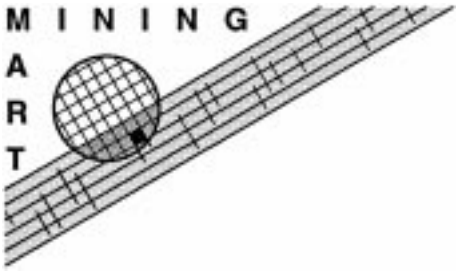

Enabling End-User Datawarehouse Mining
Contract No. IST-1999-11993
Deliverable No. D1.1

Specification of Pre-Processing Operators Requirements

Lorenza Saitta[1], Jörg-Uwe Kietz[2], and Giuseppe Beccari[3]

[1] University of Piemonte Orientale
Dipartimento di Scienze e Tecnologie Avanzate
C.so Borsalino 54, 15100 - Alessandria, Italy
saitta@di.unito.it
http://www.mfn.unipmn.it

[2] Rentenanstalt / Swiss Life
Corporate Center, IT Research & Development
General Guisan Qui 40
8002 Zürich, Switzerland
Uwe.Kietz@swisslife.com
http://www.swisslife.com

[3] CSELT, Via Reiss Romoli
10100-Torino, Italy
beccari@cselt.it
http://www.cselt.it

[4] With contribution by:
Marco Botta, Attilio Giordana, Anna Fiammengo,
Regina Zücker, Katharina Morik and Harald Liedtke.

June 24, 2000

Abstract

In this workpackage a specification of the pre-processing operations needed in order to analyse the available data warehouses is described. The collection of pre-processing requirements will serve as the unifying and organizing basis for all other workpackages, which shall provide methods for satisfying each of them. Based on partners' previous experience, those aspects of data selection and preparation that have proved to be most critical to the success in previous data mining activities have been identified, as well as the application context in which they arose.

Chapter 1

Introduction

Two technologies are emerging to extract knowledge from the mass of data nowadays available:

1. Data warehousing and On-Line Analytical Processing (OLAP) of data for the verification of hypotheses, and

2. Knowledge Discovery in Databases (KDD) for the discovering of new hypotheses.

The two ways of mining data, OLAP and KDD, are complementary: OLAP allows a dimensional view of the data, as opposed to the relational one provided by KDD. OLAP pre-computes and stores column totals, to be shown to the user in the characteristic "cube", a three-dimensional structure reporting counts for combinations of values of the three variables on the axes. The OLAP mode of exploration allows hypotheses formulated by the analyst to be verified; hence it is substantially a manual process. KDD, on the contrary, does not rely on human intuition, even though, at least in principle, previous knowledge can be of great help.

Practical experience with these techniques have proven their value. However, it is also apparent that using a datawarehouse for decision support or applying tools for knowledge discovery are difficult and time-consuming tasks. Therefore, it is currently still quite difficult to get an admissible return of investment from using them. If we inspect real-world applications of knowledge discovery, we realize that 50 - 80% of the efforts are spent on finding an appropriate transformation or sampling of the data, and on specifying the proper target of data mining; in other words, they are spent on pre-processing the data.

Actually, the mining results critically depend on the data. Extensive documentation on the importance of non-algorithmic issues in real-world applications are reported, among others, by Langley and Simon [1995], Rudenstram [1995], and Saitta and Neri [1998]. As a result, these technologies are mostly used by a few highly skilled users.

	Time	Imp.
Business understanding	20%	80%
a) Exploring the problem	10%	15%
b) Exploring the solution	9%	14%
c) Implementation specification	1%	51%
Data preparation & mining	80%	20%
a) Data preparation	60%	15%
b) Data surveying	15%	3%
c) Modeling (data mining)	5%	2%

Table 1.1: Steps of a KDD-project with time to complete and importance to success.

One of the most time consuming steps for KDD consists in preparing the source data. In real-world applications we have to deal with very heterogeneous data of doubtful quality. On the other hand, there is no universal data mining tool available, which is equally suited to handle the variety of analysis tasks promising enhanced insight about a company's position in the market, its customers, its products, and so far. As a consequence many different types of data mining algorithms have to be employed, which typically have strict and specialized input requirements. The problem becomes even more difficult if we consider all the possibilities to transform the data into the best form for mining. It is very likely that a good pre-processing of data has an even stronger influence on the quality of the results than the selection of the mining tool.

In Table 1.1 an overview of a typical timing for a KDD process is reported, while the actual data mining step is well understood, and efficient tools for it exist, clever pre-processing of the data is still a matter of trial and error, although it is known since long that data representation strongly influences the quality and utility of the analysis results [Amarel, 1986]. Since end-users cannot usually solve the difficult task of data reformulation, in order to get their desired answer, they might exploit others' experience in similar tasks. A case-base of previously solved discovery tasks or of actions to avoid shall serve as blueprints for further similar queries to similar data. The case-base thus offers a user-friendly interface to the best practise of knowledge discovery from very large and heterogenous data sets. In this framework the highly skilled data mining user is still needed, but only to create new cases, not to redo the same cases all the time. This task is delegated to the end-user, who retrieves the prepared cases, making to it some simple adaptions.

The pre-processing cases are not stored extensionally, as most common in todays KDD Support Environments (KDDSE), but intensionally as transformation specifications in the form of meta-data. This solution ensures that

the cases could be edited easily, reapplied, and - most importantly - could be compiled into a form executable by a data base system to perform the needed data transformation. In the present deliverable we are specifically concerned with the definition of a set of pre-processing operators, which will serve as a basis for partially automating the data preparation process and for defining part of the content of the case base. The definition and use of pre-processing operators shall be embedded in the KDD support environment (Brachman & Anand [1996]), described in Deliverable D2. The pre-processing environment that we consider introduces support for in-database pre-processing operations, well suited for multi-relational databases, as well.

Particularly important for the operator definition are the meta-data describing data formats, operators input and output, their parameters and actions. Meta-data are described in deliverable D6.2. Meta-data help the user in identifying his/her particular application type, and provide guidance to the user throughout the knowledge discovery task, by suggesting the actions and steps that are most likely going to be successful.

The application of pre-processing operators can be automatized to some extent; one way is by storing in the case-base entire chains of pre-processing and analysis steps for later re-use (for example, a case of pre-processing for mailing-actions, or a case of pre-processing for monthly business reports); another one is by parametrizing operators in such a way that the user has only to take high level decisions, without bothering with implementation details (for example, discretization of continuous attributes according to either equal interval width or equal cardinality).

Pre-processing operators can be applied manually, exploiting also the available background knowledge; however, machine learning techniques can well be seen as pre-processing operators that summarise, discretise, and enhance data. Making data transformations available includes the development of an SQL query generator for given data transformations, and the execution of SQL queries for querying the database. The analysis of the available data and meta-data allowed us to gather knowledge about constraints on pre-processing operations. This knowledge has been used to define a set of operators, to be supported in the project, as well as their actions, constraints and conditions of applicability. The role of domain knowledge is described in deliverable D5.

The variety in the nature of the considered applications, and the analysis performed on a large body of literature addressing the issue ensure that a broad range of applications and domains will be covered. Also the possibilities of combining the operators in pre-processing, on a syntactical-level, as been identified.

Chapter 2

Pre-Processing Operators

Pre-processing operations are necessary to prepare the raw data material properly in such a way that they have the right format for a data mining tool.

Operators can be parameterized to cope with different situations in a (semi-)automated and uniform way. In some cases, setting of parameter values leads to a complete specification of an operator, so that it can be directly applied to the data.

However, we also consider the case that a mining tool itself can be used as a pre-processing operation; here, setting a parameter equals invoking a learning procedure to find out how best performing the operation in the actual context.

The KDD process, and the pre-processing phase within it, is not a linear one: most often, it involves more passes through operator applications, in a series of trial-and-error cycles. Even though a clear definition of operator syntax, semantics and pragmatics should substantially reduce the number of these cycles, at least in the pre-processing phase, the basically iterative nature of the whole process is likely to persist.

In order to guide the pre-processing operations of data transformation, three kinds of description for operators are necessary:

Syntactic descriptions related to the formal language are specific to the particular operator;

Semantic descriptions related to the meaning of description components;

Pragmatic aspects related to condition of execution on the data warehouse (database) or KEPLER.

In order to identify the pre-processing operators, we want to provide the KDDSE with a number of typical mining cases, supplied by the partners, have been analysed. With the aim of broadening the spectrum of future

applications, case studies have been complemented with an analysis of a body of literature, allowing a few more operators to be added to our set.

The pre-processing operators that have been identified can be grouped into different categories, according to the nature of data transformations they produce.

2.1 Data Cleaning Operators

Often, data come from different sources (either internally generated or acquired from the exterior) and they need to be "consolidated". Also, they may contain redundancies and/or errors, and may be affected by noise and missing values. Finally, as data may have been collected for tasks different from the curretn one, irrilevant attributes may abund. The above operations heavily rely on domain knowledge (see deliverable D5), and they are mostly done manually. An informative and precise specification of meta-data can greatly help manual pre-processing. This issue will be investigated in WP8.

2.1.1 Data consolidation and Data quality survey

Consolidation consists in integrating data from multiple sources into a single data base. Improperly integrated data are a major sources of low data quality. There may be large differences in the way data are memorized: the same variable can be called differently by different data designers, or different variables may receive the same name. Representation format may vary more or less (for instance, there are many ways of writing an address), or measurement units may be inconsistent (for example, US and Canadian dollars cannot be added).

In some cases data to mine are already consolidated by database people, and in general, it would be better to establish procedures and checks to avoid errors since the beginning, but the data base is not always controllable, and quality check may be necessary.

If data are not yet consolidated, manual inspection of meta-data allows redundancies and contradictions in single fields and combinations of fields to be removed. In particular, we consider to check:

- same name given to semantically different variables (operator ω_1)
- different names given to the same variable (ω_2)
- contradictory values assigned within one field (ω_3) or across different fields (ω_4)
- manual removal of known irrelevant attributes (ω_5)

The strong semantic nature of the above operators require manual supervision by a human expert and extensive use of background knowledge and

meta-data. Another important aspect of preliminary data preparation is "anonymization" (ω_6). For privacy or business reasons, data handling should not disclose sensible information contained in some field. Mapping field values into a standard set of anonymous tags can be done semi-automatically.

2.1.2 Missing values and noise handling

In a database several types of noise may be present. First of all, syntactically or semantically erroneous field content. The former can be invoked (if not already done during database construction) automatically (ω_7), but the latter has to be included in operator ω_4. Numerical fields are affected by measurement noise. Reduction of noise in this case (ω_8) can be done by a number of statistical algorithms. Records with many missing values can be discarded, but the data base could be substantially restricted if any record with at least one missing value is discarded. Then, another strategy consists in substituting the missing value with an actual one. There are different methods to do this:

- The missing value is replaced by a default value. For instance, in the case of nominal attributes, a new value can be defined and considered as a default. This strategy is useful, for example, for investigating which records (if any) systematically lack the value (For example, rich people tend to hide their actual income).
- The missing value is computed: the modal value for a nominal attribute, the median for an ordinal one, and the average for continuous variables.
- The missing value is determined on the basis of the statistical incidence of values (for example, in a database in which 40% of the records concern females and 60% males, a missing sex information can be assigned according to the above probabilities).
- A more complex way of calculating a missing value is to use predictive models, constructed on the basis of other variables.

Handling missing values is done by operator ω_9.

2.2 Statistical Overview of Data

Given a database, it is always advisable to perform (if not already done at database population time) a preliminary exploration, by computing basic statistical parameters, such as low order moments (mean, variance, skewness and courtoise) over simple attributes (operator ω_9), univariate hystograms (operator ω_{10}), and correlation coefficients on attribute pairs (operator ω_{11}).

Algorithms are also avilable to identify outliers for univariate (operator ω_{12}) or multivariate (operator ω_{13}) distributions.

2.2.1 Visualization

The operators introduced in Section 2.2 provide the material for a set of simple visualization operators (not the sophisticated ones provided by OLAP of presentation of final results for interpretation). The goal of this visualization step is to gather a general feeling of the database content, useful to prepare a KDD analysis plan. The visualization operators are of three types, according to their output:

- 1D - visualization (operator ω_{14}).
- 2D - visualization (operator ω_{15}).
- 3D - visualization (operator ω_{16}).

Operator ω_{16} is meant to display the results provided by ω_9, ω_{10} and ω_{12}. Operator ω_{15}, in addition to visualize the results provided by ω_{11}, shall also consider 2D scatter plots. Finally, 3D scatter plots and output by ω_{13} shell be handled by ω_{16}.

2.3 Data Selection Operators

The operators defined in Sections 2.1, 2.2, and 2.6 provide a high level overview of the database, and several of them are actually applied even before a specific data mining application is being set up. They have a broad range of applicability and are multi-task by their very nature. Moreover, while the operators for data cleaning (see Section 2.1) may marginally alter the database (by changing or adding or removing some entries), the operators for data overview and visualization leave it unchanged. On the contrary, the pre-processing operators introduced from now on may deeply alteer the database, by changing entries, adding or removing columns or rows, modifying the number and structure of tables, and affecting the meta-data as well.

Specifically, the operators introduced in this section aim at reducing the size of data to be fed to a DM tool.

2.3.1 Sampling and Segmentation

Sampling and segmentation are used to reduce the number of data records within the training data. The underlying goals of these operations are the following:

- Improve speed and reduce memory requirements of the mining tool

- Focus on rare or special cases
- Rebalance the class distributions
- Specify a target group
- Use only clean data

These operations leave the number of attributes and the data records unchanged, but the statistical properties of the population may change dramatically. Sampling can only be applied to one data table.

Examples

- Sampling (operator ω_{17})
  Extracts data records out of the set of training data by random.
  Parameters: number of records to extract, sampling mode.
  Sampling could be done randomly or acccording to a pre-specified probability distribution.
- Typical cases / Atypical cases (operator ω_{18})
  Extracts data records which fulfill the (a)typical case within the training data.
  Parameters: Condition of the (a)typical case.
- Segmentation (operator ω_{19})
  Extracts data records which fulfill a special segmentation-pattern.
  Parameters: Segmentation condition.
  The segmentation conditions can be given implicitly (through a distance measure or attribute weights) or explicitly (through a set of constraints provided by the expert/user). For this operator there is the possibility of specifying as parameter a segmentation algorithm, letting thus an ML tool to find the segments automatically.
- Data quality-motivated sampling (operator ω_{20})
  Extracts data records which values have a certain degree of quality.
  Parameters: Qualify-condition (e.g. all records which have less than 2 unknown values).
- Rebalancing (operator ω_{21})
  Noise-tolerant mining tools cannot be used to build a classification for very unbalanced class distributions (e.g., $95\% - 5\%$, with 5% being an optimistic estimate for responses of a mailing.); then, a sampling favoring the members of the minority class has to be applied first.
  Parameters: class distribution (e.g., $55\% - 45\%$) after rebalancing.

2.3.2 Windowing and Partitioning

Windowing can be applied to data ordered according to a given attribute value (as a special case, time). It consists of selecting all records whose ordering value is within an interval of specified width (the "window"). As moving window it is intented a window, which is made to shift periodically on the records. The amplitude of the window is the main parameter of operator ω_{22}. Some times, a database requires to be partitioned, for instance to perform cross-validation for error estimation, or for reducing algorithm complexity. The number of partitions is a parameter. Even though partitioning could be considered as a special case of sampling, it is more convenient to define a new operator ω_{23}, for clarity reasons.

2.4 Data Transformation

The operators introduced in this section are those that most influence the data mining process and results, as they may introduce deep semantically relevant transformations in the database.

Discretization

There are at least two reasons for discretizing values of numerical continuous attributes: on the one hand, several DM tools are only able to handle categorical values; on the other hand, continuous values may become too sparse over the whole range.

- Discretization (operator ω_{24})
  It is applied to attribute(s) of the type ordinal or scalar and creates a new attribute of the type nominal ordered
  (ordinal/scalar $\rightarrow$ nominal ordered).
  Parameters: base attribute(s), output attribute, number of created intervals for the output attribute.

- Grouping (operator ω_{25})
  It is a kind of order discretization. It applies to attribute(s) of the type nominal that has/have no hierarchy, and creates a new nominal unordered attribute.
  (nominal unordered $\rightarrow$ nominal unordered).
  Parameters: base attribute(s), output attribute, number of created groups, number of data records in one group.

Discretization can be done at constant interval width, at constant counts in the interval, or by letting an algorithm select automatically according to a specific strategy [41, 43, 42].

Feature selection

Attribute selection drops attributes which should not be in the mining input and/or result, e.g., as they are clearly uninteresting, not usable or difficult or expensive to measure for new data. The number of different data records and also the number of the total data records stays the same.

Goals

- Drop attributes which don't fit the input requirements of a DM-tool
- Drop attributes which are strongly dependent on or are the basis for attribute construction for new attributes.
- Improve processing speed.
- Guide the built-in attribute selection process of the DM-tool.

Only the important attributes are chosen as input for a DM-tool. Selecting the attributes can be done manually, through input restrictions associated with the mining tool, or by feature selection methods [24].

Feature selection can be done either manually, with operator ω_{26} (when the expert knows precisely what is important to keep), or automatically (operator ω_{27}), by providing an automated algorithm A with selection criteria.

There are two dimensions along with classify algorithms that perform automatic feature selection (parameters of ω_{27}): one is the feature set ordering, as attributes can be either added or removed. In the former case one starts with just one attribute, and the other ones are added one at a time, until an halt condition is met. In the letter, one starts with the whole set of attributes, abd they are removed one at a time.

The other dimension refers to the relation between feature selection and target system: in the "wrapper" approach, each step of feature selection is embedded in the DM process, whose results provide evaluation feedback for the next step. This approach is highly informative, but computationally expensive.

In the classical approach, instead, feature selection and target task run in sequence, and the evaluation of feature selection is always done independently.

2.4.1 Attribute Construction within a relation

Simple attribute construction creates a new attribute within one data table or view. The new attribute is based on one or more base attributes, and groups their values into a more general form. The number of data records in total is the same as before the operation, but when considering the base attributes replaced by the newly created one, the number of distinguishable data records is possibly reduced (except for relativation and re-scaling).

Goals

- Improve data-coding relative to the capabilities of the mining tool
- Create a new attribute which can be better used for the mining task

Examples

- Relativation (operator ω_{28})
  It puts one attribute in relation to another attribute. It can only be applied on numeric or date attributes and doesn't change the number of different data records.
  (numeric, date $\rightarrow$ numeric ordered).
  Parameters: base attributes, output attribute, operation between the base attributes (e.g., calculating the age from Sysdate and birthdate, calculating the quotient of income and premium sum).

- Cleaning (operator ω_{29})
  Eliminates rare values of data records by creating a new attribute.
  (any type $\rightarrow$ any type).
  Parameters: base attributes, output attribute, which value of the base attribute shall be replaced by which new value (e.g., replacing the entry age of a person younger than one by 1).

- Scaling (operator ω_{30})
  For all distance-based mining tools (e.g., clustering and instance based learning) the scale of the numeric attributes is very important, i.e. attributes with larger values are more influential on the result. To avoid this usually unintended weighting of attributes, all attributes have to be rescaled, e.g., to a fixed standard deviation or interval.
  (scalar$\rightarrow$ scalar).
  Parameters: standard deviation or interval.

- Abstraction (operator ω_{31})
  It is applied to attribute(s) of the type nominal or scalar and creates a new attribute of the type nominal ordered.
  (nominal, scalar $\rightarrow$ nominal ordered).
  Parameters: base attribute(s), output attribute, hierarchy, output of hierarchy level (e.g. looking at the household level instead of person level).

2.4.2 Multi-Relational Attribute Construction

Joins and aggregations are used to put information from several related tables into one base table. To avoid unwanted changes of the target population distribution, the object identity within the base table has to stay unchanged (the number of different data records is still the same after the operations). To avoid the loss of base-table record-joins, outer-joins should generally be

used, and to avoid the duplication of base-table records, simple joins must not be used for 1:N or N:M related tables (as this would duplicate records in the base table).

Goals

- Fit the single table requirement of most DM-tools
- Reduce the complexity for ILP-DM-tools
- Avoid unwanted changes of the population distribution.

Examples

- Algebraic combination (operator ω_{32})
  It creates a scalar attribute, which is the result of an algebraic operation (sum, difference, ratio, ...) on a set of base attributes. income of a household.
- Min, Max (operator ω_{33})
  It creates a scalar ordered attribute (e.g., smallest, highest premium sum of a product, smallest age of a member of the household).
- Count different (operator ω_{34})
  It creates a numeric attribute (e.g., how many persons a household has, how many insurance contracts a household has).
- Count X = V (operator ω_{35})
  It creates a numeric attribute (e.g., how many children a household has, how many different values an attribute has).
- Exist X = Y (operator ω_{36})
  It creates a binary attribute (e.g., there exists an insurance contract of type 3a for one household).
- All X = Y (operator ω_{37})
  It creates a binary attribute (e.g., are all insurance contracts of a household of type 3a are all of a household's adults).

2.5 Time related Preprocessing Operators

The time related preprocessing operators introduced in this section are table-to-table converter. They will be implemented in Oracle 8i SQL and SQL/Plus. Each of the operators needs one or more specific input tables and produces one or several output tables. An example operator producing several output tables is given in Section 2.5.5. A manual operator expecting more than one input table is the multi-relational attribute construction operator of deliverable D1.

Therefore it is quite important that the meta data take into account how to define the *input tables* and *output tables*. In some cases the number of output tables does not depend on the operator itself but on the content of the input tables. That is why the meta data should support the dynamically assignment of an unknown number of output tables as input for the succedding preprocessing step. Alternatively all members of the set of output tables are applied to the succeeding step one by one, e.g., this is the case for the multiple learning operator.

2.5.1 Single Process Extraction

If information from multiple individuals (each individual representing a process) are given, the vectors corresponding to one of the processes can be extracted to derive the information for this single process.

$L_{E_{sP}}$: Let $T_{mp} := \{t_{mp_1}, \ldots, t_{mp_j}\}$ be the set of time specification for the input from which tuples for one single process sp should be extracted. Time specifications for selected vectors of sp are denoted as $T_{sp} := \{t_1, \ldots, t_i\} \subseteq T_{mp}$. At least one attribute is the same (or holds some other specific condition) for all tuples from one single process. This attribute value can be denoted as the *process identifier*;
$(\exists\{attrID_1, \ldots, attrID_i\} : (\forall j \in \{1, \ldots, i\} : attrID_j \in \{1, \ldots, k_j\}))$
$\wedge\, \forall t_m, t_n \in T_{sp} : a_{t_{m_{attrID_m}}} = a_{t_{n_{attrID_n}}} =: process\ identifier$

Since the meta data for the input includes all individual identifiers we just need to know the $individual\ identifier \in individual\ identifiers$ and its value (= process identifier) to extract the data for. The time specification is always expected to be the first (two) attribute(s).

The output table's first column (resp. the first two columns in case of time intervals) is the time specification followed by the process identifier. E.g., if the input is an instantiated database table, we will get an output of the following form:

$$
\begin{array}{lll}
L_{H_{sP}} & & \\
\quad i_l & < t_1, a_{t_{1_{attrID_1}}}, a_{t_{1_1}}, \ldots, a_{t_{1_{attrID_1-1}}}, a_{t_{1_{attrID_1+1}}}, \ldots, a_{t_{1_k}} >, & \\
& < t_2, a_{t_{1_{attrID_2}}}, a_{t_{2_1}}, \ldots, a_{t_{2_{attrID_2-1}}}, a_{t_{2_{attrID_2+1}}}, \ldots, a_{t_{2_k}} >, & (2.1) \\
& \ldots & \\
& < t_i, a_{t_{1_{attrID_i}}}, a_{t_{i_1}}, \ldots, a_{t_{i_{attrID_i-1}}}, a_{t_{i_{attrID_i+1}}}, \ldots, a_{t_{i_k}} >, &
\end{array}
$$

With the notation used in deliverable D1 the implementation of this operator could be done as:

```
Single Process Extraction
  Select time specification, individual identifier,
         remaining attributes
```

```
  From source
  where individual identifier = given process/individual

view -> view (row reduction)
```

2.5.2 Multivariate to Univariate Transformation

$L_{E_{m2u}}$: For the simplest form of multivariate to univariate transformation (I) all we need to know is the attribute which names the value of the univariate time series to produce. The output will then consist of the original time specification and a second attribute containing the unvariate time series' values.

If all possible univariate time series should be created, we will iterate with all attribute of the multivariate time series.

The output $L_{H_{m2uI}}$ is described with:

```
Multivariate to Univariate Transformation I
  Select time specification, attribute
  From Source

view -> view (column reduction)
```

A second feasible transformation (II) reads the values from a multivariate time series from the left to the right and from the top to the bottom and creates one univariate time series.

An example data set contained energy consumption data which was measured every 15 minutes for several days. The first attribute of the table is the day of the measurement, the remaining columns denote the time of the day when the value was measured. They are named "0:00 a.m.", "0:15 a.m.", ..., "11:45 p.m.".

The time specifications for the univariate time series will start with value 1 and increase one by one for each value of the series. The original time attributes were given by the values of the column named *day* and by the names of the remaining columns. They are to be integrated in the output table, too.

If the time specification is not needed feature selection (see deliverable D1) can be used to delete it.

$L_{H_{m2uII}}$:

```
Multivariate to Univariate Transformation II
  For all combinations of time specifications and attributes do
    Select time specification, attribute name, attribute value
    From source

view -> view
```

The output table will have equal or more rows than the input table.

A third multivariate to univariate transformation (III) is needed to compute frequent sequences. The input rows of a multivariate time series are considered events. Each event is defined by the values from a row with k attributes (the time attribute are not considered for defining event types). Thus each new combination of attribute values denotes a new event type. The output is a nominal valued univariate time series of format L_{E_2} based on L'_{E_1}. The single attribute A_1 of this output denotes the event type that occured at that point in time. Therefore the event types are mapped to integers starting with the value 1.

$L_{H_{m2uIII}}$:

```
Multivariate to Univariate Transformation III
  Select time specification,
         map attribute values (attribute 1, ..., attribute k)
  From Source

view -> view (feature construction)
```

A second table is created. It tells about the mapping of event types to integers. The meta data schema should include such mappings.

```
Multivariate to Univariate Transformation III
  Select attribute 1, ..., attribute k,
         map attribute values (attribute 1, attribute k)
  From Source

view -> view (feature construction)
```

Alternatively the event types could be added as constructed feature to the original view. Then feature selection has to be applied to derive the appropriate table for the frequent sequences approach.

2.5.3 Sliding Windows

The input table to the sliding windows operator consists of a time specification and one more attribute. In case of multiple attributes multivariate to univariate transformation should be applied first to provide the correct input table.

$$L_{E_{slW}} := \quad L_{E'_1}$$

In particular the multivariate to univariate transformation II produces appropriate input for the computation of sliding windows of a multivariate

time series and thus windows which include values of two rows from the original multivariate time series.

For a *window size* w the observations within this window are combined to one new vector[1]:

$$\begin{array}{l} L_{H_{slWI}} \\ \quad i_l: \quad t_{1_1}, a_{1_1}, \ldots, t_{1_k}, a_{1_k}, \ \ldots\ , t_{i_1}, a_{i_1}, \ldots, t_{i_k}, a_{i_k} \end{array} \tag{2.2}$$

The window is moved by *window movement* v steps and the new vector is computed. This will be repeated until the end of the input is reached.

Another sliding windows variation (II) compresses the time information to a new time interval:

$$\begin{array}{l} L_{H_{slWII}} \\ \quad i_l: \quad t = (start(t_{1_1}), end(t_{i_k})), a_{1_1}, \ldots, a_{1_k}, \ \ldots\ , a_{i_1}, \ldots, a_{i_k} \end{array} \tag{2.3}$$

Each vector contains only one time specification.

If the windows movement is less than the windows size then the sliding is called *overlapping*, and *non-overlapping* otherwise.

The third sliding window operator (III) is needed for computing frequent episodes. This sliding window operator ensures attribute values at the beginning and at the ending of the univariate time series to be put in as much windows as the values in the middle of the series. Therefore the first and the last window, and in some cases some more windows, extend outside the sequence.

$L_{H_{slWIII}}$: The output format is the same as the one of sliding window I, but it will consist of more output vectors.

All of the sliding window variations belong to:

```
Sliding Window
  View -> View
```

with the outout table having less or equal rows than the input table.

2.5.4 Summarizing

$L_{E_{sum}}$: Attribute values $a_j, \ldots, a_{j+w-1}$ within a window of *past observations* w are summarized by a *function* $f(a_1, \ldots, a_w)$ (e.g., average, variance). The original time series is replaced by the discretized one.

[1]Since this operation will consist of direct cursor implementation on database tables rather than of SQL-Statements the output is not specified with SQL-like statements as in the preceeding section.

$L_{H_{sum}}$: E.g., a univariate time series of representation is replaced by:

$$\begin{array}{lll} L_{H_{sum}} & & \\ i_l: & t = (start(t_1), end(t_{w+1})), f(a_1, \ldots, a_w), & \\ & \ldots & \\ & t = (start(t_{i-w+1}, end(t_i)), f(a_{i-w+1}, \ldots, a_i) & \end{array} \tag{2.4}$$

Each row of the input table contains one window of past observations. If this format is not given already it has to be created with sliding windows.

E.g., for a univariate time series sliding windows with window size w and movement v ($v \in \{1, \ldots, w-1\}$ for overlapping summarizing, $v \geq w$ for non-overlapping summarizing) can be used to generate output rows with each row containing the elements of the past observations w.

Beneath the input table we need a second meta data: the function f. MiningMart will at least provide the moving average function *simple moving average* (SMA), *weighted moving average* (WMA), and *exponential moving average* (MVA) [32]. Basically, all of this moving functions behave the same: they do some computation on observations which are given with the sliding window operator. The window size w used for computing the sliding windows has great influence on the moving function. I.e., a windows size $w = 6$ is used for a lag-six SMA (abbreviated SMA6). This parameter is implicitly given by the number of attributes of the input table passed to the summarizing operator.

With using weights WMA decreases the drawback of each value having the same contribution as the other values to the average of the observation window. EMA uses a tail weight and a head weight for weighting the previous average value and the current value. For both WMA and EMA the weights sum up to 1.

The weights have to be passed as meta date to the summarizing operator. In case of EMA the number of weights is exactly 2, in case of WMA the number of weights depends on the window size of each observation.

2.5.5 Multiple Learning

Instead of handling diverse individuals in one learning run a learning can be started for each individual. In the drug store example for each branch office and each item a separate signal to symbol processing was started.

$L_{E_{multL}}$: The splitting of the input table is controlled by one of the individual identifiers of the table which is denoted *disjunctive attribute* in this context. This operator is almost equal to the single process extraction operator: here we will produce a set of tables (not only one table), one for each value of the individual identifier.

$L_{H_{multL}}$:

```
Multiple Learning
  for all individuals
    select individual, time specification, remaining attributes
    where individual = name of the current individual
    from source;

(view -> views)
```

The output tables are specified by a separate table containing the names of all of the output tables.

2.5.6 Aggregation

$L_{E_{aggr}}$: Aggregation sums up several inputs vector by vector. For the given inputs l_E condition should hold and the set of time specifications should be the same for all inputs. If a time specification does not occur in all of the input tables then only the available data for that time specification will be summed up.

$L_{H_{aggrI}}$: An input vector is read from each of the inputs, all vectors having the same time specification. One output vector is created by summing up all values for each of the attributes.

```
Aggregation I

views -> view
```

$L_{H_{aggrII}}$: A second aggregation operator (II) takes into account that not only the time specifications should be the same for creating a new vector but also any other combination of attributes for a given input table.

There are some more combinations which might be of interest. The key question is: which attributes stay the same for one new output vector (*join attributes*)? For the examples given above these join attributes are the article number and the point in time respectively the article number and the shop number.

If the time specification does not belong to the join attributes then the output will include the start of the earliest time specification and the end of the latest time specification at the beginning of the table.

```
Aggregaton II

view(s) -> view
```

2.6 Operator Description

At a logical level, each operator description consists of three parts:

- Sintactic description, which specifies operator's input, output and parameters. A Boolean tag shall establish whether the method to apply the operator is specified or whether learning has to occur. For instance, in the case of discretization, a parameter could be "constant width interval", or, in the case learning has to occur, "Learn with Algorithm 1".

- Sematic description, which specifies the type of transformation and the effect on the database. For instance, for multi-relational attribute construction:

```
(select <v1.aggregate>
from view, v1
where v1.id = view.id
group by view.id)
->  view.cnew
```

- Pragmatic description, which specifies constraints and applicability conditions. For instance, for discretization, the attribute considered must have continuous values.

The actual description of all the operators will be the content of WP 8 and WP 9.

Chapter 3

Meta-Data for Pre-Processing

The complete description of the available data and meta-data is contained in WP 6. Here we only mention those aspects that are relevant to the definition of the pre-processing operators. Syntactically correct pre- processing is possible with minimal meta-data, i.e., meta-data just specifying the syntactic restrictions of the data, the data-mining tools and the pre- processing operators, but the guidance will improve with any further knowledge gathered about semantic and pragmatic aspects of the pre- processing task.

As often the data in a datawarehouse have different origins (in-house data or data provided by external sources), meta-data provide fundamental information about those sources, especially during data base consolidation, i.e., the integration of different data bases into a single one. Meta-data describe tables as a whole or specific fields. For instance, a table may be described in terms of number of fields/columns, number/percentage of records with missing values, names of the fields, and so on. For a field we have to specify, for example, data type, meaning, description, source of field, unit of measure, number of values, list of values/range of values, number of missing values, and so on.

To describe input formats and meta data the following notation is used:

- Input data are described by formats L_E and identified by a name l_E. L_E comprises either one or several individuals i_l. In case of one individual this individual i_l is used in the definition of L_E, in case of a set I of individuals i_l the set I is used.
- Input formats L_E for a method (operator) producing L_H.
- Attributes A_j and attribute values a_j.
- Abstract time specifications T_j and instantiations t_j, index i always denotes the actual or the last time specification. Since the time speci-

fication may consist of one or two time attributes (explanation below) the term time specification is used here rather than *displacement variable* or *index variable* [32].

Deliverable D1 introduces a first meta data schema. Some of the meta data listed below is already part of the schema. Other meta data will have to be integrated to the schema. This will be part of the WP 8 and 9 which focus on describing the meta data (model).

3.1 Time related Meta-Data

An input l_E which includes time specifications t has to provide additional information. These meta data are mandatory for all time related data.

- the *identification of an individual*. Possibly there is none in the raw data, because all observations belong to one single individual. In some cases the individual's name is given implicitly, e.g., by the database tablename. In these cases the identification has to be added.

  A time series always belongs to one individual but other representations - like sequence vectors - may describe several individuals.

- the attribute(s) containing the time specification (*time attributes*)

- the *time scale* indicated by the *scale unit* and the *point of reference*:

  - the *scale unit*
  - the *point of reference*
  - the order of time stamps

- the *representation of time*: *points in time* or *time intervals* defined by a tuple $t = (t_s, t_e)$ with starting point t_s less or equal than ending point t_e.

Data that does not include explicit time information by having time attributes might include implicit time information: Is the time information given implicitly by the order of an individual's vectors? If not, we will not apply any operator considering time phenomena for the given data.

For any given input these meta data will help to decide whether the input belongs to one of the input languages or not. Furthermore the same meta data could support the mapping of any input to one of the input languages, and from an input language to another.

3.2 Meta-Data required in order to apply Operators

Subject to the pre-processing or learning operator which should be applied some specific meta-data is required.

Time series with time specifications (time-stamped time series) are classified as *uniform* or *generic* time series depending on the time-stamps' characteristics. The time specifications t_j of uniform time series are of unique length (this is always true if time specifications are instantiated by time points) and the distances between two time specifications are the same for all pairs of sequent time specifications:

$$\begin{aligned} &\textit{individual } i_l \textit{ is generic} \\ (\textit{monotonic increasing}) :\Longleftrightarrow\quad & (t_1,\ldots,t_i) \textit{ is monotonic increasing} \\ \Longleftrightarrow\quad & \forall j \in \{1,\ldots,i-1\} : t_j \leq t_{j+1} \end{aligned} \tag{3.1}$$

$$\begin{aligned} &\textit{individual } i_l \textit{ is strictly} \\ \textit{monotonic increasing} :\Longleftrightarrow\quad & (t_1,\ldots,t_i) \textit{ is strictly} \\ & \textit{monotonic increasing} \\ \Longleftrightarrow\quad & \forall j \in \{1,\ldots,i-1\} : t_j < t_{j+1} \end{aligned} \tag{3.2}$$

$$\begin{aligned} &\textit{individual } i_l \textit{ is} \\ \textit{uniform increasing} :\Longleftrightarrow\quad & i_l \textit{ is strictly monotonic increasing} \\ & \exists d : \forall j \in \{1,\ldots,i-1\} : \\ & \wedge t_{j+1} - t_j = d \wedge |t_j| = |t_{j+1}| \end{aligned} \tag{3.3}$$

(Strictly) monotonic decreasing and *uniform decreasing* is defined likewise.

If the time points were not ordered chronologically the time series is called *unsorted* and *sorted* otherwise.

$$\begin{aligned} &\textit{individual } i_l \\ \textit{is unsorted} :\Longleftrightarrow\quad & \exists j,l \in \{1,\ldots,i-1\}, j \neq l : \\ & t_j > t_{j+1} \wedge t_l < t_{l+1} \end{aligned} \tag{3.4}$$

For each of the attributes some *attribute properties* are needed: is the attribute value numerical or nominal (*type*). An important point is that in real-world applications missing values are not only coded by the attribute

value 'NULL' (using database terms). Often mappings are used to code specific attribute values like 'unknown' or 'contractor did not fill in this blank' which can possibly be of the same meaning as 'NULL'. To take this aspect into account the meta data provide an optional list of *NULL values* which identify missing values. A simple example for missing time information is using '9.9.9999' as a date that has not been specified. In many cases learning on '9.9.9999' will not be of great importance.

Additional statistics could be of interest for certain operators: total amount of values, number of different values, mean value, standard deviation, variance, and range. The meta data has to provide corresponding slots for these information.

These meta data are provided for given data as well as a requirement of attributes to apply a specific operator. It is quite important that a step of a data mining case is not only associated with a view and its attributes (or with attributes only) which provide meta data. The meta data has also to provide explicit information about the need for specific meta data as a prerequisite for the data mining step. E.g., if an attribute is associated with a step you will have to know if the type of the attribute is numerical by coincidence or if it is numerical because the step requires the attribute to be of that specific type.

Chapter 4

Composition of Pre-Processing Operators

The development process contains several tasks. A chain of several pre-processors and ML tools, which have a defined order and specific parameters, has to be developed. In this chain ML tools can also be used as pre-processors. The chain is locally optimized to supply the best mining result on training data. The whole chain of all pre-processing operations should be generated into one optimized SQL-statement, which can be executed on the application data.

During the process of developing an optimal pre-processing chain the mining mart power-user has to execute several iterations to find the best fitting (global optimized) chain of pre-processor operations and ML tools. But even the power-user can gain from integrated ML-tools as they help him to find a local optimized solution automatically. In a first attempt a pre-processor chain is defined and executed on the training data. Then the chain is changed and executed on the training data again. After all iterations are completed several pre-processor chains with mining results exist. The chain with the best result has to be chosen to be the best fitting one[1].

An operator chain contains different manual pre-processing operations (PP) and ML tool supported pre-processing operations (MM) in a specific order. Usually the last element of an operation chain is the mining tool. The other elements of the chain can be either manual or ML tool supported pre-processing operations. The ML-tools are used to discover the parameters of the associated manual pre-processor operations. How many elements exists and which order they have depends on the task itself and on the needed input parameter of the next element. Manual pre-processing operations are based on SQL-code. ML tools are stand-alone executables. For performance reasons several pre-processor operations can be translated into just one SQL-

[1]For future versions the other ones could be stored as variants in the case base which help the adaption of a case to a new target segment.

statement in case they are one after another.

The result of a SQL-statement is a view in the database which acts as an intermediate result, i.e. the view is needed as the input for a following ML tool. The result of a ML tool is used as the parameters of the associated SQL-code. As an example consider the discovery of a discretization as the result of a ML tool. The output of the ML tool is a mapping of intervals of the base-attribute (B) to nominal values of the new attribute (A), e.g. {if $B < 18$ then $A =$ child, if $18 \leq B$ then $A =$ adult }. This (and the old column) is used as a parameter of the SQL-function defining a new column in the next intermediate or the final view.

After the whole chain is executed, the final view and several SQL-statements exist. If it is possible, the SQL-statements can be merged and optimized to one statement. This statement or the single generated SQL-statements can also be executed on the testdata or the application data.

In addition to storing chains of pre-processing operator applications, an algebra for composing them can be defined. In particular, thre operations are envisaged up to now:

- Concatenation $(\omega.\sigma\ )$ D
  Operator σ is applied to the database D and then operator ω is applied to the result. In this case the output from σ must be a superset of the input to w. Constraints on σ must be compatible with constraints and applicability conditions of ω. An example is a concatenation of discretization and visualization. Concatenation can be iterated.

- Parallelization ω(D) —— σ (D)
  Operators ω and σ are to be applied both, but their application processes are not interacting, so that they may be parallelized. For instance, for discretization, we may apply two different algorithms on two machines and then compare the outputs.

- Embedding $(\omega\ (\sigma\ ))$ D
  During an iterative application of operator w, the application of operator σ is invoked. That means the execution of ω is interleaved with execution of σ. Embedding is typically exploited when manual and automatic pre-processing steps are to be performed.

Further combination of operators could be added, such as macro-operator formation, according to the needs emerging from analysis of further cases.

Chapter 5

Case Studies

In this section some case studies, used to derive the pre-processing operator specifications, are briefly described.

5.1 A Business Case of Mailing Action at Swiss Life

Ideally, a KDD-project starts with a business case, which could be solved or optimized by analyzing available data. It should be noted that the most important prectical step is the management-agreed specification of how to use the expected mining results, because even the best mining results are worth nothing, if they are not used [Saitta & Neri, 1998]. The most time-consuming step is the preparation of the data for mining. Both problems could be solved in a related manner. The management support for the use of the mining results as well as the justification of the high data preparation costs are most easily reached, if the KDD-project is integrated into an important and repeated business case. However, the second part is only true, if the pre-processing effort can be reused. For Swiss Life there are several application areas where central business- cases could be supported by data mining, especially:

- Marketing
- Product development and control
- Business reporting

The case we analyze here is the optimization of responses to mailing actions in direct marketing as an illustrating and well known example of such a problem.

If we describe this business-case on a more technical level we come to the following steps:

- 0. Use an existing data warehouse (DWH) as base. A partial schema of the Swiss Life's datawarehouse in reported in Deliverable D6.2
- 1. Construct a household view on this DWH, which provides all relevant information in a form, that allows the next step to be done by an end- user.
- 2. Select the target segment, e.g. households, (a) which have a child with an age below 2 and which are not mailed since the birthday of this child, or (b) which have already bought a single-premium insurance, but this is more than two years ago.
- 3. Select a random-sample, with size proportional to 20% of the budget, i.e., generate a sample to gather labels for the training and test set.
- 4. Export the addresses of this sample, do the first mailing, and store the responses, i.e., label the sample.
- 5. Split the sample into training and test-set.
- 6.Select/Construct the relevant attributes for the current response prediction task.
- 7. Train the selected mining-tool, which output could be used to order (not just classify) the data.
- 8. Apply the data-transformations (pre-processing) done in step 6 to the test-data as the mined pattern relies on it.
- 9. Test the mined pattern on the test-set, i.e get an estimated response-rate for the mined pattern on the target-group.
- 10. Apply the data-transformations (pre-processing) done in step 6 to the target-segment as the mined pattern relies on it.
- 11. Select the best (ordered by the mined reponse-pattern) records (proportional to 80
- 12. Export the addresses of this selection and do the real mailing.
- 13. Compute a final evaluation, and store all the mailing-information (date, (non-) responses, segment, product, mined pattern, data-transformations, evaluation, . . .) in the DWH/DWH-meta-data-Repository, such that it could be used as background knowledge for further actions. Compared to the standard KDD-process, step 0 corresponds to data selection, collection integration and cleaning. We will not investigate that in detail, as this is well investigated in the data warehouse community, but instead rely on an already build data warehouse (DWH),

which is in fact an ideal first steps in setting up a KDD process [Inmon, 1996].

When we start the pre-processing phase on top of such a DWH, we are faced with

- a normalized, multi-relational database as data source (whereas most existing data mining algorithms are single-table based),
- many features and tables which are only partially relevant for the current business-case.
- a feature value coding optimized for maintenance (e.g., weekly update) of the data source, and not for optimal use within a specific data mining tool for a specific task.

For our mailing-case this means, that the mapping between DWH data representation and the end-users needed to specify a customer segment (step 2) is far from being trivial. So the first step (1) is to generate a more application oriented view on top of the DWH schema. As this is an application oriented view, it depends strongly on the business case to solve (e.g., for product-development the base-level is not households, but insurance-contracts). Therefore, it already builds the first group of pre-processing steps to be handled (stored, retrieved and adapted) by our case-base. The other group is the one described in step 6. Whereas, step 1 does not need very much adaption for reuse, if we understand by reuse just the application of this schema to another target segment, this is not true for step 6, as the different target segments have very different properties. Therefore quite different attributes are relevant for predicting the behaviour, e.g., as a general rule it could be stated that most households addressed by 2.a are not ensured so far, whereas the previous insurance behaviour is very likely to be important for predicting the response for segment 2.b. The base operations of pre-processing cases considered in this application are sampling and segmentation, feature construction within a table and over related tables, and feature selection. In this approach feature construction and dropping of base-features is separated, as features may be needed for more than one feature construction operation (e.g., the date of birth of a person is needed to construct the age of the person, the entry-age into a contract and the end-age of a contract for every contract).

5.2 Internet Intrusion Detection dataset

The problem of protecting intranets is a very hot one. Every day hackers become more dangerous and new attack methods are invented that defeat the existing protections. Hence it is a diffused feeling that no protection scheme can be considered as unbreakable. Therefore, a continuous monitoring activity is required by system administrators in order to discover and neutralize the new attack methods that are invented.

The monitoring activity usually consists in analyzing megabytes of firewall logs, and is very expensive in terms of man power. Data mining and KDD potentially can offer a valuable help to system administrators in order to reduce the cost and increase the effectivness of the anti intrusion monitoring activity.

The university intranets are site particularly difficult to protect because it is not possible to enforce restrictve protection policies which would negatively affect the research activity. For instance the students are strongly encouraged to use the WEB for bibliographic researches and for other kind of academic acivities. However the free access to the WEB is potentially very dangerous because many succesful intrustions occur through WEB pages supporting CGI which allow a telnet process to be started.

A furter difficulty is due to the presence inside the intranet itself of users intrinsically untrustable, which can attack the defenses also from the internal side and may collaborate with attackers from the external side.

5.2.1 Experimental Setup

In order to investigate the possibility of detecting the different forms of attack from inside and from outside the intranet the following experimental setting has been arranged. A student laboratory provided with 32 workstations has been connected to Internet and to the research intranet throug a firewall SunScreen 3.0 which provided a full set of tool for enforcing protection both at network level and at application level. The firewall is provided with log facilities which allow any relevant event to be recoded. In the extreme case the full traffic through the firewall can be recorded.

As the laboratory is finalized exclusively to training activity privacy doesn't need to be guaranteed and so the recorded data can be of freely used for research purposes. The experimental setup is described in Figure 5.1.

Using such an environment, it is easy to simulate different kinds of attack using tools available on the network and/or exploiting the cooperation of some *experts*. The data are collected both in normal conditions and during an attack phase. In this case, the data are labeled with the known attack type.

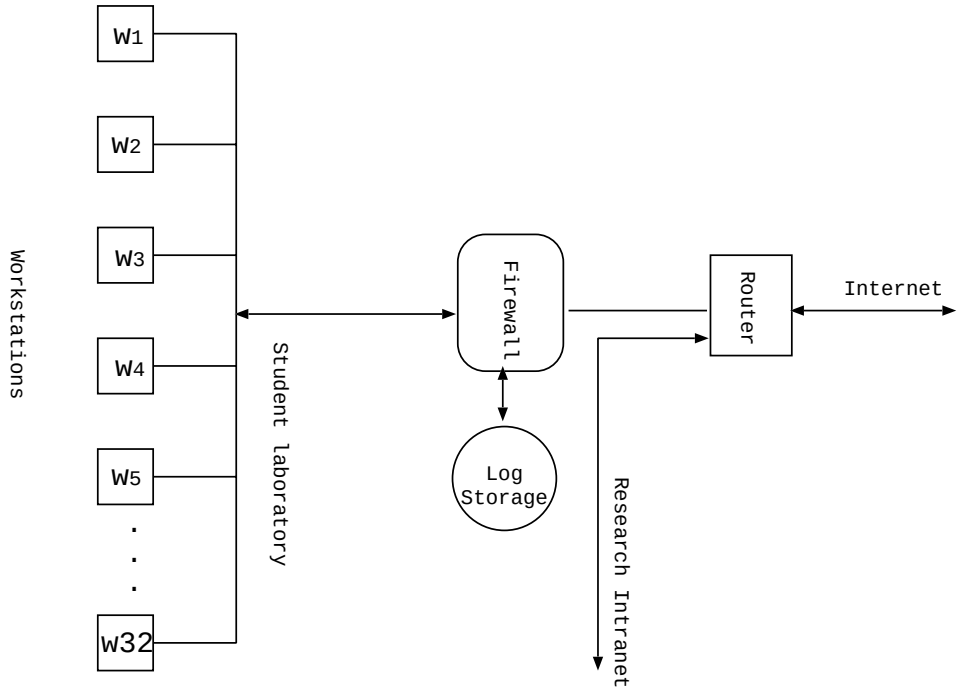

Figure 5.1: Organization of the laboratory used for collecting the datasets for anti-intrusion monitoring.

5.2.2 Data Structure

Data from the firewall LOG can be obtained with different degree of detail depending on the setting. The current choice is to adopt a medium level of detail as reported in Table 5.1

The attack type is recorded in the field "Attk" when a specific attack has been simulated or noticed. The "no" label means "no attack noticed". In this way, it is possible to group together all packets belonging to a same attack action in order to characterize the temporal evolution of the attack itself. The data are recorder in this format in a relational database (ORACLE 8.1i), in order to have an easy interface toward KDD tools used in the project.

The *time* field report the time in second from the beginning of the recording session. All the records belonging to a same session have the same session identification number.

Table 5.1: Format of the data extracted from the firewall LOG. Every record is a summary of the header of the packet arrived on one of the two network interfaces of the firewall.

Field	Example	Description
Session	3	Uniquely identfy the experimental session
Number	236	The record number
Int	hme0	Firewall' ethernet interface the packet arrived on
Exitus	pass	If the packet got a cross the firwall or not
time	3.14955	The arrival time in seconds
DIP	192.168.69.11	Destination IP number
SIP	212.239.13.31	Source IP number
LEN	1237	The total length of the packet
ID	11770	Packet' ID
TPRT	TCP	Transport level protocol (TCP/UDP)
D Port	2322	Destination port at TCP/UDP level
S Port	80	Sorce port at TCP/UDP level
Fin	no	If the packet is the last of a connection or not
Ack	1557829	Packet aknowledgment in Piggy/back
Seq	687346909	Sequence number in a TCP stream (empty for UDP)
Len	1197	Lenght of the TCP/UDP packet
Win	7584	Available buffer size in the receiver (TCP only)
ALP	WWW	Application Level Protocol (WWW, FTP, ...)
VER	HTTP/1.0	Version of the protocol
Dir	R	Receiving or calling
Attk	no	Attack type (no means no attack)

5.3 Fraud detection in mobile phones

The reduction of frauds is a potential challenge to telecommunications providers. Nowadays technology supplies several Fraud Management Systems (FMS) that allow fraudulent behaviors to be identified. A FMS performs a filtering of a high number of TICKET (Call Data Record, E-Transaction, Logs from servers and firewall over Internet, etc.) in the way that alarms are produced when strange behaviors (i.e. possible frauds/attacks) are detected. A perfect FMS should produce an alarm when a real fraud/attack occurs. Unfortunately, this is practically unfeasible because of the intrinsic nature of the problem to be faced, namely understanding, thus isolating, frauder behaviors. Today, alarms could also signal usual customers, as frauder behaviors are not accurately identified, thus resulting in a reduction of the overall performance of a fraud department at a telecommunication provider. The complexity of frauds require the design of highly adaptable detection systems of whom data filtering policy should be adjusted according to new customer behaviors. The identification of proper policy takes advantage of the analysis performed using previous known cases. The analysis takes advantage of OLAP and Data Mining tools with the goal of learning predictive fraud models, thus building antifraud strategies. The experience has demonstrated how intelligent TICKET filtering is a difficult task. As a matter of the fact, not useful alarms often flood the entire detection systems with the unpleasant consequence of possible overloads and shutdowns; hence algorithms that allow orienting the definition of narrow as selective as criteria is welcomed.

5.3.1 A fraud detection system architecture

Figure 5.2 shows detection system multi-layer architecture. A TICKET stream is furnished to a straightforward filter battery that rejects a certain amount of TICKET (presumably the most), and supplies to a second more sophisticated rule-based rule battery its result. The latter perform a deeper TICKET analysis, thus provides to the system administration a collection of signals about possible frauds. The multi-layer structure is mainly motivated by the troughtput the stream analysis requires: huge amounts of TICKETS daily have to be manipulated and usually a small amount of them could give problems. Hence, a first reject mechanisms that discard the most of the data, allows alleviating the computational load of the most sophisticated part of the detection system.

5.3.2 Decision Support System and Fraud Detection

The number of TICKET a Telecommunication Operator has to tackle is incredible large. Usually, million of customer exploit the services it offers, thus

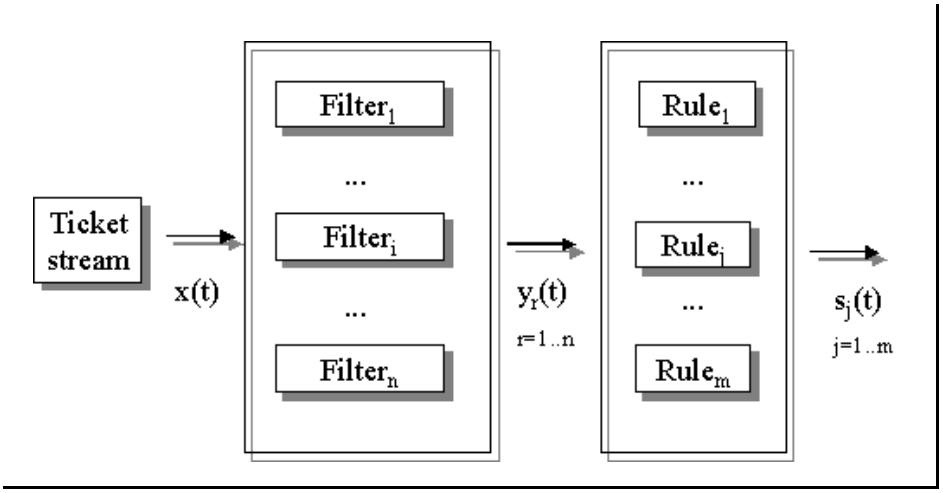

Figure 5.2: Fraud detection system architecture

also the dimension of the database (DataWarehouse/DataMart) containing customer information are considerable. If each customer makes 3 call/day and there are 3 million of customers for a certain phone service, 9 million of TICKET/day (Firewall Log analysis presents analogous problems) will be generated, and analyzed.

With the goal of ameliorating the performance of the fraud detection system, its selection capabilities, and adapt its behavior to the dynamic of the TICKET generation, it is often necessary to perform off-line mining analysis over historical TICKET properly memorized. Actually, monitoring and detection systems are difficult to control. If the supervisor does not select narrow filtering policy, the entire antifraud infrastructure will be flooded by alarms that could not be properly handled. Overload occurs and possible shutdowns could also be required.

OLAP and Data Mining are exploited as decision support mechanisms both to understand how the detection system performs and eventually which kinds of changes are required to ameliorate its behaviors. This analysis exploit TICKET tables, information about customers, and the signals produced by the monitoring and detection systems in a given time window (e.g. a few months). A DDS is exploited to analyses these data. This framework contains Federated Data Mart as shown in Figure 5.3. Three modules are included the decision support system:

- Loading module that acquires the data from OLTP environments and load the proper Federated DM and tables for the analysis; the load procedure is mainly characterized by data normalization, cleaning and trans-formation operations;

- OLAP tools that allow generating special indexes, increasing the query

N

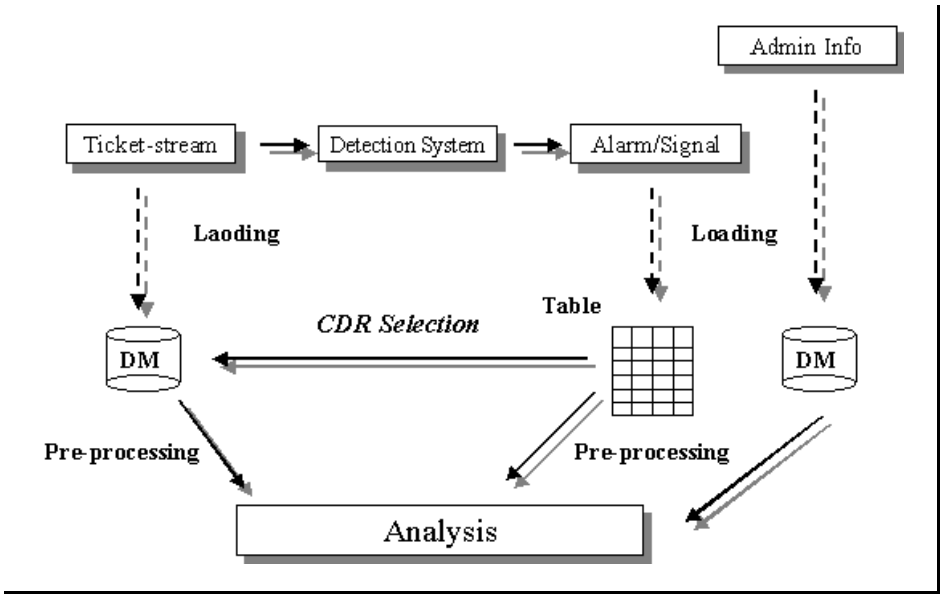

Figure 5.3: Data analysis framework.

efficiency, augmenting the parallelism.

- OLAP and Data Mining tools that permit multi-dimensional analysis, query and reporting capability, and mining.

In the DSS described in Figure 5.3, the result produced by the detection system is stored into a simple table. In fact, it could also by inserted into a DM, for instance when new detection needs are required as typical in the contest of telecommunication services. The Federated Data Mart approach is particularly adapted for this purpose. A DM is characterized by a collection of homogeneous information and small dimensions. This feature alleviates and speeds up the analysis, and allows partitioning a huge DW off into a set of smaller DM. In general, these small DM can be also replaced, created or destroyed during the life cycle of the entire framework. The classical "star schema", shown in Figure 5.4, has been used to implement the DM included in the DDS: a fact table and a collection of dimension tables are implemented as storage support. The fact table contains detailed numerical information whereas the dimension tables contain data about time, geographical area, costs, rate, etc.

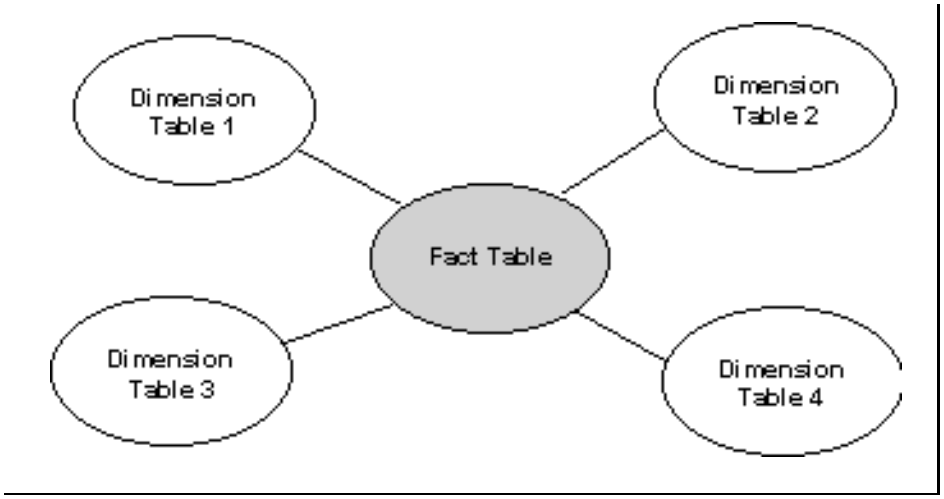

Figure 5.4: Data Mart snow flake structure.

5.4 TICKET DM: a case study

5.4.1 Internal structure

The fact table of the TICKET DM (see Figure 5.5) holds the TICKET sequence generated by the customers of a telecommunication service. Let consider a CDR flow, the TICKET will contain the following attributes:

- Caller
- Call data/time
- Call length
- Call cost
- Callee
- Direction.

The dimension tables are:

- Subscription contains customer information;
- Call type: identifies the call type, i.e. national call, call origin/destination roaming call, follow me, (there are 5 possible classes);
- Time: maintains the date/time of the call, it is structured with the hierarchy shown in Figure 5.6

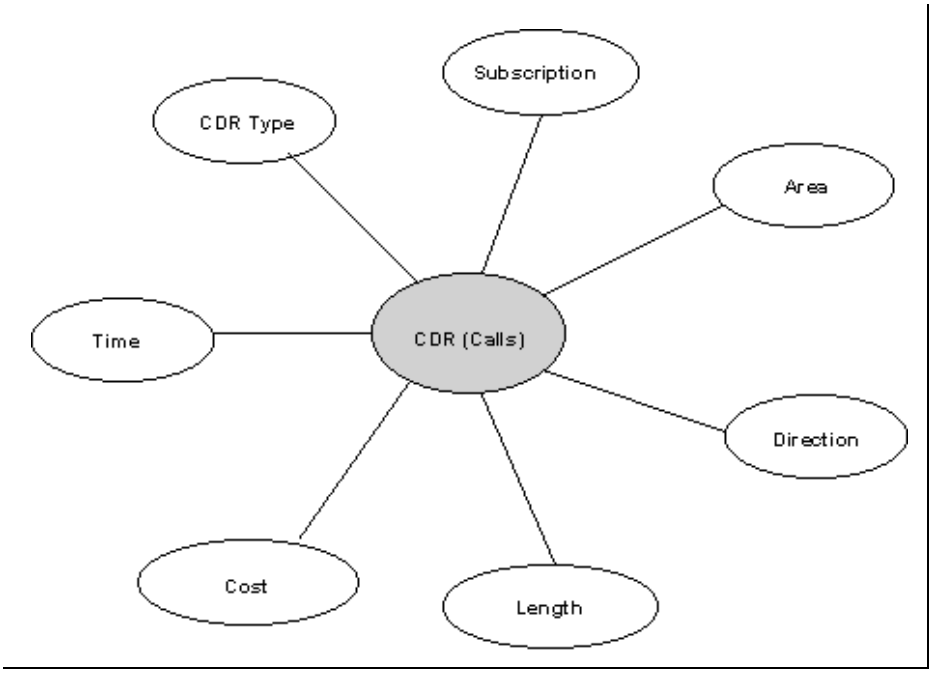

Figure 5.5: Internal structure

- Direction: contain information about the network used by the callee (telephone, cellular, international, special number, etc.); there are about 260 classes, namely one for any country code plus other dedicated classes; the direction posses a hierarchy the usually depend upon the network of the callee:
  - for the national network, the hierarchy is region, province, district;
  - for international network, the hierarchy is group of nation, nation;
  - for mobile network, the hierarchy is operator, network type, subscription type.
- Area: indicates the caller position in the case of mobile network (there are 15000 different classes) The hierarchy depends upon the caller position: * if the caller is within the country border, the complete hierarchy of all network plants that support the connection, is available; * if the caller is abroad, the hierarchy is group of nation, nation.
- Length: associated to each call there is its effective call length and a length class (20 level)
- Cost: associate with each call there is its effective cost and a cost class (20 level)

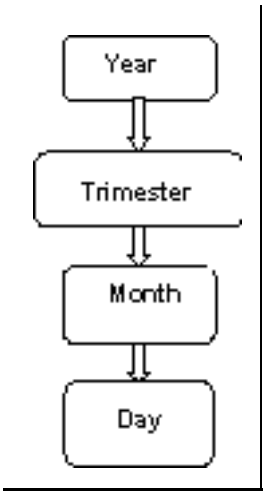

Figure 5.6: Time and date hierarchy.

5.4.2 Loading procedure

Because of the reduced number of OLTP environments employed the problems related to the data format are in general irrelevant. The practical experience demonstrates how filed with the same label but different value, or field with different label but same value are not present. The loading performs the following actions:

- Normalization of the phone numbers in order to distinguish international, national, and number identifiers;
- Data input filtering;
- Cost computation for each call;
- Computation of derivative information;
- Computation of synthetic dada;
- Discretization of cost and length of each call
- Creation/valorization of the surrogate keys both to increase the efficiency and to allow the versioning.

The loading procedure can take advantage of metadata describing each action in term of operation to be performed and logs structures (the action execution is usually monitored for debugging and postmortem analysis).

5.5 Predictive modelling

This section describes the mining process exploited to define a predictive model of the customer behaviors of a telecommunication service. The modeling process receives as input:

1. a TICKET table (Call Data Records, E-Transactions, Server/Firewall Logs, etc.);
2. enciphered administrative information about customers;
3. a table that contains information about the signals produced by the fraud detection system enriched with the attribute "fraud" (yes/no) that confirms/negates the trustworthiness of each alarm; the field "fraud" is filled by the supervisors that verified the each signal.

The following steps define the mining process.

1. Data extraction
   - TICKET extraction - the TICKET with the proper time window correlated to the signals are extracted from the TICKET table, the result is:

$$< call_I (= customer_k) > \ldots < attr_{I,j} > \ldots < attr_{I,M} > with j \in [1..M] \quad (5.1)$$

   - enciphered administrative information extraction - the enciphered administrative information related to the signals are extracted from the customers' DM, the result is:

$$< customer_k >, < attr_{k,l} > \ldots < attr_{k,j} > \ldots < attr_{k,N} > with j \in [1..N] \quad (5.2)$$

2. data manipulation
   - signal manipulation - as the detection system might produce for each customer (possible frauder) more signals, a merging operator is applied to obtain one signal for each customer, the result is

$$< customer_k >, < fraud_{k,l} \in [yes, no] > \ldots < attribute_{k,j} > \ldots < attribute_{k,N} > with j \in [1..N] \quad (5.3)$$

where $< customer >< fraud >$ relation is biunivoca; the filed $< fraud >$ might also be filled with a fuzzy value in the interval [0..1], i.e. the ratio between true vs. false signal associated to that customer in that time window;

- data manipulation about TICKET and/or customer information (contract initial time, data and time, etc.);
- variable construction (usually obtained by store-procedures) - consider this general example, let the k-th attribute in (5.1) o in (5.2) is an ordinal or categorical value, for each value it can assume and for each customer we make the table:

$$< customer_I >, < value^K_{I,l} >, \ldots < value^k_{I,j} >, \ldots < value^k_{I,A} > \tag{5.4}$$

where A is the arity of the k-th attribute, $< value^k_{I,j} >$ is a function of the other attributes computed with the value they assume for $< customer_I >$. In general, it is interesting evaluate different attribute over different function for each customer.

3. Visualization and interactive analysis

   - the use of synoptically views about statistical distributions and data correlation allow the analyst explore his/her data in order to understand the phenomena is characterizing;

4. Variable selection

   - classical algorithm are exploited here;

5. Stratified sampling

   - with binary target (fraud yes/no);
   - with typical/atypical case.

6. Profit/loss matrix

   - as the number of frauder is small a profit/loss matrix is exploited to properly drive the modeling algorithm in the way that it might emphasize the frauder behavior;

7. Predictive model construction by decision tree mechanisms (C4.5)

   - a decision tree has been chosen because self explicate, and easy to translate into detection rules; a certain number of models are built with different input data and/or different parameter settings.

8. Model evaluation

   - the obtained models are compared with standard techniques (Lift, ROC);
   - the best model obtained within a certain deadline is adopted
   - bagging & boosting techniques are also exploited.

5.6 Minor tips

The hierarchies and the discretization implemented have been carried out because of specific OLAP requirements, whereas data mining needs have not been taken into account. This choice is mainly motivated by the concrete difficulty of isolate, in this project phase, useful but not restrictive data mining requirements. The data extraction and transformation are carried out with both data mining tools (SAS/EM) or PL/SQL store procedure (Oracle). In the latter case, the tickets associated to a single user are properly manipulated (see variable construction in the previous section). Preprocessing often exploit discretization operators as a mean of reduce the number of possible value and create synthetic classes.

5.7 Intensive care

Records of patients in an intensive care unit have been collected. They offer raw data. The length of the time series is not determined once for all patients, but denotes the length of the patent's stay in the intensive care unit. Hence, the database table is organised as consecutive parts of the time series.

$L_{E_{DB2}}$ **tuples for time points:** The database does not stores all measurements of one individual in one row, but only the measurements at one point in time: $I : TA_1 \ldots A_k$. There are several rows for one individual. The number of measurements needs not be equal for different individuals.

$i_1 : t_1 a_{1_1} \ldots a_{1_k} \ldots i_1 : t_i a_{i_1} \ldots a_{i_k}$

$\ldots$

$i_z : t_l a_{l_1} \ldots a_{l_k}$

We explored a variety of learning tasks within this application. The learning when and how to change the dosage of drugs is – with respect to prepocessing and learning – similar to the shop example, but we had to combine different rows of the table first.

Chaining database rows: Select all rows concerning the same individual and group its attributes by the time points. Output a vector of the length of the time series of this individual. The result is a multivariate time series.

We used the SVM, now in the classification mode [17]. We experimented with many different features that were formed by sliding windows and different summarization methods. However, the past did not contribute to learning the decision rule. Hence, we learned from patients' state at t_i whether and how to intervene at t_{i+1}[29]. Therefore, in the end we could use the original data. The real difference to the shop application is that the decision rule is learned from a large training set of different patients and then applied to previously unseen patients and their states.

For a different learning task in the intensive care application, a method for time series analysis was applied. The learning task was to find outliers and detect level changes. A new statistical method was used [6]. It transforms measurements of one vital sign of the patient such that the length of the window is interpreted as dimensions of Euclidian space. Choosing $w = 2$, the measurements of two consecutive time points are depicted as one point in a two-dimensional coordinate system. It turns out, that outliers leave the ellipse of homogeneous measurements, and a level change can be seen as a new ellipse in another region of the space. The method is a special case of sliding windows.

In the intensive care application, current work now uses the detected level changes as input to a relational learning algorithm. This allows to combine various time series and detect dependencies among parameters, deviations from a stable (healthy) state, and therapeutical interventions. The learning task is to find time relations that express therapy protocols, in other words effective sequences of interventions.

Bibliography

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, Washington, D. C., may 1993.

[2] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 12, pages 307–328. AAAI Press/The MIT Press, Cambridge Massachusetts, London England, 1996.

[3] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *International Conference on Data Engineering*, Taipei, Taiwan, mar 1995.

[4] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

[5] James F. Allen. Maintaining knowledge about temporal intervals. In R. J. Brachman and H. J. Levesque, editors, *Readings in Knowledge Representation*, chapter VII, pages 509–523. Morgan Kaufman, Los Altos, CA, 1985.

[6] M. Bauer, U. Gather, and M. Imhoff. Analysis of high dimensional data from intensive care medicine. Technical Report 13/1998, Sonderforschungsbereich 475, Universit"at Dortmund, 1998.

[7] Francesco Bergadano and Daniele Gunetti. *Inductive logic programming:from machine learning to software engineering.* The MIT Press, Cambridge, Mass., 1996.

[8] Pvael 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.

[9] Gautam Das, King-Ip Lin, Heikki Mannila, Gopal Renganathan, and Padhraic Smyth. Rule Discovery from Time Series. In Rakesh Agrawal, Paul E. Stolorz, and Gregory Piatetsky-Shapiro, editors, *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pages 16 – 22, Ney York City, 1998. AAAI Press.

[10] Luc Dehaspe and Hannu Toivonen. Discovery of Frequent DATALOG Patterns. *Data Mining and Knowledge Discovery*, 3(1):7 – 36, 1999.

[11] Luc DeRaedt. *Interactive Theory Revision: an Inductive Logic Programming Approach*. Acad. Press, London [u.a.], 1992.

[12] G. Dong and S. Ginsburg. On the decomposition of chain datalog programs into p (left-)linear l-rule components. *Logic Programming*, 23:203 – 236, 1995.

[13] 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*, aug 1996.

[14] 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.

[15] Gerald Gazdar and Chris Mellish. *Natural Language Processing in PROLOG*. Addison Wesley, Workingham u.a., 1989.

[16] Valery Guralnik and Jaideep Srivastava. Event detection from time series data. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 33 – 42, San Diego, USA, 1999.

[17] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11. MIT-Press, 1999.

[18] J.-U. Kietz and S. Wrobel. Controlling the complexity of learning in logic through syntactic and task–oriented models. In Stephen Muggleton, editor, *Inductive Logic Programming.*, number 38 in The A.P.I.C. Series, chapter 16, pages 335–360. Academic Press, London [u.a.], 1992.

[19] Jörg-Uwe Kietz and Marcus Lübbe. An efficient subsumption algorithm for inductive logic programming. In W. Cohen and H. Hirsh, editors, *Proceedings of the 11th International Conference on Machine Learning IML–94*, San Francisco, CA, 1994. Morgan Kaufmann.

[20] Volker Klingspor. *Reaktives Planen mit gelernten Begriffen.* PhD thesis, Univ. Dortmund, 1998.

[21] Volker Klingspor and Katharina Morik. Learning understandable concepts for robot navigation. In K. Morik, V. Klingspor, and M. Kaiser, editors, *Making Robots Smarter – Combining Sensing and Action through Robot Learning.* Kluwer, 1999.

[22] S. Kramer, B. Pfahringer, and C. Helma. Stochastic propositionalization of non-determinate background knowledge. In D. Page, editor, *Procs. 8th International Workshop on Inductive Logic Programming*, pages 80 – 94. Springer, 1998.

[23] Nada Lavrač and Sašo Džeroski. *Inductive Logic Programming — Techniques and Applications.* Number 148 in Artificial Intelligence. Ellis Horwood, Hertfortshire, 1994.

[24] H. Liu and H. Motoda. *Feature Extraction, Construction, and Selection: A Data Mining Perspective.* Kluwer, 1998.

[25] R.S. Michalski and T.G. Dietterich. Learning to predict sequences. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning - An Artificial Intelligence Approach Vol II*, pages 63–106. Tioga Publishing Company, Los Altos, 1986.

[26] Ryszard Michalski. Inferential learning theory as a basis for multistrategy task-adaptive learning. In Michalski and Tecuci, editors, *Multistrategy Learning.* George Mason University, USA, 1991.

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

[28] Katharina Morik. Tailoring representations to different requirements. In Osamu Watanabe and Takashi Yokomori, editors, *Algorithmic Learning Theory – Procs. 10th Int. Conf. ALT99*, Lecture Notes in Artificial Intelligence, pages 1 – 12. Springer, 1999.

[29] Katharina Morik, Peter Brockhausen, and Thorsten Joachims. Combining statistical learning with a knowledge-based approach – A case study in intensive care monitoring. In *Proc. 16th Int'l Conf. on Machine Learning (ICML-99)*, Bled, Slowenien, 1999.

[30] Katharina Morik and Stephanie Wessel. Incremental signal to symbol processing. In K.Morik, M. Kaiser, and V. Klingspor, editors, *Making Robots Smarter – Combining Sensing and Action through Robot Learning*, chapter 11, pages 185 –198. Kluwer Academic Publ., 1999.

[31] S. Muggleton, A. Srinivasan, R. King, and M. Sternberg. Biochemical knowledge discovery using inductive logic programming. In Hiroshi Motoda, editor, *Procs. First International Conference on Discovery Science*. Springer, 1998.

[32] Dorian Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann Publishers, 1999.

[33] Anke D. Rieger. *Program Optimization for Temporal Reasoning within a Logic Programming Framework*. PhD thesis, Universität Dortmund, Germany, Dortmund, FRG, 1998.

[34] 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.

[35] Devika Subramanian. A theory of justified reformulations. In D. Benjamin, Paul, editor, *Change of Representation and Inductive Bias*, pages 147–167. Kluwer, 1990.

[36] 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.

[37] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[38] D.H. Wolpert and W.G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fé Institute, Santa Fé, CA., 1995.

[39] Stefan Wrobel. *Concept Formation and Knowledge Revision*. Kluwer Academic Publishers, Dordrecht, 1994.

[40] Wei Zhang. A region-based approach to discovering temporal structures in data. In Ivan Bratko and Saso Dzeroski, editors, *Proc. of 16th Int. Conf. on Machine Learning*, pages 484 – 492. Morgan Kaufmann, 1999.

[41] U.M. Fayyad. Data Mining and Knowledge Discovery: Making Sense out of Data. *IEEE Expert*, 11(5):20–25, 1996. October 1996.

[42] G.H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the Eleventh International Conference on Machine Learning*, volume MLC-94, pages 121–129, New Brunswick, NJ, 1994. Morgan Kaufman.

[43] R. Kohavi and D. Sommerfield. Features subset selection using the wrapper method: Overfitting and dynamic search space topology. In *Proceedings of the First International Conference on Knowlede Discovery and Data Mining*, pages 192–197, Montreal, Quebec, 1995. AAAI Press.

[44] W. H. Inmon. The data warehouse and data mining. *Communications of the ACM*, 39(11):49–50, November 1996.

[45] M. Staudt, J.-U. Kietz, und U. Reimer. A data mining support environment and its application on insurance data. In R. Agrawal und P. Stolorz, editors, *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 105–111. AAAI Press, 1998.