Kurgan, L., Cios, K.J., Sontag, M., & Accurso, F.J., Mining the Cystic Fibrosis Data
In: Zurada, J., & Kantardzic, M. (Eds.), *Next Generation of Data-Mining Applications*, IEEE Press, pp.415-444, 2005

Mining the Cystic Fibrosis Data

Lukasz A. Kurgan [1], Krzysztof J. Cios [2,3,4,5], Marci K. Sontag [4,6,7] and Frank J. Accurso [4,6,7]

[1] Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Canada
email: lkurgan@ece.ualberta.ca
[2] Department of Computer Science and Engineering, University of Colorado at Denver, USA
email: Krys.Cios@cudenver.edu
[3] Department of Computer Science, University of Colorado at Boulder, USA
[4] University of Colorado Health Sciences Center, USA
emails: Sontag.Marci@tchden.org, and Accurso.Frank@tchden.org
[5] 4cData, Golden, Colorado, USA
[6] The Children's Hospital, Denver, Colorado, USA
[7] Mike McMorris Cystic Fibrosis Center, Denver, Colorado, USA

Abstract

The chapter discusses challenges facing researchers performing Data Mining projects on real data and illustrates them using multi-layer Data Mining system for analysis of cystic fibrosis (CF) data. The goal of the cystic fibrosis project was to discover new information that may advance knowledge about CF. In spite of the complexity of the data and very high number of missing values our system, called MetaSqueezer, generated interesting results. They included finding some relationships that were already known to the CF experts, and which validated correctness of the approach. In addition, the system generated new and clinically important knowledge about the disease. The project was carried out using a Data Mining and Knowledge Discovery process model. The model guided our efforts, starting from problem specification, data preparation, Data Mining, evaluation of the results, to deployment of the discovered knowledge. Using the model resulted in significant reduction of development time.
Keywords: Data Mining, Knowledge Discovery, Meta Mining, Cystic Fibrosis, MetaSqueezer.

Introduction

Many Data Mining (DM) projects require extensive preprocessing and iterating between the steps of the knowledge discovery process to find new useful information. The reason for these efforts can be attributed to high complexity of the mined data. Many areas, especially medicine, are ripe for DM efforts to extract useful information that can help improve processes and procedures. However, considerable effort is required for design and implementation of procedures for data preparation, and collaborations between researchers and practitioners from several disciplines. The field of DM needs to adjust to those demands by providing a comprehensive range of services from understanding of the problem domain and data, through DM, to utilization of the discovered knowledge [13].

This chapter describes an application of our DM system, called MetaSqueezer, for analysis of clinical data describing patients with cystic fibrosis (CF). In the project we used a Data Mining and Knowledge Discovery (DMKD) process model [10] [13].
The chapter is organized as follows. First, we describe the overall goals and the DMKD process. Next, the DM methods used in the project are introduced and explained. In what follows we describe the project goals and the approach taken to generate useful knowledge from CF data. We finish with discussion of the results, and conclude with discussion of current DM challenges.

Background and Related Work

Introduction

Medical applications often aim at describing patterns of disease development and prediction of therapeutic effectiveness. In this work we address the former. The main goal was to discover new information that may advance knowledge about the disease and possibly a better treatment. The difficulty of the project was compounded by two factors:

1. High number of missing values and erroneous information, and complex structure of the data

2. Highly iterative manner in which the final results were achieved, which was caused by the necessity to reevaluate and improve data preparation and DM tasks.

In addition, since the CF data is temporal in nature it needed specific learning tools. As we shall see, in spite of the problems, the obtained results can enhance understanding of the disease. The results include knowledge already known by the CF experts, which was used to validate correctness of our methods, and the new knowledge. The new finding was evaluated as medically important by the domain experts, and will be utilized to better understand the pathophysiology of the disease.

Data Mining and Knowledge Discovery Process Model

The purpose of a DMKD model is to help plan, work through, and reduce the overall costs of a DM project, by prescribing procedures needed to be performed in each of the steps. The DMKD process model describes a range of steps from problem specification to interpretation and use of the results (the discovered knowledge). One of the main issues of the project was ability to structure the process in a formal way that helps dealing with highly iterative nature of the project. Several researchers described a series of steps that constitute the DMKD process, which range from few steps to more refined models like the nine-step model proposed by Fayyad et al. [25]. In this project we use the six-step DMKD process model [10] [11] [13] described below.

The six-step DMKD process is described as follows:

1. **Understanding the problem domain**. In this step the project is defined, including definition of objectives, and learning domain specific terminology and methods. A high-level description of the problem is analyzed, including the requirements and restrictions. The project goals are translated into DMKD goals and the project plan is prepared, which includes selection of suitable DM tools.
2. **Understanding the data**. This step includes collection of the data, and decision regarding which data will be used (including its format and size). Next, initial data exploration is performed to verify usefulness of the data with respect to the goals identified in step 1.
3. **Preparation of the data**. In this step, the data is chosen that will be used as input for DM tools in step 4. New data records are formed that meet specific input requirements of the given DM tools. The step may involve sampling and cleaning the data, assigning classes to data examples, etc. The cleaned data can be further processed by feature selection and extraction algorithms, by derivation of new attributes, e.g. by discretization, and by summarization.
4. **Data mining**. This step applies DM tools to discover new information from the data prepared in step 3. After the training and testing procedures are designed, the data model is constructed using one of the chosen DM tools, and the generated data model is verified by using the testing procedures. DM tools include many types of algorithms, such as inductive ML, rough and fuzzy sets, Bayesian methods, neural networks, clustering, association rules, support vector machines, etc.
5. **Evaluation of the discovered knowledge**. The goal of this step is to understand and interpret the results, check whether the new information is novel and interesting, and check their impact on the project goals. Approved models are retained.
6. **Using the discovered knowledge**. This step consists of planning where and how the discovered knowledge will be used.

The above described process model is visualized in Figure 1 [13].

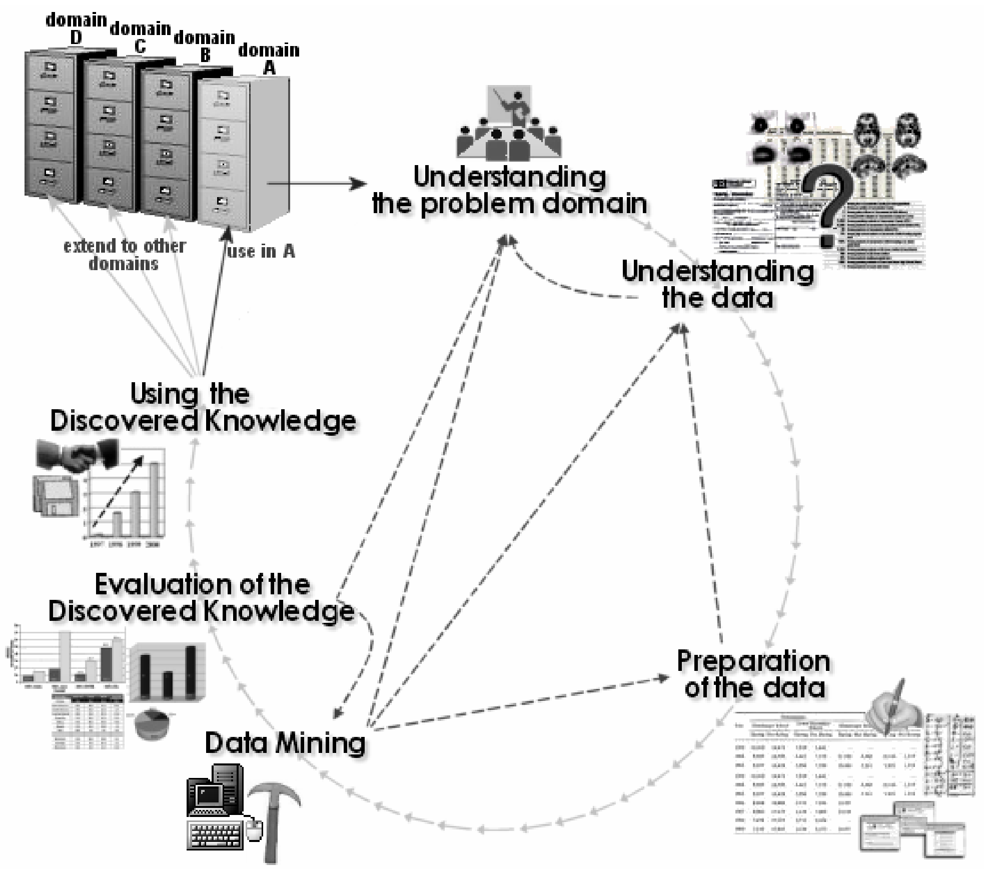

Figure 1. The six-step DMKD process model.

The iterative and interactive aspects of the process are shown in Figure 1 using dashed arrows. Since any changes and decisions made in one of the steps can result in changes in later steps the feedback loops are necessary. The loops shown in Figure 1 are by no means exhaustive.

The above and the Fayyad's models are compared in Table 1. Both models follow similar sequence of steps. The Fayyad's model provides a more detailed procedure, but performs steps concerning choosing a DM task and algorithm late in the process. The Cios's model performs this operation before data preprocessing, which results in data that is properly prepared for the DM step [13]. In case of the Fayyad's model one might have to repeat some of the earlier steps via unnecessary feedback loops to change data preparation methods used in steps 2, 3 and 4. Other advantages of the Cios's model are that it is based on an industrial tool-independent DMKD process model called CRISP-DM [53], and it provides specific guidelines concerning possible feedback loops, rather than just discussing their presence [13]. This helps to properly plan and improve efficiency of carrying out a data mining project. The model was successfully used on several medical problems such as development of a computerized system for diagnoses of SPECT bull's eye images [10], cardiac SPECT images [50] [34], and analysis of patients with vascular disease [42].

Table 1. Comparison of 6-step and 9-step DMKD models.

6 step DMKD process	9 step DMKD process
1. Understanding the domain	1. Understanding application domain, identifying the DMKD goals
2. Understanding the data	2. Creating target data set
3. Preparation of the data	3. Data cleaning and preprocessing
	4. Data reduction and projection
	5. Matching goal to particular data mining method
	6. Exploratory analysis, model and hypothesis selection
4. Data mining	7. Data mining
5. Evaluation of the discovered knowledge	8. Interpreting mined patterns
6. Using the discovered knowledge	9. Consolidating discovered knowledge

Methods

Introduction

Analysis of the CF data was done by DM system that uses supervised inductive Machine Learning (ML) combined with Meta Mining (MM) concept. ML is concerned with generation of data models from input numerical or nominal data. The models are usually inferred using induction process that searches for regularities among the data. Supervised learning is concerned with generation of a data model that represents relationship between independent attributes and a designated dependent attribute (class). Therefore, supervised inductive ML algorithms generate models that map independent attributes to the class attribute. The model generated by a supervised ML algorithm often takes form of production rules. The production rules have "IF (*conditions*) THEN (*class*$_i$)" format, where *conditions* involve one, or conjunction of several, logical expressions between independent attributes describing the data and their values, and *class*$_i$ is one of the of the values of the class attribute. Supervised inductive ML algorithms that generate production rules can be divided into three types: decision tree algorithms, rule algorithms, and their hybrids [9]. Example decision trees algorithms are CART [5], C4.5 [47], and T1 [30]; rule algorithms are the AQ family of algorithms [43] [32], FOIL [46], IREP [26], RISE [23], RIPPER [17], SLIPPER [18], and LAD [4], and hybrid algorithms are CN2 [16], and CLIP family of algorithms [12] [14]. Other types of models generated by inductive ML algorithms are also possible, for instance those generated by ML algorithms based on probability theory, statistics, and other mathematical models [44].

Examples of the latter include probabilistic algorithms like Naïve Bayes [41] and Support Vector Machines [19].
The main advantages of rules, or trees, are their simplicity and easiness of interpretation. In addition rules are modular, i.e. a single rule can be understood without reference to other rules [29]. These features are especially valuable in situations where a decision maker, say in medicine, needs to understand and validate the generated model.

Meta Mining is a novel concept, or framework, for higher order mining that generates meta-models (meta-rules) from the already generated data models, which usually are in the form of the rules (and in this framework the rules are called meta-data) [49] [51]. Inference of data models that use MM concept is a two step procedure. In case of using supervised inductive ML algorithms the first step generates productions rules from input data, while in the second step the generated rules constitute inputs to a ML algorithm (possibly the same) to generate the meta-rules. The simplest form of a meta-rule is a production rule, but other formats, such as temporal rules, are also possible. The DM system used in this project generates both meta-data and meta-rules as production rules.

The MetaSqueezer System

The analysis of the data was done using DM system called MetaSqueezer that generates data models, in terms of production rules, in three steps [37] [39]:

- Preprocessing.
  The data is repaired, validated, discretized, and transformed into the form suitable for further processing, i.e. a single relational table where a separate column holds an attribute that is used to divide the data into subsets, and class labels are generated for each data record.
- Data Mining
  Production rules are generated from data for each of the defined subsets using DataSqueezer algorithm [36] [37], which is explained in the next section. For every set of rules a rule table, that stores the generated production rules, is created.
- Meta Mining
  MM generates meta-rules from rule tables. After concatenation of all rule tables into a single table, the meta-rules (in the form of production rules) are generated by the DataSqueezer algorithm. They describe the most important patterns associated with the classes over the entire dataset. The meta-rules and rules generated in the DM step are used to compute attribute and selector importance tables, which are defined later.

The architecture of the MetaSqueezer system is shown in Figure 2.

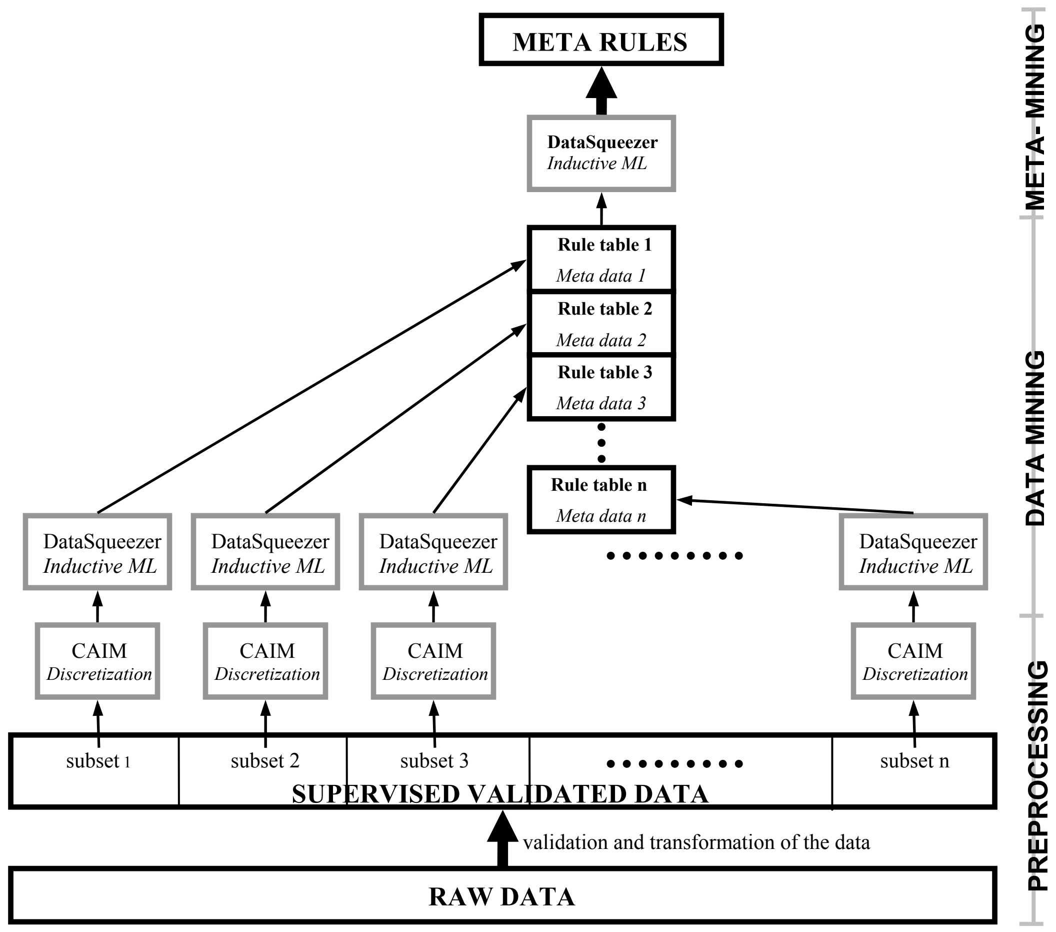

Figure 2. Architecture of the MetaSqueezer system.

The DataSqueezer Algorithm

DataSqueezer is the core algorithm used to generate meta-data during the DM step and the meta-rules during the MM step [36] [37]; it is a rule algorithm. Let us denote the input data by D, which consists of S examples. The sets of positive examples, D_P, and negative examples, D_N, must be disjoint, non-empty, and fully cover D. The positive examples are those describing the class for which we currently generate rules, while negative examples are the remaining examples. Examples are described by a set of K attribute-value pairs: $e = \wedge_{j=1}^{K}[a_j \# v_j]$, where a_j denotes j^{th} attribute with value $v_j \in d_j$ (domain of values of j^{th} attribute), # is a relation (=, <, ≈, ≤, etc.), and K is the number of attributes. DataSqueezer uses equality as a relation. An example, e, consists of set of selectors $s_j = [a_j = v_j]$. The DataSqueezer algorithm generates production rules in the form of IF (s_1 and … and s_m) THEN class = $class_i$, where $s_i = [a_j = v_j]$ is a single selector, and m is the number of selectors in the rule. Figure 3 shows pseudo-code for DataSqueezer.

```
Given:   POS, NEG, K (number of attributes), S (number of examples)
Step1.
1.1      Initialize $G_{POS}$ = []; i=1; j=1; k=1; tmp = $pos_1$;
1.2.1    for k = 1 to K                                          // for all attributes
1.2.2      if ($pos_j[k] \neq tmp[k]$ or $pos_j[k]$ = '∗')
1.2.3        then tmp[k] = '∗';                                  // '∗' denotes missing value
1.2.4    if (number of non missing values in tmp $\geq$ 2)
1.2.5        then $gpos_i$ = tmp; $gpos_i[K+1]$ ++;
1.2.6      else i ++; tmp = $pos_j$;
1.3      set j++; and until j $\leq N_{POS}$ go to 1.2.1
1.4      Initialize $G_{NEG}$ = []; i=1; j=1; k=1; tmp = $neg_1$;
1.5.1    for k = 1 to K                                          // for all attributes
1.5.2      if ($neg_j[k] \neq tmp[k]$ or $neg_j[k]$ = '∗')
1.5.3        then tmp[k] = '∗';                                  // '∗' denotes missing value
1.5.4    if (number of non missing values in tmp $\geq$ 2)
1.5.5        then $gneg_i$ = tmp; $gneg_i[K+1]$ ++;
1.5.6      else i ++; tmp = $neg_j$;
1.6      set j++; and until j $\leq N_{NEG}$ go to 1.5.1
Step2.
2.1      Initialize RULES = []; i=1;            // where $rules_i$ denotes $i^{th}$ rule stored in RULES
2.2      create LIST = list of all columns in $G_{POS}$
2.3      within every column of $G_{POS}$ that is on LIST, for every non missing value a from selected column k compute sum,
         $s_{ak}$, of values of $gpos_i[K+1]$ for every row i, in which a appears (multiply every $s_{ak}$, by the number of values the
         attribute k has)
2.4      select maximal $s_{ak}$, remove k from LIST, add "k = a" selector to $rules_i$
2.5.1    if $rules_i$ does not describe any rows in $G_{NEG}$
2.5.2        then remove all rows described by $rules_i$ from $G_{POS}$, i=i+1;
2.5.3          if $G_{POS}$ is not empty go to 2.2, else terminate
2.5.4      else go to 2.3
Output:  RULES describing POS
```

Figure 3. Pseudo-code for the DataSqueezer algorithm.

D_P and D_N store positive and negative examples in tables, where rows represent examples and columns represent attributes. POS denotes table of positive examples and N_{POS} denotes the number of positive examples, while the table and the number of negative examples are denoted NEG and N_{NEG}, respectively. Positive examples from the POS table are described by the set of values: $pos_i[j]$ where j=1,…,K, is the column number, and i is the example number (row number in the POS table). The negative examples are described similarly by a set of $neg_i[j]$ values. The DataSqueezer algorithm also uses tables that store intermediate results (G_{POS} for POS table, and G_{NEG} for NEG table), which have K columns. $Gpos_i[j]$ denotes a cell of the G_{POS} where i is a row number and j is a column number, and similarly $gneg_i[j]$ in a cell for the G_{NEG}. The G_{POS} table stores reduced subset of the data from POS, and similarly G_{NEG} for the data from NEG. The meaning of this reduction is explained later.

DataSqueezer generates production rules using inductive learning hypothesis that states that any hypothesis found to approximate the target function well (defined by a class attribute), over a sufficiently large set of training examples, will also approximate well the target function over other unobserved examples [44]. Based on this assumption in step 1 the algorithm first performs data reduction, which generalizes information stored in the input dataset, via prototypical concept learning, similar to the Mitchell's Find S algorithm [44]. The algorithm starts with the most specific hypotheses that cover individual examples, and iteratively uses generalization operator that sequentially compares current hypothesis with an example from input data and removes all attribute values that are not identical (between them).

In step 2 the algorithm generates rules by performing greedy hill-climbing search on the reduced data. A rule is generated starting with an empty rule, and adding selectors until the termination criterion fires. The max depth of the search is equal to the number of attributes. The examples covered by the generated rule are then removed, and the process is repeated.

The DataSqueezer algorithm generates production rules that involve no more than one selector per attribute, which allows for storing rules in a table that has identical structure to the original data table. The table is used as input to the same algorithm, which allows using the algorithm in a MM setting. The algorithms can also handle data with large number of missing values by simply using all available information while ignoring the missing values. The data reduction procedure of the algorithm sorts out all missing values. Every present value for every example is used to infer the rules. The algorithm has linear complexity, shown both theoretically and experimentally, with respect to the number of input examples [37].

Main Features of the MetaSqueezer System

The MetaSqueezer uses the DataSqueezer algorithm and a MM concept to generate production rules. First, the data is divided into subsets and production rules are generated for each of them by the DataSqueezer. Next, the rules are converted into the same format as the input data, concatenated into a single set, and fed back to the DataSqueezer to generate meta-rules.

The main benefits of the MetaSqueezer system are compactness of the generated meta-rules and low computational cost [37] [39]. While many other ML and DM algorithms generate rules directly from input data, the MetaSqueezer system generates meta-rules from previously generated meta-data. This results in significantly smaller number of selectors used. Low computational cost of the MetaSqueezer system is the result of its linear complexity, which is determined by complexity of the DataSqueezer algorithm, and is linear with respect to the number of examples [37] [39]. Because of the division of the data at the DM step, the system can be used to analyze ordered data. One example is temporal data that can be divided into subsets corresponding to time intervals. CF data is a good example of such data since it describes patients over a period of time, with patients being at different stages of the disease.

One of the specific features of the MetaSqueezer system is ability to generate an alternative representation of the meta-rules, which we call the attribute and selector ranking tables. The tables describe a degree of association of particular attributes and attribute-value pairs with the target class in an easy to comprehend manner. The tables are generated from rules [14] [37]. The construction of a table is based on computing goodness of each attribute and attribute-value pair using rules generated by the MetaSqueezer system, or the DataSqueezer algorithm. The goodness values are computed in three steps:

- Each rule consists of multiple selectors (attribute-value pairs) and has assigned goodness value equal to the percentage of data that describes the same class as the rule, and is described by the rule.
- Each selector is assigned a goodness value equal to the goodness of the rule it comes from. Goodness of the same selectors from different rules are summed, and then scaled to the (0,100) range.
- For each attribute, the sum of scaled goodness for all its selectors is computed and divided by the number of attribute values to obtain goodness value of the attribute.

The goodness values for each attribute and attribute-value pair are grouped into three intervals: zero that stands for no association; zero to fifty that denotes an association; and over fifty that denotes strong association between a given attribute and attribute-value pair, and the class value. The associations are visualized in a table, where rows correspond to attributes and attribute-value pairs, and columns to classes. An example is shown later in the chapter. Development of the tables resulted in significantly simplified analysis and interpretation of the results.

Discretization

The MetaSqueezer system, like many other inductive ML algorithms, handles only discrete data. This is the result of applying the data reduction procedure, which performs well only with attributes that have low number of values. Therefore for data that contains continuous attributes it uses a front-end supervised discretization algorithm. Discretization is a process of dividing a continuous attribute into a finite set of intervals to generate an attribute with small number of distinct values, by associating discrete numerical value with each of the generated intervals. A supervised discretization algorithm uses the class values assigned to input examples to achieve possibly best correlation between the discrete intervals and target classes, and therefore to minimize the information loss associated with discretization.
The MetaSqueezer algorithm uses F-CAIM (Fast Class Attribute Interdependency Maximization) algorithm to perform discretization [38] [40]. We briefly describe the algorithm next. We assume that each value of a continuous attribute can be classified into only one of the n, non-overlapping, discrete intervals, and can describe one of S classes. The class values and the discretization intervals of attribute F are treated as two random variables defining a two-dimensional frequency matrix (called quanta matrix) that is shown in Table 2, where q_{ir} is the total number of continuous values belonging to the i^{th} class that are within interval $(d_{r-1}, d_r]$, M_{i+} is the total number of values belonging to the i^{th} class, and M_{+r} is the total number of values of attribute F that are within the interval $(d_{r-1}, d_r]$, for i=1,2…,S and, r= 1,2, …, n

Table 2. 2-D quanta matrix.

Class	Interval					Class Total
	$[d_0, d_1]$	…	$(d_{r-1}, d_r]$	…	$(d_{n-1}, d_n]$	
C_1	q_{11}	…	q_{1r}	…	q_{1n}	M_{1+}
:	:	…	:	…	:	:
C_i	q_{i1}	…	q_{ir}	…	q_{in}	M_{i+}
:	:	…	:	…	:	:
C_S	q_{S1}	…	q_{Sr}	…	q_{Sn}	M_{S+}
Interval Total	M_{+1}	…	M_{+r}	…	M_{+n}	M

The F-CAIM algorithm uses Class-Attribute Interdependency Maximization (CAIM) criterion to measure the dependency (correlation) between the classes *C* and the discrete intervals *D*, for a given continuous attribute *F*, and its quanta matrix, which is defined as:

$$CAIM(C,D\,|\,F) = \frac{\sum_{r=1}^{n} \frac{\max_r{}^2}{M_{+r}}}{n}$$

where: *r* iterates through all intervals, i.e. r=1,2,...,n, max_r is the maximum value among all q_{ir} values (maximum value within the r^{th} column of the quanta matrix), i=1,2,...,S, M_{+r} is the total number of continuous values of attribute *F* that are within the interval $(d_{r-1}, d_r]$.

In general, the optimal discretization, in terms of minimal information loss, can be found by searching over the space of all possible discretization schemas to find the one with the highest value of the CAIM criterion. Since such search is highly combinatorial the F-CAIM algorithm uses a greedy approach, which searches for the approximate optimal value of the CAIM criterion by finding locally maximum values of the criterion. Although this approach does not guarantee finding the global maximum, it is computationally inexpensive and well-approximates the optimal discretization scheme [38] [40]. The algorithm consists of two steps. First, it initializes candidate interval boundaries, which define all possible boundaries of discrete intervals, and the initial discrete interval. Next, it performs consecutive additions of a new boundary that results in the locally highest value of the CAIM criterion. More specifically, the algorithm starts with a single interval that covers all possible values of a continuous attribute. It initializes candidate boundary points with minimum, maximum and all the midpoints of all the adjacent values, but only for the values that describe different classes. The algorithm computes the CAIM criterion values for all possible division points (with replacement), and chooses the division boundary that gives the highest value. It stops adding new boundary points when the best CAIM value for a current discretization schema is smaller then the highest CAIM value achieved so far.

The algorithm was recently compared with six other state of the art discretization algorithms [38], including two unsupervised algorithms: equal-width, and equal frequency [8], and four supervised algorithms: Patterson-Niblett [45], Information Entropy Maximization [24], Maximum Entropy [55] and CADD [7]. The comparison showed that the F-CAIM algorithm generates discretization schemes with, on average, the lowest number of intervals and the highest dependence between classes and discrete intervals, outperforming other discretization algorithms. The execution time of the F-CAIM algorithm is, on average, shorter than the execution time of all considered supervised discretization algorithms, while still being comparable to the time of the two fastest supervised discretization algorithms. The analysis of performance of the F-CAIM algorithm also showed that the small number of intervals that the algorithm generates helps to reduce the size of the data, and improves accuracy and the number of rules that are generated by subsequently used ML algorithms [38] [40].

Significance

After extensive literature search we found only one other application of inductive ML techniques to analysis of temporal medical data. A system that discovers limited temporal relations using Bayesian networks and C4.5 algorithm [47] was used to find relations between consecutive records, without generalizing temporal rules for the entire period of time [31]. Although the MetaSqueezer system does not discover any temporal relationships, it can be used to derive non- temporal patterns, in terms of production rules. The rules describe the data across time, but using meta-data that describes data within particular data subsets that represent temporal intervals.

Applications of Inductive Machine Learning in Medicine

Inductive ML techniques found a number of applications in analysis of medical data. They range from automated diagnostic systems to prognosis. Since it is virtually impossible to list all applications, below we describe only a few representative examples. They include diagnosis and prognosis of breast cancer [52] [54], prognosis of the survival in hepatitis [33], prognosis of traumatic brain injuries [1], and diagnosis of cardiac SPECT images [35]. Most recently inductive ML has found its use in bioinformatics (analysis of genetic data [3], analysis of mass spectrometry data [56] [2] [27]). An overview of applications of ML in medicine is given in [34]

The Cystic Fibrosis Project

Understanding the Problem Domain

CF is a genetic disease affecting approximately 30,000 children and adults in the United States [20]. One in 31 Americans and one in 28 Caucasians carry the defective gene causing CF, which translates into more than 10 million carriers in the United States.

Carriers do not exhibit the symptoms, and thus they do not know about their risk for transmitting the disease to their offspring. An individual must inherit a defective copy of the CF gene from each parent to be affected. Statistically, when two carriers conceive a child, there is a 25 percent chance that the child will have CF, a 50 percent chance that the child will be a carrier, and a 25 percent chance that the child will be a non-carrier.

CF is caused by at least 1000 different genetic mutations in the Cystic Fibrosis Transmembrane Conductance Regulator (CFTR) gene. The delta F508 mutation accounts for approximately 70% of the mutations making it the most common CF mutation [22] [21].

CF is a deadly disease that affects multiple systems, including the respiratory system, digestive system, endocrine system, and reproductive system. The symptoms are variable and may include high chloride and sodium (salt) concentration in the sweat, lung disease (persistent coughing, wheezing or pneumonia), excessive appetite but poor weight gain, and bulky stools. The vast majority of CF related mortality is caused by the progressive lung disease, caused by a vicious cycle of infection and inflammation. Thus, lungs function tests are recognized as very good indicators of the stage of the disease. CF is diagnosed usually by the sweat test, which measures amount of salt in the sweat. A high chloride level indicates that a person has CF. The treatment of CF depends upon multiple factors like stage of the disease and which organs are involved. In case of the most severely affected organ, the lungs, the disease is treated usually by chest physical therapy, and antibiotics, which are used to treat lung infections [20]. The CF data is temporal in nature. It describes several hundreds of CF patients. For each patient multiple visits are recorded. Most of the patients are monitored from the time of diagnosis throughout childhood, and are currently being followed, while some patients died during the follow-up period. For each visit multiple attributes describing demographical information, various lab results, and diagnostic information are recorded. The data describes different relationships depending on the stage of the disease. Thus, any investigation that uses such data must be able to separate the data into subsets, corresponding to particular stages of the disease.

Before the project was started, the CF experts were requested to provide only the necessary minimum background knowledge. This was done to assure that the research would not be biased toward finding solutions that would confirm accuracy of the system based on the domain knowledge. The project goals defined by clinicians were:

1. **GOAL 1** is to discover patterns (important factors) that influence the pace of development of CF. Although CF affects multiple systems, the pulmonary system is the best indicator of progression of the disease. For some patients the disease progresses very fast while for others its progress is relatively slow. There are some known factors that are related to the pace of the disease but still much is unknown. The first goal was to discover factors that are related to different, predefined based on historical data, paces of disease development.
2. **GOAL 2** is to discover important factors that are related to particular kinds of CF. CF is a genetic disease, and different distinct genotypes related to it are described. The goal was to find factors that are related to different, predefined genotypes, of CF.

In the next step we redefined the above two goals into DM goals:

1. **GOAL 1**. It can be defined as a supervised inductive learning task. Class attribute must be defined, which will group the data into subsets corresponding to patients who exhibit different paces of disease development. Since the data is temporal in nature,

additional attribute is used to divide the data according to time. The attribute will describe the stage of the disease based on the status of a pulmonary function test.

2. **GOAL 2**. Again, it can be defined as a supervised inductive learning task. Thus, a class attribute that describes patients in terms of the particular type of CF must be defined. Next, the data must be divided in a temporal manner by using attribute(s) describing patient's lung function.

After analyzing both goals, the MetaSqueezer system was chosen as the appropriate DM tool. There are three main factors that influenced the choice of the system:

- It generates small number of very simple rules. The project required physicians to be able to analyze and comprehend the rules, and the results to be displayed in an easy to understand tabular form.
- It can handle large quantities of missing values. CF data contains large quantities of missing information. The DataSqueezer algorithm, a core inductive ML algorithm used within the system, was proven to generate accurate results in presence of significant amount of missing information. [37] [39].
- It can handle temporal data. CF data is temporal in nature. The DM step of the MetaSqueezer system accepts multiple training data subsets, which represent temporally organized subsets of the CF original data.

Understanding the Data

The data was collected at the CF Center based at the University of Colorado Health Sciences Center and the Children's Hospital in Denver starting in 1982. It includes demographical information about patients, clinical information including a variety of lab tests, and diagnoses. The data was collected on 856 patients. The data was stored using seven relational tables. To comply with the HIPPA requirements, all identification information was removed from the data before it was used in the project. The resulting data holds only relevant clinical information. There are several complicating issues with the CF data. First, as expected, it contains significant amount of missing information. For example, three tables contain more than half of missing information. Second, it can be also expected that since the data was inserted manually by the physicians, it will contain also substantial amount of incorrect records. Also, the tables contain different attributes: numerical, textual, and binary, which need to be handled by the learning algorithm. The tables also possibly include large quantities of irrelevant information, which may be removed before the learning process is executed.

The attributes used to define classes for both goals and to divide the data into temporal intervals are described below:

- FEV_1% (Forced Expiratory Volume in One Second % Predicted) attribute describes the amount of air that can be forced out in one second after taking a deep breath. The volume that an individual patient can blow out in one second is compared to a normal population and the percent of predicted based on the patients age, sex, and height is reported as the FEV_1%. In CF, the test result indicates the stage of the disease better than timestamp information, such as patient's age or visit date, because different patients are diagnosed and start treatment at different ages. The FEV_1% is used as the attribute to define temporal intervals for both goals. It is also used to define the class attribute for the first goal.

- Genotype 1 and Genotype 2 attributes describe genetic mutations, and are used to distinguish different levels in severity in CF. Since the second mining goal is to find important factors that are related to particular kinds of CF, combination of the two attributes will be used to provide class labels for the goal.

In summary, the data was assumed to hold all information necessary to carry out the project. More specifically, several attributes that may be used to derive classes and temporal subsets for both learning goals were identified.

Preparation of the Data

Before the CF data could be used to generate rules it needed to be preprocessed. This usually involves removing or correcting noise and missing values, sampling and reorganizing the data, assigning classes to examples, identifying temporal intervals, etc. The cleaned data is later discretized. As with most of DM projects it is expected that this step consumes most of any project's time since data preparation greatly affects the outcome of the entire project [6] [13].

At the first step, manual data checking and cleaning were performed. As expected, the CF data contained significant amount of errors and inconsistencies. Problems connected with physician's interpretation that is written in an unstructured free-text English (text fields in the database), which is very difficult to standardize and thus difficult to mine [15], were encountered. For instance, the values of the FEV_1% attribute should be in the range [0; 200], but some records had bigger values. Another problem is presence of null attributes, which have null values for all tuples (rows) and thus should be deleted. Almost all tables in the CF data contained attributes that were null.

Two main manual operations were performed to clean the CF data. First, the consistency of attributes was corrected by merging together their corresponding values. This operation is caused by inconsistent format of data, e.g. *Yes*, *Y*, *yes* were entered as corresponding to the same value and were corrected. Next, erroneous values of attributes were identified and removed. After the data was cleaned, it was converted into a single relational table. In order to merge the seven tables several join operations were performed. The tables were merged in pairs, where results of one join operation were merged with the next table. The condition of the join operations were designed and consulted with clinicians to ensure proper handling of medical information.

The next data preparation step consisted of removing attributes that are irrelevant to the planned goals. After consultation with the physicians attributes that were found either irrelevant from the medical point of view or containing too much noise or missing information were removed.

Class Attributes

There are two class attributes, one per each goal. The first attribute describes pace of CF development, which is used to define classes for goal 1. The attribute was derived based on "visdate1" and "FEV_1%" attributes using procedure that was designed with cooperation with clinicians. The first goal was defined as the 4-class supervised inductive learning task, with the following classes: FastDegrad (for fast degrading patients), SlowDegrad (for slowly degrading patients), NoChange (for patients with no change in lung functions), and Improv (for patients for whom there was improvement). The second class attribute describes different types of CF and is used to define classes for goal 2. The attribute was derived based on "Genotype 1" and "Genotype 2" attributes using procedure that was designed with cooperation with clinicians. Three kinds of CF were defined: 1) both Genotype 1 and Genotype 2 are F508, 2) Genotype 1 is F508 or Genotype 2 is F508, with the other genotype being not F508, 3) both Genotype 1 and Genotype 2 are not F508 [37].

Time-Defining Attribute

The time-defining attribute, used to divide the data into subsets for the DM step of the MetaSqueezer system, was derived from the FEV_1% attribute. There were 5 discrete values of the attribute generated, which means that the input data was divided into 5 coherent subsets depending on the value of the FEV_1% attribute, i.e. <40%, 40-60%, 60-80%, 80-100% and >100% [37].

Discretization

After deriving class and time-defining attributes the data was discretized. First, each attribute was manually evaluated to belong to one of three categories

- Discrete. An attribute was defined as discrete if it had a small number of values, up to about 20 distinct values. The discrete attributes were left unchanged.

- Continuous, for manual discretization. An attribute was defined as continuous for manual discretization is it had large number of distinct vales, and its values had a well known specific meaning with respect to CF. These attributes were discretized manually by clinicians to accommodate for medical meaning of their values, e.g. certain test values may be usually associated with specific medical conditions.

- Continuous, for automated discretization. Attributes that are characterized by large number of distinct values, with no known medical relationship of their values and CF were considered continuous for automated discretization. Those attributes were discretized by the supervised discretization algorithm F-CAIM. Each attribute was discretized separately for each goal, since discretization process uses class labels. Over 50% of continuous attributes fell into this category.

Training Data

The objective for goal 1 was to discover factors related to different pace of development of CF. In short, the training set was derived from the original data by first removing noise and inconsistencies, merging the seven tables, removing irrelevant attributes, defining class and time-defining attributes, and finally discretizing the continuous attributes. Two more steps were performed to generate final version of the training set: 1) examples that include incomplete critical information, in terms of class or temporal information, were removed; 2) examples that had too many missing values were removed. During the join, if a tuple from one table was not matched with a tuple from another table it was padded with missing values. If an example was not matched during several subsequent joins, and it contained many missing values, then it was treated as an outlier and removed.

The objective for goal 2 was to discover factors related to different types of CF. The training set for goal 2 was derived in the same way as the training set for the goal 1, with a different class attribute. Summary information about training data sets is shown in Table 3.

Table 3. Summary of training sets for the CF project.

set	*size*	*# classes*	*# attrib.*	*test data*	*% missing values*	*% inconsistent examples*	*# subsets*
CF1	5448	4	160	10CV	56.6	0	5
CF2	6022	3	160	10CV	56.2	0	5

The step of data preparation was highly iterative. The above description only describes the final outcome, which was derived after several iterations.

Data Mining

The DM task for both training sets was very difficult. The datasets are characterized by a very large number of missing values and at the same time these missing values are distributed over a majority of attributes. For both datasets, all examples contain some missing values, and the total number of missing information is larger than the amount of complete information. Therefore, both datasets include sparse data, which is very difficult to handle while using ML algorithms. It can be expected that the data is very specific, i.e. there may be very little common patterns between different patients, and thus generated rules may be long. The first factor was overcome by application of the MetaSqueezer system, which is proven to be missing values resistant. The second factor was overcome by development of an alternative knowledge representation. Production rules, generated by the MetaSqueezer, were transformed into tables, called rule and selector ranking tables, as described above.

Experimental Results

Both training sets were used as input to the MetaSqueezer system, which generated a set of production rules. Next, the rules were evaluated by performing two tests for each of the defined goals:

- The first test takes the training set and performs 10 fold cross validation with the same setting as for the second test. The results show verification test results, running time, and the number of rules and selectors for each of the runs, and their mean values. The results of this test have shown simplicity, accuracy, efficiency, and flexibility of the system, but could not be used by the clinicians since ten different rule sets were generated. Instead, this test is used to validate results generated for the second test, using the same system settings. Tables 4 and 5 shows results on the training set for goals 1 and 2, respectively.

- The second test generates a set of rules from the entire training data set. The rules are used to generate the attribute and selector ranking tables, and tested on the same training set. The results report accuracy on the training data, number of generated rules and selectors. These results are analyzed by clinicians. The summary of results for the second test is shown in Tables 6 for both goals. The table also compares the results with the results obtained during the 10 CV tests.

The results report accuracy, sensitivity and specificity, which are defined below. The verification test, which is frequently used in evaluating medical diagnostic procedures, gives much better and specific information about goodness of the generated rules, as compared with just reporting accuracy results [15]. The verification test consists of these three evaluation criteria:

$$sensitivity = \frac{TP}{hypothesis \quad positive}100\% = \frac{TP}{TP + FN}100\%$$

$$specificity = \frac{TN}{hypothesis \quad negative}100\% = \frac{TN}{FP + TN}100\%$$

$$accuracy = \frac{TP + TN}{total}100\% = \frac{TP + TN}{TP + TN + FP + FN}100\%$$

where: positive hypotheses concerns rules for currently evaluated class; negative hypotheses the remaining rules; TP (true positive) is a number of correct positive classifications; TN (true negative) is a number of correct negative classifications; FP (false positive) is a number of incorrect positive classifications; and FN (false negative) is a number of incorrect negative classifications.

The results also report mean values for each criterion, averaged over all classes. The sensitivity measures how many of the examples classified by the rules as belonging to the currently evaluated class, truly belong to this class. The specificity measures how many of the examples classified by the rules as not belonging to the currently evaluated class, truly did not belong to it. They enable evaluation of how the rules perform on the part of the data they should describe, i.e. for the class they are intended to describe, versus their performance on the remaining part of the data. Only the results with high values for all three measures can assure high confidence in the generated rules.

Table 4. 10 fold cross validation results for goal 1.

Trial	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*StDev*	*Mean*
Accuracy	57.8	58.2	65.1	60.6	59.1	59.1	66.8	60.4	61.7	61.5	*2.94*	**61.0**
Specificity	88.9	89.3	90.2	88.8	90	89.6	91.2	90.5	91	89.8	*0.8*	**89.9**
Sensitivity	57	59.2	60.1	63.3	60.7	52.6	63.4	58.4	59	56.6	*3.22*	**59.0**
Time [ms]	8663	8501	8169	8537	8428	8794	8663	8360	8413	9010	*239*	**1m 25s 53ms**
# Rules	452	448	428	494	493	345	470	435	356	434	*50.5*	**436**
# Selector	4437	4320	4125	4710	4720	3393	4603	4213	3460	4196	*467*	**4.22E+03**

The results for the first goal show that the system generates accurate rules. Two factors need to be considered to evaluate accuracy of the system. First, the data contain only about 44% of complete information. Second, the default hypothesis, where the most frequent class is selected, for that training data set has 34.2% accuracy. Therefore, the accuracy of 61% for 10 CV tests for rules generated by the MetaSqueezer system is satisfactory. The rules achieve high and comparable values for all tests: sensitivity, specificity and accuracy, which show their high quality. The simplicity of results generated by the system is also high. The system generates only 436 rules for a sparse input data containing almost 6000 examples described by 160 attributes. Considering the input data, the average number of selectors per rule, which is 9.7, is low. The system generates the rules in about 86 seconds, which is a very good result considering the size of the training set.

Table 5. 10 fold cross validation results for goal 2.

Trial	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*StDev*	*Mean*
Accuracy	57.4	54.1	69.8	41.3	53.6	51.1	73.8	71	69.3	73.1	*11.3*	**61.4**
Specificity	88.6	86.7	89.8	88.3	85.5	91.2	92.2	88.7	89	90.1	*1.97*	**89**
Sensitivity	71	67.5	66.4	54.6	64.1	57.4	73.8	73.2	68	76.9	*7.11*	**67.3**
Time [ms]	16918	16289	17865	15709	16626	15720	17550	17232	17765	17577	*811*	**2m 49s 25ms**
# Rules	498	490	804	215	502	204	791	770	758	749	*233*	**578**
# Selector	5211	5073	8387	2238	5240	2039	8307	8093	7916	7648	*2.44E+3*	**6.02E+3**

The results for the second goal also show that the system generates accurate rules. We note that the data contain only about 44% of complete information, and the default hypothesis for that training set has 46.0% accuracy. The reported accuracy is 61%. All values of the verification test are comparably high. The system generated 578 rules. The average number of selectors per rule, which is 10.4, is low. In general, results achieved by the MetaSqueezer for the training data describing goal 2 are comparable, in terms of quality, to the results achieved for goal 1.

Table 6. Summary of test results for goal 1 and 2.

Goal	*Test type*	*accuracy*	*# rules*	*# selectors*
Goal 1	Using training set	**67.6**	**498**	**4838**
	10 CV, mean values	**61.0**	**436**	**4220**
Goal 2	Using training set	**77.2**	**790**	**4809**
	10 CV, mean values	**61.4**	**578**	**6020**

The summary results for both goals show that the MetaSqueezer generates slightly more accurate rules when using the entire training set as input. This shows that the results presented to clinicians were not overfitted.

Evaluation of the Discovered Knowledge

The DM step generated two sets of rules, one for each of the defined goals. The rules were transformed into the rule and selector tables, which were presented to CF experts. The tables were analyzed by them using the following 4 grade scale:

- 1+ was assigned for trivial, not useful results. By default such mark was simply omitted from being displayed on the table. The associations described by that mark are considered not useful from the medical perspective.

- 2+ was assigned for results that are of little interest. The associations described by that mark are considered of marginal value from the medical perspective.

- 3+ was assigned for interesting, but already known results. The associations described by that mark are considered interesting, but were already discovered by other researchers. Such results are used to provide validation of the results generated by the system.

- 4+ was assigned for very interesting, and unknown results. The associations described by that mark are of the highest value, since they show very important finding which are not yet confirmed or reported in the professional literature. Such findings, if found, are the basis for evaluating the entire project as successful.

The evaluation was performed by our CF expert. The analysis of the results was performed manually based on the attribute and selector ranking tables. The tables and the evaluation of the findings for goals 1 and 2 are shown, using marks, in Tables 7 and 8, respectively. The tables show only attributes that are assigned marks 2+, 3+, and 4+ [37].

Table 7. Evaluation of results for goal 1.

ATRIBUTE	VALUE	MARK	CLASS FASTDEGRAD					CLASS IMPROV					CLASS NOCHANGE					CLASS SLOWDEGRAD				
			TI 1	TI 2	TI 3	TI 4	TI 5	TI 1	TI 2	TI 3	TI 4	TI 5	TI 1	TI 2	TI 3	TI 4	TI 5	TI 1	TI 2	TI 3	TI 4	TI 5
CFtypes (cf)	Type4	**2+**										▨										▨
race (dem)	Black	**3+**						▨		▨	▨											
group (dem)	C	**3+/4+**	▨	▨		▨										▨		▨	▨			
group (dem)	NBS	**3+/4+**	▨					▨	▨	▨	▨	▨	▨	▨	▨	▨		▨			■	
group (dem)	MI	**3+/4+**						▨	▨													
group (dem)	FN	**3+/4+**							▨	▨												
motage (dem)	[22.50,48.50)	**3+**	▨			▨		▨			▨	▨	▨	▨	▨	▨	▨	▨			▨	
motage (dem)	[19.50,22.50)	**3+**			▨														▨			
mecil (dem)	TreatedSurgically	**2+**				▨																
sweatelectr1 (dia)	[24.50,46.00)	**4+**								▨	▨	▨	▨	▨		▨						
sweatelectr2 (dia)	[-11.00,95.50)	**3+**						▨	▨		▨											
tobraresistent? (cul)	Suscept.	**2+**																		▨		
na (mic)	[129.00,143.00)	**2+**																			▨	
prot (mic)	[-1.95,7.95)	**2+**																			▨	
vita (mic)	[0.44,748.00)	**2+**																			▨	
wbc (hem)	[4.05,18.00)	**2+**																			▨	
hct (hem)	[27.40,45.50)	**2+**																			▨	
mch (hem)	[24.90,91.40)	**2+**																			▨	

mchc (hem)	[30.40,35.80)	**2+**																			▨	
rdw (hem)	[-0.85,15.40)	**2+**																			▨	
HAZ (per)	[-2.91,-1.87)	**3+**	▨																			
age@diagnosis (dia)	025till2	**3+**		▨	▨		▨												▨	▨		

The results generated by the MetaSqueezer system for goal 1 are divided into two parts:

- confirmatory results marked by 3+ describe relationships that were known previously, but give confidence in the correctness of the performed analysis:
    - black race and improvement of the disease; the finding suggests that the patients who are black may have less severe disease, possibly less severe CF mutations or other genetic modifiers,
    - C group and degradation of the disease for small values of FEV$_1$%; the finding suggests that patients who are conventionally diagnosed may have a faster decline in FEV$_1$ during advanced stages of the disease
    - NBS groups and improvement of the disease; the finding suggests that the benefits of newborn screening may result in stable or improving lung function in childhood, which may be the result of closer follow-up in early childhood,
    - MI and FN groups and improvement of the disease for small values of FEV$_1$%; the finding suggests that presence of meconium ileus at birth and presenting as a false negative on the newborn screen are associated with the improvement in FEV$_1$% for low values of FEV$_1$%,
    - [22.50,48.50) values of motage and improvement or stable state of the disease; the finding suggests that children of mothers over the age of 22 years tend to have stable lung function,
    - [19.50,22.50) values of motage and degradation of the disease for medium values of FEV$_1$%; the finding suggests that children with moderate lung disease who have young mothers (between 19.5 and 22.5 years) tend to have a greater decline in lung function,
    - [-11.00,95.50) values of sweatelectr2 and improvement of the disease; the finding suggests that children with lower sweat chloride values (<95.5) may have less severe lung disease,
    - [-2.91,-1.87) values of HAZ and degradation of the disease for small values of FEV$_1$%; this finding suggests that children with height stunting and severe disease may have a rapid decline in FEV$_1$,
    - 025till2 values of age@diagnosis and degradation of the disease; the finding suggests that children who were diagnosed after the initial newborn period may have a more rapid decline in pulmonary function,

- new findings marked by 4+ describe findings that may be significant medically:
    - [24.50,46.00) value of sweatelectr1 and the improvement of the disease; the finding suggests that there is a possible significance of sweat electrolytes.

The results show not only that the mining system generated accurate results for goal 1, based on 10 confirmatory findings, but also that the system discovered one significant finding that concerns sweat electrolytes levels.

Table 8. Evaluation of results for goal 2.

ATRIBUTE	VALUE	MARK	CLASS TYPE1					CLASS TYPE2					CLASS TYPE4				
			TI 1	TI 2	TI 3	TI 4	TI 5	TI 1	TI 2	TI 3	TI 4	TI 5	TI 1	TI 2	TI 3	TI 4	TI 5
CFpace (cf)	Improv	3+											▨	▨		▨	
group (dem)	C	2+											▨	▨		▨	
group (dem)	NBS	2+	▨	▨	▨	▨		▨	▨		▨		▨	▨		▨	
irt1 (dia)	[391.00,759.00)	3+		▨													

The results generated by the MetaSqueezer system for goal 2 include only several confirmatory findings:

- improvement of the disease and Type 4 CF; the finding suggests that Children with 2 non-F508 mutations may have mild lung disease,
- [391.00,759.00) values of irt1 and Type 1 CF for medium values of FEV_1%; the finding suggests that children with high IRT values at birth have moderate lung disease.

The results show that the system generated accurate results for goal 2, based on 2 confirmatory findings. No new and significant findings were discovered.

Using the Discovered Knowledge

The six-step DMKD process has proven to be very practical. The iterative process used for generation of results significantly improved the results generated by the MetaSqueezer system. The system generated two kinds of results: 12 confirmatory findings that prove the correctness of our DM effort, and most importantly one significant new finding that shows that the system is capable of generation of useful results.

The outcome of the project was evaluated to be very successful by the CF experts. The new finding discovered for goal 1 was classified as possibly medically significant, and may help to bring new clinical insight into this deadly disease. The discovered information currently undergoes detailed medical scrutiny before it can find a possible clinical application.

Refining the Project

The steps described above were performed in a highly iterative manner, a common practice in all DMKD projects. Below we provide the summary of iterations, including description of changes and reasons for them. Since the beginning of the project the CF data was identified as very challenging for reasons, such as very large amount of missing information, incorrect records, and the complex structure of the CF data. The project was performed slowly, with several iterations, and careful revisions of the intermediary results. Several formal meetings with CF experts were held to evaluate the progress and direct the research in addition to many informal meetings and discussions. The results of the meetings are summarized in Table 9, which shows all major iterations during the CF project.

Table 9. Summary of the refinements performed during the analysis of the CF data.

current DMKD step	*returned to*	*reasons for modifications*	*summary of modifications*
Data Mining	Preparation of the Data	incomplete and difficult to evaluate results	modification of the join operation, hand-coded discretization of several attributes, removal of several irrelevant attributes, redesign of the "CF pace" attribute, new format of displaying the results
Evaluation of the Discovered Knowledge	Understanding of the Data	unsatisfactory results	New data table, describing weight and height percentiles was added, modification of the join operation to accommodate for the new table, hand-coded discretization of several attributes that were discretized automatically, deletion of examples with high number of missing values, removal of several irrelevant attributes
Evaluation of the Discovered Knowledge	Data Mining	invalidated results	10 fold cross validation test procedures, improvement in the new format of displaying the results, removed minor data inconsistencies

The shown iterations represent only the major modifications performed during the project. They were done at different steps of the DMKD process and resulted in the refinement of the process by returning, modifying, and repeating some of the previously performed steps. The redesign was guided by both medical and the DM experts. The main reason for the refinements was unsatisfactory quality of initial results. The modifications resulted in improving quality of the training sets and quality of the representation of the results. It is important to note that all of the performed refinements resulted in substantial and gradual improvement of the final results.

Summary

The chapter describes results of mining in cystic fibrosis data, which is used as a case study to illustrate major issues facing data mining researchers.
Very often selection of a proper DM tool can significantly help in achieving a successful outcome of the DMKD project. As an example the MetaSqueezer system helped to solve issues like dealing with large number of missing data and generation of easily understandable format of the results. Mining in CF data was used to illustrate key issues facing researchers and practitioners of DM:

- **Extensive preprocessing**. One of the important characteristics of the DMKD process is the relative amount of time spent to complete each of the steps. One of the most expensive steps is the data preparation, or preprocessing, step. The estimates for the length of this step vary from 60% [6], 30-60%, [13], and 30% [28], depending on the application domain and the status of the existing original data. The reasons for extensive preprocessing time include: large amount of erroneous data, redundancy and inconsistency of the data, and lack of all the necessary data to achieve success [48].
- **Iterative nature of DMKD projects**. The majority of DMKD projects are performed in a highly iterative manner, where the final results are achieved after performing a number of feedback loops, leading to reevaluation, redesign, and repeated execution of some of the earlier steps. It is important to recognize this necessity ahead of time, and to use a DMKD process model that can properly accommodate for the feedback mechanism.
- **Interdisciplinary nature of DMKD projects**. Main portion of the DMKD project concerns performing analysis of data that spans multiple domains. This directly leads to necessity of close collaboration between people from different disciplines, e.g. medical and DM. Currently even more multidisciplinary projects are initiated because of the increased complexity of the data and a general trend to use integrated data sources.

Being aware of the above described issues before one starts a DM project helps to reduce costs and shorten time to generation of meaningful results.

References

1. P J. Andrews, D. Sleeman, P. Sathan, A. McQuatt, V. Corruble, and P.A. Jones, “Forecasting Recovery after Traumatic Brain Injury Using Intelligent Data Analysis of Admission Variables and Time Series Physiological Data - a Comparison with Logistic Regression”, *Journal of Neurosurgery*, **97**, 326-336, (2002).

2. C. Baumgartner, D. Baumgartner, and C. Boehm, Classification on High Dimensional Metabolic Data: Phenylketonuria as an Example, *Proceedings of Second International Conference on Biomedical Engineering* (BioMED2004), Innsbruck, Austria, 2004, 357-360.

3. D. Berrar, M. Granzow, W. Dubitzky, K. Wilgenbus, S. Stilgenbauer, H. Döhner, P. Lichter, and R. Eils, New Insights in Clinical Impact of Molecular Genetic Data by Knowledge-driven Data Mining, *Proceedings of the Second International Conference on Systems Biology*, Omnipress, Wisconsin, USA., 2001, 275-281.

4. E. Boros, P.L. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz, and I. Muchnik, "An Implementation of Logical Analysis of Data", *IEEE Transactions on Knowledge and Data Engineering*, **12:2**, 292-306, (2000).

5. L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*, Chapman and Hall, 1984.

6. P. Cabena, Hadjinian, R. Stadler, J. Verhees, and A. Zanasi, *Discovering Data Mining: From Concepts to Implementation*, Perentice Hall, 1998.

7. J.Y. Ching, A.K. Wong, and K.C. Chan, "Class-Dependent Discretization for Inductive Learning from Continuous and Mixed Mode Data", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **17:7**, 641-651, (1995).

8. D. Chiu, A. Wong, and B. Cheung, Information Discovery through Hierarchical Maximum Entropy Discretization and Synthesis, in G. Piatesky-Shapiro, and W.J. Frowley, eds., *Knowledge Discovery in Databases*, , MIT Press, 1991.

9. K.J. Cios, W. Pedrycz, R. Swiniarski, *Data Mining Methods for Knowledge Discovery*, Kluwer, 1998.

10. K.J. Cios, A. Teresinska, S. Konieczna, J. Potocka, and S. Sharma, "Diagnosing Myocardial Perfusion from PECT Bull's-eye Maps - A Knowledge Discovery Approach", *IEEE Engineering in Medicine and Biology Magazine*, special issue on Medical Data Mining and Knowledge Discovery, **19:4**, 17-25, (2000).

11. K.J. Cios, ed., *Medical Data Mining and Knowledge Discovery*, Springer, 2001

12. K.J. Cios K., and L.A. Kurgan, Hybrid Inductive Machine Learning: An Overview of CLIP Algorithms, in L.C. Jain, and J., Kacprzyk, eds., *New Learning Paradigms in Soft Computing*, Physica-Verlag (Springer), 2001, 276-322.

13. K.J. Cios, and L.A. Kurgan, Trends in Data Mining and Knowledge Discovery, in N.R. Pal, L.C. Jain, and N. Teoderesku, eds., *Knowledge Discovery in Advanced Information Systems*, Springer, in print, 2004.

14. K.J. Cios, and L.A. Kurgan, CLIP4: Hybrid Inductive Machine Learning Algorithm that Generates Inequality Rules, *Information Sciences*, special issue on Soft Computing Data Mining, in print, 2004.

15. K.J. Cios, and G.W. Moore, "Uniqueness of Medical Data Mining", *Artificial Intelligence in Medicine*, **26:1-2**, 1-24, (2002).

16. P. Clark, and T. Niblett, “The CN2 Algorithm”, *Machine Learning*, **3**, 261-283, (1989).

17. W.W. Cohen, Fast Effective Rule Induction, *Proceedings of the Twelfth International Conference on Machine Learning*, Morgan Kaufmann, 1995, 115-123.

18. W.W. Cohen, and Y. Singer, A Simple, Fast, and Effective Rule Learner, *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, Menlo Park, CA, AAAI Press, 1999, 335-342.

19. C. Cortes, and V.N. Vapnik, “Support Vector Networks”, *Machine Learning*, **20**, 273-279, (1995).

20. Cystic Fibrosis Foundation, http://www.cff.org/, 2002.

21. Cystic Fibrosis Mutation Database, http://www.genet.sickkids.on.ca/cftr/, 2003.

22. Cystic Fibrosis Genetic Analysis Consortium, “Worldwide Survey of the [Delta]F508 Mutation- Report from the Cystic Fibrosis Genetic Analysis Consortium”, *American Journal of Human Genetics*, **47**, 354-359, (1990).

23. P. Domingos, The RISE System: Conquering Without Separating, *Proceedings of the Sixth IEEE International Conference on Tools with Artificial Intelligence*, New Orleans, LA, IEEE Computer Society Press, 1994, 704-707.

24. U.M. Fayyad, and K.B. Irani, Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning, *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, San Francisco, CA, 1993, 1022-1027.

25. U.M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, “Knowledge Discovery and Data Mining: Towards a Unifying Framework”, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (KDD96), Portland, OR, AAAI Press, 1996.

26. J. Furnkranz, and G. Widmer, Incremental Reduced Error Pruning, *Machine Learning: Proceedings of the Eleventh Annual Conference*, New Brunswick, New Jersey, Morgan Kaufmann, 1994.

27. E.C. Gunther, D.J. Stone, R.W. Gerwien, P. Bento, and M.P. Heyes, “Prediction of Clinical Drug Efficacy by Classification of Drug-induced Genomic Expression Profiles in Vitro”, *Proceedings of the National Academy of Sciences*, **100**, 9608-9613, (2003).

28. K.K. Hirji, “Exploring Data Mining Implementation”, *Communications of the ACM*, **44:7**, 87-93, (2001).

29. M. Holsheimer, and A.P. Siebes, *Data Mining: the Search for Knowledge in Databases*, Technical report CS-R9406, http://citeseer.nj.nec.com/ holsheimer91data.html, 1994.

30. R.C. Holte, "Very Simple Classification Rules Perform Well on Most Commonly Used Datasets", *Machine Learning*, **11**, 63-90, (1993).

31. K. Karimi, and H.J. Hamilton, Finding Temporal Relations: Causal Bayesian Networks vs. C4.5, *Proceedings of the Twelfth International Symposium on Methodologies for Intelligent Systems* (ISMIS'2000), Charlotte, NC, USA, 2000, 266-273.

32. K.A. Kaufman, and R.S. Michalski, Learning from Inconsistent and Noisy Data: The AQ18 Approach, *Proceedings of the Eleventh International Symposium on Methodologies for Intelligent Systems*, Warsaw, Poland, 1999, 411-419.

33. I. Kononenko, I. Bratko, and E. Roskar, Experiments in Automatic Learning of Medical Diagnostic Rules, *International School for the Synthesis of Expert's Knowledge Workshop*, Bled, Slovenia, 1984.

34. I. Kononenko, "Machine Learning for Medical Diagnosis: History, State of the art and Perspective", *Artificial Intelligence in Medicine*, **23**, 89-109, (2001).

35. L.A. Kurgan, K.J. Cios, R. Tadeusiewicz, M. Ogiela, and L.S. Goodenday, "Knowledge Discovery Approach to Automated Cardiac SPECT Diagnosis", *Artificial Intelligence in Medicine*, **23:2**, 149-169, (2001).

36. L.A. Kurgan, and K.J. Cios, "DataSqueezer Algorithm that Generates Small Number of Short Rules", *IEE Proceedings: Vision, Image and Signal Processing*, submitted, 2003.

37. L.A. Kurgan, Meta Mining System for Supervised Learning, Ph.D. dissertation, the University of Colorado at Boulder, Department of Computer Science, 2003.

38. L.A. Kurgan, and K.J. Cios, Fast Class-Attribute Interdependence Maximization (CAIM) Discretization Algorithm, *Proceeding of the 2003 International Conference on Machine Learning and Applications* (ICMLA'03), Los Angeles, 2003, 30-36.

39. L.A. Kurgan, and K.J. Cios, Meta Mining Architecture for Supervised Learning, *the Seventh International Workshop on High Performance and Distributed Mining*, in conjunction with *the Fourth International SIAM Conference on Data Mining*, accepted, Lake Buena Vista, FL, 2004.

40. L.A. Kurgan, and K.J. Cios, "CAIM Discretization Algorithm", *IEEE Transactions on Knowledge and Data Engineering*, **16:2**, 145-153, (2004).

41. P. Langley, W. Iba, and K. Thompson, An Analysis of Bayesian Classifiers, *Proceedings of the Tenth National Conference of Artificial Intelligence*, AAAI Press and MIT Press, 1992, 223-228.

42. M. Laura, T. Weijters, G. Vries, A, van den Bosch, and W. Daelemans, "Logistic-based Patient Grouping for Multi-disciplinary Treatment", *Artificial Intelligence in Medicine*, **26**, 87-107, (2002).

43. R.S. Michalski, I. Mozetic, J. Hong, and N. Lavrac, The Multipurpose Incremental Learning System AQ15 and Its Testing Application to Three Medical Domains, *Proceedings of the Fifth National Conference on Artificial Intelligence*, 1986, 1041-1045.

44. T.M. Mitchell, *Machine Learning*, McGraw-Hill, 1997.

45. A. Paterson, and T.B. Niblett, ACLS Manual, Edinburgh: Intelligent Terminals, Ltd, 1987.

46. J.R. Quinlan, Learning Logical Definitions from Relations, *Machine Learning*, **5**, 239-266, (1990)

47. J.R. Quinlan, *C4.5 Programs for Machine Learning*, Morgan Kaufmann, San Francisco, 1993.

48. T.C. Redman, "The Impact of Poor Data Quality on the Typical Enterprise", *Communications of the ACM*, **41:2**, 79-81, (1998).

49. J.F. Roddick, and M. Spiliopoulou, "A Survey of Temporal Knowledge Discovery Paradigms and Methods", *IEEE Transactions on Knowledge and Data Engineering*, **14:4**, 750-767, (2002).

50. J.P. Sacha, K.J. Cios, and L.S. Goodenday, "Issues in Automating Cardiac SPECT Diagnosis", *IEEE Engineering in Medicine and Biology Magazine*, special issue on Medical Data Mining and Knowledge Discovery, **19:4**, 78-88, (2000).

51. M. Spiliopoulou, and J.F. Roddick, Higher Order Mining: Modeling and Mining the Results of Knowledge Discovery, *Data Mining II – Proceedings of the Second International Conference on Data Mining Methods and Databases*, Cambridge, UK, 2000, 309-320.

52. W. N. Street, O. L. Mangasarian, and W.H. Wolberg, An Inductive Learning Approach to Prognostic Prediction, *Proceedings of the Twelfth International Conference on Machine Learning*, Morgan Kaufmann, 1995, 522-530.

53. R. Wirth, and J. Hipp, CRISP-DM: Towards a Standard Process Model for Data Mining, *Proceedings of the Fourth International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, Manchester, UK, 2000, 29-39.

54. W.H. Wolberg, W.N. Street, and O.L. Mangasarian, “Machine Learning Techniques to Diagnose Breast Cancer from Fine-Needle Aspirates”, *Cancer Letters*, **77**, 163-171, (1994).

55. A.K. Wong, and D.K. Chiu, Synthesizing Statistical Knowledge from Incomplete Mixed-Mode Data, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **9**, 796-805, (1987).

56. B. Wu, T. Abbott, D. Fishman, W. McMurray, G. Mor, K. Stone, D. Ward, K. Williams, and H. Zhao, "Comparison of Statistical Methods for Classification of Ovarian Cancer Using Mass Spectrometry Data", *Bioinformatics*, **19:13**, 1636-1643. (2003).