University of Alberta

Data Mining in Databases:

An Extended Decision Tree Approach and Methodology in Database Environment

by

Sinisa A. Iliskovic ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Department of Electrical and Computer Engineering

Edmonton, Alberta

Spring, 2000

0-612-59976-0

University of Alberta

Library Release Form

Name of Author: Sinisa A. Iliskovic

Title of Thesis: Data Mining in Databases: An Extended Decision Tree Approach and Methodology in Database Environment

Degree: Doctor of Philosophy

Year this Degree Granted: 2000

S. Ilišković

7490 Brompton Road #377,
Houston, Texas, U.S.A.
77025

13 - APRIL - 2000

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and research for acceptance, a thesis entitled **Data Mining in Databases: An Extended Decision Tree Approach and Methodology in Database Environment** submitted by Sinisa Aleksandar Iliskovic in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

....

Pr. Dr. M. Meng

....

Pr. Dr. N. Durdle

....

Pr. Dr. J. Szymanski

B. Nowrouzian

Pr. Dr. B. Nowrouzian

Krzysztof Cios

Pr. Dr. K. Cios

....

Pr. Dr. Pedrycz

ABSTRACT

The work presented in this Ph.D. thesis is in the field of Data Mining and Knowledge Discovery. The research objective was the development of a new approach towards analysis and design of decision trees in a framework of databases. The decision trees are implemented in the database structure for purpose of analysis and better usage of the data sets that are constituent parts of the database. Various novel data mining ideas are realized in this approach including merging of different data sets, handling of missing values and database approach for pruning of decision trees. Other ideas and implementation findings are in the domain of further manipulation in the process of obtaining the decision tree structures and the rules induced from them by using the database features. Such findings are presented through the use of multiattribute class identifiers, terminal node proximity algorithm, predictive capabilities of obtained rules and structures and alternate branching of decision trees. Last, mathematical design and algorithm implementation of the three new different entropy functions used in this research, namely sinusoidal, parabolic and triangular are successfully implemented and given. The implementation of a decision tree is completed using database SQL (Structured Query Language) languages. This selection is motivated by their intended functionality when operating in the database environment. Research was performed on databases because only database structure and the data in it can provide real output and feedback about the usefulness and accuracy of the final results.

In the course of this research, extensive theoretical foundation with novel ideas mentioned above was created, with the algorithm implemented in the practical software solution. Large experiment and analysis volume was generated to prove and compare theoretical ideas and practical implementation.

This research provides for the unique combination of a theoretical and scientific approach with inherent database features and capabilities. It can be used as a foundation for further data mining research in databases by the use of decision trees and algorithms developed during the course of this thesis.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Professor Dr. W. Pedrycz for his help during the course of this research. In this time, which spanned over five calendar years he was more than a Professor and Advisor to me, he was my friend.

Also, special thanks to M.D. Ph.D. Natasha Iliskovic-Slater and Mr. Scott Maclennan for their help with this research.

This thesis is dedicated to three people-my parents Nade• da and Aleksandar, who being university professors at the engineering faculty, have always had useful advice for their son during this research; And to my wife Tracey who showed me that when will and positive attitude are combined, everything is achievable no matter how difficult the obstacles are.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

Introduction

1.1 Overview

Over the years, a need for large databases and a way to extract the knowledge held within them came into the spotlight of the professional community. Prior to the early 1960s, corporations were viewing their data as a legal burden, something that was difficult to maintain and without much use outside of keeping financial track of company business and transactions. In the 1960s, the realization came that the data itself can be used to attract new business and customers, to lower the cost of production, find better suppliers, compare trends in the overall health of the population over the decades, etc. Unfortunately, with the positives came the negatives, IT (Information Technology) professionals tried to create "ageless" systems that would store the data and would never be replaced. These "misconceived" systems proved to be impossible to maintain and update, lowering everybody's desire for any future work and research on data. In the 1970s, realization of the right direction and positive changes came to the IT world and computer engineering and computer science in general. Systems with realistic life cycles, relational databases, and data analysis in the form of early expert systems and decision support systems. Today, thanks to the early pioneers and their foresight, opportunities to analyze data and simulate outcomes are almost limitless. When we compare current versions to previous versions of software and hardware architectures, it is obvious why. Storing and retrieving of data has become faster, more efficient and more economically viable. After the quest for better and more powerful databases was accomplished, the next question came. How can we use these databases to exploit all of the new features for the benefit of mankind?

In the last 5 years many companies such as Oracle, Cognos, Business Objects, Hyperion and Sybase started to develop tools which supported extended database functionality. The biggest disadvantage of these tools was that they were designed to perform only specific work or analysis for a certain type of customer. The difference between the data mining tool designed from this research and currently available tools is that the former will perform on any database, delivering high level and quality results, whereas the latter would stop at the basic query level.

Decision trees occupy an important position in the space of data mining. In the next chapter, decision trees are explained in depth together with all of the specifics, its historic evolution and its algorithm. For better understanding of this introductory part let us recall that a decision tree is a tree-like structure with nodes that correspond to the attributes encountered in the data set and used to construct the decision tree structure(s). The branches originating from the node correspond to the values assumed by the attributes.

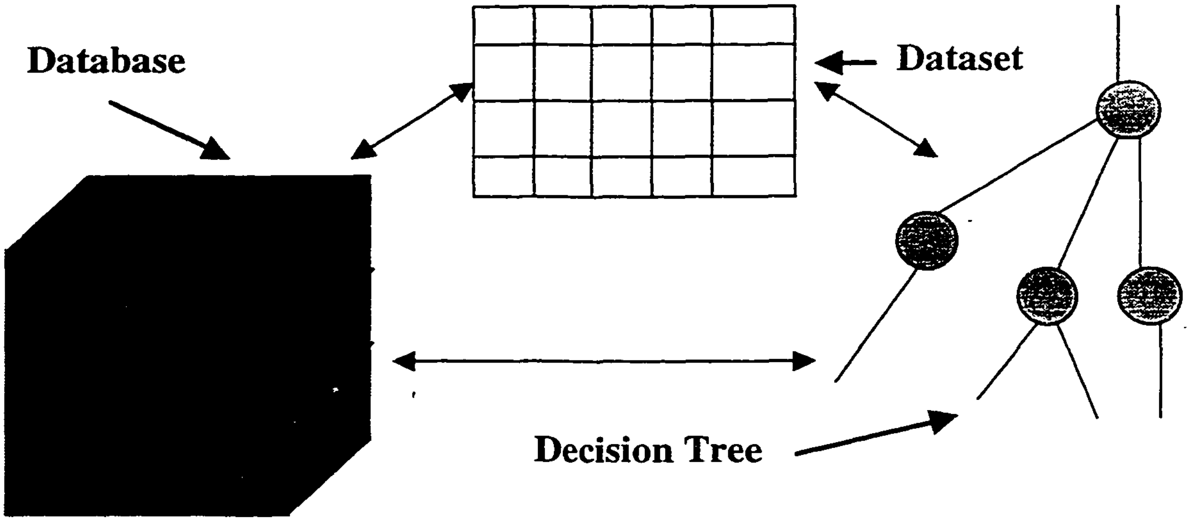


Figure 1.1 Relationship among databases, decision trees and data sets

As can be seen in Figure 1.1, a database consists of multiple data sets. A data set can be described by a decision tree constructed from its data. Data sets can also be reconstructed based on the rules extracted from the decision tree. After classification of the rules is complete, attributes deemed as weakest by the algorithm can be eliminated from the data set, hence lowering the size of the data set and number of the attributes (columns in the data set).

As mentioned, decision trees will be addressed in more detail in the following chapter. Now it is important to mention that their design can be applied to assume a general rule for varying data sets, consisting of different attributes. Also, the aspect of multi-class decision trees is considered, as one among many useful features of data mining. We will be referring to such multiclass trees as multidimensional trees. Multidimensional trees and their usage in the field of data mining prove to be very useful and an exciting enhancement to the research field itself.

Even though work outside of general (from the point of view of database research) performed by Quinlan [126, 128, 133]], had some yet very limited impact on the use of databases [89], there were still many questions and options to explore. The data sets used were of insignificant size. The problem that researchers faced was the complexity of the created decision trees. Also, the datasets were stored in the fashion of ASCII files, as opposed to tabular database structures.

Kersten and Holsheimer [89] were amongst the first who performed research on the data set using a SQL database for the purpose of rule induction and knowledge discovery. This was one of the first research attempts in this direction, performed in 1994. There were other articles and research published at that time with similar general direction and philosophy, such as work done by Brachman and Anand [21] that provide a somewhat high overview of what could be labeled as the pioneering approach to Datawarehousing and Datamining henceforth.

1.2 Scope and contributions

The approach taken in this research fully incorporates the advantages of modern databases, introducing new computational options in this field. The database tools used for this research are an Oracle 8 database and Oracle's PL/SQL language [157] that is a combination of 3GL and 4GL languages. PL/SQL (Procedural Language/SQL) has most of the functionality of a fourth generation language (4GL), which describes what to do but not how to do it. This is combined with the enhanced options of dynamic SQL and the nesting-looping structures of programming languages like C. Using this approach, a data set of any size can be analyzed without any restrictions, which is a novel idea in the field of data mining with decision trees. Also, the SQL database language has an advantage over normal programming languages in that it exploits the database structure to the fullest since it is designed for manipulation of data within databases. The intermediate steps and results are stored within the database structure, enabling the researchers to make any alterations to the testing patterns undertaken or direction change in the testing algorithms itself.

The importance of this research can be seen from the point of view of current industry, where there is a high demand to analyze and comprehend multidimensional data with linguistic, easy to interpret textual representation. Using databases and database tools, eliminates the narrowness experienced in the pioneering days of data mining when results were unclear even in scientific circles. Ways to perform data manipulation of very challenging data, such as highly complex datasets, large data records, and reusability of previously processed data sets (decision trees) can be achieved with databases and database tools. Combined with the innate computational ease of database SQL languages, this is certainly a new approach in the field of data mining.

The objective of this research is to create a data mining method for databases using a decision tree approach. One should underline that data mining is a highly heterogeneous pursuit embracing a number of fundamental approaches including statistics, pattern recognition, regression analysis, decision trees, neural networks, to name a few. More about the aforementioned methods will be discussed and explained in the following chapter.

The challenges and questions posed and successfully addressed in this thesis about decision trees being used in the context of databases are:

- How does the size and complexity of the data set (dimensions of objects) affect classification rate and building performance of decision trees?
- Can these database-driven decision trees live up to the expectations set both conceptually and computationally by previous work?
- Can all of the novelty and originality introduced and conceptualized in this research be leveraged and used in the database environment?
- What different results can be expected from the use of different entropy functional?

- Is the implementation of a multiattribute class algorithm viable with respect to its possible design complexity?
- Are all of the algorithms and novel ideas going to perform within the expected boundaries?
- How can the database centric decision tree be further manipulated based on its permanent structure captured within the database?

The originality of this research lies in the domain of enhancements of decision trees and new algorithms used within databases, namely

- ***Normalization and unification*** of data sets with different attributes using decision trees. Merging of different data sets and data structures.

- Investigation and design of ***different forms of entropy functional.***

- Handling of ***missing attributes*** using database features and tools with the decision tree like approach.

- New domain knowledge methodology for data sets, both ***partially guided*** and ***fully guided relevance*** of attributes and rule complexity calculations.

- Pruning of decision trees with the use of ***data set size criterion*** and ***one layer method***.

- ***Multi attribute class identifier*** based on all the possible combinations of selected attributes. Composite class identifier based on the mathematical permutations of chosen attributes and their values.

- The ***alternate branching*** of decision trees, in the case of nodes where the information gain between two or more competing attributes have the same information gain.

- ***Likelihood of terminal nodes*** and manipulation of decision trees using database capabilities and tools. Altering structure of decision trees by using the database approach and capabilities.

Main contributions are,

- Design of enhanced decision trees with the new type of entropy functional algorithms.
- Routines for missing values replacement that closely preserve data structure as given by known values within the data set.
- Possibility to manipulate and store the entire decision tree structure, provide enhanced data mining and knowledge discovery capabilities.
- Different type of decision trees built with respect to structure and complexity, also questions that could be asked and answered with the use of the decision tree, by leveraging the multiattribute composite class identifier.

- General research enhancements and contributions for future research that can be done by using the attribute relevance and rule calculations, two database oriented types of decision tree pruning and alternate branching approach with weighted rules.

1.3 Organization

This thesis is organized in the following manner. In Chapter I as seen from above, general terminology and underlying concepts used and explored in this research are described with the introductory statements about historical evolution of databases and data mining in general. Research issues, statement of originality and main contributions are also formulated, with the overtone on the data mining utilization with decision trees in the space of databases and datawarehouses. Some of the specific directions and enhancements that this research has created are also mentioned in a guiding and involving manner.

Chapter II provides a historic overview and discusses the most important points in the research done previously in this area. The explanations and definitions around databases-datawarehouses, programming languages and decision trees is given here. Data mining techniques and discretization techniques are also described and presented in this chapter supported by the necessary figures and examples. Examples and overview of the Relational Database Management System (RDBMS), its impact on the user community and specific rules around design of RDBMS are presented. All of the necessary facts and information around RDBMS are shown, so that readers even if not fully familiar with the concept of RDBMS can realize and comprehend its space. Next, specifics about decision trees and its enhancements done by the researchers are addressed and discussed. Namely, performance of decision tree discussion, several methods of decision tree pruning and the general guidelines around missing values and their replacement in the datasets used for data mining research. At the end of Chapter II, a separate literature overview section is given after all of the necessary terms and notions are already explained.

Chapter III elaborates on the key research objectives, and incentives for undertaking this research. Each of the key objectives and novel ideas are explained in detail in the chapter, with visual support from the numerous figures capturing process flows and diagrams of methodologies and algorithms developed, implemented and tested during the course of this research. Key objectives presented in this chapter are multiclass decision trees, multiclass (multi attribute) identifiers, merging of data sets with the use of decision trees, different forms of entropy functional, pruning of decision trees, handling of missing attributes, proximity of sub trees and terminal nodes, manipulation of decision tree substructures and intermediate nodes and alternate branching of decision trees.

Chapter IV presents the results of experiments obtained after the ideas from Chapter III were implemented using database tools and SQL language. Chapter V discusses and analyzes the results obtained in Chapter IV in detail. Analyses entail computational accuracy, speed, complexity, and other manipulative attributes and qualities of decision trees such as pruning efficiency, alternate branching and all of the database specific trial result analyses. A comparison and analysis of the results with respect to the above are given in Chapter V in textual, tabular and where necessary figure and graph form.

Recommendations for further research and enhancements that could be performed with the ideas and algorithms developed, implemented and tested in this thesis are given in Chapter VI.

CHAPTER II

Theoretical Fundamentals

In this chapter, we introduce a series of definitions and pertinent terminology used in the research. We start with a detailed introduction to the area of Knowledge Discovery in Databases and Data Mining, in particular. Next, the concepts of relational databases and associated tools are explained, along with the terminology involved with data mining and decision trees.

2.1 Data Mining: making sense of data

Everyday business and industry are faced with a flood of data. As a matter of fact, this becomes the most evident sign of the ongoing information revolution. Information is an important commodity. It comes with a genuine challenge. Just to name a few of the problems one has to tackle, let us refer to some evident facts:

- Wal-Mart completes around 20 million transactions per day
- The planned NASA earth observing system to be launched around 1999 will generate 50Gb of image data per hour
- The rapidly expanding information superhighway will require advanced tools (intelligent agents) for mining data

Indisputably, we live in a society that is data rich and knowledge poor. Meaningful efforts to distill and interpret revealing relationships become a must. Quite descriptively, a mission of knowledge discovery is to make sense of data.

The term knowledge discovery is commonly defined in the literature by,

"Knowledge discovery in databases is concerned with identifying interesting patterns and describing them in a concise and meaningful manner " [147].

If we are primarily concerned with the process of revealing patterns rather than concentrating on the underlying mechanisms of interaction with databases (and the ensuing optimization machinery including a specialized query language), then we refer to this activity as data mining (DM). For the recent overview of the area of knowledge discovery, the reader may refer to a comprehensive summary by Fayyad et al. [69] as well as a special issue of Int. J. Intelligent Systems [152] and the Communications of the ACM [45].

First, we provide a brief yet highly comprehensive overview of the area of knowledge discovery and data mining, revisiting the fundamental concepts and the resulting architectures. This section makes the paper self-contained as to the underlying concepts, methodologies, and algorithms of data mining pursuits.

A general architecture of knowledge discovery and data mining architecture

As already introduced, knowledge discovery (or knowledge discovery in databases, KDD, for short) concerns nontrivial processes of identifying patterns [*] in data that are of value to the user. This means that they are valid, novel, potentially useful, and easily understandable. The general architecture embracing main functional phases of KDD is outlined in Figure 2.1. As illustrated in this figure, the overall KDD process splits into a number of essential phases:

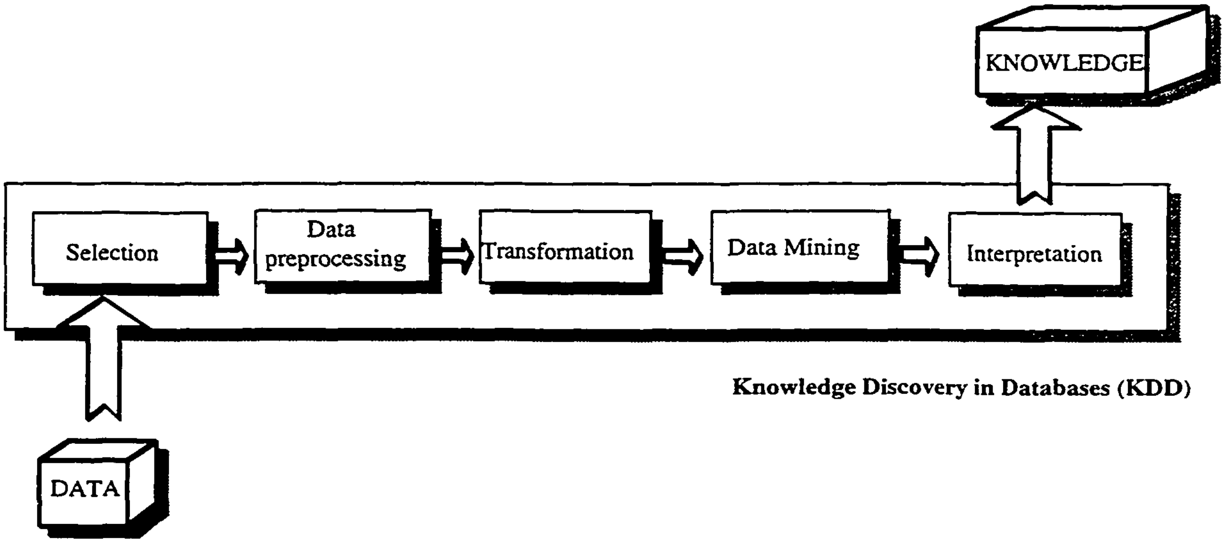


Figure 2.1 Knowledge discovery in databases - a general architecture and its main phases

- selection
- data preprocessing
- transformation
- data mining
- interpretation

Let us briefly elaborate on each phase of the process.
We start by identifying and understanding the application domain, identifying the goal of all activities (as expressed by the group of users), and collecting prior domain knowledge about the problem. This includes a detailed description of data and ensuing databases (along with their format, access mechanisms, etc.). These activities give rise to a more detailed understanding of the problem and an identification of the essential variables that are believed to be crucial to the problem along with some common sense qualitative hints as to the general relationships occurring in the problem. This phase of the KDD process leads to the formation of a target data set (database) that results from the application domain by focusing on a subset of data (sampled data) or a subset of variables in the data set.

[*] Pattern in KDD can be defined as relationship, association or any rule inducted information among two or more attributes.

Data preprocessing and cleaning is applied to focus on noise removal, identify time sequences, and handle missing variables. Subsequently, we proceed with data reduction and projection (data transformation). This concerns finding useful features that effectively reduce the dimensionality of data. The data mining activities follow this phase and focus on the formation of patterns. Their format, specificity (level of details handled), and methods pursued depend on the main goal of KDD. The back-end of the KDD process concentrates on the evaluation and interpretation of the DM results. Again, the user is provided with tools facilitating this phase [120].

The KDD process is usually iterative and requires considerable interaction with the user. This results in many feedback loops at different depth embracing one or more phases of the entire KDD process.

There are different ways of formulating the DM problem. In particular, an anticipated level of autonomy encountered in the ways of sifting through data becomes an important criterion. Let us consider several typical DM scenarios

- Tell something interesting about the data.
- Find interesting associations in the data.
- Describe data in terms of some concise functional dependencies existing between variables.

These three categories of tasks constitute a hierarchy of problems with respect to their generality. The first scenario alone (at the most general level) is an ideal DM architecture one should strive for eventually. Unfortunately, this target is difficult to implement. What becomes interesting to the user is not obvious. It is very likely that the patterns revealed (established) by the DM system may not be that relevant to the user. First, the patterns could be trivial and well known (even though well supported by experimental evidence conveyed by the database under study). Second, it could well be that the most suitable representation (rules, temporal patterns, correlation) is unknown. Third, the desired level of detail is not known in advance.

In comparison to the first category of the DM tasks, finding interesting associations in the data, forms another important but less general and more manageable category of pursuits. The primary target here is to establish and quantify relationships between variables encountered in the database. Two common ways of quantifying associations include correlation (in the form of correlation coefficients) and relations (being either two-valued relations or fuzzy relations).

Describing data in the form of some functional dependencies results in a more detailed class of patterns in data. These dependencies could be either linear or nonlinear. In the first category of dependencies, we may recall all regression models along with a vast number of their identification schemes. In contrast, neural networks are interesting examples of highly nonlinear and easily adjustable mappings. It is worth mentioning that functional dependencies are subsumed by associations (that is, variables can be related but not necessarily imply each other).

It should be stressed (which, unfortunately, has not been underlined strongly enough in the DM area) that there is a fundamental difference between associations and functional dependencies. Associations are direction-free constructs and capture relationships between the variables, yet they do not make any explicit assertion as to this direction (what is implied by what). In contrast, functional dependencies (as suggested by the name itself) are mappings from many variables to another variable in the problem. By doing so, we confine attention to some specific direction of the mapping by stating that a certain variable is implied (predicted) by the others.

All nontrivial tasks of DM are integrally linked with the notion of interestingness that arises as an important design factor to be taken into consideration in any implementation. Any meaningful quantification of interestingness is a nontrivial task. Striving for discovering interesting patterns is regarded as a focal point of the design pursuits and can be accomplished through the feedback loop involving the group of potential users of the DM system. One of the ways to raise the level of interestingness is through adjusting granularity of information and usage of the resulting information granules as generic building blocks around which all DM activities tend to revolve.
It becomes apparent that DM is subsumed by the overall process of KDD and concentrates primarily on the algorithmic issues of revealing patterns in data. The front end of the KDD includes activities that are inherent to databases and database mechanisms, including various access mechanisms (e.g., query languages) and their optimization. These database-driven activities also appear under different names such as data warehousing and on-line analytical processing (OLAP). On the other hand, the back-end of the KDD process is associated with various visualization tools.

Main technologies of data mining and their synergies

There is no doubt that there are a number of essential information technologies that contribute to the concepts of data mining and their architectures. The underlying nature of data mining stipulates a synergistic use of such technologies. Those are the same synergistic mechanisms that have driven the establishment and progress in the area of computational intelligence (CI) [108]. The list of key technologies includes the three primordial entries:
- neurocomputing
- evolutionary computing
- granular (fuzzy sets and rough sets) computing

These technologies are rarely used in their pure format in system design. In most applications, systems are designed based on an interaction between fuzzy sets, neural networks, and evolutionary methods. Based on the synergy manifesting between the technologies, we encounter such important categories such as neurofuzzy systems, neuroevolutionary networks, and so forth. Many of the useful ideas and interpretations of previous theory and past research can be seen from numerous works in the field of data mining and KDD [5, 6, 10, 11, 19, 22, 27, 28, 31, 34, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 52, 59, 60, 63, 73, 77, 78, 79, 83, 84, 85, 87, 95, 106, 111, 112, 117, 119, 127, 129, 130, 131, 132, 134, 135, 136, 137, 140, 144, 150, 151, 152, 160, 161, 163, 166, 158, 159, 168, 170, 171, 174, 175]. Worthy of noting, is that some of the numbered references

above are not directly related to the present trends in data mining and KDD as a part of its current research, but as the foundation from past research in the more general fields of analysis in mathematics, probability and statistics [59, 60, 84].
Some of the most widely used and known analytical methods in the field of data mining are addressed next.

Data Mining Techniques

Predictive Regression Analyses-Used normally in the scenario where some prediction has to be made about the numeric values of variables within the research space selected in the data set. By using this approach, a best line fit to the data points would be observed [105, 109, 112, 153]. As the outcome of the best line fit approach, a mathematical function describing the best fit through the data sample would be created as well as a correlation measure, which would provide the insights about variance in the data with respect to the best line fit. In the case of household income and education as the variables observed the following could be concluded: there is a strong correlation with the low variance of data points surrounding the best-fit line. Also, the line is at approximately 45 degrees showing a strong relationship between two variables.

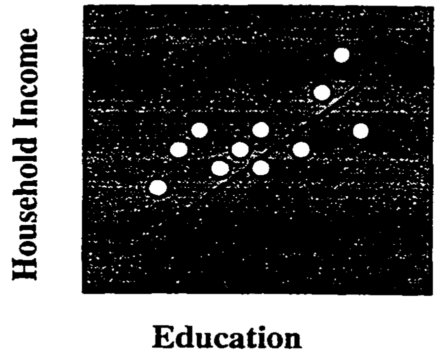


Figure 2.1.a Household Income-Education Best Line Fit

In the case of height of the individuals and their chess rankings as the variables observed, the following could be concluded, that there is no predictive link between the two variables and correlation is very low with a high variance. The line is almost horizontal showing no relationship between variables.

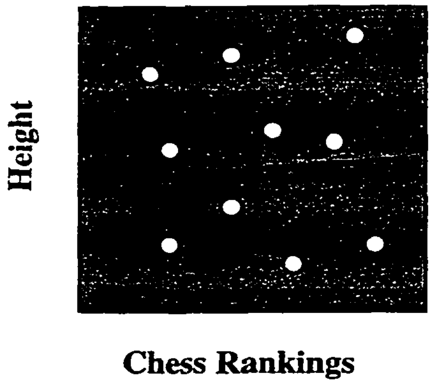


Figure 2.1.b Height-Chess Rankings Best Line Fit

For the above method to be used successfully, some criteria have to be met. Data points observed have to be discretized numeric values. To achieve this, grouping of values has to be performed and any textual description of data points and their values has to be replaced with numbers. Predicted goal of the analyses should exist beforehand. Differences and facts that are to be explored should be anticipated prior to grouping of the data (discretization). Ratio between variance and a selected data sample has to be carefully chosen to avoid any misbalance between the two. Larger data sets will tend to lower the variance in a general sense, when dealing with meaningful data and the hypothesis presented with it.

Association Rules-Type of analysis that derives information from correlations or co-occurrences of transactional events. Definition of the transactional event: car purchase, new suit, income, etc. Anything that can be correlated with some other event(s). Association rules are derived by calculating the likelihood [51] of each observed event occurring in conjunction with every other event observed in the matrix. The matrix itself is considered to be n dimensional but like any other data mining algorithm, the number of dimensions increases computational overhead almost geometrically. Association rule analysis is very useful when performing exploratory research, and looking for interesting but not always so visible relationships among events (data points) within the data set. A problem that might occur while using this method is that discovered relationships are not either meaningful, important or both. In the example below the cross-correlation matrix with four transactional events is presented. In the grocery shop there is a high likelihood that bread will be purchased with milk, fries with meat, etc.

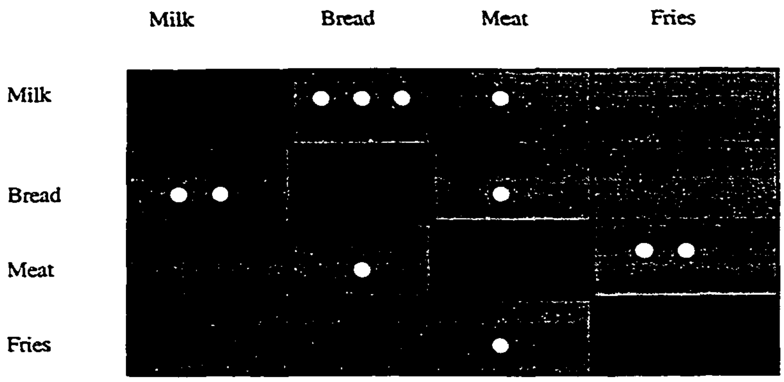


Figure 2.1.c Cross-Correlation Matrix for Groceries

A technique of particular interest for research presented here is also known (or at least very similar to) as ***rule induction*** [147]. Rules are conditional statements of the form "*If* condition *then* conclusion". The following features can characterize the rules:

- *Discriminating* abilities of the rule (that describes how well it predicts the outcome).
- *Generality* (how often the rule predicts over a certain time period, part of the database, and different decision making requirements)
- *Interestingness* (it pertains to whether the rule covers an interesting finding that a user working on data mining deems useful in his decision-making).

Neural Networks-mostly used in the world of pattern recognition and classification. Neural networks are comprised of nodes that are connected with excitatory and inhibitory types of connections. Information is presented in the distributed fashion in the NN [39, 94, 146] (neural network), which means that particular information resides through the pattern of activated nodes, and not necessarily the node itself. There are two general different types of learning that can be undertaken in NN,

- Supervised learning, where the input is given and the output feedback is used to correct the incorrect response and modify the connections and their strength between nodes. Most of the back propagation networks are based on the supervised learning, but data mining itself does not benefit from it, since no new discovery is taking place. Applications like these are useful as mentioned in pattern recognition space.
- Unsupervised learning, where NN forms its own set of outputs during training stage based on the results obtained from the network. During the learning stage, nodes with the strength closest to the input provided a stay in the process, while the weaker nodes are discarded from future consideration. Normally, process continues until a single output node is chosen as the eligible neighborhood by itself. Data mining usage of unsupervised learning can be in the discretization of classes and inputs prior to using some other algorithm for further analyses. NN would perform data grouping by itself without the user specifying any input parameters.

NN in the data mining world, do have a tendency to suffer in the case of large size data sets, because extensive training has to be performed on NN to achieve acceptable results.

Unlike decision tress, a NN has to be properly trained which proves to be a problem in very large databases and datawarehouses. Also, NN results cannot be always easy to interpret and understand. Decision trees on the other hand do not experience any problems with the data set size and selection, rules are easy to interpret and understand. Applications where NN excels amongst others are specific pattern recognition in the cell biology, disease recognition from various samples, etc. In the past decision trees were memory bound, but with the methodology proposed and developed in this research in the following chapters and better hardware systems, memory limitations and storage limitations in general are resolved. In conclusion, computational complexity, limited usefulness of supervised learning based models such as backpropagation, size of the data and interpretability of results, limitations far outweigh the storage and memory constraints present in decision trees in the past (as explained, these problems are resolved mostly with superior hardware now available).

The table below provides the classification comparison between the NN and decision tree on the same data set. NN software code and the observed data set (Wine data set) was provided from the demo software by Ward Systems Group, Inc. from their web site at www.wardsystems.com. The Wine data set consisted of 13 attributes, 178 records and the identifier with three classes in it. The number of inputs for the NN was therefore 13, with the learning algorithm of classical back-propagation NN used and 80 hidden neurons in the network. The inputs were the attributes important to the production characteristics of wine.

Method	Training set classification %	Test set classification %	Number of nodes (DT))/hidden neurons in entire NN	Number of classes
Decision Tree	100	100	40	3
Neural Network	100	97	80	3

Table 2.1 Classification and complexity comparison for NN and decision tree

The data set performed with a very high classification rate for both NN and the decision tree. The decision tree was of low complexity considering that the overall number of nodes in the tree was 40. The decision tree did have a higher classification for test data, but both methods performed with a high classification rate. Considering that this data set was provided and prepared as a demo data set for NN software and that prior to the data mining with the decision tree, a simple uniform discretization was performed without any special considerations, the results obtained with the decision tree are very promising. Also, as explained before the interpretability and understandability of decision tree rules, it was easier to follow and comprehend. Considering the other advantages named and discussed in this and other sections, especially in the context of large databases and data sets, the direction taken in this thesis and the exploration and novel approach taken on decision trees in the database environment is even more promising.

Genetic Algorithms-mostly used in the cases of optimization where a research space can be readily defined [49, 50, 72]. Due to the constraint that data has to be fairly uniform to perform successful analysis, genetic algorithms do not perform that well against data combined from several dissimilar data sources and/or types of data. These algorithms could be made to work well with NN in the space of deciphering solutions produced by NN for easier understanding.

General steps of genetic algorithm are:

- **Selection**, where natural selection occurs among population.
- **Crossover**, where two specimens are chosen and combined in such a way that the resulting "child" contains partial information from both "parents".
- **Mutation**, where perfect information imprint received from crossover can be distorted and research space enriched in general, by a small amount of mutation change.

Numerous algorithms and techniques can be used in the space of data mining, but as stated before, decision trees are chosen as the technique for this research. In the research space of very large and diverse databases with multiple and multiclass attributes that cannot be always easily discretized, due to either their nature or computational overhead that some other methods would produce, the decision trees are a logical choice. More about decision trees and a detailed explanation about their design and theory is provided further in this chapter.

Data Grouping and Discretization Techniques

Common problems in the space of data grouping and discretization, are deciding on the size of the group (granule) and the number of distinct groups (granules) in the data set. The process of grouping information within data sets should be performed in the way that meaningful grouping of individual elements is performed so that commonalties (either functional or descriptive) among elements exist [66, 67, 68]. The group of elements created in this fashion can be labeled as the Information Granule. Under the information granule we can include probabilistic sets, fuzzy sets, rough sets, etc. The rationale for using information granularization (grouping) is twofold

- Computational-reduced amount of computation time and amount of data.
- Conceptual-data and relationships within described in the more concise way.

Grouping of the data can be performed as uniform or non-uniform.

In the uniform grouping of data, groups (granules) of the same size are created and individual elements are "sliced" across the data set among granules in linear order.

In the non-uniform grouping of data, groups (granules) that could be of different sizes are created to better capture commonness amongst the individual members that are constituents of the particular group (granule). Some of the methods, such as NN from the section before can be used for better defined non-uniform grouping of input values in the data set(s) being observed in the process of data mining.

At this point, definition of data discretization should be clarified. For the purpose of better and more efficient computation on the data set, attributes within data set that consist of multiple values creating continual information in it, are good candidates for process of discretization. In that process, by using uniform or non-uniform grouping, continuous intervals of data are grouped together and replaced with a discrete value (normally numeric value). For example, in the data set in which one of the attributes is age, continual range could be anywhere between 0 and 100 years of age. In the large data set with this attribute, computational overhead on any datamining algorithm would be high due to the attribute(s). By performing uniform or non-uniform grouping (based on the preference and knowledge about the data set), few groups of data values could be created and each group assigned a discrete (numeric) value. For example, in the above-mentioned attribute (age), 5 groups of uniformly grouped data (0-20 years, 20-40 years, etc...) could be created and assigned numeric values from 1 through 5 respectively. This in return would greatly reduce the computational time and data redundancy.

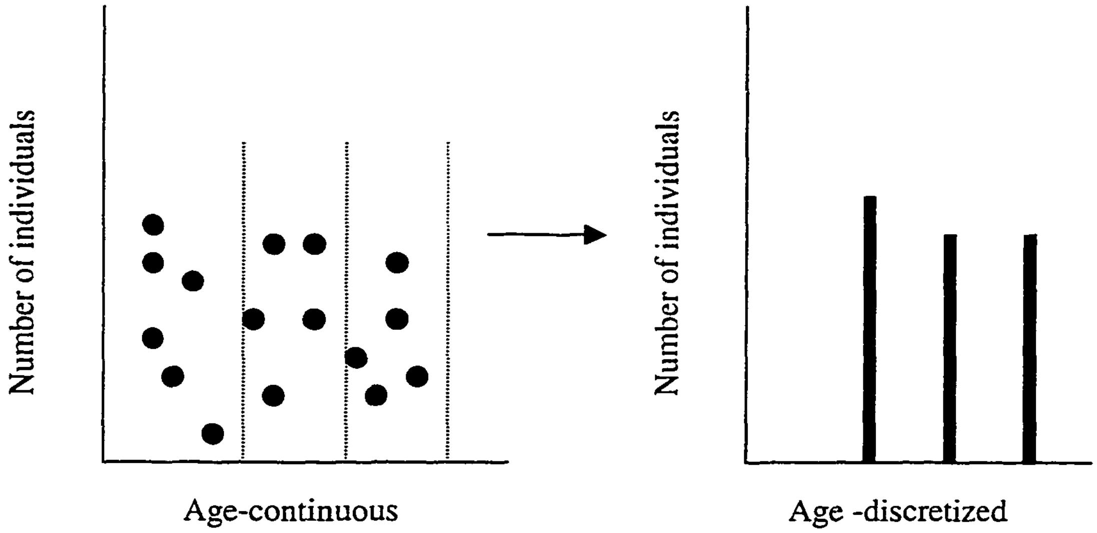


Figure 2.1.d Age attribute-before and after the discretization

All of the above data grouping can be performed as fully supervised, partially supervised or unsupervised (clustering).

- Fully supervised-boundaries and classifiers (qualifiers) are defined forehand.
- Partially supervised-some of the parameters (classifiers) are defined prior to grouping.
- Unsupervised-except for a minimum number of clusters desired or a certain performance index being minimized in the process of clustering, no parameters are defined prior to clustering.

Unsupervised-two main approaches

- Hierarchical clustering
- Objective function-based clustering

Hierarchical clustering can be done in two ways, agglomerative and divisive.

In agglomerative, massing or grouping of smaller clusters of data is performed based on certain criterion. For example, if criterion is a minimum distance between clusters, then the clusters with the minimum distance between them are merged together until either desired number of clusters is reached or minimum distance criterion is exhausted amongst remaining clusters. The starting process of this approach could be to define every individual element in the data set as a separate cluster and then perform the clustering process.

Divisive clustering would take the opposite approach, by massing the entire data set in the initial cluster and then perform grouping in the smaller clusters until desired the number is reached or clustering criterion exhausted.

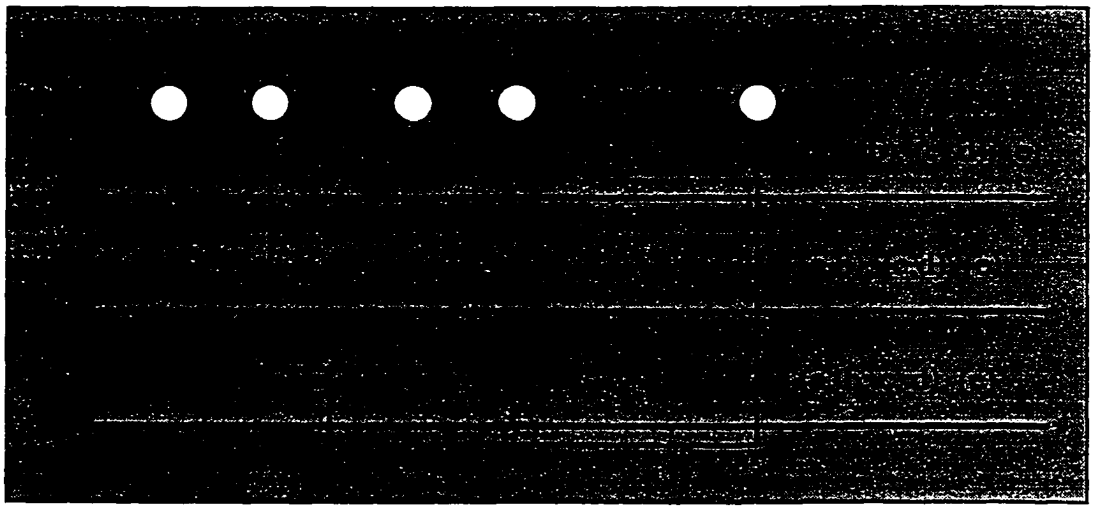


Figure 2.1.e Hierarchical clustering-dendrograms

Dendrogram-a diagram of evolutionary (clustering) changes (groupings) in which the descendant forms are depicted as treelike branchings. From the above dendrogram, five clusters were merged into two clusters, until minimal number of clusters was reached using the minimum distance between clusters criterion for the clustering algorithm.

Objective function-based clustering [18, 115, 116, 117, 118, 160], the approach is to develop and optimize a partition matrix so that a certain defined performance index can be minimized (maximized). For example, entropy based clustering would group adjacent clusters if doing so improves the overall entropy gain of the data set. Clustering would continue until certain criterion is met (in this case, a certain value of the entropy gain).

How do the different ways of data discretization affect decision trees? The different ways of grouping data would create different attribute-value pair distributions. Since the decision trees are being built using the entropy functional which captures the information gain and distribution of discretized data points, the results of decision tree design and the rules obtained would vary from one discretization technique to another. The results for two different techniques could be similar as per the design, accuracy and complexity of decision trees, but most commonly they would exhibit noticeable variations.

The data set used for this example is the Liver data set obtained from the University of Irvine site [156] in the modified format. This data set has seven attributes and a class identifier with two and three classes. More information about this data set is given in Chapter IV section 4.1.

Three different types of discretization are presented here, uniform discretization, professional discretization and fuzzy discretization as provided in the work by Pedrycz [118].

Uniform discretization:

For attribute with the continuous range of values-N and desired number of groups-c, simple formula can be assigned that would perform the uniform grouping,

$$Y(x) = i, \quad i = n,\ldots,c \text{ where } \frac{(i-1)\cdot N}{c} \leq x \leq \frac{i \cdot N}{c} \tag{2.1}$$

which would assign the group value of i to any qualifying continuous attribute value of x. n is the initial value of the cluster, user assigned.

Professional discretization:

For the attribute with the continuous range of values-N, the end user would choose a number of attributes groups (discretization points) and assign the ranges of attribute values of x based on the expert opinion about particular attribute's nature and behavior. In short, each of the attribute's continuous values would be assigned a discretized value based on the expert opinion about most appropriate ranges and grouping intervals.

Fuzzy discretization:

This particular method and formulas are provided as per Pedrycz [118], Given: The number of clusters (c). Select the distance function ||.||, termination criterion e (>0) and initialize partition matrix U ∈ U. Select the value of the fuzzification parameter "m" (the default is m=2.0)

1. Calculate centers (prototypes) of the clusters

$$v_i = \frac{\sum_{k=1}^{N} u_{ik}^{m} x_k}{\sum_{k=1}^{N} u_{ik}^{m}}$$

i=1, 2, ..., c

2. Update partition matrix

$$u_{ik} = \frac{f_k}{\sum_{j=1}^{c} \left(\frac{\| x_k - v_i \|}{\| x_k - v_j \|} \right)^{2/m-1}}$$

i=1, 2, ..., c, j=1, 2, ..., N

3. Compare U' to U, if termination criterion ||U' - U|| <e is satisfied then stop, else return to step (1) and proceed with computing by setting up U equal to U'

Result: partition matrix and prototypes

Table 2.2.a Steps of fuzzy clustering-as provided by Pedrycz [118]

ALKPHOS-liver enzyme (normal)	ALKPHOS-uniform	ALKPHOS-professional	ALKPHOS-fuzzy
92	2	2	2
64	2	2	1
54	1	2	1

Table 2.2.b Part of discretization values and the original values for ALKPHOS attribute-Liver data set

Training set classification %	Test set classification %	Number of final nodes	Number of classes	Type of discretization
71	68	36	2	uniform discretization
79	78	39	2	fuzzy discretization
72	72	36	2	professional discretization

Table 2.2.c Different discretization of data and resulting decision trees performance

Based on the data set and the method used, each decision tree built will have somewhat different characteristics with respect to training and testing data accuracy and the complexity of the decision tree designed. Each discretization method can perform better or worse than the other based on the data set chosen and random split between testing and training data. More of the discussion and analysis of discretization techniques and their influence on certain results are provided in section 4.1 of Chapter IV and throughout Chapter V, analyses and discussion of results.

2.2 Databases and SQL Language

The use of computer databases started 40 years ago. Many different formats of the database were used and abandoned until late 70s when the concept of the Relational Database Management System (RDBMS) [30] was introduced.

Early databases were a poorly organized collection of flat files. These flat files were stored on the hard drives and accessed using the drive access mechanism of the computers. These early data systems (they are not even addressed as databases in some of the reference materials) had numerous problems to cope with:

1. Relationships among data could not be maintained. Relationships such as one-to-many and many-to-many for example could not have been maintained. This created denormalized structures that are not suitable for Online Transaction Processing (OLTP) databases or for any enterprise size database.
2. Lack of uniformity and structure, which as the outcome created numerous stand alone databases and flat file systems (departmental, divisional, store by store, etc.)
3. Redundancy of data and repetition of the same data across various flat files structures.
4. Maintenance issues with non-centralized and redundant data.

As a solution to overcome most of the problems mentioned above, IBM created the first commercial database, namely IMS-Information Management System. IMS is still in use today although its presence is diminishing quickly. The problem with IMS was that even though it was more advanced than the flat file structure, it still could not handle complex data relationships (many-to-many for example) and development and administration of IMS databases was a slow and cumbersome process. In the highly normalized data structures with non- complex (natural) hierarchy, IMS systems are still existent and perform with very high speeds. Since there is much more to data handling and storing than purely sequential, non-complex data, the need for a more sophisticated and efficient database had arisen. In the late 1970s, Committee on Development of Applied Symbolic Languages-CODASYL was formed with the task of creating a model for databases. The model developed was named "Network Model for databases" and covered topics such as

- Framework for a Metadata Dictionary. Metadata Dictionary-is a data dictionary about data, relationships among data, programs interacting with data, etc. It is data about data itself.
- Standard architecture for network database systems
- Differentiations between a logical and physical database structure. Physical location of data should be transparent to the user or programmer if chosen so.

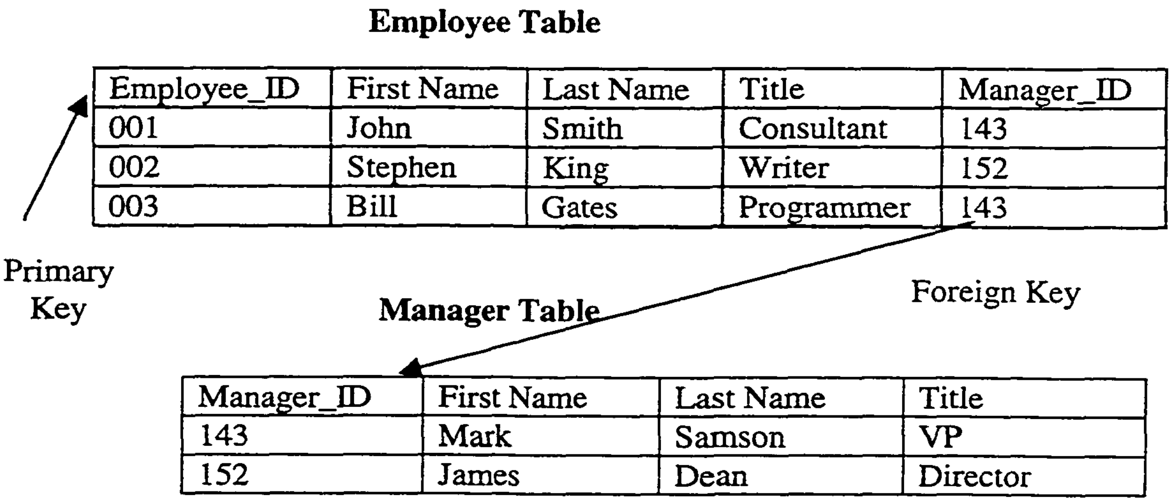
Although it was an improvement over the IMS systems with a capability to handle complex relationships among data, it still did not address the issue of easy maintenance, access of data (its transparency delivery method) and development ease.

Dr. T. Codd (IBM research lab), developed a model in which data resided in a table with no pointers. That enabled a navigation of data structures without a real need for knowing the physical layout of the database. Dr. Codd named these data sets (tables) 'relations'. The relations are two dimensional arrays consisting of rows (records) and columns (attributes).

This revolutionary idea was to use relations amongst entities and data sets to lower the amount of data entered and to increase the speed of data retrieval by avoiding duplicate entries. Later on, numerous enhancements came, such as proper index usage, efficient database engine and development of SQL languages [30, 157].

Improvements over earlier database structures were

- Contiguous record storage-speeding the retrieval of records and lowering I/O of the computer.
- Easy to describe, maintain and understand.
- Logical structure of database is sufficient, no need for knowledge of physical data layout.
- Independence of data, tables are not hard-linked to each other.
- Referential integrity (RI) notion. For better usage and implementation of business rules within data, constraints are enabled to ensure that one-to-many and many-to-many relationships within data are followed within data sets. RI constraints are primary keys (employee id number in the employee table), foreign keys (manager id number in the employee table), check constraints (salary can not exceed certain amount), unique keys (no duplicate id value can be entered for different records).



Employee Table

Employee_ID	First Name	Last Name	Title	Manager_ID
001	John	Smith	Consultant	143
002	Stephen	King	Writer	152
003	Bill	Gates	Programmer	143

Manager Table

Manager_ID	First Name	Last Name	Title
143	Mark	Samson	VP
152	James	Dean	Director

Figure 2.2 Relational Database Structure-Primary and Foreign Keys

RDBMS consists of data sets (tables) made of attributes (columns) with data represented by certain values. Each data set (table) has a number of records (rows) in it. Even though in the RDBMS literature [30, 62, 157] more preferred terms are table, column and row for the purpose of keeping consistent with previous research in the field of data mining and knowledge discovery, terminology used in this paper refers to tables as data sets, columns as attributes and rows as records (experiments).

Dr. Codd also introduced the concept of Structured Query Language (SQL). SQL language is a 4GL language constantly developed and reviewed by the ANSI organization.
Initially, SQL was supposed to perform operations of **SELECT**, **JOIN** and **PROJECT** on the data sets and structures. Simple code that explains all three, applied to the tables in Figure 2.2 could be

```
SELECT EMPLOYEE.First_Name, EMPLOYEE.Last_Name, MANAGER.Last_Name
(first row projects columns to choose)
FROM EMPLOYEE, MANAGER
WHERE EMPLOYEE.Manager_ID=MANAGER.Manager_ID
(joins two tables EMPLOYEE and MANAGER)
AND EMPLOYEE.Title='Writer';
(selects specific records)
```

Due to an increased need for more sophisticated computations in database languages such as PL/SQL [157] that have both 3GL and 4GL functionality were introduced in the late 80's and early 90's. Features that PL/SQL offers are

- Variables and types (predefined and user defined)
- Control structures such as IF-ELSE, WHILE, UNTIL statements and loops.
- Procedures and built in functions (complex mathematical formulas, character string handling and manipulation, etc.).
- Object types (latest generation of databases-object oriented databases).
- Error handling and tracking.

Why use SQL languages for datamining in databases? As explained above the SQL languages have the ability to go through the database structure and manipulate it easier than any other language would. It provides the developer of the code with the ability to create the code easily and to use without any extra interface necessary to interact with it. Inserts, updates, selects and deletes to the database are done with a simple one-line statement. When dealing with the databases this is the most important feature, the ability to manipulate and select the data for the purpose of research and the commercial use without any extra steps or code knowledge required. Also, SQL language is operating system independent, so it can be shared as a code across the platforms.
For compilers such as C, C++, Java (just to name few), to be able to query and retrieve the data from the database structures, they would have to have special interface with the database SQL like syntax to be able to achieve what SQL does naturally.
With the use of the database structure and SQL language, data sets can be manipulated and altered much easier than in any other way. Since data mining is performed on the data, and a database consists of data, logical choice was to try and explore the intended

database functionality and the languages such as SQL and PL/SQL for the purpose of research and data mining.

An early attempt to use SQL language and DBMS (Database Management System) was made by M. Kersten and M. Holsheimer [89] in 1994. The data set used had about 1,000 records and certain SQL queries were performed against it. Results and performance were better than before, but obviously it needed some tuning with respect to handling much larger data sets with more flexibility.

Today, with both better hardware and software, options of using databases for any data manipulation and knowledge extraction is very promising. For example, RDBMS has a feature of storing records (rows) in physical sequential order on the hard drive, hence reducing I/O during data retrieval and querying.

As mentioned in the overview of the PL/SQL paragraph, today object-oriented databases are being introduced. In the object-oriented databases, behavior of the data is being stored itself. For the purpose of a phone company, an object type named **profile** could be created that would consist of first name, phone number, last name. Instead of calling three different attributes from the database every time, the user can request the object called **profile** to retrieve the data. Even though this has advantages in the sense of data retrieval, code portability and maintenance, overhead is created on the storage itself. Data storage is decreasing in cost so the time is right for object oriented databases, which are nothing more than the relational database with extra code that eliminates steps performed during data manipulation, handling and retrieval (there is much more to the object-oriented databases though than the facts outlined above). Need for the new name and hype probably came from worried marketing people in major corporations, that if you do not show progress to the end user community you will be perceived as stagnant.

Data Mining and Relational Database Management Systems

The application of data mining using a Relational Database Management System (RDBMS) is useful because normally, the amount of data in the database is large and often not relevant to the analysis performed in the process of decision making. With this in mind, the most common type of database used for the purpose of data mining is a Data Warehouse.

In the process of maturing, datawarehousing community has defined many different terms and systems to better capture and explain the potential and use of data warehouses and related systems.

Decision Support Systems (DSS) and Expert Systems

Expert system could be defined as a system that makes the decision for the user, based on the fact and design given (it infers the logic based on facts and directions). For example, in the procurement department of the major corporation, a system can be used to analyze and determine usage of certain raw materials over some time period, time of year, holidays, etc. and based on that, to predict with high probability necessary reordering dynamics and quantities. Not that much input is required from the procurement manager once all of the data is entered into the system. Decision support systems (DSS) are used when dealing with questions and problems that might not be inferable by simply providing facts to the system and hoping for the legible and logical answer(s). In short, human interaction and a certain level of guidance are necessary for the analysis results to be interpreted and used.

For example, a classic DSS system is Marketing DSS (MDSS or MIS as per some other books and materials). In MDSS, the goal is normally to analyze the sales trends based on the region, time, sales person, client, product lines, etc. The system can go as far as providing spread sheets and pie charts, but the final decision is to be made by the manager who decides the changes and where they are to be made, based on the analyses performed. Obviously, human factor brings dose of subjective approach and experience for the final decision making steps.

What differentiates the DSS from an expert system is the following:

- A DSS is used mostly for the simulation of real world problems (new in themselves). Simulated scenarios can be created, some parameters can be changed to create different outcomes that are analyzed by the human later on.
- Human interaction is necessary
- Ad hoc queries are allowed. User can post more queries based on the answers already received or based on the direction they want to take with future analysis.
- Numerous answers can be produced as the result of analysis.
- External sources of data can be used.
- User's expert knowledge of the observed domain is necessary.

The DSS and datawarehouses used in the financial field, for example, to create flexible visual representation of a P&L (Profit and Loss) statement used by any company in a similar form. Based on the P&L report, managers can project the budget for the next year, compare trends, strengths and weaknesses and in general improve a company's performance.

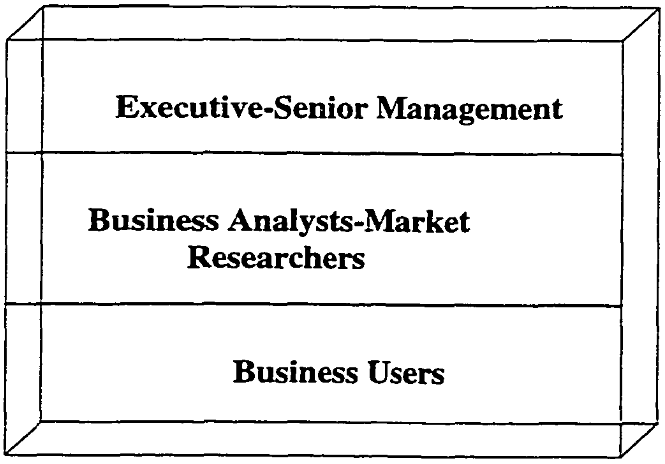

Executive-Senior Management
Business Analysts-Market Researchers
Business Users

Figure 2.2.a Audience and users of DSS financial/marketing system

User community of MDSS or any other type of DSS varies in their interests and knowledge levels, but they all have one common trait, they need to understand and manipulate the data. At the top of the user list are senior managers and a particular design of DSS that caters towards their needs is normally named EIS (Executive Information System) which is used for strategic planning decisions that effect the entire company. All of the major mechanisms in the world of datawarehousing are presented in the next figure.

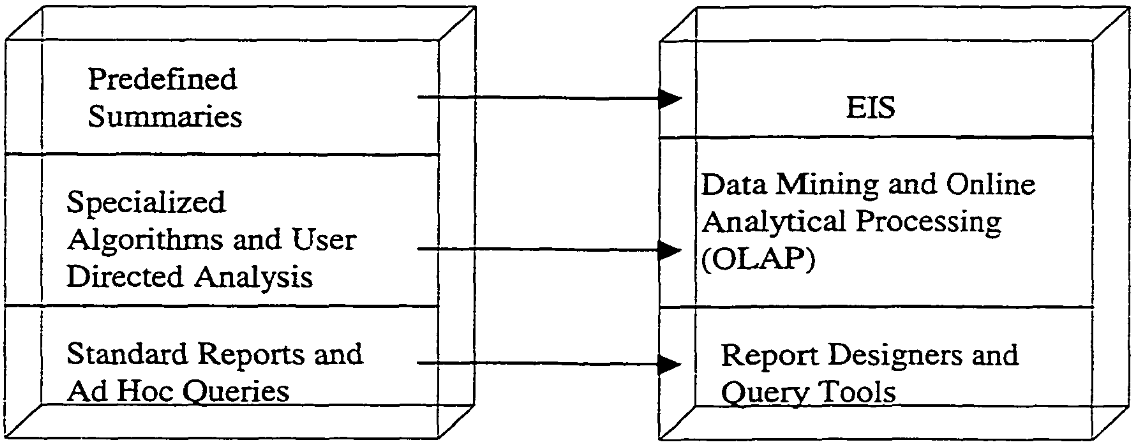
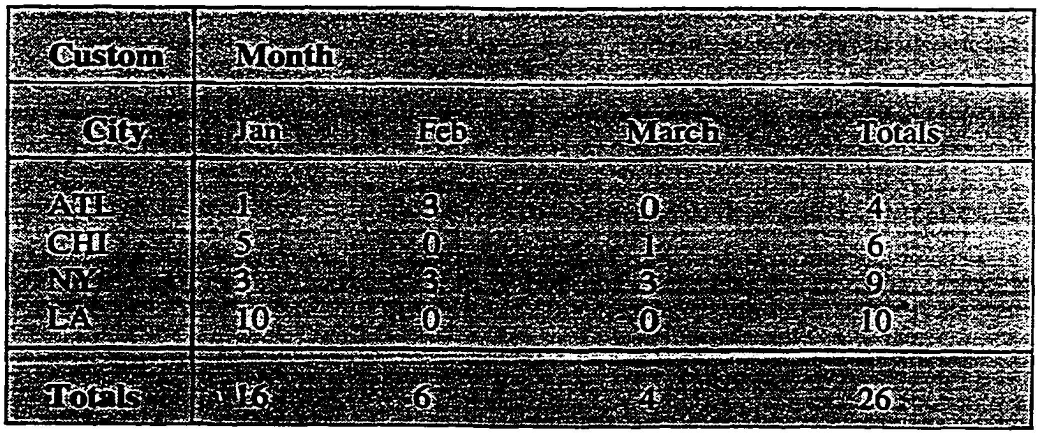

Predefined Summaries
Specialized Algorithms and User Directed Analysis
Standard Reports and Ad Hoc Queries
EIS
Data Mining and Online Analytical Processing (OLAP)
Report Designers and Query Tools

Figure 2.2.b Appropriate mechanisms of DSS financial/marketing system and existing tools (definitions)

As usual, terminology in the IT world is a challenge in itself and the reason to have the user community and delivery/tools defined is obvious. Online Analytical Processing (OLTP) is the ability and a front-end tool that enables data and business analysts to perform drill down capabilities throughout the data warehouse and present results in tabular and multidimensional format in almost real time fashion. Obviously, for an OLTP system to work correctly, the system that is serving the data warehouse has to have sufficient hardware resources. Dr. Codd also provided the definition of OLAP in 1993, where he defined a set of rules that should be followed for the system to be OLAP enabled.



Custom	Month			
City	Jan	Feb	March	Totals
ATL	1	3	0	4
CHI	5	0	1	6
NY	3	3	3	9
LA	10	0	0	10
Totals	16	6	4	26

Figure 2.2.c OLAP (front end tools) representation of desired data breakdown and information

A normal RDBMS system would have to perform multiple queries to achieve the same results as in Figure 2.2.c and probably extra front-end (report) programming based on the complexity of the output results. With OLAP tools, this extra work is not necessary.

One of the common misconceptions, 10 to 20 years ago, was that OLTP (Online Transaction Processing) systems could be used for efficient data analysis, report creation and DSS functions. Although in many cases, OLTP systems do contain the proper data necessary for all of the DSS analysis and expert systems. Due to the already explained structure of the OLTP system (which is highly normalized data without any redundancy) it is difficult to replicate and aggregate data necessary for all of the DSS tasks and to satisfy different user groups defined in the Figure 2.2.a.

Sometimes OLTP systems will have data removed after a certain period of time and certain age of the data (normally 1-2 years after the initial data transaction and storing). In the cases of financial reporting and analysis, budget and cost adjustments will not be in the OLTP database to begin with. In the case of a telecommunication company, the changing set of standards and the introduction of new equipment and its downtime analysis (known as the Service Level Agreement) will be in the almost impossible to understand form in the OLTP system. These are just a few examples of why OLTP is not suitable for serious data analysis, trend comparisons and decision-making tools.

A typical datawarehouse [2, 12, 17, 23, 30, 53, 62, 74, 81, 125, 164, 167] is built from many different data sources, normally sources with different formats and coming from various architectures; legacy systems, client-server environment, different OLTP databases and OS structures, flat files, etc. Some of the largest data warehouses are over 1 Terabyte of data and they contain records for the entire country's population. There are several well-established and common steps in building a data warehouse.

- Cost/benefit analysis (feasibility study). All of the tangible and intangible costs and benefits are quantified and compared. A sound business and technical analysis team normally performs the analysis since the building of a datawarehouse is not a cheap one. If return on investment (ROI) is as desired, the decision to proceed with the data warehouse is approved in this step.
- Systems analysis is the analysis of the data sources considered for the data warehouse, description of those sources and documentation, analysis on data extraction, transformation and loading from those sources. A commonly used term is ETT (Extraction, Transformation and Transportation). The systems analysis stage does not take much time due to the development of efficient CASE (Computer-Aided Systems Engineering) tools that can perform many of the outlined functions without too much user guidance. Also, due to the poor understanding of overall data warehouse complexity in its construction, managerial structure normally tends to enforce expedience in this step.
- Design of systems-physical implementation of the analysis and logical structure created and documented in the step above. Design of the datawarehouse, specification of software tools, processes and security considerations. Prototype is normally created at this step. Sound database knowledge is required for this step.

- Implementation step-ETT is being performed and datawarehouse populated with the necessary data. Performance of the datawarehouse is being tested and tuned to speed up the queries, data population and processing. Normally during this step, many discrepancies and errors from the previous steps are discovered and corrected, hence lengthening this step.
- Front-end integration and OLAP design. Although not a direct step in the building of the datawarehouse (which is considered to be a back end structure), will go together with any datawarehouse project, since companies normally want to see the front-end, multi dimensional representation (as per Figure 2.2.c).
- Maintenance step-ongoing process since datawarehouse creation is an iterative process that never ends. Steps such as data administration, loading, change analysis, new business rule introduction and requirements belong to this step, which is almost cyclical in its nature.

For the purpose of pure data mining only back-end steps of the datawarehouse build (excluding OLAP step) are of consideration.

The user community in the following fashion utilizes this created datawarehouse:

- Sophisticated and basic data analysis and aggregation of relevant data attributes.
- Simulation and forecasting
- Comparison of trends over time and interest/habit areas, etc.
- Data mining research

In the Data Warehouse, data is static and historical [156], also highly denormalized (all of the necessary data mostly can be found at one place-data set), which is the best layout for any data mining tool. Research can be performed and assumptions tested without fear of encountering high noise in data, change of the data sets used, etc. There are numerous data mining techniques that can be employed [18, 32, 36, 49, 50, 58, 61, 69, 76, 79, 80, 86, 90, 99, 112, 115, 116, 117, 120, 134, 142, 143, 144, 145, 149, 155, 168, 172, 173, 176], e.g., induction of rules, neural networks, clustering, etc. They all have a different conceptual approaches to the idea of data mining and some of them can not be adopted at their present stage in the databases (neural networks are known for not being able to handle databases with a large number of experiments).

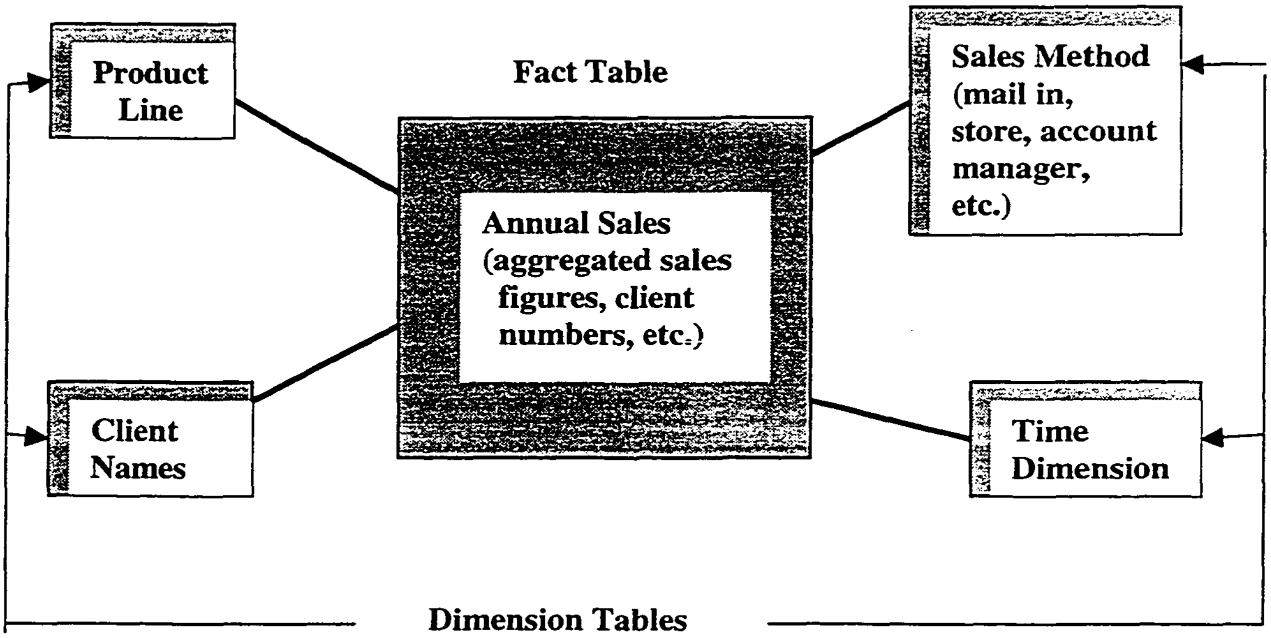
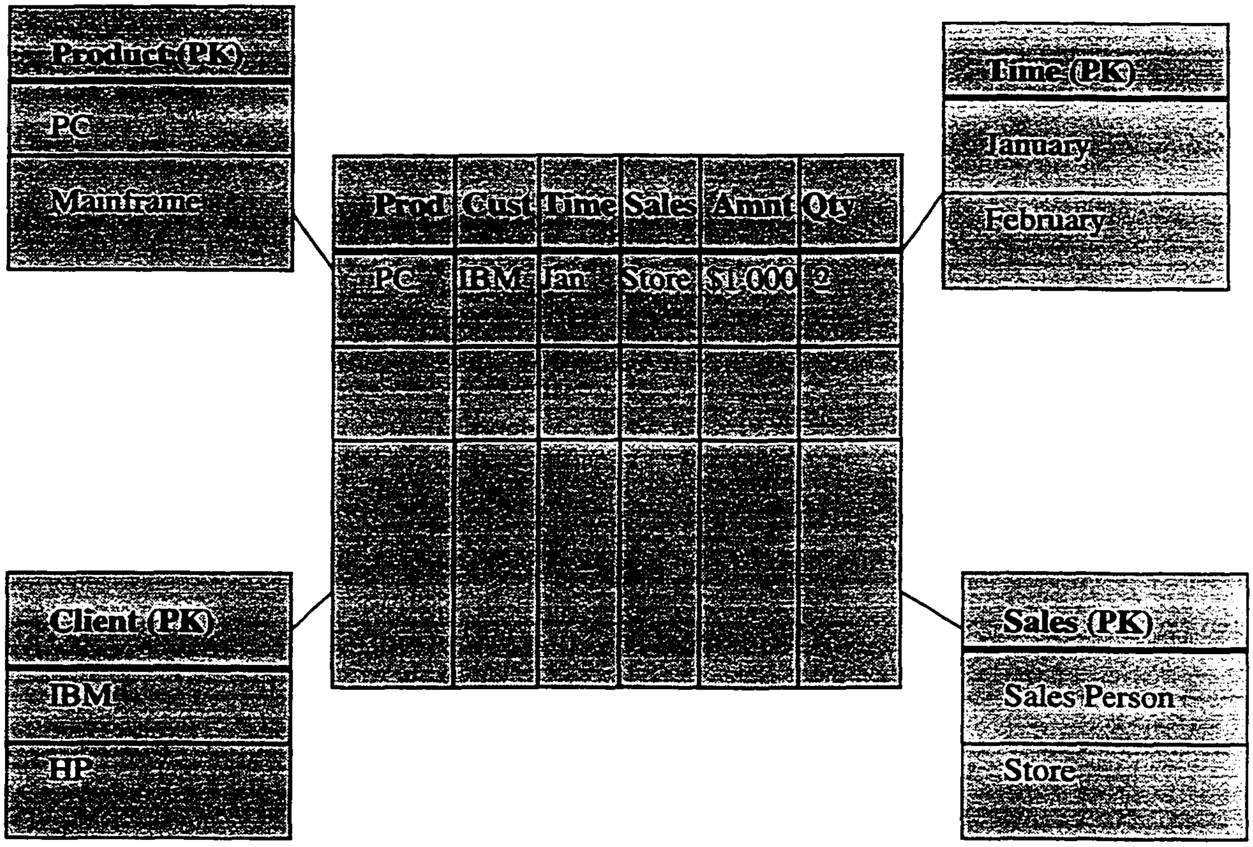

Product Line
Fact Table
Sales Method (mail in, store, account manager, etc.)
Annual Sales (aggregated sales figures, client numbers, etc.)
Client Names
Time Dimension
Dimension Tables

Figure 2.2.d Common data warehouse structure-STAR schema

In Figure 2.2.d, STAR structure (the most common one) of the data warehouse is presented. It comprises of five tables, the central fact table (highly denormalized) with columns (attributes) in it that are different aggregated values and measures relevant to the overall data warehouse and the data being observed. Some of the fact tables are more denormalized than others, hence lowering the number of necessary joins with the dimension tables. As mentioned, the fact table is joined with the single level dimension tables, necessary for the queries of various types. If information about sales figures for the month of January in the category of Personal Computers, sold in stores by IBM Corporation, the STAR query would be executed that would join four dimension tables and the relevant numbers from the fact table. The STAR query mechanism is a special database mechanism designed to speed up queries done against the datawarehouse. The fact table has all of the values from each and every dimension table and their unique combination in the relation with measures and numbers for the sales figures provides users with all of the necessary data (Figure 2.2.e).



Prod	Cust	Time	Sales	Amnt	Qty
PC	IBM	Jan	Store	$1,000	2

Figure 2.2.e Data structure within data warehouse (STAR schema)

It is obvious why the data warehouse structure is very suitable for any kind of data mining algorithms. The fact table in itself will have necessary information for most of the data mining activities. Normalized structures, such as OLTP databases have data records split to the stage where each piece of information is held in one place only.

"Normalizing data is a formalized procedure for eliminating redundancy from data by the progressive use of *non-loss decomposition*, which involves splitting records without losing information. In reducing the data model to a state where each bit of information is only held in one place, the update process is simpler and more efficient, eliminating data inconsistency "-Oracle Corporation

The next figure presents the OLTP database model, and obviates why it would be difficult to perform successful data mining on such a structure.

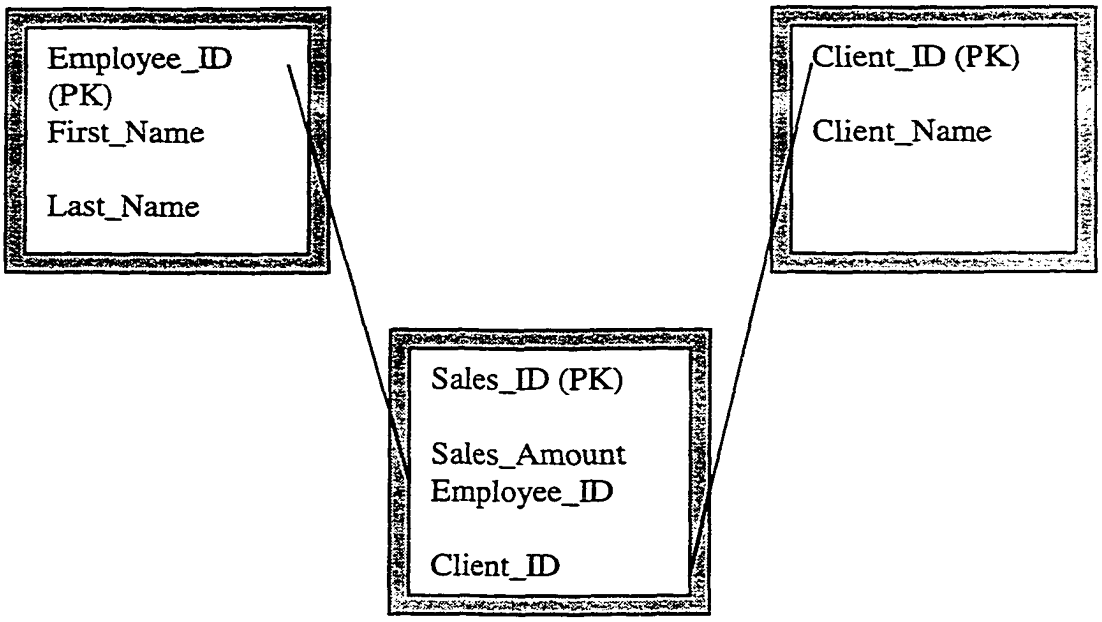


Figure 2.2.f Information records split among different tables in the OLTP database

For the data mining algorithm to perform analysis on sales data with this structure, numerous joins would have to be performed on the tables in the figure above to properly create input values and groupings of records.

As seen from this paragraph, data warehouses are very useful for data mining and data analysis functions due to their structure, data "readiness" and historical amount of it.
Some of the problems with datawarehouses are the amount of data that can make the maintenance very challenging and also updates from the various data sources with different business rules. To overcome this, IT professionals have resorted to creating data warehouses smaller in size that are spawned as children of the major datawarehouse(s). They are normally departmental datawarehouses and the same in builds as the major datawarehouses (that contain entire corporate data-financial, operational, HR, etc.) but only containing data relevant to the particular department or the research group (marketing data, cardiology data, etc.). These small datawarehouses are normally called Data Marts and since prefabricated for particular research, even easier to use and manage in the process of any data analysis, DSS or data mining.

2.3 Decision Trees

Since decision trees are chosen as the most appropriate data mining technique for databases and this particular research, we explain in more detail about the design and construction of the decision tree in the following text. Decision trees are used to classify instances by sorting through them from the top to some terminal node (one of the ending

nodes for that tree branch). In the literature, a terminal node is also referred to as a leaf node stating that there is no branching beyond that point.
The basic algorithm of a decision tree and building procedures for a decision tree are the focus of many research articles in this field, especially work done by Quinlan [126, 127, 128, 129, 130, 131, 132, 133] and the developments done by other researchers [22, 41, 42, 43, 106, 107, 108, 114, 158, 159, 160, 166]. Past work dealt with mostly two-class decision trees and the number of records of the datasets used was relatively small. Most of the original research and algorithms on decision trees are developed by Quinlan [126, 128, 133].

The goal of the basic algorithm of a decision tree is to create a tree-like structure with nodes based on values of attributes involved at that node's building.

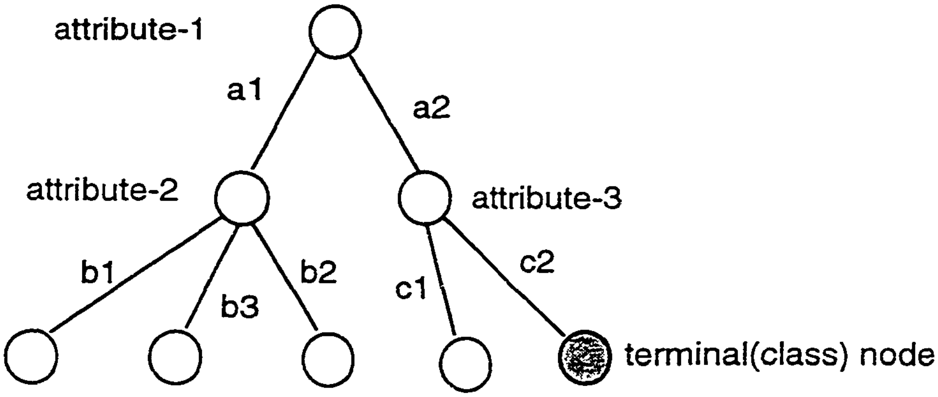


Figure 2.3 Basic decision tree structure

Being very concise, we can enlist a number of essential features of a decision tree:

- Trees are constructed through a top-down design (tree is grown top-down)
- Entropy based selection criterion is the commonly used design criterion (performance index)
- The selection of the attribute tested at each node of the tree; this means that a single attribute is used at each node.

To make the presentation of the generic decision tree complete, let us define all necessary notions. Furthermore, we denote the number of records by "N", we assume the number of attributes is "n", and the number of classes is "c".

Data set (table)-A finite collection of attributes (columns) composed of experiments (records or rows).
Attribute (column)- a column with finite set of values.
Terminal Node (leaf)- Node created due to the homogenous (only one) class left in the node or due to the fact that all of the attributes in the data set have been used so far.

Node-intermediate node used to carry on further computations in constructing the decision tree.
Target Class (class)- Identifier class(s) chosen for the decision tree.
Entropy-Measure that characterizes purity of the data set or its subsets with respect to target classes.
Gain- A measure of attribute's ability to classify the examples (records) of the data set according to their targeted class.
SplitInformation- Measure that characterizes purity of the data set with respect to observed attribute's values.
Gain Ratio- Measure of the information gain over the SplitInformation.

A brief description of the decision tree

In the first step we calculate overall entropy of the dataset with respect to the classes.

$$E(classes)=-\Sigma(n_i/N)\log_2(n_i/N) \qquad (2.2)$$

where n_i is the number of patterns (data points) belonging to the specific class i, i=1,2,...c and N is the number of records in the overall data set.

Next, gain for each attribute is calculated with respect to the dataset's entropy using the following formula.

$$Gain(attribute)=E(classes)-E(attribute) \qquad (2.2.a)$$

The entropy of the attribute, E(attribute), is computed as follows

$$E(attribute)= -\Sigma k_j/N(\Sigma(n_i/k_j)\log_2(n_i/k_j)) \qquad (2.2.b)$$

where k_j is the number of subclasses j, j=1,2...s, n_i is the number of patterns (data points) belonging to the specific subclass i, i=1,2,...c and N is the number of records in the overall data set.

Example: Observed attribute is education level. Case data has 14 values in it, with 6 people of high education, 5 people of medium education and 3 people of low education. Class identifier is gender male or female. 8 cases are males and 6 are females, hence we have two identifying classes in the case data set. 4 high-educated people are males, 2 medium educated people are males and 2 low educated people are males. Based on this, entropy of the attribute and entropy of the data set can be calculated.

$$\begin{aligned}E(educated)=&6/14(-4/6\log_2(4/6)-2/6\log_2(2/6))\\&+5/14(-2/5\log_2(2/5)-3/5\log_2(3/5))\\&+3/14(-2/3\log_2(2/3)-1/3\log_2(1/3))\\&=0.985\end{aligned}$$

$$E(classes)=-8/14\log_2(8/14)-6/14\log_2(6/14)=0.984$$

Gain(educated)=0. 985-0. 984=0.001
Based on the above information gain for the attribute, the education level is 0.04812703 for the observed case data set.

The example above serves the purpose of better understanding the general decision tree build based on the algorithm and equations presented above. Logarithmic entropy functional is used to achieve this in the example above.

The attribute with the highest gain is positioned at the top of the decision tree and repeated until the entire tree is realized either by means of achieving terminal nodes with records belonging to only one class or by achieving terminal nodes by using all of the attributes in the data set. Table 2.3 summarizes the overall algorithm of the decision tree.

It is worth stressing that any decision tree produces classification boundaries produced by any decision tree and are of rectangular shape; a simple illustration arising in the two dimensional case is included in Figure 2.4

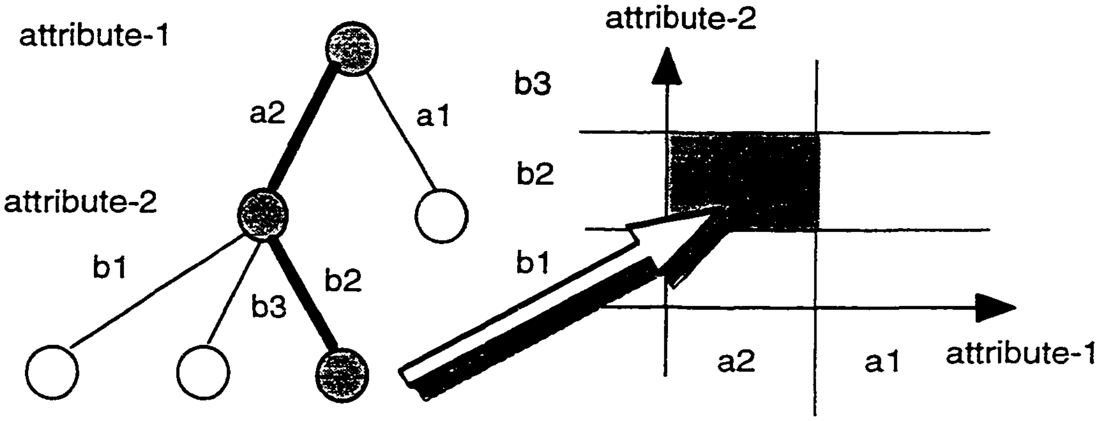


Figure 2.4 A part of a decision tree and the resulting decision boundary

The decision tree method relies on the use of attributes with finite values. Continuous attributes require discretization. The simplest method (yet quite often oversimplified method) concerns uniform discretization. Some other methods use sophisticated algorithms [44, 89, 126, 133, 144].

- Create a starting node for the tree using entropy (eq.2.2).
- If all records are in one class, then create a terminal node with label homogenous and a class identifier.
- If there are no more attributes to use in the decision tree construction, then create a terminal node with label heterogeneous and all the class identifiers involved and the percentage of their involvement.
- Otherwise, choose the attribute with the highest information gain (eq.2.2.a) and create a node (intermediate) based on that attribute.
- The attribute with the highest gain is a root of the new sub-tree.
- From the node (root of the sub-tree), create a branch for every value that an attribute can assume.
- Go back to step one and re-examine homogeneity of the nodes and a number of attributes left that can be used.
- End the process when all the attributes are exhausted or all the terminal nodes are fully homogenized whichever comes first.

Table 2.3 Summary of the generic decision tree algorithm (ID3)

Once the decision tree is constructed using a training data set (that is a data set used to train-create the tree), a testing data set is used to estimate accuracy of the tree, by measuring how well such a decision tree represents the data in the testing data set. A rate of classification is a measure of the decision tree's accuracy,. defined in the form

$$\text{accuracy} = \frac{\text{number of correctly classified patterns}}{\text{total nunber of patterns}}$$

Performance of Decision Trees

As already explained in the section above, the accuracy of both training and testing data sets determines the classification rates of a decision tree. Other method worth of noting when deciding on performance of decision trees, would be the complexity of rules generated (complexity of decision tree itself-its size). Sensitivity of decision trees is also a useful measure in some cases, where the sensitivity is represented as the ratio of correctly classified test cases for a particular class over the overall number of test cases belonging to the observed class. The tree of lower complexity with the high classification rate is the desired goal in the general data mining sense. Sometimes, obtaining rules with more than just a simple few attribute relationships involved in the rule generation is also a desired goal, since it captures the multi-linked relationship among different attributes in the data set not easily visible. An approach that uses premises of complexity (level depth of decision tree and generated rules) and relevance and weighing of attributes in the data sets is explained in the next chapter in section 3.2.2 and supported by the experimental findings and follow-up analyses in Chapter V. One of the factors that should be considered as a performance enhancement with decision trees is the interpretability of results obtained from decision trees.

In the table below, performance comparisons and results as provided by L. C. Briand, K. El Emam, D. Surmann, I. Wieczorek, K. D. Maxwell [24] are given for several different data mining algorithms. The criteria used to assess different algorithms in the referenced article are the MMRE (mean magnitude of relative error), MdMRE (median MRE) and PRED (prediction at given level l, PRED (l)=k/N, where k is the number of observations where MRE is less than or equal to l). Methods assessed in this study are Ordinary Least Squares Regression, Stepwise Analysis of Variance (ANOVA), CART (classification and regression trees), combination of CART with Regression Analysis and a couple of Analogy-Based Estimations.

It is worthy of explaining extra details about the CART algorithm and software since it uses the decision tree approach, the approach that was one of the focal points in the research presented in this thesis. The CART method uses a decision tree to display how the data is going to be classified or predicted. CART automatically searches for important relationships within data in the data set and uncovers certain hidden patterns. The authors of the original CART are Breiman, Friedman, Olshen and Stone, all of them well renowned in the field of machine learning for their pioneering efforts and work.

The data set used for this case study comparison was the database named the Experience Database. The database has data on 206 different software projects (banking, wholesale, insurance, etc.) obtained in Finland and it is considered to be of significant size as per authors.

	stepwise Regr.	stepwise ANOVA	CART	CART+ Regr.	Ana-1s	Ana. -2s	CART+ Ana.-1s	CART+ Ana-2s
MMRE	0.53	0.567	0.52	0.52	1.16	1.34	1.25	1.39
MdMRE	0.44	0.446	0.39	0.42	0.61	0.57	0.57	0.602
Pred(.25)	31%	34%	34%	34%	24%	17%	24%	16%

Table 2.4 Average results over all hold-out [*] samples using the entire database

[*] Hold-out samples are the test data samples as defined by the authors of the article.

As seen from the above table and from the conclusion section of the article above [24], the CART method performed the best in the sense of general classification (accuracy) and prediction. Also the CART model, which uses decision tree techniques, is less complex and easier to interpret and understand than more complex methods compared in this article. This study and the other studies coupled with the notion of large databases and data systems emerging every day, qualify the decision tree techniques and approach as a perfect candidate and a tool of choice for KDD.

Pruning Decision Trees

Another measure of the decision tree observed is the size of the tree. The simplest possible tree with the highest accuracy is the optimal goal to achieve. The need for simplicity has introduced many algorithms that deal with various methods of pruning decision trees [22, 107, 133]. The general idea of this method is to achieve a smaller size of decision tree without incurring a high classification error as a consequence of having these trees compacted. Quinlan, in his book "C4.5: Programs for Machine Learning" [133] poses the question "When to simplify?" (decision tree) as an introduction into explaining the pruning of decision trees. Per Quinlan, there are two ways to simplify the decision tree, either not to divide a set of training cases any further in the particular sub-branches, or to remove some of the already created sub-branches and replace them with a terminal node(s) (leafs). The first approach is normally called prepruning, and the advantage of this method is that computational time is lowered when assembling the tree. The attempt here is to assess the partitioned node and by using some of the mathematical comparison methods, such as information gain, error rate, etc. decide whether to stop computations and partitioning on that particular branch. If some preset threshold is not satisfied while comparing the particular branch, further build is stopped and the entire branch replaced with the most appropriate terminal node (leaf). Research concern here is if the threshold is set too high, termination of structure partitioning will occur prematurely, hence not capturing the proper tree structure built from the training set. If the threshold is set too low, simplification of the tree structure will be insignificant since discrimination of less relevant partitions will be bypassed. The researcher should perform enough trials and demonstrate a magnitude of research sensibility before deciding that the threshold set for the training and partitioning of the decision tree is acceptable. Due to the lowered computational time, this method is well suited for large databases (datasets) with limited number of attributes but a significant number of experiments (records) in them. In such dataset nodes that are not homogenous (all values in the terminal node belonging to the same class) are common occurrence and the overfitting of the tree seldom can happen. On the other hand, many experiment values and an almost permutation level of different outcomes in some of the datasets can increase computational and structure time in geometrical progression order without increasing the accuracy or the overall usefulness of the built tree structure.

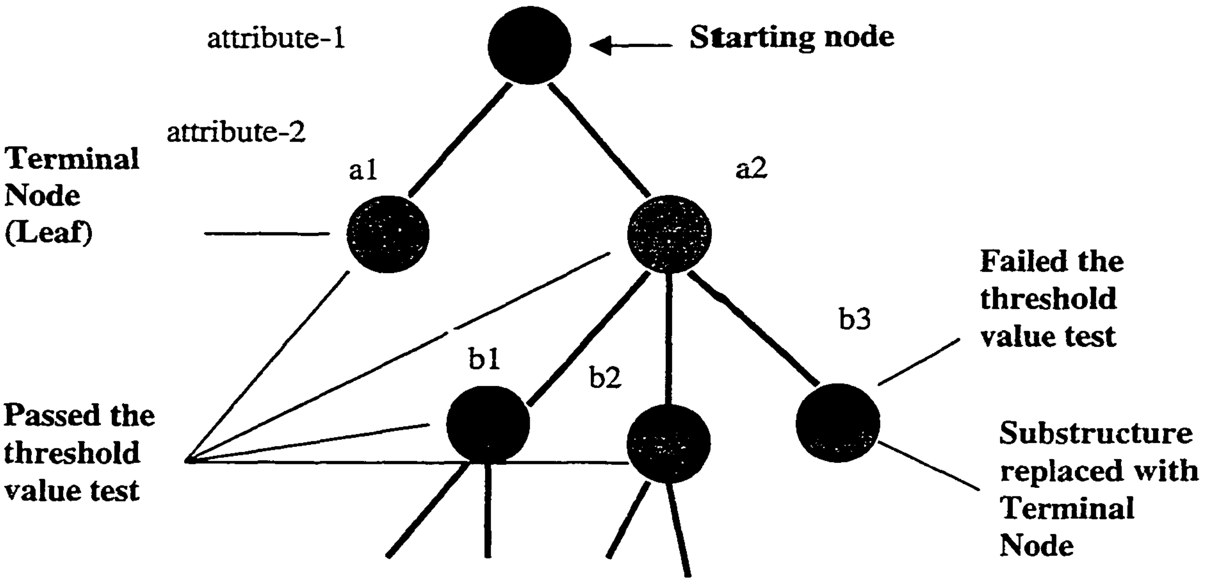

attribute-1
Starting node
attribute-2
Terminal Node (Leaf)
a1
a2
b3
Failed the threshold value test
b1
b2
Passed the threshold value test
Substructure replaced with Terminal Node

Figure 2.5 Stopping (Prepruning) method for decision trees

The second pruning method is performed after the entire decision tree structure is built. Mingers [107] discusses five different methods of pruning and their empirical comparisons.

- Error-Complexity Pruning-Breiman [22] developed a two-step method, where numerous decision trees are created by pruning the whole structured tree by different amounts (eliminating different sub-branches and levels of tree structure), and then choosing the one with the lowest classification error generated by the independent data set. For the training data, pruning of the tree structure will increase the error, hence determining by the error magnitude how important the particular branch (pruned) was to the overall decision tree structure.
- Critical Value Pruning-based on the strength of a particular node observed. Strength is determined for the attribute at the node, as how well the chosen attribute partitions data between data set's classes at that node. If strength is greater than the critical value (threshold) chosen, branching can continue at that node, otherwise it is pruned (unless the node further down the same sub-branch equals or surpasses the critical value).
- Minimum-Error Pruning-as the name states has the objective of reaching the minimum classification error after pruning the tree structure.
- Reduced-Error Pruning-Quinlan [128], introduced this method, similar to the other error pruning methods. The approach is to create a set of pruned trees by using the test data itself. Test data is run through the tree created by the training data and pruning of the sub-branches is performed after that. Removal of the particular sub-branch and replacement with the terminal node with respect to the misclassification of the test data is observed. The node at the particular sub-branch where classification error gain (expressed as the difference between sub-branch classification error rate

and observed node-potential candidate for the terminal node) is the lowest should be promoted to the terminal node and the sub tree removed from future calculations and tree structure itself.

- Pessimistic Error Pruning-also introduced by Quinlan [126] based on the performance of a pruning of the tree without using a separate testing data set. Often, by using separate test data, the misclassification rate of the tree can be high, hence preventing any significant pruning of the training decision tree. Quinlan introduced a statistical (mathematical) approach that prunes the tree without using a separate test dataset. Mingers [107] did point out that the method is crude and justification of it is somewhat doubtful.

Many of the methods outlined above are used today in many forms for the research being performed. The scope and the magnitude of the data sets has changed, so did computational efficiency of the systems, but the approach used in most of the methods still has much validity and sound foundation in it.

In summary, three questions should be always answered when pruning the decision tree,

1. When to stop pruning? Pruning should be stopped when researcher feels that the tree is getting oversimplified or the error rate too high with any further pruning undertaken.
2. How does pruning affect the performance of decision trees? Pruned trees will take less computational time to build and to test. Training set classification error rate (set used to build the decision tree) will increase, and the test set can have a higher or lower classification error rate based on the pruning method used and the nature of the data set itself.
3. What is considered a good performance for the pruning of decision trees? The researcher or user of a particular algorithm would determine good performance as the outcome of decision tree pruning. Knowledge of data sets, rules obtained from decision trees and acceptable error rates are all subjective and particular to the research driven.

Missing Data and Decision Trees

Handling missing data and noise is also addressed in different research works [126, 128, 133, 144]. The most important idea that has come out of these articles, is the estimated percentage of the values that can be replaced. What this means is that no more than a certain percentage of the original data set values should be replaced at any time. If the percent of data replaced is too high, a misleading structure of the decision tree can be created, which in return might jeopardize the validity of any follow-up data mining and data analysis activities.

The reason that research in the past, explained in this section, was performed mostly on two-dimensional class decision trees, was the inability of tools present at that time to perform in the way that would encourage researchers to think further about extra enhancements.

2.4 Literature Overview

Although in previous sections of the first two chapters, many references were already quoted and addressed, for the purpose of better understanding of the literature reviewed and assurance that work done in this research is original, state of the art and logical, general literature overview is provided in this section.

First, the overview of already addressed references is given with the emphasis on the most important ones for the research performed.

In general, work done by Quinlan [126, 127, 128, 129, 130, 131, 132, 133] has bettered the entire field of machine learning with the focus on decision trees and rule induction over the last many years. Quinlan sums up his research in [133] explaining the current trends at the time of publishing and some of the corrections that related to his previous articles. ID3 algorithm, generic decision tree manipulation, implementation and analysis were provided in his works that inspired more sophisticated and advanced research in the general space of decision trees. Quinlan also provided for some useful ideas and techniques in the space of the pruning of decision trees, which was addressed later in the work performed by Mingers [106, 107].

Some of the alternate methods to decision trees and the work provided by authors such as Bibby et al. [1979] on multivariant analysis, Bennett et al. [1999] on nearest neighborhood, Bezdek [18] on pattern recognition, Davis [49] and DeJong [50] on genetic algorithms just to name a few well articulated research articles on techniques used in KDD and data mining.

What should be also addressed here in the general overview, is work done on different clustering and data discretization techniques, such as works by Pedrycz-one of the foremost experts in the field of fuzzy computing [115, 116, 117, 118] in the space of fuzzy clustering and fuzzy computing, and some earlier works in the same space by Zadeh [172].

In the space of KDD and machine learning works by Fayyad et al. [66, 67, 68, 69] should be mentioned since he provided for many useful definitions and insights within the KDD process and overall opinion forming of the research community. Brachman and Anand [21] provided a very interesting and informative article with much common sense and a guiding direction in the space of KDD, in 1996, discussing the process of KDD and segregating it further for better understanding.

In the space of database and datawarehouse related works, authors such as Burleson [30] and Urman [157] explore and familiarize the reader with the concepts of datawarehouses, high performance databases and database languages. Burleson provides for research based insights with the historical overview of datawarehouses and their evolution within both business and scientific fields.

There are other notables within the space of KDD and decision trees, such as work done by Utgoff [158, 159] on incremental induction of decision trees and work done by Cios [39] just to name few. Cios et al. [41, 42, 43] presents the research which combined the advantages of decision trees and the AQ algorithm that learns from multiple covers.

Since most of the articles addressed thus fur in this section were already referenced in the previous sections, the rest of the literature overview section will present the research done recently in the space of KDD, data mining, decision trees and rule creation that was not addressed up to this point.

Adomavicius and Tuzhilin [1] worked on the problem of obtaining certain user profiles from such transactional tools and systems such as web interaction and differentiation between good and bad rules that can be uncovered about the user by the use of data mining tools. Their application was tested against real marketing data, providing the user profile creation from demographic and transactional data. Results of their research and prototype were DBMS storable, which does provide for extra functionality. The requirement for this method is that a human expert has to be used to the properly guide the process.

Aggarwal, Gates and Yu [3] discuss the usage of supervised learning in order to create sets of categories for the classification of documents. They later prove that supervised clustering can perform as well as manual categorization but at a lower cost. Also, a new method is attempted which better differentiates among closely related clusters.

Aggarwal, Wolf, Wu and Yu [4] present new approach for rating-based collaborative filtering. This approach uses a technique based on the concept of predictability. It requires a modest learning curve to produce results, can find atypical links among the groups and uses the hierarchical classification within the rating process.

Ankerst, Elsen, Ester and Kriegel [7] discuss the visual approach to the design of decision trees. Direction taken is that the interaction of the expert is required to build this tree in an interactive and easy to understand fashion. Authors claim that their method would produce results in a similar range to the results that could be obtained from more powerful algorithms. The benefit is that the user is involved with every step of the creation, certain manipulation can be achieved after the tree is designed. This method is proposed for numeric attribute data sets mostly.

Aumann and Lindell [8] try to further the exploration on association rules extracted from databases that contain the attributes within them with quantitative values. This article has theoretical connotation to it, in the field where the theory of problem resolution is already fairly advanced.

Ayan, Tansel and Arkun [9] explore the possibility of real-time maintenance of association rules and dynamic pruning in large sets of data. This research has a very interesting idea of trying to by-pass the need to re-run the mining algorithm and leverage the association rules already discovered in the old database structure before new data and transactions were updated against, in the database. Some of the assumptions made within this research are very original in its nature and within large database systems.

Briand, El Emam, Surmann, Wieczorek and Maxwell [24] provide for very comprehensive and well thought comparison amongst several data mining algorithms used for the software cost estimation modeling techniques. Comparisons among techniques such as regression, analogy, classification and regression trees and their mutual combinations with respect to classification rates are given in this work.

Barbara and Wu [13] try to address the issue of exploratory data analysis in multidimensional databases and cubes.. The methodology developed handles the cells with anomalous data and with no data within, but the method and algorithm implemented is I/O limited with respect to large multidimensional tables.

Bay and Pazzani [14] present the tree like method that deals with contract-sets and groups that represent the classes in the data mining method. Their focus is in the field of exploratory data analysis on the contrasting groups and classes. Also they address the issue of lowering the computational complexity by introducing the pruning routines that prune the subsets if the deviation of contract between two groups is below minimum and no useful and surprising data is to be uncovered from the subset.

Bayardo and Agrawal [15] propose a new algorithm for mining rules from the data sets. In their method, the idea of returning more rules than other existing algorithms would provide more insights in the data itself and that it would also enhance the user interaction. The research is strong theoretically with many assumptions and explanations given to support the reasoning of the authors, which will evolve in something more pragmatic and heuristic over the course of research. Some preliminary experimenting is already done as stated and supported by the article.

Bennett, Fayyad and Geiger [16] try to leverage database features such as indexing to better store and compute on highly dimensional data using the nearest neighborhood method and similarity queries with the improved indexing usage and structure. The authors have a very fresh idea, but based on the tools and technology used the results can be hampering to the overall direction.

Bonchi, Giannotti, Mainetto and Pedreschi [20] try to leverage the use of decision trees and the classification method to better the abilities of logic driven database languages to query and mine the data for the purpose of uncovering credit fraud cases within the data. An interesting fact is that the authors have used C5.0 tool by Quinlan to design their decision trees and further their research.

Brijs, Swinnen, Vanhoof and Wets [25] provide research in the direction of marketing usage of association rules. The case study performed deals with the most frequent items purchased and the store display based on this. The data used for the research is real life data obtained from the transactional in-store sales registration system.

Brin, Rastogi and Shim [26] focus on obtaining the optimal gain of association rules to uncover seasonal and local patterns within data. They also introduce the bucketing data clustering approach with coalescing of continual variables at the input.

Buntine, Fischer and Pressburger [29] articulate the nicely conceptualized idea of enabling the user to select which data mining algorithm to use based on the input provided and direction suggested through the pseudo code generated. The concept from this research could definitely benefit researchers by providing them with the valuable guidance prior to the certain type of data mining undertaken. One of the problems as seen by the authors is the lack of capability and focus on large data sets, which is one of the very important factors within the KDD process.

Cheng, Fu and Zhang [35] experiment on entropy based clustering for numerical data trying to provide the research community with the new approach to cluster the numerical data, which can be challenging.

Cortes and Pregibon [46] suggest that certain steps of the KDD process before and after the data mining method is chosen, cause higher costs both in time and resources than the data mining itself. They suggest the approach to create the platform for KDD deployment and more automation within starting and ending steps within the KDD process.

Davies and Moore [48] use in their research, Bayesian networks to achieve lossless dataset compression on the large data sets.

Domingos [54] tries to introduce the new idea of a cost-minimizing classifier and with the set of trials to contrast it to the C4.5 cost-blind classifier. Although the idea is worthy of further exploring, present findings still show the need for more progress and experimental trials before conclusive evidence can be fully supported.

Dong and Li [55] introduce the notion of "Emerging Patterns" (EPs) and provide the algorithms to uncover those within certain database structures. EPs can be used to build very determining classifiers. There are still a number of limitations to their approach as admitted but the correct direction and further implementation would provide research with new and interesting data with numerous applications.

Dorre, Gerstl and Seiffert [56] provide research in the direction of data mining and clustering within text data and information outside of database or flat file structures.

DuMouchel, Volinsky, Johnson, Cortes and Pregibon [57] focus on minimizing the size of an observed data set by a method named squashing. Squashing consists of three steps, grouping, momentizing and generating. It is suggested as an alternative to random sampling in the space of statistical analysis, large data set exploration (OLAP) and

machine learning.
Fan, Stolfo and Zhang [64] discuss the usage of the AdaBoost tool to generate classifiers over large data sets and try to prove that this tool performs as well as the others and even better for certain cases. This research is very pragmatic and application oriented as seen from above.

Fawcett and Provost [65] introduce the idea of activity monitoring. This is monitoring and searching for interesting events that require certain actions as a response to positive findings. The applications of this method can be already seen in fraud detection, general population behavior monitoring, etc.

Gaffney and Smyth [70] investigate the problem of clustering by using the maximum likelihood principle with unsupervised learning. Suggested application of developed method is in the space of computational biology, video, meteorology, etc.

Ganti, Gehrke and Ramakrishnan [71] introduce the new clustering algorithm named CACTUS and discuss its usage and advantages.

Guralnik and Srivastava [75] discuss the use of statistical databases and event detection from time series data by using sensor readouts. The linear regression approach is suggested in this research and the first set of experimental results that provide some interesting data in the field of raw sensor data analysis is obtained.

Huang and Yu [82] focus their research on time series analysis and better interpretation and query ability of interpreted and re-structured time series data.

Kelly, Hand and Adams [88] explain in their work dynamical updating of classification rules and their initiative for doing research in this space. The linear regression model is used to obtain experimental results and findings.

Larsen and Aone [91] provide yet another useful method of clustering with logically implemented refinements to the general clustering approach.

Lee, Stolfo and Mok [92] deal with data on network intrusion detection that is time dependent, raw and of different aggregation levels. For this reason they devised the entire KDD process around it with all of the necessary steps involved such as data selection, preprocessing, etc.

Lesh, Zaki and Ogihara [93] use sequence-mining algorithms as a preprocessing method for preparation prior to mining with another mining algorithm such as Naïve Bayes.

Liu, Hsu and Ma [96] address the issue of rare occurrence within the data set and rule as the outcome. A solution that does not exclude rare items and rules is given in this research, which could be beneficial to a time dependant data set of larger magnitude. The approach with decision trees that lets the user to choose multiple levels of minimum

support and rule acceptance is implemented and presented.
Liu, Hsu and Ma [97] explore the pruning and better summarization of the discovered associations and rules.

Louie and Kraay [98] provide the tool and method for better data visualization, pattern recognition and data mining with the use of the ORIGAMI tool.

Mani, Drew, Betz and Datta [100] provide business-oriented study in the space of lifetime value and tenure prediction with the use of neural networks. The techniques explored revolve around better utilization and value added solutions in the space of CRM (customer relationship management).

Mannila, Pavlov and Smyth [101] try to provide for a new method that would go beyond classic rule generation algorithms with the use of the cross-entropy function. As the outcome, authors hope to achieve better prediction with local patterns obtained.

Megalooikonomou, Davatzikos and Herskovits [102] developed the application and method that deals with biomedical imaging of the brain. An elaborate database system with enhanced SQL web-enabled capabilities is incorporated with the use of multiple mining algorithms to provide for better results. Multivariate analysis, regression, etc. are some of the methods used in this research.

Meretakis and Wüthrich [103] address the issues of Bayesian learning, association mining, and classification in their research on long itemsets.

Nag, Deshpande and DeWitt [110] explore in their research more real time application of data mining within the DSS environment. They propose the use of the cache mechanism that would expedite the result retrieval and generation of the data mining algorithm of association rules. Although this is a very useful idea, it has been explored within the business community on many different occasions through the custom coding and implementation. What may be the best value out of this research is to maybe propose to standardize the efforts and address the issue of processing time and cache storage within OLAP and DSS environments and especially association rule mining.

Oates [113] addresses the problem of clustering in the space of speech recognition and identification of distinctive sequences from multivariate and time series data.

Piatetsky-Shapiro and Masand [121] provided the thorough study on marketing and list campaign and cost/benefit analysis necessary within this space. They also address the general terminology of the target marketing and user analysis. Also important noting is that the authors have used neural nets, decision trees (CART and C5.0) and Bayesian methods for this research with roughly the same results.

Potts [123] is explaining the method of using neural networks in a better fashion for predictive data mining with better interpretability and scrutability of results by the use of

generalized additive neural networks.
Provost, Jensen and Oates [124] provide a potentially very useful theoretical approach to progressive sampling for use with the induction of rules. Experimental results support the general direction of this research with regression analysis and C4.5 decision tree algorithms used as testing methods.

Sahar [139] provides research in the space of interestingness of obtained rules with decision trees. The approach taken is to eliminate rules that are not interesting, leaving the end user with only interesting rules.

Shanmugasundaram, Fayyad and Bradley [141] try to enhance the OLAP design of multidimensional cubes that are normally fixed in the business user community and limited to roll-ups and drill-downs. Authors propose a method that would reduce the size of pre-computed cubes, would not require fixed dimensions for cubes and would provide for different and flexible aggregation functions available without additional storage. A potential problem with this method is that the user community can not risk to approximate DSS pertinent data with any algorithm. Authors introduce the term of approximate query answering which by itself might create some acceptance problems of the outlined method.

Shewhart and Wasson [148] present in their work the approach that deal with data mining on text data and time-series analysis.

Tung, Lu, Han and Feng [154] introduce in their work the idea of inter-transactional association rule mining. This would entail that certain rules would be uncovered over multiple transactions. The idea was addressed before by the business community and partially resolved with the means of different tools in the way of multidimensional cubes and databases, but a proper scholastic approach and research methodology is definitely welcome for this attempt.

Wang, Wang, Lin, Shasha, Shapiro and Zhang [162] present five distance mapping algorithms and their use in data clustering applications with experimental results and trials. This attempt might have its usefulness in large databases in the preprocessing and targeting steps of the KDD process.

Wijsen, Ng and Calders [165] presents work in the area of OLAP and mining roll-up. This method is proposed to uncover functional dependencies and concept hierarchies amongst data.

Based on the above literature review and current research findings and trials in those articles in the area of KDD and data mining, the original ideas and novel approach taken in this research which was already addressed in Chapter I and will be addressed in greater detail in following chapters, are definitely state of the art and a new approach taken to the data mining, databases and decision trees.

None of the research articles discussed in this and previous sections addresses the issues of different entropy functions, alternate branching, attribute relevance, node manipulation, merging of different and heterogeneous databases, decision tree like approach to missing value replacement which preserves the authentic structure of known values and data within data set, just to name several of the ideas that were successfully implemented and tested. Nor do they address the extended and logical use of database server engines for the purpose of better tree manipulation in the way presented in this research. This and controlled and extensive material research over the course of this research work, assures the originality and usefulness of this work.

Most of the ideas in this research will definitely find its space in some of the future research performed by the others and may be coupled with the research ideas presented in the research articles reviewed above in this section in necessary detail.

At present, ideas and algorithms developed in this research are unique and useful, leading in the direction both logical and research driven without duplication of its main points, findings and design success in other works.

2.5 Conclusions

In this chapter theoretical and historical fundamentals of work relevant to this research were presented, supported by the necessary tables, figures, references and examples. The definitions and terminology of the database world and knowledge discovery, decision trees and data mining were explained and presented with the necessary level of details. Summarizing, decision trees are driven by the key agenda of Machine Learning [33, 104, 108, 122, 126, 128, 133] meaning that one attempts to develop trees of high accuracy. When studied in the setting of data mining, decision trees need to incorporate several objectives and requirements essential to DM, such as a high interpretability of the trees. The design of the trees in the presence of large databases has to be revisited as well.

CHAPTER III

Research Objectives

In this chapter, the main objectives and the ideas of the research are introduced and discussed. Subsequently, a number of algorithms and a thorough discussion about them are given with the use of many descriptive examples.

The general objective of this research was the design and implementation of algorithms that use decision trees in the database environment for the purpose of data mining and data set manipulation in a new and research approved way.

The following objectives are outlined as the most dominant and important for the research and work accomplished in the thesis (although there are other ones of smaller magnitude that were also an important part of this research in a supporting form).

- To develop algorithms and a new approach for the data mining technique with the use of decision trees and a data set building and manipulation as a result of data mining.
- To analyze new methods of attribute discretization and analyze their impact on the overall performance of the developed decision tree (in terms of its accuracy as well as generalization abilities).
- To analyze decision trees from the standpoint of their computational efficiency.
- To analyze various generalization of the generic type of decision trees built.
- To implement the developed algorithms and perform the initial testing.
- To analyze obtained results and to achieve the necessary volume of experiments for the successful completion of research.
- To tune the developed application and modify the algorithms until a working version of software is designed with the extra features and functionality that can be easily used and altered by the end user.

3.1 Novel approach to decision trees, data mining and databases

As explained in earlier chapters, limitations on decision trees prevented their widespread usage in the industry. Although there are many companies now that try to implement decision trees as a successful tool of choice for data mining purposes, most of the implementations are on a generic decision tree algorithm without any reusable capabilities. Results are also presented in those tools by the means of variance and deviation, but how is the user supposed to utilize the results like that? A major demand in the industry is on the knowledge extraction in large data sets.

Problems that arise within it are databases with different structures from one another. Another issue is the need for more than two deterministic classes in the process of the decision-making method while using decision trees.

Maybe different data sets of different nature and origin might require different mathematical functional used to measure their purity (certainty), which will capture the internal structure of the data set better than the other would.

In an attempt to address all of these issues, the following ideas are developed, analyzed and implemented

- ***Multiclass entropy*** for decision trees.
- ***Normalization and unification*** of data sets with different attributes using decision trees.
- Investigation of ***different forms of entropy functional.***
- Handling of ***missing attributes*** using database features and tools.
- New domain knowledge methodology for data sets, both ***partially guided*** and ***fully guided relevance*** of attributes.
- Pruning of decision trees with the use of ***data set size criterion*** and ***one layer method.***
- ***Multi attribute class identifier*** based on all possible combinations of selected attributes.
- The ***alternate branching*** of decision trees, in the case of nodes where the information gain between two or more competing attributes have the same information gain.
- ***Likelihood of terminal nodes*** and manipulation of decision trees using database capabilities and tools.

3.2.1. Interpreting Decision Trees in Multi Class Environments

In this section we will address both multiclass and multi database issues. In the practical world they are often encountered together, so their implementation has to be considered in unison.

The reason for the enhancements presented in this section is not only of scientific nature. These augmentations exhibit a number of practical implications as well. Examples of multiclass ideas, for example, are three-dimensional models as navigational charts, atomic and molecular structures, diagnosis done on patients that might have classes as “sick”, “healthy”, “more diagnosis needed”, etc.

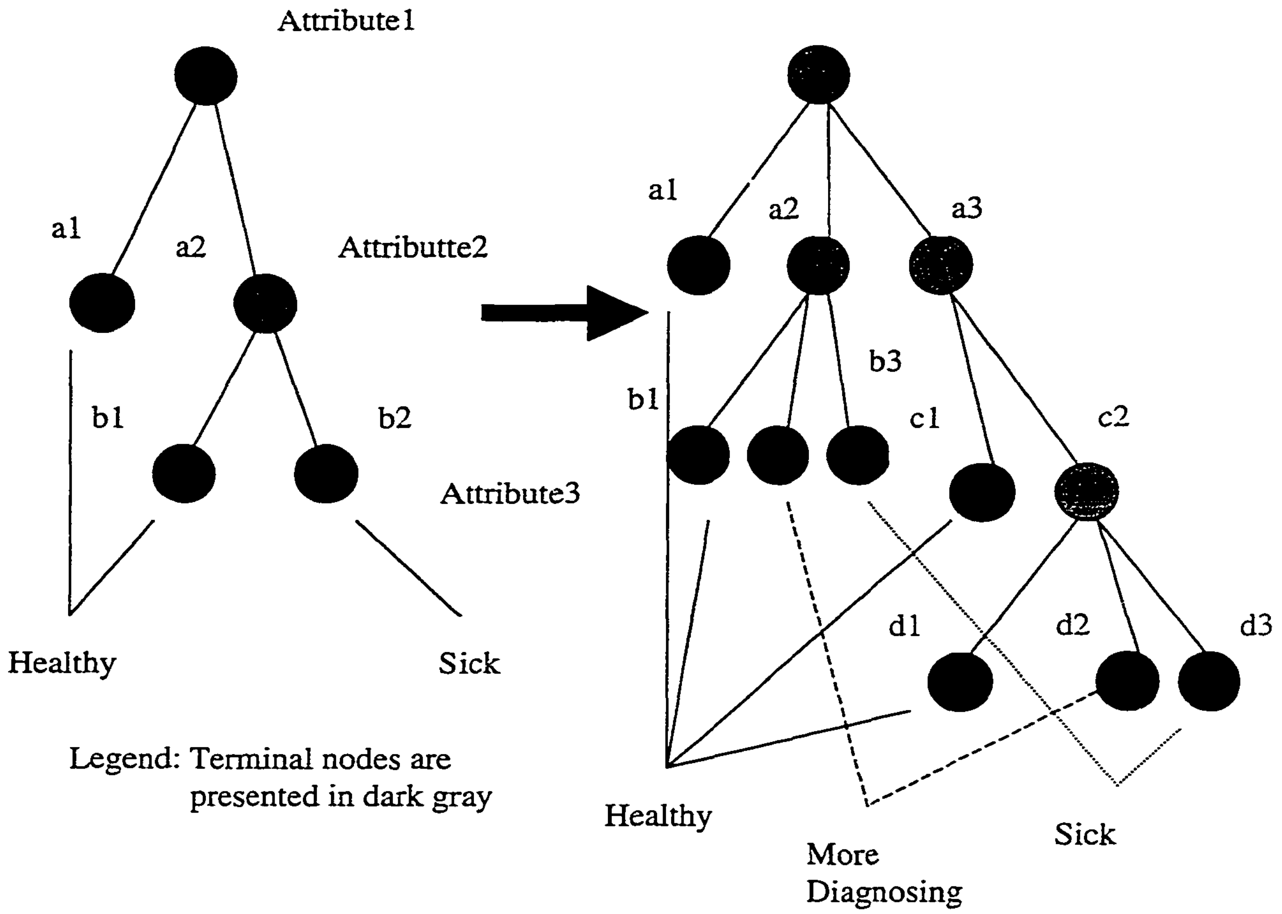


Figure 3.1 Two-class versus multiclass (three-class) decision tree-heart diagnosis data set

Obviously from Figure 3.1 the complexity of the structure built is much higher and if the data set were of non-significant size, terminal nodes for multi-class would hold a minimal number of values in them, due to the data sparsity. Real life databases and datasets are much more complex in their structure than to be presented and classified by two classes of distinction, and the size of the datasets is normally large enough for building the decision trees on them with many different classifiers (multi-classes). At this day and age, computational power of data mining systems is so high that even very complex structures and decision trees can be successfully implemented and tested.

When observing the example of the heart disease dataset and a figure above, a question that often comes to mind is if the classes of healthy and sick are sufficient to determine the accuracy of the entire data set. Normally, this is not the case. The patient can be diagnosed as healthy, sick, needs more testing, early signs, etc. This research addresses the issue using functionality of the database structure and the SQL language. The idea is to introduce an extra class of unidentified and to see how well this enhancement will perform on real hospital data.

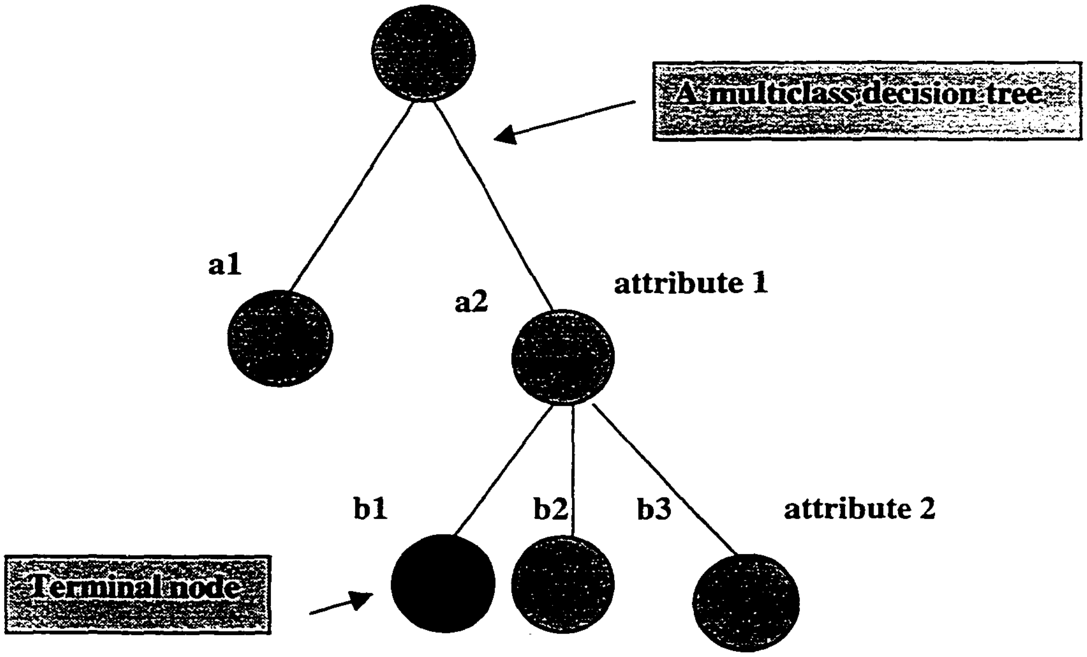


Figure 3.2 Terminal node for the multi class data set (database)

In Figure 3.2 a general approach to a terminal node in the multi class environment is observed for the decision tree that was built using three classes +, - and *. If the predominant class at that node was + then the normal confidence level of the rule at that node would be determined by taking the number of positive cases over the total number of cases as explained above. Using database tools and PL/SQL by simply adding a few lines of code in the modularly structured software package, this problem is eliminated.

This method of multiclass selection provides extremely sophisticated rule generation and data set manipulation abilities.

3.2.2 Indirect Decision Tree Approach for Multi Database Structures

As addressed in the introduction section of this chapter, normalization of data sets (databases) with different attributes and different general uniformity can be achieved with the use of decision trees, thus eliminating obstacles from institution to institution while dealing with large databases of different values and the same attempted functionality. In the hospital world, for example, there are different proprietary databases, each with a certain number of same or similar attributes used in them and many attributes are used only on that specific database. The reason for this could be historic in the sense that what is deemed to be important, as an indication of ones health might have been considered irrelevant in the past. Another reason is that every institution has different approaches in diagnosing and assessing their patients. This in return will create problems for large health organizations and researchers to distinguish among all variations from data set to

The solution approach in this thesis to achieve that, is the use of an indirect combination of decision trees. The method consists of a translation of each of the obtained decision trees for different data sets into an equivalent set of classification rules, ordering them and putting them together. Each terminal node of a decision tree is marked by the number of patterns belonging to multiple classes. The ratio

$$\lambda = n_i/\Sigma n_i \qquad (3.1)$$

where n_i is the number of patterns belonging to the specific class "i", can be regarded as a general confidence level of the associated rule originating at the initial node of the tree and leading to the already mentioned terminal node with confidence factor. Then, all the trees are indirectly combined by forming a single collection of rules. The choice of the most significant rules that will become a constituent part of the final tree, if one is attainable, is to be determined by setting a certain "threshold of acceptance" for each of the rules. Namely, if

$$\lambda \geq \lambda_{min} \qquad (3.2)$$

where λ_{min} is a threshold and λ is a confidence level of the considered rule.

The problem that can occur is that if the threshold is set too high or too low, it eliminates some rules necessary in capturing the overall data set or can increase complexity of the data set beyond optimum, as a result of it. One of the ways to eliminate this problem is to introduce ***attribute relevance***. Attribute relevance can be best described as the importance of the attribute in the data set. If the attribute has a higher influence on the outcome of the class than some other attribute, then its relevance is higher. The values of relevance can be user assigned prior to any computational step or can be computed after being partially guided by the user. The attribute relevance factor is only used during the rule generation stage for the purpose of weighting the rule. During the creation of the decision tree, this factor is not being used.

In the following example, the method of calculating attribute relevance is explained using an illustrative data set of a medical nature.

Example 3.1

If the data set deals with heart disease and it has several attribute values such as age, chest pain type, resting blood pressure, heart rate, etc., it is obvious to an expert that the chest pain type attribute and the age attribute are more detrimental in class discriminating. If the end user is a physician, one would assign relevance values to each of these attributes. These relevance values would then be multiplied with the value of λ calculated in eq.3.2.

When deciding which relevance to use, the relevance of the last attribute can be used as a multiplier, the relevance of the first one or the normalized sum of the relevance of the attributes involved in the building of that terminal node. What this means is that based on the choice of the relevance used as a multiplier, there will be three different values for the weighted rule. In the case observed in the next chapter the normalized sum of all the relevance of attributes involved in the building of the terminal node was used.

The attribute relevance can be estimated using two different approaches.

The first type of relevance is ***user assigned relevance***. Values would be subjectively setup as the input value (such as age_relevance=0.7, chestpain_relevance=0.65, etc.). This relevance does not require any extra calculations, just an expert understanding in the field that the dataset encompasses on behalf of the user. This way, relevance can be assigned properly.

The second type of relevance is ***user partially guided relevance***.
This relevance is calculated after the initial guidance by the professional user.
Example: age - The older you are, the higher your chances are for heart condition to develop (non-hereditary one). The assumption made is that the higher (older) age groups are prevalently sick in the database.

Based on this assumption, analysis is performed on the data set with respect to two age groups of older patients. If there are 20 patients all together from these two groups and 17 of them are diagnosed as sick, then **Expected_sick_size** can be introduced, as 17 out of 20 patients are sick in these 2 age groups. The same method is performed for healthy patients. Lower (younger) ages are healthier by definition and common sense. If there are 20 patients in two of the younger age groups and 16 of them are diagnosed as healthy, then **Expected_healthy_size** can be introduced as 16 out of 20 patients that fit that class. If there is a third class (unidentified), one would get maybe 8 out of 10 patients needing more diagnosing and this class is introduced as **Expected_unidentified_size.**
By summing all of the expected classes calculated and dividing the sum by the number of classes involved:

(expected_sick_size+expected_healthy_size + …expected_n_size)/number_of_classes

the resulting relevance obtained is ***user partially guided relevance***. This is mostly for a professional user to guide it in the right direction.

Since databases used in the real working environment deal with real-life cases there should be partial guidance in this process because attributes do hold unique importance levels relative to one another. At the terminal node, confidence level λ is multiplied by this relevance and results higher or equal to the value of threshold λ_{min} are considered. The qualifying threshold has to be setup to lower the value for both cases of relevance because values of relevance are always less or equal to 1.

After all is done, various data sets with the different attribute structure will be interpreted through the means of the decision trees. Decision trees are then translated into rules using

threshold values and the relevance in the attempt of creating one uniform data set from data sets tested. Each rule should be weighted to include several factors.
The formula used to create the weighted rule can be of the following form:

$$WR=(n_i/N)\cdot(1/(1+LD))\cdot(1+DSS)\cdot(1+NA)\cdot REL \tag{3.3}$$

where n_i is a number of the cases belonging to class i that was chosen as the class of the terminal node and N is the total number of cases in all classes at that node.
Each of the WR values obtained is then divided by the largest value of WR to keep the resulting weighted rules between 0 and 1.

Different cases are obtained after test data is run through the decision tree created with the use of the training data set.

Let us explain the meaning of the variables stated in eq. 3.3:

- LD is the layer depth divided by 10 (because other factors are decimal numbers usually less then 1) where the terminal node is existent. The division by 10 is assumed in the subjective fashion, meaning weighted maximum layer depth could be used as a common denominator for example.
- DSS is the factor based on the size of the initial data set-number of rows in the table observed. Values of DSS are also chosen subjectively and based on the particular data set(s) observed in the research. A simple idea is to avoid oversimplification or over-classification of final structures.
- NA is the factor based on the number of attributes in the data set. Also chosen subjectively to best capture a data set's characteristics in a non-biased way.
- REL is a relevance of the attribute. Two already discussed forms of relevance can be used in the calculations ***user assigned relevance*** and ***user partially guided relevance***.

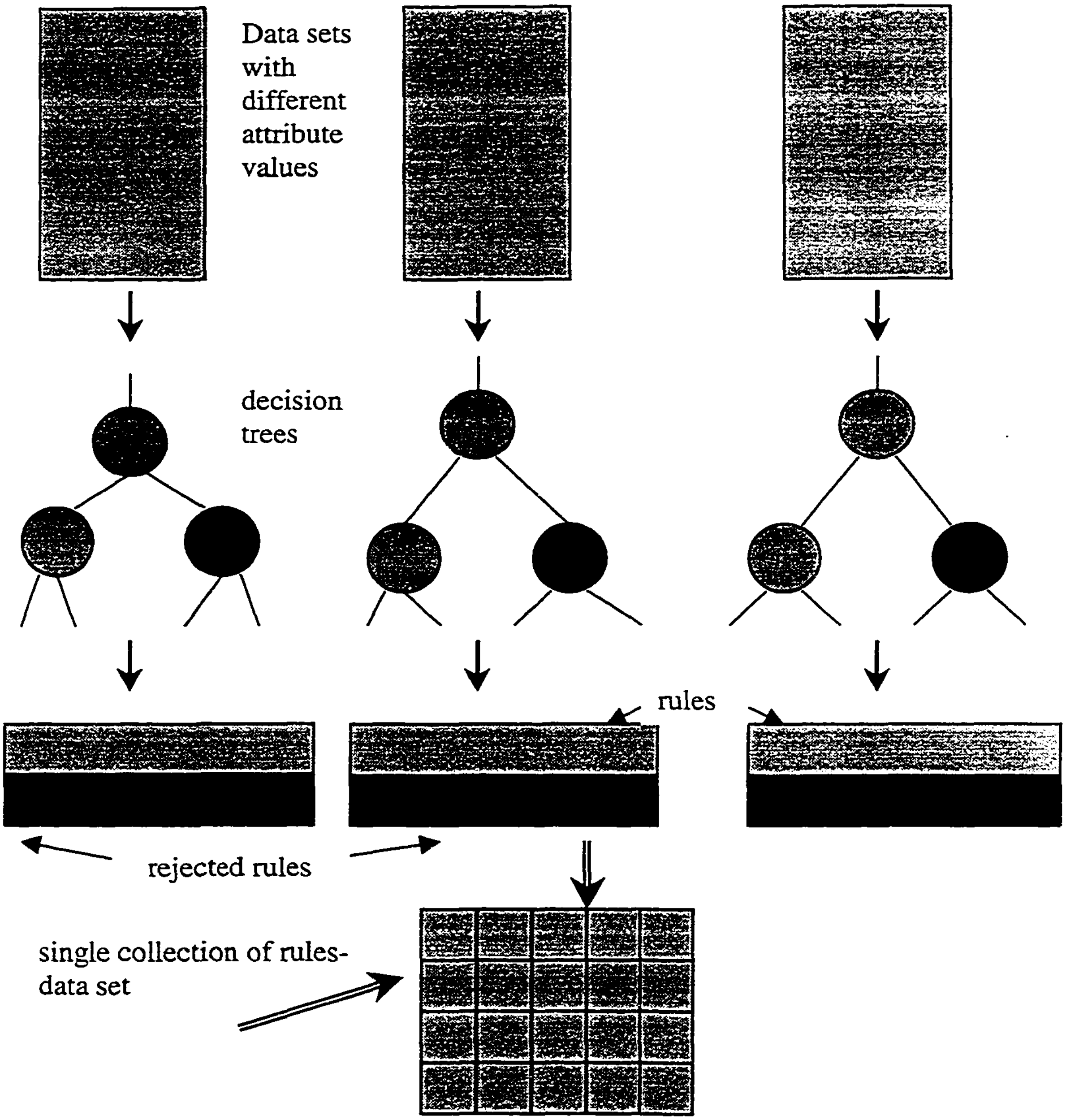

Data sets with different attribute values
decision trees
rules
rejected rules
single collection of rules-data set

Figure 3.3 Design of the single data set from multiple data sets using decision trees

Example 3.2

Consider a data set with 15,000 records and 24 attributes:

This means that NA=0.5 and DSS=0.2 (the way these values are chosen will be explained in Chapter IV. For now it is important to note that they are user assigned and that they can be changed based on the results obtained to better match the particular cases);
Observed terminal node is at layer depth 5, hence LD=5/10. After the training set is ran through, n_i is 700 and N is 1000; Rule itself involves five attributes named A, B, G, D

and E. After one or the other relevance approach is used, the total relevance is expressed as
$REL=(rel(A)+rel(B)+rel(G)+rel(D)+rel(E))/5$. For example, we can assume that the relevance calculated is $REL=0.65$;
Weighted rule value would be $WR=0.7\cdot(1/1.5)\cdot 1.2\cdot 1.5\cdot 0.65=0.546$.

Now, the value of WR is compared to the preset value-threshold and if greater or equal rule is then selected as a building rule for the final data set and attributes in it are qualifying for the final data structure. (Figure 3.1)

As can be seen from the above example and Figure 3.1, each data set will produce a certain number of selected weighted rules.

The question that is posed after that, is which rules to choose and which attributes are parts of it. The first step is to qualify only the selected rules that are made of attributes existent in each of the data sets being used. For example, if there is a rule in one of the data sets that is made of three attributes A, B and D, which are present in the other databases too, that rule has a good potential to be chosen above the threshold value for WR. The attributes from that rule are going to become attributes in the final data set in that case. The result of this selection will be a data set with a reduced number of attributes, but with a high relevance and "strong" data in it. If one is to address this from the point of view of retaining attributes that are not present in other data sets, then a missing value routine would be used as presented later in this chapter. The dilemma here is that by forcing a large number of missing values to be replaced with the estimated values from other data sets, the final data set would lose its authenticity (the number of cases assumed and artificially generated would be too high). This would be considered unacceptable to any corporation using this software.

The final questions that the end user might ask is: "Why not simply take only attributes from each data set that exists in all data sets observed and create a final data set?"

The answer is simple: many data sets are poorly designed carrying the overhead of unnecessary attributes and data in them. The goal of this approach is to data mine proper attributes from all the data sets and to create a final data set that will have only relevant data and the attributes in it that also existed in each of the observed initial data sets. That is why ***relevance*** of the attributes and all the other factors were introduced above.

3.3 Use of Different Entropy Functional

For different data sets it is interesting to see if a different form of entropy functional would create a different decision tree and hence alter any further work based on that choice. For data sets with uniform values, easy discretization, repetitious values, one can try a few different functional to see what kind of results and differences that would produce.

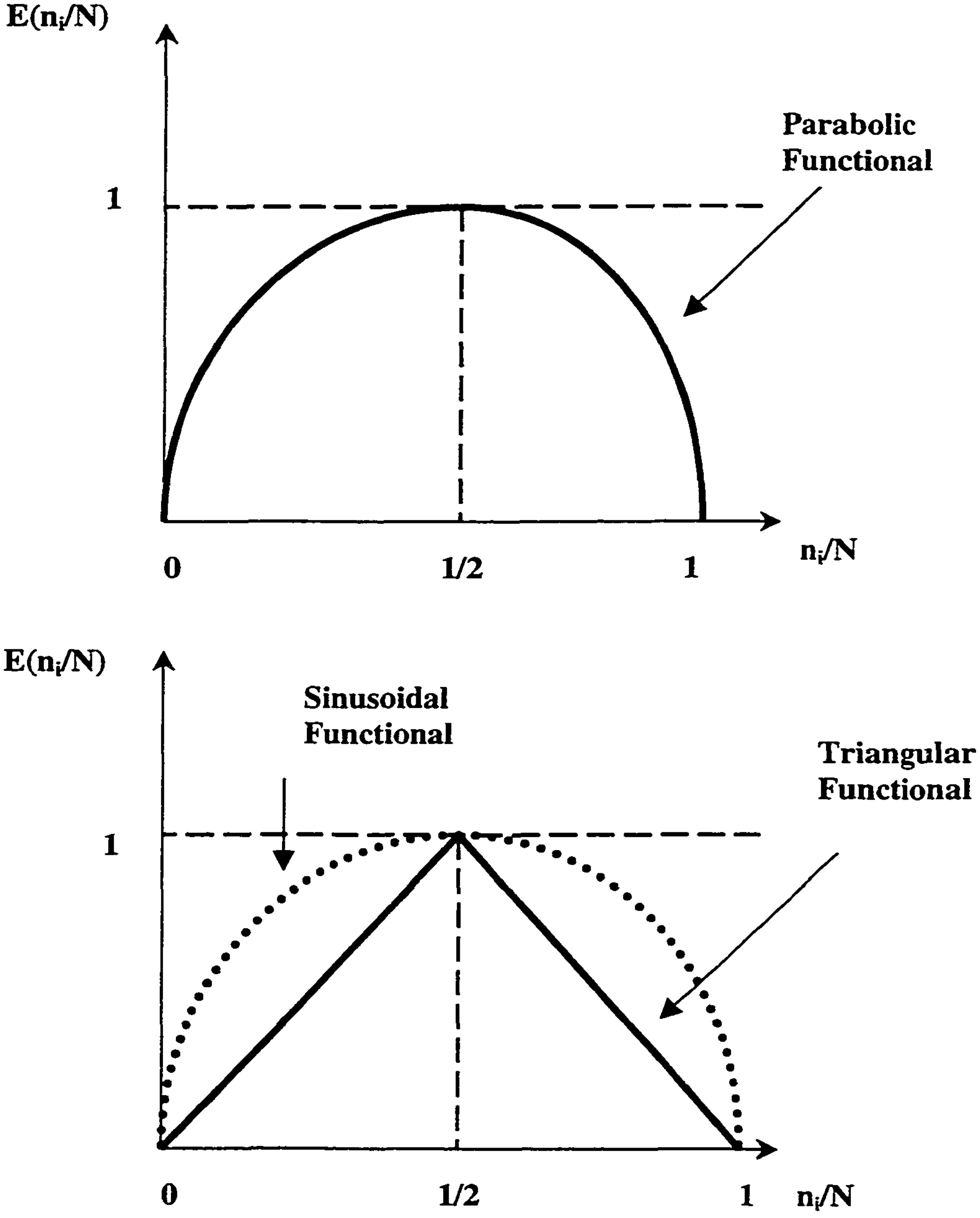


Figure 3.4 Different Forms of Entropy Functional: Parabolic, Triangular and Sinusoidal

Overall, the generalized version of entropy is viewed as a functional "h" from [0,1] to [0,1] such that

(i) h is continuous

(ii) h is monotonically increasing over [0, ½], monotonically decreasing over [½, 1]

(iii) h satisfies boundary conditions h(0)=h(1)=0, h(½)=1

Several examples of the entropy functional are commonly encountered, namely

Logarithmic entropy (Shannon entropy)

$$H(u) = -u \log_2 u - (1-u) \log_2 (1-u)$$

Quadratic

$$H(u)=4u(1-u)$$

Piecewise linear

$$H(u) = \begin{cases} 2u \text{ if } u \in [0,1/2] \\ 2(1-u) \text{ if } u \in [1/2,1] \end{cases}$$

In this thesis three new forms of entropy functional are designed and researched, triangular, parabolic and sinusoidal. The formula for triangular entropy functional is given as

$E(\text{classes})=\sum 2\cdot n_i/N\cdot x \quad (x = n_i/N \text{ if } n_i/N\le 1/2; \text{ and } x =1-n_i/N \text{ if } n_i/N>1/2)$ (3.4.a)

The formula for the parabolic entropy functional is given as

$E(\text{classes})=\sum 4\cdot n_i/N\cdot(x)^2 \ (x = n_i/N \text{ if } n_i/N\le 1/2; \text{ and } x = 1-n_i/N \text{ if } n_i/N>1/2)$ (3.4.b)

and the formula for the sinusoidal entropy functional is given as

$E(\text{classes})=\sum 1/2*\sin(n_i/N*\pi)$ (3.4.c)

Example 3.3

In this example, a case of two data sets with different gains, class identifiers and attribute relevance is used. It is found that for the first data set the decision tree built by using a ***triangular functional*** provided the best fit (testing portion of data set was classified with the lowest classification error rate). For the second data set a ***parabolic functional*** from the Figure 3.4 proved to be the best fit. Also, if the research or commercial objective was to merge these two hypothetical data sets into one by using the methodology outlined and explained in section 3.2.2, the data set that can result out of these two, if merging is possible, could be best captured by ***logarithmic functional***. Explanation for this is very simple in the sense that once the decision tree is created and testing data is being run through it, each of the decision trees based on the function used will have a different classification rate. Through the process of taking different slices of training and testing data, one functional will perform different than the other. To perform the most valid testing is to take same sets of training and testing data and apply them against each of the observed functions. Using training and testing data provided like this, results can be compared and the best functional can be assigned as a representative of that data set. The possibilities are numerous as presented in the next two chapters through the results achieved.

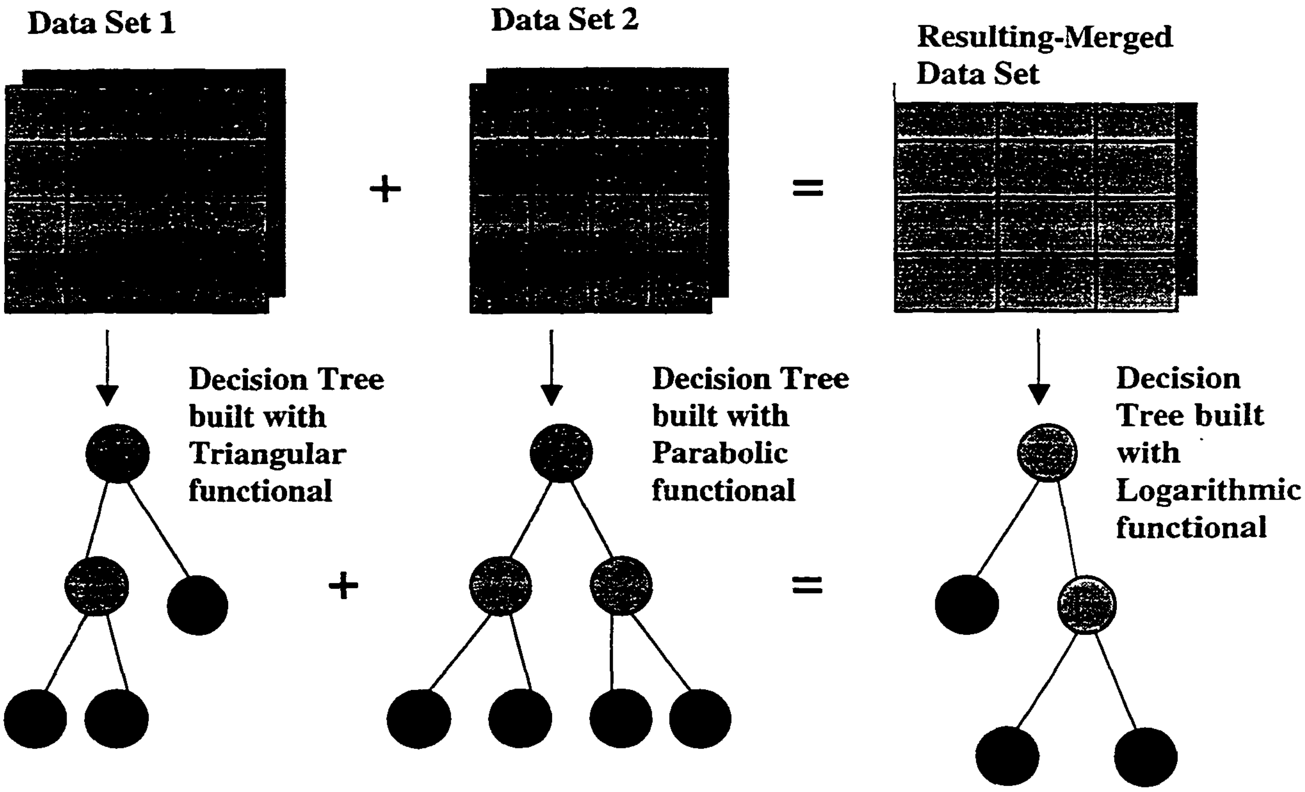

Data Set 1
Data Set 2
Resulting-Merged Data Set
+
=
Decision Tree built with Triangular functional
Decision Tree built with Parabolic functional
Decision Tree built with Logarithmic functional
+
=

Figure 3.5 Different data sets and their best representation with different functions

Entropy functions created will differently capture the data, since the data point distribution of particular data set will match certain entropy functional better or worse than the others observed. Based on the information gain calculated for a particular attribute in accordance with the functional chosen, the entire built decision tree structure, could be different based on the choice of the functional. Following chapters provide more discussion and analysis on features and functionality of different entropy functions, what determines which functional would be more suitable than the others and determination of their general tendencies uncovered in this research.

3.4 Handling the Missing Attributes Using Database Features and Approach

There are many different approaches and ideas when it comes to handling missing data in a dataset. It is outside of the scope of thesis to discuss those methods.

The idea proposed here is straightforward (in the database sense) and it reflects the concept of the database approach and methodology. The proposed methodology is as follows, a data set with several attributes that have missing values is a starting point. The size of this data set is reduced by one attribute at a time using the same approach as when building the decision trees. First the attributes without missing values are exhausted in the order of their physical appearance in the data set (from left to right in the two-dimensional table). This continues until only the attributes with the missing values are left in the data sets and available for future observations. Each of the attributes with the missing values is a part of the sub sets of a decision tree like structure created as a fraction of each of the elimination rules obtained till that point. This build continues until all of the attributes with the missing values are subdivided in subsets ending with the subsets with the attribute with the most missing values in it.

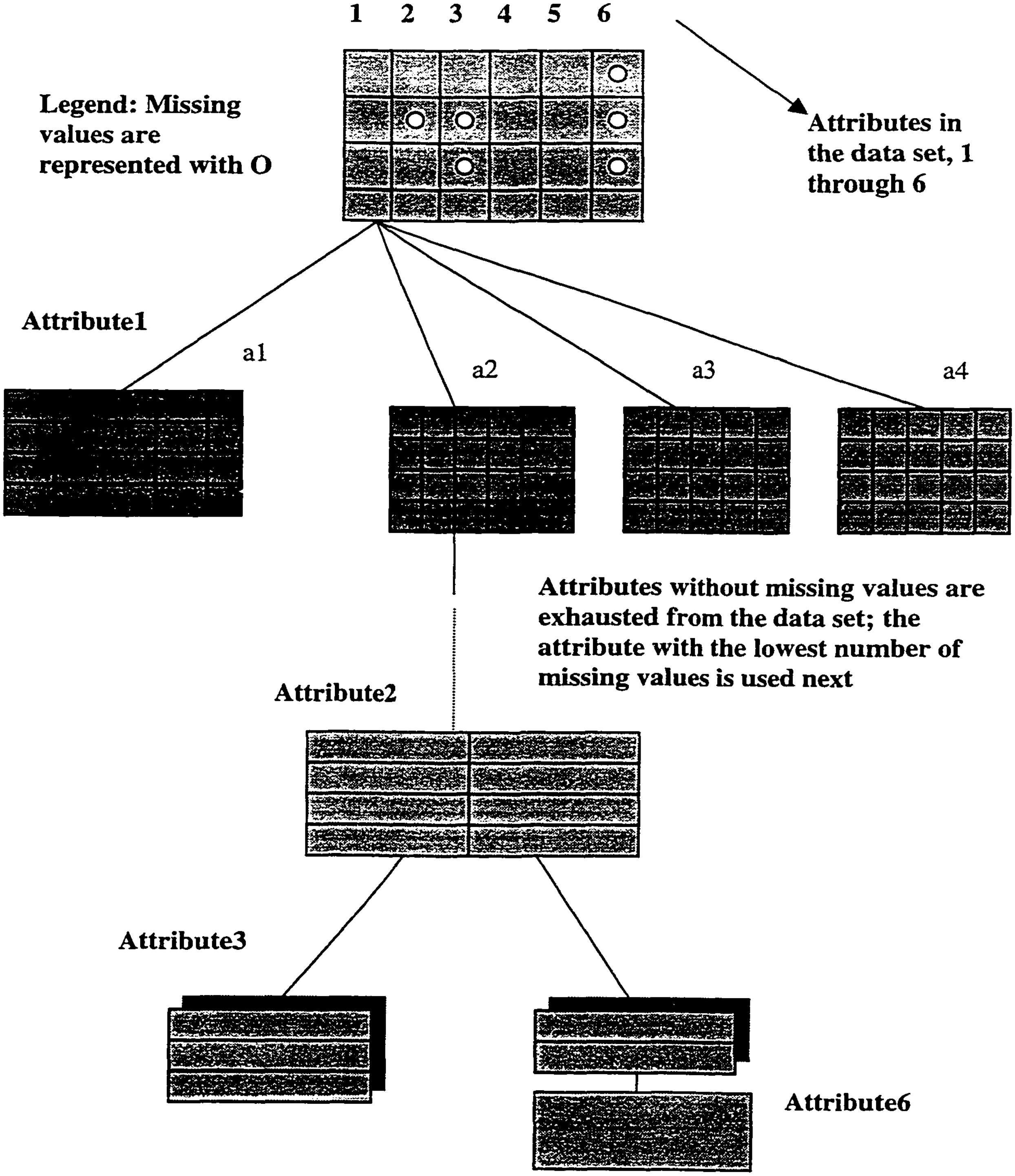

1 2 3 4 5 6
Attributes in the data set, 1 through 6
Legend: Missing values are represented with O
Attribute1
a1
a2
a3
a4
Attributes without missing values are exhausted from the data set; the attribute with the lowest number of missing values is used next
Attribute2
Attribute3
Attribute6

Figure 3.6 Handling the missing data through subsets

The best approaches to better comprehend this method and a flow represented in Figure 3.6, is to give an example with real numbers.
Example 3.4

The data set with 7 attributes is observed and 3 of the attributes from the data set have missing values in them.

The first of four attributes without missing values is eliminated and for each of the distinct values, that attribute has a partition of the initial data set that is created as a subset. If attribute A has values 1, 2 and 3 then there would be three subsets created and this attribute eliminated from the query. Next, for each of the subsets based on distinct values of the second attribute, the second layer of subsets is created. In this way, structure similar to the decision tree node structure is created.

This continues until the attribute with the lowest number of missing values is reached and all of the attributes without missing values are eliminated. In each of the subsets (there can be quite a few of them based on the complexity of the data set), replacement of the missing values are performed using the following approach.

If the attribute with missing values is named attribute G, and the observed subset contains 17 values (records), where 11 values of attribute G are known and 6 values are unknown (missing), the above outlined method can be used. Values of attribute G in observed subset can take any of the following numbers: 1, 2 and 3.
Based on this the number of distinct values of attribute G is obtained for each of the distinct cases.

7 out of 11 known values in the observed subset have a value of 1.
3 out of 11 known values in the observed subset have a value of 2.
1 out of 11 known values in the observed subset have a value of 3.

To replace missing values with known values from the same subset, the ratio of known values for each of the attribute's distinct cases over the overall number of known values for that attribute is multiplied by the overall number of unknown cases.
$(7/11)\cdot 6=3.85$ unknown values to be replaced with a value of 1.
$(3/11)\cdot 6=1.68$ unknown values to be replaced with a value of 2.
$(1/11)\cdot 6=0.545$ unknown values to be replaced with a value of 3.

Obviously, from the first case 3 values will be replaced with a value of 1, from the second case, the 1 value would be replaced with the value of 2, and so on.
This would provide replacement for 4 out of 6 missing values in the subset.
Round off of the digits behind the decimal point is performed with respect to the cases that have the largest number behind the decimal point.

The final replacement based on round off of the digits would be 3+1=4 values replaced with a value of 1, because 0.85 is the largest out of 0.85, 0.68 and 0.545.

1+1=2 values replaced with a value of 2, because 0.68 is greater than 0.545.
0 values are replaced with a value of 3. This procedure carries on for all of the subsets created with this divisive method. Once completed, this attribute is eliminated and the next attribute with a higher number of missing values is observed in its respective subsets one layer below.

The strength of this method is that each of the missing values is replaced with respect to the most probable and similar known values of the same attribute and also with the highest affinity with respect to the other attributes in the data set.

The question that arises here is how much replacement can be done on the data set without tampering the overall results and authentic structure of it. From practical experience, if any of the attributes have more than 5-10% of missing values in it, all records with the missing values should be deleted, not replaced, and estimated by this or any other method.

3.5 Pruning of Decision Trees

3.5.1 Data Set Size Criterion

Many research papers that deal with decision trees in general address the issue of complexity and computational overhead if the entire data structure and the decision tree realization are to be carried. Some of those methods and research from the past were discussed and mentioned in a previous chapter. Although, with database tools and database structure, the concern about complexity and size of the decision trees (also computational efficiency of it) is not as big of a problem as in the past. For the sake of end users and anybody else who might use these algorithms and software in the future, intent is to eliminate unnecessary node calculation and rule representation if possible, or at least to provide an option if desired to do so. In the database world, one of the most important factors (if not the most important) is size. If there is a data set with a size of a few million records and some of its subsets are only a few thousand records, then subsets such as these ones would not impact overall analysis of the data set structure in a significant way. There are two ways given in this thesis for performing pruning of decision trees, or even better, two approaches implemented and tested in this research for pruning. Both methods as outlined in the title of this section deal on the premises of database logic and approach trying to differentiate unnecessary computational steps and storage overhead created by the extra sub-trees created if a full decision tree build is to be carried on.

The first method considers the size of the initial data set and usage of LD and NA factors introduced in section 3.2.2. The following formula for the pruning algorithm is given

$$\text{Elimination\_threshold}=\text{size\_of\_subset}\cdot(1+\text{NA})\cdot(1+\text{LD}) \qquad (3.5)$$

where size_of_subset is the number of the records (experiments) in the observed subset (intermediate node).
Example 3.5.a

The initial data set with 500,000 records is observed and the elimination_threshold of 2% is set as an arbitrary value (if the subset is less than 2% of the initial data set then eliminate the node it resides at).

LD is the layer depth divided by 10 to obtain the decimal representation and NA is the factor for the number of attributes in the subset.

If pruning_threshold is set at 0.02 and the size of the subset is 7500 (number of records (rows) in it) with LD=0.2 and NA=0.2, then based on eq.3.5

Elimination_threshold=7500·1.2·1.2=10,800
which is greater than 0.02·0.5·1000,000=10,000 (2% of the initial data set) so this node and its subsets would be included in the next step of calculations.

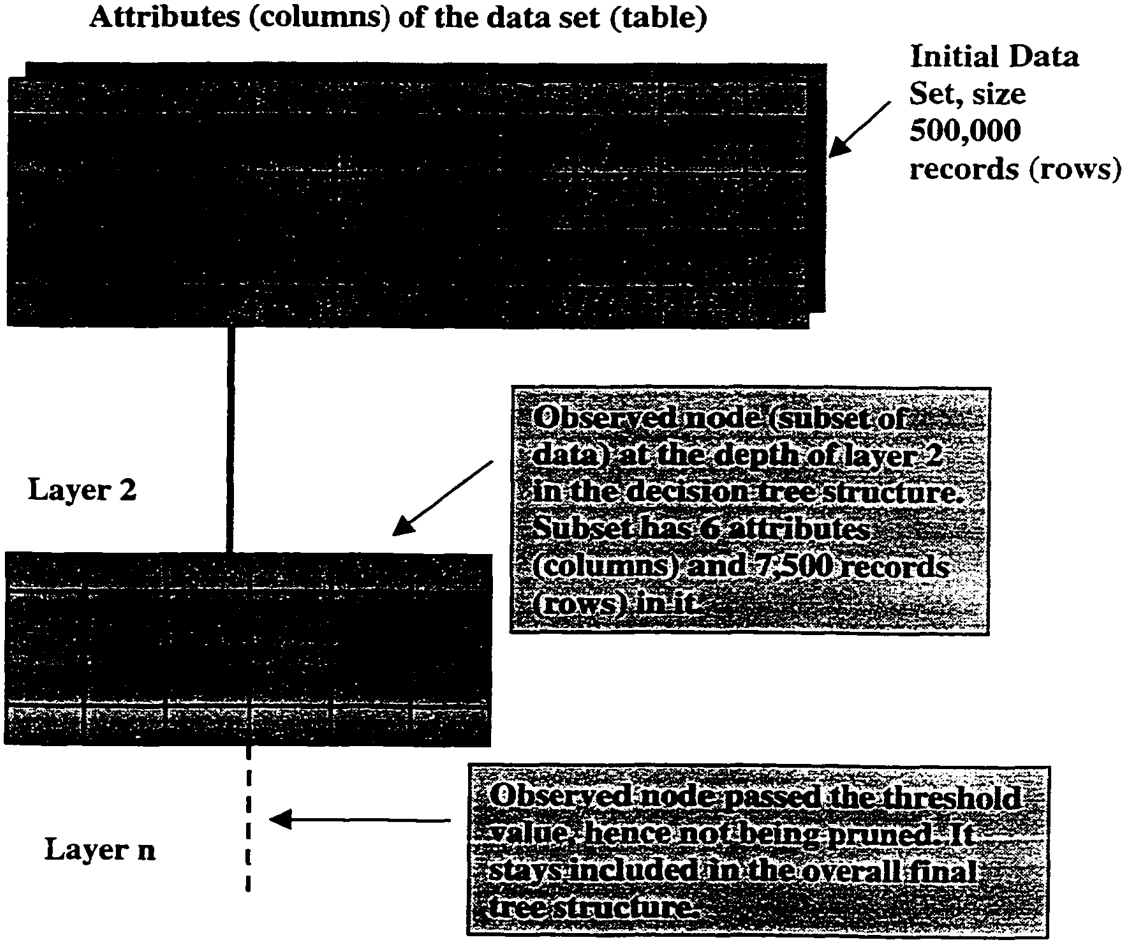

Attributes (columns) of the data set (table)
Initial Data Set, size 500,000 records (rows)
Layer 2
Observed node (subset of data) at the depth of layer 2 in the decision tree structure. Subset has 6 attributes (columns) and 7,500 records (rows) in it.
Layer n
Observed node passed the threshold value, hence not being pruned. It stays included in the overall final tree structure.

Figure 3.7 The data set size criterion method for pruning of decision trees-process flow

3.5.2 One-Layer Method

The second way is more straightforward and much easier to implement and follow. Simply, by comparing the size of the data set one step above the observed subset and determining if it satisfies the given pruning_threshold or not. Its process flow is presented in Figure 3.8. This approach is best described as a ***one-layer method.***

If the subset size is greater than pruning_threshold (a numeric value provided by researcher or subject matter expert), then calculations on that node and subset are continued. If not, the node is eliminated (pruned) from further computations and a final decision tree structure. The following example together with Figure 3.8 explains the implementation and analysis of this method.

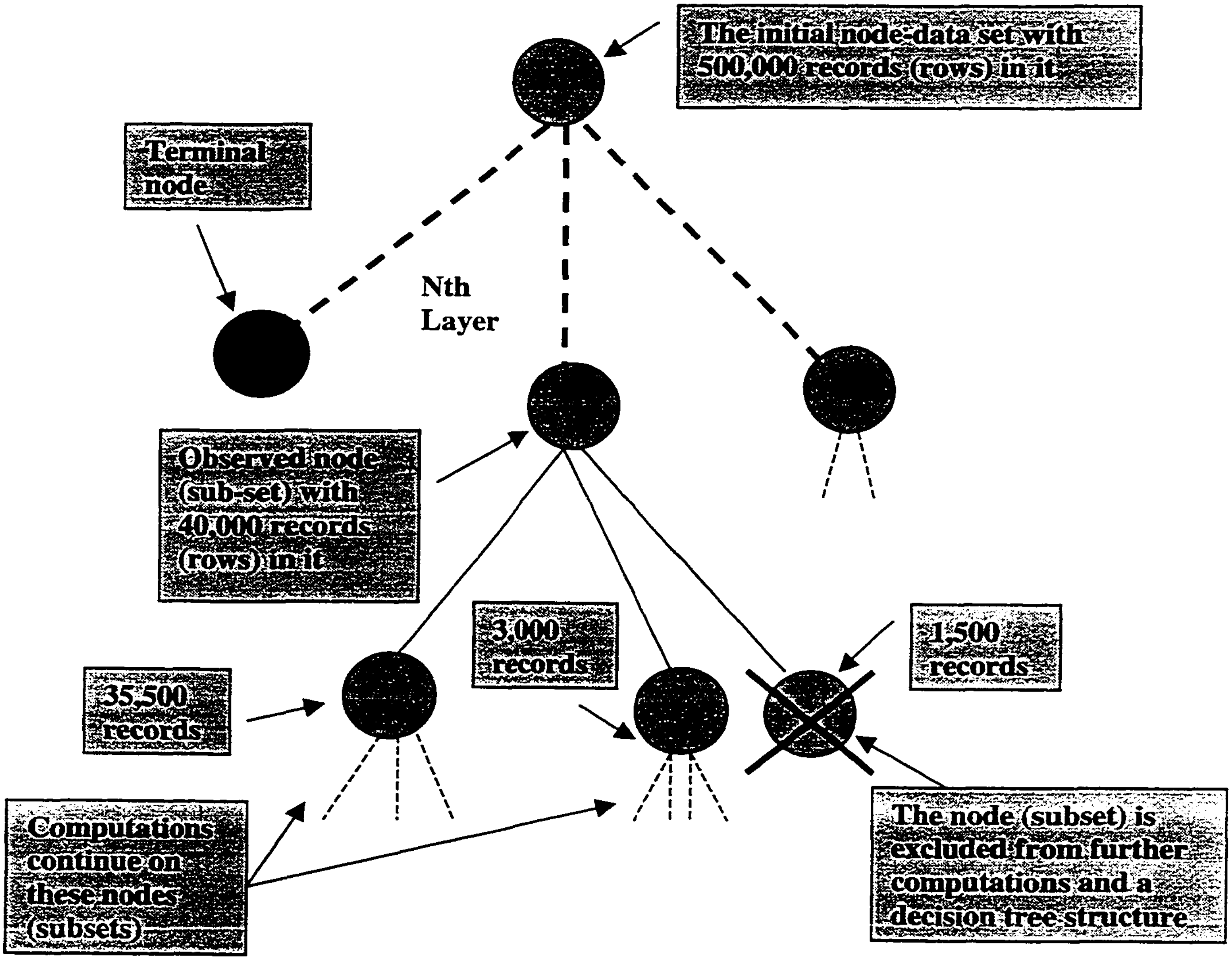

The initial node-data set with 500,000 records (rows) in it
Terminal node
Nth Layer
Observed node (sub-set) with 40,000 records (rows) in it
3,000 records
1,500 records
35,500 records
Computations continue on these nodes (subsets)
The node (subset) is excluded from further computations and a decision tree structure

Figure 3.8 One layer method for pruning of decision trees

As can be seen from the Figure 3.8, the third node under the observed node with 40,000 records is not passing the pruning_threshold rule of 0.05 (5%). The first node and the second one are included in the next computational step and the third node with its subset is excluded from further calculations. Even though the second method is slightly easier to comprehend and implement in theory, both are worthy of attention and as such, results are provided and analyzed for both methods in the following chapters. The intent of the decision tree algorithm is to create meaningful rules that can capture relations among attributes in the data set properly. Each time, after either of the pruning methods introduced here is applied to the observed decision tree, the classification rate of the pruned tree should maintain an acceptable level. If that is not the case, the elimination_threshold of the pruning algorithm should be modified until successful results are obtained. This observation is also taken into consideration during the experimental part of this research and the results of pruning methods on decision trees are compared and analyzed with respect to that.

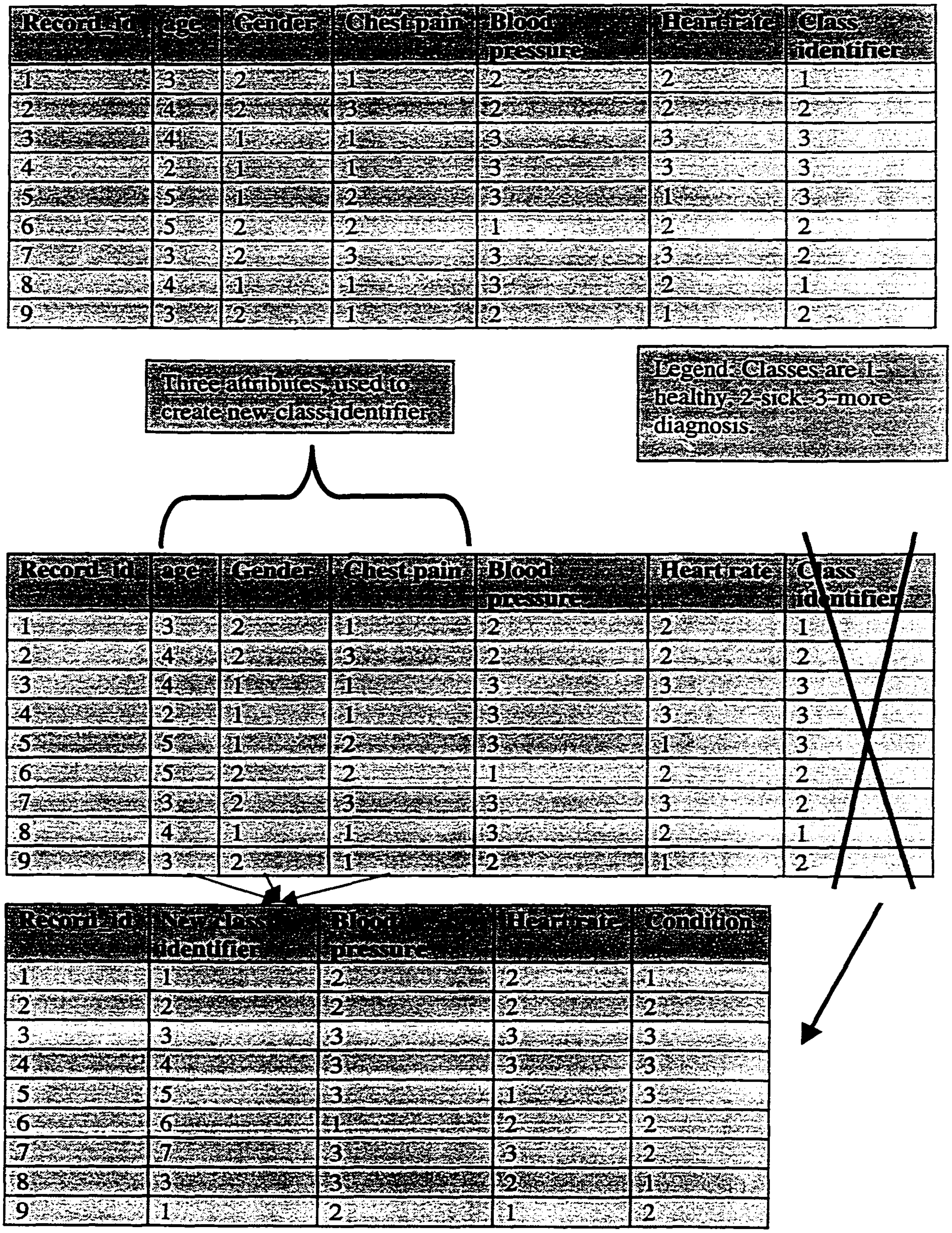
3.6 Multi Attribute Class Identifier

The need for multi-dimensional data representation and its implementation has been increasing in demand over the last decade. Often the end user or researcher would like to see the relationship of multiple attributes and their values with respect to the rules generated from the dataset. With the single attribute identifier class, only relationships of the data set and rules generated to that particular identifier class are visible and all of the other attributes in the data set are just the constituent parts of the rules generated. By combining the attributes in the composite classes, the ability to verify the overall behavior of the data set and remaining attributes with respect to the chosen set of multiple attributes is very important. For example, instead relating the health of the patient to the overall diagnosis of healthy or sick, the overall patient profile could be identified with respect to diet, gender, education, ethnicity and territory. In data mining, this can be represented by choosing the class identifier that consists of two or more attributes combined as the composite class. The idea is to choose two or more attributes and create a composite class identifier which values would consist of every possible mathematical permutation of values in the attributes chosen. The applications of multiattribute classes can be used in marketing research, healthcare or overall behavior of different attribute groupings based on the research objectives, just to name a few. Two very illustrative examples with an appropriate figure are given next, that explain the development of multiattribute classes step by step.

Example 3.6.a

Instead of a class identifier for the heart data set as the patient being sick, healthy or needing more diagnosing, researchers may want to create their own class identifier as all possible combinations of age, gender and chest pain type. If the age attribute has 5 distinct group values, gender has 2 values and chest pain has 4 values, the overall number of distinct combinations of all three attributes is 5*2*4=40. Each of the 40 possible distinct combinations is considered to be an independent class in this case. Results of the data mining are presented with respect to forty possible (not all of them have to exist based on the cases in the data set) different classes that are created as combinations of these three attributes. So, instead of one attribute hence one dimension, we have three attributes - three dimensions. The process flow and a visualization of the method that is proposed and implemented in this research are provided in Figure 3.9. The previous class identifier has become just an attribute in the modified data set, the attribute being named Condition.

Worthy noting is that the column named Record_id in the data set is used to store the information about the data set prior to the manipulation and introduction of multi-class attribute identifiers. Once the researcher decides to back step and traces the results, this innate database feature called sequence number generator, enables just that in seamless fashion. This feature is also used to track the data set all the way to the original data set with the continuous and textual data values in it. This definitely is not possible to achieve outside of database structures (at least not without serious computational overhead and theoretical design first).



Record id	age	Gender	Chest pain	Blood pressure	Heart rate	Class identifier
1	3	2	1	2	2	1
2	4	2	3	2	2	2
3	4	1	1	3	3	3
4	2	1	1	3	3	3
5	5	1	2	3	1	3
6	5	2	2	1	2	2
7	3	2	3	3	3	2
8	4	1	1	3	2	1
9	3	2	1	2	1	2

Three attributes used to create new class identifier

Legend: Classes are 1-healthy, 2-sick, 3-more diagnosis.

Record id	age	Gender	Chest pain	Blood pressure	Heart rate	Class identifier
1	3	2	1	2	2	~~1~~
2	4	2	3	2	2	~~2~~
3	4	1	1	3	3	~~3~~
4	2	1	1	3	3	~~3~~
5	5	1	2	3	1	~~3~~
6	5	2	2	1	2	~~2~~
7	3	2	3	3	3	~~2~~
8	4	1	1	3	2	~~1~~
9	3	2	1	2	1	~~2~~

Record id	New class identifier	Blood pressure	Heart rate	Condition
1	1	2	2	1
2	2	2	2	2
3	3	3	3	3
4	4	3	3	3
5	5	3	1	3
6	6	1	2	2
7	7	3	3	2
8	3	3	2	1
9	1	2	1	2

Figure 3.9 Multi-class (attribute) identifier-process flow for heart data set

Example 3.6.b
For better clarification, the multiclass idea and method can be explained by the sample data set that contains the information about the RRSPs purchased in several Canadian provinces by the sample of population. In the data set, attributes such as income, education and occupation are going to be detrimental for decision-making choices of population whether they will purchase an RRSP or not. A RRSP is a long-term investment and security, combined with a tax deduction, hence certain thinking and financial resources are expected before deciding to purchase a RRSP. In the basic normalized (discretized) RRSP data set, a class identifier is a simple two-class attribute with values ***buyer*** and ***non-buyer***. After the initial data mining is performed, the fact is that income, occupation and education are most dominant attributes in the node building of decision trees with other attributes distributed in the lower sub-structures of the decision tree created. To increase interestingness of this data set and research on it, a multiclass identifier combining of three attributes (income, education and occupation) is created. After data mining, many interesting facts can be discovered, such as the population defined by a class consisting of higher income, lower education and farming occupation attributes is represented by the attributes of being RRSP buyers, owning a farm, two cars and no other stock investments. Other classes created by the above method also yielded interesting facts not obvious by the initial data mining and research.

Example 3.6.c
Another example is the data set from the car dealership, where interesting facts could be discovered when the multiclass attribute is created from the following attributes (car color, gender and make). People who are identified by the class of blue color cars, higher quality sedans and males are found to be of technical occupation, ages between 30-55 and they are predominantly married. Such a fact would definitely help the car dealer to target the right market niche if suddenly large shipment of cars with above characteristics just arrived on the lot.

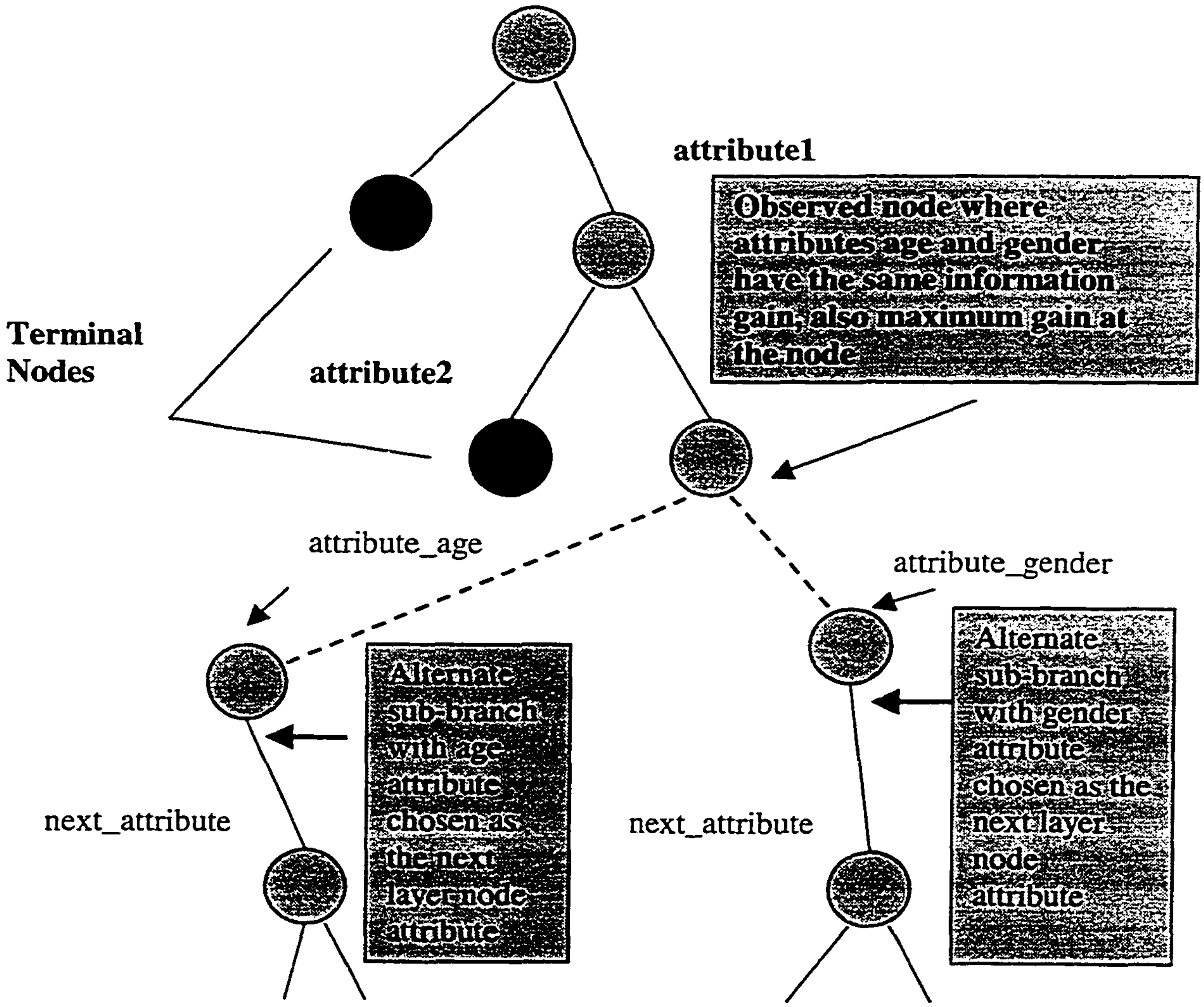
As can be seen, multiattribute class generation is very useful in its conceptual and research frame, but not less in its business, industry and health care application.
The computational overhead for this method to be implemented in the research is high, but possibilities that occur as the outcome, are outweighing it by far. The ability to choose multi dimension selection criteria is a beneficial one and as mentioned it can be implemented in any kind of health care research, marketing survey or technical field.
How does the multiattribute class affect the classification rate built? Since there will be more classes, a decision tree built will have more of a flat (less levels) structure with more terminal nodes. In the case of data sparsity (not enough cases and records of data in the data set), many test cases might not exist in the decision tree, or even the tree itself can have a low classification rate (training cases are sparse and non-matching too).
So, the answer is yes, on the data sets of characteristics specified above, the classification rate of both training and testing data could be lowered. With this in mind, perfect candidates for this approach are the data sets both context-rich and size-large or at least one of the two. Further discussion about the classification, complexity and usefulness of this method is given in the next chapters.

3.7 Alternate Branching of Decision Trees

The aspect of alternate branching of a decision tree was not found in literature before. Alternate branching of the decision tree occurs in the case of equal information gains among two or more competing attributes at the same tree node, which would result in the realization of one or more alternate decision trees.

The problem that is invisible to the public if not exposed, is what happens in the case of reaching the node that has more than one attribute with the same value of information gain of the attributes, hence qualifying more than one attribute as the most determinant attribute in that node.

The solution suggested is to create alternate tree branches at the node(s) that are experiencing such a situation and calculate all of the resulting sub-trees for each of the possible alterations.



Observed node where attributes age and gender have the same information gain, also maximum gain at the node

Alternate sub-branch with age attribute chosen as the next layer node attribute

Alternate sub-branch with gender attribute chosen as the next layer node attribute

Figure 3.10 Alternate branching of decision tree(s)

Example 3.7
In the heart disease data set used for the research, the situation occurred at one of the nodes with four attributes, that two attributes (age and gender) had the same information gain. The attribute gain for those two attributes had the highest information gain of all attributes at that particular node. What this would result in if not considered carefully, is that the algorithm would choose one or the other, probably based on the order of precedence in the computational array. This eliminates the problem of having two or more attributes competing for the place of the most dominant attribute in the node. On the other hand, this limits all of the possible outcomes and sub trees being observed and presented at the end. In the past, there was no mention of this problem or a possible solution to it. The solution is to create for each of the attributes, the alternate sub tree, as in Figure 3.10, properly mark them as alternate branches coming out off the affected node(s) and to present them at the end. This provides the researcher with the option of seeing all of these outcomes and determining if any meaningful advantage can be deciphered from one alternate tree to another. The reason that this can be done, is the characteristics of the database, to be able to store intermediate decision tree nodes permanently and reuse them any way that the researcher or algorithm sees fit.

3.8 Terminal Nodes Likelihood-Proximity of Decision Tree Subsets

The idea for terminal nodes likelihood is based on the premise that two child nodes coming from the same parent node can be merged together if both terminal nodes belong to the same predominant class. As explained in section 3.2.2 (eq. 3.1), the confidence of a terminal node is the ratio of the most predominant class in the node over the sum of all of the classes in the node. If both children terminal nodes can be merged and the resulting confidence ratio is still higher than some preset threshold value, then the simplified node structure can exist under the parent node. That node becomes the terminal node and two (or more) child nodes are eliminated or replaced by the one larger child node with the different associated rule.

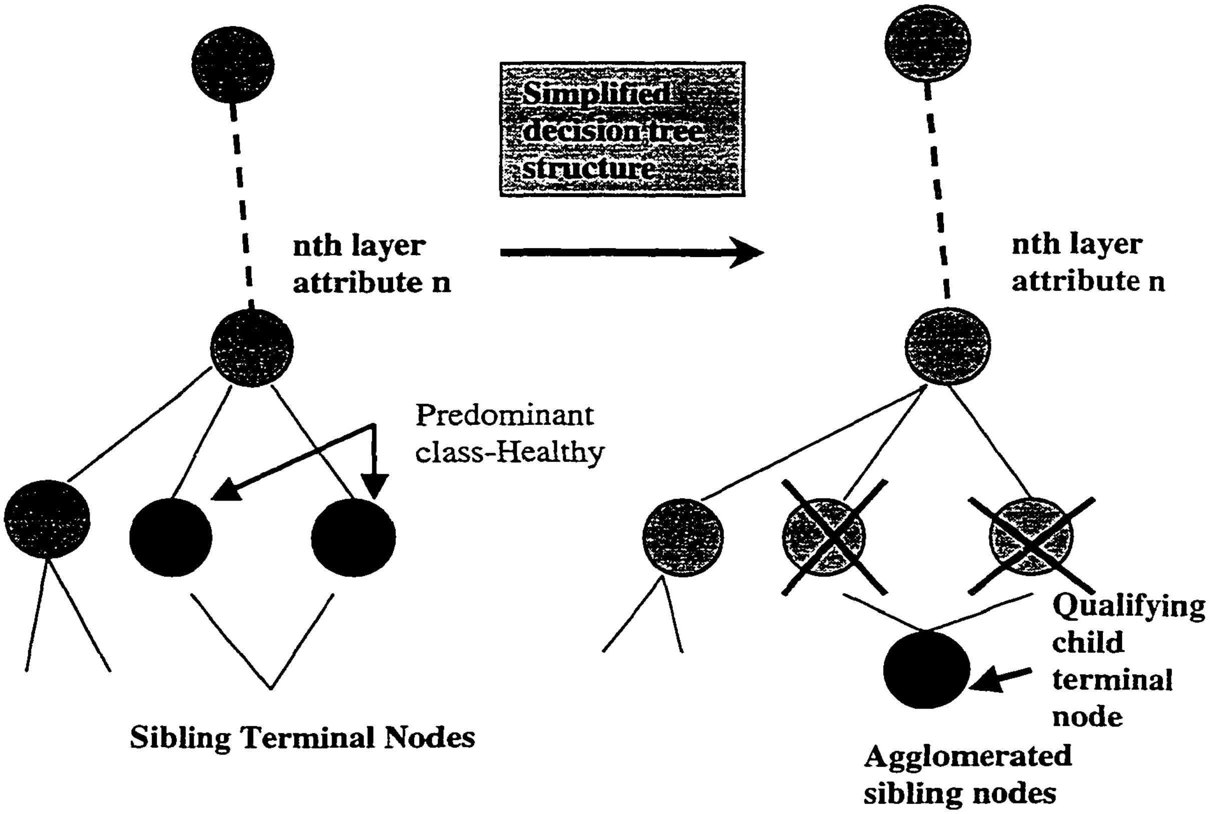

Simplified decision tree structure
nth layer attribute n
Predominant class-Healthy
Sibling Terminal Nodes
nth layer attribute n
Qualifying child terminal node
Agglomerated sibling nodes

Figure 3.11 Proximity of sibling terminal nodes-process flow

Example 3.8
In the observed heart data set, two child terminal nodes at the 8th layer of the decision tree both belong to the same predominant class of healthy patients. One node has 100% confidence ratio and two records in it, while the other one has the 50% confidence ratio with 6 records in it. When combined back into the parent node, the newly chosen parent terminal node has a confidence ratio of 62.5%. If this ratio is higher than the preset threshold value, the decision tree structure can be simplified without loss of data. This agglomeration of children nodes can continue on any given sibling terminal node if confidence ratios are of the same nature and if the outcome ratio of the merged nodes into a new terminal node also qualify. The method and its process flow are presented in Figure 3.11 above.

Also, the starting child nodes should pass some threshold value set for their initial homogeneity before proceeding with any further merging.

Even though this method has similarity with the pruning methods of decision trees in the sense that it tries to simplify the decision tree structure it is very different. The differences are that the tree is being built first to its maximal value with the full structure, and later the structure gets simplified, but unlike with pruning without any data (records) being discarded (records are agglomerated in the super-child node-subset above).

Obviously, the rule created at the merged child node will be more compact and not as simple since they will involve the Boolean OR operator to represent rules merged from two or more different sibling nodes. Still, using the inherent database tools and capabilities, even a newly created rule is a simple one. The advantage of this is that when using decision tree structures for the purpose of a testing analysis (or its predictive capabilities), the computational algorithm does not have to verify two individual rules of similar nature, but one rule represented with both AND and OR operators in it. This can be extremely useful in large databases where the test data sets can be millions of records. Another advantage is having a simplified database structure with lower storage and computational requirements.

3.9 Manipulation of Decision Tree Nodes Using Database Inherent Structuring

As a product of any computation and calculation done in the database space (with database engine), data sets (tables) are created. Whether those are temporary or permanent in nature is up to the developer or researcher to decide. When building decision trees in databases, nodes both intermediate and terminal in the decision tree structure are represented by the datasets (sub sets) in the database, as explained earlier. With each node being represented by the dataset, manipulation of the data sets (nodes) can be achieved using database tools and capabilities. This can be performed in two different ways, one of those is by enriching the select node(s) with data records which in return would bias the consequent build of nodes at that particular branch, hence altering the classification of the decision tree.

The second method is to isolate the selected intermediate node(s) and to perform a decision tree build on them by using the different entropy functional from section 3.3. After that is done, classification of the testing data set can verify whether different entropy functional did or did not improve the classification of the decision tree, or even if selection of different functional did simplify the decision tree sub structure in question, hence lowering the complexity and computational overhead.

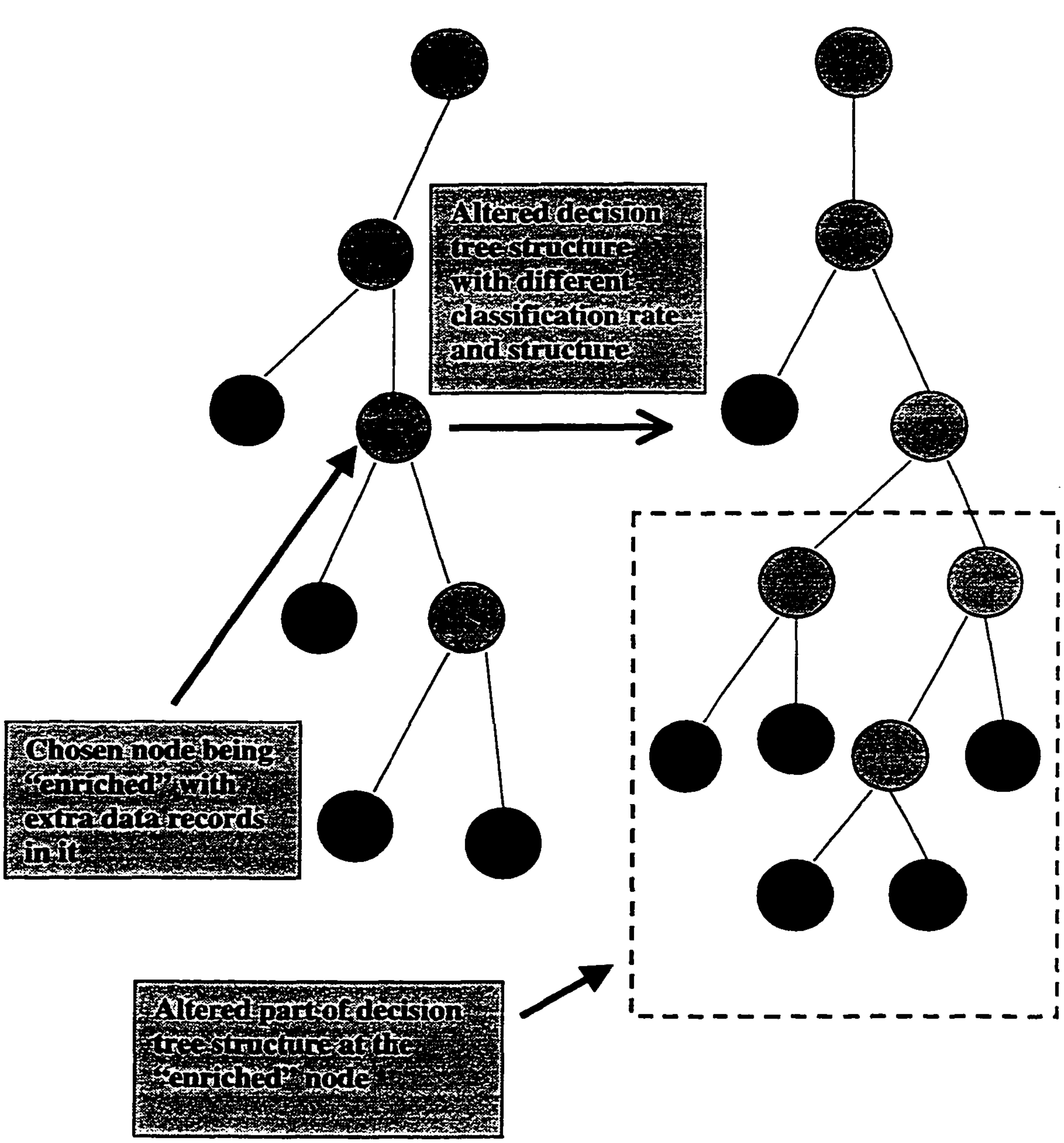

Altered decision tree structure with different classification rate and structure
Chosen node being "enriched" with extra data records in it
Altered part of decision tree structure at the "enriched" node

Figure 3.12 Node manipulation-adding (subtracting) data records from the node

The important fact to note is that data from the node(s) can be both added and removed. In this way, nodes can be pruned or added based on the direction taken by the researcher. This approach can also be utilized in the predictive method with decision trees, by biasing the predictive outcomes of the observed data and guiding it with the manipulation of the nodes (data sets). It is necessary to utilize and not over utilize the decision trees with this method, since by introducing to many data values in the observed node(s), classification of the test data set can become over-optimistic (overfitting).

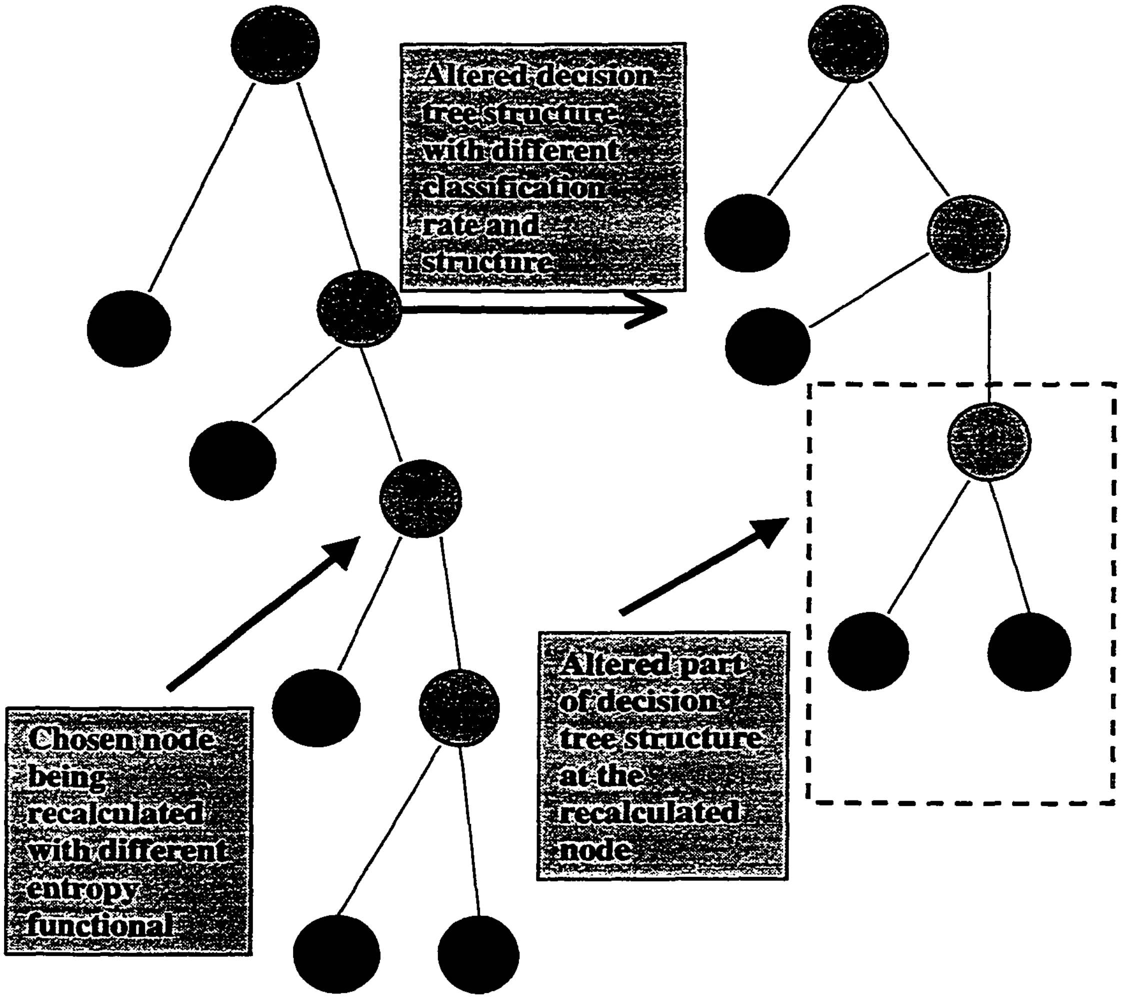

Altered decision tree structure with different classification rate and structure
Altered part of decision tree structure at the recalculated node
Chosen node being recalculated with different entropy functional

Figure 3.13 Node manipulation-recalculating node(s) with different entropy functional

As per the first method, careful consideration has to be given about which node(s) to select and recalculate with different functional, since the result might prove to be unrealistic in the sense of the decision tree structure achieved or classification rate established.

3.10 Analysis, Implementation, Tuning and Modifications of Implemented Algorithms

All of the activities named in the section title are implemented, analyzed and discussed in the following chapters. The intent was to obtain research results of high quality and a Ph.D. with respectable substance in the category of novel ideas implemented and properly tested. To achieve all of this, profound knowledge of databases and database tools is needed.

As seen from the previous work, one of the disadvantages was an insufficient hardware and software environment. Quinlan [133] was forced to use a technique called windowing to deal with data sets with a small number of records from today's perspective. The reason for this was lack of available memory and a database structure such as RDBMS databases.

3.11 Conclusions

In this chapter novel and original ideas developed and implemented in this research were given together with the necessary examples and descriptive figures and tables. Ideas given and explained were decision trees in a multiclass environment, multi datasets algorithm, new entropy functions, missing attributes algorithm, pruning methods, multiattribute class identifier, alternate branching, manipulation of decision tree structures and terminal nodes.

In the next chapter, experiments performed and research results obtained are given. Numerous data sets have been used in various forms for the purpose of testing, data manipulation and data mining are used to better utilize and modify the methods and algorithms given in this chapter. Some of the experiments are performed as mentioned for ideas (objectives) that are not the main objectives of the research, and they might have been explored in a different research capacity somewhere else, but it is worth pursuing for sake of completeness of this research. This support research experiments are in the space of predictive capabilities of decision trees obtained in the next chapter-decision trees created by using the methods suggested in this chapter as the main objectives of this research. Also decision trees and database approach can use obtained results and rules to isolate any single continuous value in the pre-discretized data set, hence providing even better understandability and comprehension for the researcher.

CHAPTER IV

Experiments and Results

The research performed in this chapter covers the ideas presented and discussed in Chapter III. Medical data sets used are obtained from the Internet site ftp://ftp.ics.uci.edu/pub/ at the University of Irvine [156]. The financial data set with the RRSP data used is provided courtesy of Faneuil Inc [138].

4.1 Data Sets-Description and Preparation

Preparation of the data sets were performed in logical steps by dealing first with the missing values and discretization of the attributes (values) afterwards.

In some of the data sets, missing values were eliminated hence lowering the size of the initial data sets obtained.

Once the analyzing of the data in the data sets obtained was completed, it was observed that values in some of the attributes were continuous (as expected).

After consulting with the experts in the medical and financial fields, the following discretization of an example data set was used with the example provided for the heart data set (not all of the data sets were discretized in this fashion).

AGE	REPLACED WITH	DESCRIPTION
32-43	1	very young
44-50	2	young
51-59	3	middle
60-65	4	older
66-80	5	old

TRESTBPS-RESTING BLOOD PRESSURE	REPLACED WITH	DESCRIPTION
80-114	1	low
115-135	2	normal
136-200	3	high

Table 4.1.a Discretization (by professional) for attributes AGE and TRESTBPS in the heart data set-example

restecg (heart rate while resting) was updated with 0 (low), 1 (normal) and 2 (high).

The other attribute's values were replaced with numbers from similar ranges and notations. An important fact is that when discretization is a guide by the professional classification and error rate of the final decision tree, structures could be expected to be lower than with other discretization algorithms mentioned below. In the Appendix A, discretization values for the RRSP data set are provided for informative purposes.

Other discretization methods used and compared with the user-guided method were a uniform discretization of continuous attributes and also context based fuzzy clustering. Context based fuzzy clustering was performed on certain attributes in the chosen data sets to compare that very useful approach with the other ones. Ground breaking work on this particular method and its application in discretization of attribute values is given in [118] work done by Pr. Dr. W. Pedrycz of Alberta University and his associates.

In the case of uniform discretization the user chooses an interval, and based on this there can be a different number of discretized attribute values. If the interval chosen is greater, than the number of the unique (discretized) attribute, values will be lower. When uniform discretization is used and compared to the professional one, if the professional one is done correctly on the complex data set, overall classification of the decision tree can be expected to be better for the professionally discretized data set and vice versa (in the case of incorrect professional assumption during the discretization process). Splitting the initial data sets randomly, created the training and testing data sets. The only fact specified during the creation of the training and testing data sets was the percentage of the split of data. Factors used in the calculations and mentioned in the examples in the previous chapter are explained and presented as follows
NA factor assumes the following values,

For the data set with 1-10 attribute values NA=0.2
For the data set with 11-20 attribute values NA=0.4
For the data set with 21-40 attribute values NA=0.5
For the data set with 40+ attribute values NA=0.6

DSS factor assumes the following values,

For the data set with 1-10,000 records DSS=0.1
For the data set with 10,000-30,000 records DSS=0.2
For the data set with 30,000-100,000 records DSS=0.3
For the data set with 100,000-500,000 records DSS=0.4
For the data set with 500,000+ records DSS=0.5

Values for DSS and NA factors are not calculated but assumed, hence they can be changed based on end user input or after several different trials with different values, to choose one set of values that captures the ideas developed and the nature of the observed data set(s) structure into software in the best possible way. Based on the previous, they can be experimentally determined if needed.

Overall description of data sets used for the research is provided next in tabular format. These are the data sets with the records without missing values in them only.

Data set name	Number of attributes	Number of records
Swiss Heart Data Set	11	105
Hungarian Heart Data Set	11	261
Liver Data Set	8	345
RRSP Data Set	15	9492

Table 4.1.b Description of data sets used and their characteristics

4.2 Generic Algorithm and Calculations

Quinlan's [133] approach was explored and incorporated on both the conceptual and structural level. The algorithm is implemented with previously discretized data sets (step 1).

The size of the data sets is of no concern. Any node of the tree can be examined later on, because of tabular structure achieved that preserves data.

First, the entropy is calculated using any of functions introduced in chapter three. Then, the gain of each attribute and node is created based on the highest gain. A decision tree generic algorithm was shown in Chapter II, section 2.3.

This is where the new ideas from Chapter III are introduced and implemented in the software package. Also important of noting, is the fact that in the calculations used in this research, only the **information gain** was calculated and used because **gain ratio** has a tendency of leveling the biases among attributes in the data set. In database research, the intent is to obtain the results that will favor the attributes that are inherently more dominant than the others, not to make them all even.

4.3 Novel Approach to Decision Tree Design and Implementation

The languages used as already explained are SQL and PL/SQL on an Oracle 8 platform. Instead of hard to search flat files and C code, database structure and PL/SQL (SQL) language were used to achieve superior structure design and size insensibility (obviously there is extra computation with the size increase or more attribute values in the data set but databases are designed for challenges in data size, dimensionality and simplification of normally complex queries). All of the ideas from Chapter III were tested and implemented in this chapter such as calculations done for four different forms of entropy functional, namely logarithmic, parabolic, sinusoidal and triangular. As expected, the results obtained for each functional were different based on choice.

Database structure allowed for easy use of more than two classes-***multiclass*** approaches.

One of the important points of the research was how to merge different data sets with different attributes into one (a case from real life and any major organization) as well as how to determine what to keep, and what to discard. Work was done also on different data sets with a lower number of attributes in them that were created from the original data sets. For example, the heart data set from Switzerland has 9 attributes, and by removing different combinations of two attributes from the original data set, four data sets with 7 non-identical attribute values were created.

For the first data set the attributes removed were OLDPEAK and SEX.
For the second data set attributes removed were SLOPE and TRESTBPS.
For the third data set attributes removed were SEX and AGE.
For the fourth data set attributes removed were OLDPEAK and AGE.

Rules obtained from these four data sets were then used to create a final merged data set with the values pertinent to all of sets. The results obtained from all of the ideas in Chapter III were tested and implemented in this chapter such as decision tree pruning cases, missing attributes, as well as the others.
In the next sections, these experimental results performed on data sets in the course of this Ph.D. research on the ideas presented in the previous chapter are given in the tabular form with the explanatory figures where relevant and necessary.

4.4 Experiments and Results

The results produced were generated using aforementioned data sets. Various test on different slicing of training and testing data sets were performed and results were analyzed. Also, different data sets were discretized using both expert guided discretization and uniform discretization. Later some context variable discretization was implemented and observed to a certain extent too. The first experiment was to successfully use database structure and database SQL language in the construction of decision trees. A generic two-class method was used on one of the data sets with heart disease data. The entropy function used was the classic logarithmic function. After some fine-tuning, satisfactory results were obtained. For the full decision tree to be calculated, it took several minutes on the data set with 9 attributes.
In the next sections, all of the new ideas are implemented and the results presented for them. Each of the sections in Chapter III about novel ideas and methodology are experimentally tested and the results are given for each of those in a separate section. Also worth of noting is that the results produced on randomly created training and testing data sets are presented in the tables as the average of several trials on them (until reasonable comfort was achieved about the results presented for the tests and trials).

4.5.1 Interpreting Decision Trees in Multiclass Environment

First, experimental results are given for two class versus multi-class (could be any number of classes higher than two classes) decision trees built using different functions. The results are provided with respect to computational time (speed), complexity (number of nodes), classification rate (accuracy) and other manipulative attributes (pruning efficiency, alternate branching, etc.). It is important to note that all of these outlined experiments are performed in further individual sections after this one in greater depth, but they are also necessary for this section, for accurate comparison between two class versus more than two class decision trees built in the same data sets. Most of the training and testing data sets are designed using a random set generation procedure, but some test and train data was separated and grouped into a permanent manner for better understanding of performance and the differences between observed trees.

Run time-minutes	Training set classification %	Test set classification %	Number of final nodes	Number of classes	Random set creation %	Type of functional
2.7	96	90	52	2	Random 20	logarithmic
2.78	95	95	51	2	Random 20	logarithmic
3.57	97	80	49	2	Random 30	logarithmic
3.32	94	88	53	2	Random 20	parabolic
3.56	93	85	51	3	Fixed	logarithmic
3.33	93	83	50	3	Fixed	sinusoidal
3.81	93	86	57	3	Random 20	sinusoidal
3.36	93	87	50	3	Fixed	triangular

Table 4.2. Two-class vs. multi-class decision tree characteristics

Run time is expressed in minutes. Classification of the training set represents a homogeneity of terminal nodes, namely if values in the nodes are all belonging to a single class or not. The test set classification represents how well the test data records were represented and captured by the rules obtained from the decision tree built by the data in the training set. The number of final nodes represents the number of terminal nodes (leafs) built in the decision tree from the training data set. The random set creation column specifies if training and testing data sets were created using a random procedure or if they are fixed data sets (percentage represents the split of data, for example 20 would mean that 20% of the initial data set was used to create the test data set-used for classification rate testing, and 80% was used for creation of a training data set-used for building of the decision tree). The type of functional is provided as a reference for future sections and it will not be analyzed in the corresponding section in the next chapter, but in its separate section about usage of different functions and their outcomes on the overall decision tree design. There were ten experiments performed for each data set observed.

The reason that experiments with fixed training and testing data sets take less time on average is due to the lowered amount of time that is not needed for a random procedure to create those. Since the trials are compared versus multi class trees in this section, the run times for each of the trials should be compared against corresponding trials in the multi class table(s). The purpose of the fixed data set trials is to provide better insights in the accuracy (classification) of observed decision trees.

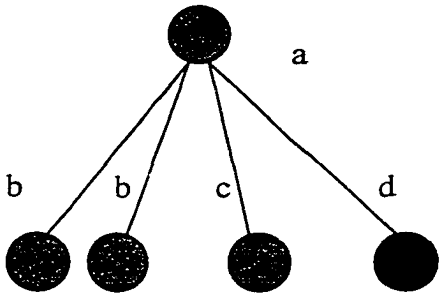


Figure 4.1 Partial upper part of the two-class decision tree-predominant trial structure I

Dark nodes represent terminal nodes (leafs) that have only one class (homogenous class).

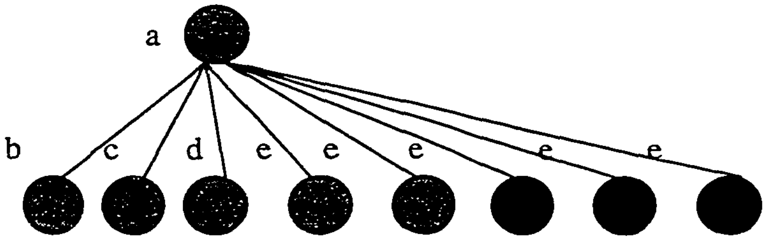


Figure 4.2 Partial upper part of the multi-class decision tree-predominant trial structure I

Attribute Name	Definition
a	CP-chest pain
b	TRESTBPS-resting blood pressure
c	RESTECG-heart rate
d	AGE-age of subject
e	SLOPE-slope of the peak exercise segment
f	SEX-gender
g	THALACH-maximum heart rate achieved

Table 4.3 Attribute definitions for a multi-class structures-upper tree (predominant ones)

Next, results in the tabular form for alternate branching cases of two-class and multi-class decision trees are presented.

Run time-minutes	Training set classification %	Test set classification %	Number of final nodes	Number of classes	Random set creation %
6.33	99	78	125	2	Random 20
4.41	91	85	87	2	Fixed
4.81	98	79	85	2	Fixed
3.87	96	79	69	2	Fixed
13.57	92	88	313	2	Fixed
5.68	94	92	97	3	Random 20
4.95	94	90	94	3	Fixed
5.58	92	86	104	3	Fixed
4.7	93	85	80	3	Fixed
5	92	92	94	3	Fixed

Table 4.4 Two-class vs. multi-class decision tree characteristics-Swiss heart data set alternate branching (two-class and multi-class)

Decision tree nodes most commonly encountered as alternate branch nodes are displayed in the next two figures.

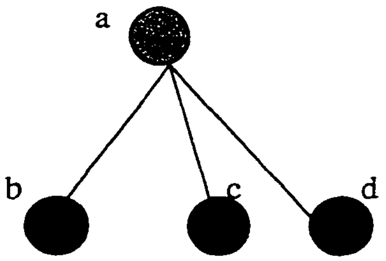


Figure 4.3 Most common alternate branching attributes-Swiss data set (all cases)

Attribute Name	Definition
a	AGE-age of subject
b	TRESTBPS-resting blood pressure
c	RESTECG-heart rate
d	SEX-gender
e	SLOPE-slope of the peak exercise segment

Table 4.5 Attribute definitions for a Swiss data set-alternate branching attributes (the most dominant case)

Attributes shaded dark are alternate branches (children of same information gain) coming from the AGE attribute. Each of them can be chosen as the next most dominant attribute for that particular branch. In the next section results for the indirect usage of decision trees in the case of incompatible databases and combining of those databases on the base of their common and chosen attributes are given.

4.5.2 Indirect Decision Tree Approach for Multi Database Structures

Trials are performed on data sets with some attributes common to each set observed per particular trial and some attributes "indigenous" to individual data sets. Characteristics measured and presented here are about common attributes in the final data set, threshold value for qualifying attributes, average computational time, relevance of attributes provided and a method of obtaining relevance (user-guided, default assigned or partially user-guided). For the first set of trials four data sets were observed **nooldpeaksex, noslopetrestbps, nosexage,** and **nooldpeakage.** As explained earlier they were derived and created based on the Swiss heart data set.

Run time-minutes	Name of the data set	Relevance %	Rule threshold %	Number of attributes	Final attributes
13.4	nosexage	user assigned	45	7	CP, RESTECG
	noslopetrestbps	user assigned		7	
	nooldpeaksex	user assigned		7	
	nooldpeakage	user assigned		7	
14.23	nosexage	user assigned	45	7	CP, RESTECG, THALACH
	noslopetrestbps	user assigned		7	
	nooldpeaksex	user assigned		7	
	nooldpeakage	user assigned		7	
11.01	nosexage	user assigned	50	7	CP, RESTECG, THALACH
	noslopetrestbps	user assigned		7	
	nooldpeaksex	user assigned		7	
	nooldpeakage	user assigned		7	
12.78	nosexage	user assigned	55	7	CP
	noslopetrestbps	user assigned		7	
	nooldpeaksex	user assigned		7	
	nooldpeakage	user assigned		7	

Table 4.6 First set of trials-results for multi database approach, user assigned relevance

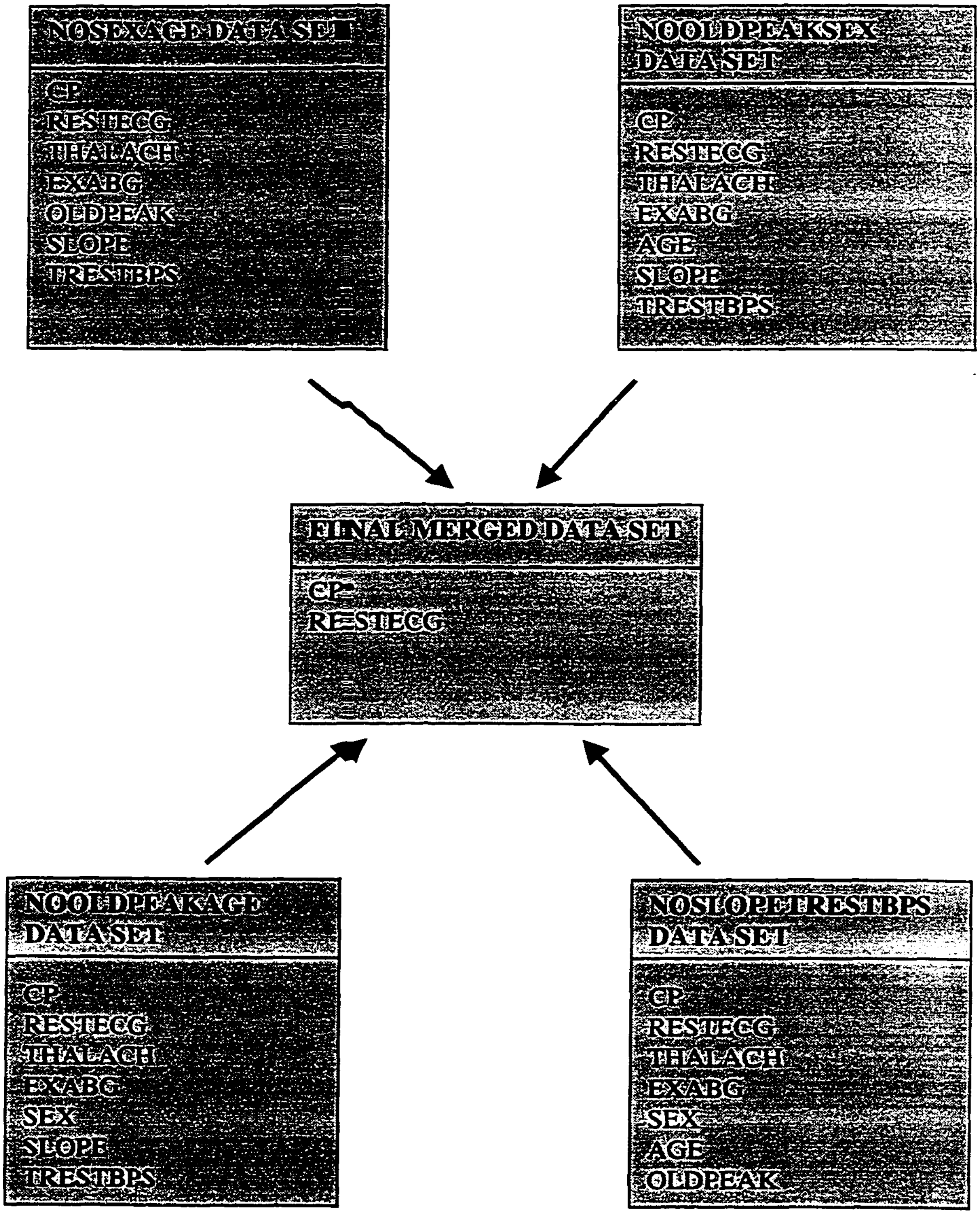


Figure 4.4 Final merged set ffrom the first set of trials-most predominant case

Attributes in the final merged data set were explained earlier in the previous section.

For the second set of trials four data sets were observed **nooldpeaksex, noslopetrestbps, nosexage**, and **nooldpeakage.** Relevance is partially guided by the user for the data sets.

Run time minutes	Name of the data set	Relevance %	Rule threshold %	Number of classes	Number of attributes	Final attributes
12.10	nosexage	partially guided	40	2	7	CP, EXABG, THALACH, RESTECG
	noslopetrestbps	partially guided		2	7	
	nooldpeaksex	partially guided		2	7	
	nooldpeakage	partially guided		2	7	
11.26	nosexage	partially guided	45	2	7	CP, RESTECG
	noslopetrestbps	partially guided		2	7	
	nooldpeaksex	partially guided		2	7	
	nooldpeakage	partially guided		2	7	
11.37	nosexage	partially guided	50	2	7	NO VALUES PASSED
	noslopetrestbps	partially guided		2	7	
	nooldpeaksex	partially guided		2	7	
	nooldpeakage	partially guided		2	7	
11.34	nosexage	partially guided	55	2	7	NO VALUES PASSED
	noslopetrestbps	partially guided		2	7	
	nooldpeaksex	partially guided		2	7	
	nooldpeakage	partially guided		2	7	

Table 4.7 Second set of trials-results for multi database approach, relevance partially guided

In the next figure, the most dominant case of the final merged set for partially guided relevance and four observed data sets is presented. In the case of the final data set obtained at the lower threshold value, the relevance values are altered in the hope of obtaining a potentially different result.

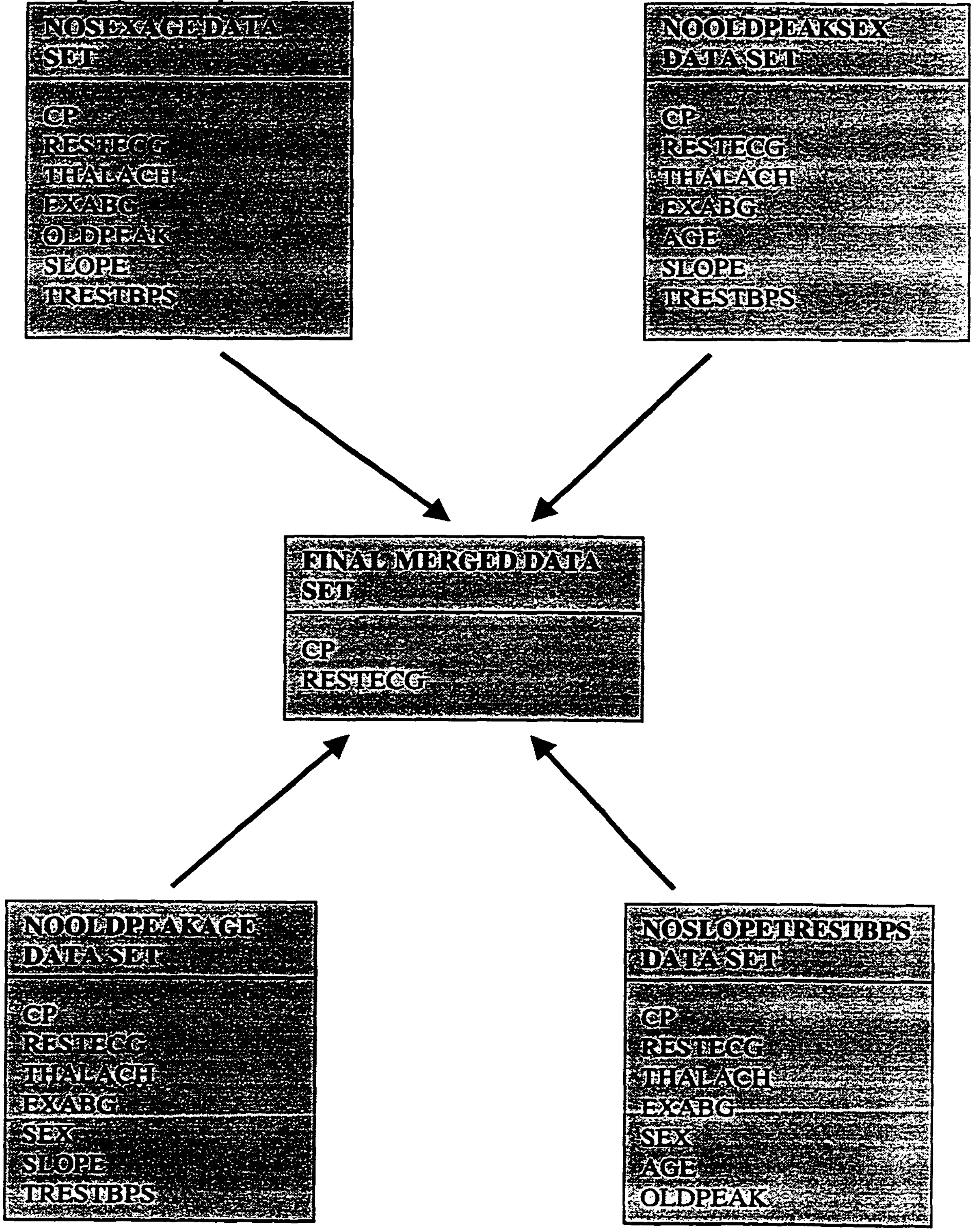


Figure 4.5 Final merged set from the second set of trials-most predominant case

In the next trial, four data sets of different classes and origin are observed as a starting point. These are **swiss12, swiss123, hungdiscphd12** and **hungdiscphd123.**

Run time minutes	Name of the data set	Relevance %	Rule threshold %	Number of classes	Number of attributes	Final attributes
6.25	swiss12	default	50	2	9	CP, OLDPEAK
	hungdiscphd12	default		2	8	
7.33	swiss12	user assigned	50	2	9	CP, AGE, OLDPEAK
	hungdiscphd12	user assigned		2	8	
11.53	swiss123	default	50	3	9	AGE, CP, TRESTBPS, OLDPEAK
	hungdiscphd123	default		3	8	
12.01	swiss123	user assigned	50	3	9	AGE, CP, TRESTBPS, OLDPEAK
	hungdiscphd123	user assigned		3	8	
7.87	swiss12	default	50	2	9	AGE, CP, RESTECG, THALACH
	hungdiscphd123	default		3	8	
8.45	swiss12	user assigned	50	2	9	AGE, CP, RESTECG, THALACH
	hungdiscphd123	user assigned		3	8	
11.14	swiss123	default	50	3	9	AGE, CP, OLDPEAK, TRESTBPS
	hungdiscphd12	default		2	8	
9.88	swiss123	user assigned	50	3	9	AGE, CP, OLDPEAK, TRESTBPS, RESTECG
	hungdiscphd12	user assigned		2	8	

Table 4.8 Third set of trials-results for multi database approach

Figures representing final sets for each of the data set combinations are given next.

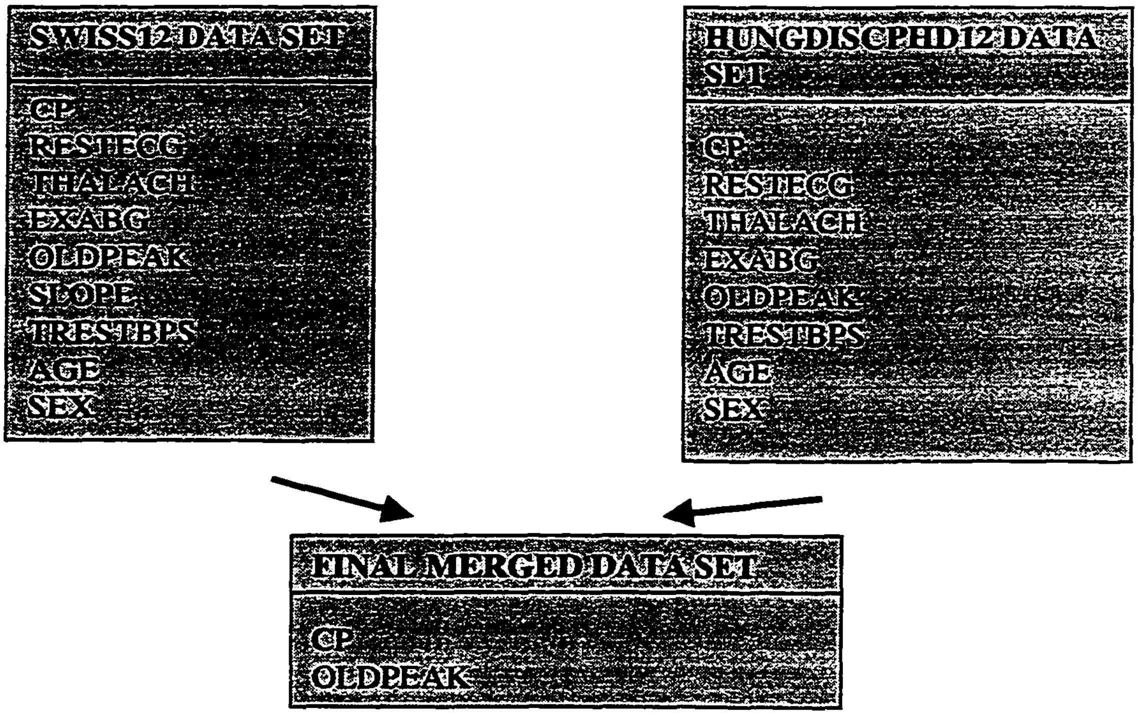


Figure 4.6.a Final merged set from the third set of trials-most predominant case (combination of two two-class data sets)

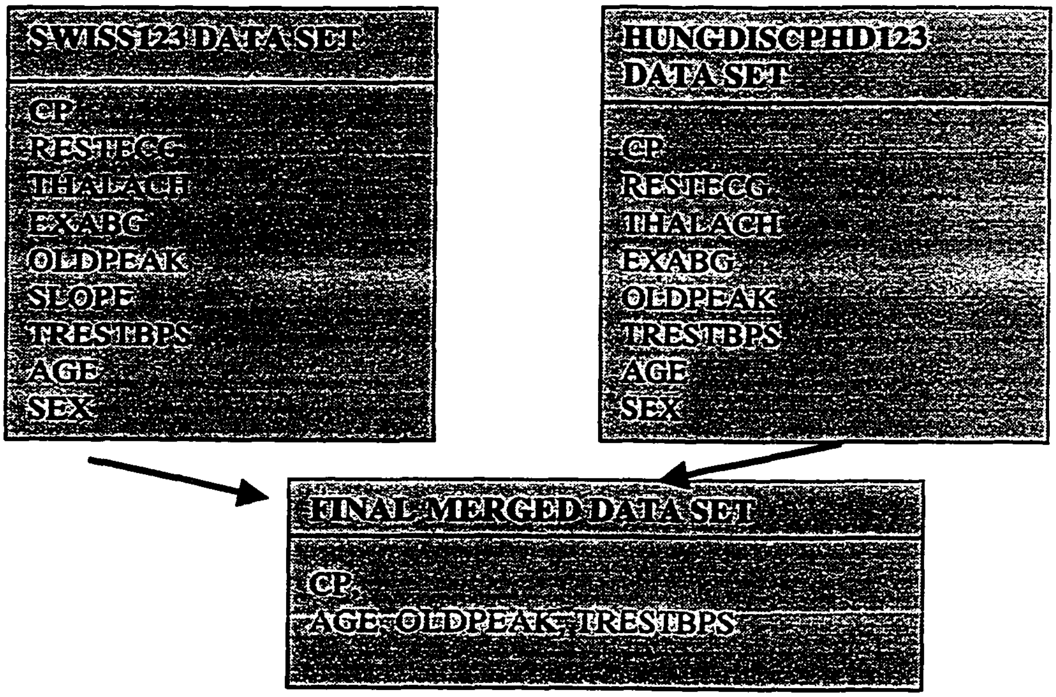


Figure 4.6.b Final merged set from the third set of trials-most predominant case (combination of two three-class data sets)

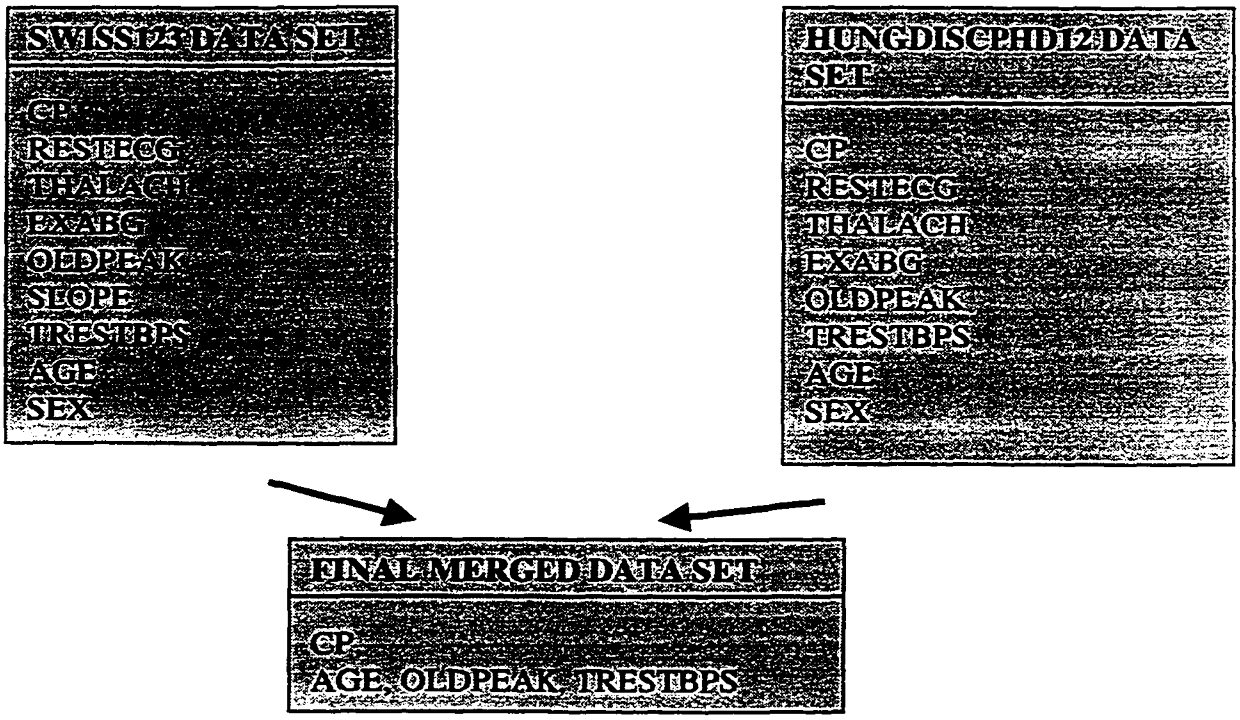


Figure 4.6.c Final merged set from the third set of trials-most predominant case (combination of two-class and three-class data sets)

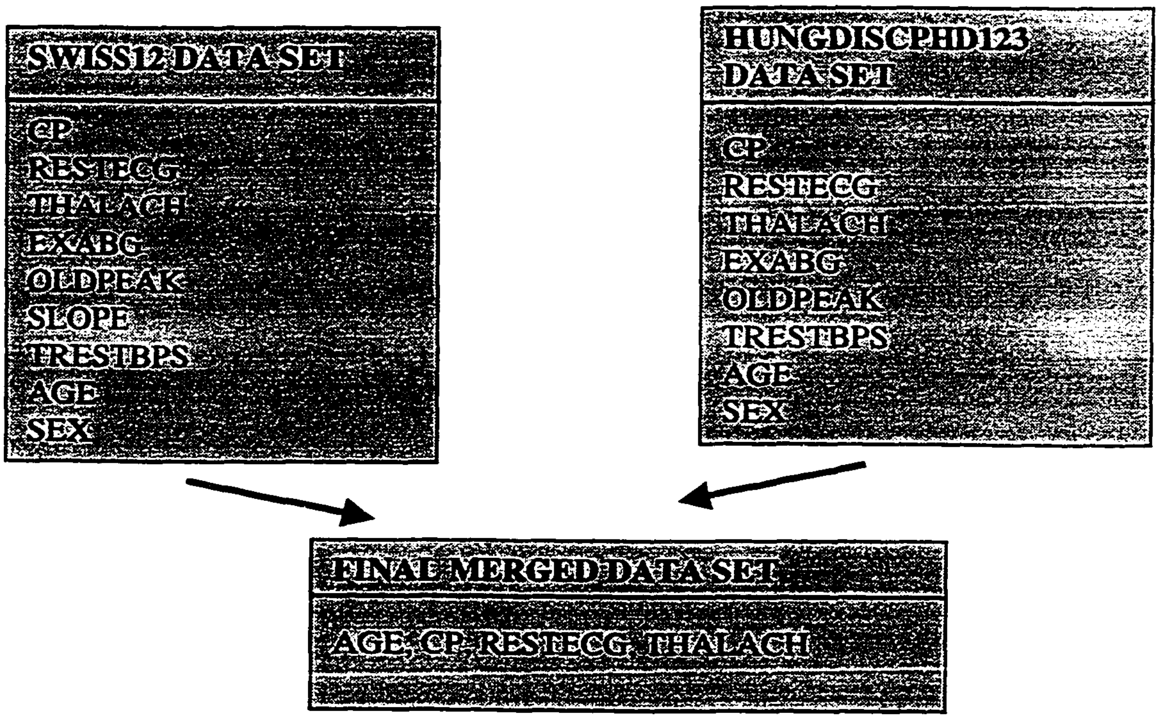


Figure 4.6.d Final merged set from the third set of trials-most predominant case (combination of two-class and three-class data sets)

In this section results for merging of different data sets were presented based on the algorithm used in chapter III. Values of NA and DSS as introduced in section 3.2.2, equation 3.3 were chosen after several trials and they performed well with the particular data sets observed. Since future research in this direction with new data sets could require different experimental guidance or end user input, values chosen here are not presented since they are pertinent only for the particular data set(s).

As expected, the outcome of merging was a final set with fewer attributes in it but each qualifying attribute had strong rules induced from it. In the next chapter, section 5.1 further analyses and discussions are provided.

In the next section experimental results of decision trees built by using different entropy functions that were presented in the previous chapter are given.

4.6 Use of Different Entropy Functional Experiments and Results

Several trials involving different data sets and different entropy functions are presented together with the differences in the decision tree structures as the outcomes.

The first set of trials is performed on the Swiss heart data both for two and three class cases. No extra options were observed in this trial, such as pruning of decision trees or alternate branching.

Run time minutes	Training set classification %	Test set classification %	Number of final nodes	Random set creation %	Number of classes
2.7	96	90	52	Random 20	2
2.78	95	95	51	Random 20	2
2.94	97	95	54	Random 25	2
2.61	95	95	48	Random 25	2
3.57	97	90	49	Random 30	2
3.98	94	89	58	Random 20	3
4.93	95	72	53	Random 20	3
3.3	95	74	48	Random 25	3
3.4	96	72	47	Random 25	3
3.66	93	86	48	Random 30	3

Table 4.9 Swiss data set results for logarithmic functional (professional discretization)

As in the previous section, explanations for tabular results and columns are as follows,

- Run time is expressed in minutes.
- Classification of training set represents a homogeneity of terminal nodes, namely if values in the nodes are all belonging to single class or not.
- Test set classification represents how well the test data records were represented and

captured by the rules obtained from the decision tree built by the data in the training set.

- Number of final nodes represents the number of terminal nodes (leafs) built in the decision tree from the training data set.
- Type of functional: which functional is used for trial cases presented.
- Number of classes in the data set.

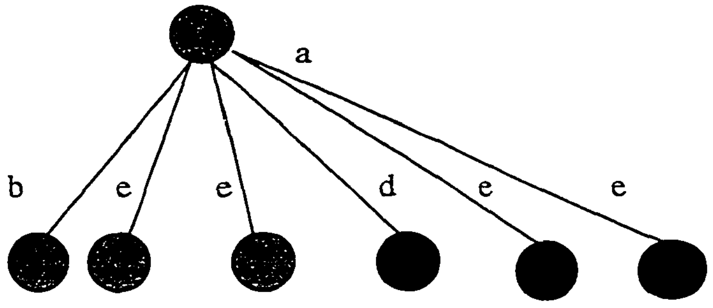


Figure 4.7.a Partial upper part of the two-class decision tree-predominant trial structure

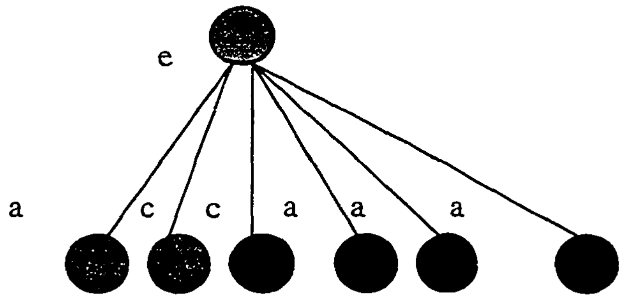


Figure 4.7.b Partial upper part of the three-class decision tree-predominant trial structure

The definitions of nodes and attributes for the heart data set-Switzerland are provided in Table -4.10 below.

For two-class and three class decision trees, predominant decision tree builds were observed.

Attribute Name	Definition
a	AGE-age of subject
b	TRESTBPS-resting blood pressure
c	RESTECG-heart rate
d	OLDPEAK-exercise depression
e	CP-chest pain

Table 4.10 Attribute definitions for a two-class and three-class structures

Next, results are given for the parabolic functional and the same (Swiss) data set.

Run time-minutes	Training set classification %	Test set classification %	Number of final nodes	Number of classes	Random set creation %
3.32	94	88	53	2	Random 20
4.1	95	93	63	2	Random 20
3.05	96	86	52	2	Random 25
3.61	96	86	47	2	Random 25
3.49	96	84	48	2	Random 30
3.8	94	93	63	3	Random 20
3.43	96	95	62	3	Random 20
3.34	91	82	56	3	Random 25
3.17	95	88	53	3	Random 25
3.18	99	83	54	3	Random 30

Table 4.11 Swiss data set results for two-class decision trees and parabolic functional

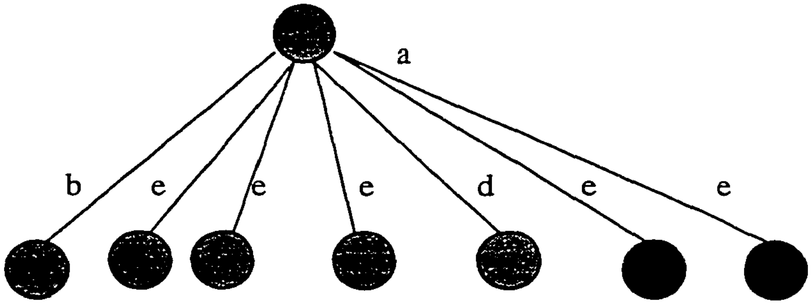


Figure 4.8.a Partial upper part of the two-class decision tree-predominant trial structure

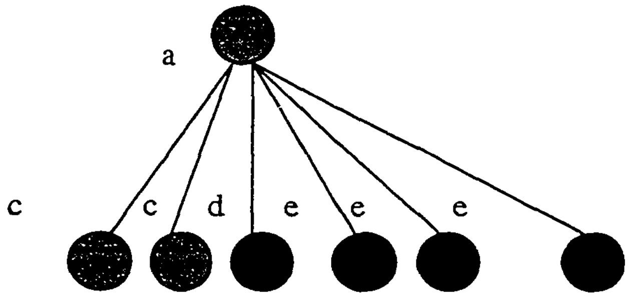


Figure 4.8.b Partial upper part of the three-class decision tree-predominant trial structure

The definitions of nodes and attributes for the heart data set-Switzerland are provided in Table 4.12 below.

For two-class and three class decision trees, predominant decision tree builds were observed.

Attribute Name	Definition
a	TRESTBPS-resting blood pressure
b	RESTECG-heart rate
c	CP-chest pain
d	SLOPE-peak exercise
e	AGE-age of subject

Table 4.12 Attribute definitions for a two-class and three-class structures-upper tree (predominant ones) for professionally discretized data set

Next, results are given for the triangular functional and the same (Swiss) data set.

Run time-minutes	Training set classification %	Test set classification %	Number of final nodes	Number of classes	Random set creation %
3.48	95	100	54	2	Random 20
3.41	99	88	56	2	Random 20
2.43	94	92	35	2	Random 25
2.99	96	82	41	2	Random 25
2.8	96	81	46	2	Random 30
3.62	94	85	55	3	Random 20
3.58	93	94	53	3	Random 20
3.74	95	92	52	3	Random 25
3.4	96	82	47	3	Random 25
3.55	93	82	44	3	Random 30

Table 4.13 Swiss data set results for two-class decision trees and triangular functional

Next, results are given for the sinusoidal functional and the same (Swiss) data set.

Run time minutes	Training set classification %	Test set classification %	Number of final nodes	Number of classes	Random set creation %
3.11	95	80	47	2	Random 20
2.87	95	90	47	2	Random 20
3.29	97	88	47	2	Random 25
3.43	96	89	47	2	Random 25
3.49	96	84	45	2	Random 30
3.63	95	89	56	3	Random 20
3.81	93	86	57	3	Random 20
3.34	93	85	54	3	Random 25
3.22	95	89	55	3	Random 25
3.63	96	78	52	3	Random 30

Table 4.14 Swiss data set results for two-class decision trees and sinusoidal functional

Next, combinations of Swiss and Hungarian data sets where the Swiss data set is used as a training data set and the Hungarian as a test data set were experimented with.

Run time minutes	Training set classification %	Test set classification %	Number of final nodes	Number of classes	Type of functional
2.86	93	80	54	2	logarithmic
3.94	91	82	58	2	parabolic
3.96	92	77	55	2	triangular
3.23	93	80	55	2	sinusoidal
3.86	90	72	63	3	logarithmic
3.91	87	77	67	3	parabolic
3.4	88	75	61	3	triangular
3.92	90	73	66	3	sinusoidal

Table 4.15 Heart data set results for two and three-class decision trees and different functional used-Swiss and Hungarian data sets

This creates for each functional a fixed data set situation where performance of each functional can be compared against data coming from completely different data sources. One run was compiled for each functional, both for two-class and three-class data sets respectively. The next table presents the most dominant attribute in the training decision tree (at the top of the tree).

Most Dominant Attribute (Top of the Tree)	Functional	Number of Classes
CP- Chest pain	Logarithmic	2
TRESPBPS-Resting blood pressure	Parabolic	2
TRESPBPS-Resting blood pressure	Triangular	2
CP- Chest pain	Sinusoidal	2
AGE-age of subject	Logarithmic	3
CP- Chest pain	Parabolic	3
CP- Chest pain	Triangular	3
TRESPBPS-Resting blood pressure	Sinusoidal	3

Table 4.16 Complete Swiss data set-most dominant attribute (node) in the decision tree

Last in this section, results for the RRSP data set and the usage of different functional is given.

Run time-minutes	Training set classification %	Test set classification %	Number of final nodes	Random set creation %	Type of functional
88.76	100	98	991	Random 20	logarithmic
155.49	100	98	2230	Random 20	parabolic
141.37	100	97	1290	Random 20	triangular
199.69	100	96	19990	Random 20	sinusoidal

Table 4.17 RRSP data set results for four-class decision trees (partial professional discretization) and different functional used

Most Dominant Attribute (Top of the Tree)	Functional
PR- Canadian province	Logarithmic
SEX-Gender	Parabolic
INC-Income	Triangular
INC-Income	Sinusoidal

Table 4.18 RRSP data set-most dominant attribute (node) in the decision tree (attribute at the top of the decision tree structure)

Each of the functional performed in the range expected; some had performed better with one data set than the others. Also, some performed better with respect to discretization being used for the data set(s) observed, but none displayed significant deviation. Future discussion and analysis is provided in detail in the next chapter, section 5.2.

4.7 Pruning of Decision Trees

In this section, two methods presented in Chapter III, used to prune decision trees in a database logical manner while retaining the necessary scholastic and research approaches in its implementation are observed and the results from the trials are presented.

4.7.1 Pruning of Decision Trees- Data Set Size Criterion

In this sub-section the first of two presented methods, namely data set size criterion pruning method of decision trees is being analyzed and experimented with.

The first set of trials is performed on the Swiss heart data set both for two-class and three-class cases. The value of the pruning threshold is set at 5% of the initial data set as explained in Chapter III.

Run time-minutes before/after	Training set classification% before/after	Test set classification% before/after	Number of final nodes before/after	Number of classes	Type of functional
2.5/1.92	96/93	75/75	37/35	2	logarithmic
3/2.28	96/94	84/84	44/43	2	parabolic
2.62/2.25	96/94	80/80	39/38	2	triangular
2.59/2.14	96/93	72/72	39/37	2	sinusoidal
3.56/2.47	93/83	85/81	51/44	3	logarithmic
3.25/2.62	93/92	87/88	49/48	3	parabolic
3.36/2.81	93/92	87/88	50/49	3	triangular
3.06/3	95/87	81/84	48/46	3	sinusoidal

Table 4.19 Pruning of decision trees-Swiss data set with 5% threshold value

As in the previous sections, explanations for tabular results and columns are as following,

- Run time is expressed in minutes before/after the pruning of decision tree.
- Classification of training set represents a homogeneity of terminal nodes, namely if values in the nodes are all belonging to single class or not. Before/after the pruning.
- Test set classification represents how well the test data records were represented and captured by the rules obtained from the decision tree built by the data in the training set before/after the pruning.
- Number of final nodes represents the number of terminal nodes (leafs) built in the decision tree from the training data set before/after the pruning.
- Type of functional-which functional is used for trial cases presented.

- Number of classes in the data set.

The second set of trials is performed on the Swiss heart data set both for two-class and three-class cases. Value of pruning threshold is set at 10% of initial data set.

Run time-minutes before/after	Training set classificatio n % before/after	Test set classificatio n % before/after	Number of final nodes before/after	Number of classes	Type of functional
2.5/1.65	96/85	75/75	37/30	2	logarithmic
3/2.28	96/85	84/77	44/32	2	parabolic
2.62/1.75	96/83	80/74	39/27	2	triangular
2.59/1.76	96/86	72/72	39/32	2	sinusoidal
3.56/2.14	93/76	85/81	51/36	3	logarithmic
3.25/2.35	93/79	87/89	49/37	3	parabolic
3.36/2.3	93/76	87/88	50/37	3	triangular
3.06/2.23	95/77	81/85	43/37	3	sinusoidal

Table 4.20 Pruning of decision trees-Swiss data set with 10% threshold value

The third set of trials is performed on the Swiss heart data set as a training data set and the Hungarian data set as a testing data set both for two-class and three-class cases. The value of the pruning threshold is set at 10% of the initial data set.

Run time-minutes before/after	Training set classification % before/after	Test set classificatio n % before/after	Number of final nodes before/after	Number of classes	Type of functional
2.86/1.96	93/81	80/80	54/37	2	logarithmic
3.94/2.15	91/82	82/76	58/39	2	parabolic
3.96/2.01	92/78	77/74	55/34	2	triangular
3.23/1.93	93/79	80/79	55/36	2	sinusoidal
3.86/2.24	90/71	72/68	63/36	3	logarithmic
3.91/2.66	87/75	77/75	67/45	3	parabolic
3.4/2.41	88/74	75/74	61/41	3	triangular
3.92/2.34	90/72	73/73	66/41	3	sinusoidal

Table 4.21 Pruning of decision trees-Swiss/Hungarian data set with 10% threshold value

The fourth set of trials is performed on the Liver data set fuzzy discretized for both two-class and three-class cases. The value of the pruning threshold is set at 10% of the initial data set.

Run time-minutes before/after	Training set classification % before/after	Test set classification % before/after	Number of final nodes before/after	Number of classes	Type of functional
1.69/1.15	79/77	78/78	39/28	2	logarithmic
1.75/1.18	79/77	75/74	31/23	2	parabolic
1.33/1.21	78/76	79/78	29/25	2	triangular
1.68/1.12	79/77	78/78	39/28	2	sinusoidal
2/1.55	67/65	66/66	41/30	3	logarithmic
1.74/0.8	58/54	64/63	27/17	3	parabolic
1.48/0.75	58/54	64/63	26/17	3	triangular
1.61/0.91	64/60	62/63	32/20	3	sinusoidal

Table 4.22 Pruning of decision trees-Liver data set with 10% threshold value

The fifth set of trials is performed on the RRSP data set for a four-class case. The value of the pruning threshold is set at 2% of the initial data set.

Run time-minutes before/after	Training set classification in % before/after	Test set classification % before/after	Number of final nodes before/after	Number of classes	Type of functional
88.76/16.44	100/95	98/96	991/163	4	logarithmic
155.49/26.23	100/95	98/95	2230/296	4	parabolic
141.37/17.1	100/94	97/95	1290/163	4	triangular
199.69/23.43	100/95	96/96	1990/224	4	sinusoidal

Table 4.23 Pruning of decision trees-RRSP data set with 2% threshold value

Trials were also performed for threshold values of 5, 10 and 17 % but due to the over-pruning of decision trees, the results were not presented, since final trees were of two layer depth creating over-classification (over-optimistic classification) of the test data set. Final decision tree structures were of 5th layer depth after pruning.

4.7.2 Pruning of Decision Trees- One Layer Method

In this sub-section, the second of two presented methods, namely one layer criterion pruning method of decision trees is being analyzed and experimented with.
The first set of trials is performed on the Swiss heart data set both for two-class and three-class cases. The value of the pruning threshold is set at 5% of the initial data set as explained in Chapter III.

Run time-minutes before/after	Training set classification % before/after	Test set classification % before/after	Number of final nodes before/after	Number of classes	Type of functional
2.5/2.36	96/96	75/75	37/35	2	logarithmic
3/2.69	96/96	84/84	44/42	2	parabolic
2.62/2.3	96/96	80/80	39/38	2	triangular
2.59/2.37	96/96	72/72	39/37	2	sinusoidal
3.56/3.92	93/93	85/83	51/48	3	logarithmic
3.25/3.58	93/92	87/88	49/44	3	parabolic
3.36/3.64	93/93	87/87	50/46	3	triangular
3.06/3.79	95/93	81/83	48/46	3	sinusoidal

Table 4.24 Pruning of decision trees-Swiss data set with 5% threshold value

As in the previous sections, explanations for tabular results and columns are as following,

- Run time is expressed in minutes before/after the pruning of decision tree.
- Classification of training set represents a homogeneity of terminal nodes, namely if values in the nodes are all belonging to a single class or not. Before/after the pruning.
- Test set classification represents how well the test data records were represented and captured by the rules obtained from the decision tree built by the data in the training set. Before/after the pruning.
- Number of final nodes represents the number of terminal nodes (leafs) built in the decision tree from the training data set. Before/after the pruning.
- Type of functional-which functional is used for trial cases presented.
- Number of classes in the data set.

The second set of trials is performed on the Swiss heart data set both for two-class and three-class cases. The value of the pruning threshold is set at 10% of the initial data set.

Run time minutes before/after	Training set classificati on % before/after	Test set classificati on % before/after	Number of final nodes before/after	Numbe r of classes	Type of functional
2.5/2.19	96/95	75/75	37/31	2	logarithmic
3/2.63	96/95	84/84	44/36	2	parabolic
2.62/2.18	96/95	80/80	39/29	2	triangular
2.59/2.23	96/95	72/72	39/31	2	sinusoidal
3.56/3.55	93/91	85/88	51/42	3	logarithmic
3.25/3.29	93/92	87/88	49/41	3	parabolic
3.36/3.51	93/91	87/87	50/40	3	triangular
3.06/3.65	95/92	81/83	43/42	3	sinusoidal

Table 4.25 Pruning of decision trees-Swiss data set with 10% threshold value

The third set of trials is performed on the Swiss heart data set as a training data set and the Hungarian data set as a testing data set both for two-class and three-class cases. The value of the pruning threshold is set at 10% of the initial data set.

Run time minutes before/after	Training set classification % before/after	Test set classificatio n % before/after	Number of final nodes before/after	Number of classes	Type of functional
2.86/2.77	93/92	80/74	54/41	2	logarithmic
3.94/3	91/92	82/78	58/41	2	parabolic
3.96/2.89	92/93	77/75	55/40	2	triangular
3.23/2.77	93/94	80/74	55/38	2	sinusoidal
3.86/2.64	90/90	72/72	63/46	3	logarithmic
3.91/2.98	87/85	77/69	67/51	3	parabolic
3.4/2.86	88/86	75/66	61/47	3	triangular
3.92/2.34	90/90	73/74	66/52	3	sinusoidal

Table 4.26 Pruning of decision trees-Swiss/Hungarian data set with 10% threshold value

The fourth set of trials is performed on the Liver data set fuzzy discretized for both two-class and three-class cases. The value of the pruning threshold is set at 10% of the initial data set.

Run time-minutes before/after	Training set classification %before/after	Test set classification %before/after	Number of final nodes before/after	Number of classes	Type of functional
1.69/1.02	79/78	78/81	39/8	2	logarithmic
1.75/0.97	79/78	75/76	31/7	2	parabolic
1.33/0.82	78/77	79/81	29/7	2	triangular
1.68/0.91	79/78	78/81	39/8	2	sinusoidal
2/1.01	67/62	66/66	41/9	3	logarithmic
1.74/0.43	58/51	64/61	27/4	3	parabolic
1.48/0.49	58/51	64/61	26/4	3	triangular
1.61/0.6	64/58	62/60	32/4	3	sinusoidal

Table 4.27 Pruning of decision trees-Liver data set with 10% threshold value

Trials were performed for higher values, but there was over-classification and over pruning of the decision trees as the outcome of a high threshold value, so results were not included. There is over-simplification already noticeable in the table above.

The fifth set of trials is performed on the RRSP data set for the four-class case. The value of the pruning threshold is set at 2%, 5% and 10% for the one layer method.

Run time-minutes before/after	Training set classification % before/after (and pruning threshold %)	Test set classification % before/after	Layers before/after	Number of final nodes before/after	Number of classes	Type of functional
88.76/83.52	100/100 (2)	98/98	12/11	991/958	4	logarithmic
155.49/135.38	100/100 (5)	98/97	12/12	2230/1812	4	parabolic
141.37/84.55	100/99 (10)	97/96	12/12	1290/647	4	triangular
199.69/13.12	100/99 (15)	96/98	12/10	1990/87	4	sinusoidal

Table 4.28 Pruning of decision trees-RRSP data set with 2%, 5%, 10% and 15% threshold value for one layer method

Trials were also performed for threshold values of 20 % and 25 % but due to the over-pruning of decision trees the results were not presented, since final trees were creating over-classification (over-optimistic classification) of the test data set.
Given experiments proved that both of the methods chosen to perform pruning performed well, with the expected oversimplification in the cases of a high chosen pruning threshold. Computational efficiency gain was significant and though as mentioned, a necessary tradeoff between classification and efficiency occurred, there was a healthy medium were for certain values of threshold chosen and overall gain was very positive. More discussion was provided in the next chapter, section 5.3.

In the next section, alternate branching experiment and trial results are presented in tabular form with the appropriate figures and descriptions.

4.8 Alternate Branching of Decision Trees

In this section results and trials performed on alternate branching of decision trees as explained and presented in the Chapter III are given.

The first set of trials is performed on the Swiss heart data set (professional discretization) both for two-class and three-class cases.

Run time minutes before/after	Number of final nodes before/after	First layer with alternate branch	Number of classes	Type of functional
2.5/4.51	37/82	4	2	logarithmic
3/8.67	44/191	3	2	parabolic
2.62/19.72	39/391	3	2	triangular
2.59/4.99	39/68	4	2	sinusoidal
3.56/8.74	51/117	3	3	logarithmic
3.25/8.12	49/142	3	3	parabolic
3.36/10.66	50/171	3	3	triangular
3.06/7.07	43/101	3	3	sinusoidal

Table 4.29 Alternate branching of decision trees-Swiss data set

As in the previous sections explanations for tabular results and columns are as following,

- Run time is expressed in minutes before/after the alternate branching of decision tree.
- Number of final nodes represents the number of terminal nodes (leafs) built in the decision tree from the training data set. Before/after the alternate branching.
- The first layer in the overall decision tree structure where the first occurrence of

alternate branches happens.

- Type of functional-which functional is used for trial cases presented.
- Number of classes in the data set.

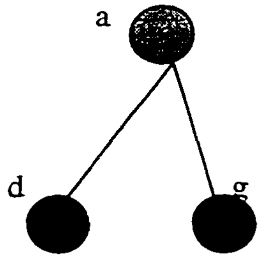
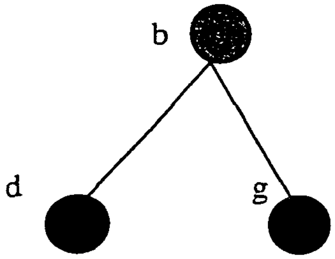


Figure 4.9.a Most common alternate branching attributes-Swiss data set two-class (logarithmic)



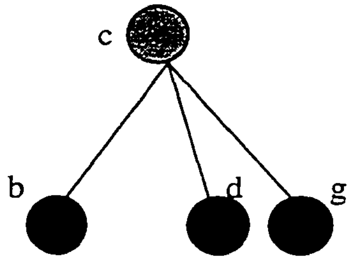


Figure 4.9.b Most common alternate branching attributes-Swiss data set two-class (parabolic)

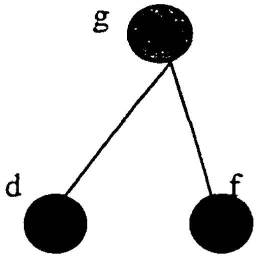


Figure 4.9.c Most common alternate branching attributes-Swiss data set three-class (logarithmic)

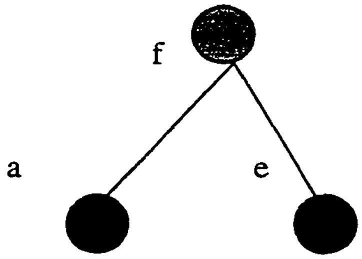


Figure 4.9.d Most common alternate branching attributes-Swiss data set three-class (parabolic)

Attribute Name	Definition
a	TRESTBPS-resting blood pressure
b	RESTECG-heart rate
c	CP-chest pain
d	OLDPEAK-exercise peak
e	AGE-age of subject
f	EXABG-exercise induced angina
g	SLOPE-slope of exercise peak
h	AGE-age of subject
i	THALACH-maximum heart rate achieved

Table 4.30 Attribute definitions for alternate branches of decision tree structures-first layer of alternate branching (predominant ones) for Swiss data sets

Attributes shaded dark are the alternate branches (children of same information gain) coming from the top attribute. Each of them can be chosen as the next most dominant attribute for that particular branch.

The second set of trials is performed on the Liver data set fuzzy discretized for both two-class and three-class cases.

Run time-minutes before/after	Number of final nodes before/after	First layer with alternate branch	Number of classes	Type of functional
1.69/2.73	39/48	3	2	logarithmic
1.75/3.21	31/56	3	2	parabolic
1.33/5.69	29/104	3	2	triangular
1.68/2.66	39/48	3	2	sinusoidal
2/3.24	41/60	4	3	logarithmic
1.74/2.61	27/48	3	3	parabolic
1.48/1.99	26/37	5	3	triangular
1.61/2.43	32/46	4	3	sinusoidal

Table 4.31 Alternate branching of decision trees-Liver data set two-class and three-class

The third set of trials is performed on the RRSP data set for the four-class case.

Run time-minutes before/after	Number of final nodes before/after	First layer with alternate branch	Number of classes	Type of functional [illegible]
88.76/456.23	991/5011	3	4	logarithmic
155.49/489.11	2230/5872	3	4	parabolic
141.37/573.58	1290/7859	3	4	triangular
199.69/412.76	1990/4814	3	4	sinusoidal

Table 4.32 Alternate branching of decision trees-RRSP data set

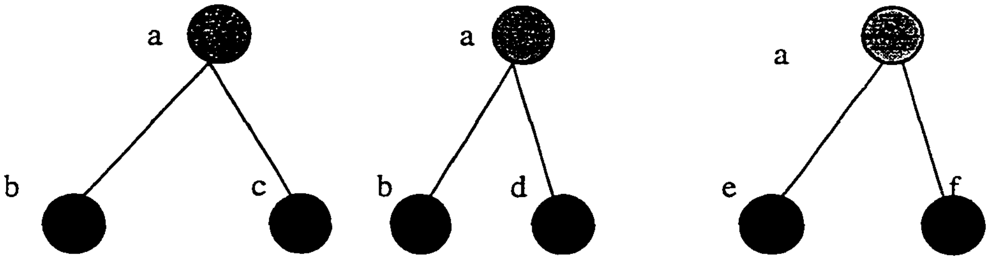


Figure 4.10 Most common alternate branching attributes-RRSP data set (all cases)

(Attribute Name)	Definition
a	PR-Canadian province
b	COMM-community type
c	OCC-occupation
d	FA-family type
e	TEN-house ownership type
f	CAR-car ownership type

Table 4.33 Attribute definitions for alternate branches of decision tree structures-first layer of alternate branching (predominant ones) for RRSP data sets

Certain entropy functions and differently discretized data sets have created more alternate terminal nodes and alternate sub-branches than the others. Future research could determine full benefits from identifying all of the alternate and possible rules induced from the same sub-branch from a particular decision tree. Detailed discussion and suggestions are provided in the next chapter, section 5.4.

In the next section, a missing attributes routine was tested and the results of decision trees built from the data sets with missing values replaced by this algorithm are presented.

4.9 Missing Attributes Experiment

In this section the missing attributes routine presented in Chapter III was tested and the results of the decision trees built from the data sets with missing values replaced by this algorithm are presented. Trials are performed on the data sets and the results of decision trees are presented together with the results from the original data sets before missing values in attributes were introduced and replaced. The first set of trials is performed on the Swiss data set-professionally discretized (two-class and three-class cases) with 80 individual values replaced from the randomly chosen attributes.

Run time-minutes before/after	Training set classificatio n % before/after	Test set classificatio n % before/after	Number of final nodes before/after	Number of classes	Type of functional
2.5/2.42	96/96	75/74	37/36	2	logarithmic
3/3.12	96/96	84/84	41/41	2	parabolic
2.62/2.58	96/97	80/79	39/39	2	triangular
2.59/2.66	96/95	72/72	39/38	2	sinusoidal
3.56/3.66	93/92	85/85	51/50	3	logarithmic
3.25/3.31	93/93	87/87	49/48	3	parabolic
3.36/3.40	93/94	87/88	50/50	3	triangular
3.06/2.98	95/95	81/80	41/41	3	sinusoidal

Table 4.34 Comparisons for Swiss data set results before and after the missing values were introduced and replaced by the missing value algorithm

As in the previous sections explanations for tabular results and columns are as following,

- Run time is expressed in minutes before/after the replacement of missing values in the data set.
- Classification of training set represents a homogeneity of terminal nodes, namely if values in the nodes are all belonging to single class or not.
- Test set classification represents how well the test data records were represented and captured by the rules obtained from the decision tree built by the data in the training set.
- Number of final nodes represents the number of terminal nodes (leafs) built in the decision tree from the training data set. Before/after the missing values routine was used.
- Type of functional-which functional is used for trial cases presented.
- Number of classes in the data set.

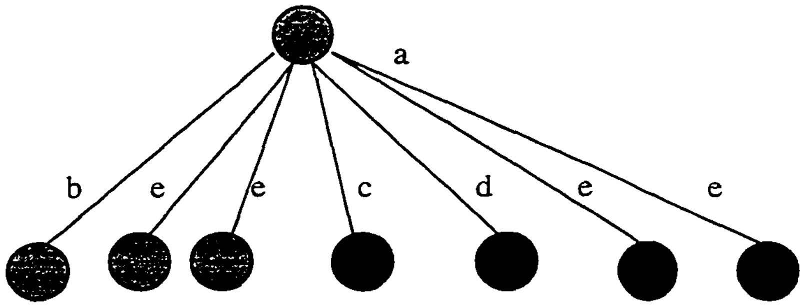


Figure 4.11.a Partial upper part of the Swiss data set two-class decision tree-predominant trial structure before missing value trial

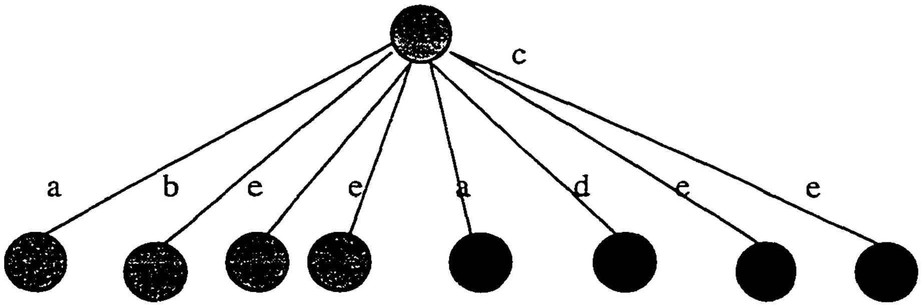


Figure 4.11.b Partial upper part of the Swiss data set two-class decision tree-predominant trial structure after missing value trial

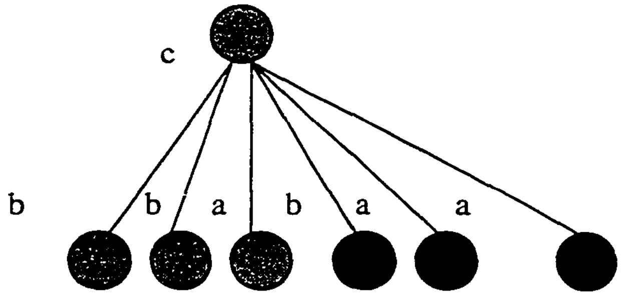


Figure 4.12.a Partial upper part of the Swiss data set three-class decision tree-predominant trial structure before missing value trial

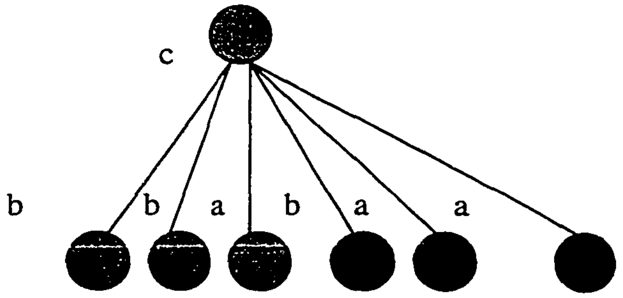


Figure 4.12.b Partial upper part of the Swiss data set three-class decision tree-predominant trial structure after missing value trial.

The definitions of nodes and attributes for the heart data set-Switzerland are provided in the Table 4.35 below.

For two-class and three class decision trees, predominant decision tree builds were observed.

Most Dominant Attribute (Top of the Tree)	Functional	Before/after missing value	Number of Classes
CP- Chest pain	Logarithmic	Before	2
TRESPBPS-Resting blood pressure	Parabolic	Before	2
TRESPBPS-Resting blood pressure	Triangular	Before	2
CP- Chest pain	Sinusoidal	Before	2
AGE-age of subject	Logarithmic	Before	3
CP- Chest pain	Parabolic	Before	3
CP- Chest pain	Triangular	Before	3
TRESPBPS-Resting blood pressure	Sinusoidal	Before	3
CP- Chest pain	Logarithmic	After	2
TRESPBPS-Resting blood pressure	Parabolic	After	2
CP-Chest pain	Triangular	After	2
CP- Chest pain	Sinusoidal	After	2
AGE-age of subject	Logarithmic	After	3
CP- Chest pain	Parabolic	After	3
CP- Chest pain	Triangular	After	3
TRESPBPS-Resting blood pressure	Sinusoidal	After	3

Table 4.35 Complete Swiss data set-most dominant attribute (node) in the decision tree

In the table above, the attribute at the top of the tree is presented for each functional and number of classes used.

Attribute Name	Definition
a	TRESTBPS-resting blood pressure
b	RESTECG-heart rate
c	CP-chest pain
d	OLDPEAK-exercise peak
e	AGE-age of subject

Table 4.36 Attribute definitions for a two-class and three-class structures-upper tree (predominant ones) for professionally discretized data set before and after the missing value routine

The second set of trials is performed on the Liver data set-partially fuzzy discretized (two-class and three-class cases) with 20 individual values replaced from the randomly chosen attributes for the two-class and 30 values replaced for the three-class data sets.

Run time minutes before/after	Training set classificatio n % before/after	Test set classification % before/after	Number of final nodes before/after	Number of classes	Type of functional
1.69/1.89	79/80	78/78	41/41	2	logarithmic
1.75/1.47	79/75	74/75	32/32	2	parabolic
1.33/1.56	78/79	79/77	32/33	2	triangular
1.68/1.61	79/79	78/78	43/43	2	sinusoidal
2/1.86	67/69	66/65	50/50	3	logarithmic
1.2/1.4	58/58	64/60	28/28	3	parabolic
1.48/1.2	58/59	64/65	26/25	3	triangular
1.61/1.26	64/65	62/62	33/34	3	sinusoidal

Table 4.37 Comparisons for Liver data set results before and after the missing values were introduced and replaced by the missing value algorithm

The different amount of missing values chosen for two-class vs. three-class Liver data sets is to provide more diverse and comparable results with respect to the amount of missing data in the data sets that can be introduced without altering the structures of the data set beyond its inherent structure.

The third trial is performed on the RRSP data set-partially professionally discretized (four-class case) with 350 individual values replaced from the randomly chosen attributes with the logarithmic and parabolic functional and the fixed training and testing data sets.

Run time minutes before/after	Training set classificatio n % before/after	Test set classificatio n % before/after	Number of final nodes before/after	Number of classes	Type of functional
100.41/105. 33	100/100	98/98	989/1033	4	logarithmic
178.03/158. 63	100/100	98/98	2330/2298	4	parabolic

Table 4.38 Comparisons for RRSP data set results before and after the missing values were introduced and replaced by the missing value algorithm

Data sets with missing values replaced, performed without significant bias or deviation when compared with the initial data sets without missing values. The testing was to justify the performance after the missing values were replaced using the algorithm developed in chapter III. By using the algorithms and software developed in this thesis, decision trees were compared before/after the missing value routine was employed as presented in the tables given in this section. More analyses are provided in the next chapter, section 5.5.

In next section the idea of creating a class identifier that is a composite of multiple attributes in the data set and back tracking the original attributes and their values is presented.

4.10 Multi Attribute Class Identifier

The ideas from Chapter III about the multi attribute class identifier were implemented and the trial results are presented in this section.

Since the results and performance of the decision trees has been presented up to this point in great detail, only several trials are performed with statistics given.

The first set of trials is performed on the Swiss data set with the explanation of attribute and class creation in Table 4.40.

Run time-minutes	Training set classification %	Test set classification %	Number of final nodes	Number of classes	Type of functional
6.6	91	62	64	15	logarithmic
4.93	85	76	63	15	parabolic
4.54	78	61	52	15	triangular
4.59	85	83	54	15	sinusoidal

Table 4.39 Comparisons for Swiss data set results multi attribute class case

As in the previous sections explanations for tabular results and columns are as following,

- Run time is expressed in minutes.
- Classification of training set represents a homogeneity of terminal nodes, namely if values in the nodes are all belonging to single class or not.
- Test set classification represents how well the test data records were represented and captured by the rules obtained from the decision tree built by the data in the training set.
- Number of final nodes represents the number of terminal nodes (leafs) built in the decision tree from the training data set.
- Type of functional-which functional is used for trial cases presented.

- Number of classes in the data set.

Attribute Name	Attribute Name and Definition
Part of new class	CP-chest pain
Part of new class	AGE-age of subject
Changed to normal attribute as a part of the data set	HEALTH-identifies patient's health

Table 4.40 Multi attribute class definitions and change process explanation-Swiss data set

The data set used as the original source had three classes previously: healthy, sick and needs more diagnosing. The new class identifier consists of combinations of values for AGE and CP attributes. The first few homogenous nodes in the decision tree created with the multi attribute approach are given in Table 4.41 with relevant explanations and comparison to the original data set.

Rule created	Class	Identical to attribute combination
HEALTH=3 and SLOPE=3	14	CP=4 and AGE=4
HEALTH=1 and SLOPE=3 and OLDPEAK=1	11	CP=4 and AGE=1
HEALTH=2 and TRESTBPS=1 and RESTECG=0	11	CP=4 and AGE=1

Table 4.41 Multi attribute class definitions and change process explanation-Swiss data set rule interpretation and class comparison

The second set of trials is performed on the Liver data with the explanation of attribute and class creation in Table 4.43.

Run time-minutes	Training set classification %	Test set classification in %	Number of final nodes	Number of classes	Type of functional
1.15	36	47	20	9	logarithmic
0.95	35	35	15	9	parabolic
0.99	31	36	17	9	triangular
0.91	36	34	15	9	sinusoidal

Table 4.42 Comparisons for Liver data set results-multi attribute class case

Attribute Name	Attribute Name and Definition
Part of new class	DRINKS-number of drinks
Part of new class	ALKPHOS-liver enzymes level
Changed to normal attribute as a part of the data set	HIEALTH-identifies patient's health

Table 4.43 Multi attribute class definitions and change process explanation-Liver data set

The data set used as the original source had two classes previously: healthy and sick. The new class identifier consists of combinations of values for DRINKS and ALKPHOS attributes.

It is obvious due to the low number of attributes, relatively low number of records and high class differentiation that the decision trees created are not performing that well. More discussion will be provided in the next chapter. The results and trials performed on this data set were given due to the completeness of this section.

The first few homogenous nodes in the decision tree created with the multi attribute approach are given in Table 4.44 with relevant explanations and comparisons to the original data set.

Rule created	Class	Identical to attribute combination
MCV=1 and GAMMAGT=1	1	DRINKS=1 and ALKPHOS=1
MCV=1 and GAMMAGT=2	1	DRINKS=1 and ALKPHOS=1
MCV=2 and SGPT=1	2	DRINKS=1 and ALKPHOS=2

Table 4.44 Multi attribute class definitions and change process explanation-Liver data set rule interpretation and class comparison

The third set of trials is performed on the RRSP data with the explanation of attribute and class creation in Table 4.46.

Run time-minutes	Training set classification %	Test set classification %	Number of final nodes	Number of classes	Type of functional
518.34	99	80	24556	15	logarithmic
580.4	99	81	26500	15	sinusoidal

Table 4.45 Comparisons for RRSP data set results multi attribute class case

Attribute Name	Attribute Name and Definition
Part of new class	EDU-type of education
Part of new class	TEN-type of property ownership
Changed to normal attribute as a part of the data set	DV-identifies dwelling value of house

Table 4.46 Multi attribute class definitions and change process explanation for second experiment (bottom line in the Table 4.45)-RRSP data set

The data set used as the original source had four classes previously relating to the dwelling value of the property for RRSP buyers. The new class identifier consists of combinations of values for FA, INC and CAR attributes.

The first few homogenous nodes in the decision tree created with the multi attribute approach are given in Table 4.47 with relevant explanations and comparison to the original data set.

Rule created	Class	Identical to attribute combination
DBI=1 and OCC=5 and PR=11	22	FA=2 and INC=5
DBI=3 and OCC=1 and PR=4	22	FA=2 and INC=5
DBI=3 and OCC=1 and PR=8	12	FA=2 and INC=3

Table 4.47 Multi attribute class definitions and change process explanation-RRSP data set rule interpretation and class comparison

As seen from the tables above, computational efficiency of newly formed data sets with multiattribute classes has decreased significantly as expected. The advantage of this approach is what can be labeled as a multi data mining approach where class can consist of multiple identifiers and if necessary, be broken down into generic data-mining search patterns. More result analyses is provided in the next chapter, section 5.6, and future use and research on this particular idea is given in chapter VI.

In the next section research experiments were performed on the proximity and simplification of tree structures using database tools and advantages in the build of decision trees.

4.11 Terminal Nodes Likelihood-Proximity of Decision Tree Subsets

Experiments were performed mostly on the Swiss data set, with the objective of trying to merge some of the terminal nodes based on their likelihood. This in return would lower the complexity of decision trees and create different rule generation patterns in the successful cases. The first set of trials is performed on the Swiss data set, both two-class and three-class with a different type of functional.

Number of final nodes before/after	Individual threshold per node	Threshold for merged nodes	Number of classes	Type of functional
59/55	65%	70%	2	logarithmic
70/60	65%	70%	2	parabolic
65/58	65%	70%	2	triangular
61/53[1]	65%	70%	2	sinusoidal
67/62	65%	70%	3	logarithmic
77/66	65%	70%	3	parabolic
68/63	65%	70%	3	triangular
71/67	65%	70%	3	sinusoidal

Table 4.48 Swiss data set-comparison before/after merging of proximate terminal nodes

Rule created	Class	Final Attribute
AGE=4 and TRESTBPS=3 and RESTECG=1 and (THALACH=1 or THALACH=2)	2	THALACH
AGE=3 and TRESTBPS=2 and THALACH=3 and (OLDPEAK=0 or OLDPEAK=1)	3	OLDPEAK
AGE=5 and (CP=2 or CP=3)	1	CP

Table 4.49 Some values for merged nodes and resulting rules

The final attribute at the terminal node represents the attribute that was present at one of the qualifying nodes (nodes at the same level of the tree with the same predominant class and different attribute values).

[1] For the sinusoidal functional two-class case, the possibility of merging over two layers occurred. Hyper-node created with hyper rule was of the nature presented in Table 4.50

Rule created	Class	Final Attribute
CP=4 and TRESTBPS=1 and (SLOPE=1 and AGE=4 or (SLOPE=2 and (AGE=1 or AGE=2)))	1	AGE

Table 4.50 Hyper-node (two-layer) rule generation and characteristics

In the following chapters, it will be discussed if this is worthy of future research in the direction of whether the computational difficulties and extra computational time necessary to fully automate such a merging process would be outweighed by saved storage (complexity of decision tree and structure in the node) and rule testing speed (computer efficiency for testing the data set).

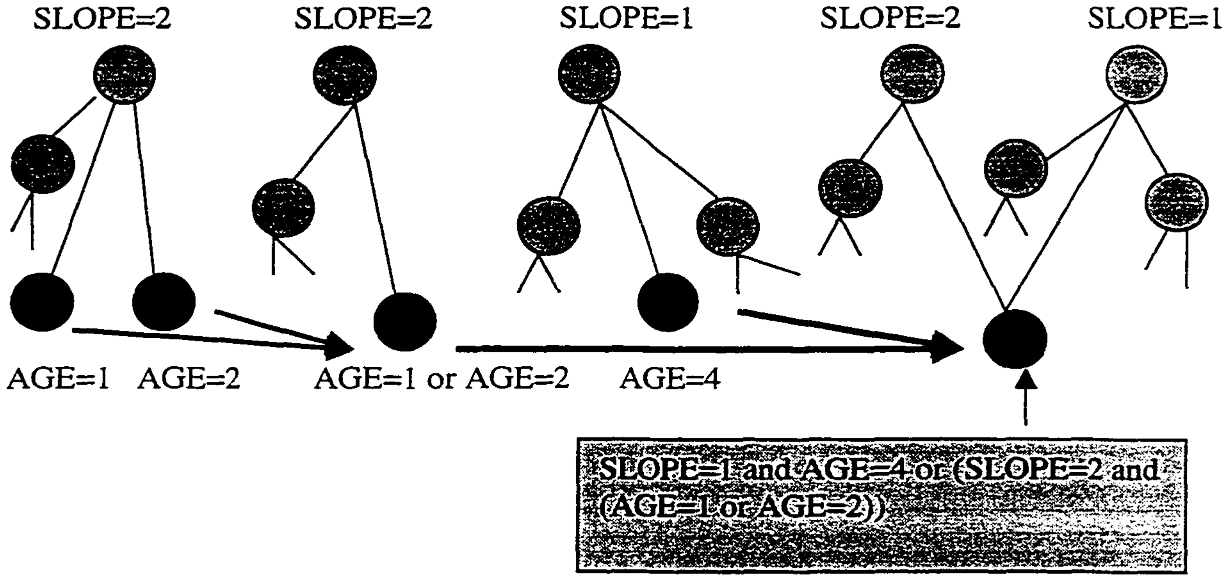


Figure 4.13 Hyper-node creation process with attributes and values involved

The dark nodes represent terminal nodes in the original and merged form. Lines and gray nodes represent other branches. The final rule is presented in a light-gray text box at the bottom right of Figure 4.13. This figure will be helpful in the discussions about the hyper-node and in conclusion whether or not it should be pursued in any future research.
The second trial is performed on the RRSP data set with the logarithmic functional.

Number of final nodes before/after	Individual threshold per node	Threshold for merged nodes	Number of classes	Random set creation %	Type of functional
1232/978	65%	70%	4	Fixed	logarithmic

Table 4.51 RRSP data set-comparison before/after merging of proximate terminal nodes

The third trial is performed by the combination of the Swiss/Hungarian heart data sets where the Swiss data is used as a training data set and the Hungarian data is used as a testing data set with the sinusoidal functional. After the new rule generation, the time necessary to test the rules with the test data set is compared with the time before the merging and rule regeneration. Time is calculated as the average of the twenty rule-testing trials for better accuracy.

Number of final nodes before/after	Individual threshold per node	Threshold for merged nodes	Testing rule run-time before/after seconds	Number of classes	Random set creation %	Type of functi onal
66/61	65%	70%	6.3/5.1	3	Fixed	sinus oidal

Table 4.52 Swiss/Hungarian data set-comparisons before/after merging of proximate terminal nodes

As mentioned above, in the next chapter, section 5.7 thorough discussion is given for further use and research of the developed method.

In the next section the results of different manipulations on the decision trees obtained by using the database inherent structuring, is provided.

4.12 Manipulation of Decision Tree Nodes Using Database Inherent Structuring

In this section, trial results performed on the ideas implemented in Chapter III about the database inherent structures and the advantages it provides are given. There are two sets of trials performed, first on the usage of the different functional on the sub-tree with the rest of the tree built by other type of functional. The second set of trials is performed on the manipulation of nodes and sub-trees by changing the data in the intermediate and terminal nodes. Based on this some predictive and comparative findings were obtained. Also, general predictivity of decision trees in databases is given with a small set of tabular results. Lastly, result examples of the tracking of discrete attribute values to its continuous values in the pre-discretized data set(s) are provided. It proves the extra usefulness of databases in data mining research and an easy ability to track the data to its original form before the data mining process and results were started and obtained.

Experiments were performed on the Swiss/Hungarian data sets, both two-class and three-class cases.

The first set of trials deals with usage of the initial functional, and at a certain chosen level of different functional used to recreate the decision tree at that particular sub-branch. After that, results are compared and presented before/after the above process took place.

Number of final nodes before/after	Training set classification % before/after	Test set classification % before/after	Trial layer	Number of classes	Type of functional (original/second)
54/55	93/92	80/80.5	2	2	logarithmic/parabolic
58/59	91/92	82/80.5	2	2	parabolic/triangular
60/57	93/92.5	80/81	2	2	triangular/sinusoidal
55/55	93/93	80/80	2	2	sinusoidal/logarithmic
63/62	90/89.5	74/73.5	2	3	logarithmic/parabolic
67/66	87/87	78/77.5	2	3	parabolic/triangular
63/62	88/88	76/76.5	2	3	triangular/sinusoidal
66/66	90/89	75/74	2	3	sinusoidal/logarithmic

Table 4.53 Swiss/Hungarian data sets-comparison of decision tree performance before/after the usage of different functional at the chosen sub-tree

As in the previous sections, explanations for tabular results and columns are as following,

- Number of final nodes represents the number of terminal nodes (leafs) built in the decision tree from the training data set before/after the procedure.
- Classification of training set represents a homogeneity of terminal nodes, namely if values in the nodes are all belonging to a single class or not. Values are presented for before/after the functional change procedure.
- Test set classification represents how well the test data records were represented and captured by the rules obtained from the decision tree built by the data in the training set before/after the procedure.
- Replacement layer represents the layer in the tree structure where the different functional was used on the chosen subset of data (particular sub-branch).
- Number of classes in the data set.
- Type of functional-which functional is used for trial cases presented. Both the original functional and the sub-branch testing functional.

The second set of trials deals with the usage of the addition/deletion of data in the subsets (sub-branches) used to recreate the decision tree at that particular sub-branch. After that, results are compared and presented before/after the above process took place.

Number of final nodes before/after	Training set classification % before/after	Test set classificati on % before/after	Data added or deleted	Trial layer	Number of classes	Type of functional
54/61	93/94	80/79	added	2	2	logarithmic
58/55	91/91	82/82	deleted	2	2	parabolic
63/67	88/88	76/76.5	added	2	3	triangular
66/61	90/89.5	75/75	deleted	2	3	sinusoidal

Table 4.54 Swiss/Hungarian data sets-comparison of decision tree performance before/after the adding/deleting of data at the chosen subset (sub-tree)

Next, predictivness and tracking of continuous data values is given with test examples.

Predictivness of decision trees in databases can be seen from the simple case of determining if a patient with AGE=1 and CP=4 belongs to which CLASS based on the decision tree and the data set that tree was created from.
A second case is if SEX=1 and CP=3 and TRESTBPS=2 and CLASS=3 what will be the most probable AGE of the patient.

For the Swiss heart data set with three classes following is obtained by a simple database query either against the rules generated or against the source data set. The following is the sample result of the above question. Of course, for this query to obtain a predictive pattern, previous data of the same structure had to exist, otherwise predictivity would not be possible.

Query (question) presented	Predominant value	Percentage
AGE=1 and CP=4	CLASS=3	50%
SEX=1 and CP=3 and TRESTBPS=2 and CLASS=3	AGE=3	100%

Table 4.55 Predictivity query examples for Swiss heart data set with three classes

Last, tracking of data to its continuous form is presented. Before the data source is used for the building of decision trees, it has to be discretized as explained in previous chapters. During that process, unique identifiers can be assigned to each record (row) in the data set using data base tools. Due to the structure of decision trees in databases and the nature of them being stored in the subsets existent in the database, tracking the discretized values in terminal nodes all the way back to the original values in the source (pre-discretized data set) is possible.

In the table below, a few examples of such tracking are presented for the Swiss heart data set with three classes.

Discretized values	Continual values
ID= 57 and AGE=3 and TRESTBPS=2 and THALACH=2	ID=57 and AGE=53 years and TRESTBPS=125 and THALACH=112
ID=108 and SEX=1 and CP=4 and AGE=4 and CLASS=3	ID=108 and SEX=male and CP=asymptomatic and AGE=61 years and CLASS=needs more diagnosing

Table 4.56 Tracking of the original (pre-discretized) values from the terminal nodes using database tools for Swiss heart data set with three classes (chosen values)

Attribute descriptions are provided in the table below, for easier reading.

Attribute Name	Definition
TRESTBPS	Resting blood pressure
AGE	Age of subject
CP	Chest pain
THALACH	Maximum heart rate achieved
SEX	Gender of subject
CLASS	Class identifier-determining attribute
ID	Unique record identifier

Table 4.57 Attribute definitions for Swiss heart data set-three classes

This is useful in the sense that the actual data in the terminal node can be grouped in the linguistic (verbal) way, easy to follow by any type of user or interested party. More of this discussion will be provided in the following chapter in section 5.8.

Trials and examples provided and the results obtained in this section should prove the database approach and database tools usefulness in data mining attempts.

4.13 Conclusions

Further analyses and discussion of the trial results from the last and previous sections are provided in depth in the next chapter. The important points proven are that all of the novel ideas developed in chapter III, implemented and tested in this chapter, proved its usefulness and logical nature. Variations of certain parameters provided for controlled deviation in the research results, which was positively expected. Data sets that were differently discretized performed with different rules induced, trees built and classification achieved. This is also true for different types of entropy functions used in this chapter and throughout the research. The algorithms and ideas of missing value replacement, alternate branching, multiattribute class, terminal node proximity and database manipulative capabilities should be definitely addressed and further developed in new research where each of those can be considered as a separate focal research point.

CHAPTER V

Discussions

In this chapter the results from the previous chapter are analyzed, discussed and properly presented. Structure of this chapter is the same as the structure of the previous two chapters. Namely, the respective section of necessary result discussions and analyses in Chapter V presents each result section from Chapter IV.

Result analyses are given in the textual, tabular and where necessary, figure and graph form.

5.1.1 Interpreting Decision Trees in Multiclass Environment

In this section result analyses and discussion are provided to compare the performance of two-class vs. multi-class decision trees.

Computational speed- normal pruning and alternate branching- minutes	Training set classification (accuracy) %- normal, pruning and alternate branching	Test set classification (accuracy) %- normal, pruning and alternate branching	Number of final nodes (complexity)- normal, pruning and alternate branching	Data set name and number of classes
2.936/2.74/6.598	95.7/95.4/NA	84.2/83.4/NA	46.7/43.7/135.8	Swiss heart data-2
3.771/3.163/5.182	93.5/93.4/NA	85.7/84.5/NA	53.3/48/93.8	Swiss heart data-3
1.685/1.348/1.856	77/72.6/NA	77/75.3/NA	34.9/21.3/38.2	Liver data-2
1.424/1.279/2.064	63.5/62.6/NA	63.5/63/NA	31.1/25/45.6	Liver data-3

Table 5.1 Summary table of results for two-class vs. multi-class experiments

Table 5.1 provides averages of the results obtained from section 4.5.1.

- Computational speed is represented in minutes.
- Accuracy of decision trees in percents-both training and testing data.
- Total number of terminal nodes in decision trees created.
- Data set type used for the experiment.

Since both of the records were of smaller size than several hundred records, there is some bias based on this fact, but it should not influence the overall results.

For example, for the Swiss data set two-class and three-class cases, computational speed of the normal decision tree induction was higher for the two-class data set by 22%.

- Overall, for both two-class data sets normal computational speed was 11% higher than for the three-class. Pruning of the two-class decision trees was also faster than the three-class pruning case by 8%. On the other hand, alternate branching computational time (speed) was higher for the two-class than the three-class data sets by 16.5%.
- Overall complexity for normal two-class decision trees is 3% lower than for the three-class case. Complexity after pruning, the two-class has 11% lower complexity than the three-class case. Complexity for the two-class is 25% higher than for the three-class in the case of alternate branching.
- Training set accuracy (homogeneity) of two class decision trees for the two-class case is 10% higher than for the three-class case for a normal decision tree. Accuracy after pruning is 7% higher for the two-class case.
- Training set accuracy (rule classification) of the two-class decision trees is 8% higher for normal decision trees (maximum tree). In the case of pruning, accuracy of the two-class decision trees was 7.6% higher than for the three-class case.

As seen from the above, computational speed for the creation of the maximum tree is better for two-class cases. It is as expected since the two-class trees have less possible outcomes than the trees with multi-classes. Pruning results also perform better for the two-class case, since in the case of lower complexity, any pruning attempt will prune larger subsets and sub-branches. Accuracy of two-class decision trees is higher, since determination for the training and testing set accuracy is only between two possible outcomes (classes). Also, since data sets tested in this section are on the smaller size, classification rate of test data set is lowered since fewer rules are created and tested by the decision tree-data mining process. Alternate branching proves all of the above findings in the sense that since two-class decision trees take less time to compute and have lower complexity due to only two classes, there are multiple attributes in the build of decision tree nodes with the same information gain. Less determining classes will result in lower attribute differentiation. By the theoretical process of the design of decision trees, the first attribute is chosen as the most dominant one when deciding among multiple attributes of equal gain at the particular sub-branch. This is why two-class decision trees will have more alternate branches than three-class decision trees.

5.1.2 Indirect Decision Tree Approach for Multi Database Structures

In this section, result analyses are given for different data set merging trials by the use of a decision tree approach.

This ambitious idea of merging different data sets with some attributes of the same nature

and some not, has proven to be very successful. As explained in Chapter III, section 3.2.2 relevance types are used for result computations, namely user assigned, partially guided user assigned and a default relevance (which can be any chosen number of the same magnitudes for all of the attributes in the data set observed).
Values of NA and DSS were chosen based on example 3.2 in Chapter III, section 3.2.2 and a table provided in Chapter IV section 4.1.

Number of qualifying attributes from merge routine	Relevance type	Threshold value for qualifying rules
3.25/3.3/3.5	User assigned/partially guided/default	less than 50%
3.6/2/2.3	User assigned/partially guided/default	50%
1.5/1.5/0	User assigned/partially guided/default	greater than 50%

Table 5.2 Summary table of results for data set merge-different threshold values and relevance types

Table 5.2 provides grouped findings of the results obtained from section 4.5.2.

- Number of attributes in the final data set that qualified from all of the individual data sets.
- Type of relevance-user assigned, partially guided or no relevance at all (default).
- Threshold value for rules to qualify with its attributes for the final (merged) data set.

Trials were performed using different functions, data sets and the way these data sets were discretized but it did not make any significant difference in the direction of final results for any of above described factors to be specifically addressed here.

It is seen from Table 5.2 that the best results on average are obtained for lower threshold values (less than 50%) for qualifying rules which is as expected (value is less-discriminative). Among all of the relevance types, user assigned relevance produced the highest number of qualifying attributes, 23% higher than the partially guided relevance and 44% higher than the default relevance (no relevance). For the threshold values less than 50% default, relevance slightly outperforms the other two. Since user assigned relevance is completely guided by the user, above results are as expected-user assigned relevance did qualify the most attributes on average. This could create a possible problem of an over-optimistic choice of relevance, which would qualify a high number of attributes from each data set in the final data set. On the other hand, as can be seen in the previous chapter some of the user assigned relevance trials did not produce a single qualifying result because relevance values were assigned as too discriminate. It is necessary for the user to know what kind of relevance values to assign to each of the attributes in all of the participating data sets, otherwise the final result can be unrealistic as being too high or too low in number of the selected attributes at the end of the merging process.

Partially guided relevance has produced what could be labeled as the most toned and realistic results, and the relevance itself did not fully depend on user input, although user guidance was necessary as explained in Chapter III, section 3.2.2.

Default relevance (no relevance) did perform better for the threshold values fewer than 50%. Other than for this case, default relevance did not qualify attributes that well, but on the other side it did not require any expert knowledge or user interaction to initialize these trials for data set merging. What should be worthy to point out here is that the different values of NA and DSS (more liberal ones for example) would qualify more attributes in the final set. Also, data sets with more attribute values would qualify more attributes, which would be as expected. The merged data set had fewer attributes than the original data sets and all of the attribute chosen were relevant in the initial data sets, hence the conclusion of success for this particular routine is obtained.

5.2 Use of Different Entropy Functional Discussion and Analyses

In this section, performance and comparison on different types of functional used for the build of decision trees is analyzed and discussed based on the results obtained in section 4.6.

Computational speed-L, P, T and S	Training set classification (accuracy) %-L, P, T and S	Test set classification (accuracy) %-L, P, T and S	Number of final nodes (complexity)-L, P, T and S	Data set name, discretization method used and number of classes
2.712/3.243/2.907/3.079	95.9/95.6/96/95.6	88.5/85.9/85.7/85.1	45.7/53/42.9/44.3	Swiss heart data-P, 2
3.517/3.326/3.445/3.287	95/95/94.3/95.1	78/85.5/84.4/91.5	48.7/53/47.4/50.7	Swiss heart data-P, 3
4/4.1/4.48/4.38	97/95/97/93	96/91/88/96	74/84/81/80	Hungarian data-P, 2
5/4.89/4.5/5.53	93/92/92/93	90/91/87/94	87/94/87/94	Hungarian data-P, 3
1.89/1.49/1.51/1.8	71/71/71/71	68/67/66/68	36/27/33/37	Liver data-U, 2
1.79/0.9/0.93/1.14	56/50/53/52	71/55/56/62	33/16/17/22	Liver data-U, 3
1.69/1.75/1.33/1.68	79/79/78/79	78/75/79/78	39/31/29/39	Liver data-F, 2
2/1.74/1.48/1.61	67/58/58/64	66/64/64/62	41/27/26/32	Liver data-F, 3
88.76/155.49/141.37/ 199.69	100/100/100/100	98/98/97/96	991/2230/1290/ 1990	RRSP data-P, 4

Table 5.3 Summary table of results for four different types of entropy functional

Table 5.3 provides averages of the results obtained from section 4.6.

- Computational speed is represented in minutes for each of the functions. L-logarithmic, P-parabolic, T-triangular and S-sinusoidal functional.
- Accuracy of decision trees in percents-both training and testing data.
- Total number of terminal nodes in decision trees created.
- Data set type used for the experiment, discretization method and a number of classes. P-professional discretization, U-uniform discretization and F-fuzzy discretization

ALKPHOS-liver enzyme (normal)	ALKPHOS-uniform	ALKPHOS-professional	ALKPHOS-fuzzy
92	2	2	2
64	2	2	1
54	1	2	1
78	2	2	2
70	2	2	2
55	1	2	1
62	2	2	1
67	2	2	2
54	1	2	1
60	1	2	1
52	1	2	1
62	2	2	1
64	2	2	1
77	2	2	2
67	2	2	2
78	2	2	2
67	2	2	2
79	2	2	2
107	3	2	3
116	3	2	3
59	1	2	1
23	1	1	1
60	1	2	1
68	2	2	2
80	2	2	2
70	2	2	2
47	1	2	1
66	2	2	2
102	3	2	3

Table 5.4 Example of different attribute values based on the discretization method chosen for ALKPHOS attribute (portion of values)-Liver dataset

Table 5.4 represents values that were discretized differently, part of the attribute set and its original continual values. The data set was provided from the University of Irvine web site [156] and modified properly for use within the research space.

For the professionally discretized Swiss and Hungarian data sets (sets with less than 1000 records), both two-class and three-class cases following is observed,

1. T functional achieves best computational speed, followed by the L, P and S in descending order of efficiency.
2. L functional achieves best training set homogeneity, followed by the T, P and S in descending order of efficiency.
3. S functional achieves best test data set classification rate, followed by the P, L and T in descending order of efficiency.
4. L functional creates decision trees of lowest complexity, followed by the T, S and P in descending order of efficiency.

For the liver data set, which is of different nature than the heart data sets above, the results are different. For the uniformly discretized Liver data set, the following is observed,

1. P functional achieves best computational speed, followed by the T, S and L in descending order of efficiency.
2. L functional achieves best training set homogeneity, followed by the T, S and P in descending order of efficiency.
3. L functional achieves best test data set classification rate, followed by the S, P and T in descending order of efficiency.
4. P functional creates decision trees of lowest complexity, followed by the T, S and L in descending order of efficiency.

For the fuzzy discretized Liver data set, the following is observed,

1. T functional achieves best computational speed, followed by the S, P and L in descending order of efficiency.
2. L functional achieves best training set homogeneity, followed by the S, P and T in descending order of efficiency.
3. L functional achieves best test data set classification rate, followed by the T, S and P in descending order of efficiency.
4. T functional creates decision trees of lowest complexity, followed by the P, S and L in descending order of efficiency.

When we compare the uniformly discretized Liver data set vs. the fuzzy discretized data set, the following is obtained,

1. U (uniformly) discretized data set has 14.4% better computational speed on average than the F (fuzzy) discretized data set.

2. Homogeneity of F data set is 12% higher than for the U data set.
3. Test data classification rate of F data set is 10% higher than for the U data set.
4. Complexity of U data set is 16.3% lower than for the F data set.

Obviously, in the case of the observed Liver data sets, both Uniformly and Fuzzy discretized, the Fuzzy data set provided better accuracy of both training and testing data sets, but at the cost of increased computational time and complexity.

For the RRSP data set with professional discretization and more than 10,000 records, the following is observed,

1. L functional achieves best computational speed, followed by the T, P and S in descending order of efficiency.
2. Each function provided same accuracy (homogeneity) of the training data set used for creating the decision trees. Accuracy was 100%.
3. L and P functional achieve best test data set classification rate, followed by the T and S in descending order of efficiency.
4. L functional creates decision trees of lowest complexity, followed by the T, S and P in descending order of efficiency.

What the RRSP data set results show is that due to the higher number of records, general accuracy of the decision tree is improved for both training and testing, with dramatic increase in computational time and complexity when compared to other data set trials. But, the RRSP data set also had more determining classes (4) in it and more attributes as constituent parts of the data set, hence creating more of the computational challenge.

Worthy noting for all of the trial results obtained, is that many trials were performed on a random split between training and testing data sets, and as can be seen from Chapter IV, section 4.6 even some of the random splits on the same data set performed in different fashion with respect to computational speed, accuracy and complexity. When a different selection of testing and training data was used for the above entropy functions, overall performance and trends changed. In the trend and knowledge discovery analysis where the data from the selected year is used as training data, used for the building of decision trees, and the data from each of the observed years is used as separate test data sets, differentiation among functions would be more obvious and permanent based on that fact. Of course, the fact of preprocessing and target data selection would play a significant role in determining which functional would perform better than the others and on which parameters.

Finally, overall comparison of functional performance for all three of the smaller data sets, namely Swiss, Hungarian and Liver (both U and F) is summarized in the table below followed by result discussion. In this table, results are only analyzed with respect to the functional type and overall computational speed, accuracy and complexity obtained.

Computational speed-L, P, T and S	Training set classification (accuracy) %-L, P, T and S	Test set classification (accuracy) %-L, P, T and S	Number of final nodes (complexity)-L, P, T and S
2.851/2.68/2.57/2.813	81.74/79.45/79.91/80.34	79.43/76.8/76.26/79.58	50.55/48.12/45.41/49.87

Table 5.5 Overall performance results for three data sets

As seen from table 5.5 above,

- T functional achieves best computational speed, followed by the P, S and L in descending order of efficiency.
- L functional achieves best training set homogeneity, followed by the S, T and P in descending order of efficiency.
- S functional achieves best test data set classification rate, followed by the L, P and T in descending order of efficiency.
- T functional creates decision trees of lowest complexity, followed by the P, S and L in descending order of efficiency.

Each functional chosen can perform better or worse than the other based on the way it was discretized, inherent structure of pre-discretized data set (attribute complexity and nature, etc.), size of the data set, user preferences in general, etc. In the next chapter, further research suggestions are given for the use of different entropy functional.

What is important is that each functional performed well with respect to classification rates and provided this research with extra options and directions for testing. There were no significant deviations from one functional to another. Also, different discretization methods did produce somewhat different results, such as fuzzy discretized Liver data performed better in the sense of decision tree characteristics than the uniformly discretized Liver data as observed above. When compared to logarithmic functional which is used in the previous works and theory as the entropy functional for decision tree structures, the other three did perform either the same, slightly worse or better on the performance characteristics and measures of computational speed, accuracy and complexity. The option of having four different entropy functions that are successfully tested and can be used for any future research and industrial work, in capturing the rules and associations in the different ways is invaluable. General tendencies and choice of selection of a particular functional over others have to be carefully selected after the trials and data preparation steps are completed. The data set and its natural structure, together with the way it is preprocessed for the data mining process, would effect the build of the decision tree and also would have a particular data points distribution, which in return would make different entropy functions of different shapes to capture and structure the data differently.

Future work and directions that could be achieved with the use of designed entropy functions, their uniqueness and ability to differently capture the data, resulting rules and the way decision trees are designed is given in the next chapter.

5.3 Pruning of Decision Trees

In the next two subsections, results obtained from Chapter IV subsections 4.7.1 and 4.7.2 are analyzed and discussed.

5.3.1 Pruning of Decision Trees- Data Set Size Criterion

In this section, results of numerous trials performed by using the data set size criterion for decision tree pruning as explained in Chapter III are analyzed and discussed.

- Computational speed is represented in minutes before and after pruning.
- Accuracy of decision trees in percents-both training and testing data before and after pruning.
- Total number of terminal nodes in decision trees created before and after pruning.
- Threshold value for data set size criterion.
- Data set type used for the experiment and number of classes.

Computational speed normal and pruning	Training set classification (accuracy)% normal and pruning	Test set classification (accuracy)% normal and pruning	Number of final nodes (complexity) normal and pruning	Threshold pruning value %	Data set name and number of classes
2.99/2.44	94.75/91	81.375/81.5	44.625/42.5	5	Swiss heart data-2 and 3
2.99/2.058	94.75/80.875	81.375/80.125	44.625/33.5	10	Swiss heart data-2 and 3
2.99/1.803	94.75/77.625	81.375/78.375	44.625/28.375	15	Swiss heart data-2 and 3
2.99/1.485	94.75/74.625	81.375/76.375	44.625/23.25	20	Swiss heart data-2 and 3
2.99/1.285	94.75/72	81.375/73.375	44.625/18.875	25	Swiss heart data-2 and 3
1.655/1.35	70.25/67.5	70.75/70.375	33/23.5	10	Liver data-2 and 3
1.66/0.911	70.25/65.75	70.75/68.375	33/18.875	20	Liver data-2 and 3
146.33/20.8	100/94.75	97.25/95.5	1625.25/211.5	2	RRSP data-4

Table 5.6 Summary table of results for data set size criterion pruning

Table 5.6 provides averages of the results obtained from section 4.7.1.

As seen from Table 5.6, different data sets of different nature, number of classes and attributes can have different response to the same pruning algorithm and threshold value. As explained in Chapter III, section 3.5 a sensible choice of pruning threshold has to be achieved for pruning to take in effect correctly.

1. Computational speed increase for Swiss data set is achieved of approximately 20% with maximum deviation of 8.2% for each 5% of pruning threshold value increase.
2. Accuracy of training decision trees dropped by 4% to 31% when the pruning threshold value was increased from 5% to 25%.
3. Accuracy of testing rule classification data was positive 0.175% for the pruning threshold of 5%. Gradual drop of 2% for each 5% of increase in pruning threshold value has occurred starting at 10% pruning threshold.
4. Complexity of decision tree structures dropped 136% from the original value for 25% pruning threshold. For 5% pruning threshold complexity of decision tree dropped by 5%.

As can be seen, values of 15% and higher did lower the computational time significantly at the expense of accuracy and over simplifying of the decision tree. The best pruning threshold value achieved with respect to overall change in the decision tree's performance and computational characteristics was achieved at 10% pruning threshold. Similar findings as ones above are observed for the Liver data set with two different pruning threshold values of 10% and 20% where the 10% pruning threshold value gives the best results and the 20% threshold exhibits the same type of problems as the counterparts in the Swiss data discussion.

As for the RRSP data set, even at 2% pruning threshold value some level of over simplifying is observed with a drop in complexity of more than ten-fold. Computational speed has increased 7 times on average and accuracy of the decision tree does not seem to be adversely effected too much. Again, the number of final nodes and the rules that represent those is drastically lower with highest node level achieved at the 5th layer of the decision tree, whereas in the original data set and decision tree, creation of terminal nodes (rules) went to the 12th layer of decision tree. For higher values of the pruning threshold, decision trees created were induced in a matter of a few minutes with several terminal nodes, that is why 2% threshold value was the only one presented with its results in Chapter IV, section 4.7.1. In conclusion, one has to decide whether speed and simplicity on one side are as desired and necessary before sacrificing all of the possible rules obtained from the maximum design tree and the accuracy that comes with it.

5.3.2 Pruning of Decision Trees- One Layer Method

In this section, results of numerous trials performed by using the one layer method for decision tree pruning as explained in Chapter III are analyzed and discussed.

- Computational speed is represented in minutes before and after pruning.
- Accuracy of decision trees in percents-both training and testing data before and after

pruning.

- Total number of terminal nodes in decision trees created before and after pruning.
- Threshold value for one layer method.
- Data set type used for the experiment and number of classes.

Table 5.7 provides averages of the results obtained from section 4.7.2.

As seen from the Table 5.7 below, different data sets of different nature, number of classes and attributes can have a different response to the same pruning algorithm and threshold value. As explained in Chapter III, section 3.5 and in subsection 5.3.1 above, a sensible choice of pruning threshold has to be achieved for pruning to take in effect correctly.

Computational speed-normal and pruning	Training set classification (accuracy) %-normal and pruning	Test set classification (accuracy) %-normal and pruning	Number of final nodes (complexity)-normal and pruning	Threshold pruning value %	Data set name and number of classes
2.99/3.08	94.75/94.4	81.375/81.5	44.625/42.5	5	Swiss heart data-2 and 3
2.99/2.904	94.75/93.25	81.375/82.125	44.625/36.5	10	Swiss heart data-2 and 3
2.99/2.312	94.75/91	81.375/80.75	44.625/24.5	15	Swiss heart data-2 and 3
2.99/2.285	94.75/89.625	81.375/80.875	44.625/19.5	20	Swiss heart data-2 and 3
2.99/1.946	94.75/84.625	81.375/80.25	44.625/12.25	25	Swiss heart data-2 and 3
1.655/1.405	70.25/68.375	70.75/70.125	33/23.875	5	Liver data-2 and 3
1.66/0.78	70.25/66.625	70.75/70.875	33/6.375	10	Liver data-2 and 3
88.76/83.52	100/100	98/98	991/958	2	RRSP data-4
155.49/135.38	100/100	98/97	2230/1812	5	RRSP data-4
141.37/84.55	100/99	97/96	1290/647	10	RRSP data-4
199.69/13.12	100/99	96/98	1990/87	15	RRSP data-4

Table 5.7 Summary table of results for one layer pruning method

1. Computational speed has initial drop for the Swiss data set due to the algorithm which compares each layer with the next one, hence for the low value of the pruning

threshold, simplification of the decision tree would not be high enough to outmatch the necessary time for the layer comparison routine. After that a gradual increase in computational speed is achieved of approximately 47% maximum increase for value of 25% of pruning threshold.

2. Accuracy of training decision trees dropped by 0.5% to 12% when the pruning threshold value was increased from 5% to 25%.
3. Accuracy of testing rule classification data was positive 0.175% for the pruning threshold of 5% and 1% for the pruning threshold value of 10%. Drop of 1-1.5% is experienced for pruning threshold value of 15%, 20% and 25%.
4. Complexity of decision tree structures dropped 5% for pruning threshold value of 5%. As expected maximum simplification of almost 4 times simplified tree structure (over the original one) is observed for 25% pruning threshold.

As can be seen, a value of 25% and higher did lower the computational time significantly at the expense of accuracy and over simplifying of the decision tree.

The best pruning threshold value achieved with respect to overall change in the decision tree's performance and computational characteristics was achieved at 10% pruning threshold. It is debatable though that more simplification is required for a certain data set and the rules induced from it, in that case 15% pruning threshold would suffice even better.

1. Computational speed and accuracy of Liver data set trials is acceptable for both 5% and 10% pruning threshold values.
2. For the 10% pruning threshold values, over-simplification of decision tree and the rules induced occurred, which invalidates the 10% threshold as a viable choice.

The RRSP data set produced more sensible results and tolerated higher threshold values for the one layer method, when compared to the data set size criterion method.

1. Computational speed gradually decreased with the increase in pruning threshold.
2. Accuracy of training data (homogeneity) was almost identical to the original accuracy.
3. Accuracy of testing data (rule classification) performed almost the same as the original test data.
4. Complexity of the decision tree at 2% was of very small magnitude with a gradual decrease in complexity with the increase of pruning threshold values. At a threshold value of 15%, over-simplification and drastically increased computational speed are noticeable.

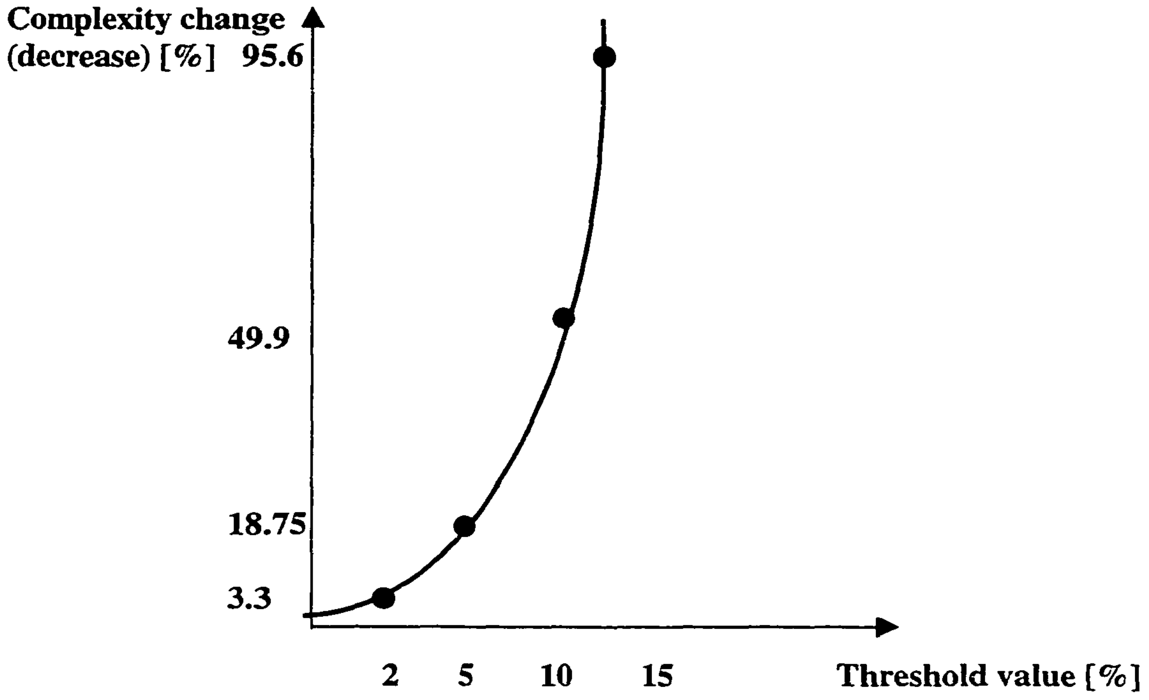


Figure 5.1 Decrease in complexity as a function of pruning threshold value for one layer method-RRSP data set

The number of final nodes and the rules that represent those was not as low as for the data set size criterion method. In section 4.7.2 maximum layer depth of the original decision tree was compared to the maximum layer depth of the pruned decision tree and the highest discrepancy is observed for 15% threshold value. Difference between the original and pruned was 12th layer vs. 10th layer of maximal tree depth. For higher values of pruning threshold, decision trees created were induced in a matter of a few minutes with several terminal nodes, that is why 2%, 5%, 10% were threshold values chosen as ones with legible results. The value of 15% is given to support the obvious fact of this pruning method; that at a certain threshold value, performance deterioration in a non-desired direction is going to happen. The same as for the data set size criterion, one has to decide whether speed and simplicity on one side are as desired and necessary before sacrificing all of the possible rules obtained from the maximum design tree and the accuracy that comes with it.

Based on the results obtained on the particular data sets, the following can be observed about the two pruning methods developed and given in this thesis,

- One layer was more gradual in its approach of pruning than the data set size criterion for the large data sets.

- Computational speed and lower complexity benefits were more dominant in the data set size criterion approach for the large data sets.
- Accuracy, more gradual implementation and flexibility can be achieved by the use of the one layer method for the large data sets.
- Data sets with a higher number of attributes (complexity) and more records (size) are candidates for the one layer method.
- Both performed well in the case of accuracy for the smaller, less complex data sets.
- Data set size criterion performed better in the case of decision tree (rule) complexity for smaller data sets (it did not over-simplify as much as the one layer method for the same threshold value-25% for Swiss and 10% for Liver).

It is important to understand that for some data sets one method will produce better results than the other will. The Liver data set performed better with the data set size criterion due to the lower number of attributes in it. Also, the user has to decide what are the benefits to be achieved by pruning the decision trees to fully qualify one pruning method as the more desirable one over the other for a particular research, particular data set and its characteristics. The difference between structure of two pruning methods given has to be amounted for the different result. One method (data set size criterion) stops computational design of the sub-branch and replaces that branch with the node, the other (one-layer) removes the rules completely from the pruned branches without replacing them

In the next chapter, further directions that can be taken for testing and researching on these two pruning methods developed (designed) and implemented in this thesis are given.

5.4 Alternate Branching of Decision Trees Discussion and Analyses

In this section, trial results obtained for alternate branching are analyzed and discussed. The result are generated in section 4.8 based on the novel ideas given in Chapter III, section 3.7.

Run time-minutes before/after	Number of final nodes before/after	First layer with alternate branch	Number of classes	Data set name and number of classes	Type of functional
2.095/3.62	38/65	3.5	2	Swiss and Liver data	logarithmic
2.375/5.94	37.5/123.5	3	2	Swiss and Liver data	parabolic
1.975/12.705	34/247.5	3	2	Swiss and Liver data	triangular
2.135/3.825	39/58	3.5	2	Swiss and Liver data	sinusoidal
2.78/5.99	46/88.5	3.5	3	Swiss and Liver data	logarithmic
2.495/5.365	38/95	3	3	Swiss and Liver data	parabolic
2.42/6.325	38/104	4	3	Swiss and Liver data	triangular
2.335/4.75	37.5/73.5	3.5	3	Swiss and Liver data	sinusoidal
88.76/456.23	991/5011	3	4	RRSP	logarithmic
155.49/489.11	2230/5872	3	4	RRSP	parabolic
141.37/573.58	1290/7859	3	4	RRSP	triangular
199.69/412.76	1990/4814	3	4	RRSP	sinusoidal

Table 5.8 Summary table of results for alternate branching method

Table 5.8 provides averages of the results obtained from section 4.8.

- Computational speed is represented in minutes before and after alternate branching.
- Total number of terminal nodes in the decision trees created before and after pruning.
- First layer with alternate branching-average of two data set trials for Swiss and Liver data sets, single valued for RRSP data set.
- Number of classes
- Data set types used for the experiment.

Aggregation is done for the Swiss and Liver data sets first for two-class and then for three-class in the consecutive rows in Table 5.8 above, with respect to each different functional observed with the alternate branching method. The RRSP data is provided from Chapter IV as is.

As seen from Table 5.8, the lowest amount of alternate branching (alternate nodes) in the case of smaller data sets (Swiss and Liver), has occurred for the sinusoidal functional followed by the logarithmic, parabolic and triangular as the one with the most alternate branches in the tree created. This pattern is evident for both two-class and three-class data sets.

Conclusions are obvious, alternate branching increases computational time significantly, with respect to the number of attributes and number of classes involved.

The advantage for researchers is that all of the pseudo rules generated from the same attributes but in the different order can be assigned a certain attribute relevance (as in the case of data set merging). Such modified rules would then reveal rule biases and classify the strongest rules that could create a "super-tree". The tree is biased by the attributes and their respective relevance as chosen by the researcher. Further ideas for research on alternate branching will be given in the next chapter.

What this method proved is that it is almost never possible to obtain only one specific tree, regardless of the type of entropy functional used or the data set type. With respect to the functional chosen, based on the research objective, which is both lowered in complexity and computational time, or more rule options and generation, different functional would be chosen to best capture the desired direction by the researcher.

5.5 Missing Attributes Trial Discussion and Analyses

In this section, results given in Chapter IV, section 4.9 and implemented based on the ideas introduced and designed in Chapter III, section 3.4 are summarized, analyzed and discussed

Computational speed-normal and missing	Training set classification (accuracy) %-normal and missing	Test set classification (accuracy) %-normal and missing	Number of final nodes (complexity)-normal and missing	Data set name and number of classes
2.9925/1.9462	94.75/94.75	81.375/81.125	43.375/42.875	Swiss data-2 and 3
1.5925/1.5312	70.25/70.5	70.625/70	35.625/35.75	Liver data-2 and 3
139.22/131.98	100/100	98/98	1659.5/1665.5	RRSP data-4

Table 5.9 Summary table of results before and after the missing values method

Table 5.9 provides averages of the results obtained from section 4.9.

- Computational speed is represented in minutes before and after missing attribute method was applied.

- Accuracy of decision trees in percents-both training and testing data before and after missing attribute method was applied.
- Total number of terminal nodes in the decision trees created before and after the missing attribute method was applied.
- Data set type used for the experiment and number of classes.

1. For all three data sets observed overall computational speed after the replacement of missing values is 5.2% better than for the original data sources.
2. Accuracy of the original data sets for the training data (homogeneity) is less than 1% (0.094%) worse than for the data sets after the missing attribute routine was used.
3. Accuracy of the original data sets for the testing data (rule classification) is less than 1% (0.35%) better than for the data sets after the missing attribute routine was used.
4. Complexity of data sets created after the missing attribute routine was used is less than 1% (0.32%) higher than the complexity of the original data sets.

Conclusions for the missing attribute routine are quite simple. As seen in Chapter IV, section 4.9, where the different decision tree structures were observed, due to the missing attribute routine used the decision tree structures were altered in the sense of a certain sub-branch and attribute order change. But, the alteration of decision tree structure would have happened with any kind of data set manipulation. What is very important, is that the accuracy, complexity and computational speed were almost the same for both normal data sets and the ones with missing values replaced in them by the algorithm. This proves that the algorithm developed in Chapter III and implemented and tested in Chapter IV had preserved the inherent structure of the original data sets.

Further possible use of the implemented novel idea for replacement of missing values using the database approach is given in the next chapter.

5.6 Multi Attribute Class Identifier Discussion and Analyses

In this section, results given in Chapter IV, section 4.10 and implemented based on the ideas introduced and designed in Chapter III, section 3.6 are summarized, analyzed and discussed.

Computational speed	Training set classification (accuracy) %	Test set classification (accuracy) %	Number of final nodes (complexity) %	Data set name and number of classes
5.165	84.75	70.5	58.25	Swiss data-15
1	34.5	38	16.75	Liver data-9
549.37	99	80.5	2552.5	RRSP data-15

Table 5.10 Summary table of results for multi attribute method

Table 5.10 provides averages of the results obtained from section 4.10.

- Computational speed is represented in minutes for multi attribute method.
- Accuracy of decision trees in percents-both training and testing data for the multi attribute method.
- Total number of terminal nodes in decision trees created for multi attribute method.
- Data set type used for the experiment and number of classes.

As seen, the number of final nodes for the Swiss and RRSP data sets is in the same range as the number of final terminal nodes for previous experiments with normal decision trees built from the data sets with a lower number of classes.
Computational time has increased for the RRSP and Swiss data set. Reason for this was that both of the data sets have enough attributes left in them (complexity) after choosing two attributes to form a determining class attribute. In return, computations on the altered data sets have increased in volume.
As for Liver data set, since it had only six attributes in the original data set, after choosing two attributes to create a determining class, both the complexity of the decision tree (data set) and computational time were lowered significantly. At the same time, unacceptable degradation in accuracy of both training (homogeneity) data and testing (rule classification) data had happened.

General conclusions about the multi-attribute method are,

- It is useful if grouping and relationships among multiple attributes and the relationship of such a group to the rest of the data in the data set is to be observed and utilized. This was explained in Chapter III, section 3.6.
- This method is a good candidate for data sets with many attributes and unique values for the attributes.
- For the data sets with low complexity (number of attributes and unique values in them) the multiple attribute method can create over-simplification and degradation in accuracy of both training and testing data.
- To have sufficient testing and training accuracy in general, data sets used for the multi attribute method should have enough values in them to create enough candidates for the testing and classification of generated rules.

This method provides for increased data mining capabilities in data. The possibility to combine the different attributes to create a single attribute that captures the relationships among them and perform knowledge discovery and rule induction on such combined attributes is very useful and potent. Also, by using the database capabilities and having all of the nodes stored permanently in the database, enables the researcher to backtrack the data in the terminal nodes based on the unique value combinations of the chosen attributes in this multi attribute process. What this provides for is a split of the newly created classes in the respective terminal nodes to the original attributes and their values that created that particular class. Again, to fully utilize this method, the data set(s) considered should comply with the findings above, about the general nature of data sets to use for the multi attribute method.

Further ideas and directions for this particular method are given in the next chapter.

5.7 Terminal Nodes Likelihood-Proximity of Decision Tree Subsets Discussion

In this section a result analyses and discussion based on the trial results obtained in Chapter IV, section 4.11, and developed and conceptualized in Chapter III, section 3.8 are given.

Number of final nodes before/after	Individual threshold per node	Threshold for merged nodes	Testing rule run-time before/after seconds	Data set name and number of classes
67.25/60.5	65%	70%	NA	Swiss data-2 and 3
66/61	65%	70%	6.3/5.1	* Swiss/Hungarian-3
1232/978	65%	70%	NA	RRSP-4

Table 5.11 Summary table of results for proximity of terminal nodes method

Table 5.11 provides averages of the results obtained from section 4.11.

- Total number of terminal nodes in decision trees, created before and after terminal nodes were merged.
- Individual qualifying threshold-confidence level (as given in Chapter III, section 3.2, subsection 3.2.2, equation 3.1) value of the observed terminal node (homogeneity).
- Qualifying threshold value of the merged terminal node (homogeneity).
- Runtime of the altered testing rule linguistic structure (transformed in the database query).
- Name of the data set observed and the number of classes.

*In the second row, Table 5.11, combination of the Swiss and Hungarian heart data sets was used, the former as the training data set and the later as the testing data set.

As seen from Table 5.11 above, merging of proximate terminal nodes lowered the number of terminal nodes in the decision trees created from the Swiss data set by 11% on average. Individual threshold and merged threshold values were chosen after several different trials in which was attempted to merge terminal nodes.
As the outcome, rules were modified to capture the logical OR statement that transpired as the outcome of merging.
In one of the trials, the case of a possible two-layer merge of terminal nodes occurred. As presented in that section, a more complex rule was the outcome of such a merger. Computational time and effort to track all of the possible two-layer mergers and its rules, by far outweighs any reasonable gain in lowered complexity and number of nodes created. The general recommendation is that at present, implementation of "hyper-nodes"

would not be beneficial with present computational systems (power, speed, etc.).

For the combination of the Swiss and Hungarian data sets (second row in Table 5.11), merging of proximate terminal nodes lowered the number of terminal nodes in the decision trees created from Swiss data set by 8.1%.

The column with the runtime of the original testing rules vs. the rules created after the merging of terminal nodes (average of 20 trials) shows that the rule testing time was 23.5% higher for the original rules than for the rules with the OR clause in them (rules created as the outcome of the node merge).

The RRSP data set just proves more along the lines, that the more there are terminal nodes in the original tree structure, the more likeliness that some of them can be candidates for the node merge process. For the RRSP data set, the merging of proximate terminal nodes lowered the number of terminal nodes in the decision trees created from the RRSP data set by 25.9%.

General conclusions about trial results summarized in Table 5.11 and already addressed to a certain extent in Chapter IV and analyzed are following,

- One-layer merge at present proves that there are advantages in the sense of complexity (lowered number of nodes-database tables). Structure of less tables and creation time for decision tree nodes is better/lower.
- Lowered amount of time necessary to test generated rules on test data with the OR modified rules.
- Two-layer merge is not a viable option at present.
- There is a tradeoff between increased computational time to look for the proximate (likely similar) nodes, as explained in Chapter III, section 3.8 on one side and a rule testing time and lowered complexity on the other.

Once using this approach, it has to be carefully considered if increased initial time to test for the proximate nodes and rule recreation at those is lower than the benefit of increased rule testing speed and lowered complexity and structure creation.

In the next chapter, further research direction is suggested for this idea and in particular viability of the possible multi layer merge of nodes.

5.8 Manipulation of decision trees using database inherent structuring-discussions

In this section, results given in Chapter IV, section 4.12 and implemented based on the ideas introduced and designed in Chapter III, section 3.9 are summarized, analyzed and discussed.

Number of final nodes before/after	Training set classification % before/after	Test set classification % before/after	Replacement final layer	Data set name and number of classes
56.75/56.5	92.5/92.375	80.5/80.5	2	Swiss/Hungarian data-2
64.75/64	88.75/88.375	75.75/75.375	2	Swiss/Hungarian data-3

Table 5.12 Summary table of results for different functional method

Table 5.12 provides averages of the results obtained from section 4.12.

- Total number of terminal nodes in decision trees created before and after using different functional.
- Accuracy of decision trees in percents-both training and testing data before and after using different functional.
- Replacement layer represents the layer in the tree structure where the different functional was used on the chosen subset of data (particular sub-branch).
- Data set type used for the experiment and number of classes.

The Swiss data set is used for both two-class and three-class cases to create a decision tree. The Hungarian data set is used for testing data and rule classification before and after the second functional was introduced at the second layer and at the chosen subset of data (sub-branch).

The following is observed,

1. Number of final nodes for both two-class and three-class cases was smaller on average than before second functional method was used.
2. Homogeneity was less than 1% decreased on average after the method was used, but in some of the individual trials homogeneity was higher after the second functional method was used.
3. Accuracy of the test data-rule classification was almost identical for both two-class and three-class cases before and after the method was applied on the original decision tree structure.

Certain functional combination will produce better results than the others will and it should be carefully implemented to benefit research itself. In the larger data sets, even small gain in one characteristic (increase in accuracy for example) can benefit the research direction greatly. The best way to use this method is to perform several trials on a given original data source, until the desired results are obtained by using the particular combination of the original and second functional. Further ideas for research and implementation of this method are given in the next chapter.

Next, addition and deletion of data from the chosen subset(s) and the influence they have on the overall decision tree structure is analyzed and discussed.

Number of final nodes before/after	Training set classification % before/after	Test set classification % before/after	Data added or deleted	Trial layer	Data set name and number of classes
58.5/64	90.5/91	78/77.75	added	2	Swiss/Hungarian-2 and 3
62/58	90.5/90.25	78.5/78.5	deleted	2	Swiss/Hungarian-2 and 3

Table 5.13 Summary table of results for add/delete method

Table 5.13 provides averages of the results obtained from section 4.12.

- Total number of terminal nodes in decision trees created before and after using different functional.
- Accuracy of decision trees in percents-both training and testing data before and after using different functional.
- Status field-information whether the process of addition or deletion took place in the particular sub-tree.
- Replacement layer represents the layer in the tree structure where the different functional was used on the chosen subset of data (particular sub-branch).
- Data set type used for the experiment and number of classes.

Same as for the second functional method, the Swiss data set is used for both two-class and three-class cases to create a decision tree. The Hungarian data set is used for testing data and rule classification before and after the second functional was introduced at the second layer and at the chosen subset of data (sub-branch).

The following is observed,

1. Logically, the deletion method will lower the complexity of the final decision tree, therefore the addition method would increase it.
2. Accuracy of both training and testing data will vary based on the added or deleted records from the data set-the nature of the records deleted and their influence on the decision tree structure.

The same can be concluded as for the different functional method, several trials should be performed on a given original data source, until the desired results are obtained by using the particular add/delete function.
Further ideas for research and implementation of this method are given in the next chapter.

Lastly in this section, ideas from Chapter III, section 3.10 and results of simple predictivity and querying against the data structure from Chapter IV and discussion on tracking of the data to its continual (pre-discretized) form is given.

Based on the inherent capabilities of the database and its tools it is easy to predict certain patterns in the data itself. Questions can be asked such as in Chapter IV, section 4.12, if by knowing a few attribute values for a particular patient, its overall well-being and some other attributes can be predicted. It can be done easily either using the database tables or the rules generated as the part of the decision tree build. As mentioned in Chapter IV, a certain level of data of similar nature has to exist to be able to recognize the pattern or make a predictive choice. Due to the complex challenge of result presentation in this particular case, much of the explanation was already given in Chapter IV, section 4.12 on this subject. Each predictive question will be answered with a certain probability (level of correctness) based on the underlying data in the data set. If the absolute answer exists for the question asked, then the answer will be 100% correct as seen in section 4.12. If there are more patterns to the question than one, then the answer will be labeled by the most predominant case in the data set. By using the databases, as explained earlier, it is not necessary to use rules generated from the decision tree build, but it certainly would be helpful, especially if exploratory research is performed without firm direction about the questions to be asked.

As for the tracking of the data records to its original form, before discretization was performed on attributes and its values, the database provides unparalleled advantage. By simply tagging each record (row) in the data set (table) with the unique identifier, all of the values from any terminal node(s) observed in the decision tree, the continual case could be deciphered for all of the records in the nodes. Due to the database structure these nodes are tables itself, hence being enabled against any database query and mining tools. For example, if the Swiss heart data set is found that a particular node with 100% homogeneity and class of sick patients has the most determinant attribute in it to be age with its discretized value of 5, by backtracking each of the records in that particular node (table) to the original data source (table), it would be easy to discover and group the results by the fact that the particular patients were of ages of 81, 81, 87 and 88 years old and all of them males.

Possible further ideas on use and enhancements that could be achieved by using databases and their tools in the space of data mining are given in the next chapter.

5.9 Conclusions

The last section in this chapter has a form of the chapter summary with respect to each of the sections and the analyses and discussions performed in each of those.

1. By using database tools and software it was easier to implement algorithms and to test it on the data sets. Decision tree creation is less size-sensitive even in the generic form without any pruning applied to it.

2. Pruning algorithms developed and tested are very successful in the creation of decision trees, without loss of the tree's accuracy.

3. Different entropy functions developed and implemented prove to have advantages over each other and provide for more flexible and enhanced research in the general space of data mining and decision trees.

4. The method of replacing missing values is simple and database oriented with low bias in the missing values replaced.

5. Reusability of intermediate nodes is possible. Intermediate nodes can be used and analyzed because they stay preserved in the database in the form of a table (data set).

6. Generation of rules is easy to implement and follow, and the rules itself can be manipulated and altered.

7. Multi attribute class solution is flexible to realize and analyze, with extra options provided for research as the outcome.

8. Merging of data sets is very successful and it can be applied in any real life application dealing with mismatch among multiple sources that should be joined on a certain level. Creation of the final data set is as simple as the merged final data set following the logic developed in Chapter III and retains the most relevant attributes in the data set.

9. Authenticity and tracking of the data set can be preserved and subtly manipulated if needed, by using database tools and structure.

10. Ability to obtain alternate branches and alternate tree structures provides any researcher with the option of manipulating and comparing the attributes on a more complete level, and assigning biases (relevance) to the alternate trees and rules if desired in the course of research to observe the behavior of attributes in the rules itself.

It is obvious that decision trees created in the database are very robust since they can be stored and further manipulated within the database, but the overall computational speed and efficiency is dependant on the size of data observed, the complexity of attributes in the data set observed and the number of determining classes in it.

This is confirmed in the experimental results obtained in this research.

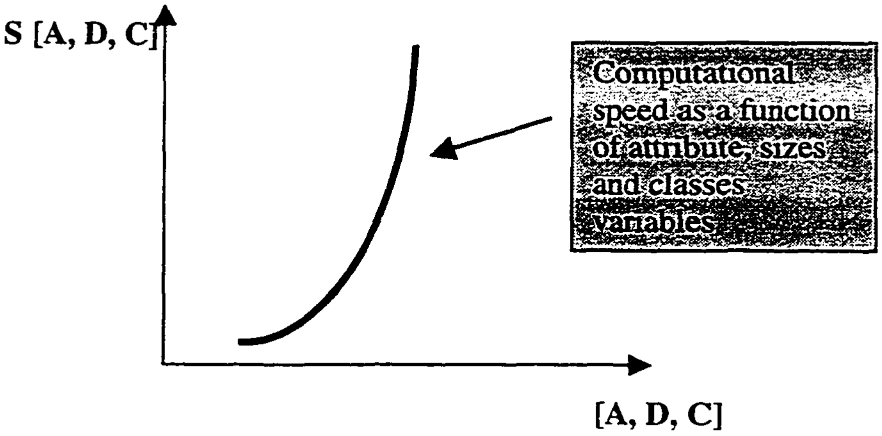


Figure 5.2 Relationship between computational speed and data characteristics

In Figure 5.2, direct relationship between computational speed-S and attribute complexity-A, data set size-D and number of determining classes-C is presented.

This is given as a general anticipated form based on the research results obtained.
In the case of the RRSP data set, this becomes more obvious for the multiattribute class case where with the increase in the number of classes computational speed (efficiency) increased several times when compared to the original results. In the case of the database environment where data sets with millions of records are existent and they can have more than several values for its class identifier, then the computational speed would be even more hampered by the complexity. A final decision tree could require days of computational processing to successfully build. It is important to realize that in the real database environment computational speed is directly related to the hardware and software infrastructure itself, meaning that a better system will eliminate certain slow-downs due to the increased complexity.

In the next section recommendations for further research and enhancements that could be performed on the ideas and algorithms developed, implemented and tested in this thesis is given.

CHAPTER VI

Conclusions

This research has shown that the database approach and SQL language functionality have great potential in the data-mining world. The process of creating the decision trees and all the novel ideas presented in this thesis went according to expectations and even surpassed them, which is a solid accomplishment by itself.

There is a number of important issues worthy of further exploration based on the research and the results obtained.
Further enhancements and possible directions for the ideas developed, implemented and tested in this research are given below.

1. Indirect decision tree approach for multi database structures-further research could be performed in the direction of relevance exploration and rule strength. Possibly, a combination with a missing attributes routine to create matching attributes in the target data sets from the attributes in the source data set, based on the algorithms developed in this thesis.

2. Use of different entropy functional:further directions are numerous. Research on each of the new functions developed can be lengthy and beneficial to the research community. From different applications in pruning of the decisision tree to designing a better decision tree for a particular research field and data source possibilities are numerous. Also, comparison in more depth, which functional captures data of a certain type, complexity and context better than the others. This can all be coupled with different discretization techniques, such as the ones used in this reseacrh, namely uniform, professionally guided and fuzzy discretization.

3. Pruning of decision trees:by using the two algorithms developed, significant pruning that is database centric can be achieved. More research could be done on the accuracy aspect of pruned decision trees and the application of the two algorithms in the different fields of research on the large data sets.

4. Alternate branching of decision trees:manipulation of alternate trees and its usefulness in determining a "super-tree" and best bias in attributes could be a research topic in itself.

5. Missing attributes method:this method can be enhanced and tested against many different data sources. It could also be used in analyzing possible predictive variations in data structures, by changing the source data into guided data.

6. Multi attribute class identifier:this can definitely find research application and attention in attribute grouping and “super-class” data mining.

7. Terminal nodes likelihood:proximity of decision tree subsets-possible further research is in the space of merging over multiple layers, creation of hyper-rules and its computational advantages and disadvantages in more details.

8. Manipulation of decision trees using database inherent structuring:further research goes in direction of manipulation of data sets and decision trees with different functional used, ability to research on data in terminal nodes and alter it, to enrich the data itself and to utilize the database capabilities better in the space of data mining.

Obviously, this list can be expanded during the future exploration and research to be done based on the ideas introduced and implemented in this research, but even to approach each of the points given above individually could require a serious effort and potentially would provide very useful results and findings.

REFERENCES

[1] G. Adomavicius, A. Tuzhilin, "User Profiling in Personalization Applications Through Rule Discovery and Validation", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 377-381, 1999.

[2] P. Adriaans, "Data Mining", Addison-Wesley, 1996.

[3] C. C. Aggarwal, S. C. Gates, P. S. Yu, "On the Merits of Building Categorization Systems by Supervised Clustering", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 352-356, 1999.

[4] C. C. Aggarwal, J. L. Wolf, K. Wu, P. S. Yu, "Horting Hatches an Egg: A New Graph-Theoretic Approach to Collaborative Filtering", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 201-212, 1999.

[5] R. Agrawal, T. Imielinski, A. Swami, "Database Mining: A Performance Perspective", IEEE Transactions on Knowledge and Data Engineering, vol. 5, 6, 914-925, 1993.

[6] M. R. Anderberg, "Cluster Analysis for Applications", Academic Press, New York, 1973.

[7] M. Ankerst, C. Elsen, M. Ester, H. Kriegel, "Visual Classification: An Interactive Approach to Decision Tree Construction", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 392-396, 1999.

[8] Y. Aumann, Y. Lindell, "A Statistical Theory for Quantitative Association Rules", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 261-270, 1999.

[9] N. F. Ayan, A. U. Tansel, M. E. Arkun, "An Efficient Algorithm to Update Large Itemsets with Early Pruning", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 287-291, 1999.

[10] E. Backer, "Computer-Assisted Reasoning in Cluster Analysis", Prentice Hall, New York, 1995.

[11] W. M. Bain, "A Case-based Reasoning System for Subjective Assessment", AAAI-86, 523-527, 1986.

[12] P. Baldi, S. Brunak, "Bioinformatics: The Machine Learning Approach", Adaptive Computation and Machine Learning, MIT Press, 1999.

[13] D. Barbará, X. Wu, "Using Approximations to Scale Exploratory Data Analysis in Datacubes", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 382-386, 1999.

[14] S. D. Bay, M. J. Pazzani, "Detecting Change in Categorical Data: Mining Contrast Sets", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 302-306, 1999.

[15] R. J. Bayardo Jr., R. Agrawal, "Mining the Most Interesting Rules", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 145-154,1999.

[16] K. P. Bennett, U. M. Fayyad, D. Geiger, "Density-Based Indexing for Approximate Nearest-Neighbor Queries", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 233-243, 1999.

[17] M. J. A. Berry, G. Linoff, "Data Mining Techniques: For Marketing, Sales, and Customer Support", John Wiley & Sons, 1997.

[18] J. C. Bezdek, "Pattern Recognition with Fuzzy Objective Function Algorithms", Plenum Press, New York, 1981.

[19] J. M. Bibby, J. T. Kent, K. V. Mardia, "Multivariant Analysis", NY: Academic Press, 1979.

[20] F. Bonchi, F. Giannotti, G. Mainetto, D. Pedreschi, "A Classification-Based Methodology for Planning Audit Strategies in Fraud Detection", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 175-184, 1999.

[21] R. Brachman, T. Anand, "The Process of Knowledge Discovery in Databases", In: Advances in Knowledge Discovery and Data Mining (U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy, editors.), American Association for Artificial Intelligence, California, 353-373, 1996.

[22] L. Breiman, J.H. Friedman, R.A. Olsen and C. J. Stone, "Classification and Regression Trees", Belmont, California, 1984.

[23] W. Brenner, R. Zarnekow, H. Wittig, "Intelligent Software Agents: Foundations and Applications", Springer Verlag, 1998.

[24] L. C. Briand, K. El Emam, D. Surmann, I. Wieczorek, K. D. Maxwell, "An Assessment and Comparison of Common Software Cost Estimation Modeling Techniques", ACM Proceedings, ICSE '99 LA, CA, 313-322, 1999.

[25] T. Brijs, G. Swinnen, K. Vanhoof, G. Wets, "Using Association Rules for Product Assortment Decisions: A Case Study", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 254-260, 1999.

[26] S. Brin, R. Rastogi, K. Shim, "Mining Optimized Gain Rules for Numeric Attributes", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 135-144, 1999.

[27] C. Brunk, J. Kelly, and R. Kohavi, "MineSet: An integrated system for data mining", In: Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining, Newport Beach, CA, August 14-17, 135-138, 1997.

[28] W. Buntine, "Operations for Learning with Graphical Models", Journal of Artificial Intelligence Research, 2: 159-225, 1994.

[29] W. L. Buntine, B. Fischer, T. Pressburger, "Towards Automated Synthesis of Data Mining Programs", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 372-376, 1999.

[30] D. Burleson, "Oracle Data Warehousing", Coriolis Group Books, Arizona, 1997.

[31] P. Cabena, P. Hadjinian, R. Stadler, J. Verhees, A. Zanasi, "Discovering Data Mining from Concept to Implementation", Prentice Hall, 1997.

[32] Y. Cai, N. Cercone, J. Han, "Attribute-Oriented Induction in Relational Databases", In knowledge Discovery in Databases, ed. G. Piatetsky-Shapiro and W. J. Frawley, 213-228, California, AAAI Press, 1994.

[33] J. G. Carbonell, R. S. Michalski, T. M. Mitchell, "An Overview of Machine Learning", Machine Learning, an Artificial Intelligence Approach, vol. 1, 3-24, Morgan Kaufmann, California, 1983.

[34] J. Chattratichat, "Large scale data mining: challenges and responses", In: Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining, Newport Beach, CA, August 14-17, 143-146, 1997.

[35] C. H. Cheng, A. W. Fu, Y. Zhang, "Entropy-based Subspace Clustering for Mining Numerical Data," Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 84-93, 1999.

[36] W. W. Chu, K. Chiang, "Abstraction of High Level Concepts from Numerical Values in Databases", In knowledge Discovery in Databases: Papers from the 1994 AAAI workshop, 37-48, AAAI Tech. Rep. WS-94-03, California, 1994.

[37] K. J. Cios, R. Tjia, N. Liu and R. A. Langenderfer, "Study of continuous ID3 and radial basis function algorithms for recognition of defects in glass" In: International Joint Conference on Neural Networks, IJCNN'91, Seattle, July, I4954, 1991.

[38] K. J. Cios and L. Sztandera L., "Continuous ID3 algorithm with fuzzy entropy measures." In: First IEEE Int. Conf. on Fuzzy Systems, San Diego, March, 469-476, 1992.

[39] K. J. Cios, "Fuzzy CID3 algorithm: combining neural networks and fuzzy sets", In: Ohio Area Neural Networks Workshop, Cleveland, June, 1993.

[40] K. J. Cios, N. Liu and L. S. Goodenday, "Generation of diagnostic rules via inductive machine learning", Kybernetes, 22(5): 44-56, 1993.

[41] K. J. Cios and N. Liu, "An algorithm which learns multiple covers via integer linear programming, Part I - the CLILP2 algorithm", Kybernetes, 24(2): 29-50, 1995.

[42] K. J. Cios and N. Liu, "An algorithm which learns multiple covers via integer linear programming, Part II -experimental results and conclusions" Kybernetes, 24(3): 28-40, 1995.

[43] K. J. Cios, D. K. Wedding and N. Liu, "CLIP3: cover learning using integer programming", Kybernetes (invited paper), 26(4-5): 513-536, 1997.

[44] P. Clark and T. Niblett, "The CN2 Induction Algorithm", Machine Learning 3: 261-283, 1989.

[45] Communications of the ACM - a special issue on Data Mining, 11, 1996.

[46] C. Cortes, D. Pregibon, "Information Mining Platforms: An Infrastructure for KDD Rapid Deployment", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 327-331, 1999.

[47] R. Dave, "Characterization and detection of noise in clustering", Pattern Recognition Letters, 12, 657 – 664, 1992.

[48] S. Davies, A. Moore, "Bayesian Networks for Lossless Dataset Compression", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 387-391, 1999.

[49] L. Davis, "Handbook of Genetic Algorithms", Van Nostrand Reinhold, New York, 1991.

[50] K. A. DeJong, "Learning with Genetic Algorithms", Machine Learning, 3, 121-138, 1988.

[51] A. Dempster, N. Laird, D. Rubin, "Maximum Likelihood from Incomplete Data Via the EM Algorithm", Journal of the Royal Statistical Society, B 39:1-38, 1977.

[52] M. Derthick, J. Kolojejchick, and S. F. Roth, "An interactive visualization environment for data exploration", In: Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining, Newport Beach, CA, August 14-17, 2-9, 1997.

[53] V. Dhar, R. Stein, "Seven Methods for Transforming Corporate Data Into Business Intelligence", Prentice Hall, 1996.

[54] P. Domingos, "MetaCost: A General Method for Making Classifiers Cost-Sensitive", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 155-164, 1999.

[55] G. Dong, J. Li, "Efficient Mining of Emerging Patterns: Discovering Trends and Differences", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 43-52, 1999.

[56] J. Dörre, P. Gerstl, R. Seiffert, "Text Mining: Finding Nuggets in Mountains of Textual Data", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 398-401, 1999.

[57] W. DuMouchel, C. Volinsky, T. Johnson, C. Cortes, D. Pregibon, "Squashing Flat Files Flatter", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 6-15, 1999.

[58] J. Edmonds, "Optimum Branching", J. Res. NBS, 71B: 233-240, 1967.

[59] B. Efron, "Bootstrap Methods: Another Look at the Jackknife", Annals of Statistics 7: 1-26, 1979.

[60] B. Efron, G. Gong, "A Leisurely Look at the Bootstrap, the Jackknife, and Cross-Validation", American Statistician 37: 36-48, 1983.

[61] R. Elmasri, S. B. Navathe, "Fundamentals of Database Systems", Second Edition, California, Benjamin/Cummings, 1994.

[62] L. P. English, "Improving Data Warehouse and Business Information Quality: Methods for Reducing Costs and Increasing Profits", John Wiley & Sons, 1999.

[63] B.S. Everitt, "Cluster Analysis", Heinemann, Berlin, 1974.

[64] W. Fan, S. J. Stolfo, J. Zhang, "The Application of AdaBoost for Distributed, Scalable and On-Line Learning", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 362-366, 1999.

[65] T. Fawcett, F. J. Provost, "Activity Monitoring: Noticing Interesting Changes in Behavior", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 53-62, 1999.

[66] U. M. Fayyad, K.B. Irani, "On the Handling of Continuos-Valued Attributes in Decision Tree Generation", Machine Learning, 8(2), 1992.

[67] U. M. Fayyad, K. B. Irani, "The Attribute Selection Problem in Decision Tree Generation", In Proceedings of the Tenth National Conference on Artificial Intelligence AAAI-92, 104-110, AAAI Press, California, 1992.

[68] U. M. Fayyad, K. B. Irani, "Multi-Interval Discretization of Continuos-Valued attributes for Classification Learning", In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, San Francisco, Morgan Kaufmann, 1993.

[69] U. M. Fayyad and R. Uthurusamy, editors. "Workshop Knowledge Discovery in Databases", AAAI-94, Seattle, Washington, 1994.

[70] S. Gaffney, P. Smyth, "Trajectory Clustering with Mixtures of Regression Models", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 63-72, 1999.

[71] V. Ganti, J. Gehrke, R. Ramakrishnan, "CACTUS - Clustering Categorical Data Using Summaries", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 73-83, 1999.

[72] E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", NY: Addison-Wesley Publishing Co., 1989.

[73] I. Good, "The Estimation of Probabilities", Cambridge, Mass.: The MIT Press, 1965.

[74] R. Groth, "Data Mining: A Hands-on Approach for Business Professionals", Data Warehousing Institute Series, Prentice Hall, 1997.

[75] V. Guralnik, J. Srivastava, "Event Detection from Time Series Data", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 33-42, 1999.

[76] J. Han, S. Nishio, H. Kawano, "Knowledge Discovery in Object-Oriented and Active Databases", In Knowledge Building and Knowledge Sharing, ed. F. Fuchi and T. Yokoi, 221-230, Ohmsha, Ltd. and IOS Press, 1998.

[77] A. Hart, "The Role of Induction in Knowledge Elicitation", Expert Systems, 2, 24-28, 1985.

[78] A. Hart, "Knowledge Acquisition for Expert Systems", London: Kogan Page, 1996.

[79] M. Holsheimer and A. Siebes, "Data Mining: the Search for Knowledge in Databases", CS-9406, CWI, 1994.

[80] R. Howard, "Decision Analysis: Perspectives on Inference, Decision and Experimentation", Proceedings of the IEEE, 58: 632-643, 1970.

[81] K. Huang, Y. W. Lee, R. Y. Wang, "Quality Information and Knowledge", Prentice Hall, 1998.

[82] Y. Huang, P. S. Yu, "Adaptive Query Processing for Time-Series Data", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM,

282-286, 1999.

[83] P.J. Huber, "From large to huge: a statistician's reaction to KDD and DM", In: Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining, Newport Beach, CA, August 14-17, 304-308, 1997.

[84] E. Hunt, J. Marin, P. Stone, "Experiments in Induction", NY: Academic Press, 1966.

[85] S. Jain, D. Osherson, J. S. Royer, A. K. Sharma, "Systems That Learn: An Introduction to Learning Theory (Learning, Development and Conceptual Change)", MIT Press, Mass., 1999.

[86] M. Jarke, J. Koch, "Query Optimization in Database Systems", ACM Computer Surveys, 16:111-152, 1984.

[87] L. Kaufman and P.J. Rousseeuw, "Finding Groups in Data", J. Wiley, New York, 1990.

[88] M. G. Kelly, D. J. Hand, N. M. Adams, "The Impact of Changing Populations on Classifier Performance", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 367-371, 1999.

[89] M. Kersten and M. Holsheimer, "On the Symbiosis of a Data Mining Environment and a DBMS", CS-R9521, CWI, Amsterdam, 1994.

[90] W. Klösgen, "Efficient Discovery of Interesting Statements in Databases", The Journal of Intelligent Information Systems 4(1): 53-69, 1995.

[91] B. Larsen, C. Aone, "Fast and Effective Text Mining Using Linear-Time Document Clustering", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 16-22, 1999.

[92] W. Lee, S. J. Stolfo, K. W. Mok, "Mining in a Data-Flow Environment: Experience in Network Intrusion Detection", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 114-124, 1999.

[93] N. Lesh, M. J. Zaki, M. Ogihara, "Mining Features for Sequence Classification", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 342-346, 1999.

[94] R. P. Lippmann, "An Introduction to Computing with Neural Nets", IEEE ASSP

Magazine, April: 4-22, 1987.

[95] N. Liu and K. J. Cios, "Learning rules by integer linear programming." In: IEEE International Symposium on Industrial Electronics, Xian, China, May, 246-250, 1992.

[96] B. Liu, W. Hsu, Y. Ma, "Mining Association Rules with Multiple Minimum Supports", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 337-341, 1999.

[97] B. Liu, W. Hsu, Y. Ma, "Pruning and Summarizing the Discovered Associations", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 125-134, 1999.

[98] J. Q. Louie, T. Kraay, "Origami: A New Data Visualization Tool", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 405-408, 1999.

[99] D. W. Loveland, "Finding Critical Sets", Journal of Algorithms 8: 362-371, 1987.

[100] D. R. Mani, J. Drew, A. Betz, P. Datta, "Statistics and Data Mining Techniques for Lifetime Value Modeling", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 94-103, 1999.

[101] H. Mannila, D. Pavlov, P. Smyth, "Prediction with Local Patterns using Cross-Entropy", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 357-361, 1999.

[102] V. Megalooikonomou, C. Davatzikos, E. Herskovits, "Mining Lesion-Deficit Associations in a Brain Image Database", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 347-351, 1999.

[103] D. Meretakis, B. Wüthrich, "Extending Naïve Bayes Classifiers Using Long Itemsets", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 165-174, 1999.

[104] R. S. Michalski, R. L. Chilausky, "Learning by Being Told and Learning from Examples: An Experimental Comparison of the Two Methods on Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease

Diagnosis", Policy Analysis and Information Systems, 1980.

[105] J. A. Miller, "Subset Selection in Regression", NY: Chapman & Hall, 1990.

[106] J. Mingers, "An Empirical Comparison of Selection Measures for Decision-Tree Induction", Machine Learning 3, 4,319-342, 1989.

[107] J. Mingers, "An Empirical Comparison of Pruning Methods for Decision Tree Induction", Machine Learning 4, 2, 227-243, 1989.

[108] T. M. Mitchell, "Machine Learning", McGraw Hill, New York, 1997.

[109] F. Mosteller, J. W. Tukey, "Data Analysis and Regression", Reading Mass.: Addison-Wesley, 1997.

[110] B. Nag, P. Deshpande, D. J. DeWitt, "Using a Knowledge Cache for Interactive Discovery of Association Rules", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 244-253, 1999.

[111] G. Nakhaezadeh, A. Schnabl, "Development of multi-criteria metrics for evaluation of data mining algorithms", In: Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining, Newport Beach, CA, August 14-17, 37-42, 1997.

[112] N. J. Nilsson, "Learning Machines", New York, McGraw Hill, 1965.

[113] T. Oates, "Identifying Distinctive Subsequences in Multivariate Time Series by Clustering", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 322-326, 1999.

[114] J. J. Oliver, "Decision Graphs-An Extension of Decision Trees", In Proceedings of Fourth International Workshop on AI and Statistics, 343-350, 1993.

[115] W. Pedrycz, "Fuzzy Sets Engineering", CRC Press, Boca Raton, Florida, 1995.

[116] W. Pedrycz, "Fuzzy Multimodels", IEEE Trans. On Fuzzy Systems, 4, 139-148, 1996.

[117] W. Pedrycz, "Data Mining and Knowledge Discovery through Fuzzy Neurocomputing", Fuzzy Sets & Systems, 3, 279-290, 1998.

[118] W. Pedrycz, Z. A. Sosnowski, "Designing Decision Trees with the Use of Fuzzy Granulation", Alberta, 1999.

[119] D. Pelleg, A. Moore, "Accelerating Exact k-means Algorithms with Geometric Reasoning", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 277-281, 1999.

[120] G. Piatetsky-Shapiro and W. J. Frawley, editors. "Knowledge Discovery in Databases", AAAI Press, Menlo Park, California, 1991.

[121] G. Piatetsky-Shapiro, B. M. Masand, " Estimating Campaign Benefits and Modeling Lift", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 185-193, 1999.

[122] G. Plotkin, "A Note on Inductive Generalization", In Machine Intelligence 5, eds. B. Meltzer and D. Michie, 153-163, Edinburgh, UK: Edinburgh University Press, 1969.

[123] W. J. E. Potts, "Generalized Additive Neural Networks", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 194-200, 1999.

[124] F. J. Provost, D. Jensen, T. Oates, "Efficient Progressive Sampling", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 23-32, 1999.

[125] D. Pyle, "Data Preparation for Data Mining", Morgan Kaufmann, California, 1999.

[126] J. R. Quinlan, "Induction of Decision Trees", Machine Learning 1, 1, 81-106, 1986.

[127] J. R. Quinlan, "Generating Production Rules from Decision Trees", Proceedings of the Tenth International Joint Conference on Artificial Intelligence, 304-307, San Mateo, California, 1987.

[128] J. R. Quinlan, "Simplifying Decision Trees", Int. J. of Man-Machine Studies, 27, 221-234, 1987.

[129] J. R. Quinlan, "Decision Trees and Multi-Valued Attributes", Machine Intelligence 11, 305-318, Oxford, UK, 1988.

[130] J. R. Quinlan, "Unknown Attribute Values in Induction", Proceedings of the Sixth International Machine Learning Workshop, 164-168, San Mateo, California, 1989.

[131] J. R. Quinlan, "Decision Trees and Decision Making", IEEE Transaction Systems, Man and Cybernetics 20, 2, 339-346,1990.

[132] J. R. Quinlan, "Learning Logical Definitions from Relations", Machine Learning 5, 3, 239-266, 1990.

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

[134] N. J. Radcliffe and P. D. Surry, "Co-operation through Hierarchical Competition in Genetic Data Mining ", CS-R9430, CWI, Amsterdam, 1994.

[135] R. L. Rivest, "Learning Decision Lists", Machine Learning, 2, 229-246, 1987.

[136] S. Rogers, P. Langley, C. Wilson, "Mining GPS Data to Augment Road Models", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 104-113, 1999.

[137] S. Rosset, U. Murad, E. Neumann, Y. Idan, G. Pinkas, "Discovery of Fraud Rules for Telecommunications - Challenges and Solutions", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 409-413, 1999.

[138] RRSP data set provided courtesy of Faneuil ISG Inc., Winnipeg, Manitoba, Canada, 1998.

[139] S. Sahar, " Interestingness via What is Not Interesting", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 332-336, 1999.

[140] G. Schwarz, "Estimating the Dimension of a Model", Annals of Statistics, 6:461-464, 1978.

[141] J. Shanmugasundaram, U. M. Fayyad, P. S. Bradley, "Compressed Data Cubes for OLAP Aggregate Query Approximation on Continuous Dimensions", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 223-232, 1999.

[142] W. M. Shen, "Discovering Regularities from Knowledge Bases", International Journal of Intelligent Systems, 7(7), 623-636, 1992.

[143] S. T. Shenoy, Z. M. Ozsoyoglu, "Design and Implementation of a Semantic Query Optimizer", IEEE Transactions on Knowledge and Data Engineering I (3): 344-361, 1989.

[144] A. Siebes, "Homogenous Discoveries Contain no Surprises: Inferring Risk-profiles from Large Databases", CS-R9430, CWI, Amsterdam, 1992.

[145] M. D. Siegel, "Automatic Rule Derivation for Semantic Query Optimizer", In proceedings of the Second International Conference on Expert Database Systems, 371-385, ed. L. Kerschberg, VA, George Mason Foundation, 1988.

[146] D. J. Spiegelhalter, D. Michie, C.C. Taylor, "Neural and Statistical Classification", Machine Learning, NY: Ellis Horwood, 1994.

[147] E. Simoudis, B. Livezey and R. Kerber, "Integrating Inductive and Deductive Reasoning for Data Mining", In: Advances in Knowledge Discovery and Data Mining (U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy, editors.), American Association for Artificial Intelligence, California, 353-373, 1996.

[148] M. Shewhart, M. Wasson, "Monitoring a Newsfeed for Hot Topics", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 402-404, 1999.

[149] P. Spirtes, C. Glymour, R. Scheines, "Causality, Prediction and Search", Berlin, Springer-Verlag, 1993.

[150] N. A. Syed, H. Liu, K. K. Sung, "Handling Concept Drifts in Incremental Learning with Support Vector Machines", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 317-321, 1999.

[151] N. A. Syed, H. Liu, K. K. Sung, "A Study of Support Vectors on Model Independent Example Selection", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 272-276, 1999.

[152] H. Toivonen, "Sampling large databases for association rules". In: Proc. 22nd Int. Conf. on Very Large Databases, 134-145, 46, 1996.

[153] J. W. Tukey, "Exploratory Data Analysis", Reading, Mass.: Addison-Wesley, 1977.

[154] A. K. H. Tung, H. Lu, J. Han, L. Feng, "Breaking the Barrier of Transactions: Mining Inter-Transaction Association Rules", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 297-301, 1999.

[155] J. Ullman, "Principles of Database and Knowledge Base Systems", Volume I, Rockville, Mass.: Computer Science Press, 1988.

[156] University of Irvine data sets being used are from Internet site ftp://ftp.ics.uci.edu, A. Janosi-Hungarian Institute of Cardiology, W. Steinbrunn-University Hospital in Zurich, M. Pfisterer- University Hospital in Basel, R. Detrano-V. A. Medical Center, 1997.

[157] S. Urman, "Oracle8 PL/SQL Programming", Oracle Press, Osborne McGraw Hill, California, 1997.

[158] P. E. Utgoff, "Shift of Bias for Inductive Learning", Machine Learning: An Artificial Intelligence Approach, Vol. II, Morgan Kaufmann, Los Altos, California, 1986.

[159] P. E. Utgoff, "Incremental Induction of Decision Trees", Machine Learning 4, 2, 161-186, 1989.

[160] P. E. Utgoff and C. E. Brodley, "Linear Machine Decision Trees", COINS Technical Report 91-10, University of Massachusetts, Amherst, Massachusetts, 1991.

[161] V. Vapnik, "Estimation of Dependencies Based on Empirical Data", NY: Springer-Verlag, 1982.

[162] J. T. Wang, X. Wang, K. Lin, D. Shasha, B. A. Shapiro, K. Zhang, "Evaluating a Class of Distance-Mapping Algorithms for Data Mining and Clustering", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 307-311, 1999.

[163] S. Weiss, R. Galen, P. Tadepalli, "Maximizing the Predictive Value of Production Rules", Artificial Intelligence 45: 47-71, 1990.

[164] S. M. Weiss, N. Indurkhya, "Predictive Data Mining: A Practical Guide", Academic Press, Morgan Kaufmann, 1997.

[165] J. Wijsen, R. T. Ng, T. Calders, "Discovering Roll-Up Dependencies", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 213-222, 1999.

[166] J. Wirth and J. Catlett, "Experiments on the Costs and Benefits of Windowing in ID3", Proceedings of the Fifth International Conference on Machine Learning, 87-99, San Mateo, California, 1988.

[167] C. Westphal, T. Blaxton, "Data Mining Solutions", John Wiley & Sons, 1998.

[168] J. Wnek and R. S. Michalski, "Conceptual Transition from Logic to Arithmetic in Concept Learning", Reports of Machine Learning and Inference Laboratory, MLI 94-7, Center for MLI, George Mason University, December 1994.

[169] R. R. Yager, Measuring tranquility and anxiety in decision making: An application of fuzzy sets. Int. J. Gen. Syst., 8, 139-146, 1982.

[170] R. R. Yager, "Entropy and specificity in a mathematical theory of evidence". Int. J. Gen. Syst., 9, 249-260, 1983.

[171] K. Yoda, T. Fukuda, Y. Morimoto, "Computing optimized rectilinear regions for association rules", In: Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining, Newport Beach, CA, August 14-17, 96-103, 1997.

[172] L. A. Zadeh, "Fuzzy Sets and Information Granularity", In: Advances in Fuzzy Set Theory and Applications (M.M. Gupta, R. K. Ragade, R. R. Yager, editors.), North Holland, Amsterdam, 3-18, 1979.

[173] S. B. Zdonik, M. Hammer, "Knowledge-based Query Processing", In Proceedings of the Sixth VLDB Conference, D.C., 137-146, 1980.

[174] T. Zhang, R. Ramakrishnan, M. Livny, "Fast Density Estimation Using CF-Kernel for Very Large Databases", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, CA, USA. ACM, 312-316, 1999.

[175] J. Zytkow, "Automated discovery of empirical laws", Fundamenta Informaticae, 27, 299-318, 1996.

[176] J. Zytkow, R. Zembowicz, "Database Exploration in Search of Regularities", Journal of Intelligent Information Systems 2: 39-81, 1993.

APPENDIX

Discretization values-RRSP data set

Province:

1 is AB
2 is BC
3 is MB
4 is NB
5 is NS
6 is Nfld
7 is ON
8 is PEI
9 is PQ
10 is SK
0 is 11 (not classified)

Community:

1 is A_FARM
2 is B_RURAL
3 is C_CITY_TWN
4 is D_METRO
5 is E_NA (not available)

Family:

1 is Couples_no_kids
2 is Couples_w_kids
3 is Lone_parent
4 is Non_Family_F
5 is Non_Family_M

Income:

7 is <20
1 is 20to30
2 is 30to40
3 is 40to50
4 is 50to60
5 is 60to80
6 is 80+

Tenure:

1 is Own
2 is Own_Mortgaged
3 is Rent

Sex_ref:

1 is F
2 is M

Educ_ref:

1 is A_LT_9 (less than 9 grades)
2 is B_HS (high school)
3 is C_COLLEGE
4 is D_UNIV
5 is E_NS (not available)

Occ_ref:

1 is A_NS (not available)
2 is B_not_wkg
3 is C_blue
4 is D_other
5 is E_sales_sr (supervisor)
6 is E_sales_sv (service)
7 is F_prof_tec
8 is G_mgr_adm

Rrsp_buyer:

1 is N
2 is Y

Investor:

1 is N
2 is Y

Double_inc_earner:

1 is N
2 is NA
3 is Y