Instituto de Ciências Matemáticas e de Computação

ISSN - 0103-2569

Data Preparation, Reduction and Prediction in the Context of Data Mining: A Case Study with Insurance Policies

Maria Carolina Monard/ILTC
Chandler Wellington Caulkins
José Augusto Baranauskas
Robson Butaca T. de Oliveira
Solange Oliveira Rezende

Nº 81

RELATÓRIOS TÉCNICOS DO ICMC

São Carlos

Novembro/1998

Data Preparation, Reduction and Prediction in the Context of Data Mining: A Case Study with Insurance Policies*

Maria Carolina Monard/ILTC

Chandler Wellington Caulkins

José Augusto Baranauskas

Robson Butaca T. de Oliveira

Solange Oliveira Rezende

Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação e Estatística
Caixa Postal 668, 13560-970 - São Carlos, SP, Brasil
e-mail: {mcmonard, chandler, jaugusto, butaca, solange}@icmc.sc.usp.br

Abstract

Although data preparation and transformation are critical for Data Mining, they often do not receive enough attention in the research literature. In this work we describe some preprocessing steps we have used over a real data set to make it possible to apply supervised propositional and relational concept learning algorithms to extract knowledge from that data set. The Machine Learning algorithms used are CN2, C4.5 and C4.5-rules which learn propositional rules, and *GOLEM* which learns first-order rules.

November 1998

*Work partially supported by National Research Councils — CNPq and FINEP, Brasil

Contents

List of Figures

List of Tables

1 Introduction

Predictive Data Mining – DM – can be decomposed into four major tasks[1, 13]

1. Data Preparation
2. Data Reduction
3. Data Modelling and Prediction
4. Analysis of Solutions

Although data preparation and transformation are critical for DM, they often do not receive enough attention in the research literature. In this work we describe some preprocessing steps we have used over a real data set to make it possible to apply supervised concept learning algorithms to the original data set.

Once the data has been prepared, some sort of method should be used to extract knowledge from the data. In this work we decided to use four Machine Learning algorithms: CN2, C4.5, C4.5-rules and *GOLEM*. All of them are able to learn hypotheses (concept descriptions) in a human readable representation as a set of if-then rules, given a set of examples.

CN2, C4.5 and C4.5-rules learn propositional rules (in fact, C4.5 learns decision trees which can be transformed in decision rules). In propositional rules there are no variables or quantifiers, and no relations among example components. On the other hand, the relational algorithm *GOLEM* learns sets of first-order rules that contain variables, which are more expressive than propositional rules. It should be observed that relational learning algorithms can be easily applied to relational databases, taking advantage of the representation to find new relations.

This work is organized as follows. In Section 2 a concise description is made of the real dataset used in this work. This data is summarized with a few tables in Section 3. The actual task to be performed by the Machine Learning algorithms is detailed in Section 4, along with the preprocessing steps taken to accomplish this task. Having shown this, the algorithms applied to the dataset are described, in Sections 5 and 6, as well as the additional processing steps needed to adapt the data to the propositional and relational approaches. Finally, in Section 7, some conclusions are presented.

2 Original Data Set Description

The KDD Sisyphus I data set is an excerpt from a data warehouse system called MASY which is employed to collect and analyse data from the private life insurance business at Swiss Life. Swiss Life is the market leader for life insurances and pension schemes in Switzerland and one of the leading companies in these areas in Europe.

The data set was prepared by the Swiss Life Information Systems Research group – SLISR. Comments, questions and criticisms should be directed to Dr. Joerg-Uwe Kietz or Dr. Martin Staudt e-mail: `{Uwe.Kietz,Martin.Staudt}@swisslife.ch`.

2.1 File Overview

The KDD Sisyphus I data set consists of several data files. Table 1 shows only the original data files used in this work to tackle taska.

1	eadr.dat	data file for relation eadr
2	padr.dat	data file for relation padr
3	parrol.dat	data file for relation parrol
4	part.dat	data file for relation part
5	tfkomp.dat	data file for relation tfkomp
6	tfrol.dat	data file for relation tfrol
7	vvert.dat	data file for relation vvert
	taska.dat	data file for task A

Table 1: Data Files

Each data file contains records in Prolog-like style, one tuple per line

```
<relname>(comp_1,....,comp_n).
```

Table 2 describes the contents of the original data files. It should be observed that the item Missing Values (y/n) refers to a file which has at least one missing attribute value (value `$null$` in the original data files). The number of missing attribute values can be seen in Table 4. Although the name of the relation is in fact different than the file name, from now on we shall refer to the file name in place of the actual relation name.

Data File	Relation/Arity	No. of Records	Missing Values	Description
1.-eadr	o__rel5o/3	505	n	Electronical Address
2.-padr	o__rel6o/4	17970	y	Postal Address
3.-parrol	o__rel2o/5	111077	n	Partner Roles
4.-part	o__rel13o/9	17267	y	Partners Involved in Insurance Contracts
5.-tfkomp	o__rel4o/26	73502	y	Tariff Components
6.-tfrol	o__rel3o/8	73332	y	Tariff Roles
7.-vvert	o__rel1o/18	34986	y	Life Insurance Policy
taska	o__rel18o/2	17262	n	
Total		305901		

Table 2: Data Files Description

2.2 General Schema Explanation

The original SLISR relation schema, used in this work to tackle taska, consists of 7 relations (8, including relation **taska**) describing the relationship between Swiss Life partners (i.e. particular customers), insurance contracts and components of insurance tariffs.

The relation **part/2** contains all partners. The relations **eadr/3** and **padr/4** relate them to their electronical and postal addresses, respectively. Each partner can play roles in certain insurance policies, or contracts, (relation **vvert/18**) which is shown by the relation **parrol/5**. If a partner is the insured person of the contract then tariff role records (relation **tfrol/8**) specify certain further properties. An insurance contract can have several components (e.g. the main contract part plus a component for insuring the case of the insured person becoming invalid) each of which (records in **tfkomp/26**) is related with a tariff role of the respective partners. In Appendix I pg. 26 there is a description of each relation and its arguments in the original SLISR data set. The data type for each argument is either NUMBER-(N) or STRING-(S). It should be observed that only one argument, kanton, from relation **padr** is of type STRING. A graphical overview of the original SLISR schema used to tackle taska, as well as the key attributes used to link the relations, are shown in Figure 1. Each arrow indicates that the relation which is pointed to references an attribute which is contained in the relation from which the arrow originated.

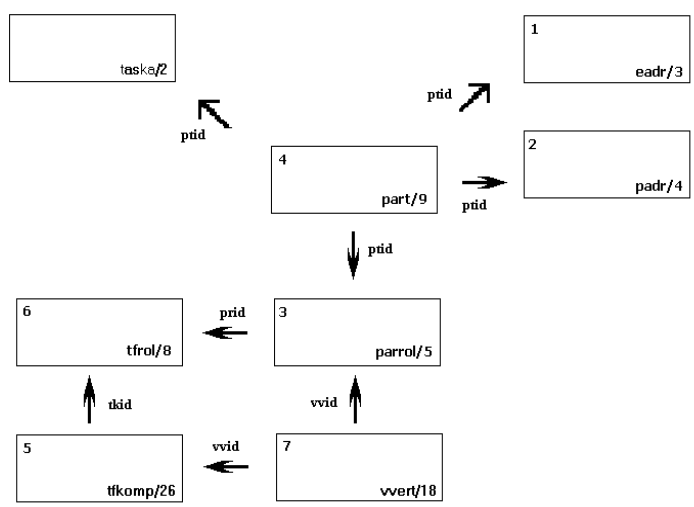

Figure 1: Original Schema Overview

2.3 Taska Description

The main objective of this work is related to supervised concept learning tackled with arbitrary Data Mining, Statistics or Machine Learning algorithms applied to the data set. The main focus lies on the preprocessing steps which make the algorithms applicable rather than on achieving high-precision mining results. Specifically we tackle taska which aims to find concept descriptions for subsets of partners. Data set taska — Table 2 pg. 2 — relates the respective identifiers ptid † with a value indicating concept membership.

Relation **taska/2** gives class membership wrt. concept. It relates partner identifier (ptid) with class membership 0 - No, 1 - Yes. The main objective is to try to find a concept description for those partners who have value 1 for attribute class.

3 Original Data Set Summary

In Appendix I pg. 26 the original data set supplied by SLISR is described in some detail.

First of all and before doing any preprocessing, the notation $X_{i,j}$, meaning attribute j of relation i, for relations and attributes in the original data set, was defined as shown in Table 3.

Attribute	1 eadr/3	2 padr/3	3 parrol/5	4 part/9	5 tfkomp/26	6 tfrol/8	7 vvert/19
N-ptid	$X_{1,1}$	$X_{2,1}$	$X_{3,2}$	$X_{4,1}$			
N-azart	$X_{1,2}$	$X_{2,2}$					
N-eatyp	$X_{1,3}$						
S-kanton		$X_{2,3}$					
N-gbeadmgeb		$X_{2,4}$					
N-prid			$X_{3,1}$			$X_{6,2}$	
N-vvid			$X_{3,3}$		$X_{5,2}$		$X_{7,1}$
N-prtyp			$X_{3,4}$				
N-prtypnr			$X_{3,5}$				
N-hhid				$X_{4,2}$			
N-epberuf				$X_{4,3}$			
N-eperweart				$X_{4,4}$			
N-epgebudat				$X_{4,5}$			
N-epsta				$X_{4,6}$			
N-eplnamecd				$X_{4,7}$			
N-epsexcd				$X_{4,8}$			
N-epzivstd				$X_{4,9}$			
N-tkid					$X_{5,1}$	$X_{6,3}$	
N-tknr					$X_{5,3}$		
N-tfid					$X_{5,4}$		
N-tkbeg					$X_{5,5}$		
N-tkend					$X_{5,6}$		
N-tkexkzwei					$X_{5,7}$		
N-tkstacd					$X_{5,8}$		
N-tkleist					$X_{5,9}$		
N-tkinkprl					$X_{5,10}$		
N-tkinkpre					$X_{5,11}$		
N-tktarpra					$X_{5,12}$		
N-tkuebvwsysp					$X_{5,13}$		
N-tkuebvwsysl					$X_{5,14}$		

continued on next page

†Note that the original data sets include an identifier - hhid - which is immaterial for taska.

continued from previous page

N-tkprfin					$X_{5,15}$		
N-tkdyncd					$X_{5,16}$		
N-tkausbcd					$X_{5,17}$		
N-tkrauv					$X_{5,18}$		
N-tksmed					$X_{5,19}$		
N-tkrizucd					$X_{5,20}$		
N-tklfleist					$X_{5,21}$		
N-tktodleista					$X_{5,22}$		
N-tkerlleista					$X_{5,23}$		
N-tkrenleista					$X_{5,24}$		
N-tkeuleista					$X_{5,25}$		
N-tkunfleista					$X_{5,26}$		
N-trid						$X_{6,1}$	
N-trteceinal						$X_{6,4}$	
N-truntcd						$X_{6,5}$	
N-trklauscd						$X_{6,6}$	
N-trstafcd						$X_{6,7}$	
N-trricd						$X_{6,8}$	
N-pdid							$X_{7,2}$
N-vvstacd							$X_{7,3}$
N-vvinkzwei							$X_{7,4}$
N-vvbeg							$X_{7,5}$
N-vvend							$X_{7,6}$
N-vvinkprl							$X_{7,7}$
N-vvinkpre							$X_{7,8}$
N-vvwae							$X_{7,9}$
N-vvversart							$X_{7,10}$
N-vvaendart							$X_{7,11}$
N-vvaendat							$X_{7,12}$
N-vvabvb							$X_{7,13}$
N-vvabga							$X_{7,14}$
N-vvstifcd							$X_{7,15}$
N-vvvorscd							$X_{7,16}$
N-vvbvgcd							$X_{7,17}$
N-vveucd							$X_{7,18}$

Table 3: $X_{i,j}$ Notation for Original Data Set

For example $X_{5,2}$ stands for the second attribute (attribute name vvid) of relation 5 (tfkomp/26) while $X_{7,1}$ stands for the first attribute (attribute name vvid) of relation 7 (vvert/18). Thus, attributes that are common to two or more relations can be clearly noticed and initial links can be made between the relations. In doing this, information can be quickly back referenced to the original data set relations/attributes. Results obtained using different techniques can easily be compared and analysed, allowing decisions to be quickly made as to how to continue attacking the problem.

As shown in Table 2 pg. 2 there are missing attribute values in the data set. Table 4 shows, for each attribute, the total number of missing attribute values. In general, it was observed that missing values are clustered in the same record.

Attribute	1 eadr/3	2 padr/3	3 parrol/5	4 part/9	5 tfkomp/26	6 tfrol/8	7 vvert/19
N-ptid	0 (0%)	0 (0%)	0 (0%)	0 (0%)	-	-	-
N-azart	0 (0%)	0 (0%)	-	-	-	-	-
N-eatyp	0 (0%)	-	-	-	-	-	-
S-kanton	-	7468 (42%)	-	-	-	-	-
N-gbeadmgeb	-	7054 (39%)	-	-	-	-	-

continued on next page

continued from previous page

N-prid	-	-	0 (0%)	-	-	0 (0%)	-
N-vvid	-	-	0 (0%)	-	0 (0%)	-	0 (0%)
N-prtyp	-	-	0 (0%)	-	-	-	-
N-prtypnr	-	-	0 (0%)	-	-	-	-
N-hhid	-	-	-	4322 (25%)	-	-	-
N-epberuf	-	-	-	624 (4%)	-	-	-
N-eperweart	-	-	-	624 (4%)	-	-	-
N-epgebudat	-	-	-	624 (4%)	-	-	-
N-epsta	-	-	-	624 (4%)	-	-	-
N-eplnamecd	-	-	-	624 (4%)	-	-	-
N-epsexcd	-	-	-	624 (4%)	-	-	-
N-epzivstd	-	-	-	624 (4%)	-	-	-
N-tkid	-	-	-	-	0 (0%)	0 (0%)	-
N-tknr	-	-	-	-	0 (0%)	-	-
N-tfid	-	-	-	-	0 (0%)	-	-
N-tkbeg	-	-	-	-	0 (0%)	-	-
N-tkend	-	-	-	-	1191 (2%)	-	-
N-tkexkzwei	-	-	-	-	45695 (62%)	-	-
N-tkstacd	-	-	-	-	46813 (64%)	-	-
N-tkleist	-	-	-	-	0 (0%)	-	-
N-tkinkprl	-	-	-	-	0 (0%)	-	-
N-tkinkpre	-	-	-	-	45762 (62%)	-	-
N-tktarpra	-	-	-	-	46813 (64%)	-	-
N-tkuebvwsysp	-	-	-	-	0 (0%)	-	-
N-tkuebvwsysl	-	-	-	-	0 (0%)	-	-
N-tkprfin	-	-	-	-	0 (0%)	-	-
N-tkdyncd	-	-	-	-	0 (0%)	-	-
N-tkausbcd	-	-	-	-	26689 (11%)	-	-
N-tkrauv	-	-	-	-	73051 (74%)	-	-
N-tksmed	-	-	-	-	72527 (73%)	-	-
N-tkrizucd	-	-	-	-	0 (0%)	-	-
N-tklfleist	-	-	-	-	73502(75%)	-	-
N-tktodleista	-	-	-	-	0 (0%)	-	-
N-tkerlleista	-	-	-	-	0 (0%)	-	-
N-tkrenleista	-	-	-	-	45695 (62%)	-	-
N-tkeuleista	-	-	-	-	0 (0%)	-	-
N-tkunfleista	-	-	-	-	0 (0%)	-	-
N-trid	-	-	-	-	-	0 (0%)	-
N-trteceinal	-	-	-	-	-	0 (0%)	-
N-truntcd	-	-	-	-	-	0 (0%)	-
N-trklauscd	-	-	-	-	-	0 (0%)	-
N-trstafcd	-	-	-	-	-	45695 (62%)	-
N-trricd	-	-	-	-	-	0 (0%)	-
N-pdid	-	-	-	-	-	-	620 (2%)
N-vvstacd	-	-	-	-	-	-	0 (0%)
N-vvinkzwei	-	-	-	-	-	-	0 (0%)
N-vvbeg	-	-	-	-	-	-	316 (1%)
N-vvend	-	-	-	-	-	-	940 (3%)
N-vvinkprl	-	-	-	-	-	-	0 (0%)
N-vvinkpre	-	-	-	-	-	-	618 (2%)
N-vvwae	-	-	-	-	-	-	0 (0%)
N-vvversart	-	-	-	-	-	-	0 (0%)
N-vvaendart	-	-	-	-	-	-	0 (0%)
N-vvaendat	-	-	-	-	-	-	472 (1%)
N-vvabvb	-	-	-	-	-	-	14521 (42%)
N-vvabga	-	-	-	-	-	-	73 (0%)
N-vvstifcd	-	-	-	-	-	-	0 (0%)
N-vvvorscd	-	-	-	-	-	-	0 (0%)
N-vvbvgcd	-	-	-	-	-	-	0 (0%)
N-vveucd	-	-	-	-	-	-	0 (0%)

Table 4: Number of Missing Values for each Attribute

4 Initial Relevant Data for Task

As described in Section 2.3 pg 4, taska is related to find a concept description for those partners who have value 1 for the attribute *class*. However, the original data already described is a subset of a more general scheme from the data warehouse. For this reason, relation **taska/2** also contains instances where its second argument is instantiated with 2, meaning not applicable. These cases have no meaning for taska class membership, where only class values 0-no and 1-yes are relevant. In order to reduce the quantity of information searched for, as well as clear out the unwanted data, the database was pre-processed. We shall refer to this pre-processing as steps 1 and 2. The first step refers to the cleaning of whole records while the second refers to reducing the number of attributes in the records.

4.1 Step 1

First the relation **taska/2** was cleaned of its records which contained the attribute $class =$ 2 as well as the ones related to them.

The relation **padr/4** was also cleaned of all its records in which the attributes kanton and gbeadmgeb both contained the value `$null$`. Both relations have an attribute ptid, which *links* them to a corresponding record in the relation **part/9**. It should be observed that there is a corresponding record in **taska/2** and **padr/4** for every distinct ptid in **part**. For each ptid eliminated from the two relations mentioned above, a corresponding record was eliminated from **part/9**. This caused a reduction of close to 50% in the number of partners to study.

Every partner can play a number of roles in a contract, this relationship is shown in the relation **parrol/5**. The roles related to the eliminated partners were also done away with as they only had to do with those particular partners. This caused some contracts to no longer be necessary. These records in the relation **vvert/18** were also cleaned so as to maintain only the information pertinent to the partners being studied. Tariff components can belong to certain contracts and are represented in relation **tfkomp/26**. With the elimination of their corresponding contracts, records for components were erased, making it necessary to clean the relation for tariff roles, **tfrol/8**. The roles containing a tkid (identifier of the corresponding component) were deleted. These roles also have to do with the partner roles in parrol, thus for each partner role deleted, the corresponding tariff role was also eliminated. This caused only a few more records to be cleaned from **tfrol/8**, as there was already an indirect connection between the two relations through the relations **vvert/18** and **tfkomp/26** (see Figure 1 pg 3).

Finally, relation **eadr/3** was left out since it does not hold interesting information as well as having very few records related to the problem at hand. Table 5 describes the number of records left in each relation after having performed this "cleaning" of the original data set — step 1.

Data File	No. of Records	Missing Values	Description
1.-eadr	NA	NA	Electronical Address
2.-padr	9054	y	Postal Address
3.-parrol	47932	n	Partner Roles
4.-part	8827	y	Partners Involved in Insurance Contracts
5.-tfkomp	37767	y	Tariff Components
6.-tfrol	32987	y	Tariff Roles
7.-vvert	17420	y	Life Insurance Policy
taska	8827	n	
Total	162813		

Table 5: Data File Description After Step 1

4.2 Step 2

The next step for "cleaning" the database was an analysis of the attributes which had a large number of *missing values*, represented by **`$null$`** in the dataset. The number of missing values for each attribute after step 1 (see Table 5) are shown in Table 6

Attribute	1 eadr/3	2 padr/3	3 parrol/5	4 part/9	5 tfkomp/26	6 tfrol/8	7 vvert/19
N-ptid	0 (0%)	0 (0%)	0 (0%)	0 (0%)	-	-	-
N-azart	0 (0%)	0 (0%)	-	-	-	-	-
N-eatyp	0 (0%)	-	-	-	-	-	-
S-kanton	-	336 (4%)	-	-	-	-	-
N-gbeadmgeb	-	0 (0%)	-	-	-	-	-
N-prid	-	-	0 (0%)	-	-	0 (0%)	-
N-vvid	-	-	0 (0%)	-	0 (0%)	-	0 (0%)
N-prtyp	-	-	0 (0%)	-	-	-	-
N-prtypnr	-	-	0 (0%)	-	-	-	-
N-hhid	-	-	-	NA	-	-	-
N-epberuf	-	-	-	255 (3%)	-	-	-
N-eperweart	-	-	-	255 (3%)	-	-	-
N-epgebudat	-	-	-	267 (3%)	-	-	-
N-epsta	-	-	-	255 (3%)	-	-	-
N-eplnamecd	-	-	-	255 (3%)	-	-	-
N-epsexcd	-	-	-	255 (3%)	-	-	-
N-epzivstd	-	-	-	255 (3%)	-	-	-
N-tkid	-	-	-	-	0 (0%)	0 (0%)	-
N-tknr	-	-	-	-	0 (0%)	-	-
N-tfid	-	-	-	-	0 (0%)	-	-
N-tkbeg	-	-	-	-	0 (0%)	-	-
N-tkend	-	-	-	-	459 (1%)	-	-
N-tkexkzwei	-	-	-	-	22735 (60%)	-	-
N-tkstacd	-	-	-	-	23154 (61%)	-	-
N-tkleist	-	-	-	-	0 (0%)	-	-
N-tkinkprl	-	-	-	-	0 (0%)	-	-
N-tkinkpre	-	-	-	-	22750 (60%)	-	-
N-tktarpra	-	-	-	-	23154 (61%)	-	-
N-tkuebvwsysp	-	-	-	-	0 (0%)	-	-
N-tkuebvwsysl	-	-	-	-	0 (0%)	-	-
N-tkprfin	-	-	-	-	0 (0%)	-	-
N-tkdyncd	-	-	-	-	0 (0%)	-	-
N-tkausbcd	-	-	-	-	14613 (39%)	-	-
N-tkrauv	-	-	-	-	37557 (99%)	-	-
N-tksmed	-	-	-	-	37310 (99%)	-	-

continued on next page

continued from previous page

N-tkrizucd	-	-	-	-	0 (0%)	-	-
N-tklfleist	-	-	-	-	37767(100%)	-	-
N-tktodleista	-	-	-	-	0 (0%)	-	-
N-tkerlleista	-	-	-	-	0 (0%)	-	-
N-tkrenleista	-	-	-	-	22735 (60%)	-	-
N-tkeuleista	-	-	-	-	0 (0%)	-	-
N-tkunfleista	-	-	-	-	0 (0%)	-	-
N-trid	-	-	-	-	-	0 (0%)	-
N-trteceinal	-	-	-	-	-	0 (0%)	-
N-truntcd	-	-	-	-	-	0 (0%)	-
N-trklauscd	-	-	-	-	-	0 (0%)	-
N-trstafcd	-	-	-	-	-	19541 (59%)	-
N-trricd	-	-	-	-	-	0 (0%)	-
N-pdid	-	-	-	-	-	-	NA
N-vvstacd	-	-	-	-	-	-	0 (0%)
N-vvinkzwei	-	-	-	-	-	-	0 (0%)
N-vvbeg	-	-	-	-	-	-	163 (1%)
N-vvend	-	-	-	-	-	-	402 (2%)
N-vvinkprl	-	-	-	-	-	-	0 (0%)
N-vvinkpre	-	-	-	-	-	-	281 (2%)
N-vvwae	-	-	-	-	-	-	0 (0%)
N-vvversart	-	-	-	-	-	-	0 (0%)
N-vvaendart	-	-	-	-	-	-	0 (0%)
N-vvaendat	-	-	-	-	-	-	230 (1%)
N-vvabvb	-	-	-	-	-	-	7054 (40%)
N-vvabga	-	-	-	-	-	-	38 (0%)
N-vvstifcd	-	-	-	-	-	-	0 (0%)
N-vvvorscd	-	-	-	-	-	-	0 (0%)
N-vvbvgcd	-	-	-	-	-	-	0 (0%)
N-vveucd	-	-	-	-	-	-	0 (0%)

Table 6: Number of Missing Values for each Attribute After Step 1

The attributes tkrauv, tksmed and tklfeist of relation **tfkomp** were eliminated as there were almost no values left after step 1(99-100% *missing*). The attributes tkexkzwei, tkstacd, tkinkpre, tktarpra and tkrenleista, from relation **tfkomp** as well as the attribute trstafcd from **tfrol**, were also gotten rid of as they had more than 50% of their values *missing*. Besides these attributes, three others were excluded as they related tables which were not to be considered or were not supplied by SLISR. They are

- hhid, from **part**, which refers to another relation **hhold** not considered for taska
- tfid, from a relation **tariff** not used by **taska** but referenced in **tfkomp**, and
- pdid from the relation **prod** (referenced in **vvert**).

It should be observed that relations **tariff** and **prod** were not made available by the insurance company.

4.3 Clean-up Summary

Summarizing the results of this *clean-up*:

- **taska** had one attribute removed (class)

- **part** had one one attribute removed (hhid)
- **vvert** had one attribute removed (pdid)
- **tfrol** had one attribute removed (trstafcd)
- **tfkomp** had eight attributes removed (tkrauv, tksmed, tklfeist, tkexkzwei, tkstacd, tkinkpre, tktarpra and tkrenleista)

The final reduced scheme with the new number of attributes for each relation is shown in Figure 2.

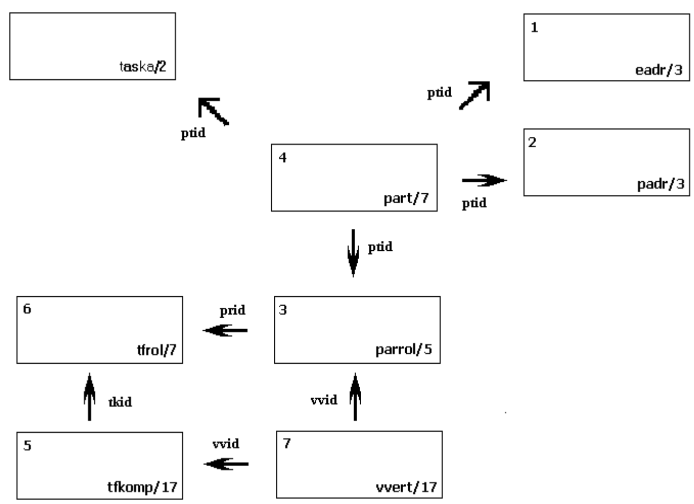

Figure 2: Reduced Schema

From this point on we decided to try to extract knowledge from this data using two different approaches

1. Propositional Approach: using C4.5, C4.5-Rules and CN2 as inducers
2. Relational Approach: using the ILP system *GOLEM* as the inducer

The following sections describe the actions taken in each of these approaches.

5 Propositional Approach

Once the data was prepared as described in the previous sections, three propositional Machine Learning — ML — algorithms CN2 [2], C4.5 and C4.5-rules [12] were used in order to try to extract some knowledge from the data.

These algorithms are non-incremental and start with large sets of cases belonging to known classes, i.e. supervised learning. The cases, which can be described by any mixture of nominal and numeric attributes, are scrutinized for patterns that allow the classes to be reliably discriminated. These patterns are then expressed as models, in the form of decision trees (C4.5) or sets of **if-then** rules (CN2 and C4.5-rules), that can be used to classify new cases, with emphasis on making the models understandable as well as accurate.

These algorithms learn propositional rules (C4.5 learns decision trees which can be transformed into decision rules). In propositional rules there are no variables or quantifiers, and no relations among example components.

5.1 Attribute-Value Format

Propositional inducers may differ greatly, but they share a common format to describe the cases. The common format is the attribute-value (or spreadsheet) format. Table 7 illustrates this organization where a row i refers to the i-th example or case and column entries v_{ij} refer to the value of the j-th feature f_j of example i. The column labelled *class* refers to the label or classification of that example.

f_1	f_2	...	f_k	*class*
v_{11}	v_{12}	...	v_{1k}	c_1
v_{21}	v_{22}	...	v_{2k}	c_2
...	...	...	...	...
v_{n1}	v_{n2}	...	v_{nk}	c_n

Table 7: Attribute-Value Format

5.2 Special Care for Removing Relationships

In order to apply propositional inducers for **taska** it is first necessary to transform the data to the attribute-value format. Special care should be taken during this process. Depending on the data, it is not always "fair" to link all datasets together to obtain one huge attribute-value format dataset to feed a propositional learning algorithm. This problem appears when there are relationships N-to-M, with $N > 1$ and $M > 1$.

The example in Figure 3 illustrates this point. Considering two relations — **relation1/5** and **relation2/3** — where the first column of **relation1/5** refers to the primary key and the last one to its class, then if we transform that data to the attribute-value format (*i.e.* linking **relation1/5** and **relation2/3** through the primary key), we obtain **relation12/7**.

Several issues should be considered. First of all

what is the meaning of inexistent values shown as ∗?

Machine Learning algorithms should be able to process not only known values but also they should handle values such as:

- unknown;

A	x1	y1	z1	Y
B	x2	y2	z2	Y
C	x3	y3	z3	Y
D	x4	y4	z4	N
E	x5	y5	z5	N

relation1/5

A	w1	v1
A	w2	v2
C	w3	v3
C	w4	v4
E	w5	v5

relation2/3

A	x1	y1	z1	w1	v1	Y
A	x1	y1	z1	w2	v2	Y
B	x2	y2	z2	*	*	Y
C	x3	y3	z3	w3	v3	Y
C	x3	y3	z3	w4	v4	Y
D	x4	y4	z4	*	*	N
E	x5	y5	z5	w5	v5	N

relation12/7

Figure 3: An Example of Removing a Relationship

- don't care and
- not applicable

all of them having a different meaning. In the example shown in Figure 3 *what meaning should be given to values with an ∗?* In general, the answer is domain dependent.

Another problem is related with the distribution of the examples by attribute as well as class. Observe that the class distribution in **relation1/5** is 60% for class **Y** and 40% for class **N**, but this proportion — as well as the attribute distribution — is not the same in **relation12/7**. Unless the cardinality from **relation1/5** to **relation2/3** is 1-to-N for a fixed N and all records in **relation1/5** also appear in **relation2/3** in equal quantities, we shall have this sort of problems. It can be seen that a straightforward transformation to the attribute-value format does not always preserve — from the point of view of the *bias* of the propositional learning algorithm — the same knowledge as that which is contained in the original datasets.

For these reasons, in the exploratory steps described in the next section, we only consider individual datasets having relationships with cardinality 1-to-1, in order to extract some propositional knowledge from the data.

5.3 Linking Individual Datasets to Their Class

As we mentioned earlier, in this case datasets are considered independently from each other. Relations **padr/4**, **part/9**, **parrol/5** and **tfrol/26** were considered separately, using all features. The attribute value related to ptid (responsible to link **taska** to other relations) allows to uniquely define the class of each record in these relations. The other relations were not used.

Relation	#Instances	Feature	#Distinct Values	Class 0	Class 1
padr+*class*	9054	azart	4 discrete	28.82%	71.18%
		kanton	29 discrete		
		gbeadmgeb	544 discrete		
part+*class*	8827	epberuf	85 discrete	28.40%	71.60%
		eperweart	5 discrete		
		epgebudat	88 continuous		
		epsta	5 discrete		
		eplnamecd	4 discrete		
		epsexcd	3 discrete		
		epzivstd	7 discrete		
parrol+*class*	47932	prtyp	9 discrete	6.16%	93.84%
		prtypnr	5 discrete		
tfrol+*class*	28627	trteceinal	78 continuous	0.07%	99.93%
		truntcd	3 discrete		
		trklauscd	2 discrete		
		trstatfcd	3 discrete		
		trricd	2 discrete		

Table 8: Initial Attribute-Value Tables Description

It can be observed that the number of instances (or records) in relations **tfrol** (32987) and tfrol+*class* (28627) are different. This is because the linking of **tfrol** to *class* was done indirectly through the relation **parrol**. Table 8 summarizes the number of attributes in each of these newly constructed attribute-value tables, number of examples (#Instances), features used, as well as the distribution of classes 0 and 1.

5.4 Additional Clean-Up

From Table 8 it is possible to observe the following:

1. The discrete feature gbeadmgeb from table padr+*class* has a high number of distinct values (544). The meaning of feature gbeadmgeb — see Appendix I, pg. 26 — is related to a postal address region in Switzerland. This sort of information can be considered as too specific (quite typical in real data bases), since the discrete feature kanton from the same relation, having 29 distinct values, refers to the district inside Switzerland. With this in mind, feature gbeadmgeb was not considered further.

2. The discrete feature epberuf from table part+*class* has also a high number of distinct values (85). The meaning of this feature is related to profession code. In fact, if the meaning of each of those codes were known, it would be possible to group them semanticaly at a higher level. As nothing is known about them, we decided to drop feature epberuf although this information, albeit at a higher level of abstraction, could be important.

3. Attribute-Value table tfrol+*class* consists mainly of positive examples since there are very few class 0 instances (less than 0.1%). As learning in the presence of only one class is not practically possible the entire table tfrol+*class* was not considered further.

After these modifications, the initial attribute-value tables reduce to the ones shown in Table 9.

Relation	#Instances	Feature	#Distinct Values	Class 0	Class 1
padr+*class*	9054	azart	4 discrete	28.82%	71.18%
		kanton	29 discrete		
part+*class*	8827	eperweart	5 discrete	28.40%	71.60%
		epgebudat	88 continuous		
		epsta	5 discrete		
		eplnamecd	4 discrete		
		epsexcd	3 discrete		
		epzivstd	7 discrete		
parrol+*class*	47932	prtyp	9 discrete	6.16%	93.84%
		prtypnr	5 discrete		

Table 9: Final Attribute-Value Tables Description

5.5 Inducers & Preliminary Accuracies

As stated earlier, in the propositional approach the C4.5, C4.5-rules [12] and CN2 [2] inductive algorithms were used. The algorithms were executed using $\mathcal{MLC}$++ (Machine Learning Library in *C*++) [3, 4] a software package developed at Stanford University which contains various machine learning algorithms, offering the advantage of not having to change the format of the input data when using another algorithm. It also provides standardized methods for running experiments using these algorithms.

All three algorithms were applied to the three sets of examples described in Table 9. Since, at Step 1, the "not applicable" values were removed and there is no additional information about the nature of **$null$**, special care was taken to replace these values in the original datasets with **?**, a special value that means "unknown" to the propositional ML algorithms used in this work.

Accuracies obtained using 10-fold stratified cross validation for each attribute-value table and inducer are shown in Table 10, along with the standard deviation. It can be observed that for all inducers accuracy is greater than the majority class, meaning that some learning has taken place. Furthermore, as expected, the accuracy obtained by C4.5-rules is equivalent or greater than the one obtained by C4.5.

5.6 Results

Table 11 shows the total number of leaves (or number of disjunct decision rules) for C4.5 as well as the number of rules generated by C4.5-rules and CN2. It should be observed

Relation	Class 0	Class 1	C4.5	C4.5-rules	CN2
padr+*class*	28.82%	71.18%	74.00%±0.15%	73.97%±0.15%	73.87%±0.21%
part+*class*	28.40%	71.60%	91.13%±0.40%	94.13%±0.33%	93.90%±0.41%
parrol+*class*	6.16%	93.84%	97.92%±0.05%	97.92%±0.05%	97.92%±0.05%

Table 10: Class Distribution and Accuracies using 10-fold Stratified Cross Validation

that rules generated by C4.5-rules and CN2 are not necessarily disjunct rules.

Relation	C4.5	C4.5-rules	CN2
padr+*class*	5	2	58
part+*class*	22	13	183
parrol+*class*	5	4	12

Table 11: Number of Leaves and Rules Generated

As expected, the number of rules generated by C4.5-rules is less than the ones representing the C4.5 decision tree. This is due to the fact that C4.5-rules uses a variant of the *rule post-pruning* method on the decision tree generated by C4.5 for finding high accuracy hypotheses. This means that in the worst case, if no prunning is possible, the number of rules generated by both algorithms would be equal.

Tables 12, 13 and 14 show some initial results for each one of the three relations considered, where only simple and significant rules are presented. The Coverage column indicates the number of instances covered for [class 0, class 1]. It should be observed that non integer results related to rule coverage indicate the presence of unknown values. Thus, for instance, the first C4.5 rule in Table 12

$$\textbf{if } \text{azart in } \{2, 3, 6\} \textbf{ then } \text{class} = 0$$

correctly "classifies" 253.6 instances as class 0 and "misclassifies" 1.4 instances as class 1. On the other hand, the last CN2 rule in Table 12

$$\textbf{if } \text{azart} = 5 \textbf{ and } \text{kanton} = \text{BE} \textbf{ then } \text{class} = 1$$

"misclassifies" 434.1 instances as class 0 and correctly "classifies" 1215.3 instances as class 1.

Besides individual rule coverage, two additional useful measures (Support and Confidence) are also shown in the tables (intermediate results are also shown for better understanding). In rules like

$$\textbf{if } X \textbf{ then } Y \text{ where } X \cap Y = \emptyset$$

Support and *Confidence* can be calculated as indicated in Equations 1 and 2.

$$\text{Support} = \frac{\#\text{Instances with } X \text{ and } Y}{\text{Total } \#\text{Instances}} \tag{1}$$

and

$$\text{Confidence} = \frac{\#\text{Instances with } X \text{ and } Y}{\#\text{Instances with } X} \qquad (2)$$

These two measures allow a better understanding of generated rules, mainly to the final user. For instance, it can be observed from Table 12 that, although the first C4.5 rule has Confidence = 99.45%, meaning that the proportion of erroneously covered examples by this rule is very low, its Support is poor (2.80%), meaning that a small proportion of the complete data base is correctly covered by this rule. On the other hand, the second C4.5 rule has Confidence = 73.39% but Support = 69.49% which, generally, seems to be a better or more interesting rule than the first one.

Inducer	Rule	Coverage	Support	Confidence
C4.5	**if** azart in {2, 3, 6} **then** class = 0	[253.6, 1.4]	253.6/9054 2.80%	253.6/255 99.45%
C4.5	**if** azart in {1, 5} **then** class = 1	[2281.8, 6292]	6292/9054 69.49%	6292/8573.8 73.39%
C4.5-rules	**if** azart = 6 **then** class = 0	[255, 0]	255/9054 2.82%	255/255 100%
CN2	**if** azart = 5 **and** kanton = ZH **then** class = 1	[272.1, 1012.3]	1012.3/9054 11.18%	1012.3/1284.4 78.82%
CN2	**if** azart = 5 **and** kanton = BE **then** class = 1	[434.1, 1215.3]	1215.5/9054 13.43%	1215.3/1649.4 73.68%

Table 12: Some of the Rules Learned for padr+*class*

Another point to consider when analyzing and interpreting these rules individualy, is how the examples are covered by them. For example, C4.5 generates a decision tree, so rules that represent that knowledge are disjunct. Rules generated by CN2 are unordered and not disjunct. This means that two (or more) different rules may cover the same example. On the other hand, rules generated by C4.5-rules are unordered inside its class subset but are ordered for each (outside) class subset.

Inducer	Rule	Coverage	Support	Confidence
C4.5	**if** epgebudat > 1975 **and** eperweart in {0, 3} **then** class = 0	[1196.8, 1.4]	1196.8/8827 13.56%	1196.8/1198.2 99.88%
C4.5	**if** epgebudat ≤ 1975 **and** eplnamecd = 1 **and** epzivstd in {0, 1, 2, 5} **and** eperweart = 0 **then** class = 0	[382, 24.8]	382/8827 4.33%	382/406.8 93.90%
C4.5	**if** epgebudat ≤ 1975 **and** eplnamecd in {0, 2} **and** eperweart in {1, 2, 3} **then** class = 1	[388.1, 5480.3]	5480.3/8827 62.09%	5480.3/5868.4 93.39%

continued on next page

continued from previous page

C4.5	**if** epgebudat $\leq$ 1969 **and** eplnamecd in $\{0, 2\}$ **and** eperweart = 0 **and** epsexcd = 1 **then** class = 1	$[78.2, 260.1]$	260.1/8827 2.95%	260.1/338.3 76.88%
C4.5-rules	**if** epgebudat > 1976 **then** class = 0	$[1145, 6]$	1145/8827 12.97%	1145/1151 99.48%
CN2	**if** eperweart = 0 **and** epgebudat > 1974.5 **then** class = 0	$[1191.8, 0]$	1191.8/8827 13.50%	1191.8/1191.8 100.00%
CN2	**if** eperweart = 0 **and** epgebudat > 1971.5 **and** epzivstd = 0 **then** class = 0	$[463.6, 0]$	463.6/8827 5.25%	463.6/463.6 100.00%
CN2	**if** epgebudat > 1977.5 **and** epsexcd = 2 **then** class = 0	$[656.5, 0]$	656.6/8827 7.44%	656.5/656.5 100.00%
CN2	**if** epgebudat > 1977.5 **then** class = 0	$[1336, 2]$	1336/8827 15.14%	1336/1338 99.85%
CN2	**if** eperweart = 1 **and** 1934.5 $\leq$ epgebudat $\leq$ 1942.5 **and** epsexcd = 1 **and** epzivstd = 2 **then** class = 1	$[3, 490]$	490/8827 5.55%	490/493 99.39%

Table 13: Some of the Learned Rules for part+*class*

5.7 Analysis of the Results

- Results in Table 12 — padr+*class* — shows that for this case it was not possible to discover good rules for any of the algorithms. It can also be observed from Table 10 pg. 10 that the learning accuracy is the lowest for this relation, being only about 2% higher than the majority class.

- Analyzing Table 13 — part+*class* — it can be seen that attributes epgebudat (year of birth) and eperweart (professional status) are good discriminants. Furthermore, young people (*i.e.* born arround or after 1975) seem to be classified as *class* = 0. There is also a C4.5 rule that covers more than 5000 examples (the third C4.5 rule on Table 13). Still, it is hard to interpret since we do not know the meaning of the possible values for the attributes eplnamecd and eperweart.

- From Table 14 — parrol+*class* — it can be seen that there are some interesting rules. For example, the third C4.5 rule, which covers the majority of examples with an error of 3.24%. Also, the third rule generated by CN2 which covers 31.36% of the examples shows the value of prtyp = 11 is a good discriminant for *class* = 1.

Inducer	Rule	Coverage	Support	Confidence
C4.5	**if** prtypnr **in** {0, 1} **and** prtyp **in** {6, 14, 15, 17, 18} **then** class = 0	[520, 10]	520/47932 1.08%	520/530 98.11%
C4.5	**if** prtypnr **in** {2, 3, 4} **and** prtyp = 12 **then** class = 0	[1465, 20]	1465/47932 3.06%	1465/1485 98.65%
C4.5	**if** prtypnr in {0, 1} **and** prtyp **in** {9, 10, 11, 12} **then** class = 1	[968, 44939]	44939/47932 93.76%	44939/45907 97.89%
C4.5-rules	**if** prtyp = 11 **then** class = 1	[0, 15033]	15033/47932 31.36%	15033/15033 100.00%
C4.5-rules	**if** prtyp in {9, 10, 11, 12} **and** prtypnr in {0, 1} **then** class = 1	[968, 29916]	29916/47932 62.41%	29916/30884 96.87%
CN2	**if** prtyp = 14 **then** class = 0	[438, 2]	438/47932 0.91%	438/440 99.55%
CN2	**if** prtyp = 12 **and** prtypnr = 2 **then** class = 0	[1442, 20]	1442/47932 3.01%	1442/1462 98.63%
CN2	**if** prtyp = 11 **then** class = 1	[0, 15033]	15033/47932 31.36%	15033/15033 100.00%
CN2	**if** prtyp = 9 **then** class = 1	[201, 14756]	14756/47932 30.79%	14756/14957 98.66%
CN2	**if** prtypnr = 1 **then** class = 1	[766, 30034]	30034/47932 62.66%	30034/30800 97.51%

Table 14: Some of the Rules Learned for parrol+*class*

6 Relational Approach

Standard Machine Learning algorithms using propositional logic representation formalism can be directly used in rare cases, i.e. when the data is stored in a single relation table. When the data is described using several interrelated tables, as in this work, it seems that the Data Mining process would benefit from systems using the first-order logic representation formalism. Inductive Logic Programming — ILP — a recently emerged subfield of ML is appropriate for this task [11]. ILP integrates the techniques already available and established for logic programming in a framework of learning, aiming to induce first-order logic programs from examples, using background knowledge [5, 6, 7, 8, 9].

In general, ILP algorithms can exploit the data directly, or after very few ajustments, as it comes from the database. Still, it should be kept in mind that this greater flexibility comes at a price. Learning within the first-order framework is much harder than learning in the propositional framework. Within the first-order framework the search space is infinitely larger and the underlying inference mechanism is much more complex than in the propositional framework. Sometimes, due to this fact, ILP algorithms may perform worse on learning tasks where the learned concept can be conveniently expressed using a propositional description.

In short, ILP algorithms should be used when the concept that is being learned can be conveniently expressed using a first-order logic representation, and also is very difficult to describe using a propositional representation.

In our relational approach for extracting knowledge from the data base, we use the inductive logic programming system *GOLEM* which restricts the model of the background knowledge as well as the hypotheses language to Horn clause logic programming, such as Prolog.

6.1 The *GOLEM* Inducer

GOLEM constructs logic programs — restricted to Horn clauses — from positive and negative examples using information given by the background knowledge. In *GOLEM*, examples and background knowledge are represented as ground atomic facts. The algorithm has a good level of efficiency and is able to learn predicates (rules) such as list reverse, quicksort, member, integer multiply and other real-world applications. Rules can be used to classify new examples. The syntax of atoms and rules (or clauses) — see Table 15 — follows that of Edinburgh Prolog (without operator definition and queries) [10].

structure	format	comments
ground atom	$predicate(arg_1, arg_2, \ldots, arg_n).$	arg_1, arg_2, ..., arg_n are terms, possibly with function symbols and no variables
clause (or rule)	$Head : -A_1, A_2, \ldots, A_n$	$Head$, A_1, A_2, ..., A_n are also atoms, but containing at least one variable

Table 15: Syntax of Atoms and Rules in *GOLEM*

GOLEM needs three types of input data files and produces (depending on the choice of the user) an output file as shown in Figure 4. The type and extension of the necessary

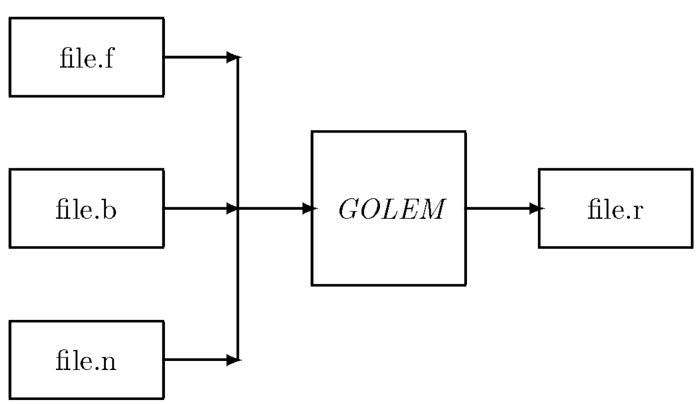

Figure 4: Data Files Used by *GOLEM*

files, as well as their contents are described in Table 16.

type	extention	comments
foreground	$.f$	set of positives examples. Each element is a ground atom implied by the target predicate
background	$.b$	set of examples which are believed to be relevant to the example in the foreground set It also contains the declarations part which defines the modes and determinations of each predicate
negative	$.n$	set of ground atoms that share the same predicate symbol as the elements of the foreground set, but are false with respect to the target predicate
rules	$.r$	set of input (optional) or output rules

Table 16: Types of Data Files used by *GOLEM*

GOLEM can be used in two different modes of operation, batch and interactive. It also provides commands which can be used for setting system parameters to control the learning process. These parameters, together with their default values are described in Table 17. In this work we have used different values for noise and different combinations of relations as background knowledge.

settings	default value	description
i	2	$i - determinacy$
j	2	$j - determinacy$
noise	0	limit of the permissible number of negative examples being misclassified by each rule
rlggsample	8	limit of a sample of example pairs used to construct clauses
testsample	500	limite of a sample of negative examples used to test clauses

Table 17: *Golem* System Settings

6.2 Results

The experiments made using the relational approach concentrate on learning the concept **taska**, using as background knowledge the relations directly related to it through the attribute ptid. These relations are: **part**, **padr** and **parrol**. The other relations not having a direct relationship with **taska** were not considered, since the chosen algorithm could not handle the memory requirements. In other words, to link the other relations with **taska**, it would have been necessary to include one or more of the relations that were directly related to it in the background, implying memory requirements not have available at the time of the experiments. According to the notation used to describe the original dataset — see Table 3 pg. 5 — the relations that were used have the following attributes:

padr	(ptid $\rightarrow X_{2,1}$, azart $\rightarrow X_{2,2}$, kanton $\rightarrow X_{2,3}$)
part	(ptid $\rightarrow X_{4,1}$, eperweart $\rightarrow X_{4,4}$, epgebudat $\rightarrow X_{4,5}$, epsta $\rightarrow X_{4,6}$, eplnamecd $\rightarrow X_{4,7}$, epsexcd $\rightarrow X_{4,8}$, epvizstd $\rightarrow X_{4,9}$)
parrol	(prid $\rightarrow X_{3,1}$, ptid $\rightarrow X_{3,2}$, vvid $\rightarrow X_{3,3}$, prtyp $\rightarrow X_{3,4}$, prtypnr $\rightarrow X_{3,5}$)

The relation **taska** has two attributes: ptid and class. It should be observed that the relations **part** and **padr** have, due to the cleaning process, different arity than in the original dataset.

GOLEM was run with settings rlggsample = 10 and testsample = 2507. Recall that the total number of examples in **taska** is 8827 — see Table 5 pg. 8 — where 6320 are positive and 2507 are negative. The noise level was initially fixed at 250 (approximately 10% of the total amount of negative examples). When the algorithm was not able to find any rules at this noise level, the amount of noise was raised either until the number reached 25% (625 examples) of the negative set or until rules were found.

The algorithm was run with a series of combinations of the relations as background knowledge. The negative and positive cover obtained by the induced logic programs, are shown in Table 18. Support and confidence measures for each rule set are also given based on Equations 1 and 2 presented in Section 5.6 pg. 5. The combination **padr** + **part** + **parrol** as background knowledge, was not considered since the default i-determinacy, which is set at 2 in *GOLEM*, was used.

Background Knowledge	Noise	Negative Cover	Positive Cover	Support	Confidence
parrol	625	-	-	-	-
padr	450	931	2902	32.9%	75.7%
padr + parrol	250	374	1207	13.7%	76.3%
padr + part	250	638	6132	69.5%	90.6%
part	250	408	5954	67.5%	93.6%
part + parrol	250	474	6313	71.5%	93.0%

Table 18: Negative and Positive Cover Varying Background Knowledge and Noise, along with Support and Confidence Measures

As can be seen, no rules were found when using only **parrol** as background knowledge. When using only **padr** as background knowledge it was necessary to increase the noise level until it reached 450 for *GOLEM* to produce some rules. The rules generated in each case are shown in Table 19.

Background K.	Rule	Propositional Form
padr	taska(A) :- padr(A,'5','BE').	**if** azart = '5'**and** kanton = 'BE' **then** class = 1
	taska(A) :- padr(A,'5','AG').	**if** azart = '5'**and** kanton = 'AG' **then** class = 1
	taska(A) :- padr(A,'5','ZH').	**if** azart = '5'**and** kanton = 'ZH' **then** class = 1

continued on next page

continued from previous page

padr + parrol	taska(A) :- padr(A,'5','BE').	**if** azart = '5'**and** kanton = 'BE' **then** class = 1
padr + part	taska(A) :- part(A,'2',B,'0',C,D,E). taska(A) :- part(A,'3',B,'0','0',C,D). taska(A) :- padr(A,'5',B), part(A,'1',C,'0','0',D,E). taska(A) :- padr(A,'5',B), part(A,'1',C,'0',D,E,F). taska(A) :- padr(A,'5',B), part(A,'3',C,'0','1','2',D). taska(A) :- padr(A,'5',B), part(A,C,D,'0','0','1','2'). taska(A) :- padr(A,'5',B), part(A,C,D,'0','0',C,E). taska(A) :- padr(A,'5',B), part(A,C,D,'0','0',E,F). part(G,H,D,'0',I,J,'5').	**if** eperweart = '2'**and** epsta = '0' **then** class = 1 **if** eperweart = '3'**and** epsta = '0' **and** eplnamecd = '0' **then** class = 1
part	taska(A) :- part(A,'1',B,'0','0',C,D). taska(A) :- part(A,'1',B,'0',C,D,E). taska(A) :- part(A,'2',B,'0','0',C,D). taska(A) :- part(A,'3',B,'0','1','2','3'). taska(A) :- part(A,'3',B,'0','0',C,D). taska(A) :- part(A,B,C,'0','0','1','2').	**if** eperweart = '1'**and** epsta = '0' **and** eplnamecd = '0' **then** class = 1 **if** eperweart = '1'**and** epsta = '0' **then** class = 1 **if** eperweart = '2'**and** epsta = '0' **and** eplnamecd = '0' **then** class = 1 **if** eperweart = '3'**and** epsta = '0' **and** eplnamecd = '1' **and** epsexcd = '2' **and** epvizstd = '3' **then** class = 1 **if** eperweart = '3'**and** epsta = '0' **and** eplnamecd = '0' **then** class = 1 **if** epsta = '0'**and** eplnamecd = '0' **and** epsexcd = '1' **and** epvizstd = '2' **then** class = 1
part + parrol	taska(A) :- part(A,'1',B,'0',C,D,E). taska(A) :- part(A,'2',B,'0','0','1',C). taska(A) :- part(A,'3',B,'0','0',C,D). taska(A) :- part(A,B,C,'0','0','1','2'). taska(A) :- part(A,B,C,'0','0',B,D). taska(A) :- part(A,B,C,'0','2','2','2'). taska(A) :- part(A,B,C,'0',D,'1','2'). taska(A) :- part(A,B,C,'0',D,B,E). taska(A) :- parrol(B,A,C,'11','2'). taska(A) :- parrol(B,A,C,'9','0').	**if** eperweart = '1'**and** epsta = '0' **then** class = 1 **if** eperweart = '2'**and** epsta = '0' **and** eplnamecd = '0' **and** epsexcd = '1' **then** class = 1 **if** eperweart = '3'**and** epsta = '0' **and** eplnamecd = '0' **then** class = 1 **if** epsta = '0'**and** eplnamecd = '0' **and** epsexcd = '1' **and** epvizstd = '2' **then** class = 1 **if** epsta = '0'**and** eplnamecd = '2' **and** epsexcd = '2' **and** epvizstd = '2' **then** class = 1 **if** epsta = '0'**and** epsexcd = '1' **and** epvizstd = '2' **then** class = 1 **if** prtyp = '11'**and** prtypnr = '2' **then** class = 1 **if** prtyp = '9' and prtypnr = '0' **then** class = 1

Table 19: Rules Generated with Different Sets of Background Knowledge

Many of the rules generated by *GOLEM* are simply propositional rules. For those rules the equivalent propositional form is also shown so as to compare them with the results from the propositional approach and analyze their similarities.

6.3 Analysis of the Results

Considering the positive coverage of each run — see Table 18 — it can be seen that the rules generated using **padr** as well as **padr** + **parrol** as background knowledge cover few positive examples. On the other hand, when using **part** + **parrol** as background knowledge, the best cover was obtained. In fact only 7 positive examples were not covered. This could probably be refined using manual techniques or other tools to obtain other rules for the uncovered examples.

The numbers for support and confidence seem to confirm the observations based on coverage. The rules obtained when using **padr** + **parrol** or just **padr** as background knowledge have considerably lower percentages than those of the other experiments.

In the case of all rules generated when using **part** as background knowledge (either by itself or in conjunction with another relation), there is a large occurrence of evidence for married people (epvizstd = '2'). These rules also have the best coverages, in comparison with **padr** and **parrol** by themselves, for example. Looking at the support and confidence measures for the rule sets, there is strong evidence to believe that **part** is a good source of background knowledge. Both measures are relatively high when **part** is used in the background. This should be relatively obvious as ptid is the main identifier for **part**, but maybe it can also indicate a starting point for obtaining more meaningful rules.

The last rule obtained when using **padr** + **part** as background presents an interesting relationship between instances or records of the relation **part**. Apparently, a partner who is a member of **taska** can also be related to some other partner who has the same year of birth (third attribute in the relation) and whose marital status is 'separated' (value '5' for attribute epzivstd), among other things.

When **padr** is used in the background, the rules generated always have the attribute azart = 5. Apparently this kind of address (azart is the attribute indicating the type of the address) occurs most often. This could be a clue as to where the majority of the contracts are made, say in a business, for example. If we had access to an expert, more interesting conclusions might appear.

7 Conclusion

For realistic data mining, the study of real-world data is essential. This kind of data has several peculiarities which make it necessary to consider details not otherwise noticed. In this work we have carried out a general review of the structure of the data, some measuring of its quality and identified some invalid and missing values. Afterwards the data was further refined to suit the input format requirements of the propositional and relational Machine Learning algorithms used to extract knowledge.

In all the steps described in the previous sections we have to face the problem of our poor insight into the underlying data content. Not only preparing the data, but also analyzing the results, are very important steps of the whole process. Without the aid of an expert to go over the results and supply background and commonsense knowledge, the whole process can become almost impossible.

It should be stressed that several of the reductions that we have made on the original databases can be qualify as being based on some superficial considerations. The main reason being not having an expert to give a qualified opinion and semantic information related to the data. Not only does the data end up having little real value, but it is also not possible, after extracting some rules, to answer the important question

Have we found something that is interesting and valid?

Still, in this case study, one interesting point to notice is that some of the rules generated by the different algorithms used overlap, i.e. some of the rules found by *GOLEM* are the same as some found by the propositional algorithms. Since the bias of the Machine Learning algorithms used in this work are quite different, this sort of rule should be analyzed further with the aid of a domain expert.

For example, the second and third rules found with **padr** as background knowledge for *GOLEM* are the same as those found by CN2 with padr+*class* as background. In fact, many of the rules from the relational approach can be re-written as propositional rules, simply by concentrating on the variables that are instantiated with some value.

Other rules can be seen as specializations of the propositional rules. When using **part** and **parrol** as background knowledge, two relational rules were found with only information about **parrol** in the conditions. These correspond to specializations of rules found by all three propositional algorithms using parrol+*class* as background, specifically those that consider only prtyp as a classifier for class 1.

This work could be continued along many different lines such as: varying the parameters of each algorithm to obtain rules that are both more concise and possibly interesting to the experts; reducing the database at different points to see the effects of these reductions on the knowledge that is obtained and others. Still, to do that, we consider it will be necessary to have a domain expert involved.

References

[1] Cabena, P.; Hadjinian, P.; Stadler, R.; Verhees, J.; Zanasi, A. *Discovering Data Mining: From Concepts to Implementation.* Prentice Hall, 1997.

[2] Clark, P.; Niblett, T. *The CN2 Induction Algorithm.* Machine Learning, Vol. 3, 1989, pp. 261-283.

[3] Félix, L.C.M.; Oliveira, R.S.; Doi, C.Y.; De Paula, M.F.; Romanato, M.J. *MLC++: Biblioteca de Aprendizado de Máquina em C++.* Technical Report 72, ISSN-0103-2569, ICMC, USP, São Paulo, Brazil, 1998, 55 pg.

[4] Kohavi, R.; John, G.; Long, R.; Manley, D.; Pfleger, K. *MLC++: A Machine Learning Library in C++.* In: Tool with Artificial Intelligence, IEEE Computer Society Press, 1994, pp. 740-743.

[5] Lavrač, N.; Džeroski, S. *Inductive Logic Programming: Techniques and Applications.* Ellis Horwood, 1994.

[6] Muggleton, S.H. *Inductive Acquisition of Expert Knowledge.* Turing Institute Press, Glasgow, 1990.

[7] Muggleton, S.H.; Feng, C. *Efficient Induction of Logic Programs.* TIRM-90-044, Turing Institute Press, Glasgow, October 1990.

[8] Muggleton, S.H. *Inductive Logic Programming.* New Generation Computing, Vol. 8, 1991, pp 295-318.

[9] Muggleton, S.H. *Inductive Logic Programming: Derivations, Successes and Shortcomings.* P.B. Bradzil (ed.), Lecture Notes in Artificial Intelligence 667, 1993.

[10] Nicoletti, M.C.; Monard, M.C. *Learning Horn Clauses Using the ILP System GOLEM.* Notas do ICMC, Série Computacão, No. 4, ISSN-0103-2577, Novembro 1993, 35 pg.

[11] Pompe, U.; Kononenko, I. *Efficient Induction and Effective use of First-Order Knowledge.* Applied Artificial Intelligence, 1998, pp 421-453.

[12] Quinlan, J.R. *C4.5 Programs for Machine Learning.* Morgan Kaufmann Publishers, San Mateo, California, 1993.

[13] Weiss, S.M.; Indurkhya, N. *Predictive Data Mining: A Practical Guide.* Morgan Kaufmann Publishers, 1998.

8 Appendix I - Schema Description

A description of each relation and its arguments in the original SLISR data set is presented here.

1-eadr/3 Electronical Address

1. (N) ptid: partner identifier (see attribute ptid of relation part/9)
2. (N) azart: type of address usage (e.g. private, business)
3. (N) eatyp: type of address (e.g. Fax, Phone etc.)

2-padr/4 Postal Address

1. (N) ptid: partner identifier (see attribute ptid of relation part/9)
2. (N) azart: type of address usage (e.g. private, business)
3. (S) kanton: district (inside Switzerland)
4. (N) gbeadmgeb: region

3-parrol/5 Partner Roles

1. (N) prid: role identifier
2. (N) ptid: partner identifier; refers to attribute ptid of relation part/9
3. (N) vvid: identifier of the insurance contract where the partner ptid plays the role prid
4. (N) prtyp: type of role, i.e.

6	special agent
9	premium payer
10	pension receiver
11	insurance holder
12	insured person
14	pledgee
15	swiss representative
17	legal representative
18	guardian

5. (N) prtypnr: role number wrt. type and contract.

4-part/9 Partners Involved in Insurance Contracts

1. (N) ptid: partner identifier
2. (N) hhid: household identifier (in fact this atribute is a key related to a relation not considered by taska)
3. (N) epberuf: profession code
4. (N) eperweart: professional status (independent or employed)
5. (N) epgebudat: year of birth
6. (N) epsta: personal state

0	nothing special
1	dead
2	missed
3	under revision as disabled person
4	disabled person

7. (N) eplnamecd: name code for married persons
8. (N) epsexcd: sex: 1 - male, 2 - female
9. (N) epzivstd: marital status

0	unknown
1	single
2	married
3	widowed
4	divorced
5	separated

5-tfkomp/26 Tariff Components

1. (N) tkid: component identifier
2. (N) vvid: insurance contract number the component belongs to (see attribute vvid of relation vvert/18)
3. (N) tknr: (running) component number wrt. contract
4. (N) tfid: number of tariff the component is bound to (this atribute is related to a relation not considered by taska)
5. (N) tkbeg: starting date of tariff component
6. (N) tkend: end date of tariff component
7. (N) tkexkzwei: payment mode (payments per year in particular for pension insurances)

8. (N) tkstacd: status of the component (1 -regular premium, 2 - single premium)
9. (N) tkleist: insured benefits of the component
10. (N) tkinkprl: regular premium for the component
11. (N) tkinkpre: single premium for the component
12. (N) tktarpra: calculated (internal) premium
13. (N) tkuebvwsysp: type of profit spending for running contracts
14. (N) tkuebvwsysl: type of profit spending during pension payment period
15. (N) tkprfin: current component state wrt. premium: required or not
16. (N) tkdyncd: states whether tariff component is dynamic i.e. regular adaption of premium dependent on inflationary rate. 0 - no, 1 - yes, 2 - not applicable or unknown
17. (N) tkausbcd: states whether extension guarantee (higher insured sum without additional risk check) was claimed for tariff component 0 - no, 1 - yes, 2 - not applicable or unknown
18. (N) tkrauv: type information for the special case of risk extension and exchange insurances
19. (N) tksmed: states special medical (risk) cases
20. (N) tkrizucd: extra premium charge for certain risk cases 0 - no, 1 - yes, 2 - not applicable or unknown
21. (N) tklfleist: (regular) current payment (dynamically increasing)
22. (N) tktodleista: payment for death of insured person
23. (N) tkerlleista: (initial) payment for regular contract termination
24. (N) tkrenleista: (initial) payment of pension
25. (N) tkeuleista: (initial) payment for the case of disablement
26. (N) tkunfleista: (initial) payment for the case of accident

6-tfrol/8 - Tariff Roles

1. (N) trid: tariff role identifier
2. (N) prid: partner identifier (see relation part/9)
3. (N) tkid: link to tariff component (see attribute tkid of relation tfrol/8)
4. (N) trteceinal: (technical) age of insured person at contract agreement

5. (N) truntcd: states whether medical risk check was necessary 0 - no, 1 - yes, 2 - not applicable or unknown
6. (N) trklauscd: states whether special contract clause was negotiated 0 - no, 1 - yes, 2 - not applicable or unknown
7. (N) trstafcd: states whether staggered payment was negotiated 0 - no, 1 - yes, 2 - not applicable or unknown
8. (N) trricd: states that premium surcharge for higher risks was set 0 - no, 1 - yes, 2 - not applicable or unknown

7-vvert/18 - Life Insurance Policy

1. (N) vvid: insurance policy number
2. (N) pdid: identifier of the product the insurance contract is bound to
3. (N) vvstacd: status code of the contract
4. (N) vvinkzwei: modus of payment for premium rates
5. (N) vvbeg: year of contract start i.e. earliest start date among all tariff components (see attribute tkbeg of relation tfkomp/26)
6. (N) vvend: year of contract end (last date among all tariff components)
7. (N) vvinkprl: yearly overall premium to be payed by the policy holder sum over all tariff components (see attribute tkinkprl of relation tfkomp/26)
8. (N) vvinkpre: single premium to be payed at contract start sum over all tariff components (see attribute tkinkpre of relation tfkomp/26)
9. (N) vvwae: currency for payments
10. (N) vvversart: type of insurance contract, e.g. capital insurance, pension insurance
11. (N) vvaendart: type of last modification
12. (N) vvaendat: year of last modification
13. (N) vvabvb: responsible agent
14. (N) vvabga: responsible general agency
15. (N) vvstifcd: place where policy is left; most (own) employee's contracts are kept in a collective depot
16. (N) vvvorscd: type of (official Swiss) precaution type of the contract
17. (N) vvbvgcd: slot for storing whether the insured person participates in a companys pension insurance scheme
18. (N) vveucd: code for included disablement insurance; depends on respective tariff components