HEARTFAID

D29 – Models and methods for knowledge discovery

Submission date: 15/03/2008
Due date of document: 01/02/2008

HEARTFAID

A KNOWLEDGE BASED PLATFORM OF SERVICES FOR SUPPORTING MEDICAL-CLINICAL MANAGEMENT OF THE HEART FAILURE WITHIN THE ELDERLY POPULATION

Project summary	
Project acronym:	HEARTFAID
Project identifier:	IST – 2005 – 027107
Duration of the Project:	01/02/2006 – 31/01/2009
Project Co-ordinator:	UNICAL University of Calabria (Italy)
Thematic Priority:	Information Society Technology
Instrument:	Specific Targeted Research or Innovation Project

Consortium

- UNICAL - Università della Calabria (Italy)
- UNICZ - Università degli studi Magna Graecia di Catanzaro (Italy)
- UNIMIB - Università degli studi di Milano Bicocca (Italy)
- JUMC - Jagiellonian University Medical College (Poland)
- VMWS - Virtual Medical World Solutions Ltd (United Kingdom)
- FORTHNET S. A.- Hellenic Telecommunications and Telematic Applications Company S. A. (Greece)
- SYNAP - Synapsis s.r.l. (Italy)
- CNR - Consiglio Nazionale delle Ricerche (Italy)
- FORTH - Foundation for Research and Technology Hellas (Greece)
- RBI - Rudjer Boskovic Institute (Croatia)
- AUXOL - Istituto Auxologico Italiano (Italy)

D29 – Models and methods for knowledge discovery

Document summary	
Document title:	D29 – Models and methods for knowledge discovery
Document classification:	Derivable D29
Dissemination level:	PU
Submission date:	15 March 2008
Due date:	01 February 2008
Authors:	Domenico Conforti, Vincenzo Lagani, Antonio Candelieri, Rosita Guido - UNICAL Dragan Gamberger, Tomislav Šmuc, Mislav Malenica, Matko Bošnjak, Rajko Horvat, Marin Prcela – RBI Ioannis Tsamardinos, Franco Chiarugi, Eirini Koytaki – FORTH
Work package:	WP4 – Knowledge representation, discovery and management
Report version:	1.1

Short description
The deliverable presents the knowledge discovery methodology relevant for the Heartfaid platform and presents details of the implemented knowledge discovery processes on the collected heart failure datasets.

Change record		
Version number	Changes	Release date
0.0	first draft of the table of contents	21/11/2007
0.1	first draft of the document	01/01/2008
0.2	second draft of the document	25/01/2008
0.3	third draft of the document	08/02/2008
0.4	fourth draft of the document	03/03/2008
1.0	first complete deliverable	11/03/2008
1.1	final document version	15/03/2008

Table of contents

1. Executive Summary

The document presents the results related to the analysis and development of the knowledge discovery (KD) methodology and its application on the available heart failure datasets. KD methodology is important for the Heartfaid project from two aspects: as a tool that can help in the development of the knowledge necessary for the integration into the platform's knowledge base and as a tool that should enable understanding of data collected by the platform. Heartfaid platform is challenging in the sense that it requires *closed loop integration* of KD processes into a complex knowledge based environment: at the same time we need results of the KD process in order to enable decision support functionality of the platform and platform's data collection functionality in order to collect data representing input for this process. Additionally, we should implement the methodology that can be integrated into the KD Web service.

The presentation begins with the overview of the KD process characteristic for medical applications. Special attention has been given to the problems of missing data, unbalanced datasets, overfitting prevention, model evaluation and appropriate result interpretation. The central part is presentation and critical evaluation of very different KD techniques. Very shortly we presented classical techniques like decision trees, radial basis function networks and Bayesian learning. The attention is devoted to most modern methodologies: random forest learning, subgroup discovery, and support vector machines. Random forest learning is relevant because it will be used in the on-line KD Web service due to its outstanding non-overfitting characteristic, subgroup discovery deserved special attention because of its applicability in intelligent and descriptive data analysis, while support vector machines are currently the best approach for building predictive models from complex datasets.

All methodologies described in the deliverable have been tested on a few heart failure datasets. We have experimented with retrospective data, data collected by the platform, and data specially collected for decompensation modelling. Some of the most relevant and medically interesting results are presented in the last part of the document. By application of decision trees and decision lists, we have developed a few rules for the detection of patient's decompensation. Random forest methodology have been used for the analysis of the data collected by the platform, both for the detection of some patient clusters and for characterization of patients with properties like diastolic dysfunction or dyslipidemia. Special attention has been devoted to the heart failure severity analysis and prediction. It turned to be a very difficult task although on our disposal was among others also a dataset with more than 17,000 heart failure patients. The results demonstrated the relevance and usefulness of the NYHA classification, but additionally we managed to detect some novel and potentially significant coexisting factors like third tone, atrial fibrillation, and left branch block. Finally, based on such factors we demonstrated how KD methodology may be used for the definition of a novel HF severity scale.

2. List of abbreviations

TERM	DEFINITION
ANMCO	Associazione Nazionale Medici Cardiologi Ospedalieri
AUC	Area Under Curve
CPT	Conditional Probability Tree
DB	Data Base
DBMS	Data Base Management System
DL	Decision List (algorithm or methodology)
DM	Data Mining
DSS	Decision Support System
DT	Decision Tree (algorithm or methodology)
DW	Data Warehouse
eCRF	Electronic Case Report Form
GUI	Graphical User Interface
HF	Heart Failure
HFP	HEARTFAID Platform
i.i.d.	independent and identically distributed
KD	Knowledge Discovery
KDD	Knowledge Discovery in Databases
MDS	MultiDimensional Scaling
ML	Machine Learning
NYHA	New York Heart Association Functional Classification
OOB	Out-Of-Bag error estimate
OSH	Optimal Separating Hyperplane
PARF	PArallel Random Forests
RBF	Radial Basis Function
RF	Random Forests (algorithm or methodology)
ROC	Receiver Operating Characteristic
RSF	Random Survival Forests
SD	Subgroup Discovery (algorithm or methodology)
SDP	Semidefinite Programming
SVM	Support Vector Machine (algorithm or methodology)

3. Introduction

Knowledge discovery (KD) is an area of computer science concerned with the discovery of models, patterns and other regularities from data. In many aspects it overlaps with classical statistical data analysis; however, the unique added value is systematic application of machine learning algorithms for hypotheses generation from large datasets. Large potentials of these algorithms in intelligent data analysis are already recognized in many domains.

Models obtained by machine learning algorithms serve two different purposes: discrimination and characterization. This means actionability in terms of determining class membership of individual non-classified instances and in the sense of uncovering the characteristic properties of instances in different populations. Both can have high applicability in human understanding of the domain and decision support purposes.

Knowledge discovery is a vivid research area for two reasons a) development of effective machine learning algorithms able to construct high quality models from data and b) recognition of techniques specifically appropriate for data analysis tasks. Presentation of the state of the art in the KD methodology appropriate for the application inside Heartfaid platform, together with the illustration of its application on a few datasets related with heart failure is the main topic of this deliverable.

3.1. KD in Heartfaid project

Knowledge discovery is important for the Heartfaid project from two aspects: as a tool that can help in the development of the knowledge necessary for the integration into the platform's knowledge base and as a tool that should enable understanding of data collected by the platform.

Although there are already many KD applications in medical domains, and a few among them enabled induction of relevant novel knowledge, KD application inside Heartfaid platform is challenging in the sense that it requires *integration of KD processes* into a complex knowledge based environment. It means that at the same time we need results of the KD process in order to enable decision support functionality of the platform and platform's data collection functionality in order to collect data representing input for this process. Such closed loop application of the KD methodology is also uncommon outside medical domains and it is a real scientific challenge. Its solution is essential for building any real knowledge based and intelligent data manipulation platform.

Our work starts from a broad range of already developed machine learning tools. The special care is devoted to those approaches and their modifications appropriate for delivering models useful for decision support purposes and medical expert understanding of collected data. The problem of the KD process

closed loop integration is solved by breaking it into a few almost independent tasks:

- data collection processes aimed at preparation of the data necessary for KD (D21)
- construction of the knowledge base which in the first iteration contains only existing medical knowledge related to the heart failure (D22)
- preparation of appropriate KD tools (this deliverable)
- building a Web service for the KD applications (D31)

These tasks together constitute necessary steps for the knowledge discovery process. However, it must be noted that in the applications like Heartfaid it is not allowed, and is practically impossible, to realize the closed loop that will automatically update the knowledge base with the KD results. Only medical experts can allow inclusion of some KD results into the knowledge base. The iterative loop must also include humans. Nevertheless, that means that the results obtained by KD tools should be either interpretable by humans or there should be a possibility for their independent verification.

The most difficult decision making problems for the platform are the situations requiring models based on a series of clinically collected data, especially in combination with continuously monitored data. These are typical situations for which explicit medical knowledge does not exist in medical guidelines. Although medical experts are able to effectively solve such situations, their tacit knowledge cannot be directly coded into the knowledge base so that the only solution is application of KD tools. The problem is that there is no retrospective datasets that could be used as input for the KD process. Only platforms like Heartfaid are able to collect such complex datasets. This implies that, besides in supporting the disease management, perhaps equally relevant role of the platform is to collect data that will enable formulation of novel medical knowledge.

The closed loop functionality of the platform's KD process cannot be implemented before the platform is practically realized and collects patient data for some relevant time period. Because of that, real implementation of the results of the KD process can be expected iteratively in the normal life circle of the platform. The goal of this deliverable, and other KD related deliverables, is to prepare the methodology that will be used in this iterative process. Most of it will be used off-line in a collaboration of technical and medical experts with intention to achieve optimal quality of induced models by experimenting with different tools on various example and attribute subsets. A part of the methodology will also be available for on-line experiments by medical personnel. The latter will operate on already predefined data forms with significantly reduced set of available options.

3.1.1. Goals of intelligent data analysis

Based on the deliverable D5 and data and domain understanding presented in D21 five different goals for the KD process have already been recognized.

I. Find the single biomedical parameter or the combination of parameters that reliably define different degrees of HF severity and that may increase the diagnostic capability for HF.

II. Find the single biomedical parameter or the combination of parameters that identify that the subject is in stable, improving, rapidly worsening, or slowly worsening condition.

III. Test if the quality of results for both I and II can increase in the case of continuous data monitoring of relevant parameters.

IV. Among continuously monitoring data, identify precipitating and exacerbating factors of decompensated CHF.

V. Evaluate consequences of applied therapies.

For goals I and II baseline evaluation data can be sufficient but better results can be expected if data from more visits are included. In the latter case the data preparation process should include data flattening of short sequences described in Section 8.2.3 of D21. The prerequisite to accomplish goals III and IV is availability of the continuously monitored data. Continuously monitored data should be prepared by data flattening of long sequences described in Section 8.2.2 of D21. Goal V requires special data preparation procedure which includes information both before and after the therapy has been changed. The datasets may include both clinical and continuously monitored data and the used time window may depend on the type of the therapy change. In this situation expert knowledge is also very valuable for the selection of most relevant attributes that should enter the KD process.

3.1.2. Heartfaid specific requirements on KD methodology

The goals of the KD process require different methodological tools.

The first are clustering tools able to recognize patient clusters with similar properties. These tools are necessary for goal I in order to detect groups of patients with similar degrees of HF severity. Currently NYHA classification is used for this purpose and in goal I we have to test if NYHA class can be reliably recognized from available patient data. Additionally, we have to test if there is a better way to describe patient status. For this task we need clustering tools to identify homogeneous patient subpopulations.

The second are classification tools able to build reliable classifiers for some given patient classes. The patient classes can be very different (i.e. patients in danger of decompensation, patients in a stable condition or patients in NYHA classes III and IV). As a result of that, these tools are applicable for all goals I-V. The only problem is that high quality models are typically not in a symbolic form. It means that that the classifiers a) cannot be verified by medical experts and some other

(experimental) verification of the model quality is necessary and b) the models cannot be implemented as part of the procedural ontology. The latter property implies that they must be included into the decision support system as separate building blocks in the model base. These drawbacks are significant and classification tools should be used only if other approaches cannot give satisfactory results.

The third are feature or attribute selection tools which are able to extract a set of most relevant patient characteristics related to some target concept. These are actually the classification tools which use the constructed models in order to determine which attributes are most useful for distinguishing the patient classes. These tools are applicable for all goals I-V. This type of output is very understandable by humans and it is the most appreciated output of descriptive induction. Especially because, unlike the statistical approaches, these tools determine the feature relevancy in the context of all available attributes. In spite of modest statistical relevance, an attribute can happen to be very relevant if it is useful in the cooperation with other relevant attributes. Additionally, some of the feature selection tools explicitly specify which property (value or range) is actually relevant for the target concept. In this case they may suggest optimal decision points based on these attributes. The only problem is that this information cannot be directly used for classification because it is not in the form of rules but in the form of a list of attributes or features. In some cases medical experts can, after understanding the domain, try to construct rules themselves from these attributes but then such models require additional verification.

The last are descriptive induction tools producing rules that can be interpreted by humans. They are applicable for all types of classification problems. Typically rules are built of only two to four features connected by logical *AND* operations. Although prediction quality of such rules is often lower than the prediction quality of models built by classification tools, it may be appropriate for some decision support purposes. The main advantage of such rules is that they explicitly point out relevance of coexisting properties that should be satisfied in order for a patient to be classified into an associated class. This is very useful for human understanding of complex medical concepts in descriptive induction and it gives additional information to the one obtained by feature and attribute selection tools. If their classification quality is satisfactory, after expert evaluation those rules can also be integrated directly into ontological form of the knowledge base. In this respect inductive systems with variable generality of induced rules are especially interesting because they enable construction of so called confirmation rules that are especially appropriate for inclusion in decision support systems.

3.1.3. Methodological foundations for on-line KD service

The main task of the KD methodology in the Heartfaid project is development of knowledge that can be included into the platform's knowledge base. Besides that it should enable analysis of the data collected by the platform. This should also be partially offered to the platform's users as an on-line Web service. Although the central issues in this service are on-line data extraction, transformation and

preparation, as well as effective communication with end-users, for its realization we need KD methodology appropriate for such imbedded tasks.

Specific requirements for the tools that can be used in this service are:

a) time efficiency (long execution times are not acceptable in an on-line service);

b) relatively small set of free parameters that should be adjusted by users (the service should be as simple as possible);

c) results must be in a human understandable form.

These requirements eliminate classification tools that do not produce output in human understandable form (such as artificial neural networks), as well as clustering tools because their results are not directly interpretable and their use requires additional processing and therefore significant methodological expertise. From the set of attribute and feature selection tools as well as descriptive induction tools especially interesting are those that are relatively fast in execution.

Typically KD tools are used by technical experts and they tend to have very complicated forms for the presentation of the results. For the Web service which is supposed to be used by medical people it is decisive to use tools able to communicate interpretable results efficiently over the Web interface. This is still a non-standard requirement for publicly available tools.

Last but not the least is the problem of the software licenses. Most of the publicly available software is free for educational and scientific purposes. It means that off-line data analyses of the data collected by the platform can be done without restrictions. But KD tools under open source licenses that enable imbedding the software into the platform are rare.

3.2. Deliverable organization

Section 4 defines the KD process and describe its main steps. Special attention is devoted to properties characteristic for the application of KD in medical applications. Data preparation has been in more detail discussed already in the deliverable D21. Here we have devoted our attention mainly to the handling of missing data and dataset balancing. They represent data pre-processing steps that can significantly influence the quality of obtained results. Section 4 ends with very detailed presentation of the techniques applicable for the evaluation of obtained results and their interpretation that must be combined with existing medical knowledge and understanding of the conditions in which the data has been collected.

Section 5 is the central methodological part of the deliverable. It presents different machine learning approaches applicable for the analysis of the data related to the heart failure, either those collected by the platform or retrospective data. The goals are the construction of the knowledge that might be integrated into the Heartfaid knowledge base or the construction of models that can be directly integrated into the Heartfaid decision support subsystem. The presentation is

organized so that we start with the decision tree algorithm in Sections 5.1 and continue with random forest methodology (Section 5.2) and survival analysis based on random forests (Section 5.3). We continue with approaches that enable human understanding of the induced concepts (decision lists and subgroup discovery in Sections 5.4 and 5.5, respectively). We conclude with approaches constructing complex models that cannot be directly evaluated by humans but that typically have excellent prediction properties in Sections 5.6-5.8. With more details we have presented algorithms that are prepared for the integration into the Web-based KD service (random forest learning in Section 5.2, subgroup discovery in Section 5.5, and support vector machines in Section 5.6). Special attention is devoted to the presentation of the survival analysis methodology in Section 5.3 which is typical for the analysis of some medical data types.

The results of the application of the described KD tools on a few different datasets are the topic of Section 6. This section begins with the presentation of the datasets that have been available for the analysis: data already collected by the platform (eCRF dataset presented in Section 6.1.1), dataset specially collected by UNICZ for the purposes of the Heartfaid project about continuously monitored patients and their decompensation events (Section 6.1.2), retrospective dataset (ANMCO dataset presented in Section 6.1.3), and a publicly available genomic dataset (Section 6.1.4). The main part is presentation of very different results obtained on these datasets by tools described in Section 5. The results are rules for decompensation alarming (Section 6.2.1) and descriptive analysis of data collected by the platform (Section 6.2.2). Special attention has been devoted to the heart failure severity analysis and prediction of events like hospitalization or patient death based on the large available retrospective dataset. In Section 6.2.3 are presented results of the NYHA class description by other patient properties and an approach to the construction of a novel HF severity scale. In Sections 6.2.4 and 6.2.5 different KD approaches for the construction of prognostic models are analysed while in Section 6.2.6 results obtained for the HF diagnosis from the transcriptional genomics dataset are presented.

4. KD process for medical applications

Knowledge discovery is a very fast developing field of artificial intelligence. Practically there is no data analysis domain in which it cannot be applied. In this section we will concentrate on the presentation of the KD process, specifically stating those steps that are relevant for the medical applications and which are expected to be decisive for the successful KD implementation in the Heartfaid platform. Figure 4-1 depicts the general circle of KD process, starting with problem and data understanding, followed by the data preparation activity, which is again followed by the central part representing actual model generation. The process ends by model interpretation, evaluation, and deployment.

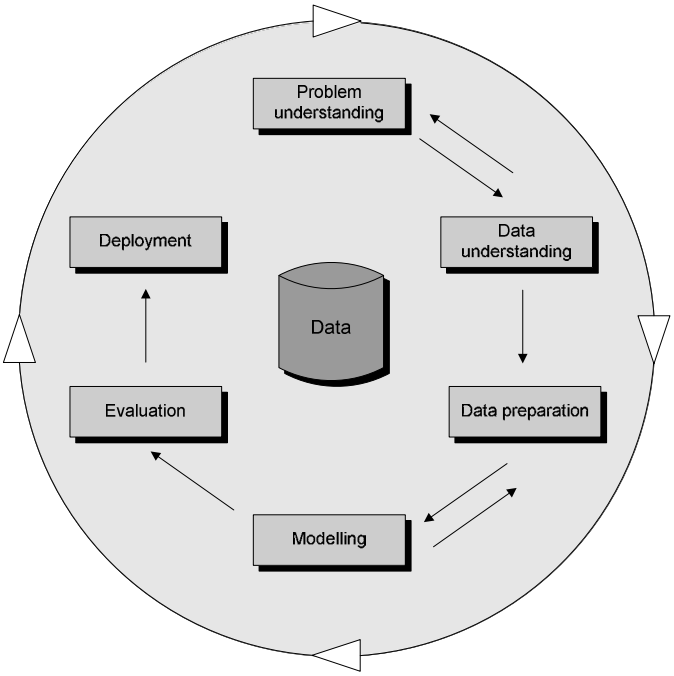

Figure 4-1. Knowledge discovery process

The complete KD process is iterative and data preparation, model generation, and model evaluation are repeated as long as necessary to achieve the goals. Often different KD tools are used and combined so that output of one tool is used to determine the data input for the next iteration and application of an another tool. In this way, the possible variety of approaches that can be implemented to analyze the same data is enormous. The selected approach, as well as its results, significantly depend on the experience of the expert guiding the process. KD

methodology cannot suggest the optimal approach; the optimal solution is defined by the preferences of the medical experts evaluating novelty, actionability, prediction quality, or prediction reliability of the constructed classifiers, as well as descriptive relevance or ethical acceptability of detected properties.

The experiments with different KD tools on many domains have demonstrated that there is no single ideal approach for all situations. There has also been some effort in the direction of systematization or standardization of the KD process. The most well known is the CRISP approach presented in Figure 4-1 which has been primarily developed for business applications.

In medical applications there are two main approaches for the organization of the KD process. The first, that can be also called *expert guided approach,* is the one in which a medical domain expert has the central role and uses KD tools as general purpose black boxes. The approach is in line with the *active mining concept* developed also for other types of data analysis. Its major advantage is that available background knowledge is smoothly integrated into the KD process by the active role that the expert has in directing the process. This approach will be implemented in the Heartfaid on-line service.

The other approach that can be called *insightful data analysis*, is based on systematic generation of a large set of solutions with very different properties. KD experts have the central role in this approach. Their work needs well defined medical goals and experience with medical data analysis and decision support applications. After the interpretation and potential visualization of all aspects of obtained results, medical experts evaluate the results and select only those appropriate for implementation and/or publication. The approach is appropriate for off-line data analysis and integrated projects like Heartfaid which include both technical and medical experts. Its main advantage is the possibility to effectively integrate different, sometimes also very specific KD methodologies while the evaluation of the results can be done by consensus of a group of medical experts.

4.1. Types of medical KD tasks or problems

As depicted in Figure 4-1, a process of knowledge discovery starts with the definition of the problem we want to solve. In medical domain, in general, predictive models are built for two general problems or tasks: diagnosis and prognosis problems [1].

Diagnosis essentially stands for identification of a disease or disorder. Diagnosis usually requires availability of information describing patient properties related to medical history, physical examination results, results of specific tests and diagnostic procedures (e.g. blood analysis, diagnostic imaging) in order to confirm it. A list of possible causes is developed and subsequently narrowed down by further tests that eliminate or support specific possibilities.

The term *prognosis,* in medical practice, is defined as "*the prediction of the course and possible outcome(s) of the disease in future*". There are potentially a number of decisions or actions in the treatment of the patient (either medication

change, changing habits, surgical treatments) which influence the changes in the course and outcomes of the disease. Prognosis actually changes after each decision/action/change made either by health care provider or patient herself.

In any case, the prediction of "future" events relative to the available information and actions prior or at the time of prediction is the essential aim of the prognostic modelling. It is the time dimension and its inclusion in prognostic models that represents the key difference between the concepts of prognosis and diagnosis. Prognostic models in clinical practice are viewed as decision tools which are essentially conditioned on diagnostic and treatment information. Another important practical observation is that the prognosis is usually expressed in terms of a single patient although general course and outcomes of the disease is something that is systematically specified in e.g. clinical guidelines or medical literature.

There are number of practical uses of prognostic models such as [2]:

a) selecting appropriate tests and therapies in individual patient management including supporting decisions on withholding or withdrawing therapy

b) serving as guides in health-care policies

c) determining study eligibility of patients for new treatments

d) defining inclusion criteria for clinical trials to control for variation in prognosis

4.2. Data preparation

KD methodology and machine learning tools, regardless how advanced and complex they are, cannot be successful in detecting meaningful relations if data analyzed by these tools does not implicitly contain the required information. The principal role of data collection and data warehousing is to ensure that potentially relevant data are stored in the database. In order not to miss something relevant, the generally accepted approach is to collect everything that can be collected at a reasonable cost and is potentially relevant for learning about the target concept. The motivation for such approach is that if data analysis demonstrates that something relevant has not been collected, typically there is no possibility to collect the missing data later. The result is a large amount of data practically limited only by technical constraints, like availability of the equipment necessary for data recording, effort and cost of medical staff necessary to collect the data, and ethical reasons.

The first data preparation step is data extraction from different sources (e.g. databases, files, etc.) and their transformation into the form appropriate for application of KD tools. This step significantly depends on the goals of the KD process. Deliverable D21, in its Section 8 on functional specification of the data preparation process, defines the goals of the KD process and presents the methodology for the data extraction and data transformation. Special attention is devoted to the problem of the transformation of short and long time sequences

into patients attributes that can be used as input for KD tools. At this stage no instance and attribute selection mechanism are foreseen. D21 in its appendix lists the attributes that are available in the platform.

The first data preparation step is performed only once. It ends with potentially huge flat tables including all available patient information. Different KD tasks can be defined from each of these tables. The second data preparation step, depending on the current task, selects attributes and instances appropriate for the analysis. In contrast to the first, the second step has to be repeated in different iterations of the KD process depending on the results and their interpretation in previous experiments. Instances are typically selected by previously generated models while attributes are selected based on medical goals and expected form of the results. The results of this step are significantly reduced flat tables.

Theoretically speaking, the tables prepared by the second data preparation step could be directly used as an input for KD tools. However, this is the right moment to perform data cleansing, elimination of irrelevant attributes, elimination of unknown (missing) attribute values as well as elimination of noisy examples. Although most KD tools can also solve these tasks during the induction process, it is suggested to do this in preprocessing in order to increase the expected quality of the KD results. The reasons for this are that in this way noisy or incomplete data cannot influence the induction process and that the users can explicitly control each of these steps. Details of noise detection, data cleansing, and attribute selection processes are described in Section 7 of D21. The third data preparation step is also the right place for data transformations required by specific properties of KD tools whose application will follow. For example this means that either all attributes should be numeric or categorical, or special handling of missing attribute values is necessary.

4.2.1. Missing values handling

In very large datasets with a small number of missing values the problem can be solved by elimination of problematic examples. For small datasets, this is not an option. This problem can be dealt with in the data preprocessing at least in the following five different ways: by deleting attributes, by deleting examples, by imputing values, by categorizing attributes, and by introducing missing-value indicator variables. Every method has consequences which can effect the quality of the dataset and therefore the quality of induced models. The analysis that follows demonstrates that none of the approaches is ideal. The conclusion is that the problem can be better handled inside the induction process and when missing values handling is necessary in data preprocessing it should be done with precaution.

Deleting attributes with high percentage of missing values seems, at first, like an obvious choice. We are not affecting the population in any way; we are just switching to experiment with smaller number of measured attributes. However, we are clearly loosing potentially valuable information. We can easily imagine a situation in which attribute values have been inconsistently collected and as a result we have an extremely noisy attribute. Excluding such an attribute is a

reasonable thing to do. Figure 4-2 illustrates this procedure. We have two classes of examples: squares and circles. Black squares or black circles in left graph denote examples for which we do not know the value of the attribute 1. Right graph shows the situation when we have eliminated the attribute 1 from the dataset. It is clear that although we can now work with all examples, attribute 1 carried valuable information that is now lost.

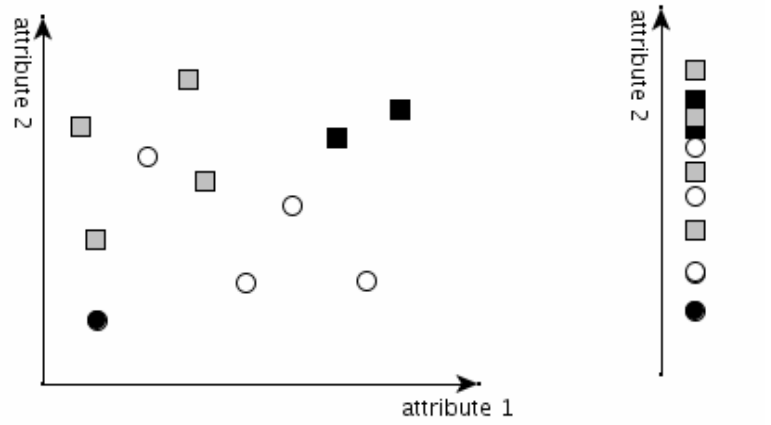

Figure 4-2. Black colour denotes examples with missing Attribute 1 value. Left graph shows an ideal situations with no missing values, right graph shows a situation when Attribute 1 is deleted from the dataset

When we realize that with deleting attributes we lose valuable information, next logical attempt would be to delete examples with missing values. If we delete all examples with missing values and at the end still have a dataset large enough it could seem that we solved our missing-values problem. But what if we are dealing with this scenario: laboratory technician writes down only results that are out of normal range, only indicative results. In this situation, all patients with normal results will be excluded from the dataset. In Figure 4-3 we have depicted this situation. Left graph shows complete population, black colour denotes examples with missing value for attribute 1. Right graph shows this dataset without examples with missing values. It is clear that when erasing examples we are affecting the distribution of our population which can lead to incorrect machine learning models. But, we can also think of a situation in which deleting examples could be justified. For instance, we could have a problematic patient. One that often misses his appointments with the doctor because of the reasons not related to his disease. Another situation is when results for a specific patient are recklessly collected. We could have a number of such patients and could still exclude them, but we must be sure that the reason why they have these missing values is not correlated with the problem we are trying to solve.

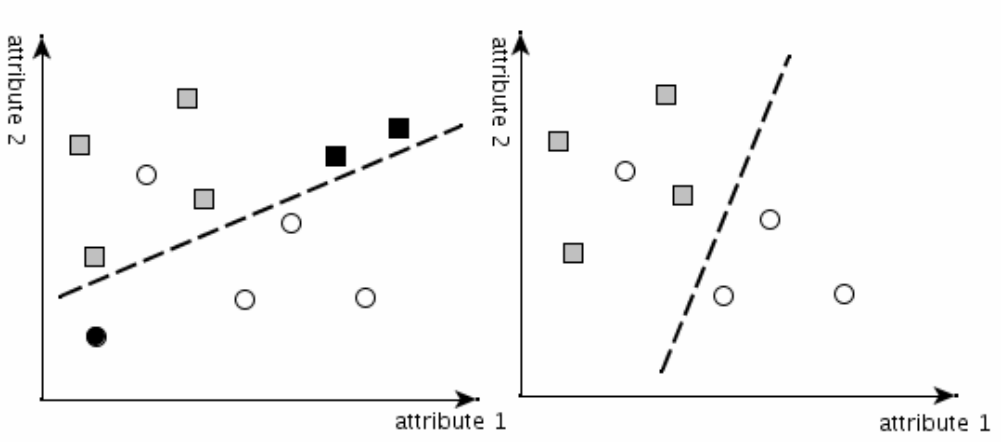

Figure 4-3. Black colour denotes examples with missing Attribute 1 value. Left graph shows an ideal situation with no missing values. Right graph shows situation when examples with missing values are excluded from the dataset.

When we excluded attributes and examples that we could safely exclude, the next logical attempt could be to impute other missing values. We are trying to substitute a missing value with an educated guess value. There are various techniques, from simplest: using arithmetic mean value or median value to more complex like using a prediction of a machine learning algorithm.

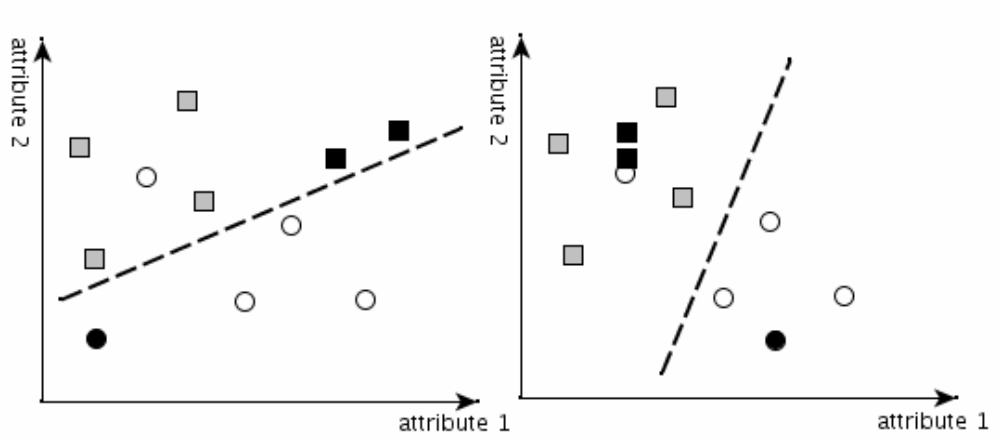

Figure 4-4. Black colour denotes examples with missing Attribute 1 value. Left graph shows an ideal situation with no missing values. Right graph shows situation when missing values are imputed with class mean values.

Figure 4-4 shows us an example situation. Left graph shows us a general population. Black colour denotes examples for which we do not know the attribute 1 value, therefore we do not know in which place in graph should this examples be positioned. Right graph shows us a situation when we filled missing spots with category specific arithmetic mean values. It is obvious that this does not work in this situation, because we only emphasized something that already known and that we severely distorted population's distribution. Although complex

methods for data imputation which try to statistically eliminate chances for such situations as depicted in our Figure 4-4 exist, imputation alters distribution of the population and generally it cannot be used safely.

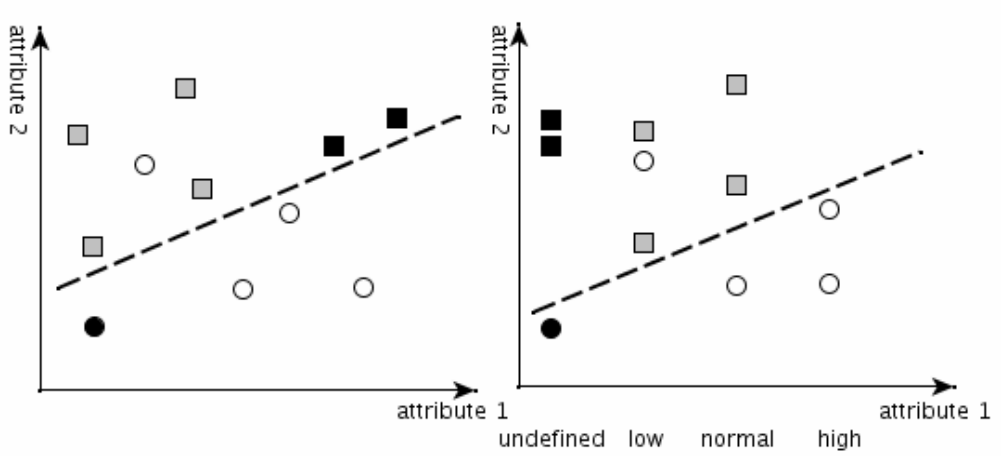

Figure 4-5. Black colour denotes examples with missing Attribute 1 value. Left graph shows an ideal situation with no missing values. Right graph shows situation when Attribute 1 is categorized into four categories: undefined, low, normal, high.

Attribute categorization follows a simple line of thought: in medicine, it is usually not important what the absolute value of a measurement is but rather whether it is high, normal or low, or maybe normal or abnormal. If we categorize our numerical variables in such a way, we can easily put all missing values in an additional category of undefined values. If missing values originate from a fact that expert thought that this measurement would be unnecessary, or he omitted the result because it was normal and therefore not indicative, this additional category of unknown values could prove itself very useful.

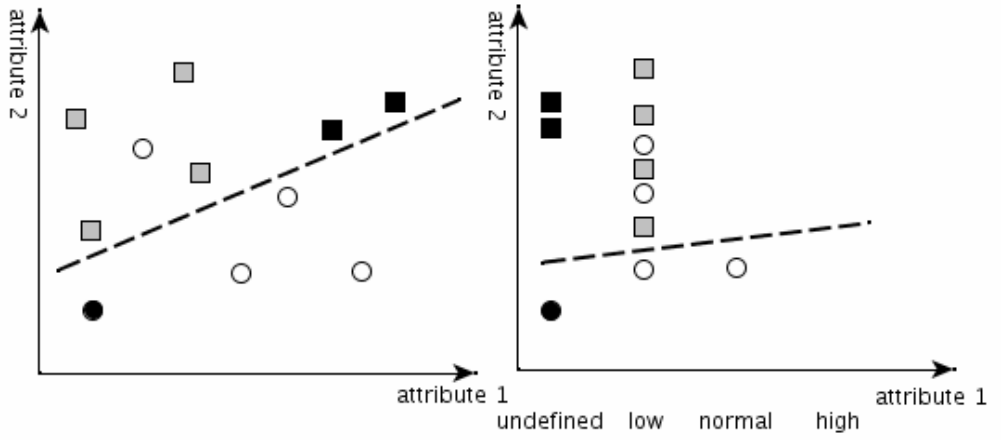

Figure 4-6. Black colour denotes examples with missing Attribute 1 value. Left graph shows an ideal situation with no missing values. Right graph shows situation when Attribute 1 is categorized into four categories: undefined, low, normal, high. Cutoff values are badly set resulting by losing most of the information contained in Attribute 1.

Figure 4-5 gives us an example: left graph presents us an ideal case when all information is known. Black colour denotes examples for which we do not know value of the attribute 1. Right graph shows a situation when we categorized attribute 1 in four categories: undefined, low, normal and high. This example shows us that most of the information can be preserved. However, let us look at Figure 4-6. Cut-off values are badly set, and we lost important bits of information which resulted in bad prediction model.

This means that good definition of categories is the crucial step for this approach. If domain experts are able to set the categories appropriately based on their working experience then attribute categorization can be a very good approach. The intention should be to provide the lowest possible number of categories, because although we can call them undefined, low, normal and high, ordering is lost for data mining algorithms. They cannot discern that high is higher than normal, and much higher than low. Therefore, a part of information is lost. The best possible scenario is to have normal, abnormal and undefined in which case the ordering of categories is irrelevant.

A variation of the above technique is to introduce missing-value indicator variables. For each original variable with missing values, a corresponding indicator variable is introduced. Such a variable takes values 1 on the examples where the original variable has a corresponding missing value and 0 otherwise. The previous approach instead, introduces a new value (rather than a new variable): the value “undefined”. After the introduction of indicator variables, the database is still left with missing values. These can be imputed with any imputation algorithm. The difference from simple imputation where there are no indicator variables is that a machine learning algorithm can use the information of what has been imputed. For example, an algorithm may learn not to trust an imputed value as much or infer that there is a predictive pattern hidden in the missing values.

4.3. Model induction

Application of KD (ML) tools is the central part of the KD process but from the time consumption perspective, this is often the most simple and straightforward task. Its complexity stems from the fact that these tools are often based on rather sophisticated and very specific algorithms constructed for the application by technical experts. They typically have a number of different options, or free parameters, that have to be fine-tuned during the model development, and it is assumed that the user has detailed understanding of underlying algorithm, as well as experience in using it. By inappropriate usage of these options, as well as in case of inadequately prepared data, it can easily happen that obtained results are formally correct but without real significance for the domain. The most common problem is data overfitting which often happens when the user tries to inappropriately optimize the classification performance of the induced models.

KD tools today are still a hot research topic and collections of different KD algorithm implementations are available as the results of scientific projects. One

of the most well known is Weka (http://www.cs.waikato.ac.nz/ml/weka/), which integrates dozens of algorithms in a framework very convenient for experimental work. The basic problem of Weka and other similar packages like Tanagra (http://eric.univ-lyon2.fr/~ricco/tanagra/en/tanagra.html) and Orange (http://magix.fri.uni-lj.si/orange/) is that they are mainly intended for the development and evaluation of KD methodology. Their application, especially by non-experts in medical and similar domains is not a simple task, but currently it is practically the only possible solution when a few different algorithms have to be applied on the same dataset.

Different classifications of problem/analysis types in KD field are given in the Table 4-1.

Table 4-1. Types of KD tasks

By the type of learning	Supervised (known target concept), unsupervised (target concept not specified)
By the type of methodology (for supervised learning)	Classification (for categorical target concepts), regression (for numerical target concepts)
By the type of methodology (for unsupervised learning)	Association mining (applicable only for categorical attributes), clustering of instances
By the type of input data	Propositional, relational, time-series, data streams, text, graphs
By the type of problem (specific for the medical domains)	Diagnosis, prognosis, survival analysis, descriptive analysis, attribute significance

4.3.1. Overfitting prevention

In the process of model induction special care has to be given to the selection of the model complexity. The selection of a too simple models leads to underfitting which means the inability to describe complex models accurately enough. The selection of a too complex model leads to overfitting which means that the model can become an expert on the training data by learning too complex details. The process is illustrated in Figure 4-7. The undesired result of overfitting is the loss of the generalization power and reduced accuracy on previously unseen data.

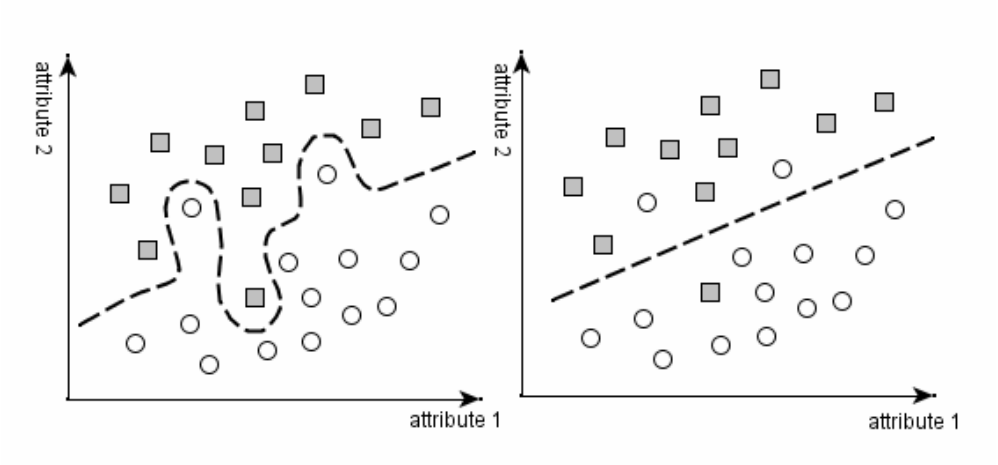

Figure 4-7. Illustration of the concept of data overfitting. Left graph shows the overfitting problem – the perfect classification that performs badly on unseen data. Right graph shows the best fit for the given data.

Overfitting prevention is possible by obtaining more data, noisy data exclusion or even addition of random data, but is not always possible. Since it cannot always be prevented, every machine learning approach must be able to solve the problem of data overfitting. Classical approaches coming from decision tree learning include pre-pruning (stopping induction before overfitting actually happens) and post-pruning (reducing the model complexity after overfitting has occurred). Another approach coming from neural networks is based on stopping the learning process when the model starts to perform worse on unseen data. Cross-validation described in Section 4.4.1 can be successfully used for detecting overfitting. More modern approaches to overfitting prevention include voting of many independent classifiers (random forest approach) and reducing the hypotheses space (relevancy constrains in rule learning like subgroup discovery approach).

4.3.2. Handling unbalanced datasets

Datasets with a highly skewed class distribution are very common in a variety of challenging real-world data mining problems, such as fraud detection, medical diagnosis and various problems in business decision-making. In these domains, classifier learning methods that do not take this skewness into account do not perform well. In extreme cases, ignoring it may produce models that are useless because they predict the more frequently occurring classes much more often than the infrequently occurring classes. This is largely due to the fact that most classifiers are designed to maximize accuracy. In many instances, such as for medical diagnosis, this classification behaviour is unacceptable because the minority class is the class of primary interest (i.e., it has a much higher misclassification cost than the majority class).

Cost-sensitive classification

The performance of a classifier for a two-class problem can be described by the confusion matrix (Table 4-3). Corresponding to a confusion matrix is a cost matrix that provides the costs associated with the four possible outcomes shown in the confusion matrix. We refer to the costs as C_{TP}, C_{FP}, C_{FN}, and C_{TN}. as depicted in Table 4-2.

Table 4-2. Cost matrix for a two-class problem

		ACTUAL	
		POSITIVE CLASS	NEGATIVE CLASS
PREDICTED	POSITIVE CLASS	C_{TP}	C_{FP}
	NEGATIVE CLASS	C_{FN}	C_{TN}

Costs are not necessarily monetary, e.g. a cost can also be a waste of time, or the severity of an illness. As is often the case in cost-sensitive learning, we assign no costs to correct classifications, so C_{TP} and C_{TN} are set to *0*. Let the minority class be the positive class and the majority class be the negative class. Since the minority (positive) class is often more interesting than the majority (negative) class, typically $C_{FN}>C_{FP}$. When misclassification costs are known the best metric for evaluating classifier performance is the total cost. The formula for total cost is:

$$\text{Total Cost} = (FN \times C_{FN}) + (FP \times C_{FP}) \quad (4.1)$$

There are several methods that can be used when dealing with skewed class distributions with unequal misclassification costs. The most direct method is to use a learning algorithm that is itself cost-sensitive, that is, the learning algorithm factors in the costs when building the classifier, like naïve Bayes, random forest etc. These cost-sensitive algorithms usually implement class weighing – allowing weight factor setting for each of the classes prior to modelling. This allows algorithms to take into account the unbalance of data.

Another way to obtain a classifier that is useful for cost-sensitive decision-making is by instances-rebalancing. In the two-class case, standard learning algorithms implicitly make decisions based on the probability threshold *0.5*. The most common method to make a standard learning algorithm able to produce a classifier that makes decisions based on a general p^* (instead of 0.5) is to rebalance the training set given to the learning algorithm, i.e. to change the proportion of positive and negative training examples in the training set. The following formula provides the general solution for how to do it correctly. To make a target probability threshold p^* correspond to a given probability threshold p_0 (actual proportion of positive examples in the training set), the number of negative examples in the training set should be multiplied by:

$$\frac{p^*}{(1-p^*)}\frac{(1-p_0)}{p_0} \tag{4.2}$$

Given a training set *S* and a desired probability threshold p^*, the formula says how to create a training set *S'* by changing the number of negative training examples in order to get the optimal classifier.

The formula does not say in what way the number of negative examples should be changed. If a learning algorithm can use weights on training examples, then the weight of each negative example can be set to the factor given by the formula. Otherwise, sampling must be used, in order to alter the class distribution of the training data.

The reason that altering the class distribution of the training data aids learning with highly skewed datasets is that it effectively imposes non-uniform misclassification costs. This equivalency between altering the class distribution of the training data and altering the misclassification cost ratio is well known and was formally described by Elkan [3].

Sampling

There are two basic sampling methods that can be used: oversampling and undersampling, both of them have been used to deal with class imbalance [4,5,6]. Oversampling replicates minority-class examples while undersampling discards majority-class examples.

Sampling can be done either randomly or deterministically. While deterministic sampling can reduce the variance, it risks introducing bias, if the non-random choice to duplicate or to eliminate examples is correlated with some property of the examples. Undersampling that is deterministic in the sense that the fraction of examples with each value of a certain feature is held constant is often called stratified sampling. It is important to keep all available examples of the rare class.

There are known disadvantages associated with the use of sampling to implement cost-sensitive learning [7,8,9]. The disadvantage with undersampling is that it discards potentially useful data. The main disadvantage with oversampling is that by making exact copies of existing examples, it makes overfitting likely. In fact, with oversampling it is quite common for a learner to generate a classification rule to cover a single, replicated, example. A second disadvantage of oversampling is that it increases the number of training examples, thus increasing the learning time.

Nevertheless, there are also several reasons for using sampling to implement cost-sensitive learning. The most obvious reason is there are not cost-sensitive implementations of all learning algorithms and therefore a wrapper-based approach using sampling is the only option. While this is certainly less true today than in the past, many learning algorithms (e.g., C4.5 decision tree learner) still do not directly handle costs in the learning process. The second reason for using sampling is that many highly skewed datasets are enormous and the size of the

training set must be reduced in order for learning to be feasible. In this case, undersampling seems to be a reasonable, and valid, strategy.

Snowball method

Snowball method [10] is a training method used only for iterative modelling approaches such as neural networks. The idea of the method is to first train a neural network purely with examples of the minority class in order to establish a set of connection weights favourable to these examples. The network is then dynamically trained using a training set which includes all the examples of minority class and an increasing number of examples of the majority class so that the network extends its capacity to recognize the examples of the majority class. In this way, the undo effect of the presentation of minority class examples can be greatly reduced.

The described techniques can balance the accuracy of generated models among classes, but they usually increase the overall error rate and the error rate of the majority class, while increasing the accuracy of the minority class.

4.4. Model evaluation and interpretation

Model evaluation and interpretation is the most relevant part of post-processing of the results obtained by application of KD tools. Model evaluation implies measurement or estimation of the quality of induced models. Interpretation implies recognition of the actual meaning of the obtained results in the context of the domain properties and available data. Implementation of both activities partially depends on the properties of the KD methodology used to obtain the model. Only after implementing the both steps, the results of the KD process have their real relevance.

4.4.1. Model evaluation

Model evaluation is mainly concerned with estimation of the prediction quality of the induced models. In order to be useful, models should have classification accuracy as high as possible. That is also true for descriptive induction tasks. Although maximal prediction accuracy is not the main goal of descriptive induction tasks, high generalization error or large difference in prediction quality for the training and the test set are a reliable sign that the induction of a classifier was not successful in finding really relevant relations between attribute values and the classes.

It is important to notice that classification accuracy which represents the quality of the model must be measured on examples that are not used in the induction process. Good classification accuracy is rather easy to obtain on the training cases and the real danger is to optimize accuracy on the training set while accuracy on unseen examples remains low. We say that overfitting happened if the difference between measured accuracies on the training and the independent test sets is large.

The easiest way to estimate the classification accuracy of a model is to use a sufficiently large test set to measure the model accuracy. In cases when we are not satisfied with the measured accuracy, we have to repeat the learning process with other parameters. It must be noted that if we try to repeat this process many times, we can finally end with the model that will have very good prediction quality for this test set only. The effect is called *test set overfitting*. The lessons to be learned here is that the learning process may be repeated many times in order to obtain the optimal model, but the test set can be used only once, or a small number of times, in order just to estimate the quality of the induced model.

The main problem of the previous approach is that typically, and especially in medical application, we do not have many examples that can be used for learning. In such situations we cannot afford to put too many examples into the test set because in this way we would reduce the size of the training set even more. The well-known solution is cross-validation. In this approach all available data are used for learning while the accuracy of the resulting theory is estimated from a small hold-out subset.

In cross-validation the data is divided into *n* parts. In *n* separate experiments, *n-1* parts are combined into a single training set, and the remaining part is used for testing. This is repeated exactly *n* times, until each part (and thus each training example) has been used once in the role of the test set. The final accuracy is then estimated as an average of the accuracy estimates computed in each of the hold-out experiments. The algorithm is presented in Figure 4-8. It shows a schematic depiction of a 10-fold cross-validation which is most commonly used in practice. An interesting special case is leave-one-out cross-validation, where in each iteration only a single example is held out from training and subsequently used for testing.

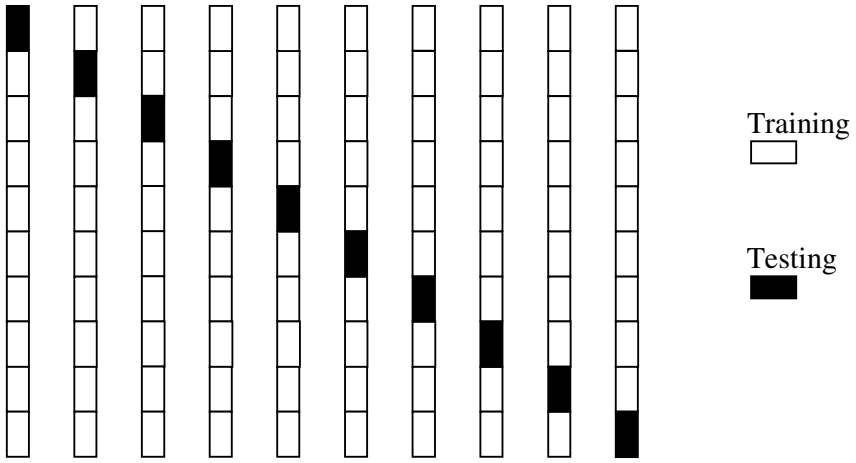

Figure 4-8. A schematic depiction of 10-fold cross-validation where predictive accuracy of the final model is estimated as the average accuracy on 10 hold-out testing datasets

For some algorithms, like random forest, it is characteristic that they construct many classifiers (decision trees in the random forest approach) and for each of them they use only a randomly selected subset of the set of available learning examples. Such situation is ideal to use the remaining training cases to evaluate the respective classifier. This approach is known as *out-of-bag* approach. The measured accuracy is used as the weight of the classifier when its output is combined with other classifiers. Additionally, it can be used to estimate the accuracy of the complete ensemble of classifiers.

There are several metrics to evaluate the prediction performance of a prediction model. For ease of presentation we will consider prediction tasks where there are only two classes to categorize which are generally named Positives and Negatives. For example, a model may predict that a patient will be re-hospitalized after her first visit within a year (Positive) or that she will not (Negative). When we apply the model on a test set of new patients we will obtain what is called a *confusion matrix* showing the correct and erroneous classifications of the model for each category as in the below:

Table 4-3. Confusion matrix for a two-class classification problem

		ACTUAL	
		POSITIVE CLASS	**NEGATIVE CLASS**
PREDICTED	**POSITIVE CLASS**	*TRUE POSITIVE (TP)*	*FALSE POSITIVE (FP)*
	NEGATIVE CLASS	*FALSE NEGATIVE (FN)*	*TRUE NEGATIVE (TN)*

Within the table, the first True/False part refers to the correctness of the classification while the second part (Positive/Negative) refers to the prediction. Thus, True Positive is an example predicted positive that is in reality a positive example. To create the confusion matrix we obviously need the true classification of the examples in the test set.

One of the most commonly used performance metrics is the *classification accuracy*. Classification accuracy is defined as the percentage (rate) of the correct classifications of the model. For example, a classification accuracy of 80% implies that the model on average makes 80% correct classifications. Expressed with the quantities in the above table this rate is estimated as:

$$\frac{TP + TN}{TP + TN + FP + FN} \tag{4.3}$$

Although generally accepted, expressing the performance using the classification accuracy has at least two significant disadvantages:

- It depends on the prior distribution of the two classes. For example, if the prior class distribution is 80% positive examples and 20% negatives, then just by classifying any new example as positive we can obtain a classification

accuracy of 80%. Thus, just reporting classification accuracy does not convey how much better our classification model performs relatively to the most trivial classifier.

- It equally penalizes for all misclassifications costs, i.e., misclassifying a positive as a negative counts equally as misclassifying a negative as a positive. This is typically not desired, especially in medicine. For example, it may be more preferable to misclassify a patient without the disease (negative) as having the disease and perhaps force her to undergo further diagnostic tests, than misclassify a patient having the disease (positive) as not having the disease and forgo the appropriate treatment.

As an alternative to reporting the classification accuracy that does not exhibit the above two disadvantages is the Receiver Operating Curve analysis (ROC). To perform an ROC analysis we need to produce *ranking models* instead of classification models as described below. A classification model outputs a Positive or Negative prediction on any new example. A ranking model on the other hand outputs a *score* of confidence that a new example is a Positive. Most learning algorithms such as the naïve Bayes, support vector machines, artificial neural networks and even decision trees can be easily modified to produce ranking models. The lower the score the more confident is the ranker that the new example is a Negative and the higher the score the more confident is the model that the example belongs in the Positive class. The score of some ranking models correspond to the probability of the new example being a positive: $P(Class=Positive \mid X)$. Such models are called rankers because they can rank a test set of new examples from the most probable to belong to the Positive class to the least probable (highest score to lowest).

Each ranker can be easily converted to a classifier by selecting a threshold. For example, for $t=0.5$ we can classify as positive all examples X for which the ranker reports that $P(Class=Positive \mid X)>t$ and as negative all the remaining ones. When $t=0.5$ and the score corresponds to the probability of belonging to the Positive class, the classifier essentially outputs the most probable class. If $t=0.2$ the classifier becomes more biased towards classifying to the Positive class: even if an example has only 0.4 probability (according to the model) to be a Positive and 0.6 to be a Negative, it will be classified as Positive. By appropriately selecting a threshold, an expert can bias the classifier towards reducing the false negatives or reducing the false positives depending on their relative cost of misclassifications.

During model construction, the data analyst is not aware of the misclassification cost for each class. Typically in medicine, it is the physician that decides how probable a diagnosis needs to be to classify a patient as having the disease or not. Thus, the data analyst cannot select the desired threshold for any given ranking model. The problem then becomes how to compare two ranking models and how to evaluate the final model, when the best threshold is unknown.

To solve the problem, a data analyst measures the performance for all possible thresholds. Let us assume we have a ranker that outputs a score for each example and we convert it to a classifier by selecting a threshold t. We denote with C_t the

resulting classification model. Each such model C_t produces a confusion matrix. We can define two useful quantities for each confusion matrix, the *true positive rate* (TPR) as the fraction of positives correctly predicted *TP/(TP+FN)* (also called *sensitivity*), and *false positive rate* (FPR) as the fraction of negatives incorrectly predicted as positives, *FPR=FP/(TN+FP)* (also called 1-specificity). If we plot the TPR_t (*sensitivity*) versus the FPR_t (*1-specificity*) for several possible thresholds *t* we obtain a graph that looks as depicted in Figure 4-9.

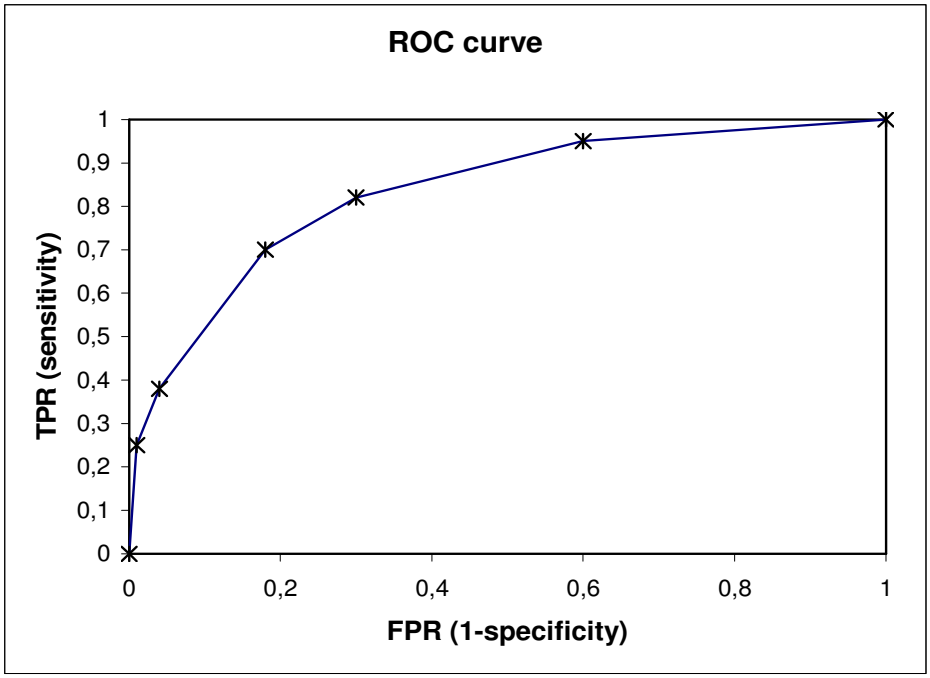

Figure 4-9. A sample ROC graph based on several model (ranker) thresholds (points in the graph, values of TPR and FPR are given for each point).

This type of graphs containing the TPR_t versus the FPR_t for all possible thresholds *t* is the ROC curve and are heavily used in medicine in order to compare diagnostic tests. In our case the medical diagnostic tests correspond to a ranker. Given such a curve, a physician or other expert can select a desired operation threshold *t* for which the pair (TPR_t , FPR_t) is acceptable. The ROC can be used to select between two rankers: if the whole ROC curve of ranker A is always above the ROC curve of ranker B, then ranker A is always preferable than ranker B. If for a specific part of the curve ranker the situation is reversed (i.e., the two ROC curves cross), then for the thresholds that correspond to that FNR of the two models, it is ranker B that is preferred over ranker A. Thus, it may turn out that sometimes ranker A is better than B and vice-versa.

In order to compare two ranking models A or B one can use the following common method. The Area under the ROC curve (AUC) is computed for both models. If a ranker A has an AUC larger than another model B, this roughly implies that its ROC is above B's ROC more often than the opposite. The AUC

quantifies within a single number how predictive is a ranking model for all possible thresholds. The perfect ranker that produces scores for the Positive examples that are always larger than the scores produced for the Negative examples has *AUC=1*. The random ranking model that produces random scores has an *AUC=0.5*. Intermediate AUC's of 0.8, 0.9 etc. imply less than perfect models. AUC's of less than 0.5 imply models that are worse than random predictions! Thus, when these models predict Positive, most likely the example is actually Negative. These models could thus be improved if we reverse their predictions. The AUC of a ranking model has an important intuitive interpretation too. It is the probability that the model will correctly rank from the most likely to be positive to the least likely a pair of a Positive and a Negative example.

4.4.2. Model interpretation

The results of machine learning represent only relations among attributes describing the training set. What is the actual meaning of these results follows from the meaning of attributes and the context in which the examples have been collected.

For example, if it has been detected that the property that patient receives a therapy (i.e. antihypertensive therapy) is characteristic for patients with some diagnosis then this result cannot be interpreted so that the antihypertensive therapy itself is dangerous for the incidence of the diagnosis. The appropriate interpretation is that hypertension is dangerous, therefore people with detected hypertension problems, characterized by the fact that they already take antihypertensive therapy, have a greater probability of being in the target population. Indirectly, this rule also means that we have little chance to recognize the danger of high blood pressure directly from their measured values because seriously ill patients have these values artificially low due to the previously prescribed therapy.

From this example it follows that model interpretation is not possible without at least some understanding of the meaning of attributes describing the patients (it is not reasonable that antihypertensive therapy is dangerous), connections among them (patients with antihypertensive therapy have artificially low values of blood pressure), and conditions in which data are collected (therapies should be taken into account when searching for the relations between laboratory measurements and patient status).

However, sometimes this is not enough. For example, if some diagnosis is connected with patient's positive alcohol consumption and negative stress, it is rather difficult to recognize the connection that has "no stress" property with alcohol consumptions and the target diagnosis. Remedy for such situations could be the detection of *supporting factors* which additionally characterize the subpopulation described by positive alcohol consumption and negative stress. Such factors can be detected by statistical comparison of the subpopulation with all negative cases (patients that do not have the target diagnosis). In the described situation it has been detected that supporting factors are increased age (mean value 72 years) and no preventive therapy (no aspirin). After that it is clear that

"no stress" property comes from the fact the diagnosis is characteristic for elderly people while the connection with "alcohol consumption yes" property actually describes the subpopulation that does not care much about its health, suggested also by the property that even very simple and cheap prevention therapy like aspirin is not used. When combined together the picture of the subpopulation becomes much more clear and the relation with the target diagnosis (brain stroke in the concrete situation) becomes understandable. More details about computation of supporting factors can be found in Section 5.5.7.

4.5. Model deployment

Classification and/or regression models that are the results of the knowledge discovery (data mining) process can, in principle, be deployed in decision support services. Those models can be deployed in two basic ways: as a predicting "black box" model or as a descriptive model.

Black box models serve as predictors of outcomes. They don't necessarily give insight into the "why" of the prediction and they don't give the valuable information on the sole process of prediction, just the final result (e.g. stock market price prediction, weather forecast, etc.); therefore accuracy of those models is the key of their success.

Such models are marginally useful in medical domain tasks because medical domain problems require interpretable results in order to be useful for diagnostic or prognostic decisions support. In medical domain, decision support services are valuable because of their additional descriptive information that enables the medical personnel the explanation of the possible decision and aids them in the decision process. Medical domain decision support services therefore usually deploy description models because of their interpretability.

However, the sole deployment of knowledge discovery models is not of the utmost importance in decision support services. Additional results of the KD process like patient stratification into risk groups, assessment of the variable importances, mutual variable correlations, etc. are very valuable outcomes of KD process which do not require deployment of a fully formed model in order to be used as an interpretation and evaluation tool. The extracted knowledge itself can also be of great importance for the development of decision support services.

4.6. Bibliography and references

[1] Bellazzi, R., Zupan, B. (2008) *Predictive data mining in clinical medicine: Current issues and guidelines*. International Journal of Medical Informatics, 2008, 77(2): pp.81–97

[2] Abu-Hanna, A., Lucas, P. J. F. (2001) *Prognostic Models in Medicine: AI and Statistical Approaches*, Method Inform Med; 40: pp.1–5

[3] Elkan, C. (2001) *The foundations of cost-sensitive learning.* In proceedings of the Seventeenth International Joint Conference on Artificial Intelligence

[4] Kubat, M., Matwin, S. (1997) *Addressing the curse of imbalanced training sets: one-sided selection.* In proceedings of the Fourteenth International Conference on Machine Learning, pp.179–186

[5] Weiss, G. M., McCarthy, K., Zabar, B. (2007) *Cost-Sensitive Learning vs. Sampling: Which is Best for Handling Unbalanced Classes with Unequal Error Costs?* Department of Computer and Information Science Fordham University Bronx, NY, USA

[6] Abe, N., Zadrozny, B., Langford, J. (2004) *An iterative method for multi-class cost-sensitive learning*. KDD '04, August 22-25, 2004, Seattle, Washington, USA

[7] Chan, P., Stolfo, S. (1998) *Toward scalable learning with non-uniform cost and class distributions: a case study in credit card fraud detection.* American Association for Artificial Intelligence

[8] Chen, C., Liaw, A., Breiman, L. (2004) *Using random forest to learn unbalanced data*. Technical Report 666, Department of Statistics, University of California at Berkeley, http://www.stat.berkeley.edu/users/chenchao/666.pdf

[9] Japkowicz, N., Stephen, S. (2002) *The class imbalance problem: a systematic study*. Intelligent Data Analysis Journal, 6(5)

[10] Murphey, Y.L., Guo, H., Feldkamp, L.A. (2004) *Neural learning from imbalanced data*. Appl. Intell. Neural Networks Appl. 21: pp.117–128

5. Project specific modelling methodology

This section presents modern machine learning methodology which may be relevant for the analysis of data collected by the platform, analysis of retrospective medical data, and the methodology appropriate for inclusion into the KD Web service. This section in conjunction with the next section represents the central part of the deliverable.

The presentation begins with the short presentation of decision trees which are perhaps the best known machine learning approach that is the basis for many other algorithms. Among other properties, decision trees are characterized by the explicit representation of the knowledge extracted from the data. This property is especially important for medical domains. The relevance of the decision trees is that they are used as the basic building block in the approach known as random forest. This is a very novel and modern algorithm with quite a few nice properties, among which perhaps the most relevant is an original and simple approach to overfitting prevention. This algorithm is foreseen as the main building block for the KD Web service and that is the reason why we have devoted special care to its presentation in this section. The first part of the section ends with the description of the methodology for survival analysis. This is a typical KD methodology developed primarily for medical applications. The significance of the work for the Heartfaid project is the demonstration of how RF learning can be used for survival analysis.

The second part of this section is devoted to rule based learning. We start by decision lists which are a natural extension of decision tree learning and continue with subgroup discovery methodology. The latter is a very modern approach especially appropriate for descriptive data analysis tasks. Some of the most relevant results obtained from the retrospective HF datasets and presented in the following section have been obtained by this methodology. Additional significant effort is necessary for the integration of this methodology into the KD Web service and it is our long-term goal.

The last part of the section includes presentation of a few methodologies that are intended for the construction of high quality prediction models. Among them special care has been devoted to the support vector machine methodology that is currently very successfully applied in the most difficult classification tasks. The central issue is construction and selection of most appropriate kernels. The section ends with the short presentations of radial basis function networks and Bayesian Learning. All of them have been tested on available HF datasets and presentation of the obtained results is the topic of the following section.

5.1. Decision trees

Algorithms for constructing decision trees are among the most well known and widely used machine learning methods. Among decision tree algorithms, J. Ross Quinlan's ID3 (Iterative Dichotomiser 3) [1] and its successors C4.5 [2] and J4.8 are probably the most popular in the machine learning community. These algorithms and variations on them have been the subject of numerous research papers since Quinlan introduced ID3.

A decision tree is a structure built recursively which has leaf nodes labelled with a class value and test nodes having two or more outcomes, each linked to a subtree. Decision tree learners start selecting an attribute to place at the root node and then make one branch for each possible value. This splits up the instance set into subsets, one for every branch, using only those instances that reach the branch. If at any time all instances at a node belong to the same class, the algorithm stops developing that part of tree because the node is a leaf.

Strategy adopted by decision tree learners for building the tree-structure from a training set *S* is known as divide-and-conquer strategy [3]:

- If all the cases in *S* belong to the same class C_j (trivial partition of S), the decision tree is a leaf labelled with C_j .
- Otherwise, let *B* be some test with outcomes b_1 , b_2 , ..., b_t that produces a non-trivial partition of *S*, and denote by S_i the set of cases in *S* that has outcome b_i of *B*. The decision tree is depicted in Figure 5-1.

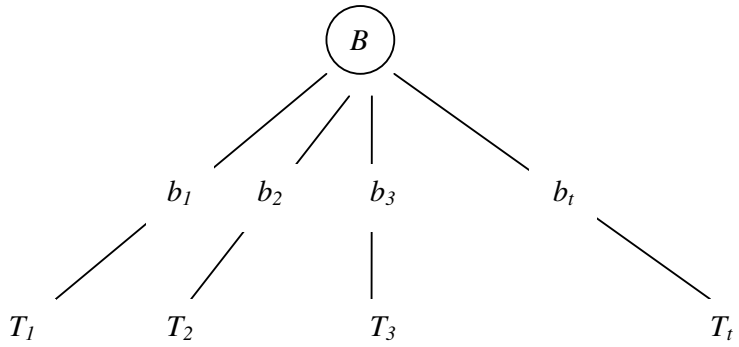

Figure 5-1. An example of a decision (sub)tree in the process of tree growing where T_i is the result of growing a decision tree for the cases in S_i.

Every internal node (test node) in a decision tree involves testing a particular attribute. Usually the test at a node compares an attribute with a constant. However, some trees compare two attributes with each other or utilize some function on one or more attributes.

To classify an unknown instance, it is routed down the tree according to the values of the attributes tested along the path, and when a leaf is reached the instance is classified according to the class of that leaf.

If the attribute that is tested is a nominal one, the number of children is usually the number of possible values of the attribute. In this case the attribute will be tested only once down the tree. However sometimes the attribute values could be divided into two subsets and the tree branches two ways depending on which subset the value lies in; in that case the attribute might be tested more than once in a path.

If the attribute is *numeric* the test at the node usually determines whether its value is grater or less than a predetermined constant, giving a two way split. Sometimes a three-way split may be used, in which case there are several different possibilities (e.g. if *missing value* is treated as an attribute value in its own right, or if integer-valued attributes are split into *less than*, *equal than* and *grater than*). A numeric attribute is often tested more than once in a path from root to leaf, each test involving a different constant. If the algorithm restricted splits on numeric attributes to be binary, it introduces an important difference between numeric attributes and nominal ones: once the algorithm branches on a nominal attribute, all its information is used, whereas successive splits on a numeric attribute may continue to yield new information. A nominal attribute can only be tested only once on any path from the root a tree to a leaf, a numeric one can be tested many times. This can yield trees that are difficult to understand because the tests on a single numeric attribute are not located together but can be scattered along the path.

For decision tree building is necessary to decide how to determine which attribute to split on, given a set of examples with different classes. C4.5 generates a set of candidate tests and then selects among them. The algorithm uses tests of three types each involving only a single attribute A_a. Decision regions in the instance space are thus bounded by hyperplanes, each orthogonal to one of the attribute axes (as can be seen in Figure 5-3b).

If A_a is a nominal attribute with z values, possible tests are [3]:

- "A_a = const" with z outcomes, one for each A_a value. (This is the default)
- "$A_a \in G_i$" with outcomes true and false, where $G = \{G_1, G_2, \ldots, G_g\}$ is a partition of the values of attribute A_a and $2 \leq g \leq z$ outcomes. Tests of this kind are found by a greedy search for a partition G that maximizes the value of splitting criterion. This criterion will be discussed below.

If A_a is a numeric attribute, the form of the test is *"$A_a \leq \Theta$"* with outcomes true and false, where Θ is a constant threshold. Possible values of Θ are found by sorting the distinct values of A_a that appear in S, and identifying one threshold between each pair of adjacent values.

Most learning systems attempt to keep the tree as small as possible because smaller trees are more easily understood and, by Occam's Razor arguments ("The simplest explanation is usually the best"), are likely to have higher predictive accuracy. Since it is infeasible to guarantee the minimality of the tree, C4.5 relies on greedy search selecting the candidate test that maximizes the heuristic splitting criterion.

Two such criteria are used in C4.5, *information gain* and *gain ratio* [3]. Let $RF(C_j,S)$ denote the relative frequency of cases in S that belong to class C_j. The information content of a message that identifies the class of a case in S is then

$$I(S) = -\sum_{j=1}^{x} RF(C_j, S)\log(RF(C_j, S)) \tag{5.1}$$

After S is partitioned into subsets S_1 , S_2 , …, S_t by a test B, the information gained is then

$$G(S,B) = I(S) - \sum_{i=1}^{t} \frac{|S_i|}{|S|} I(S_i) \tag{5.2}$$

The gain criterion chooses the test B that maximizes $G(S,B)$.

A problem with this criterion is that it favours tests with numerous outcomes – for example, $G(S,B)$ is maximized by a test in which each S_i contains a single case. The gain ratio criterion sidesteps this problem by also taking into account the potential information from the partition itself:

$$P(S,B) = -\sum_{i=1}^{t} \frac{|S_i|}{|S|} \log\left(\frac{|S_i|}{|S|}\right) \tag{5.3}$$

Gain ratio then chooses, from among the tests with at least average gain, the test B that maximizes $G(S,B)/P(S,B)$.

Another splitting criteria that is also frequently used (specifically in random forest) is Gini index [4]. Gini index heuristic (Gini impurity measure) is defined as:

$$i(t) = \sum_{\substack{k=1,l=1 \\ k\neq l}}^{M} p(k \mid t)\, p(l \mid t) = 1 - \sum_{k=1}^{M} p(k \mid t)^2 \tag{5.4}$$

where $p(k|t)$ denotes a probability of getting a value k in node t which is measured as proportion of records assigned to node t for which response variable y equals k. As can be seen, Gini impurity is an impurity measure based on squared probabilities of membership for each target category in the node. It reaches zero when all cases in the node fall into a single target category (least impure means that a node contains members of the same category). At each node, the split that maximizes the decrease in impurity is chosen. At every split of a node, the Gini impurity criterion for the two descendent nodes is less than the parent node.

The divide and conquer algorithm partitions the data until every leaf contains cases of a single class, or until further partitioning is impossible because two cases have the same values for each attribute but belong to different classes. Consequently, if there are no conflicting cases, the decision tree will correctly classify all training cases what can lead to overfitting and to lower predictive accuracy in most applications [1].

Overfitting can be avoided by two different strategies: *prepruning* and *postpruning*. Prepruning (sometimes called *forward pruning*) would involve trying to decide during the tree-building process when to stop developing subtrees, while postpruning (or *backward pruning*) removes some of the structure of the decision tree after it has been produced. Most authors agree that the latter strategy is preferable since it allows potential interactions among attributes to be explored before deciding whether the result is worth keeping. Moreover postpruning may cause the accuracy on the training set to decrease but it may increase the accuracy on an independent test set. C4.5 employs a postpruning mechanism.

Two different operations have been considered for postpruning: *subtree replacement* and *subtree raising*. At each node, a learning scheme might decide whether it should perform subtree replacement, subtree raising, or leave the subtree unpruned. Subtree replacement selects some subtrees and replace them by single leaves, proceeding from the leaves and working back up toward the root. Subtree rising is a more complex and a potentially time-consuming operation. Figure 5-2 shows an artificial example for subtree rising.

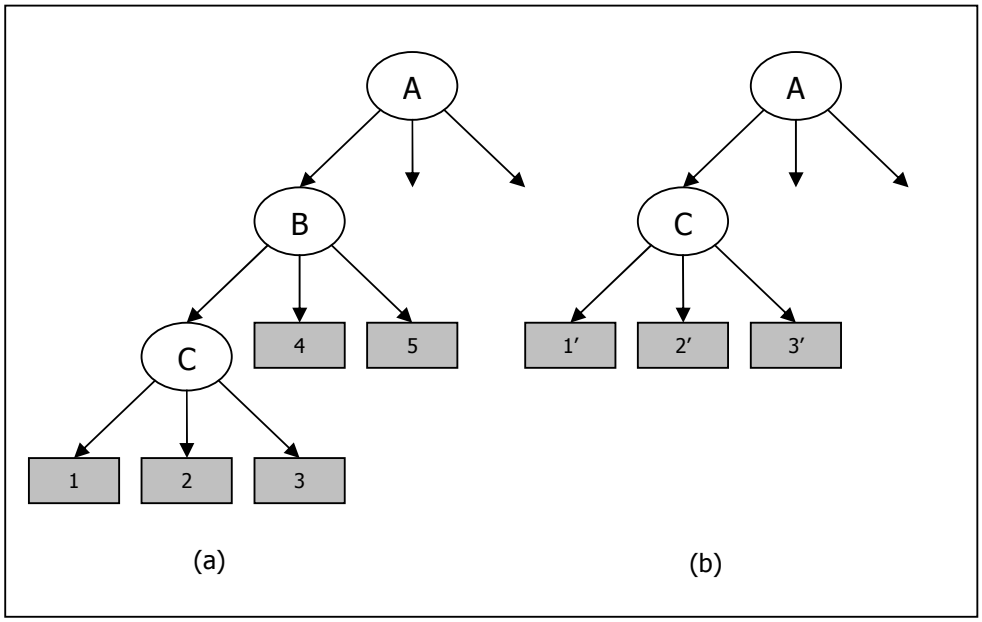

Figure 5-2. An artificial example of subtree rising. The original tree is in (a) and the resulting one is shown in (b)

The entire subtree from C downward has been raised to replace the B subtree. The children of B and C are represented as leaves but they can be subtrees. If this subtree raising operation is performed it is necessary to reclassify the instances at the nodes 4 and 5 into the new subtree headed by C. This is why children nodes of C are named 1', 2', 3': they are not the same as the original 1, 2, 3, because they differ by the inclusion of the instances originally covered by nodes 4 and 5. In the

actual decision tree implementations subtree raising is generally restricted to raising the subtree of the most popular branch (for the raising illustrated in Figure 5-2, the branch from B to C has more training examples than the branches from B to 4 or from B to 5).

In order to decide rationally whether to replace an internal node by a leaf (subtree replacement) or whether to replace an internal node by one of the nodes below (subtree raising), it is necessary to estimate the error rate that would be expected at a particular internal node as well as at a leaf one, given an independently chosen test set. One way of obtaining an error estimate is to hold back some of the training data and use it as an independent test set to estimate the error at each node. This method is known as *reduced-error pruning*. It suffers from the disadvantage that the actual tree is based on less data.

An alternative method is to try to make some estimate of error based on the entire training set. C4.5 adopts this strategy using a heuristic based on some statistical reasoning. The statistical underpinning is rather weak and ad hoc but it seems to work well in practice.

Consider some classifier Z formed from a training set S, and suppose that Z misclassifies M of the instances in S. The true error rate of Z is its accuracy over the entire universe from which the training set was sampled. The true error rate is usually markedly higher than the classifier's *resubstitution* error rate on the training cases (here $M/|S|$), which might be near zero for an unpruned decision tree.

The true error rate is often estimated by measuring Z 's error rate on a collection of unseen cases that were not used in its construction. This is the best strategy when a substantial set of unseen cases is available. In many applications, though, data is scarce and all of it is needed to construct the classifier. C4.5 estimates the true error rate of Z using only the values M and $|S|$ from the training set as follows.

If an event occurs M times in N trials, the ratio M/N is an estimate of the probability p of the event [3]. We can go further and derive confidence limits for p; for a given confidence CF, an upper limit p_r can be found such that $p \leq p_r$ with probability $1\text{-}CF$.

p_r satisfies the following equations:

$$CF = \begin{cases} (1-p_r)^N & for \quad M = 0 \\ \sum_{i=0}^{M} \binom{N}{i} p_r^i (1-p_r)^{N-i} & for \quad M > 0 \end{cases} \qquad (5.5)$$

Now, the classifier Z can be viewed as causing M error events in $|S|$ "trials". Since Z was constructed to fit the cases in S, and so tends to minimize the apparent error rate, the upper bound p_r is used as a more conservative estimate of the error rate of Z on unseen cases. C4.5 uses a default CF value of 0.25, but this can be altered to cause higher or lower levels of pruning.

5.2. Random forest

Some classification and regression methods are unstable, i.e. small perturbations in training sets or in construction procedure may result in large changes in constructed predictors (e.g. classification and regression trees, subset selection, neural networks) [5]. Those methods can have their accuracy greatly improved by perturbing and combining. This can be achieved by generating multiple versions of the predictor by perturbing the training set or construction method and then combining these versions into a single, ensemble predictor [6]. Random forest is one of those predictors. It is a relatively new method which gives very good results in terms of efficiency (execution time) and quality and reliability of results.

Random forest is a general purpose classification and regression ML method developed by Leo Breiman and Adele Cutler [7]. The term random forest comes from term random decision forests, proposed by Tin Kam Ho of Bell Labs in 1995. The RF method is a meta-learning algorithm that works by combining Breiman's bagging and Ho's random subspace method [8] approaches to construct an ensemble of decision trees. An illustration of random forest classification on an artificial dataset with two input dimensions and $y=x$ decision boundary is presented in Figure 5-3.

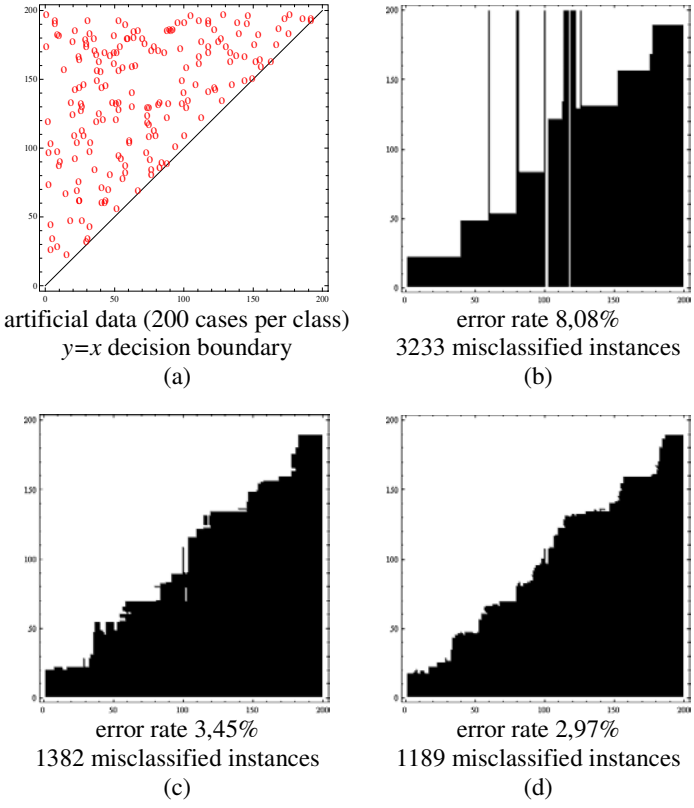

Figure 5-3. Illustration of the RF classification performance in respect to the number of constructed trees. Learning set with 200 positive and 200 negative examples (a), model (separation line) and error rate obtained by a single tree tested on 40,000 cases (b), the result obtained by 10 trees (c), and 100 trees (d).

In order to obtain a good ensemble of classifiers, it is necessary to build a set of diverse (uncorrelated), but not necessarily very accurate base classifiers. A very important advantage of an independent decision tree is its low bias, but its main drawback is its high variance, therefore decision trees are not as accurate as one would desire. Nevertheless, using multiple diverse decision trees as base classifiers for an ensemble, (i.e. trees, which then make a forest) yields good results because the bias stays low, and because of multiple trees, variance is substantially decreased - single trees predict differently, but as a group they are generally remarkably accurate. The base classifier in the case of RF is a random decision tree which depends on the value of an independently sampled random vector of training data and a random perturbation in the procedure of tree growing.

RF method therefore combines two sources of randomness:

- randomization of training data using bootstrapping
- random subspace method for attribute selection while growing a tree

Bootstrapping is a statistical resampling method that uses sampling with replacement from the original (training) sample. When sampling with replacement, not all of the samples are included in the final data. Data not included into the training set is called out-of-bag (oob) data and is used in later accuracy assessment of the generated tree. The bootstrapping method is carried out for each tree in the forest, thus providing each tree with its own set of learning and oob data. An example of result of the bootstrapping method is shown on Figure 5-4.

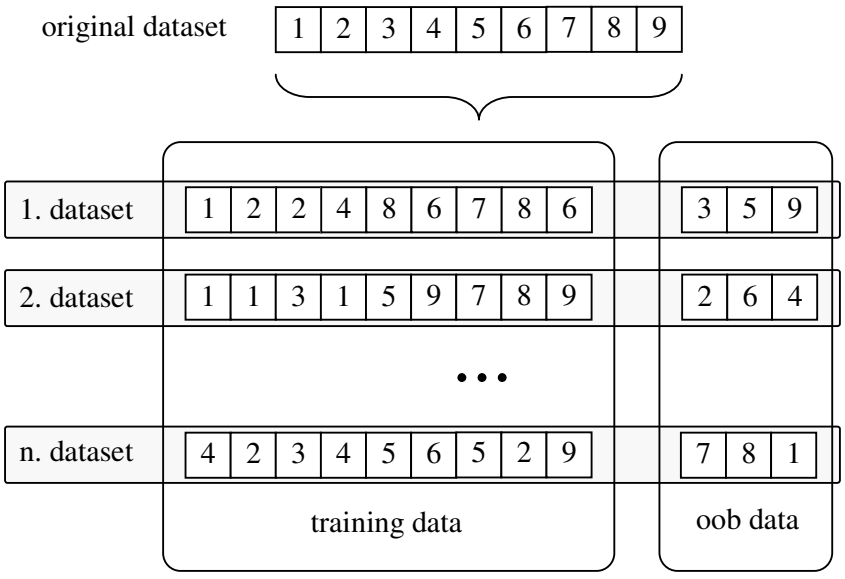

Figure 5-4. An example of the bootstrapping method

Random subspace method [8] is a method for choosing an attribute subset on which the search for the best split will be executed. The tree is then grown to its full extent without pruning in order to minimize its bias. After the process of growing the tree, its performance (accuracy) is assessed by passing the oob data down the tree. Growing a large number of such trees and repeating the oob error estimation for each of the trees, and finally combining all the estimates of oob errors of all the trees in the forest, gives the forest oob error rate, an unbiased error estimate for unseen data, which also excludes the need for separate test set error estimate.

Using the trained forest as a predictor is straightforward for a given input *X*. Each tree-structured classifier casts an output value which is differently processed, depending on the type of prediction. When classifying, the output value is a unit vote for the most popular class, and the class with the highest number of votes is chosen as a predicted class (known as plurality voting) while for regression, the output is a numerical value, and the result is average over all the outputs.

Ensemble methods are systematically better than single classification and regression methods. RF method has been proven as a computationally effective method with an excellent prediction performance.

5.2.1. RF algorithm

RF algorithm is a rather simple algorithm that relies on the procedure for tree growing (the pseudocode of the RF algorithm can be found in Appendix A1). It is important that the procedure of tree growing builds a controlled random tree rather than a pruned one (e.g. C4.5) in order to minimize its bias.

Growing random forest

The forest is grown to its full extent by iterative tree construction in the following three phases, also illustrated in Figure 5-5:

Bootstrap sampling

> Sample *N* instances (where *N* is the number of cases in the training set) at random from the training data with replacement (i.e. take a bootstrap sample). The bootstrap sample is used to grow the tree, the rest is out-of-bag data (i.e. *1/e* of the cases) that serves as a test set for the tree.

Tree growing

> Randomly select *m* variables out of *M* possible variables (independent for each node) and find the best split on those *m* variables using Gini index heuristics (see the equation 5.4 in Section 5.1) for impurity measure (the attribute with the highest Gini index is chosen as split in that node). Variable *m* is usually set automatically to $\sqrt{M}$ (it can also be set manually) and held constant during the forest growth. This procedure is known as "random subspace method" or "random feature selection" [8].

Grow the tree to the largest extent (maximum depth) possible without pruning. The stopping criteria is the point of trivial separation in every analyzed node.

Oob error estimation

Put each oob case (left out in the construction of the tree) down the tree to get classification. At the end of the run, take the class *c* that got most of the votes every time case *n* was oob. The proportion of times that *c* is not equal to the true class of *n* averaged over all cases is the oob error estimate. Oob error estimate has been proven as unbiased in many tests (as opposed to cross-validation which presents bias to an unknown extent).

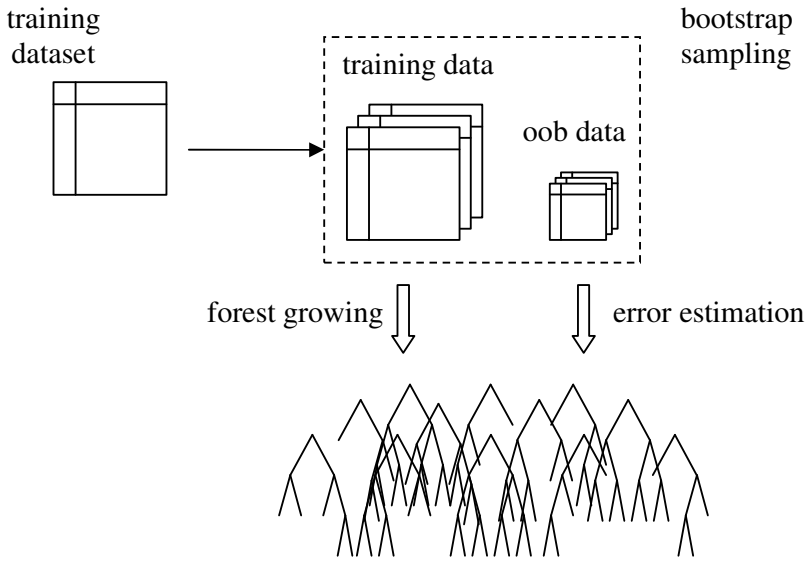

Figure 5-5. Steps in the process of growing the random forest

Using grown random forest for prediction

Using a grown random forest is rather straightforward. Unseen data is presented to the forest and each case is passed down every single tree in the forest and the corresponding tree results are collected. The final result depends on the type of prediction: classification or regression. In the case of classification, plurality voting is performed over the collected results of all the trees. Plurality voting is a single-winner voting system where the winner is chosen as a class with the most votes. As for regression, the final result is average of all the single tree results. The described process is illustrated in Figure 5-6.

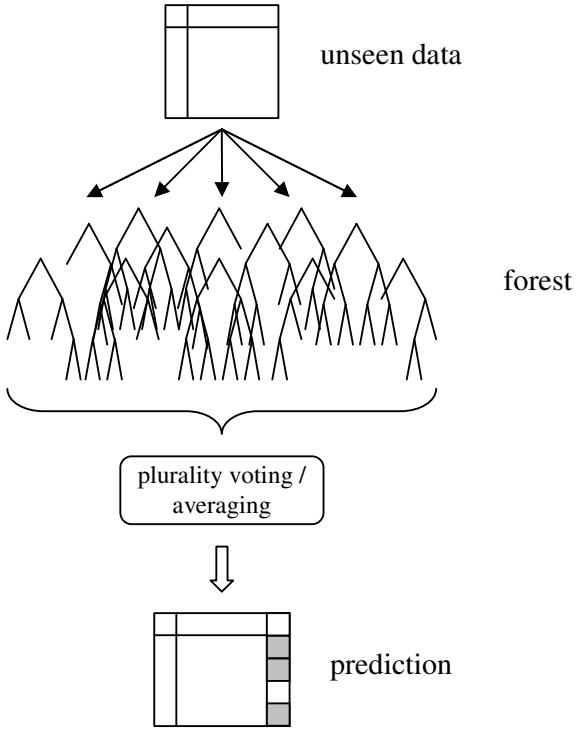

Figure 5-6. Using random forest for prediction

Formal definition

Given an ensemble of classifiers [7, 9]:

$$h_1(x), h_2(x), ..., h_k(x) \tag{5.6}$$

where $h_i(x)$ denotes a single classifier, and a training set Θ_k consisting of i.i.d. random vectors, drawn with replacement from the initial training set of N instances:

$$\Theta_k = \{(\mathbf{x}_n, y_n), \quad \text{i.i.d. from } (\mathbf{X}, Y), \quad n = 1, ..., N\} \tag{5.7}$$

where $(\mathbf{X}, Y)$ is a distribution of predictor variables $\mathbf{X}$ and response vectors Y. We formally define RF as an ensemble of classifiers consisting of a collection of tree-structured classifiers (5.6) built on training set (5.7) as:

$$\{h(x, \Theta_k), \quad k = 1, ..., K\} \tag{5.8}$$

where K denotes number of trees in a collection. In order to find functions $h_k(x)$ so that the prediction error of the ensemble is small, loss function is needed. The loss function for classification differs from the loss function for regression.

If Y represents unordered labels (i.e. nominal values), the problem is defined as classification and the loss function is *0/1* loss or margin function. Margin function measures the extent to which the average number of votes at $(\mathbf{X}, Y)$ for the right class exceeds the average vote for any other class; the larger the margin, the more confidence in classification. The margin function is therefore defined as:

$$mr(\mathbf{X},Y) = P_{\Theta}(h(\mathbf{X},\Theta) = Y) - \max_{j \neq Y} P_{\Theta}(h(\mathbf{X},\Theta) = j) \tag{5.9}$$

If Y is a numerical value, the problem is defined as regression and the loss function is mean-squared generalization error, generally used for numerical prediction:

$$E_{\mathbf{X},Y}(Y - h(\mathbf{X},\Theta))^2 \tag{5.10}$$

Using defined loss functions (5.9) and (5.10), generalization error for two cases can be defined. Generalization error for classification is given using defined margin function (5.9) with the next equation:

$$PE^* = P_{\mathbf{X},Y}(mg(\mathbf{X},Y) < 0) \tag{5.11}$$

and for regression defined using mean-squared generalization error (5.10):

$$PE^* = E_{\mathbf{X},Y}(Y - E_{\Theta}h(\mathbf{X},\Theta))^2 \tag{5.12}$$

Generalization error of a grown forest is estimated using oob data in the following manner:

$$Error(oob) = \frac{\sum_{n=1}^{N} \delta(c_n, y_n)}{N} \tag{5.13}$$

where c_n denotes true class for n-th instance, y_n denotes a class that gets the most votes every time when n-th instance is oob, and δ denotes a Kronecker's delta function. The oob error rate therefore measures the error rate of a forest as an averaged discrepancy of true class state and a predicted class state when the observed instance is oob.

Using a grown forest for prediction is based on plurality voting defined as:

$$y = \arg\max_{c} \sum_{k=1}^{K} (h(x,\Theta_k) = c) \tag{5.14}$$

where c denotes a possible class of prediction. The resulting class prediction y is equal to the class which has obtained the highest number of votes. As concerning regression, the result of it is the unweighed average of single-tree results over the forest:

$$y = \frac{\sum_{k=1}^{K} h(x,\Theta_k)}{K} \tag{5.15}$$

5.2.2. Paralel random forest

PARF (Paralel RF Algorithm) is a parallel RF algorithm implementation developed by G. Topić and T. Šmuc at Ruđer Bošković Institute. In contrast to the original code, written in Fortran 77, PARF is written in Fortran 90 which is a structured and parallel programming language. Parallelization procedures have

been accomplished by using MPI (Message Passing Library). In overall, the resulting code is much easier to read and use, than the original. PARF was financially supported by Ministry of Science, Technology and Sports (i-Project 2004-11) and is licensed under GNU GPL 2.0 license. More information about the implementation, including usage help and source code can be found on PARF's homepage: http://www.parf.irb.hr/en/.

As an ML method, RF algorithm demonstrated its excellence through important properties like accuracy, overfitting prevention, but also with additional features like missing value treatment, variable importance determination, multidimensional scaling (MDS) and others. With respect to parallelization, the core process of growing a large number of trees is inherently highly parallelizable. All these features were the reasons for making a new, parallel version of RF algorithm. In the following text, we elaborate these features in more detail.

Low time complexity

RF is computationally quite effective. It is able to learn a forest faster than bagging or boosting (e.g. growing 100 trees with RF is considerably quicker than growing 50 trees with AdaBoost [7]) and much faster than CART (e.g. a forest of 100 trees on a dataset with 100 variables can be grown in the same time as 3 single CART trees).

Example:

> On a system with 12 SPECint2006 rate, growing a forest of 500 trees on a dataset with 11061 instances and 42+1 attributes takes 3:15 minutes, and growing a forest of 5000 trees takes 36:50 minutes.

Large dataset handling

RF is successful in managing various spectra of data: high-dimensional data, data of unknown distribution, data with missing values, badly unbalanced data, etc. It is also very successful in handling thousands of categorical and continuous attributes without their deletion, as opposed to many ML methods which are inappropriate for processing that kind of data without a step of preprocessing called attribute selection. Handling thousands of attributes therefore makes RF suitable for a large number of unusual data, e.g. microarray analysis. Microarrays represent specific datasets characterized by a small number of instances (usually no more than 200) and a disproportionally large number of attributes (more than 20 000).

Missing values treatment

Random forest supports three different methods of dealing with missing values in the training set:

Ignoring missing values

> Missing values in the dataset won't be filled and the instances with missing values will participate in tree growing only on known attributes. When using

this method, algorithm is sometimes unable to build a forest on that kind of data. This is the option that should be avoided in practice.

Rough replacement

Rough fills (replacement values are called fills) are fills calculated on first pass. Missing values are filled according to their type; for non-categorical variables, all missing values are replaced with a median of all non-missing values of this variable in its corresponding class, and for categorical variables, missing values are replaced with the most frequent non-missing value in its class. This option should be used with care, and mostly in cases where the number of missing values per attribute is less than 5%.

Proximity-based fills

This is a missing value replacement method that relies on RF feature for calculating proximities between individual cases. It begins by doing a previously described rough filling of the missing values. After that, a forest is grown, and proximities between each pair of instance are calculated:

- Non-categorical missing values for a case are filled with an average of non-missing values of the corresponding attribute, weighted by the proximities between the case and the non-missing value cases.
- Categorical missing value cases are replaced by the most frequent non-missing value where frequency is weighted by proximity.

The algorithm then constructs the forest again using these newly filled-in values. These iterations can be repeated numerous times in order to obtain better-quality fills (according to algorithm's author, 4-6 iterations suffice). This method of replacement can be used only on the training set and is computationally more expensive, but gives much better performance, even with large amounts of missing data. This option should be used for the replacement of up to 25% of missing values per attribute.

As for a test set, two different methods of replacement exist, depending on whether labels exist for the test set.

- If they do, the fills derived from the training set are used as replacements,
- If they don't exist, each case in the test set is replicated number of classes times with its fill that corresponds to its class. This augmented test set is run down the tree and in each set of replicates, the one receiving the most votes determines the class of the original case.

Treatment of unbalanced datasets

RF implements procedures for balancing unbalanced datasets – datasets where a substantial imparity in class distribution of examples exists. Latter mentioned imparity usually causes so called default classification, i.e. all the samples are

classified as majority class. RF allows handling of unbalanced datasets through a user-defined set of weights per class. The examples corresponding to the weighted classes are appropriately weighted in the process of tree growing, while calculating Gini impurities. The use of weighting usually results in increase of error rate, but it also balances the class-wise error rates.

Example:

For a given dataset (nyha12-34)

Class 1, no. of instances: 13099

Class 2, no. of instances: 4563

Unbalanced dataset (weight 1:1)

Output:

Tag\Cl	NotCl	posit	negat
NoTag	0	0	0
posit	0	12596	503
negat	0	3803	760

Overall error rate: 24,38%

Class1 error rate: 3,84%

Class2 error rate: 83,34%

Imparity of class distribution reflects itself on to class-wise error rates.

Balanced dataset (1:2.8 weight ratio)

Output:

Tag\Cl	NotCl	posit	negat
NoTag	0	0	0
posit	0	8517	4582
negat	0	1521	3042

Overall error rate: 34,55%

Class1 error rate: 34,98%

Class2 error rate: 33,33%

The resulting class-wise error balancing is obvious.

5.2.3. RF - model interpretation tools

One of the most important features of ML algorithms is interpretability of results. In general, a ML method doesn't have to yield interpretable results (e.g. neural networks and SVM return prediction results without a possibility of interpretation), but when dealing with data from a domain where the understanding of data and the process of prediction is important (e.g. medical domain), interpretability surfaces out as a crucial requirement. RF is an example of a relatively simple ensemble predictor but with a number of additional interpretability tools [10], which emerge thanks to the ensemble approach to the classification problem. RF obtains this from the different statistical operations on the trees that make the forest: either comparing coincidence of attributes in the trees, or samples in the nodes of the trees. In the following text, we briefly introduce most important RF model interpretation tools.

Proximities

Proximities in RF are an intrinsic measure of similarity between pairs of cases. In comparison to Euclidean distance, RF proximity measure does not directly relate to spatial distance of cases in higher dimension. It is instead based on the way the forest deals with the data: proximity is equal to a proportion of trees for which two different cases end in the same terminal node of a tree when passed down the tree. As an example of measure difference, cases which are distant in Euclidean space can have high proximity because they stay together in the same terminal nodes in all the trees, or relatively low proximity if the cases are near the classification boundary. Computed proximities are generally very useful, and can be used for unsupervised learning, missing data replacement, MDS and outlier detection.

Variable importances (attribute ranking)

One of the features of random forest is the ability to obtain excellent estimates of variable importance which can then be used to gain valuable insight into the structure of data and the process of prediction. Variable importance is defined in terms of the contribution to prediction accuracy, i.e. predictive power of the variable – variables with high importance are more significant for the data because they describe differences among examples in different classes. Variable importance is especially significant when using data with a large number of attributes, e.g. satellite data, web data, EPOS, microarrays, medical data, etc.

RF implements two different ways for computing variable importance:

Fast variable importance

> Based on the fact that the split of a node made on a specific variable causes Gini index decrease of the descending nodes, fast variable importance is defined as the sum of all decreases in Gini impurity for each individual variable, normalized over all trees in the forest. The given measure is often very consistent with the permutation importance measure.

Permutation variable importance

Permutation variable importance differs substantially from fast variable importance because it is based on the original data vs. to randomly permuted data misclassification rate. For each tree, oob cases are passed down the tree and the votes cast for the correct class are collected. Then values of the observed variable are randomly permuted among oob cases. Those perturbed cases are then passed down the tree, and the votes for the correct class are collected. By subtracting the number of correct votes for the variable-m-permuted data from the number of correct votes for the untouched data (i.e. misclassification rate), and averaging them over all trees in the forest, raw importance score for the observed variable is obtained.

Correlation of these scores is proved to be quite low through a series of experiments, therefore standard errors can be computed and used: z-score (raw score divided by the standard error) and significance level to the z-score assuming normality. An example of variable importance is shown in Figure 5-7.

Imp	Z-Sc	Attribute name
1.46	34.46	EF
1.09	31.34	THIRDTONE
1.17	30.24	PreviousHospitalizations
0.98	26.81	PAS
0.97	23.39	CF
0.56	18.95	CREATIN
0.63	15.88	Age
0.46	15.22	LVTDD
0.36	14.49	LVTSD
0.39	14.22	HEMOGLOB
…		

Figure 5-7. Variable importance and their z-scores computed for the ANMCO dataset for the discrimination between patients in NYHA classes III and IV versus patients in NYHA classes I and II. Only first 10 attributes are shown.

Multidimensional scaling

Proximities between individual cases can be used to obtain squared distances in a Euclidean space of dimension not greater than the number of cases. These distances can then be used for a form of multidimensional scaling called metric scaling. Metric scaling is a process of approximating (i.e. lowering) a vector space to a lower dimension suitable for graphic representation (2D and 3D graphs). Despite the error, it inputs into the approximated data, metric scaling can give valuable insight into the structure of data via a simple interpretable graph. More accurate ways of projecting distances down to lower dimensions exist, but metric scaling performs quite satisfactory and although it is time consuming, it is the

quickest method for downscaling to lower dimensions. Important advantage of the RF metric scaling over more traditional approaches like PCA or SVD (principal component analysis or singular value decomposition) is that it involves incremental approach (i.e. does not involve matrix algebra), and is therefore more suitable for handling datasets with large number of attributes.

Prototypes

Prototypes are artificially constructed cases which can be used for inferences on the variable-to-classification relation. They are a kind of representative samples of corresponding classes (as seen in Figure 5-8). For a specific class, the case with the largest number of same class cases among its *k* nearest neighbours is selected. Among these *k* cases, median and quartiles are calculated for each variable. Prototypes for continuous variables are standardized median values and for categorical variables, prototypes are the most frequent values. For any subsequent prototype, the procedure is repeated but considering cases that are not among the original *k*.

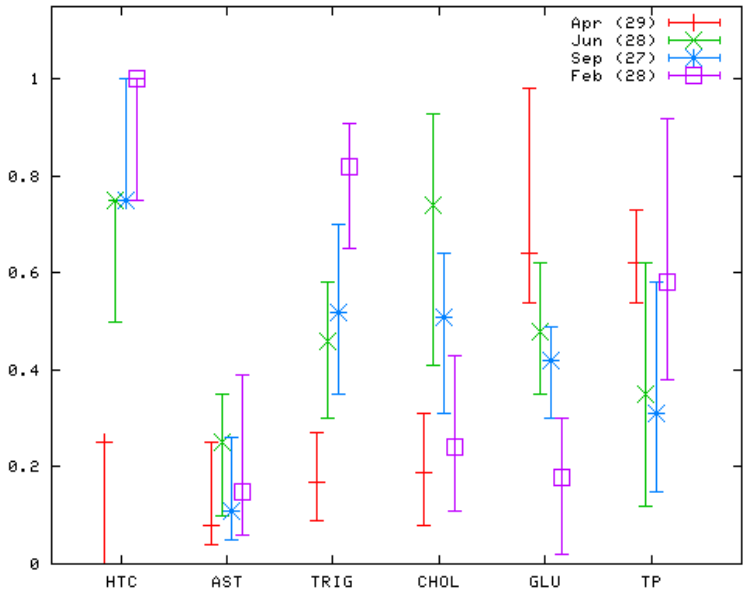

Figure 5-8. Illustration of PARF prototypes. Four most pronounced prototypes of the four classes in the problem of classification of seasonal blood biochemical data (Coz-Rakovac L, Smuc, T, et al., J. Appl. Ichthyol. (2007), 1-4) .

Outlier case detection

Outliers are defined as cases that are distant from the rest of the samples (data) related to their class. Small proximity between the case and the rest of the class data indicates a large outlier measure. Average proximity is defined as a sum of squared individual proximities. Outlier measure equals the median of average proximity (within each class) divided by their absolute deviation. By setting a threshold (usually around 10), outliers can be identified and therefore removed

from the main body of the data because they can be misleading to the final classifier and/or conclusion(s).

Variable interactions detection

In RF, by definition, two (independent) variables interact if a split on one variable in a tree makes a split on the other one either systematically less or more possible. These interactions are based on the Gini index values of an attribute for each tree in the forest. Variables are ranked on their Gini index and for each pair of variables the absolute difference of their ranks is averaged over all trees. Large positive number should imply that a split on one variable inhibits a split on the other one and conversely. This is an experimental procedure in original version of RF, and should be still cautiously used in PARF, since the reliable results can be obtained only with a large number of trees.

5.2.4. Unsupervised random forest

Unsupervised learning is a type of machine learning in which data has no explicit target output(s) and therefore no figure of merit to optimize. The goal of unsupervised learning is to find patterns, determine structure in the data and discern the ways in which the structure differs from pure unstructured noise. One of the forms of unsupervised learning is clustering, i.e. assignment of different groups (clusters) to observed objects (cases) in which data share common traits and can be assigned some meaning. These methods rely on distance measures between cases and are therefore sensitive to outliers and noise and are computationally (spatially) expensive. However, given that unsupervised learning has no figure of merit to optimize, it should be approached with great care while ambiguous conclusions might be drawn.

Random forest can be used as a method for unsupervised learning [10] although it is by itself a supervised method. The original data is to be labelled as *class 1*. Then a synthetic class of the same size as original data is created and labelled as *class 2*. The synthetic class is created by independent random sampling from univariate distribution of the original data (randomly selected values from all observed values of each of the corresponding M variables independently), as seen on Figure 5-9. Each of the variables of the *class 2* has the same univariate distribution as the corresponding variable of the original data (i.e. *class 1* data). The resulting synthetic class has the same marginal distribution as the original data but with destroyed dependencies between variables.

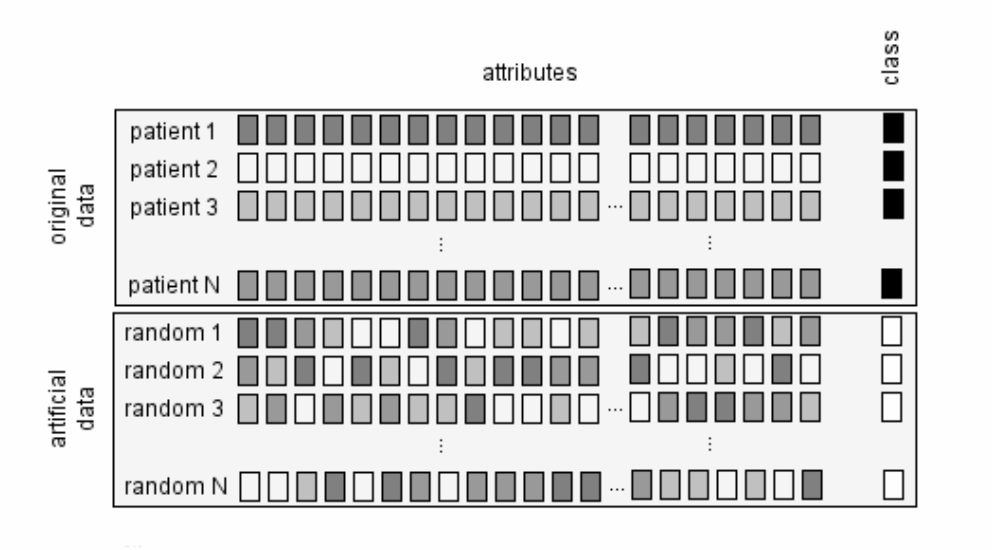

Figure 5-9. Data preparation of original data by addition of artificial random data for two-class problem of unsupervised learning using random forest

Artificial two-class problem (Figure 5-9) created in that manner can be processed with RF using all the interpretation tools that can be applied to the data like replacement of missing values, outlier detection, variable importance measurement and one of the most important – multidimensional scaling. MDS gives a valuable insight into data, enabling a simple view into its structure – if data contains clusters, MDS enables its easy inspection (if original data had labels, unsupervised MDS often retains the structure of the original data as shown in [11]).

If the oob error rate of the model/predictor built on that data is around 50%, RF cannot distinguish between the original data and synthetic data which implies that the original data looks like random sampling from M independent random variables and therefore no significant conclusions can be drawn. If the misclassification rate is lower, it implies that dependencies among variables exist and therefore RF interpretation tools can be used to learn about the structure of data.

Unsupervised learning with RF cannot be simply understood as a clustering method because while it is trying to discover structure, it may or may not discover structures which we usually think of as clusters.

5.3. Survival analysis

In many medical studies, time to a certain event (in many studies this event is death, hence the name survival analysis) what is of the utmost interest [12, 13, 14]. In some studies it is important to measure the time between response to treatment and recurrence or relapse-free time (also called disease-free survival time). What is important in the data collection process is to state what the event is and when the period of observation starts and finishes.

Survival data are generally described and modelled in terms of two related probabilities, namely survival and hazard. The survival probability (which is also called the survivor function) $S(t)$ is the probability that an individual survives from the time origin (e.g. diagnosis of heart-failure) to a specified future time t. It is fundamental to a survival analysis because survival probabilities for different values of t provide crucial summary information from time to event data. These values describe directly the survival experience of a study cohort.

The hazard is usually denoted by $h(t)$ and is the probability that an individual who is under observation at a time t has an event at that time. Put another way, it represents the instantaneous event rate for an individual who has already survived to time t. Note that, in contrast to the survivor function, which focuses on not having an event, the hazard function focuses on the event occurring. In summary, the hazard relates to the incident (current) event rate, while survival reflects the cumulative non-occurrence.

The specific difficulty related to the survival type of follow-up-in-time studies (or survival analysis) is that only some of the followed individuals experience the target event. Therefore, in principle, survival times will be unknown for a subset of the group of patients in the stud. This problem or phenomenon is called censoring and it may arise in the following ways:

a) patient has not (yet) experienced the target outcome by the time of the closing of the follow-up study;

b) a patient is lost to follow-up study during the study period;

c) a patient experiences another event that makes further follow-up impossible

In all three cases we deal with so called censored survival times, and if these situations are not properly taken care off, they tend to underestimate the true but unknown time to event. There are different types of censored data (left, interval), but in most of the situations, one has to deal with so-called *right censored* data. If we visualize the survival process of an individual on a time-line, their event is to happen beyond the end of follow-up period.

The survival probability can be estimated non-parametrically from observed survival times, both censored and uncensored, using the KM (or product-limit) method (Kaplan and Meier, 1958). Suppose that k patients have events in the period of follow-up at distinct times $t_1<t_2<t_3<t_4<t_5<\ldots<t_k\ldots$ As events are assumed to occur independently of one another, the probabilities of surviving

from one interval to the next may be multiplied together to give the cumulative survival probability. More formally, the probability of being alive at time t_j, $S(t_j)$, is calculated from $S(t_{j-1})$ the probability of being alive at t_{j-1}, n_j the number of patients alive just before t_j, and d_j the number of events at t_j, by:

$$S(t_j) = S(t_{j-1}) \cdot (1 - \frac{d_j}{n_j}) \tag{5.16}$$

where $t_0=0$ and $S(0)=1$. The value of $S(t)$ is assumed to stay constant between times of events, and therefore the estimated probability is a step function that changes value only at the time of each event.

There is a clearly defined relationship between $S(t)$ and $h(t)$, which is given by the expression:

$$h(t) = -\frac{d}{dt}[\log(S(t)] \tag{5.17}$$

The expression simply states that when $S(t)$ is known it is straightforward to determine $h(t)$.

One of the key requirements for the analysis of survival data to be valid is that censoring is "non-informative". In practical terms, this means that censoring carries no prognostic information; in other words, those who are censored because of loss to follow-up at a given point in time should be as likely to have a subsequent event as those individuals who remain in the study.

Informative censoring may occur when patients withdraw from a study of worsening clinical condition. Standard methods for survival analysis are not valid when there is informative censoring, but, when the number of patients lost to follow-up is small, very little bias is likely to result from applying methods based on non-informative censoring.

In survival analysis, probabilities are calculated not just for groups but also for individuals in a group, a major advantage for example to use such analysis or models for single patient prognosis.

5.3.1. Machine learning tools for survival analysis

Although Kaplan-Meir method is a default tool for survival data analysis, it does not offer the interpretability and cannot treat prospective dataset problems as efficiently as more modern, machine learning based methods. For obvious reasons in this deliverable, we will introduce a version of the random forest algorithm capable of dealing with survival type of data – survival random forest.

5.3.2. Survival random forest

Random Survival Forests (RSF) are a variant of random forests and naturally inherit many of its good properties. Two features especially important in the context of survival analysis are:

(1) User-friendliness - only three, fairly robust, parameters need to be set (the number of randomly selected predictors (very robust), the number of trees grown in the forest (the larger the better), and the splitting rule to be used).

(2) RF is highly data adaptive and virtually model assumption free. This last property is especially helpful in survival analysis. Standard analyses often rely on restrictive assumptions such as proportional hazards. Also, with such methods there is always the concern whether associations between predictors and hazards have been modelled appropriately, and whether or not non-linear effects or higher order interactions for predictors should be included. In contrast, such problems are handled seamlessly and automatically within a random forests approach.

The main additions/changes that distinguish RSF from RF are:

- Splitting rules: since nodal splitting is a crucial part of RF, it is clear that for survival analysis this has to be changed. In RSF node splitting includes 4 different survival splitting criteria: (i) log-rank (ii) conservation of events rule (iii) log-rank score and (iv) fast approximation of log-rank rule;
- The predictor variable in RSF is cumulative hazard function and coupled to this specific variable is the error estimation – in RSF it is so called concordance index.

These are the main parts of the algorithm that are significantly different from original RF classifier. However, much more details can be obtained in [15].

Outcomes from RSF and their use for prognosis

Similarly to standard random forest, RSF generates a number of different outputs. The information about variable importance (variable importance ranking) and behaviour of the error rate with respect to the forest size, illustrated in Figure 5-10, is practically the same information as obtained with RF.

However, Figure 5-11 and Figure 5-12 illustrate information unique to RSF algorithm.

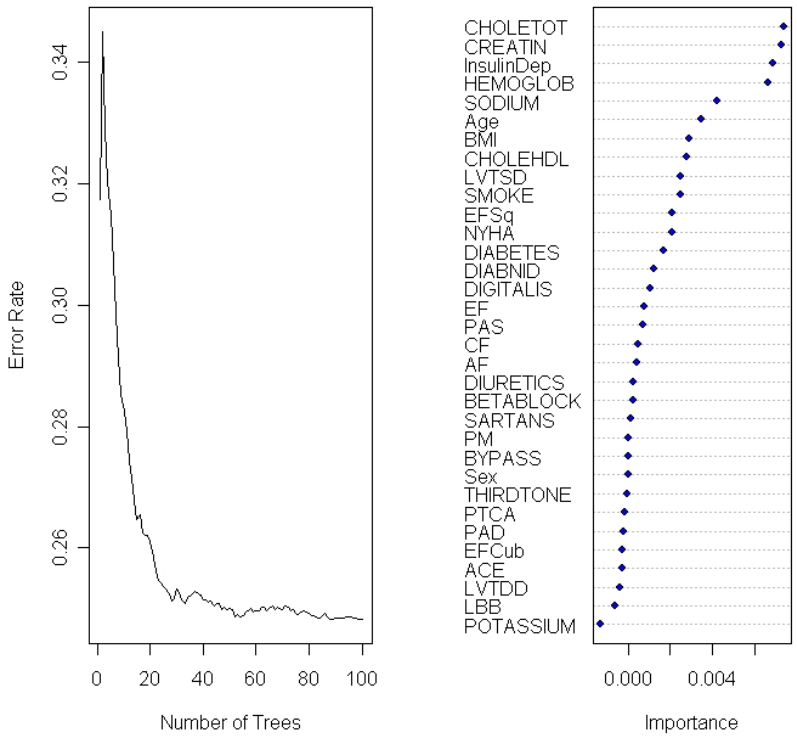

Figure 5-10. Standard outcomes of RSF analysis. On the left is error rate vs. number of trees in the forest. On the right is variable importance plot.

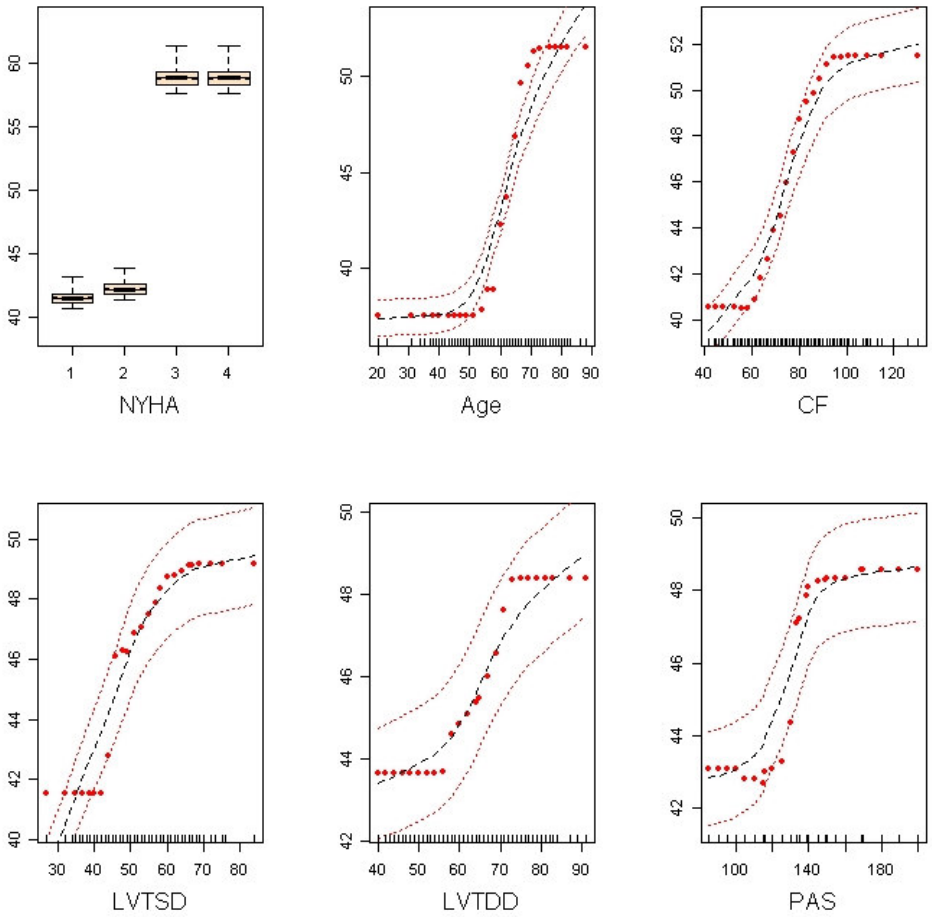

Figure 5-11. Plots showing correlation between expected number of deaths and value of the particular variable (here NYHA, age, CF, LVTSD, LVTDD and PAS) for a small fragment of data .

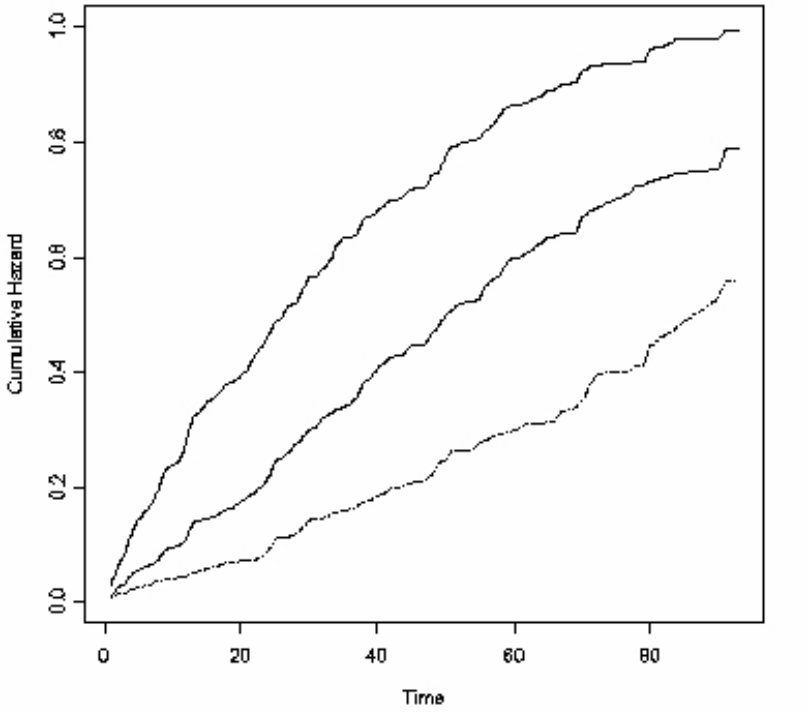

Figure 5-12. Illustration of individual patient prognosis (cumulative hazard function) based on RSF model.

We can conclude that RSF algorithm and in particular, the results of the analysis of survival type of datasets with RSF algorithm, provide valuable information for the support in prognostic type decision making. Moreover, the RSF model or survival forest as a main result, can directly be used in assessing individual patient hazard rate.

5.3.3. Conditional probability trees

Although random forests algorithm has many features that give important insights into relations hidden in data, and finally provide models for classification comparable to any other advanced classification algorithm, the actual model – i.e. random forest is not neither tractable nor directly interpretable. This characteristic of RF extends naturally to RSF.

In order to have human readable models one has to use other type of algorithms, that can deliver interpretability, in principle on the expense of the quality of the model with respect to accuracy or other evaluation measures. However, in many situations, simpler algorithms give quite good results, and in such situations, they have the advantage of providing simultaneously the interpretability.

In order to have an algorithm providing human readable models in the context of survival analysis we have added conditional probability trees (CPT) [16] to our algorithm portfolio (an example of CPT is shown on Figure 5-13).

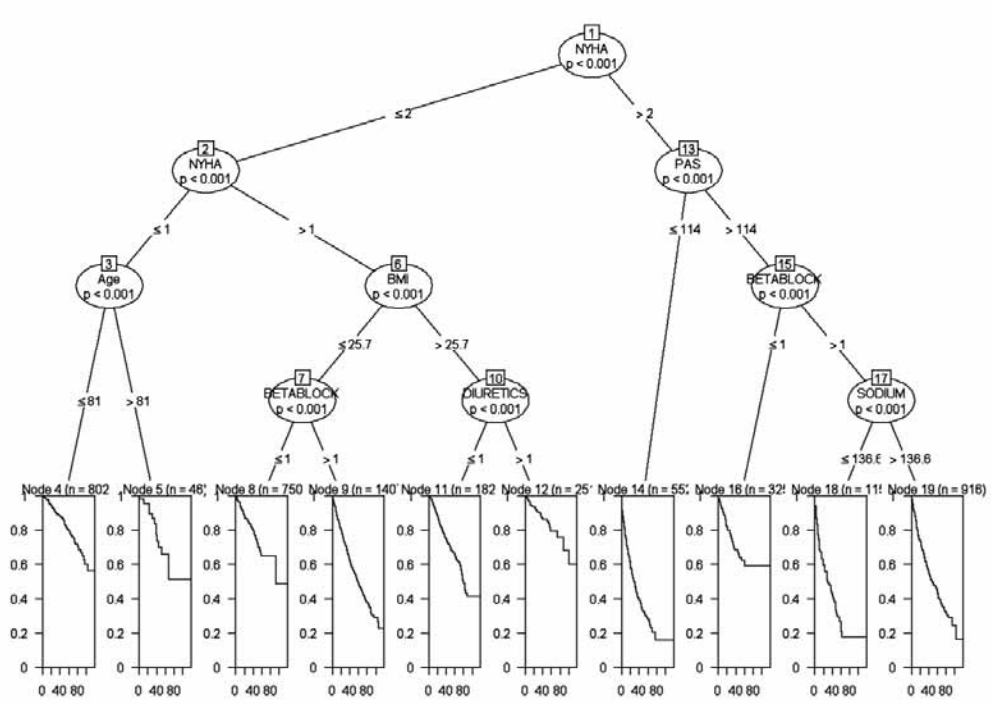

Figure 5-13. Illustration of decision tree for survival analysis type of data as produced by the conditional probability trees (CPT) algorithm

5.4. Decision lists

If-then rules are the basis for most popular concept description languages used in machine learning [17]. They provide an explicit representation of the knowledge extracted from a dataset that is easy for people to understand. This has a number of advantages: rules are generally compact, modular and explicit, plus they can be analyzed by domain experts, checked for plausibility and combined with previously known facts about the domain.

Among the rule learning algorithms there are two different approaches: one adopts the "divide-and-conquer" strategy (in particular we are interested in an approach based on extraction of rules after a decision tree construction), and the other adopts the "separate-and-conquer" strategy (also known as the covering approach). Two dominant and well known implementations of the aforementioned approaches are respectively C4.5rules and RIPPPER, both performing global optimization process on the set of rules that is induced initially. Another algorithm known as PART (Partial decision trees) algorithm combines these two different approaches for learning rules, however, it avoids global optimization but nevertheless produces accurate and compact rule set.

C4.5rules

C4.5rules initially creates an unpruned decision tree and then transforms it into a rule set by generating one rule for each path from root to a leaf. Most rule sets derived in this way can be simplified without losing predictive accuracy. Each rule is simplified separately by greedily deleting conditions in order to minimize the rule's estimated error rate. The rules for each class in turn are considered and a "good" subset is sought, guided by a criterion based on minimum description length principle. The next step ranks the subsets for the different classes with respect to each other to avoid conflicts, and determines a default class. Finally, rules are greedily deleted from the whole rule set one by one, so long as this decreases the rule set's error on training data. The whole process is complex and time consuming. Moreover, despite the lengthy optimization process, rules are still restricted to conjunctions of those attribute-value tests that occur along a path in the initial decision tree.

RIPPER

RIPPER [18] is an implementation of the rule learning approach based on the "separate-and-conquer" strategy, representing a more direct approach to learning decision rules. The algorithm generates one rule at time (the most powerful rule underling the dataset) and removes all the instances covered by it, and iteratively repeats the procedure on remaining examples. In a multi-class setting, this automatically leads to an ordered list of rules, a type of classifier that has been termed "decision list". RIPPER implements this strategy using reduced error pruning, which sets some training instances aside in order to determine when to drop the tail of a rule, and incorporates a heuristic based on the minimum description length principle as stopping criterion. The algorithm considers "replacing" or "revising" individual rules, guided by the error of the modified rule set. It decides whether to leave the original rule or to use its replacement or revision. The decision is made according to the minimum description length heuristic. The basic strategy for building a single rule and pruning it back can lead to a problematic form of overpruning, which is called "hasty generalization". It is the consequence of the fact that pruning interacts with the covering heuristic.

PART

PART [17] adopts the separate-and-conquer strategy so that it generates a rule, removes all the instances covered by it, and continues building new rules recursively from the remaining examples, until none are left. It differs from the standard separate-and-conquer approach (RIPPER implementation) in the way that each rule is generated. In order to create a single decision rule a pruned decision tree is built for the current instances, the path from root to the leaf with the largest coverage is transformed into a rule, and the tree is discarded. This avoids hasty generalization by only generalizing once the implications are known (i.e., all the subtrees have been expanded).

Using a pruned tree to obtain a rule instead of building it incrementally by adding conjunctions one at time avoids the over-pruning problem of the basic

separate-and-conquer rule learner. Using separate-and-conquer approach in conjunction with decision trees adds flexibility and speed. It is indeed wasteful to build a full decision tree just to obtain a single rule, but building a "partial" decision tree instead of a fully one can accelerate the process without sacrificing the above advantages. The tree-building algorithm splits a set of instances recursively into a partial tree (Figure 5-14). The first step chooses a test and divides the instances into subsets accordingly; PART makes this choice in the same way as C4.5. Then the subsets are expanded in order of their average entropy, starting with the smallest entropy. This continues recursively until a subset is expanded into a leaf. But as soon as an internal node appears which has all its children expanded into leaves, pruning begins: the algorithm checks whether that node can be replaced by a single leaf. PART performs the standard "subtree replacement" operation of decision-tree pruning in the same way as C4.5 (as described in Section 5.1). If replacement is performed the algorithm backtracks in the standard way, exploring siblings of the newly-replaced node. However, if during backtracking a node is encountered whose children are not leaves then the remaining subsets are left unexplored and the corresponding subtrees are left undefined. Due to the recursive structure of the algorithm, this event automatically terminates tree generation.

A node can only be pruned if all its successors are leaves, and this can only be happen if all its subtrees have been explored and either found to be leaves, or are pruned back to leaves. This ensures that over-pruning effect cannot occur.

Once a partial decision tree has been built, a single rule is extracted from it. Each leaf correspond to a possible rule, and the algorithm searches for the "best" one of those subtrees that have been expanded into leaves. PART aims at the most general rule choosing the leaf that covers the greatest number of instances.

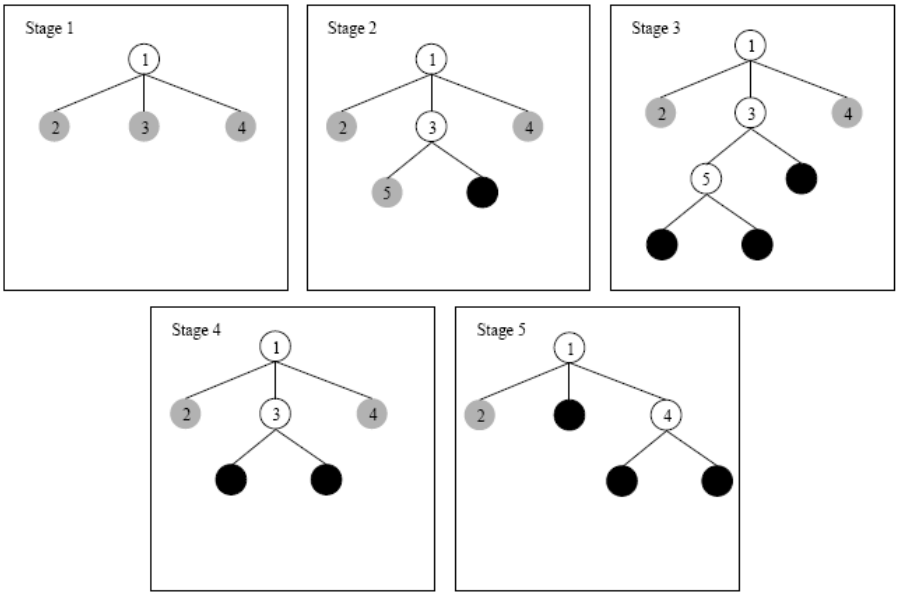

Figure 5-14. Example of how PART builds a partial decision tree [17]

When constructing a partial decision tree PART treats missing values in the same way as C4.5. If an instance cannot be assigned deterministically to a branch because of a missing attribute value, it is assigned to each of the branches with a weight proportional to the number of training instances going down that branch, normalized by the total number of training examples with known values at the node.

5.5. Subgroup discovery

Rule learning and decision tree learning are two most common approaches to symbolic predictive induction. In decision tree learning, the concepts which can be formed of paths leading from the root node to class labels in the leaves represent descriptions that best discriminate between the classes. On the other hand the goal of rule learning is to generate separate models, one for each class, inducing class characteristics in terms of class properties occurring in the descriptions of training examples. Classification rule learning results in characteristic descriptions, usually generated separately for each class by repeatedly applying the covering algorithm. The distinguishing property of both approaches is that the induced models can be understood and verified by humans. It means that they are potentially very useful for different intelligent data analysis applications. Also, after human verification the models can be combined with already existing domain knowledge for decision support purposes. Such intelligent data analysis and decision support applications fall in the broad field of *descriptive induction* tasks. Simplicity and human interpretability of the results are the main requirements for these tasks in contrast to classification tasks in which good prediction quality is the ultimate goal.

The models obtained by symbolic predictive induction are actionable in terms of determining the class membership of individual non-labelled instances but not necessarily in the sense of descriptive induction which means uncovering the properties of subpopulations which can guide a decision maker in directing some actions or understanding their distinguishing characteristics [19]. The reasons are:

a) Rules formed of decision tree paths are *discriminant* descriptions; hence they are not actionable for the above tasks of understanding coexisting properties characteristic for subpopulations.

b) Classification rules generated by a covering algorithm have the property that only first few rules induced may be of interest as subpopulation descriptions with sufficient coverage. Subsequent rules are induced from smaller and strongly biased example subsets, e.g., subsets including only positive examples not covered by previously induced rules. This bias prevents the covering algorithm to induce descriptions uncovering significant subgroup properties of the entire population.

c) Both discriminating descriptions obtained through decision tree learning and characteristic descriptions obtained by classification rule learning tend to have relatively low prediction quality in the situation when their

complexity is strongly limited in order to ensure human interpretability of the results. The problem with all types of the induction of low dimensional non-redundant classifiers is that they are very sensitive to training set overfitting (described in Section 4.4.1). Although maximal prediction accuracy is not the main goal of descriptive induction tasks, high generalization error or large difference in prediction quality for the training and the test set are a reliable sign that the induction of a classifier was not successful in finding really relevant relations between attribute values and the classes.

Subgroup discovery (SD) is a type of inductive rule learning specifically targeted at descriptive induction. It differs from classification rule learning in a few small but relevant details. The most relevant are: application of weighted covering approach with intention to ensure global significance of induced concepts, introduction of the parameterized rule quality measure enabling induction of models with different covering properties, and systematic application of the relevancy constraints that should help in overfitting prevention. We start with the presentation of the standard covering approach to rule learning and then we continue by presenting the differences that characterize subgroup discovery approach. We end with the presentation of the process for the detection of supporting factors which is, strictly speaking, not the part of the subgroup discovery approach but which significantly helps in human interpretation of the obtained results.

5.5.1. Covering rule learning

The covering rule learning approach iteratively constructs a set of rules that build a hypothesis describing differences between examples in the positive (target) class in contrast to all other examples (negative or non-target class). The approach constructs one by one rule until all positive examples are covered by the hypothesis. In each its iteration the covering approach uses the procedure *LearnOneRule*. After a rule is added to the hypothesis, examples covered by that rule are deleted from the current set of examples.

---

procedure LearnOneRule
Input: set of positive and negative examples
Procedure:

Rule : positive class ← Conditions, where Conditions := ∅
repeat
    build all refinements for Rule
        {positive class ← Conditions AND SomeFeature}
    evaluate the refinements according to a quality criterion
    Rule := the best refinement
until Rule satisfies a quality threshold or covers no examples from negative class
Output: Rule

Figure 5-15. A generic procedure for rule induction.

The procedure *LearnOneRule* presented in Figure 5-15 is the principal component and the main iterative loop of any rule learning approach. It stresses three basic aspects of rule induction: rules are built as logical combinations of features, we need a measure to evaluate and compare the covering quality of rules, and we need some stopping criteria (quality threshold) which prevents overfitting. The iterative process of adding features goes on as long as the rule covers also some negative examples but the process can be stopped if adding features cannot increase the rule quality or the stopping criteria has been reached. The fundamental differences among the rule learning approaches is in the way that these aspects have been solved. In this deliverable we concentrate only on the presentation how they are solved in the subgroup discovery methodology and on its differences to classification rule learning. The overview and analysis of other rule learning approaches can be found in [20].

5.5.2. Feature generation

The features that are used for rule refinements can be constructed inside the *LearnOneRule* procedure from attributes that describe examples. That is a standard approach implemented in most rule learning systems. Subgroup discovery uses a more systematic approach. The complete set of features is generated before the rule learning actually starts. The drawback of the approach is the extra memory space necessary to save the features. This space can be rather significant for complex domains with many attributes.

---

procedure GenerateFeatures
Input: set of positive and negative examples
Procedure:
for each attribute A_i do
  V := set of all different values of A_i appearing in the positive examples
  W := set of all different values of A_i appearing in the negative examples
  if A_i is a discrete attribute
    for each v in V generate feature $\{A_i = v\}$
    for each w in W generate feature $\{A_i \neq w\}$
  endif
  if A_i is a continuous attribute
    sort the attribute values
    for each possible pair (v,w), v in V and w in W
      if v < w and does not exist y: (v < y < w) generate feature $\{A_i<(v+w)/2\}$
      if v > w and does not exist y: (v > y > w) generate feature $\{A_i>(v+w)/2\}$
  endif
endfor
Output: FeatureSet

Figure 5-16. Feature generation for a given set of positive and a given set of negative examples.

The procedure presented in Figure 5-16 is repeated for every available attribute. If the attribute is discrete then all distinct values in positive examples are detected and for them features of the form *Att=value* are generated. Also all distinct values for negative examples are detected and from them features of the form *Att≠value* are generated. The number of generated features for a discrete attribute is equal to the sum of distinct attribute values occurring in positive and negative examples.

For a continuous attribute, the features are generated in the following way: We identify pairs of neighbouring values, where neighbouring means that there is no other value between them. From these pairs, we compute the mean of the two neighbouring values *mean_value*. If there are two neighbouring values from different classes, then if the smaller of the two values is from the positive class we generate the feature *Att<mean_value*, while if the smaller of the values is from the negative class we generate the feature *Att≥mean_value*. This part of the feature generation algorithm can be implemented in a more efficient way, similar to the algorithm suggested by [21], by first sorting the attributes values, independently of the example class in an increasing value list, and then considering only neighbouring pairs in this list. In this case, additional precautions have to be made for the case when multiple examples with possibly different class labels have identical attribute values. The number of features generated for a continuous attribute depends on the grouping of classes in the increasing value list but typically the number of generated features is proportional to the number of examples.

There are some cases when it is completely justified to generate features of the form *Att=value* for numerical attributes. This is the case in situations when attribute values are integers representing distinct, well specified and disjunctive concepts, such as floors in a building, or school years as educational levels. In such cases, it is a good practice to treat such attributes as being continuous and discrete, resulting in both types of features.

Besides speed, the main advantage of generating features by the procedure presented in Figure 5-16 is the possibility to test the features on their relevancy and to eliminate irrelevant features already in the preprocessing, even before entering the rule learning process. By eliminating irrelevant features, the quality of the complete rule learning process can increase.

5.5.3. Feature irrelevancy

There are three types of feature irrelevancy [22].

The first it *total irrelevancy*. Totally irrelevant features are those that discriminate no positive from negative examples and as such they are useless for the induction process.

Constrained irrelevancy eliminates features that correctly cover very low number of either positive or negative examples. These features can construct only very specific rules and as such, they can lead to the hypotheses potentially overfitting the training set.

For the quality of the induced rules decisive is *relative irrelevancy* that eliminates all features of lower covering quality if better features are available. In this way, the relevancy approach ensures that only the best features can enter the rule induction process.

It is very important to notice that the concept of relevancy is applicable also to the logical combinations of features as well as for complete rules. Relevancy checking of all intermediate results is consistently performed in the subgroup discovery procedure presented in Figure 5-18.

5.5.4. Rule quality measure for subgroup discovery

Result of the rule induction process significantly depends on the quality measure used to evaluate the rule refinements. Differently defined quality measures influence the order in which features are added into the rule body. That is the reason that used rule quality measures are also called search heuristics. The most significant difference among rule learning strategies is actually in respect to the definition of the implemented rule quality measure.

Often classification accuracy measure is used. It is defined as the fraction of all (both positive and negative) correctly classified examples. Although this measure is very appropriate for measuring the quality of complete hypothesis, its application for induction of a single rule can lead to non-optimal solutions. Rule accuracy, also called precision, is in this respect much better. It is defined as the fraction of positive examples among all examples covered by the rule body. A very reasonable measure is also weighted relative accuracy [23].

Although there can be rather large differences in the way the rule quality measures are defined, all classification rule learning approaches are characterized by the fact that in the concrete situation they use only one of them and that it is aimed at the optimization of the prediction quality. In contrast to that, subgroup discovery approach uses a quality measure with user selectable generalization parameter representing the notion of user expected generality of the induced rules. In this way, by changing the value of the parameter the user can search the complete space of possible solutions, from very specific to very general ones. It means that induction problem has not an unique solution what is completely in line with the goals of descriptive induction aimed at detecting many, possibly all relevant relations existing in the available dataset.

The rule measure used in the SD approach is defined as

$$q = \frac{|TP|}{|FP| + g} \tag{5.20}$$

where q is the rule quality, $|TP|$ is the number of true positive cases (number of positive examples (correctly) covered by the rule body), $|FP|$ is the number of false positive cases (negative examples (erroneously) covered by the rule body, and g (coming from generality) is the user selectable parameter. According to this measure, high quality rules will cover many examples from the positive class and a low number of negative examples. The number of tolerated negative cases,

relative to the number of covered positive cases, is determined by parameter *g*. For low *g* (*g<1*), induced rules will have high specificity (low false alarm rate) since covering of every single negative example is made relatively very "expensive". On the other hand, by selecting a high *g* value (*g>10* for small and *g>50* for large domains), more general rules will be generated, covering also some negative instances.

5.5.5. Weighted covering approach

Removing examples during training, as done by the covering algorithm, distorts the training set statistics and introduces order dependencies between rules. For instance, the last rule learned is heavily dependent on the previous rules and the positives they cover, and it may not be meaningful (or statistically significant) when interpreted individually.

A remedy to this problem is the use of a weighted covering algorithm in which the subsequently induced rules (i.e., rules induced in the later stages) also represent interesting and sufficiently large subgroups of the population. The weighted covering algorithm (presented in Figure 5-17) modifies the classical covering algorithm in such a way that covered positive examples are not deleted from the current training set. Instead, in each run of the covering loop, the algorithm stores with each positive example a count indicating how often (with how many rules) the example has been covered so far. Initial weights *c(e)* of all positive examples are equal 1 denoting that the example has not been covered by any rule. Lower weights between 0 and 1 mean "do not try too hard on this example". Consequently, the examples already covered by one or more constructed rules decrease their weights while the uncovered target class examples whose weights have not been decreased will have a greater chance to be covered in the following iterations of the algorithm. The result is that every induced rule will be as general as possible even if this means that some rules will overlap.

---

procedure WeightedCovering
Input: set of positive and negative examples
Procedure:

set of rules ← ∅
weight *c(e)* for every positive example ← 1
repeat *n* times
    call procedure SD to find the best *Rule* for the current example weights
    set of rules ← set of rules *U Rule*
    decrease the weight of positive examples covered by the *Rule*
until
Output: Set of relevant subgroups

Figure 5-17. The weighted covering procedure

For a weighted covering algorithm to be used, we have to specify the weighting scheme, i.e., how the weight of each example decreases with the increasing

number of covering rules. In the so-called additive weighting scheme, weights of covered positive examples decrease according to the formula $c(e)=1/(i(e)+1)$, where $i(e)$ is the number of rules already covering example e. In the first iteration all positive examples have $i(e)=0$ and in this way they contribute the same weight $c(e)=1$. In the following iterations the contributions of examples are inversely proportional to their coverage by previously induced rules.

Note that example weights also need to be appropriately incorporated into the heuristics used for rule quality evaluation. This provides the means to consider different parts of the example space in each iteration of the weighted covering algorithm. Specifically, the original subgroup discovery parameterized rule quality measure (X) assumes equal weight 1 for all positive examples. Now, in the weighted covering framework it must be accordingly modified into

$$q = \frac{\sum_{TP} c(e)}{|FP| + g} \tag{5.21}$$

This measure is actually used in the SD procedure presented in the next section.

5.5.6. SD procedure

Subgroup discovery algorithm consists of the outer loop in the form of the weighted covering procedure (Figure 5-17) and the inner loop called the SD procedure (Figure 5-18) which is invoked in each iteration of the outer loop. The features necessary for this procedure are constructed in preprocessing by the procedure described in Section 5.5.2. The used stopping criteria is the maximal number features that can be included into the rule body [24].

In contrast to the *LearnOneRule* procedure presented in Section 5.5.1, the SD procedure uses the beam search. It means that there is not one, but potentially many intermediate results that represent ideal solution of each iteration. In this way myopic property of any heuristic search can be significantly reduced. The only problem is that the increased size of the beam significantly increases the time complexity of the procedure. The SD procedure actually uses two beams, Beam which holds current best results and *NewBeam* in which we save outputs of the current iteration.

The procedure (presented in Figure 5-18) begins by the initialization of all the rules in Beam and *NewBeam* by empty rule conditions and their quality values q to the default value. The rule initialization is followed by the main procedure loop that stops when, for all rules in the Beam, it is no longer possible to further improve their quality or when the maximal rule complexity is reached. Rules can be improved only by conjunctively adding features from F. After the first iteration, a rule condition consists of a single feature, after the second iteration up to two features, and so forth. The search is systematic in the sense that for all rules in the beam all features from F are tested in each iteration. For every new rule, constructed by conjunctively adding a feature to rule body quality q' is computed. If the support of the new rule is greater than *MinSup*, if its quality q' is greater than the quality of any rule in *NewBeam*, and if the new rule is relative relevant

with respect to the rules that are already in the *NewBeam*, the worst rule in *NewBeam* is replaced by the new rule. The rules are reordered in *NewBeam* according to their quality. At the end of each iteration, *NewBeam* is copied into Beam. When the procedure terminates, the first rule in Beam is the rule with maximum quality and it is the output of the SD procedure that is used by the weighted covering procedure and included into its set of relevant subgroup descriptions (Figure 5-17).

---

procedure SD
Input: set of examples $E=PosUNeg$, set of features F
Parameters: g, *BeamWidth*, *MinSup*, *MaxRuleLength*
Procedure:
for each *Rule* in Beam and NewBeam do
    positive class ← Conditions, where Conditions := ∅
    $q=(\sum_{P} c(e))/(N+g)$
end for
repeat while there are improvements in Beam
        and number of iteration ≤ *MaxRuleLength*
    for each *Rule* in Beam do
        for each feature f in F do
            NewRule ← *Rule* AND f
            $q'=(\sum_{TP} c(e))/(|TP|+g)$
            if (support(*Rule*) ≥*MinSup)* and (Rule is relevant) and
                q' is larger than the quality of any rule in NewBeam do
                replace the worst rule in NewBeam with *Rule*
        end for
    end for
    copy NewBeam →Beam
until
Output: all rules from the Beam

Figure 5-18. The main subgroup discovery rule learning procedure.

The SD procedure implements level-wise search what means that in each iteration of the main loop all possible refinements of all rules from the Beam are tested. That is important because in this way maximal complexity of the rules, defined by parameter *MaxRuleLength*, can be easily controlled by limiting the number of iterations of the main loop of the SD procedure. For descriptive induction applications obtained solutions should be simple for domain expert interpretation and acceptable number of features included into the rules is typically limited to up to four.

The procedure also strictly implements the concept of relevance of features and rules. At first it means that even before the start of the procedure, all irrelevant features are excluded from the set of features F, as described in Section 5.5.3. Tested and applied are conditions for total, relative, and constrained feature

irrelevancy. But, as already noted, the concept of relevancy is applicable also on logical combinations of features and this is implemented in SD procedure. The constrained relevancy is controlled by parameter *MinSup*. Relative relevancy is ensured by the condition that no feature or feature combination can enter *NewBeam* if *NewBeam* already contains another feature or feature combination that is equally or more relevant. Additionally, after inclusion of a new feature or feature combination it is immediately tested if perhaps there are some features or feature combinations that are less relevant than the entering one. Such features or feature combinations are deleted from the *NewBeam*. This approach ensures the significance of constructed subgroups and helps in overfitting prevention.

5.5.7. Supporting factors for descriptive analysis

Short rules obtained by the subgroup discovery approach are convenient for human understanding and interpretation. But applications like human decision making process based on these subpopulations or human understanding of the subpopulations require much more information than a few features contained in the rule body can give. The solution is statistical evaluation of detected subpopulations. This analysis leads to a set of properties called *supporting factors*, in contrast to conditions that are contained in rules and which are called *principal factors* of detected subgroups. The number of supporting factors is typically not limited, and together with principal factors, they can give a lot of very valuable information about relevant properties of subpopulations that have been detected by the subgroup discovery process [25].

Statistical evaluation of subgroups differs from standard statistical analysis it two important details. The first is that unlike standard analysis which compares statistical properties of all positive class examples versus complete negative class, statistical evaluation of subgroups compares properties of the *set of positive examples in the subgroup* versus all negative cases. The difference is small but significant because it can happen, and it happens very often, that subgroups have different properties than the complete population of positive examples. Especially relevant is situation when two subgroups have contradictory supporting factors, i.e. one subgroup has low attribute *X* values as its supporting factor while the other has supporting factor which are high values of the same attribute. In this situation it can be expected that statistical properties of the complete positive population are not very different from the negative population and the relevance of attribute *X* may remain undetected although it is actually very relevant and characteristic for both (all) subpopulations of the positive class.

The second characteristic of statistical evaluation of subgroups is that a property of a subgroup is acceptable as its supporting factor only if it is at the same time different from the complete negative and the complete positive population. The latter condition is necessary because properties that are different only from the negative population present supporting factors for the complete positive population and are not characteristic specifically for the subgroup. Theoretically and practically it is not necessary to use the same significance levels for both tests. Good default values are $P<.01$ for the significance of the difference between

positive examples in the subgroup with respect to the complete negative class and $P<.05$ for the significance with respect to the complete positive set.

The supporting factors detection process is repeated for every attribute separately. For numerical attributes we compute their mean values while for categorical attributes we compute the relative frequency of the most frequent or most relevant category. The statistical significance between example sets can be for numerical attributes determined using Mann-Whitney test and for categorical attributes using the chi-square test of association. A practical tutorial on using these tests, as well as other potentially applicable tests, can be found in [26] (Ch. 11a and 8, respectively).

5.6. Support vector machines

Support vector machines (SVMs) and other kernel based algorithms can be developed and implemented for the solution of classification and regression analysis problem. The goal of classification is to build a set of models that can correctly predict the class of the different samples.

Let $\mathcal{X}$ the input space, and Y the output space, the goal of machine learning in classification problem is to determine the optimal classifier, that is to predict the unknown label of new pattern, using either prior knowledge of problem and the training data. In particular, in binary classification problem the output space is $Y=\{-1,+1\}$. A desirable situation is one in which the SVM algorithm is able to detect new patterns with high accuracy.

In this section we report a description about support vector machine, semidefinite programming and also the basis of kernel methods; for major details we refer to texts reported in bibliography section.

The support vector machine, developed by Cortes and Vapnik [27] as a method for binary classification problem, is currently a hot topic in the machine learning community [27, 28, 29]. Several applications of the support vector machine improve the results obtained with other methods such as neural networks and this characteristic makes the SVM very important.

Mathematically, let the dataset consist of l vector $\mathbf{x}_i \in \Re^n$, with a prior known corresponding class label (output value) $y_i \in \{-1,+1\}$. Each instance $\mathbf{x}_i \in \Re^n$ belongs only to one class A^+ or A^-. The class separation is based in a looking for the optimal separating hyperplane (OSH) between the two classes of data point, by maximizing the margin between the classes' closest points: the OSH is in the middle of the margin. When the data points are not linearly separable they are projected into a higher-dimensional space $\mathcal{F}$ where a linearly separation is possible: this projection is realized via the kernel trick.

Let thus $S=\{(\mathbf{x}_i, y_i), i=1...m\}$ the training data separable by a hyperplane.

The important question is the following: what is the best linear classifier (Figure 5-19) of the type

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b \quad = w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_mx_m + b. \tag{5.22}$$

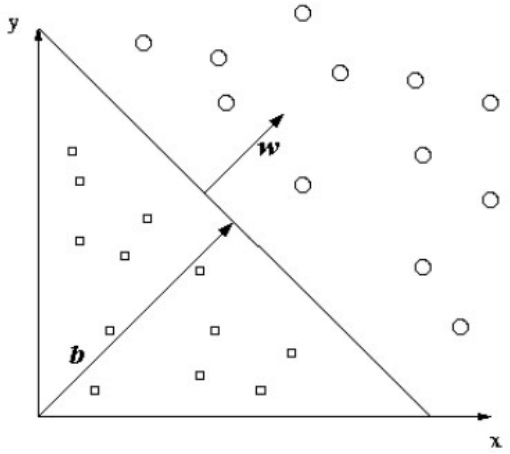

Figure 5-19. An example of a linear classifier (5.9)

Many linear classifiers (hyperplanes) separate the data and there can be an infinite number of hyperplanes that achieve 100% accuracy on training data *S* (Figure 5-20).

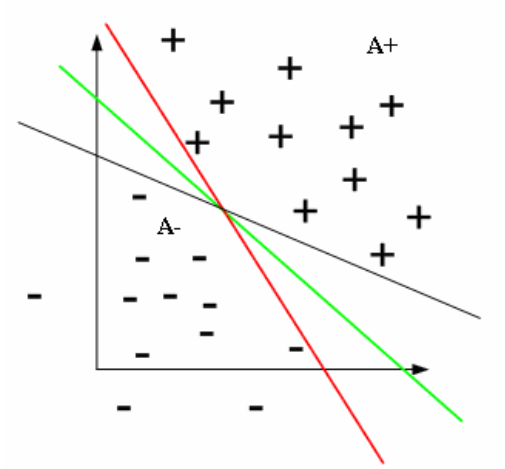

Figure 5-20. The elements of the two classes A+ and A$^-$ can be separated by an infinite number of linear separations

The most important question is to find the optimal hyperplane with respect to the accuracy on test data, given by instances unseen in training phase; it is used to estimate the generalization performance of the final classifier. Given a dataset, only one hyperplane achieves the maximum separation.

Define the margin as the smallest distance between data points and the class boundary, the solution of SVM is the best linear classifier $f(\mathbf{x}) = (\mathbf{w}^T\mathbf{x}+b)$. The optimal separating hyperplane is the hyperplane with larger margin such that for points closest to the separating hyperplane one have $\left|\mathbf{w}^T\mathbf{x}_i+b\right|=1$ (this points are called the support vectors), and for other points one have $\left|\mathbf{w}^T\mathbf{x}_i+b\right|>1$. In this way, the SVM avoids the overfitting.

The corresponding decision function, used to predict the class of the unseen pattern $\tilde{\mathbf{x}}$ for which class information is not known, is given using the OSH

$$\tilde{y} = sign(f(\tilde{\mathbf{x}})) = \begin{cases} +1 & \text{if } f(\tilde{\mathbf{x}}) \geq 0 \\ -1 & \text{if } f(\tilde{\mathbf{x}}) < 0. \end{cases} \tag{5.23}$$

The SVM implements the hinge loss function.

In this way the support vector machines attempt to minimize an upper bound of the generalization error rather than minimize the training error, which leads to better generalization performance than other classification methods.

It is possible to obtain different solutions (**w,** b) in the identical classification problem. One can apply any scalar β such that:

$$\tilde{y} = sign\left(\beta(\mathbf{w}^T\mathbf{x}+b)\right) = sign\left(\mathbf{w}^T\mathbf{x}+b\right). \tag{5.24}$$

5.6.1. Learning problem

Assuming a linearly separable dataset, the task of learning variables **w** and *b* of support vector machine $f(\mathbf{x}) = (\mathbf{w}^T\mathbf{x}_i+b)$ reduces to solving the following constrained optimization problem:

$$\begin{aligned} &\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 \\ &s.to \\ &y_i(\mathbf{w}^T\mathbf{x}_i+b) \geq 1, \quad \forall i. \end{aligned} \tag{5.25}$$

This optimization problem can be solved by using the Lagrangian function defined as:

$$L(\mathbf{w},b,\boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} - \sum_{i=1}^{N}\alpha_i[y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)-1], \alpha_i \geq 0, \forall i \tag{5.26}$$

where $\alpha_1, \alpha_2, ..., \alpha_m$ are Lagrange multipliers ($\boldsymbol{\alpha} = [\alpha_1, \alpha_2, ..., \alpha_m]^T$). The solution of the original constrained optimization problem is determined by the saddle point of $L(\mathbf{w},b,\boldsymbol{\alpha})$, which has to be minimized with respect to **w** and *b* and maximized with respect to $\boldsymbol{\alpha}$. If $y_i(\mathbf{w}^T\mathbf{x}_i+b)>1$, the value of α_i that

maximizes $L(\mathbf{w},b,\boldsymbol{\alpha})$ is $\alpha_i = 0$. If $y_i(\mathbf{w}^T\mathbf{x}_i + b) < 1$, the value of α_i that maximizes $L(\mathbf{w},b,\boldsymbol{\alpha})$ is $\alpha_i = +\infty$. However, since $\mathbf{w}$ and b are trying to minimize $L(\mathbf{w},b,\boldsymbol{\alpha})$, they will be changed in such a way to make $y_i(\mathbf{w}^T\mathbf{x}_i + b)$ at least equal to +1. Thus, the so-called Kuhn-Tucker conditions follow: $\alpha_i\{y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1\} = 0, \forall i$. Data points $\mathbf{x}_i$ with $\alpha_i > 0$ are called the *support vectors*.

The necessary optimality conditions for the saddle point of $L(\mathbf{w},b,\boldsymbol{\alpha})$ are:

$$\begin{aligned} \frac{\partial L}{\partial w_j} &= 0, \quad \forall j = 1,\ldots,n \\ \frac{\partial L}{\partial \alpha_i} &= 0, \quad \forall i = 1,\ldots m. \end{aligned} \tag{5.27}$$

Solving for the necessary conditions results $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$, $\sum_{i=1}^{m} a_i y_i = 0.$

By replacing $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$ into the Lagrangian function and by using $\sum_{i=1}^{m} a_i y_i = 0$ as a new constraint, the dual optimization problem can be constructed as the following convex quadratic programming problem:

$$\begin{aligned} &\max \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i{}^T \mathbf{x}_j \\ &s.to \\ &\sum_{i=1}^{m} \alpha_i y_i = 0 \\ &\alpha_i \geq 0, \quad \forall i. \end{aligned} \tag{5.28}$$

The classification function can be expressed as $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}_i + b = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i{}^T \mathbf{x} + b$, given the values $\alpha_1, \alpha_2, \ldots, \alpha_m$ obtained by solving the dual problem. The term $b = \frac{1}{|SVs|} \sum_{k \in SVs} \left(y_k - \sum_j \alpha_j y_j \mathbf{x}_j{}^T \mathbf{x}_k \right)$, where SVs is the set of support vectors. In this way, only support vectors are used in giving a decision function, since the corresponding Lagrangian multipliers $\alpha_i \neq 0$ only for them. An example of SVM's result is depicted in Figure 5-21.

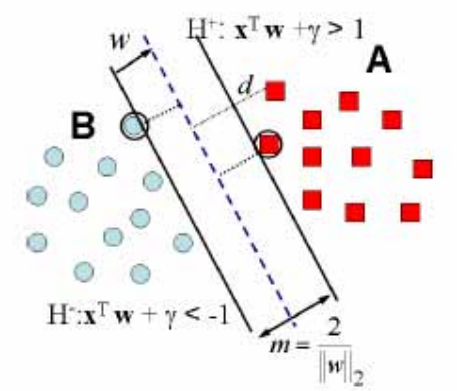

Figure 5-21. An example of an optimal separating hyperplane on an artificial problem

5.6.2. Linearly non-separable case

In the above section one have a very strong assumption, also unrealistic in most real life applications: the dataset is linearly separable. To tackle the problem given by a non linearly separable dataset (Figure 5-22), the slack variables $\xi_i, i = 1,...,m$, are introduced to relax the constraint $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$ to $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \xi_i \geq 0$. Ideally, one would prefer all slack variables to be zero since this means a linearly separable case. Therefore, the optimization problem for construction of SVM on linearly non-separable data is defined as:

$$\begin{aligned} & \min \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i \\ & s.to \\ & \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \forall i \\ & \quad \xi_i \geq 0, \quad \forall i \end{aligned} \tag{5.29}$$

where $C > 0$ is an appropriately selected parameter. The trade-off parameter C puts penalty on patterns that are misclassified or they are close to the SVM decision boundary; it controls the complexity of the decision function versus the training error minimization. The additional term $C\sum_i \xi_i$, into the objective function, enforces all slack variables to be as close to zero as possible.

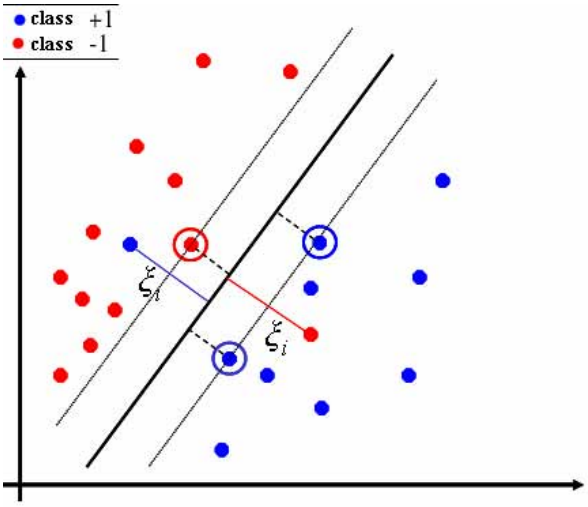

Figure 5-22. An artificial linearly non-separable case

As in the linearly separable problem, the above optimization problem can be converted to its dual problem formulation:

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$s.to \qquad (5.30)$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

$$0 \le \alpha_i \le C, \quad \forall i.$$

The direct consequence of introducing the parameter C is in constraining the range of acceptable values of Lagrange multipliers α_i. The most appropriate choice for C will depend on the specific dataset available.

5.6.3. Nonlinear case

Most real-world problems involve non separable data for which there does not exist a hyperplane that successfully separates the positive from the negative examples (that is the elements of class A^+ from the elements of class A^-).

One solution to the linearly inseparability problem is to map the data into a higher-dimensional space and define a separating hyperplane there. This higher dimensional space $\mathcal{F}$ is called Feature Space as opposed to the Input Space $\mathcal{X}$, where live the training examples. By the Cover's theorem, given a dataset S, non linearly separable in the original attribute space, it is possible to transform them into a new attribute space, where S is linearly separable: this space is called feature space and it can be a high dimensional space.

Denote Φ: $\Re^n \rightarrow \mathcal{F}$ as a mapping from the original *n*-dimensional attribute space to the highly dimensional attribute space $\mathcal{F}$. By solving the following dual problem

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \\ &s.to \\ &\sum_{i=1}^{m} \alpha_i y_i = 0, \\ &0 \le \alpha_i \le C, \quad \forall i \end{aligned} \tag{5.31}$$

the resulting SVM is given in the following form:.

$$f(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}_i) + b = \sum_{i=1}^{m} \alpha_i y_i \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}) + b \tag{5.32}$$

5.6.4. Solution: Kernel trick

Kernel methods [30, 31, 32] work by embedding data items into a vector space $\mathcal{F}$ in which to look for linear relation. This embedding is defined implicitly, by specifying an inner product for the feature space via a positive semidefinite kernel function *k*.

SVMs and other kernel methods derive their power from their ability to incorporate prior knowledge via the kernel function. Different kernel functions correspond to different embedding of the data. The kernel matrix contains the value of the kernel for every couple of data points, and every function that satisfies the Mercer's Theorem is a valid kernel [33]. There is a class of mappings Φ that has the following property: $\Phi(\mathbf{x})^T \Phi(\mathbf{y}) = K(\mathbf{x}, \mathbf{y})$, where K is a corresponding kernel function. The kernel function *K*, on all pairs of data points, yields a symmetric, positive semidefinite matrix **K,** known as *kernel matrix*, which can be regarded as a matrix of generalized similarity measures among the data points. Defined the kernel matrix as follows:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) \tag{5.33}$$

that is $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j), \forall i, j = 1, ..., m$. **K** is an *m*-by-m matrix; it is positive semidefinite if it is a Gram matrix.

$$K_{training} = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \dots & k(x_1, x_m) \\ k(x_2, x_1) & k(x_2, x_2) & \dots & k(x_2, x_m) \\ \dots & \dots & \dots & \dots \\ k(x_m, x_1) & k(x_m, x_2) & \dots & k(x_m, x_m) \end{bmatrix} \tag{5.34}$$

By introducing the kernel trick, the dual problem is given as:

$$\begin{aligned} &\max_{\alpha} \sum_i \alpha_i - 1/2 \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ &s.to \\ &\sum_{i=1}^{m} \alpha_i y_i = 0, \\ &0 \le \alpha_i \le C, \quad \forall i. \end{aligned} \tag{5.35}$$

The resulting SVM is: $f(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}_i) + b = \sum_{i=1}^{m} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$, that is a linear function in the feature space, implicitly defined by *K*.

Positive semidefinite symmetric kernel matrices are also referred to as Mercer Kernels, and they are the heart of several kernel-based algorithms, such as the SVM.

There are varieties of possible kernels. Typical used kernel functions are:

Gaussian Kernel: $K(\mathbf{x}, \mathbf{z}) = e^{\frac{\|\mathbf{x}-\mathbf{z}\|^2}{\sigma}}$

Polynomial Kernel: $K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z} + 1)^d$, where *d* is a constant.

Some practical issues with SVM are for instance relative to the modelling choices. A user should choose: 1) kernel function and its parameter(s); 2) constant *C* related to the slack variables. Several choices should be examined, using validation set, in order to find the best SVM.

Recently, kernel parameters in support vector machines are researched: Ong et al. [34] applied semidefinite programming to learn the kernel function (hyperkernel). Lanckriet et al. [35] proposed the optimization criterion for kernel-based learning algorithms and solved it by semidefinite programming. This strategy can effectively learn both the model class and the function without local minima.

5.6.5. Support vector machines as semidefinite program formulation

Choosing the kernel function in support vector machine is crucial for the algorithm; the very problem is how to choose a suitable data transformation for the given task. Recently, there are developed new approaches with the aim to learn the optimal kernel function in SVM using the semidefinite programming (SDP), special case of optimization over symmetric cones [36, 37].

The SVM model can be cast into the framework of semidefinite programming.

Several kernel functions are combined in linear mode to obtain the optimal kernel, corresponding to a particular Feature Space in which the margin is maximized.

Elementary properties of the set of kernel functions, such as its closure under pointwise, are directly inherited from well known results in Kronecker and Schur (or Hadamard) algebras of matrices. Kernels are also integral part of functional analysis, since with each kernel k on $\mathcal{X}$ is associated a Hilbertian space $\mathcal{H}_k$ of real valued functions on $\mathcal{X}$.

To integrate multiple data sources, the semidefinite programming based SVM method was introduced by Lanckriet et al. [35] where SDP is used to learn an optimal kernel matrix for transductive problem. There are different advantages to want to combine kernels: the data are of many different types or it is not sure which one should use or, most generally, it is not sure how to select one kernel from a handful of possible kernels. It is known that the sum and the product of kernels is a kernel, and also multiplying a kernel by a non-negative scalar is a kernel. The positive semidefiniteness of kernel functions translates in practice into positive semidefinite matrices.

Several researchers have investigated the problem of selecting an optimal kernel. Lanckriet et al. for to compute the optimal kernel matrix in a transductive setting for pattern recognition problems have used the SDP [36]. Lanckriet et al., use a weighted sum of kernel matrices, where the non-negative weights are automatically determined such that irrelevant datasets can be discarded.

The SDP programs are a class of convex optimization: the linear objective function is minimized over the intersection of cone of positive semidefinite (PSD, for short) matrices and affine sets. The important property of SDP, which make it more attractive, is the convexity: the set of positive semidefinite matrices is convex since any positive combination of semidefinite matrices is semidefinite: all the eigenvalues are nonnegative. An important result is that a local optimal solution will be also the global optimal solution since the objective function and the previous constraints are all convex; in this way, those nonlinear programs that can be modelled as SDP are convex programs. The set of PSD matrices is a closed convex cone and the clodness of the cone means that the boundary in included (the boundary is the set of the singular PSD matrices). The kernel formalism allows to combine various matrices: given two kernel functions k_1 and k_2, inducing the embeddings $\phi_1(\mathbf{x})$ and $\phi_2(\mathbf{x})$ respectively, it is possible to define the kernel $K = K_1 + K_2$, inducing the embedding $\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}))$.

As demonstrate in [37] the SVM soft-margin is formulated as SDP problem in standard form.

$$\min_{K \in \mathrm{K}} \max_{\alpha} 2\alpha^T e - \alpha^T diag(Y) K diag(Y) \alpha$$

$$\alpha^T y = 0, trace(K) = c, C \geq \alpha \geq 0. \tag{5.36}$$

where $Diag(Y) = diag(y_1, ..., y_m)$. The dual formulation of SVM-soft margin can be cast as SDP as follows:

$$\begin{aligned} &\min_{\bar{K},t,\beta,\delta,\lambda} \quad t \\ s.t. \quad & \\ & \bar{\mathbf{K}} \in \mathcal{K} \\ & \begin{pmatrix} G(K) & (e+\beta-\delta+\lambda y) \\ (e+\beta-\delta+\lambda y)^T & t-2C\delta^T\mathbf{e} \end{pmatrix} \succeq \mathbf{0} \\ & \beta \geq 0 \\ & \delta \geq 0 \end{aligned} \tag{5.37}$$

where $G(K)$ is defined by $G(K) = y_i K_{ij} y_j = k(\mathbf{x}_i, \mathbf{x}_j) y_i y_j$, and $\mathcal{K}$ is the convex cone of semidefinite positive kernel matrices.

The margin of separating hyperplane is maximized using the kernel matrix **K** as problem's variable. The algorithm used to solve the mathematical formulation is for instance the interior point method.

Combining several known kernel functions, one obtain the *support kernel functions* (likewise the support vectors), that is the kernel functions with non-zero weight coefficients in the resulting optimal linear combination. In this way, we have a method to assign a weight-importance to each kernel function into the given set of kernel functions and thereby to select important ones. Given a dataset, kernel function combination means linear combination of the corresponding kernel matrices; the result is a Gram matrix associated to a particular kernel function *k* that we define *hybrid kernel function*.

In contrast to paper of Lanckriet et al., we are working on mathematical model that classifies the data points to induction (while the problem tackled in Lanckriet in the transduction setting). We have modified some constraints, presented in their model: the trace of the optimal kernel matrix is not constrained to be equal to a constant. To restrict the research domain and to avoid overfitting we have considered different classes of kernels given as convex, affine and conic combination.

This approach can be viewed as the search of kernel $\mathbf{K}^*$ that have good generalization properties. The model is then used to search the best kernel function (hybrid or not) rather than the optimal kernel matrix. The solution of the model is function of the set of selected kernel functions, thus function of data points and so of the kernel matrices associated. Let SVs the set of support vectors, obtained as solution of the mathematical formulation, the resulting optimal hyperplane (classifier) is tested on test set.

5.7. Radial basis function networks

Radial basis function networks emerged as variant of artificial neural networks but their roots are in much older pattern recognition techniques as for example potential functions, clustering, functional approximation, spline interpolation and mixture model, and they are used also for time series prediction and control [38].

Since Broomhead and Lowe's seminal paper [39] radial basis function networks have traditionally been associated with radial functions in network having three layers (as depicted in Figure 5-23):

- an input layer
- a hidden layer with a non linear RBF activation function
- a linear output layer

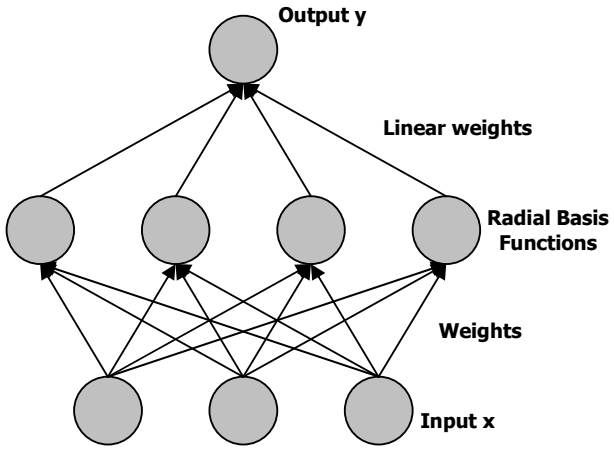

Figure 5-23. A depiction of radial basis function network with three layers

Radial basis function networks used for pattern classification are based on *Cover's theorem on the separability of patterns*. This theorem states that nonlinearly separable patterns can be separated linearly if the pattern is cast nonlinearly into a higher dimensional space. Therefore RBF networks are able to convert the input to a higher dimension after which it can be classified using only one layer of neurons with linear activation functions. Due to their nonlinear approximation properties, RBF networks are able to model complex mappings, which perceptron neural networks can only model by means of multiple intermediary layers [40]. To convert the input into a higher dimension, special activation functions are used in the hidden layer. All possible functions display *radial* symmetry and the hidden layer provides a new linear *basis* to solve the problem, hence the name "radial-basis". Examples of radial basis functions are given in Figure 5-24.

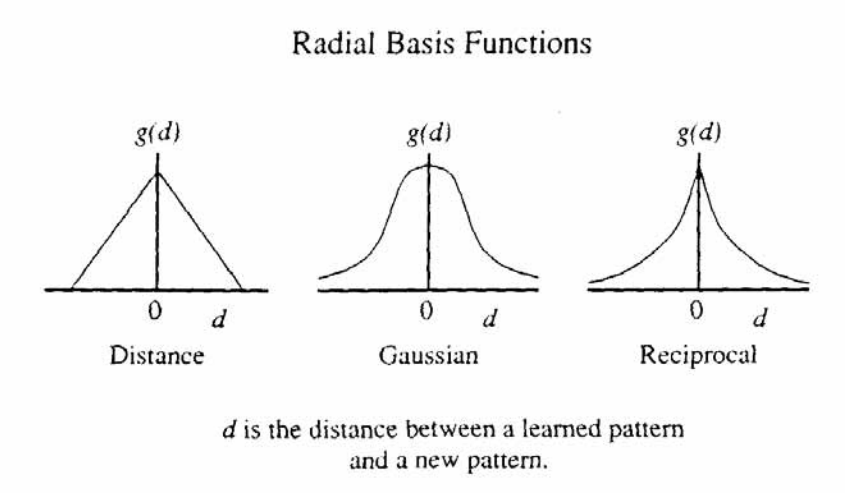

Figure 5-24. Different kinds of radial basis functions

The radial basis functions are characterized by their localization (centre) and by an activation surface. In a general case, the hypersurface is a hyperellipsoid and the activation function influence decreases according to the Mahalanobis distance from the centre. This means that data samples located at large Mahalanobis distance from the RBF centre will fail to activate that basis function. The maximum activation is achieved when the data sample coincides with the mean vector. The hypersurface is a hypersphere in the case when the covariance matrix is diagonal and has the diagonal elements equal.

Various functions have been tested as activation functions for RBF networks, but the Gaussian function is preferred, especially for pattern recognition problems.

The Gaussian activation function for RBF network is given by:

$$\phi_j(X) = \exp\left[-(X-\mu_j)^T \Sigma_j^{-1}(X-\mu_j)\right], \quad \text{for } j=1,\ldots,L \tag{5.38}$$

where X is the input feature vector, L is the number of hidden units and μ_j and Σ_j are the mean and the covariance matrix of the *jth* Gaussian function.

In certain approaches a polynomial term is added to the expression afore shown, while in others the Gaussian function is *normalized* to the sum of all Gaussian components. Geometrically, a radial basis function represents a bump in the multidimensional space, whose dimension is given by the number of entries. The mean vector μ_j represents the *location*, while Σ_j models the *shape* of the activation function. Statistically, an activation function models a probability density function where μ_j and Σ_j represent the first and the second order statistics.

The input into a radial basis function network is nonlinear while the output is linear. Each unit in the hidden layer implements a radial activation function, while the output layer implements a weighted sum of hidden-unit outputs:

$$\psi_k(X) = \sum_{j=1}^{L} \lambda_{jk}\phi_j(X), \quad \text{for k}=1,\ldots,M \tag{5.39}$$

where λ_{jk} are the output weights, each one corresponding to the connection between a hidden unit and an output unit and M is the number of output units. The weights λ_{jk} show the contribution of a hidden unit to the respective output unit. In a classification problem if $\lambda_{jk}>0$ the activation field of the hidden unit j is contained in the activation field of the output unit k.

In pattern classification applications, the output of the radial basis function is limited to the interval (0,1) by a sigmoidal function:

$$Y_k(X)=\frac{1}{1+\exp[-\psi_k(X)]}, \quad \text{for k=}1,\ldots,M \tag{5.40}$$

In order to use a radial basis function network it is necessary to specify the hidden unit activation function, the number of processing units, and a training algorithm for finding the weights of the network (network training). The RBF network extracted from data (training set) can be validated and evaluated through well known machine learning validation methodologies, as for example cross validation. The trained RBF network then can be used to classify new unseen data.

RBF networks are used mainly for supervised learning tasks. The network parameters are found minimizing the cost function:

$$\min \sum_{i=1}^{Q}(Y_k(X_i)-F_k(X_i))^T(Y_k(X_i)-F_k(X_i)) \tag{5.41}$$

where Q is the total number of vectors from training set, $Y_k(X_i)$ denotes the RBF output vector and $F_k(X_i)$ represents the output vector associated with the data sample X_i from the training set.

Many training algorithms have been tested for training RBF networks. The basic solution proved to be expensive in terms of memory requirement and in the number of parameters. Additionally, overfitting on the training set may cause bad generalization.

Other approaches implement two different levels of learning:

- Centre and spread learning (determination)
- Output layer weights learning

In the basic approach the location of the centres can be chosen randomly from the training set. Different values of centres and widths for each radial basis function can be used. The output weights λ_{jk} are obtained by solving a system of equations whose solutions is given in the training set. Matrix inversion is required in this approach and this operation is computationally expensive and can cause numerical problems when the matrix is singular. Pseudo-inverse method can be used as well. Moreover, this learning strategy requires a large training set for a satisfactory level of performance.

The second approach assumes that the radial basis function centres are uniformly distributed in the data space. The function to be modelled is obtained by

interpolation. Also less basis functions than given data samples can be used adopting a least squares solution that minimizes the interpolation error.

Another learning approach is based on a self-organized selection of centres. Clustering algorithms as *k*-means or learning vector quantization are employed for estimating the centres of the radial basis function in the hidden layer. A supervised learning algorithm is then used to estimate the linear weights of the output layer (e.g., least minimum squares).

Algorithms for supervised selection of centres exist and they look at the entire network training it as a normal multilayer neural network. The procedure used for training is the *gradient descent procedure*.

5.8. Bayesian learning

In recent years, the theory of Bayesian networks has led to several new approaches in machine learning algorithms for density estimation, classification, causal discovery and variable selection.

A Bayesian network (or a Belief network) is a mathematical object $\langle V, G, P \rangle$, where V is a set of random variables (interchangeably also called nodes or features in the rest of this section), G a graph defined among the variables in V, and P a joint probability distribution defined over the variables in V. For a triplet $\langle V, G, P \rangle$ to be a Bayesian network the Markov condition needs to hold among each variable $X \in V$: X is probabilistically independent of any subset of its non-descendant nodes given X's parents in the graph G. A descendant node of X in a graph is defined to be any other node Y for which a directed path from X to Y exists. Given a Bayesian network $\langle V, G, P \rangle$ the joint probability distribution $P(V_1, \ldots, V_n)$ can be factored as follows:

$$P(V_1, \cdots, V_n) = \prod_i P(V_i \mid Pa(V_i)) \tag{5.42}$$

where $Pa(V)$ are the parent nodes of V in the graph G (i.e., the nodes with a direct edge towards V). In other words, each graph G that is related by the Markov condition to a probability distribution P allows the factoring of the distribution according to the equation above. This factoring allows, for sparse graphs, a dramatic reduction in the parameters required to capture the distribution P. If there are 20 binary variables V_i the joint distribution P has $2^{20} - 1 \approx 10^6$ parameters. However, if there is a sparse network with at most 2 parents for each node capturing distribution P, we need to parameterize just 20 distributions of the form $P(V_i|Pa(V_i))$. Each of these distributions has one parameter for each possible value of V_i for each possible combination of the values of the $Pa(V_i)$, i.e., at most $2^3 = 8$ parameters. Thus, in total we need to specify at most $20 \times 8 = 160$ parameters to completely specify the distribution (the actual number is less if we take into account that some groups of parameters sum to 1).

A Bayesian network thus encodes some independencies of the joint probability distribution and allows a compact representation of that distribution. A large class

of Bayesian networks is the class of *faithful* Bayesian networks defined as the networks that are such that all and only the independencies in the distribution are the ones entailed by the Markov condition. In other words, the dependencies and independencies in the distribution are due to the structure of the network's graph. This is a large class of networks and non-faithful network "rare" to find (under some network distributional assumptions) [41].

A particular case of Bayesian networks is represented by the naïve Bayes classifier. This kind of classifier is based on the strong assumption that each input variable is conditionally independent from all the other inputs variables; thus, in the case of naïve Bayes classifier the graph defined among the variables reports uniquely direct connections from the output variable to all the input variables.

5.8.1. Markov – blanket based feature selection

The theory of Bayesian networks is related to variable selection by the concept of the Markov blanket. A Markov blanket of a node T is denoted as $MB(T)$. A Markov blanket of a variable T is a minimal set of nodes such that every node other than T and $MB(T)$ is independent of T conditioned on the nodes of the $MB(T)$. More formally, for any subset of nodes $X \subseteq V \setminus \{T\} \cup MB(T)$ it holds: $Ind(X, T \mid MB(T))$, where $Ind(X, T \mid MB(T))$ denotes that X is independent of T given $MB(T)$, and V is the set of all variables in our data. Intuitively, the $MB(T)$ is a set of minimal size that conveys all the information in out data for specifying the distribution of T. *Given the values of the variables in the MB(T), any other variable carries superfluous information for the prediction of T.* Bayesian networks and the Markov blanket are connected by the following important result: the $MB(T)$ in a faithful network is unique and it has a graphical interpretation; *it is the set of parents, children and spouses of T in the graph of the network capturing the data distribution.* The parents of T are the nodes with a direct edge to T, the children the nodes with a direct edge from T, and the spouses are the parents of the children of T.

A Markov blanket of T is the optimal (i.e., minimum) feature subset for the prediction of T under some general assumptions: (i) that the model used to predict T from the variables in $MB(T)$ can actually perfectly learn the functional relation from $MB(T)$ to T. In other words, the $MB(T)$ is useful only if there are powerful enough learner that can utilize the information carried by the $MB(T)$ variables. (ii) The performance metric of the final model requires complete knowledge of the conditional distribution $P(T \mid V \setminus \{T\})$. Otherwise, the $MB(T)$ maybe not be the minimum subset required for perfect prediction.

Computationally efficient and provably correct algorithms (in the sample limit) for identifying the $MB(T)$ in a dataset were developed. The algorithms perform a series of statistical tests of conditional independence in the data; by using Bayesian network theory, the results of the test allow reconstruction of the $MB(T)$ of the underlying distribution. Thus, the algorithms can guarantee (under the assumption of faithfulness of the underlying distribution and in the sample limit) that they will correctly identify the $MB(T)$. In turn, according to the discussion

above, under most common analysis situations, this is the minimum subset of variables with the maximum prediction power, i.e., the solution to the feature selection problem. In contrast, most other feature selection algorithms do not have well-defined theoretical properties nor do they provide any theoretical guarantees. In empirical results, the Markov-blanket based algorithms have also shown to outperform several typical and state-of-the-art univariate and multivariate feature selection methods in a number of biomedical datasets [42].

5.9. Bibliography and references

[1] Quinlan, J.R. (1986), *Induction of decision trees*, 1: pp.81–106.

[2] Quinlan, J.R. (1993) *C4.5: programs for machine learning*. Morgan Kaufmann

[3] Kohavi, R., Quinlan, J.R. (2002) *Decision tree discovery*. Handbook of Data Mining and Knowledge Discovery, pp.267-276, http://ai.stanford.edu/~ronnyk/treesHB.pdf

[4] Wikipedia: The Free Encyclopedia (2008) *Decision tree learning - gini impurity*, http://en.wikipedia.org/wiki/Decision_tree_learning#Gini_impurity

[5] Breiman, L. (1996a), *Bagging Predictors*, Machine Learning, 24(2): pp.123–140. http://citeseer.ist.psu.edu/breiman96bagging.html

[6] Breiman, L. (1998) *Arcing classifiers*. The Annals of Statistics, 26(3), pp.801–849. http://citeseer.ist.psu.edu/breiman98arcing.html

[7] Breiman, L. (2001) *Random Forests*, Machine Learning, 45: pp.5–32., http://citeseer.ist.psu.edu/breiman01random.html

[8] Kam Ho, T. (1998) *The Random Subspace Method for Constructing Decision Forests*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 20: pp.832–844., http://citeseer.ist.psu.edu/ho98random.html

[9] Robnik-Šikonja, M. (2004) *Improving Random Forests*, In proceedings of ECML 2004, http://lkm.fri.uni-lj.si/rmarko/papers/robnik04-ecml.pdf

[10] Breiman, L., Cutler, A. (2004) *Random Forests*, University of California, Berkeley, 8. Jan. 2008, http://www.stat.berkeley.edu/users/breiman/RandomForests/cc_home.htm

[11] Breiman, L., Cutler, A. (2004) *ENAR Short Course*, http://www.stat.berkeley.edu/users/breiman/RandomForests/ENAR_files/frame.htm

[12] Clark, T.G., Bradburn, M.J., Love, S.B., Altman, D.G. (2003) *Survival Analysis Part I: Basic concepts and first analyses,* British Journal of Cancer, 89: pp.232–238.

[13] Bradburn, M.J., Clark, T.G., Love, S.B., Altman, D.G. (2009) *Survival Analysis Part II: Multivariate data analysis- an introduction to concepts and methods,* British Journal of Cancer (2003), 89: pp.431–436.

[14] Bradburn, M.J., Clark, T.G., Love, SB, Altman, D.G. (2003) *Survival Analysis Part III: Multivariate data analysis-choosing a model and assessing its adequacy and fit,* British Journal of Cancer (2003) 89: pp.605–611.

[15] Ishwaran, H., Kogalur, U.B. (2007) *Random survival forests for R.* Rnews, 7(2): pp.25–31. http://www.bio.ri.ccf.org/Resume/Pages/Ishwaran/papers/randomSurvivalForests.pdf

[16] Hothorn. T., Hornik, K., Zeileis, A. (2006) *Unbiased Recursive Partitioning: A Conditional Inference Framework*, Journal of Computational and Graphical Statistics, 15(3): pp.651–674.

[17] Eibe, F., Witten, I. H. (1998) *Generating Accurate Rule Sets Without Global Optimization*, In proceedings of the Fifteenth International Conference on Machine Learning, pp.144-151, http://citeseer.ist.psu.edu/frank98generating.html

[18] Cohen, W. W. (1995) *Fast effective rule induction*, In proceedings of the Twelfth International Conference on Machine Learning, pp.115-123, http://citeseer.ist.psu.edu/cohen95fast.html

[19] Lavrač, N., Gamberger, D., Flach, P. (2002) *Subgroup discovery for actionable knowledge generation: Defiences of classification rule learning and lessons learned.* "Data Mining Lessons Learned" Workshop at ICML 2002 Conference, pp.48–56.

[20] Fürnkranz, J. (2005) *From local to global patterns: Evaluation issues in rule learning algorithms.* In Morik K., Boulicaut J.-F., and Siebes A., editors, Local Pattern Detection, pp.20–38., Springer.

[21] Fayyad, U.M., Irani, K.B. (1992) *On the handling of continuous-valued attributes in decision tree generation.* Machine Learning, 8: pp.87–102.

[22] Lavrač, N., Gamberger, D. (2005) *Relevancy in constraint-based subgroup discovery.* In Boulicaut, JF., De Raedt, L., Mannila, H. editors, Constraint-Based Mining and Inductive Databases. Springer, pp.243–266.

[23] Lavrač, N., Kavšek, B., Flach, P., Todorovski, L. (2004) *Subgroup discovery with CN2-SD.* Journal of Machine Learning Research. 5: pp.153–188.

[24] Gamberger, D., Lavrač, N. (2002) *Expert-guided subgroup discovery: Methodology and application.* Journal of Artificial Intelligence Research, 17: pp.501–527.

[25] Lavrač, N., Kralj, P., Gamberger, D., Krstačić, A. (2007) *Supporting factors to improve the explanatory potential of contrast set mining: Analyzing brain ischaemia data.* In Proc. of 11th Mediterranean Conference on Medical and Biological Engineering (MEDICON 2007), pp.157–161.

[26] Lowry, R. (2007) *Concepts and Applications of Inferential Statistics*. Web Site for Statistical Computation, Vassar College. URL: http://faculty.vassar.edu/lowry/webtext.html

[27] Cortes, C., Vapnik, V. (1995) *Support-Vector Networks.* Machine Learning, 20: pp.273–297.

[28] Cristianini, N., Shawe-Taylor, J. (2000) *An introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press.

[29] Burges, C. J. C. (1998). *A Tutorial on Support Vector Machines for Pattern Recognition*. Data Mining Knowledge Discovery, 2: pp.121–167.

[30] Scholkopf, B., Smola, A. J. (2002) *Learning with Kernels*. MIT Press, Cambridge, MA.

[31] Cristianini, N., Schoelkopf, B. (2002) *Support vector machines and kernel methods*. *AI Magazine* 23(3): pp.31–41.

[32] Shawe-Taylor, J. Cristianini, N. (2004) *Kernel Methods for Pattern Analysis*. Cambridge University Press.

[33] Mercer, J. (1909) *Functions of positive and negative type and their connection with the theory of integral equations*. Philos. Trans. Roy. Soc.London, pp.415–446.

[34] Ong, C.S., Smola, A.J., Williamson, R.C. (2005). *Learning the kernel with hyperkernels*. Journal of Machine Learning Research, 6: pp.1043-1071.

[35] Lanckriet, G.R.G., Cristianini, N., El Ghaoui, L., et. al. (2004) *Learning the kernel matrix with semidefinite programming*. Journal of Machine Learning Research, 5: pp.27–72.

[36] Boyd, S., Vandenberghe, L. (1996). *Semidefinite programming*. SIAM Review, 38(1): pp.49–95.

[37] Boyd, S., Vandenberghe, L. (2001). *Convex optimization*. Course notes for EE364, http://www.stanford.edu.class/ee364

[38] Bors, A. G. (2001) *Introduction of the Radial Basis Function (RBF) Networks*. Online Symposium for Electronics Engineers, 1(1), pp.1–7.

[39] Broomhead, D.S., Lowe, D. (1988) *Multivariate functional interpolation and adaptive networks*. Complex Systems, 2: pp.321–355.

[40] Haykin, S. (1994) *Neural Networks: A comprehensive Foundation*. Upper Saddle River, NJ:Prentice Hall

[41] Meek, C. (1995) *Strong completeness and faithfulness in Bayesian networks*. In Conference on Uncertainty in Artificial Intelligence, pp.411–418.

[42] Aliferis, C.F., Tsamardinos, I., Statnikov, A. (2003) *HITON, A Novel Markov Blanket Algorithm for Optimal Variable Selection*. In the American Medical Informatics Association meeting 2003 (AMIA 2003)

6. Modelling results for HF datasets

While the previous section presented the KD methodology, this section begins with the description of several heart failure related datasets and then continues with the detailed presentation of the results that have been achieved by using the KD methods.

The analysed datasets come from very different sources. The first data included is the data already collected by the platform. In order to ensure homogeneity of the analysed dataset and comparative evaluation of the results, patient records collected by only one medical partner (UNICZ) have been included and the dataset has been frozen during the complete process. This is a relatively small dataset with very various attribute types which have been used to determine types of the data analyses appropriate for inclusion into the KD Web service.

The second dataset is a dataset specially collected by UNICZ for the purposes of the Heartfaid project. It includes information about continuously monitored patients and their decompensation events. This dataset has been collected with the intention to develop decompensation models that will be built into the decision support subsystem. Although it is currently a small dataset, it can be expected that Heartfaid platform will in future enable automatic collection of significantly larger datasets of this type. The performed experiments demonstrate usefulness of the existing methodology for the development of required decision making models, but medically relevant result can be expected only after collecting large amounts of the data. In the future this is expected as the most relevant KD result of the Heartfaid platform.

The largest dataset available for the analysis is a retrospective dataset obtained by extraction from the Italian database of heart failure patients prepared specifically for the KD purposes on the Heartfaid project. It is an impressively large database that is very rare in medical applications. After the cleansing and example elimination procedures have been executed, it consists of more than 17,000 examples. The dataset has been used for many different experiments, especially those related to patient prognosis, patient survival analysis and HF severity modelling. The obtained results have not been included into the platform's decision support subsystem but they are potentially very relevant for understanding the HF population.

Finally, we have tested the methodology on the publicly available genomic dataset describing 196 heart failure patients. The dataset has been an excellent test bed for most complex modelling tools because the datasets is described with about 22,000 gene expression levels.

6.1. Datasets

This section describes four different datasets. Besides their origins and meaning, information on data-preparation steps that have been done in order to enable application of KD tools is also included. The datasets are used to generate heart failure decompensation models, to analyse heart failure severity in respect to the time to the hospitalization, death, or some other relevant events, and to test the KD methodology on the data collected by the platform and on the data from other sources. The results are presented in the Section 6.2.

6.1.1. Patient records collected by the platform (eCRF dataset)

Patient data are saved in the patient database constructed according to the specifications of the heart failure Case Report Form described in deliverable D9. The database is also known as the *Electronic Case Report Form* (eCRF). The platform provides a user friendly interface that enables patient data collection.

Patients are described by data collected in a sequence of visits. Every patient has at least a *Baseline evaluation* (i.e. first visit), which may be followed with a list of *Additional visits*, and finally with the *Final evaluation* (visit at patient release).

The dataset used for the KD analysis consists of 74 heart failure patients described by their baseline evaluation data. All of the patients data have been collected and entered by the project partner UNICZ (University "Magna Graecia", Catanzaro, Italy, Faculty of Medicine).

The dataset includes following patient descriptors:

- cardiovascular and other medical history
- lifestyle information
- family history
- current cardiovascular therapy
- patient anamnesis
- patient examination:
  - physical examination
  - laboratory assessment
  - chest X-ray
  - 12-lead electrocardiography
  - echocardiography
  - 24h Holter electrocardiography
  - six minute walking test
- quality of life questionnaire

More details on the CRF and the related database can be found in the deliverable D9.

6.1.2. Decompensation onset dataset

For the clinical management of CHF patient the crucial mid-long term goal is early detection of new acute decompensation events. In this way, re-hospitalizations and/or eventually death of the patients may be avoided. All clinical partners agreed that monitoring some basic parameters may allow early detection of a new decompensation status. A well tuned and personalized therapy, high quality outcomes and reduction of the health care costs may be achieved if patient decompensation is early identified and appropriately tackled.

Within the relevant clinical protocols and guidelines, a general consensus has not been reached about the definition and assessment of criteria on how to predict when a patient will further decompensate, even though many different evidence-based indications are known. KD approaches may be a practical and effective solution in order to extract new potentially useful models about this clinical problem from repositories of pertinent data.

As indicated in section 8.2 of the deliverable D5, variations of some relevant parameters were selected for monitoring:

- Decrease of systolic blood pressure
- Increase of heart rate
- Increase of respiratory rate and width of chest movements
- Increase in the percentage of body water
- Variation of body temperature

A KD task was planned in order to give preliminary indications about the relevance of the selected parameters and about the KD methodologies which could be the most effective and efficient approach to this specific clinical problem.

UNICZ built an own repository “emulating”(within the clinical environment) the data acquisition from the home environment, obviously with some differences.

Since February 2007, 49 patients with established CHF diagnosis have been recruited in different periods during the survey campaign. Each patient went to cardiologist office every two weeks, where medical doctor measured and stored values of the selected parameters for that patient’s visit. Furthermore, the medical doctor stored indications of the patient’s health condition with respect to the decompensation status.

Monitored parameters during each visit were the following:

- Systolic blood pressure
- Heart rate

- Respiratory rate
- Body temperature
- Body weight
- Body water

Both domain experts and some results from preliminary analysis on the available data suggested that variations between two successive visits are potentially relevant. Two diverse types of differences were considered: simple and relative.

A simple difference, on a parameter, between two successive visits is

$$\Delta P(i) = P(i) - P(i-1), \quad \text{for } i=2,\ldots,N \tag{6.1}$$

while a relative difference is

$$\Delta_r P(i) = \frac{P(i) - P(i-1)}{P(i-1)}, \quad \text{for } i=2,\ldots,N \tag{6.2}$$

where P is the measured parameter and N is the number of visits for a given patient. An initial data transformation step was necessary in order to realize the classification task. First, for each patient, all pairs of measured parameters in two successive visits, where the patient health condition at the previous visit was "no decompensation", were selected. In this way, two "status transitions" are possible:

- the patient is in "no decompensation" condition at the previous visit and remains in the same condition at the current visit (marked as negative class cases)
- the patient is in "no decompensation" condition at the previous visit but he/she undergoes a transition to "decompensation" at the current visit (marked as positive class cases).

The dataset built in this way consists of 263 instances (8 positive and 255 negative instances, respectively) and the following 30 attributes:

- Sex
- Age
- NYHA class
- Smoke activity
- Alcohol use
- Systolic blood pressure (current and previous value, simple and relative variation)
- Heart rate (current and previous value, simple and relative variation)
- Respiratory rate (current and previous value, simple and relative variation)
- Body temperature (current and previous value, simple and relative variation)

- Weight (current and previous value, simple and relative variation)
- Total body water (current and previous value, simple and relative variation)
- Class (decompensation or no-decompensation)

Sex, *NYHA class*, *Smoke activity*, *Alcohol use* and *class* are the only nominal attributes into the dataset, while all the others are numeric.

The dataset was completed on January 2008 and it has highly skewed class distribution (only 8 examples for the class "decompensation"). Moreover, it has a small number of instances in respect to the complexity and relevance of the medical problem. However, the data acquisition campaign is still in progress at UNICZ in order to enlarge the dataset for further KD activities.

If useful knowledge is extracted from the available data, it will be used for suggestions regarding the patient decompensation status within the home-care environment on a daily basis. We can suppose that for each patient sub-enrolled for home-monitoring, the data from home will be acquired daily and that the constructed models will provide suggestions about the patient condition, based on the values of the parameters at the current day, those ones collected two weeks before and their simple and relative variations. This approach will allow to detect if the patient shows a new acute event of decompensation or, hopefully, to early asses risk for it.

6.1.3. Large retrospective database (ANMCO dataset)

The dataset has been prepared by the by ANMCO, a non-profit organization composed by Italian Cardiologists operating within the National Health Service. The preparation of the dataset has been requested by UNICZ (University "Magna Graecia" of Catanzaro who is IN – CHF project partner) for the research purposes within Heartfaid project.

ANMCO and IN–CHF Project

Founded in 1963, ANMCO objectives are the promotion of validated clinical procedures and the prevention of cardiac diseases; these aims are pursued through the promotion of professional education, the realization of studies and researches, and the development of new standards and guide lines.

One of ANMCO activities consists in the IN – CHF initiative. In 1994, the HF Working Group Board of ANMCO decided to entrust ANMCO Research Centre with the development of software to gather clinical and epidemiological data related to HF outpatients. This program has been given out for free to centres collecting the data.

The goal was to provide the Italian cardiologic wards (assigned to diagnosis and treatment of HF) with a software which could allow information gathering by a common language and a methodology shared by everyone. The top priority objective was to create an educational instrument.

The second, but no less important, aim was to direct all the gathered information to one national database, in order to use it for scientific purpose.

At the beginning, a pilot group of doctors fixed the items to be gathered and during 1994 the development and the test phase of the product went off. In particular, the program was planned to be used also by the nursing staff in real time during the HF outpatient examination.

From 1994, the IN – CHF participant numbers have significantly increased, and nowadays the number of registered patients is about 20,000 in over more than 100 centres.

In order to promote scientific research activities, IN – CHF the central database is open to all participant of the project, who are allowed to require partial view of the database, depending by the specific study that the participant centre intends to perform.

HEARTFAID – ANMCO dataset

This dataset (in the following referred as ANMCO dataset) has been specifically devised in order to develop predictive models aimed at the evaluation of patient prognosis after the first visit. ANMCO dataset is composed of a set of patients' records collected during the first visit or immediately before/after it. Furthermore for each patient the known follow-up data consisting of adverse events were reported. First visit information includes clinical parameters, electrophysiological and echocardiography measurements, blood exams parameters, anamnesis information, and therapy prescriptions. The total number of records overcomes 18,000 observations.

Although devised for a study about prognosis evaluation, ANMCO dataset is also suitable for other data mining task, including NYHA class evaluation and HF severity modelling. Moreover, the presence of the time interval from the first visit to the first adverse event enables the execution of different survival analysis tasks.

ANMCO dataset preprocessing: data cleansing and feature construction

In order to avoid processing of incorrect values, a univariate analysis of the range covered by each numeric attribute was performed. In fact, many attributes in the original dataset showed non-realistic values; in particular, many values were not compatible with human physiology, and probably were typing errors.

In order to point out this problem, upper and lower bounds for each numeric attribute were defined, according with the physicians recommendation. The irregular values have been considered as missing values.

Moreover, all the patients' records reporting an age below 18 years and a high ejection fraction (over 55%) were deleted. The deletion of such records was necessary in order to preserve the significance of the study. In particular, an elevated ejection fraction denotes an absence of the HF conditions, or a slight HF condition, or at least the presence of a right – sided HF. In any case, patients with

elevated ejection fractions are clinically significantly different from patients with compromised ejection fraction.

Regarding feature construction, three variables were constructed:

- Age
- Cubic Formula EF
- Squared Formula EF

Age has been computed from the dates of the first visit and the birth date. Regarding the other two new attributes, it should be known that EF is usually calculated by echocardiography device by using left ventricular diameters. The most used formulas for EF calculation are:

$$\text{Cubic formula} = \frac{LVTDD^3 - LVTSD^3}{LVTDD^3} \tag{6.3}$$

$$\text{Squared formula} = \frac{LVTDD^2 - LVTSD^2}{LVTDD^2} \tag{6.4}$$

Where LVTDD is Left Ventricle Telediastolic Diameter, and LVTSD is Left Ventricle Telesystolic Diameter.

It seems that some EF values reported in the original dataset were calculated with the first formula, and some other ones with the second one; moreover, some EF values have been calculated in some other manner. We decided to add the EF values calculated with both standard formulas to the dataset, leaving the experimental phase to choose the most informative parameter.

Missing values in ANMCO dataset

The problem of missing values in the ANMCO dataset was treated by introducing missing-value indicator variables for the prognostic models based on classification and through attribute categorization for the prognostic models based on survival analysis (see Section 4.2.1). The details of each are presented in the corresponding sections below.

6.1.4. Genetic HF patient data (public dataset)

In the recent work, Hannenhalli et al. [1] collected a dataset about RNA microarray data from 212 patients consisting of 196 Heart Failure patients and 16 healthy patients. Each record in this transcriptional genomics dataset is described by approximately 22,000 gene expression levels, measured directly from cardiac cellules. Data were collected from myocardium samples during cardiac transplantations. The complete dataset is publicly available from the web site of the National Center for Biotechnology Information (NCBI), at the following page:

http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE5406

From a scientific point of view, the dataset is interesting because it is extremely difficult to collect microarray data directly from cardiac cellules. Thus, examining

this dataset, the possibility of looking inside some of the basic mechanisms underlying the HF pathogenesis, at a cellular level is offered. For this reason, we decided to study this dataset within the context of the Heartfaid project. Data acquisition didn't present any difficulties, and a simple data transformation script has been implemented in order to transform the data from the SOFT format to the CSV (comma separated values) format that can be accepted by most of the tested KD tools.

6.2. Presentation of results for specific goals

The available HF datasets, in combination with different KD methodologies, enable definition and implementation of a broad range of knowledge discovery and data analysis tasks. They are rules and models for decompensation alarming, analysis of data collected by the platform, HF survival analysis, severity modelling and HF prognosis. The tasks are grouped by goals and for each of them the most relevant results, as well as the methodology that enabled their realization and lessons learned from the process are presented.

6.2.1. Decompensation alarming

The analysis of decompensation onset dataset, that has been collected specially for the purposes of developing the knowledge necessary for the Heartfaid decision support purposes, started by the application of decision tree and decision list methods which are able to extract knowledge in a form understandable by domain experts. The relations extracted from the data were initially assessed by validation techniques, and then the most relevant models were evaluated by the experts. Finally, methods like SVM and RBF networks were used for modelling in order to estimate predictive accuracy that may be expected from methods that do not generate human interpretable results. The learning methodologies were combined with cost sensitive classification approaches since classes in the dataset are highly unbalanced. The dataset is described in Section 6.1.2.

"Leave one patient out validation" (i.e. "leave-one-out" validation as described in Section 4.4.1) has been used for the evaluation. For each patient there are potentially many instances in the dataset, in particular one instance for each pair of patient's visits satisfying one of the two criteria defined in Section 6.1.2. The used validation technique works so that in turn all the instances relative to one specific patient are discarded from the dataset, and the remaining examples are used as the training set. After that the trained model is used for the classification of the previously discarded instances. In this way all the information relative to that patient does not influence the training process, and the instances used for model testing are really "unseen" and really new instances. This procedure is repeated for every patient in the dataset and the accuracy is computed as the mean value of all experiments.

The obtained results demonstrate relatively good performance in terms of accuracy, although in spite of the used cost sensitive classification approaches, the obtained models have low sensitivity.

The first obtained result is a decision list consisting of three rules predicting decompensation and a default rule classifying all remaining cases as no-decompensation. It must be noted that decision list is a set of rules that is interpreted in sequence so that the first satisfied rule performs the classification about the patient health condition. It means that, for the expert evaluation of the rules, the order of the rules is relevant. Evaluation of a rule requires that all rules earlier in the list should be taken into account as well. Taken individually out of the relevant context, rules of the set can be misinterpreted and/or their prediction quality can be low.

Rule 1:

IF
Heart Rate for current visit > 59 bpm
AND
Systolic Blood Pressure for previous visit > 118
AND
Total Body Water (simple variation) > 5.5L
AND
previous Weight > 83.5Kg
THEN
The patient is in decompensation

Rule 2:

IF
–3.9% < Total Body Water (relative variation) ≤ 0%
AND
Systolic Blood Pressure (simple variation) ≤ 0
AND
Heart Rate for current visit > 67 bpm
THEN
The patient is in decompensation

Rule 3:

IF
Systolic Blood Pressure (simple variation) ≤ –30
THEN
The patient is in decompensation

Rule else:

The patient is NOT in decompensation

For the complete training set, the list has prediction accuracy equal to 96.20%. When tested by the leave one patient out methodology the prediction quality is 90.11%. The main concern is that the sensitivity in the latter case is low (below 40%) which is not acceptable for the situations like decompensation which significantly influence the future of the patient.

Table 6-1. Prediction quality of the decision list for the training set with 263 instances (training set accuracy 96.20%).

Classified as		
Decompensation	NO decompensation	Real class
8	0	Decompensation
10	245	NO decompensation

Table 6-2. Prediction quality of the decision list measured by the "leave one patient out" methodology (cross-validation accuracy 90.11%).

Classified as		
Decompensation	NO decompensation	Real class
3	5	Decompensation
21	234	NO decompensation

The constructed decision list was evaluated by the UNICZ domain experts, who found it consistent with the existing medical experience about the domain.

The first rule covers 2 of the 8 "decompensation" labelled instances. It is based on the significant change in the total body water measured between two visits. The second one is used if the first one is not satisfied, and it covers 4 of the 8 "decompensation" labelled examples, assessing for the relative patient health condition a grater risk than the other rules. This rule is correct, in the experts' opinion, because usually, when an impaired heart function occurs, the systolic blood pressure decreases while the heart increases its rate in order to activate a compensation mechanism, and the total body water does not usually decrease too much. The third rule – used if the previous ones are not satisfied – performs the classification about the patient's health condition on the basis of the significant systolic blood pressure variation. The rule suggests that a relevant decrease in systolic blood pressure (more than 30 mmHg in two weeks) can be associated with a patient decompensation. This condition seems to be more unusual (only one instance in the dataset) than those of the previous rules.

Another relevant result is the following decision tree:

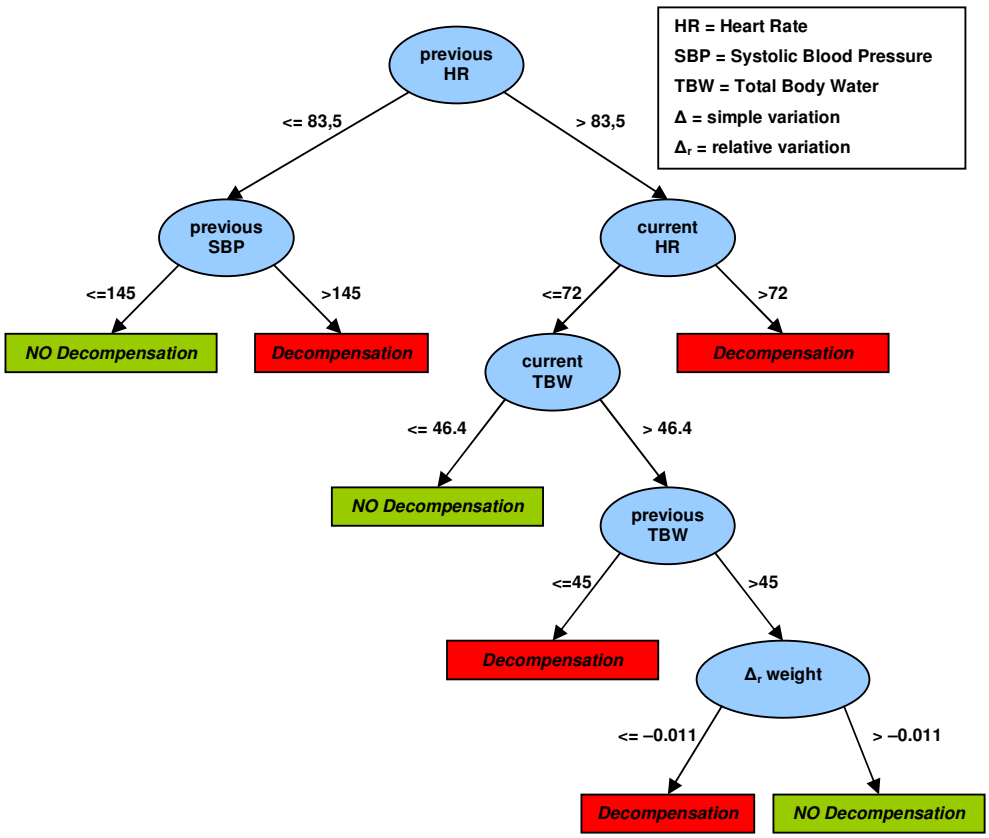

Figure 6-1. Decision tree constructed from the decompensation dataset.

Table 6-3. Prediction quality of the decision tree for the training set with 263 instances (training set accuracy 96.20%).

Classified as		
Decompensation	NO decompensation	Real class
8	0	Decompensation
10	245	NO decompensation

Table 6-4. Prediction quality of the decision tree measured by the "leave one patient out" method (cross-validation accuracy 87.83%).

Classified as		
Decompensation	NO decompensation	Real class
3	5	Decompensation
26	229	NO decompensation

The path corresponding to the following rule:

Weight for previous visit ≤ 83.5 Kg

AND

SystolicBloodPressure for previous visit ≤ 145

covers 161 of 255 no-decompensation instances, identifying a "pattern" with no risk of decompensation. Furthermore, many other no-decompensation instances (54) are covered by the rule defined by the path:

Weight for previous visit > 83.5 Kg

AND

HeartRate for current visit ≤ 72 bpm

AND

TotalBodyWater for current visit ≤ 46.4 L

On the other hand, the following rule covers 4 of the 8 decompensation cases labelled instances (but also 5 false negatives).

Weight for previous visit > 83.5 Kg

AND

HeartRate for current visit > 72 bpm

Finally, SVM and RBF networks were used to build high quality classifiers. The best results have been obtained by C-SVM with C equal to 1 and the polynomial Kernel with degree equal to 2.

Table 6-5. Prediction quality of the best SVM model for the training set with 263 instances (training set accuracy 100%).

Classified as		
Decompensation	NO decompensation	Real class
8	0	Decompensation
0	255	NO decompensation

Table 6-6. Prediction quality of the best SVM model measured by the "leave one patient out" method (cross-validation accuracy 85.17%).

Classified as		
Decompensation	NO decompensation	Real class
2	6	Decompensation
33	222	NO decompensation

Table 6-7. Prediction quality of the best RBF network for the training set with 263 instances (training set accuracy 98.10%).

Classified as		
Decompensation	NO decompensation	Real class
7	1	Decompensation
4	251	NO decompensation

Table 6-8. Prediction quality of the best RBF network measured by the "leave one patient out" method (cross-validation accuracy 95.06%).

Classified as		
Decompensation	NO decompensation	Real class
3	5	Decompensation
8	247	NO decompensation

The general conclusion of all experiments is that overfitting cannot be avoided, especially in respect to the minority decompensation class. The result is low sensitivity of all models when measured on the independent test set. Only larger dataset can enable construction of more reliable models.

6.2.2. Descriptive analysis of data collected by the platform

eCRF dataset consists of the first visit data from the patient records entered into the Heartfaid database. The dataset consists of 74 patients described with a vast number of attributes with a high percentage of missing values. The goal of the analysis has been extraction of previously unknown relations using knowledge discovery methods and testing which methodology might be useful for the Web-based KD service. As the dataset has no specified target attribute, at first unsupervised experiments were undertaken in order to obtain insight into the data structure and to detect potentially relevant patient subpopulations. After that, we have selected some attributes that may be interesting as target concepts and for each of them we have repeated the descriptive analysis process.

Results of unsupervised learning

Multiple experiments using unsupervised RF methodology described in Section 5.2.4 have been performed. In order to ensure that the problem doesn't lie in random data permutation range, the experiments have been done using at least 10,000 trees. Interesting data separation, presented in Figure 6-2, was attained. MDS projection of the data on three axes reveals two clusters of patients, one consisting of 63 and the other of 11 patients.

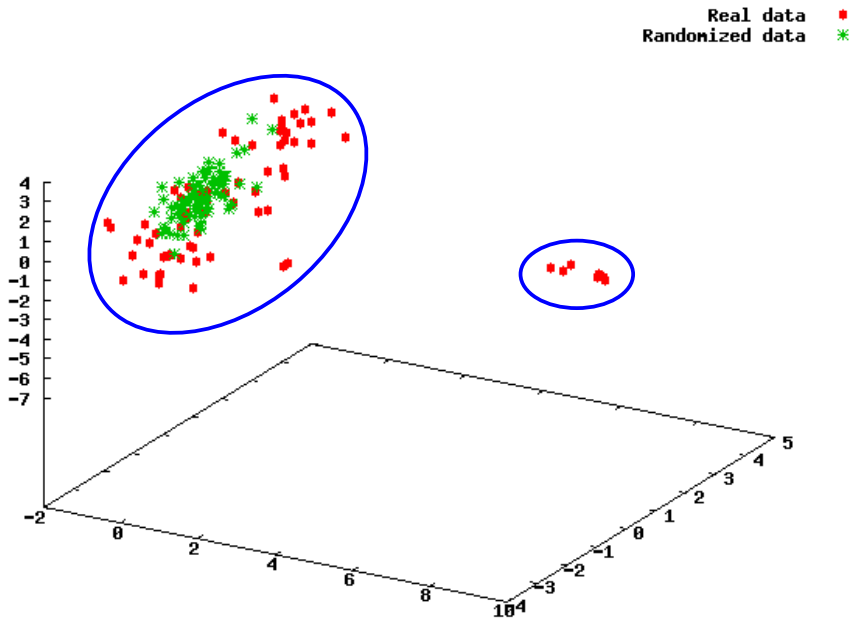

Figure 6-2. Two clusters detected in the eCRF dataset. The larger includes 63 patients and all randomly generated cases included for experimental purposes. The smaller and actually interesting is the second cluster with 11 patients that is obviously distinct from the first cluster.

In order to obtain information about the properties that differ the two detected clusters, we have constructed a classification task. The artificial class attribute was constructed in a manner which defines if a patient belongs to first or the second cluster. Using RF based variable importance calculation, attributes that are different in these two clusters were unveiled. Most relevant attributes are *Takes_Beta_Blockers*, *Takes_Diuretic*, *Takes_ACE* and *Takes_Loop_Diuretics*. By a closer insight into the data, it was discovered that the cluster of 11 patients differs from the rest of the patients on the matter of medicine usage: the cluster represents 11 people that do not use any medication or for whom those data have not been entered. These patients are also characterized by a vast number of other missing values. It means that the methodology managed to detect an interesting, but medically not very relevant property of one patient subpopulation that entered into the Heartfaid database.

Results of supervised descriptive analysis

Supervised classification for descriptive analysis can be implemented so that some property of interest is defined as the target concept. For the data collected by the platform, this can be done for very different tasks like discrimination between patients with and without endocrine disorders or discrimination between patients

with and without renal failure. In this section we report only on the results of two tasks: the discrimination between patients that have and patients that do not have diastolic dysfunction at the first visit, and the discrimination between patients with and without dyslipidemia at the first visit. For descriptive analysis we have used RF based detection of most important variables that characterize the target concept presented in Section 5.2.3. The results illustrate the type of the results that can be expected from the KD Web service in which RF algorithm will be implemented. Additionally, the section includes some results obtained by the SD methodology that stress the difference between result types of these two descriptive induction approaches.

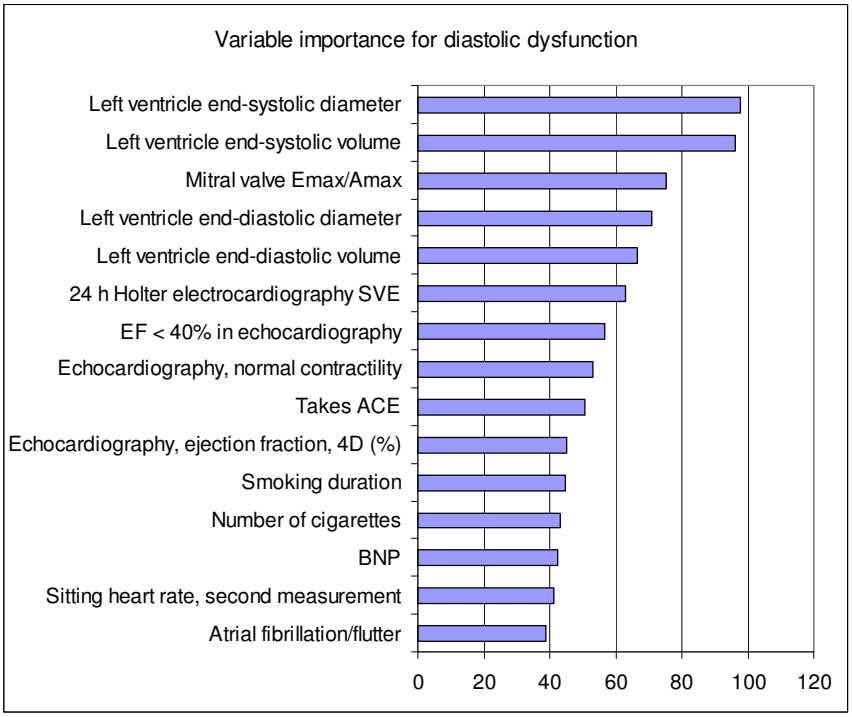

Figure 6-3. The list of variables that are at most significantly different between the patient that have and do not have diastolic dysfunction at the first visit. The values on the X-axis are only relative measure of variable importance in one experimental setting and cannot be used for comparison of variable importance among different experiments.

From the results obtained by the variable importance RF based tool and presented in Figure 6-3, it is clear that both left ventricular end-systolic diameter and volume are the most significant variables and almost equally relevant variables that characterize differences between patients with positive and negative diastolic dysfunction. It is interesting to notice that property *EF<40%* is ranked 7th, below the variables denoting supraventricular ectopic beats in Holter electrocardiography (6th place) and mitral valve *Emax/Amax* values (3rd place).

By application of subgroup discovery methodology, the obtained results are similar although the ordering of relevant properties is different. For example,

property *EF<40%* is ranked as the most significant, followed by end-diastolic volume and end-systolic volume. Generally, ranking obtained by RF approach can be accepted as more reliable due to the fact that they are the result of voting of many relative independent classifiers. That is the reason that RF will be used in the Web-based KD tool. However, the distinguishing property of the SD approach is the ability to detect not only relevant variables but also important decision points in these variables. For example, patients with diastolic dysfunction are characterized by end-diastolic volume less than 242 and by end-systolic volume less than 115.

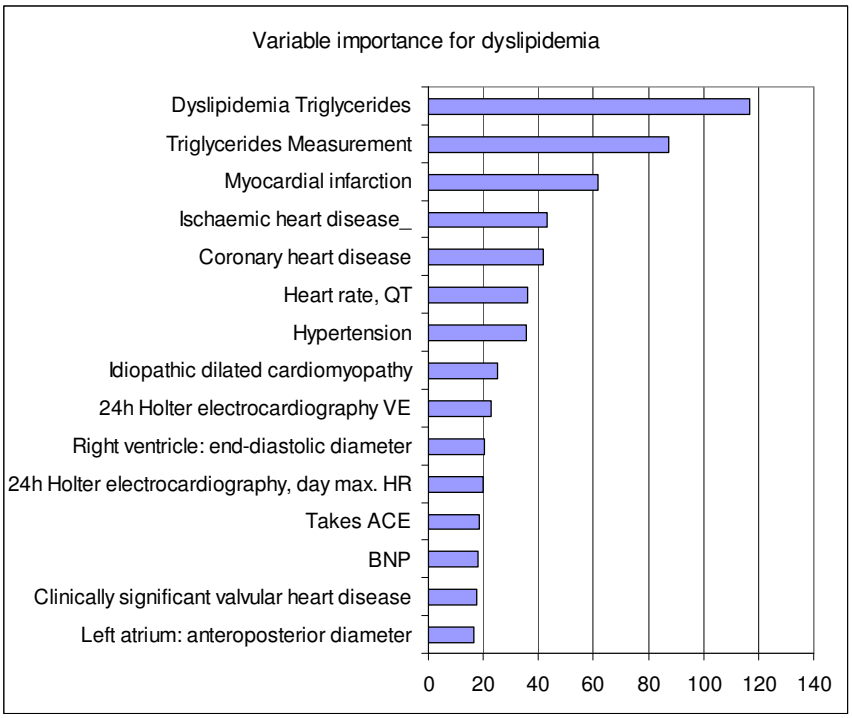

Figure 6-4. The list of variables that are at most significantly different between the patient that have and do not have dyslipidemia at the first visit.

Figure 6-4 presents the results obtained by analysing the patients with detected dyslipidemia at the first visit. From the most important variable *dyslipidemia_tryglicerides* it can be concluded that most of the patients that have detected dyslipidemia do have that because of increased tryglicerides and not because of cholesterol. This is confirmed also by the second most important variable selecting the measured values of tryglicerides as the distinguishing property for the two populations. In this sense expected is relevance of the variables correlated with the aetiology of HF: *myocardial_infarction*, *ischeamic_heart_disease*, and *coronary_heart_disease*.

It is interesting to notice that the SD approach selected following most important properties for patients with the dyslipidemia: *ischaemic_heart_disease* yes, *anemia* yes, *smoking* yes, *physical_activity* regular, and *endocrine_disorders* yes.

Although rather different from the results obtained by the RF approach, these results are also medically expected. When coexisting factors have been tested, the most relevant description of patients with dyslipidemia has been defined as:

Patient with dyslipidemia →

fatigue	*YES*
peripheral_oedema	*NO*
pathological_Q_waves	*NO*

Detection of such coexisting factors is potentially relevant for medical evaluation and understanding of the data collected by the platform. The SD tool will enable their detection also through the Web-based KD service. The open issue is still the way for optimal presentation and potentially visualization of such results.

6.2.3. HF severity models

Current medical practice exclusively uses NYHA classes for the estimation of the severity of the heart failure. The advantages of this classification are that the classes correspond to subjective patient problems, that they can be relatively easy determined, and that current medication practice is largely dependent on these classes.

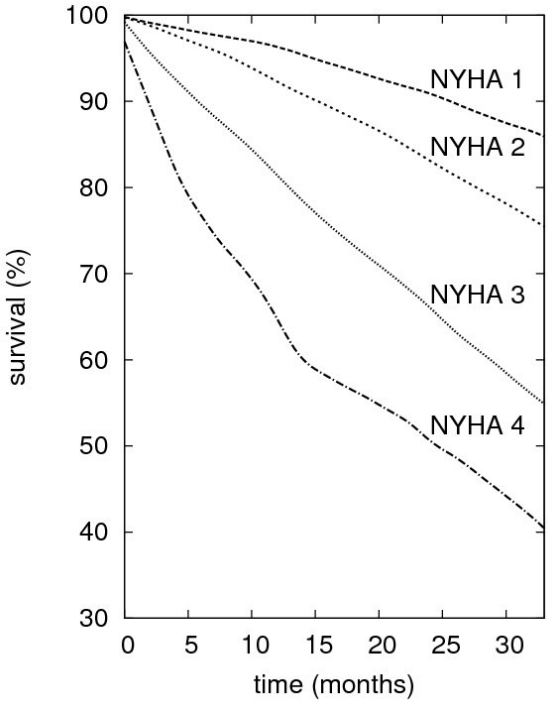

Figure 6-5. Survival rate for different NYHA classes in the ANMCO dataset

It is known, and it is also confirmed by our analysis of ANMCO data, that NYHA class also significantly determines the prognosis of HF patients. This is presented

in Figure 6-5 for the survival rate and in Figure 6-6 for the non-hospitalization rate.

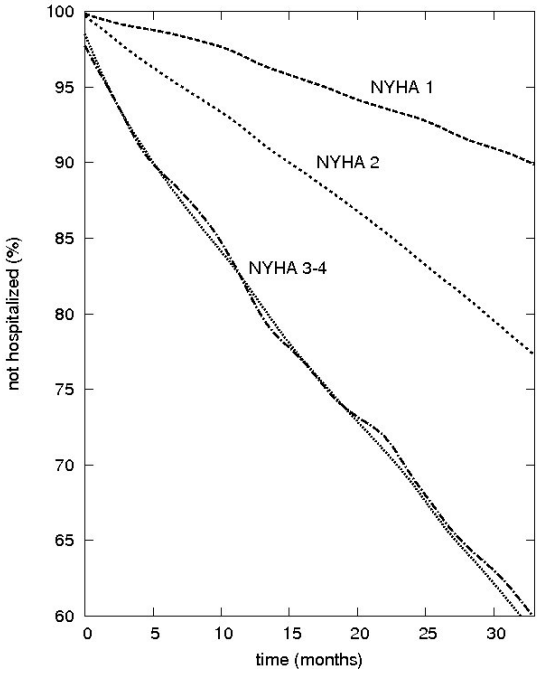

Figure 6-6. Probability of the first hospitalization for different NYHA classes in the ANMCO dataset

Our experiments have been concentrated on two sub-goals. The first is description of the current patient NYHA class by some other medical properties. The second is construction of another severity scale that could be useful for HF patient status description and his/her prognosis.

NYHA class description by patient properties

The dataset used for these experiments is the ANMCO dataset where NYHA class determined at the first visit is used as the target attribute. For its description we have used only attributes collected at the first visit what means that all data reporting on later events, like hospitalizations, have been neglected. We have also eliminated data about medication because this information actually represents the reaction of medical doctors on the patient status and in this way it is not useful for objective and independent estimation of the NYHA status.

There are four NYHA classes and this enables construction of a few different binary classification tasks. The general one-versus-all approach is not good in this situation because the classes are ordered in their meaning and severity. Much more appropriate is grouping of classes so that the target class includes NYHA III and NYHA IV while patients in NYHA I and NYHA II build the opposite non-target class. This grouping is partially also supported by the results presented in Figure 6-5 and Figure 6-6 The grouping enables detection of properties of difficult

HF patients in respect to those with relatively mild symptoms. The dataset is called "nyha34" and it is used in all experiments reported in this section. In total 25.8% of patients has been in classes III or IV at the first visit. For men this probability is lower, 25.0% in contrast to 28.3% for women. There are much more male HF patients in the dataset (73.0%) but their chance to be among heavy HF patients is slightly smaller.

By the application of feature ranking methodology described in Section 5.5 we have selected the most significant properties characterizing patients in NYHA classes III and IV.

Table 6-9. Most significant properties of patients in NYHA classes III and IV

Rank	Property	Sens.	Spec.
specific properties			
1	pacemaker = yes	9%	96%
2	third tone = yes	29%	85%
3	atrial fib. = yes	25%	84%
4	diastolic BP < 67	15%	91%
5	PTCA = yes	7%	94%
6	left branch block = yes	23%	81%
7	arrhyt = yes	4%	96%
8	bypass =yes	15%	88%
general properties			
1	diastolic BP < 75	39%	69%
2	age > 50	92%	13%

By the subgroup discovery methodology a few interesting patterns have been detected. For the negative class (patients in NYHA I and NYHA II) relevant is combination of "*third tone = no*" and "*atrial fib. = no*". In Figure 6-7 it is illustrated that the probability for the patient to be in the negative class is significantly higher if both properties are present.

For the positive class two patterns are more relevant. The first is combination of "*sex = M*" and "*left branch block =yes*". This means that the probability that a male patient is in the positive class increases from value 25.0% (characteristic for the complete male population) to 29.5% if it known that the patient has left branch block. This is an expected result knowing that left branch block is a risk factor for the HF. In that sense it is a surprise that the second pattern is "*sex = F*" and "*left branch block = no*", stating for female patients that the probability of being in the positive class *decreases* if the it is known that the patient has left branch block. This surprising result is presented in Figure 6-8.

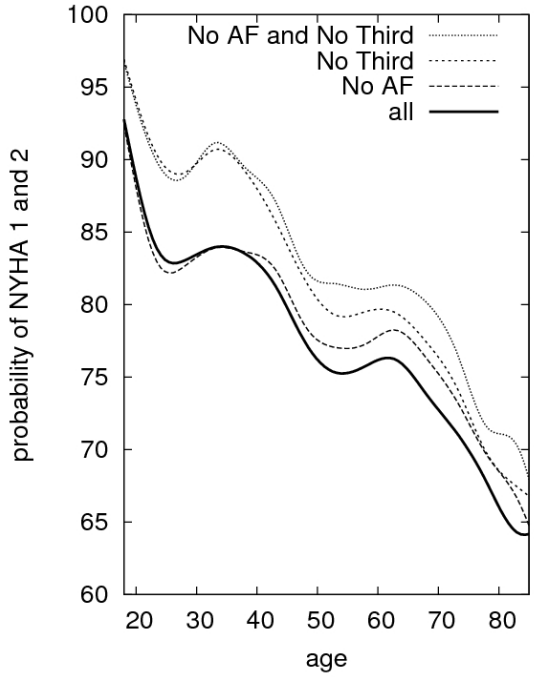

Figure 6-7. Probability to be in the NYHA I or NYHA II depending on age for different patient subgroups

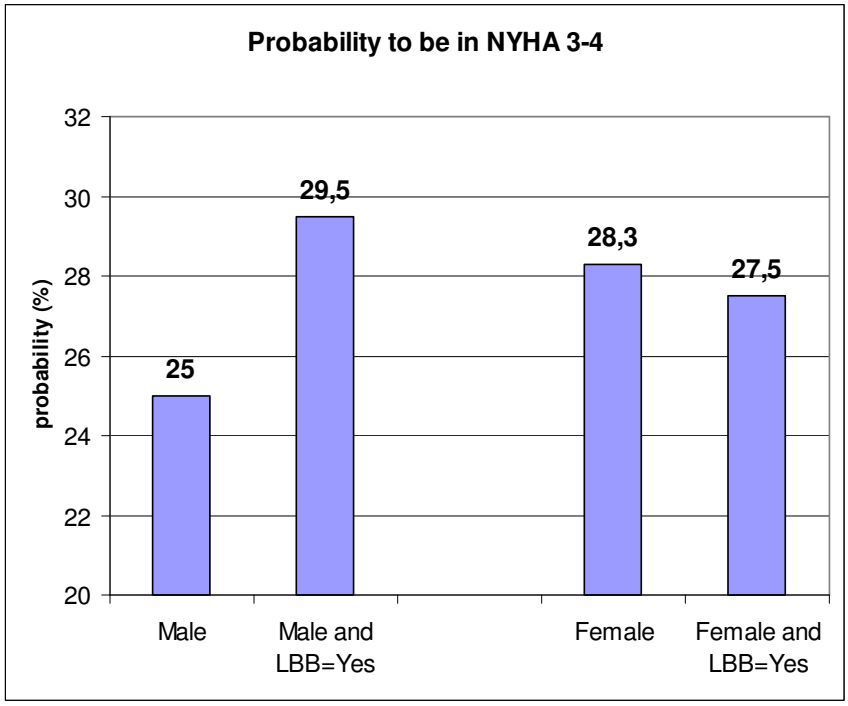

Figure 6-8. Illustration of a surprising effect that the property of having left branch block increases the probability to be in NYHA III or NYHA IV for male patients while it decreases this probability for female patients.

Novel HF severity scale based on data mining results

The most significant properties of patients being in classes NYHA III or NYHA IV listed in Table 6-9 are in accordance with significant properties detected for short survival and fast hospitalization. It does not only mean that NYHA class is a very good HF severity estimator but that other properties confirming NYHA status are identical with properties correlated with bad prognosis. This characteristic opens a possibility to try to construct an other scale based on objective patient properties.

In general, this is not an easy task. It requires deep understanding of the medical domain and understanding of the best-suggested medical practice. Additionally, it is questionable if the available ANMCO dataset is the appropriate starting point. However, in order to illustrate the potentials of the knowledge discovery methodology at this point we will assume that the dataset is a good representation of the general HF patient population and that combination and integration of different patient properties is medically justified. The knowledge discovery approaches with the goal of constructing novel example attributes (features) are *constructive induction tasks* which are known to be very difficult although very relevant both as independent results as well as a part of predictive model construction process.

In our work we started from the results presented in Table 6-9 and used human interpretation of the relevance of the independent properties to combine them into an attribute called HF severity scale (HFSS). The definition of scale is presented in Table 6-10. At first, we have decided to combine only categorical patient properties in order to avoid the problem of selecting the most appropriate decision point for the very relevant diastolic blood pressure value. Especially because it is known that blood pressure measurements can vary significantly from one measurement to another. Additionally, based on the ordering of the properties we have decided to group three most relevant properties into group A and the remaining four properties into group B.

Table 6-10. Definition of the novel HF severity scale (HFSS)

Risk Factors Group A	
pacemaker = yes	
atrial fib. = yes	
third tone = yes	
Risk Factors Group B	
arrhyt. = yes	
left branch block = yes	
PTCA = yes	
bypass =yes	
HF Severity Scale (HFSS)	
1	nothing from A and nothing from B
2	one from group A and nothing from B
3	one from group A and one from group B
4	at least two from group A

It must be noted that presented definition of HFSS classes is the result of human reasoning and that different scales could be defined using the same patient properties. Experiments with different scales based on the current medical practice are potentially the further work if we are able to demonstrate the potential usefulness of the process.

In order to test the significance of the novel scale we have repeated the experiments from Figure 6-5 and Figure 6-6 but now with the HFSS scale instead with the NYHA classes. The results are presented in Figure 6-9 for the survival of HF patients and in Figure 6-10 for their first hospitalization rate.

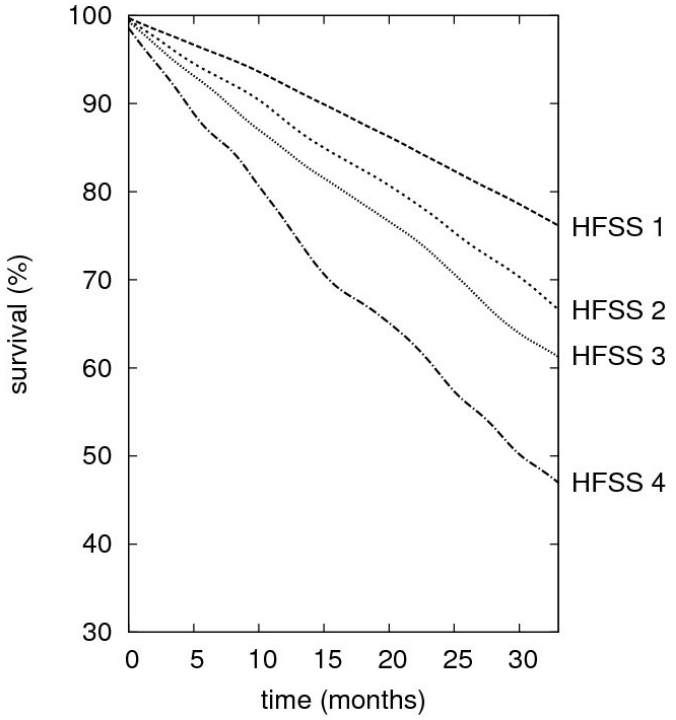

Figure 6-9. Survival rate for different HFSS classes in the ANMCO dataset

From these figures it can be concluded that HFSS behaves very similarly to the NYHA classes. The major difference is only in respect to the sensitivity because, for example the differences in respect to survival are larger between NYHA classes I and IV than between HFSS classes I and IV.

However, NYHA and HFSS scales are significantly different because they describe different patient properties. This can be concluded also from Table 6-11 which presents distributions of patients in different classes for both scales. Additionally, Table 6-12 presents the distribution of patients from different HFSS classes for each NYHA class.

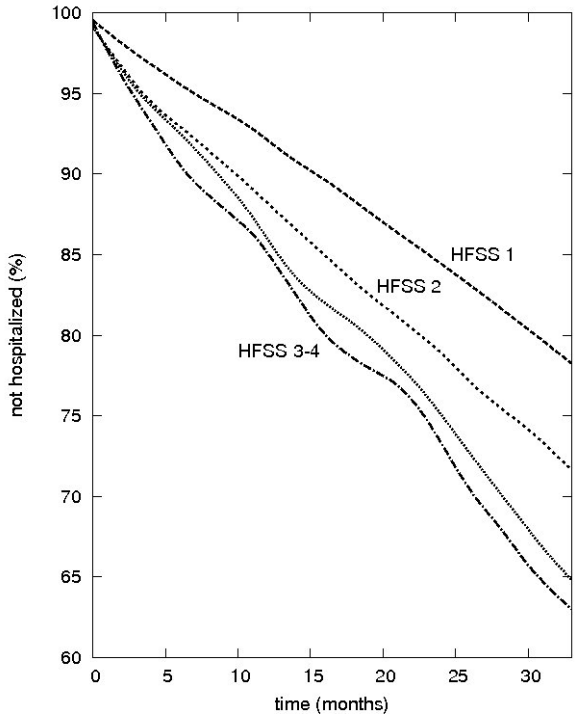

Figure 6-10. Probability of the first hospitalization for different HFSS classes in the ANMCO dataset

Table 6-11. Percentages of patients in different NYHA and HFSS classes

Class	NYHA	HFSS
1	15.1%	61.1%
2	59.1%	23.1%
3	23.8%	12.1%
4	2.0%	3.7%

Table 6-12. Percentages of HFSS classes for each NYHA class

		HFSS class			
		1	2	3	4
NYHA class	1	78.0%	14.8%	6.3%	0.9%
	2	63.9%	22.4%	11.1%	2.6%
	3	46.4%	29.1%	17.4%	7.1%
	4	28.2%	33.9%	23.8%	14.1%

The fact that HFSS scale is significant but different from the NYHA class means that HFSS scale cannot substitute NYHA classification but it may be useful by giving additional, potentially relevant information about patient status. This is illustrated in Figure 6-11 presenting how the survival changes for different NYHA classes if we have information that the patient HFSS class is larger than 1.

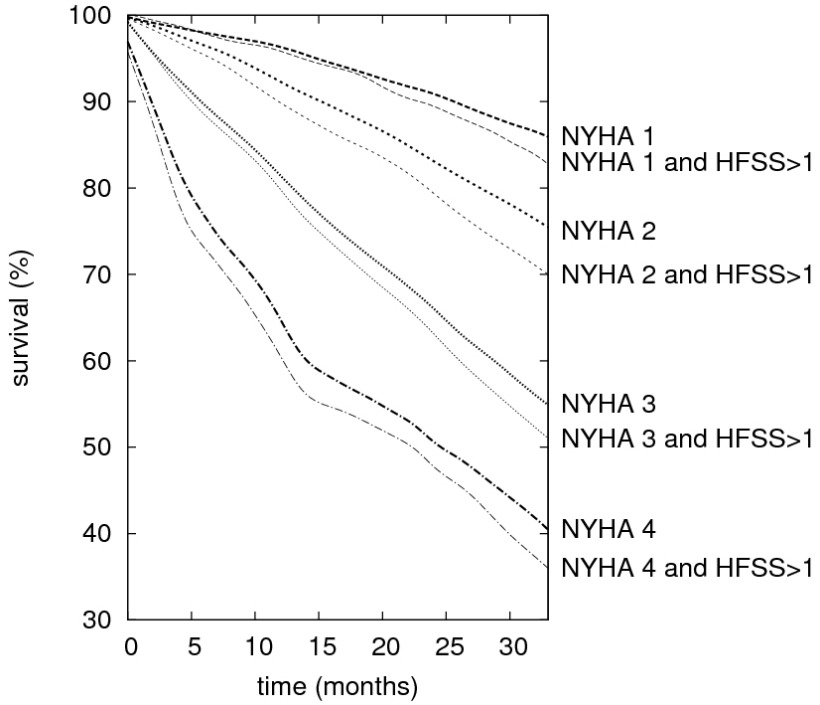

Figure 6-11. Survival rate for different NYHA classes with and without information about HFSS class being larger than 1

Finally, let us demonstrate that HFSS scale is more stable in respect to the relation to the sex than the NYHA scale. It is well known that there are more male HF patients (73% males versus 27% females in the ANMCO dataset) and that they die sooner. The latter fact is confirmed by the graph presented in Figure 6-12. A surprising effect has been noticed when the same graph has been presented for each NYHA class separately (Figure 6-13). For NYHA classes II and III the result is as expected, while for NYHA classes I and IV females tend to die earlier than males! The result is even more surprising when it is noticed that the number of male patients in class I is also unexpectedly high (81.4% males versus 18.6% of females). Knowledge discovery methodology was unable to give a medically reasonable answer. However in contrast to that, HFSS scale is completely consistent and in every class males die sooner than females. The result is presented in Figure 6-14.

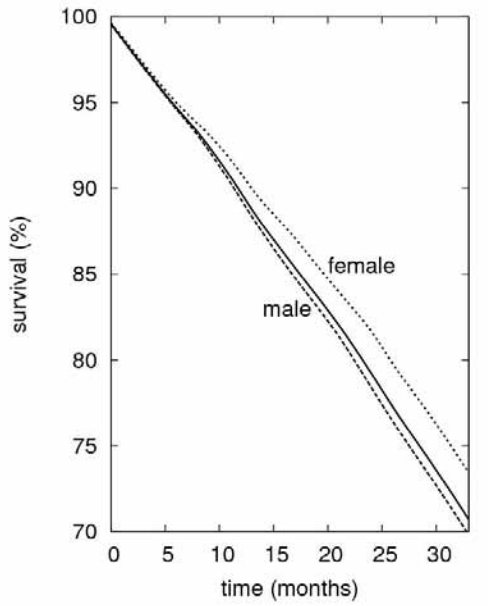

Figure 6-12. Survival rate for male and female patients in the ANMCO dataset. Thick line presents the mean value for the complete population.

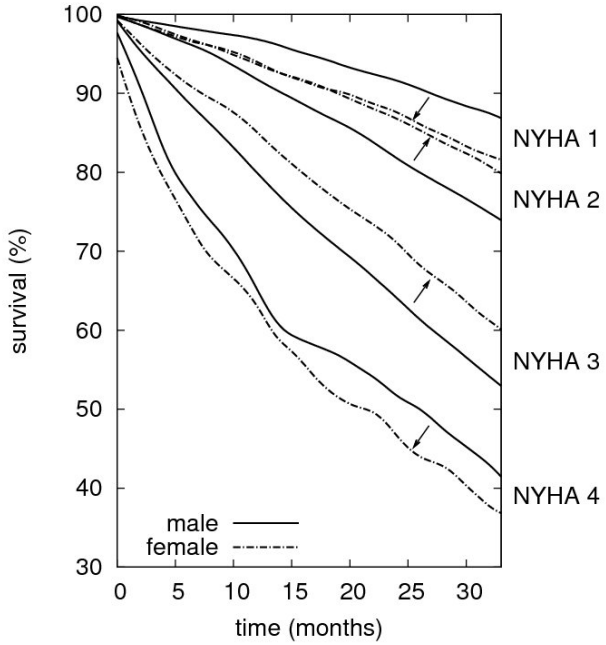

Figure 6-13. Survival rate for male and female patients separately for each NYHA class. Arrows present the direction from the curve for males to the curve for females demonstrating the differences among NYHA classes.

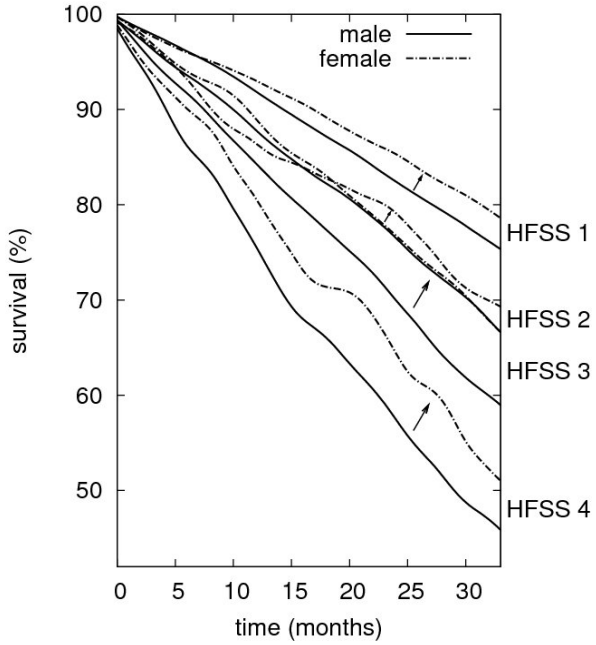

Figure 6-14. Survival rate for male and female patients separately for each HFSS class. Relation between male and female patients is consistent for all classes.

6.2.4. Prognostic models based on classification learning approach

We have analyzed the ANMCO dataset presented in Section 6.1.3 with the following goals:

a) To evaluate methodology for producing patient specific prognostic models

b) To derive useful HF prognostic models

c) To identify significant factors that affect HF prognosis

For the analysis we have used modern machine learning classification techniques combined with state-of-the-art variable selection techniques. We have also applied standard survival analysis techniques to preprocess the data.

We have achieved goals (a) and (c) above while we have medium success with goal (b). It can be expected that prognostic models should be updated and evolved over time as the data characteristics change. So, arguably the methodology for producing the models is as important as the models themselves.

ANMCO dataset preparation: further steps

In addition to the preprocessing of the ANMCO data as described in Section 6.1.3 we have introduced missing-value indicator variables (see Section 4.2.1). For each variable with missing values (40 variables overall) and name of the form VARNAME a corresponding indicator variable was added with name VARNAME_M. Each missing numerical/categorical value is substituted by the average/mode of that variable over the complete training dataset. This provides

the information to the classifier about imputed values. The classifier may decide to give less weight to an imputed value, or discover that some missing values are not at random but are proxies of prognosis.

We have also added 126 binary variables, one for each hospital participating in the study, each indicating whether a particular patient record is available from a specific hospital or not. This way, we allow the classifiers to learn and specialize their prognosis depending on the specific hospital subpopulation. The names of these variables take the form HCOD=X, where X is the corresponding hospital code.

ANMCO dataset preparation: conversion to classification tasks

The ANMCO dataset contains information on the following heart failure-related adverse events:

Table 6-13, Description of adverse events in the ANMCO dataset.

Outcome	Description
DEATH	Patient's death
HOSP	Hospitalization for HF causes
HOSP_TOT	Hospitalization for all causes
WORS_SC	HF Worsening
ARRHYEVENT	Arrhythmic event
ISCHEMEV	Ischemic event
STROKE	Stroke
SYNCOPE	Syncope
EMBOLUS	Embolus
CCH	Cardiac surgery
NYHAVAR	Variation of class NYHA.

Based on the above we have defined two additional events that we decided to focus on:

- *H*: the patient is rehospitalized after the first visit for any cause or died
- *G*: the patient experiences any of the above adverse events

The reason for focusing on *H* and *G* is their medical importance and because preliminary experiments showed they are some of the most promising events for accurate prognosis. Some of the data are right-censored (as expected in this type of data), i.e., there are patients for whom the time of experiencing the adverse event is not known (see Section 5.3). This could be because they did not experience the event before the data collection ended, or were simply lost for follow up. Removing right-censored records contains well-known pitfalls that skew the data distribution. To convert the problem to a standard classification task we have followed a common practice: we have selected a threshold *t* (within a set of thresholds) and defined the following tasks: For each threshold t and each

adverse event A, predict whether A will occur before or after t in respect the first visit of the patient.

The above tasks define three classes of patients:

a) Those known to have experienced the adverse event before time *t* (class label 1). These are the patients with time-to-event less than *t*.

b) Those known not to have experienced the adverse event before time *t* (class label 0). These are the patients with time-to-event *or* time-to-last-follow-up greater than *t*.

c) Those for which it is unknown whether they have experienced the adverse event before time *t*. These records are removed from the training set for the given task. These are the patients not having experienced the event until time *t and* time-to-last-follow up is less than *t*.

Assuming that patients are lost to follow up for non-disease specific reasons, removing the above records does not affect the data distribution.

We have selected as thresholds times 3, 6, 9 and 12 months from the first visit and the events H and G above, thus defining 8 different classification tasks.

As already mentioned the data distribution is not expected to be stationary but changing over time. In order to use standard learning algorithms that assume identically and independently distributed data (i.i.d.) we tried to select a time period where the data distribution remains relatively stationary. Other approaches require to explicitly handle non-stationarity, e.g., to weight less the older patient records, etc. However, this would require significant technical complications when using standard algorithms and software.

For each year worth of data and each event *H* and *G* we have estimated the survival curves and used a Mantel-Cox log-rank test [2, 3, 4] for testing the null hypothesis that the data population collected during year *X* is the same as the data population collected during year *Y*. The results for event *H* are presented in the following table; a value less than 0.05 is typically considered statistically significant to allow the rejection of the null hypothesis and accept the hypothesis that data from year *X* has a different distribution than data from year *Y*.

Table 6-14. The p-values of a Mantel-Cox log-rank test for comparing the survival curves for the ANMCO data between the subpopulation of patients for each pair of a year's worth of data. The table regards the adverse even "H: the patient is rehospitalized after the first visit for any cause or died".

YEAR	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005
1995		.011	.000	.000	.000	.000	.000	.000	.000	.000	.000
1996	.011		.000	.000	.000	.000	.000	.000	.000	.000	.000
1997	.000	.000		.011	.225	.001	.000	.000	.103	.013	.000
1998	.000	.000	.011		.130	.565	.021	.025	.427	.758	.071
1999	.000	.000	.225	.130		.053	.000	.000	.442	.045	.006
2000	.000	.000	.001	.565	.053		**.084**	**.074**	**.407**	**.926**	**.075**
2001	.000	.000	.000	.021	.000	**.084**		**.898**	**.005**	**.400**	**.079**
2002	.000	.000	.000	.025	.000	**.074**	**.898**		**.014**	**.364**	**.434**
2003	.000	.000	.103	.427	.442	**.407**	**.005**	**.014**		**.050**	**.015**
2004	.000	.000	.013	.758	.045	**.926**	**.400**	**.364**	**.050**		**.209**
2005	.000	.000	.000	.071	.006	**.075**	**.079**	**.434**	**.015**	**.209**	

Ideally, we would like the data distribution to be indistinguishable. We have tried to identify the more recent run of consecutive series of years in which the data seem to have similar distributions. We have decided to use the data entered from year 2000-2005. The p-values from the comparison of each year with each other year are shown in bold in the Table 6-14. Most of them are above 0.05. We decided to use the data entered at year 2000-2003 (inclusive) for training and the data from years 2004-2005 for testing. The test set was never available during training either for variable selection for model production. The survival curves of the training and testing subpopulations are shown in Figure 6-15. The survival curves do have small differences.

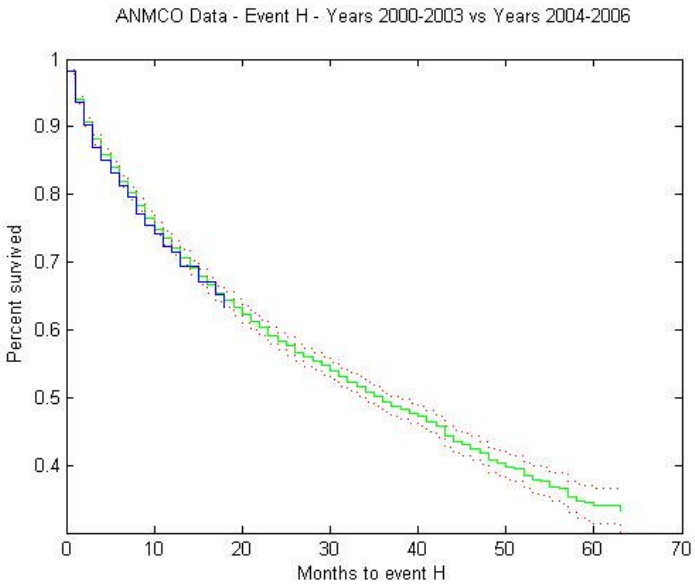

Figure 6-15. The survival curves for event H comparing the training data (in bold for years 2000-2003) and the test data (thin line for years 2004-2005). The dotted lines are the 0.95 confidence intervals

Obviously, in practice one cannot test whether the training population has similar distribution to the testing population. However, if the prognostic models are regularly updated using only the most recent data, one can hope that the patient distribution in the near future will remain stationary (assuming no major changes in medical measurement equipment, treatment procedures etc. have been implemented). We have repeated the procedure for event G. The table with the p-values is shown below:

Table 6-15. The p-values of a Mantel-Cox log-rank test for comparing the survival curves for the ANMCO data between the subpopulation of patients for each pair of a year's worth of data. The table regards the adverse even "G: the patient experiences any of the adverse events listed in Table 6-13"

YEAR	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005
1995		.533	.317	.378	.452	.000	.000	.000	.005	.000	.000
1996	.533		.083	.099	.134	.000	.000	.000	.004	.000	.000
1997	.317	.083		.876	.798	.000	.000	.000	.175	.001	.000
1998	.378	.099	.876		.924	.000	.000	.000	.100	.000	.000
1999	.452	.134	.798	.924		.000	.000	.000	.055	.000	.000
2000	.000	.000	.000	.000	.000		**.705**	**.485**	.065	.324	.000
2001	.000	.000	.000	.000	.000	**.705**		**.584**	.026	.159	.000
2002	.000	.000	.000	.000	.000	**.485**	**.584**		.028	.243	.000
2003	.005	.004	.175	.100	.055	.065	.026	.028		.000	.000
2004	.000	.000	.001	.000	.000	.324	.159	.243	.000		.000
2005	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	

A quick inspection of Table 6-15 indicates that it is much harder to identify a consecutive set of years with similar survival characteristics. We have identified years 2000-2002 as our target data; the p-values of their pairwise comparisons are shown in bold in the Table 6-15. We decided to use the data entered at year 2000-2001 (inclusive) for training and the year 2002 for testing. The test set was never available during training neither for variable selection nor model production. The survival curves for the training and testing subpopulations are shown in Figure 6-16.

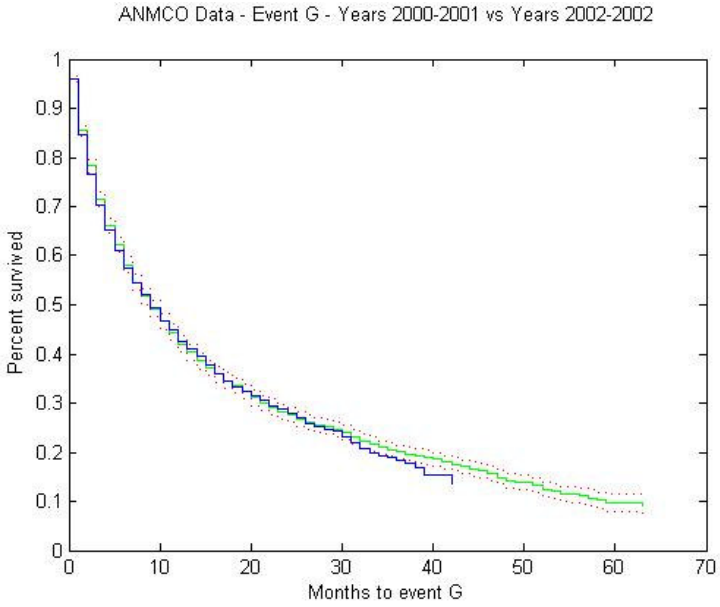

Figure 6-16. The survival curves for event G comparing the training data (in bold for years 2000-2001) and the test data (thin line for year 2002). The dotted lines are the 0.95 confidence intervals.

The survival curves are relatively similar in the first few months after the first patient hospitalization. Recall that, for each selected threshold *t* some records must be deleted from the respective dataset. In addition, for each threshold the class of each patient may be different. Thus, each threshold corresponds to a different training and test dataset. In the table below, we show some datasets characteristics, for each event and threshold *t*.

Table 6-16. Characteristics of the training and test datasets for each event (H, G) and each time threshold t (3, 6, 9, 12 months). #Pos(#Neg) is the number of patients that experienced (did not experience) the event before time t for the corresponding event and time threshold. Pos (Neg) is the percentage of the #Pos(#Neg) in the total patient distribution.

			Training				**Test**			
			Years 2000-2003				**Years 2004-2005**			
		t	# Pos	# Neg	Pos	Neg	# Pos	# Neg	Pos	Neg
Event	**H**	3	704	6628	0.10	0.90	172	1427	0.11	0.89
		6	1188	5740	0.17	0.83	278	985	0.22	0.78
		9	1557	5019	0.24	0.76	340	613	0.36	0.64
		12	1869	4432	0.30	0.70	373	392	0.49	0.51
			Years 2000-2001				**Year 2002**			
			# Pos	# Neg	Pos	Neg	# Pos	# Neg	Pos	Neg
	G	3	839	2954	0.22	0.78	450	1434	0.24	0.76
		6	1438	2265	0.39	0.61	736	1079	0.41	0.59
		9	1819	1821	0.50	0.50	889	836	0.52	0.48
		12	2073	1527	0.58	0.42	1001	684	0.59	0.41

The prior class distribution between the training and test sets are in general in good accordance for most classification tasks. There are however, some notable differences too: event H and thresholds 9 and 12.

Computational experiments and modelling

The prognostic modelling task was divided between the FORTH and UNICAL groups, led by Dr. Tsamardinos and Dr. Lagani respectively. The two groups used different variable selection and classification methods, different methods for optimizing the parameters of the learning algorithms and selected a few final models for each task. This allowed several algorithms to be applied in a search for the best prognostic models. It can be noted that the total number of final models selected for each task is quite small (5 per task) that makes overfitting of the test sets due to multiple testing unlikely, especially considering that every test set contains at least a few hundred cases.

The experimentation protocols, the methodologies and the results of both groups are explained in the following paragraphs. A final section related to the comparison of the obtained models is also included.

FORTH – Experimentation protocol and algorithms

For each of the above classification tasks and corresponding training set, we produced several prognostic models. A first class of models was produced using the following procedure. We applied a variable selection algorithm or no variable selection and optimized a support vector machine model over a range of parameters. In order to optimize the SVM parameters and at the same time to produce an unbiased estimation of the performance of the model, we used a

double-nested 3-fold cross-validation procedure. The procedure is complicated and details are given in [5]. In short, the procedure produces a model by finding the best-performing parameters of the classifier, while at the same time guaranteeing that the performance estimate of the model is unbiased. The simple procedure of training classifiers with all possible combinations of parameters, selecting the one with the best cross-validation performance, and reporting that performance as the final performance estimate may be subject to overfitting and overestimation due to multiple testing. If one tries numerous parameter options one is likely to find a set that exhibits high performance on the test sets by pure chance.

The classification algorithms used are the 1-norm, soft-margin support vector machines with both polynomial and Gaussian kernels (see section 5.6 for details about SVMs). The ranges of the cost parameter and the kernel parameters were optimized within the following ranges:

- Cost within {0,001 , 0,1 , 10, 1000}
- Degree of polynomial kernel within {1, 2, 3}
- Gamma parameter of the Gaussian kernel within {0.01 , 0.1 , 1}

For selecting variables we used HITON-PC algorithm that is based on Bayesian network theory (see [6] and Section 5.8.1 for details). HITON-PC does not theoretically guarantee to return the Markov blanket, i.e. the minimal set of variables with the maximum prediction performance, but it is much more time-efficient than the theoretically sound counterpart (HITON-MB) while extensive experimentation has shown that there is no significant reduction in quality of selecting variables [7].

For each combination of variable selection (no variable selection and HITON-PC), SVM kernel and classification task the best model was produced with the optimal parameters as selected by using the above double-nested cross-validation procedure. The model was then applied to the test data and its performance was evaluated.

FORTH – Results

The performance metric reported for the models is the area under the receiving operating characteristic curve (AUC) (see Section 4.4.1 for details). Recall that this is a metric independent of the class distribution and the misclassification costs between classes. A synopsis of all results is shown in Table 6-17. A synopsis of the results of the best-performing models in the *test set* is shown in

Table 6-18 along with their parameters. Finally, the best-performing models in the test set that *employ variable selection*, along with the selected variables are shown in Appendix A2. The variables in Table A2-1 are ranked according to the strength of their pairwise association with the class variable.

Table 6-17. The area under the ROC curve (AUC) (times 100) for each classification task on the training and corresponding test set. For both training and test sets, the highest-performing model in each row is shown in bold. NVS means "no variable selection" method.

			Training				Test			
			Years 2000-2003				Years 2004-2005			
		t	Poly NVS	Poly HITON	Gaussian NVS	Gaussian HITON	Poly NVS	Poly HITON	Gaussian NVS	Gaussian HITON
Event	H	3	65.3	57.48	**66.20**	55.76	58.60	**64.88**	58.00	64.18
		6	66.76	58.61	**67.47**	62.33	61.74	61.69	61.70	**62.77**
		9	68.95	66.69	**69.45**	65.26	67.05	**68.52**	66.46	68.18
		12	70.66	67.33	**70.73**	69.13	73.28	69.20	**73.55**	70.92
			Years 2000-2001				Year 2002			
			Poly NVS	Poly HITON	Gaussian NVS	Gaussian HITON	Poly NVS	Poly HITON	Gaussian NVS	Gaussian HITON
	G	3	70.13	68.93	**70.24**	68.65	65.74	**72.75**	71.42	69.65
		6	70.60	69.65	**70.91**	69.51	74.68	73.69	**74.81**	73.62
		9	70.85	69.62	**71.51**	69.61	73.73	70.69	**74.59**	71.88
		12	**71.72**	70.17	71.68	70.32	73.32	70.33	**74.06**	70.47

Table 6-18. Parameter specifications of the best-performing models. Each model used either no variable selection method (NVS) or the HITON-PC algorithm for selecting variables. VS means variable selection method. The kernel of the SVM was either Gaussian or Polynomial (denoted by Poly in the table). C is the cost parameter of the SVM, d is the degree of the polynomial kernel, and γ is the gamma parameter of the Gaussian kernel.

		Training		Test	
	t	AUC	VS and SVM parameters	AUC	VS and SVM parameters
H	3	66.20	Gaussian, NVS C=10, γ=0.01	64.88	Poly, HITON-PC, C=0.001, d=1
	6	67.47	Gaussian, NVS C=10, γ=0.01	62.77	Gaussian, HITON-PC C=1000, γ=0.01
	9	69.45	Gaussian, NVS C=10, γ=0.01	68.52	Poly, HITON-PC, C=0.001, d=1
	12	70.73	Gaussian, NVS C=10, γ=0.01	73.55	Gaussian, NVS C=10, γ=0.01
G	3	70.24	Gaussian, NVS C=10, γ=0.01	72.75	Poly, HITON-PC, C=0.001, d=1
	6	70.91	Gaussian, NVS C=10, γ=0.01	74.81	Gaussian, NVS C=10, γ=0.01
	9	71.51	Gaussian, NVS C=10, γ=0.01	74.59	Gaussian, NVS C=10, γ=0.01
	12	71.68	Poly, NVS,C=1000, γ =0.01	74.06	Gaussian, NVS C=10, γ=0.01

FORTH – Interpretation of results

Some conclusions specific to the above analysis are the following:

- The best performing models typically do not employ variable selection and use a Gaussian kernel with *C*=10 and γ=0.01. Whenever a polynomial kernel was selected, the degree of the kernel was selected by the nested cross validation to be 1. In other words, the linear kernels were performing better than quadratic or third degree polynomial kernels.

- A very interesting phenomenon is the following: *the models produced by applying HITON-PC for the H event (t=3, 6, 9) are typically underperforming on the training set and over-performing on the test set.* One possible explanation regards the way HITON-PC selects variables. HITON-PC, under certain broad assumptions, selects the variables that *directly cause or are caused-by* the variable to predict. Even if the distribution of the data changes, its local causal structure is less likely to change so these variables remain highly predictive. In contrast, if we do not apply variable selection or apply a non-causally based variable selection method, certain variables may be predictive in the training data distribution, but become non-predictive when the distribution changes. Consider the following fictitious example: physicians may prescribe medicine *A* (e.g., ACE inhibitors or sartans) if she is suspecting the presence of condition *C*. Medicine *A* is working by altering the levels of a factor *X* (e.g., EF, CF or PAS) that is a risk factor of the disease. Prescribing *A* is predictive of the presence of condition *C*; similarly, abnormal values of *X* are predictive of condition *C*. A classifier using both variables will predict with higher confidence the presence of *C* when *A* is prescribed and *X* is abnormal. However, if in the next year the medical guidelines change and now physicians prescribe medicine *B* instead of *A* in certain cases, then the classifier will wrongly deduce that the patients not taking *A* have a smaller chance of having the condition *C*. What HITON-PC does instead is identify that prescribing *A* is conditionally independent of having the condition *C* when the levels of *X* are known. In other words, it deduces that *A* does not provide more information for predicting *C* (is superfluous), when the levels of *X* are known. Under broad conditions, removing such variables retains variables that are causally closer to the variable to predict. HITON-PC will only select the levels of *X* to include in the selected variable subset. A classifier trained on just *X* instead of both *X* and *A* will perform equally well even after the new medical guideline is issued. From Table A2-1 we see that HITON-PC typically does not select variables corresponding to ACE inhibitors or sartans while it often selects the variables EF, CF and PAS.
- The results also show that variables NYHA and HOSP_PREV_YEAR, corresponding to the NYHA index and whether the patient was hospitalized zero, one or two, or more than three times in the previous year, are consistently selected by the variable selection method. Most models also contain a few other medical variables. It is worth making two observations: (a) the variables selected often include the hospital codes where the data were entered (variables of the form HCOD=X). This indicates that the data distribution is not only changing across time, but also geographically. Different hospitals treat different populations and it is natural that the algorithm chose to select some of these variables as predictive. In addition, the different methods for recording and measuring the patients' data may also make the data distributions across certain hospitals different. (b) Some of the indicator variables of the missing data do appear in the selected variables subsets by HITON-PC. Because of how the method works, this implies that they are predictive of the class variable

even in the context of the remaining variables and carry useful information. The finding justifies the inclusion of these variables in the data.

UNICAL – Experimentation protocol and algorithms

We decided to use a decision tree approach in order to produce "easy to understand" models that could be validated both via statistical measurement (e.g. AUC metric over cross validation results), and through the judgment of clinicians. Decision tree and their properties have been extensively explained in Section 5.1.

We used J48 implementation of the well-known C4.5 algorithm which is part of the data mining software WEKA. In order to avoid the problems related to the presence of unbalanced dataset, we used J48 in conjunction with a cost sensitive meta classifier which allows the specification of misclassification costs matrices. Such matrices are used within the inner procedures of J48 algorithm, in order to modify the normal building of the trees (for an explanation about cost sensitive classification, see section 4.3.2).

In order to define correctly the misclassification costs matrices, we used intra class ratios. That is, we calculated the ratios among the number of instances belonging to different classes. For example, in the case of the datasets related to the "H" outcome definition and to the 3 months threshold, the number of instances for the class "Event" was 704, in respect of 6628 "No event" instances.

Thus, the misclassification cost for an "Event" instance classified as "No event" was 6628/704 = 9.414 (approximated to 10), with a related misclassification costs matrix:

Table 6-19. Example of misclassification costs matrix.

"Event"	**"No event"**	**← Classified as**
0	10	"Event"
1	0	"No event"

The same procedure has been repeated in order to calculate the other misclassification costs matrices.

Regarding the experimentation protocol, our primary goals have been:

1. to produce predictive models with high performances in terms of AUC metric
2. to avoid overfitting of the models

Given the aforementioned goals, we defined the following experimentation protocol:

1. For each training set, we optimized the parameters of the J48 algorithm using the 10-fold cross-validation procedure. We used a paired t test in order to evaluate the differences in terms of performance among the several produced models. Models with performances not significantly different were considered equivalent. At the end of the optimization step, all the models with the best performances in terms of AUC metric were considered.

2. For each training set, we chose only one tree among the different models produced in the previous step. In particular, we chose the tree with the minimal number of nodes, following the Occam's razor principle (in our case, such principle can be interpreted in the following way: simpler models are better than the more complex ones). When more than one tree showed the same number of nodes, it has been chosen the model with less "false negative" misclassifications.
3. The best model chosen at the second step has been tested on the test set.

The test sets haven't been used during the phase of models building, in order to avoid biases in the results.

UNICAL - Results

The results in terms of AUC metric are summarized in the following table:

Table 6-20. The classification quality of best-performing decision trees.

			Training	Test
		t	AUC	AUC
Event	H	3	0.654	0.643
		6	0.659	0.653
		9	0.647	0.665
		12	0.643	0.664
	G	3	0.687	0.708
		6	0.672	0.710
		9	0.673	0.702
		12	0.667	0.686

The extracted models are reported in the Appendix A2.

UNICAL – Interpretation of decision trees

The most evident characteristic of the obtained decision trees is that the structures of all trees are very similar. In particular, the root node (and thus the most important variable) is the NYHA class in almost all cases, directly followed by the number of hospitalization in the year previous the first visit. More precisely, the NYHA class is often sufficient in order to provide a prognosis: NYHA I good prognosis, NYHA III and NYHA IV bad prognosis. If the patient has a NYHA class II, the models suggest to check the number of hospitalizations experienced in the previous year. Almost in all models, three or more hospitalizations lead to a bad prognosis, no hospitalization to a good prognosis, while the cases with a number of hospitalizations between one and tree require further information. The number and the type of further information vary.

It is easy to understand why the number of hospitalization in the previous year has a high predictive power for the outcome "H". In fact, if a patient had more than

tree hospitalization during the last year, it is reasonable that the same patient will have a high probability to experience the hospitalization again.

More complex is the interpretation of the high predictive power of the "HOSP_PREV_YEAR" attribute for the outcome "G". It can be stated that if a patient experiences an adverse event, the patient is probably also hospitalized. Under this point of view, a high number of hospitalizations during the last year means a high number of adverse events (maybe the same event repeated over time). Thus, patients with more than three hospitalizations during the last year will probably have an adverse event (that is the definition of outcome "G") after the first visit. A similar reasoning can be conceived for the patients with a low number of hospitalizations in the year previous to the first visit.

Last but not the least, it should be discussed also the great predictive value of the NYHA class. From the clinical literature point of view, such predictive value is widely known (see for example [8]). It should be reminded that the assignment of the patient to the correct NYHA class is a process that is still partially subjective [9]. Thus, our predictive models are based on a parameter that depends on the judgment of the visiting physicians. Further analysis and discussions on this and other points will be lead in conjunction with the medical partners.

Discussion of combined results

In Tables A2-2 - A2-9 we juxtapose the variables selected by the decision trees inherent variable selection method and the variables selected by HITON-PC. Regarding the decision trees, variables nearest to the root node have higher influence in the classification process, while the branches near to leaf nodes have reduced importance. Thus, we sort decision tree variables higher the closer they are to the root. Variables selected by HITON-PC are sorted by pairwise association with the class variable.

Based on the results presented in Table 6-17, Table 6-18, and Tables A2-2 to A2-9 there are several relevant observations.

- The results clearly show that there is useful information in the ANMCO dataset since the AUCs of all derived models are above the 0.5 value that is characteristic for the random classifier. However, most of the AUC's are in the range of 0.64-0.74 indicating that the prognosis using the models would have not have been that accurate, had the models been used during the testing-dataset periods. Although it is possible that other learning algorithms could significantly improve the performance, it seems more promising to augment the data with more information.
- The distribution of the timing of the adverse events regarding heart failure is not stationary and varies widely from year to year. This is not in general surprising because there are many changing environmental factors that affect heart failure: general lifestyle, medications, patient demographics, diet, etc. What is surprising however is that the distribution is changing with such high rate: every years' data seems to follow a different distribution to some degree. Perhaps, one reason that explains this observation is that the medical

technology for measuring the data is rapidly changing, as well as the information systems in the hospitals, both potentially affecting the way data is gathered and stored. Other factors that may play an important role could be medical guidelines that change relatively frequently and the medical staff's education, ethics and habits regarding storing the patients' data in the medical records. *Because of the non-stationarity of the process, we recommend to first identify the most recent subset of data that follows a similar distribution, before proceeding with any other statistical or machine learning technique that does not explicitly address non-stationary processes.*

- In several cases, there is a significant difference between the performances of the models in the training set versus the performance in the test set. This does not seem due to overfitting (some models often perform *better* on the test sets) but is probably best explained by the non-stationarity of the process. Thus, selection of the best-performing models is difficult: what seems to be the best model in the training data is not the best model when it is applied on new data.
- For several best-performing models, the prognosis for a patient depends on the hospital where the data is entered. This may be explained in two ways. First, the patient population varies among hospitals and so having different prognostic characteristics is justified: people in large city centres may get a different prognosis due to leading a different lifestyle than people in rural areas. Second, the way data is gathered, reported and stored in different hospitals may differ thus also changing the data distribution. Thus, great care is required to analyze such heterogeneous datasets that span a long time interval and large geographical areas.
- Two variables are consistently selected in all models: *NYHA* and *HOSP_PREV_YEAR*, in spite of the fact that quite different variable selection techniques have been used. These two variables represent respectively the NYHA class and the number of hospitalization (0, between 1 and 3, more than 3) during the year prior to the first visit. The prognostic value of the NYHA class is already well known and deeply studied in the medical literature. On the other hand, the number of previous hospitalizations is not largely employed in medical studies related to the survival analysis/prognostic evaluation of HF patients. Our results suggest to consider the number of hospitalizations as an important parameter, and to include such parameter in the future studies.
- The results in terms of AUC produced by the combination of no variable selection or variable selection using HITON–PC and learning a prognostic model using a SVM learner have been generally superior to the results obtained through the decision trees approach. A possible explanation for the performance difference may be that the simplest trees were selected out of the best-performing trees in the training set. However, the decision trees have the advantage that they employ fewer variables to determine the prognosis and are easier to interpret by a human expert.

6.2.5. Prognostic models based on survival analysis approach

Prognostic modelling using survival analysis approach, survival random forest (SRF) and conditional probability trees (CPT) allow us to treat more accurately the right-censored data in the ANMCO follow-up study and obtain time dependent survival probability profiles of distinct risk groups or even individual patients. For this particular analysis, we have done data pre-processing having in mind specific constraints for the enrolment of patients into the Heartfaid platform. As the platform is intended for the improved health care of elderly patients, we have extracted only the patients of the age 65 and above. To obtain more insight and possibly more accurate prognostic models, splitting into separate male and female datasets was performed. The events we have analysed are limited to:

a) death by cardio-vascular reasons

b) hospitalization events

c) worsening of heart failure

The results for SRF analyses that follow, encompass the accuracy of the model and variable importances and CP decision tree models as descriptive models, separately for the female and male population. Due to their extension, in this section we present only the results for the death by cardio-vascular reasons. The results for the events b) and c) are given in the Appendix A3 for the interested reader, and follow the same structure of presentation.

Treatment of missing values

We eliminate missing values which are result of negligent data collection, for instance: some patients could have been very problematic and refused to give the needed data, or some medical workers that collected data could have been careless and forgot to retrieve or to write down such data. In order to detect such cases we followed this simple logic: if there are attributes that should be collected for every patient, then missing values in these attributes could be indicative for negligent data collection and these patients should be excluded from the dataset. Attributes could be divided in several categories: physical examination, patient history, laboratory results, investigations, drugs and target attributes. For every patient we should know physical examination attributes (age, sex, height, weight...). If for example we do not know the sex of the patient, we can conclude that this is a problematic example and we can exclude it from the dataset.

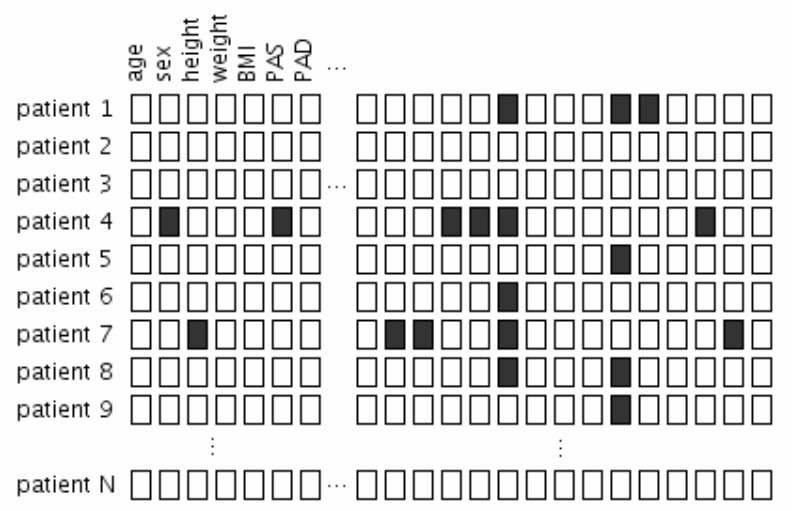

Figure 6-17. Graphical representation of the dataset. Missing values are marked with black rectangles.

This process is illustrated in Figure 6-17 and Figure 6-18. Figure 6-17 shows us an example of the dataset. Patients are listed in rows, attributes in columns. Black fields denote missing values. We search through physical examination (PE) attributes for missing values. When we find a missing value, we exclude from the dataset a row to which this missing value belongs to (Figure 6-18).

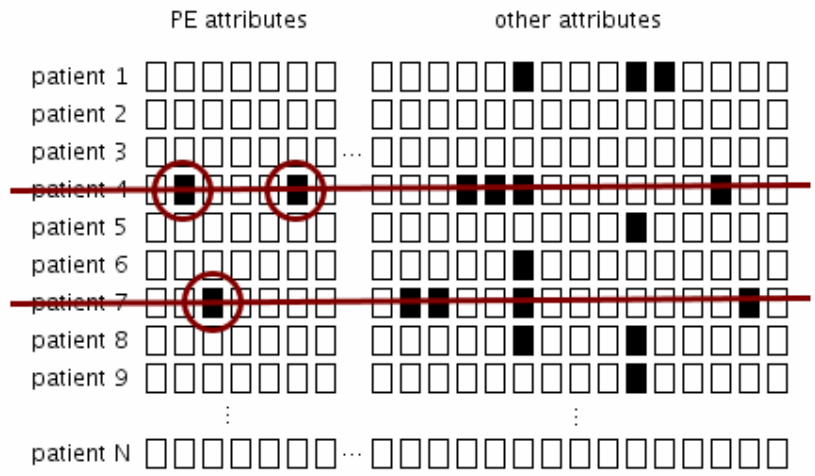

Figure 6-18. Exclusion of patients with missing values among physical examination (PE) attributes.

We handle missing values in other attributes (other than physical examination attributes) by categorizing them. Figure 6-19 gives us a distribution of attribute EF (ejection fraction) for two classes of patients, those that were alive at the end

of the study and those that were not. Patients with no value for EF attribute were excluded from this graph.

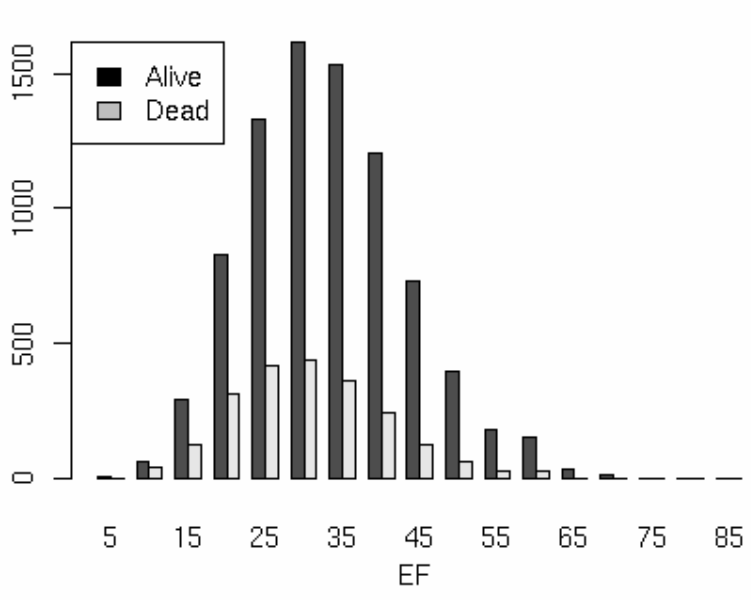

Figure 6-19. Distribution of the attribute EF.

Figure 6-20 shows the situation for categorized attribute EF. We have four categories: low, medium, high and undefined. These categories cannot be ordered and we lost a part of information, but all patients can now be included in a study.

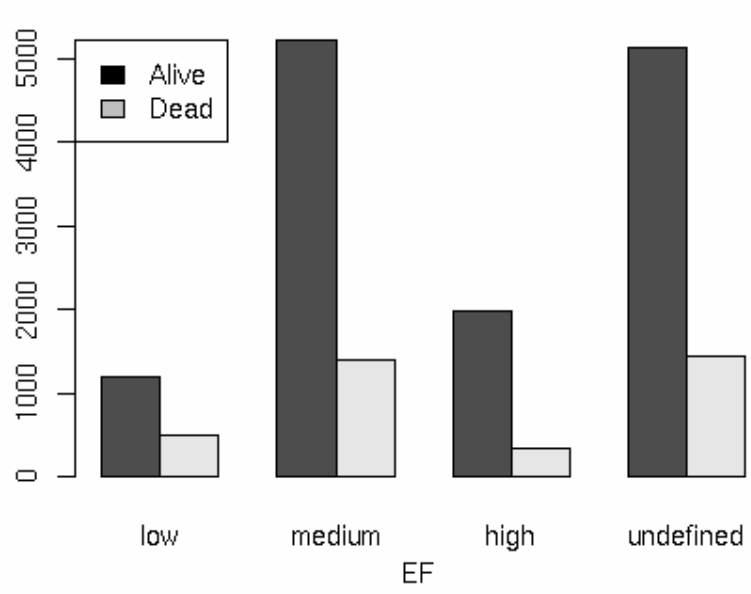

Figure 6-20. Distribution of the categorized attribute EF. Missing values are represented within undefined category.

Figure 6-21 shows this new dataset, where all attributes which contained missing values were categorized. Different shades of grey denote different categories. It is obvious that we did not fill missing values (denoted with black squares) we just labelled them differently; they are no longer missing values, as now they are undefined.

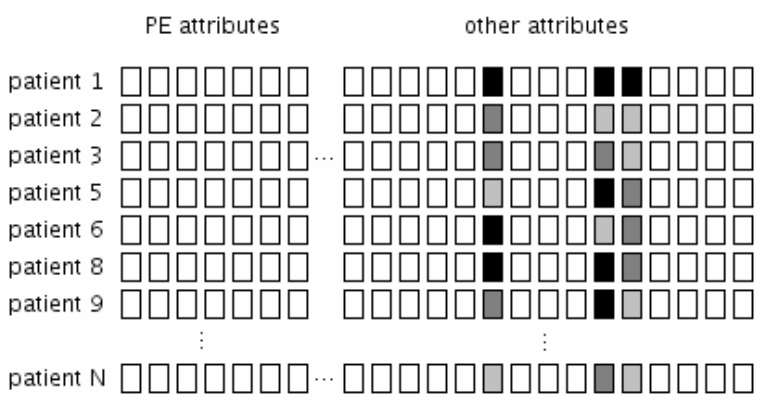

Figure 6-21. Graphical representation of the dataset. Attributes with missing values are categorized. Black rectangles represent missing values which belong to undefined category, shades of grey represent other categories.

Female population survival analyses results

Total number of ANMCO patients in the pre-processed dataset was 2971. Figure 6-22 depicts the results obtained with SRF (error rate of the forest and variable importances), while Figure 6-23 shows the results obtained using CP tree algorithm. Closer analysis reveals that the SRF has ranked as most important variables those which are used in building the CP decision tree, albeit not exactly in the same order

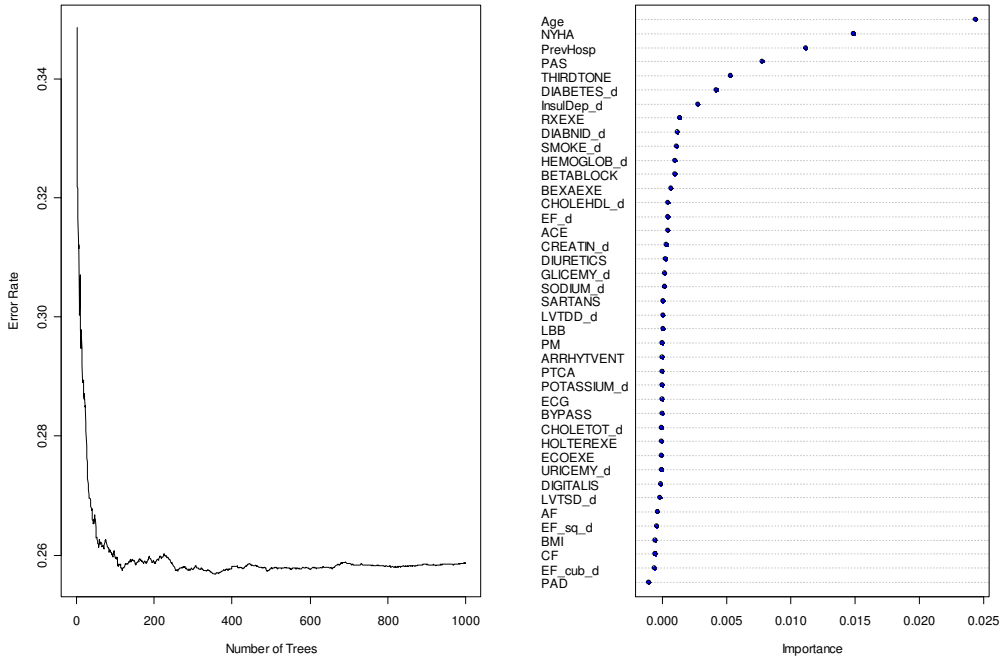

Figure 6-22. Results obtained using SRF for the ANMCO dataset female population over 65 years of age, for the death_cv as target event. As expected Age and NYHA classification are the most important variables for the prognosis of the death by cardiovascular reasons, followed by PrevHosp, PAS, THIRDTONE and diabetes related variables DIABETES and InsuDep.

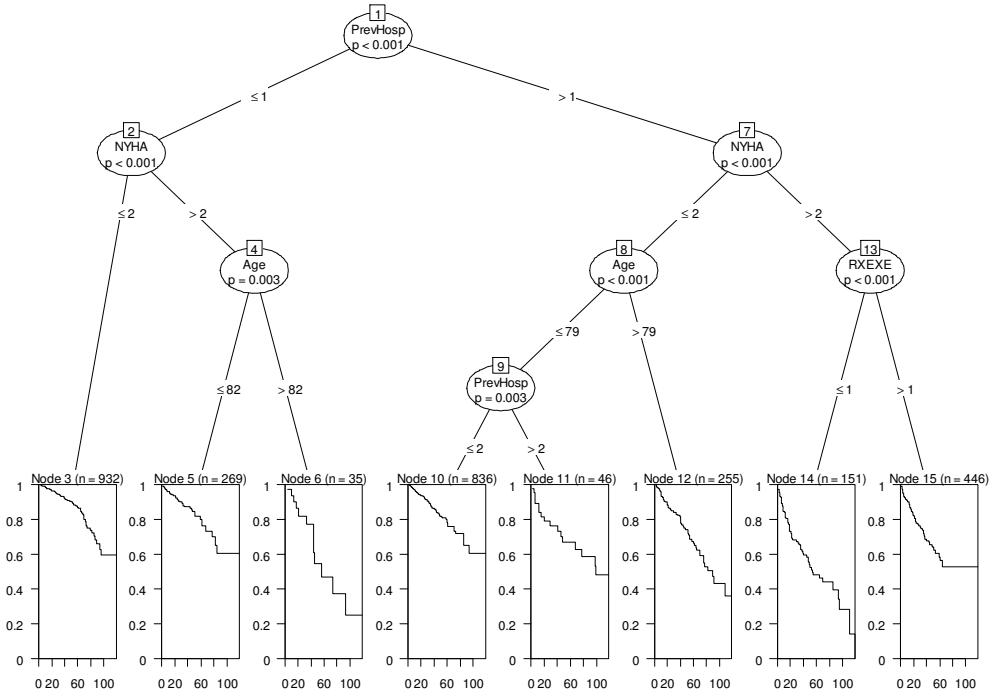

Figure 6-23. The CP (survival analysis) decision tree model for the death event for the female patients over 65 years old. NYHA classification together with the previous hospitalization record and age, seem to be most significant variables for the splitting into distinct survival risk groups (fixed parameters for splitting: p<0.01, maximum depth=4).

Male population survival analyses results

Total number of ANMCO patients in the pre-processed dataset was 6851. Figure 6-24 depicts the results obtained with SRF, while Figure 6-25 shows the results obtained using CP tree algorithm. In this case there is discrepancy in findings of the two algorithms, since contrary to SRF model, *BMI* and *THIRDTONE* seem to be also significant in CP tree model.

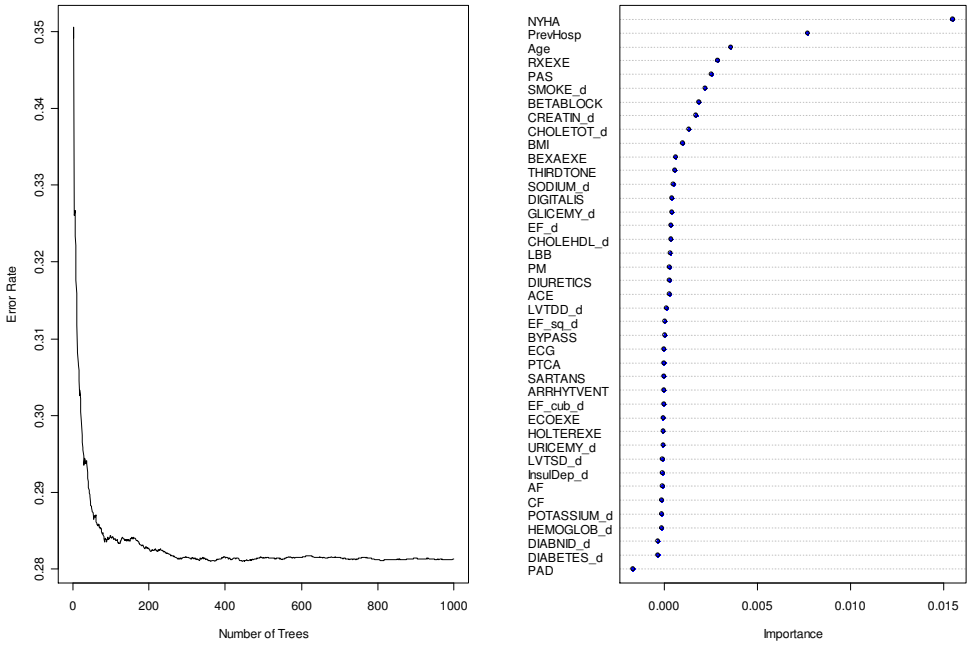

Figure 6-24. Results obtained using SRF for the ANMCO dataset male population over 65 years of age, for the death_cv as target event. Similarly to female population, NYHA classification, PrevHosp, PAS and Age are the most important variables for the prognosis of the death by cardiovascular reasons. The rest of the variables are less important.

Summary of findings by survival analyses

From the modelling results obtained using SRF and CP tree algorithms it is obvious that for different (but related) events, such as death from cardio-vascular reasons, worsening of HF and hospitalization, NYHA classification and previous hospitalization record are most significant for the future prognosis. However, different (physiological) variables are detected as important for finer structuring of the risk groups, for each event. Table 6-21 gives a summary of findings for different models and target events. The discrepancy in findings on significant variables could be explained by the nature of employed algorithms: while SRF orders the variables by the merit of their independent contribution to the particular discrimination problem, CPT algorithm is a greedy approach in which redundancy between variables is not explicitly revealed.

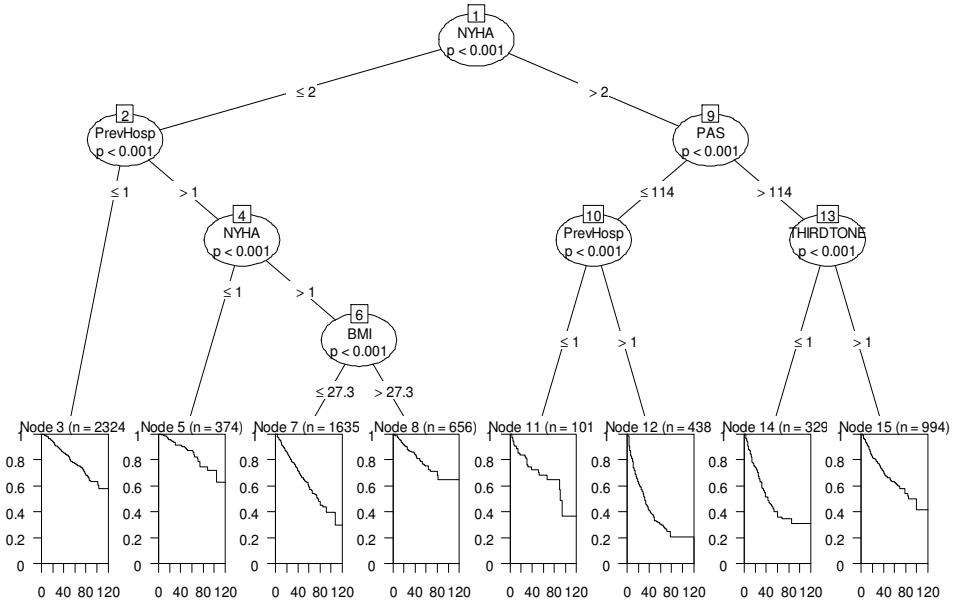

Figure 6-25. The CP (survival analysis) decision tree model for the death_from_cv event for the male patients over 65 years old (ANMCO dataset). NYHA classification and the previous hospitalization record and PAS, are the most important variables. Contrary to SRF model, BMI and THIRDTONE seem to be highly significant. (Fixed node splitting criteria for forming distinct risk probability groups: p<0.001, maximum depth=4).

Table 6-21. Most important variables across different population-target-event models resulting from survival analysis using SRF and CPT algorithms.

	Target event for the survival analysis		
Population model	**Death from CV**	**Worsening**	**Hospitalization**
ALL MODELS	Age, NYHA, PrevHosp	NYHA, PrevHosp	NYHA, PrevHosp
Female SRF+	PAS, THIRDTONE, DIABETES	PAS, DIURETICS, CHOLETOT	PAS, DIABETES, CF
Female CPT+	RXEXE	PAS, DIURETICS, GLICEMY, BYPASS	ARRHYTEVENT, InsuDep, HEMOGLOB, DIURETICS, PAD
Male SRF+	RXEXE, PAS, SMOKE	PAS, DIURETICS, CHOLETOT,	PAS, DIABETES, DIABND
Male CPT+	PAS, THIRDTONE, BMI	PAS, DIURETICS, HEMOGLOB, BYPASS	PAS, DIURETICS, InsuDep

6.2.6. Diagnostic models based on transcriptional data

In the field of molecular biology, a *transcription factor* (sometimes called a sequence-specific DNA binding factor) is a protein that binds to specific parts of DNA using DNA binding domains and is part of the system that controls the transfer (or transcription) of genetic information from DNA to RNA.

Loosely speaking, transcriptional factors have a principal role in the production of proteins within the cellules. Thus transcriptional factors have a great impact in determination of the internal operation of the cellules themselves. In particular, several diseases have their principal cause in malfunctioning of transcription processes, e.g. certain types of cancer or diabetes.

Given the afore discussed reasons, it make sense to study the transcription factors operations within failing and not failing cardiac cellules; in fact, from this kind of studies, it would be possible to identify which transcriptional factors are involved in the pathogenesis of HF.

Even if scientifically relevant, especially for the development of new therapies operating at a cellular level, the determination of the transcriptional factors involved in the HF seems to have a poor relevance in respect to the diagnostic evaluation of HF patient. In fact, if a diagnostic model able to assess HF condition based on transcriptional factors analysis existed, a cardiac biopsy would be necessary in order to extract the myocardium tissue. Biopsy is probably one of the most invasive medical tests, and it is performed only under strict conditions. On the other hand, nowadays several non-invasive or minimally invasive tests exist in order to assess clearly the presence of HF. However, it is not possible to exclude a priori the usefulness of a diagnostic model able to recognize HF condition starting from transcriptional factors analysis, e.g. for those cases in which disease assessment cannot be clearly assessed.

In their work Hannenhalli et al. [1] used a complex approach in order to analyze their dataset. In particular, they employed several concepts coming from the knowledge of the specific context (including the information extracted from previous studies performed on animal models).

For our data mining experiments we didn't utilize that "a priori" knowledge, both because not all the needed information were easily available and because using "a priori" knowledge requires an extensive interaction with an expert of the specific sector. Thus, we utilized standard machine learning technique and compared our results with the results reported in [1].

In particular, we utilized a standard support vector machine (SVM) approach, in order to build classifiers based on the available data.

The use of SVM in the field of RNA microarray data has been successful in several previous studies. In particular, SVM has been originally designed in order to effectively detect complex patterns hidden inside the data. This characteristic identifies SVM as the ideal approach for working with the transcriptional genomic dataset.

The structure of our study has been the following:

I. Step: relevant feature selection

II. Step: SVM application and validation via cross-validation.

For the feature selection phase we decided to adopt the following criteria: a) use the entire dataset, b) select the first *k* attributes as ranked by info gain ratio where *k* has been set to 10, 100, and 1000.

For the second phase we used a 10-fold cross-validation, with the following settings for the SVM parameters: a) Gaussian Kernel, b) *C* parameter values: 1, 10, 100, c) Gamma values: 0.01, 0.1.

The results have been particularly positive: for the setting with ten attributes, gamma 0.1 and *C* 100, we have obtained the accuracy of 99%, with a true positive rate related to the less represented class of 87,5%. In practice, during the 10-fold cross-validation only two examples over 212 cases were misclassified. In general, we obtained similar results for every value of the *C* parameter when sufficiently elevated.

However, what is the real value of the obtained results? When the number of features is greater than the number of cases, it is possible that results are not significant, due to statistical reasons.

In order to test the significance of our results, we generated 100 datasets, each one with the same characteristics of the transcriptional genomic dataset (22,000 attributes, 196 positive cases, 16 negative cases). Values within these datasets were randomly generated. For every generated dataset we applied the same procedure as used for the real dataset. The results demonstrated that we obtained results comparable with the results obtained on the transcriptional genomic dataset only for three among 100 repetitions. Thus, we can state that our results are meaningful, in the sense that the relationship we detected wasn't obtained by chance.

However, it must be noticed that the subset of features selected from our approach is utterly distinct from the subset identified in the work [1]. It is very difficult to explain that result: in fact, it would be more meaningful if we found the same subset of informative variables that Hannenhalli et al. found in their work. More deep studies are still necessary in order to understand this point well.

6.3. Bibliography and references

[1] Hannenhalli S., Putt, M.E., Gilmore, J.M., Wang, J. et al. (2006) *Transcriptional genomics associates FOX transcription factors with human heart failure.* Circulation 2006 Sep 19, 114(12): pp.1269–1276, http://circ.ahajournals.org/cgi/content/abstract/114/12/1269).

[2] Cox, D. R. (1972) *Regression models and life tables (with discussion). Journal of the Royal Statistical Society, Series B,* 34: pp.187–220

[3] Tarone, R.E. (1975) *Tests for trend in life table analysis*. Biometrika, 62: pp.679–682

[4] Tarone, R.E., Ware, J. (1977) *On distribution free tests for equality of survival distributions*. Biometrika, 64: pp.156–160

[5] Statnikov, A., Aliferis, C.F. Tsamardinos, I., Hardin, D., Levy, S. (2005) *A Comprehensive Evaluation of Multicategory Classification Methods for Microarray Gene Expression Cancer Diagnosis*, in Bioinformatics 21(5): pp.631–643

[6] Aliferis, C.F., Tsamardinos, I., Statnikov, A. (2003) *HITON, A Novel Markov Blanket Algorithm for Optimal Variable Selection*. In the American Medical Informatics Association meeting 2003 (AMIA 2003)

[7] Aliferis, C.F., Statnikov, A., Tsamardinos, I., Mani, S., Koutsoukos, X. (2007) *Local Causal and Markov Blanket Induction Algorithms for Causal Discovery and Feature Selection for Classification*. Technical Report DSL TR-07-02

[8] The Task Force for the diagnosis and treatment of CHF of the European Society of Cardiology (2005) *Guidelines for the diagnosis and treatment of Chronic Heart Failure*. European Heart Journal

[9] Raphael et al. (2007) Limitations of the New York Heart Association functional classification system and self-reported walking distances in chronic heart failure. Heart 2007, 93: pp.476–482

7. Conclusions

The deliverable presents the results of the task T4.3. It has been prepared by UNICAL, FORTH, and RBI. National Center for Biotechnology Information Task T4.3 continues almost till the end of the project and in this period the goals will be thorough comparison of different KD methods, construction of more reliable models that can be integrated into the platform's decision support system, and knowledge discovery based on the data collected by the platform.

Current work demonstrated known problems related to the KD methodology that have already been experienced in other medical domains. The basic problem is datasets. They should be large both in respect to the number of included patients and the number of variables that describe them. Additionally, there should be variables that really describe target concepts and especially those that are not their consequences in any respect (e.g. prescribed medication is the consequence of the patient's HF severity status and some laboratory tests can have different results due to the prescribed medications). Finally, the data need to be consistently collected, what is especially difficult to achieve in cases when many persons or medical centres physically collect the data and when the type of the data can reflect medical experience of local personnel. We have devoted care to the data pre-processing methods like missing data handling, noise detection, dataset balancing, but the experiments clearly demonstrate that data pre-processing is unable to solve all problems inherited from the data collection. The lesson learned, and confirmed throughout the analyses presented in this deliverable is that only high quality data can ensure reliability and significance of induced results, regardless which KD tools and pre-processing methods are used.

Significant effort has been devoted to the selection and preparation of tools appropriate for the implementation into the KD Web service. Besides software licenses and automatic visualisation of the results, data overfitting has been identified as the main problem for such applications. It is common that inexperienced users trying to obtain ideal results from KD tools actually forget about the danger of overfitting. A possible solution is to strongly restrict the number of parameters that can be controlled by users. The results with many different KD tools in the HF domain demonstrated that besides experience, strict use of cross-validation experiments with different parameter values on the same domain is necessary. Practically, only random forest approach is the exception in this respect: the chance for overfitting always exists but it decreases with the number of generated decision trees. That is the main reason why RF has been selected as the first and main algorithm for the Web-based KD service. The results representing variable importances for the dataset collected by the platform have been prepared off-line by using the RF tool; however, they demonstrate the type of the results that can be expected by the on-line KD service.

From the point of knowledge relevant for inclusion into the knowledge base and into the decision support system, analysis of the decompensation dataset has been

the most relevant result. Similar results do not exist in medical literature. The problem is that the currently available dataset is relatively small and consequently, the reliability of induced models is rather low. One of the future goals is to collect significantly larger decompensation dataset. Actually, one of the goals of the platform is to enable automatic collection of large and complex datasets about such and similar events. The experiments demonstrate that we already have the methodology for their analysis, and that obtained results should also be important outside of the Heartfaid project.

A1 Random forest: Algorithm pseudocode

procedure GrowRandomForest
Input: set of *Examples*, set of *Attributes*,
Parameters: *k*
Procedure:
$m = \sqrt{Attributes.count}$
for *i=1* to *k* do{
    Generate a bootstrap examples vector θ_i
    $oob_i = Examples - \theta_i$
    $h_i(x) = GrowATree(\theta_i, m)$
}
for *i=1* to *k* do{
    Calculate *misclassification_rate* between oob_i and $h_i(oob_i)$
}
error_rate = misclassification_rate / total_oob.count
Output: A random forest

Figure A1-1. The main random forest learning procedure

procedure GrowATree
Input: set of examples θ
Parameters: *m*
Procedure:
Attribute_subset = Random(*Attributes*, *m*)
for each attribute *A* in *Attribute_subset*
    find Gini index for each *A*
end for
Best = attribute *A* with minimal Gini index
node.attribute = A
do the split on *node*
if *node.terminal* then
    node = most frequent class
else
    θ_{yes} = examples from θ where split condition is positive
    θ_{no} = examples from θ where split condition is negative
    $node.right = GrowATree(\theta_{yes}, m)$
    $node.leftt = GrowATree(\theta_{no}, m)$
end if
return *node*
Output: node of a random tree

Figure A1-2. Procedure for growing a single random tree

```
procedure Predict
Input:   forest Forest and examples Examples
Procedure:
for each Example in Examples
        if classification
                for i=1 to Forest.tree_count do
                        ++votes[Forest(Example)]
                end for
                result = max(votes)
        else if regression
                for i=1 to Forest.tree_count do
                        result += Forest(Example)
                end for
                result /= Forest.tree_count
        end if
end for each
Output: Prediction result
```

Figure A1-3. Procedure for RF based prediction – using RF for classification/regression

A2 Results on ANMCO dataset related to the prognostic models obtained through the classification learning approach

Table A2-1. The best-performing models that employ variable selection (HITON-PC) algorithm, along with the selected variables. The variables with names HCOD=X are indicator variables taking values 1 if a patient's data were recorded in the hospital with hospital code X. Each variable that appears with a name of the form VARNAME_M is a missing-value indicator variable for the variable with name VARNAME. Only 10 best variables are presented.

	T	AUC	Parameters	Variables
H	3	64,88	Poly, HITON-PC, C=0,001, d=1	NYHA HOSP_PREV_YEAR HCOD=H60001 PAS CREATIN EF CF HEMOGLOB SODIUM CHOLEHDL
	6	62,77	Gaussian, HITON-PC C=1000, γ=0,01	NYHA HOSP_PREV_YEAR CREATIN HCOD=H60001 PAS EF HEMOGLOB URICEMY CF SODIUM
	9	68,52	Poly, HITON-PC, C=0,001, d=1	NYHA HOSP_PREV_YEAR HCOD=H60001 CREATIN EF PAS HEMOGLOB URICEMY SODIUM BETABLOCK
	12	70,92	Gaussian, HITON-PC C=1000, γ=0,01	NYHA HOSP_PREV_YEAR HCOD=H60001 EF CREATIN PAS HEMOGLOB URICEMY AGE BETABLOCK

G	3	72,75	Poly, HITON-PC, C=0,001, d=1	NYHA HOSP_PREV_YEAR THIRDTONE CF PAS SODIUM_M URICEMY CHOLETOT CUBIC_FORMULA_EF_M HCOD=H60001
	6	73,69	Poly, HITON-PC, C=10, d=1	NYHA HOSP_PREV_YEAR EF CF PAS HCOD=H30130 CREATIN HCOD=H30065 SODIUM HCOD=H100001
	9	71,88	Gaussian, HITON-PC C=1000, γ=0,01	NYHA HOSP_PREV_YEAR DIURETICS EF CF HCOD=H30130 PAS CREATIN HCOD=H30065 HCOD=H30046
	12	70,47	Gaussian, HITON-PC C=10, γ=0,01	NYHA HOSP_PREV_YEAR EF DIURETICS CF CREATIN PAS HCOD=H30046 HCOD=H30130 HCOD=H30065

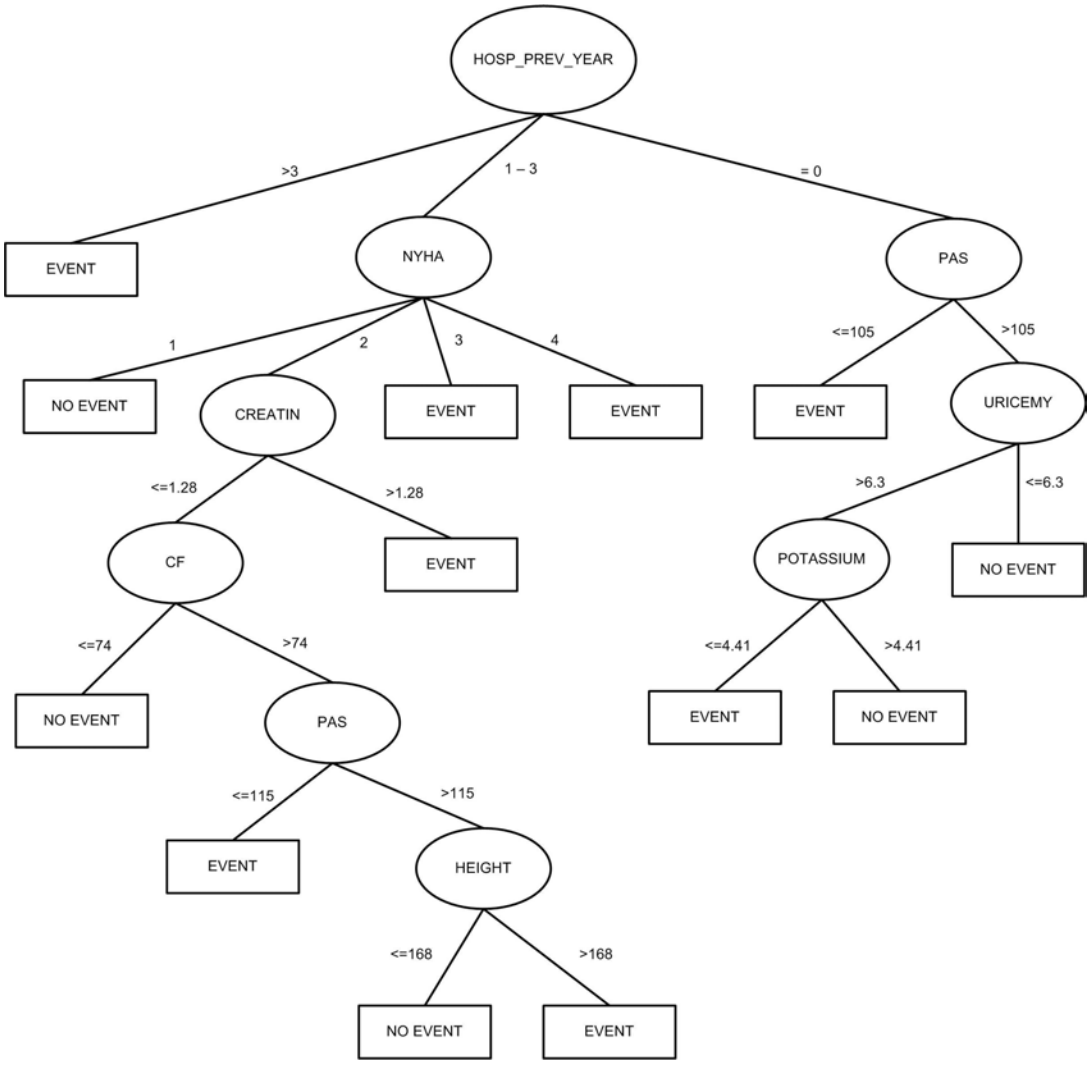

Figure A2-1. "H" outcome definition, 3 months threshold. The best performing decision tree for the outcome "H" (the patient is rehospitalized after the first visit for any cause or died) with a follow up limit of three months. The decision tree has been created by the J48 algorithm on ANMCO data following the "classification approach" (see Section 6.2.4).

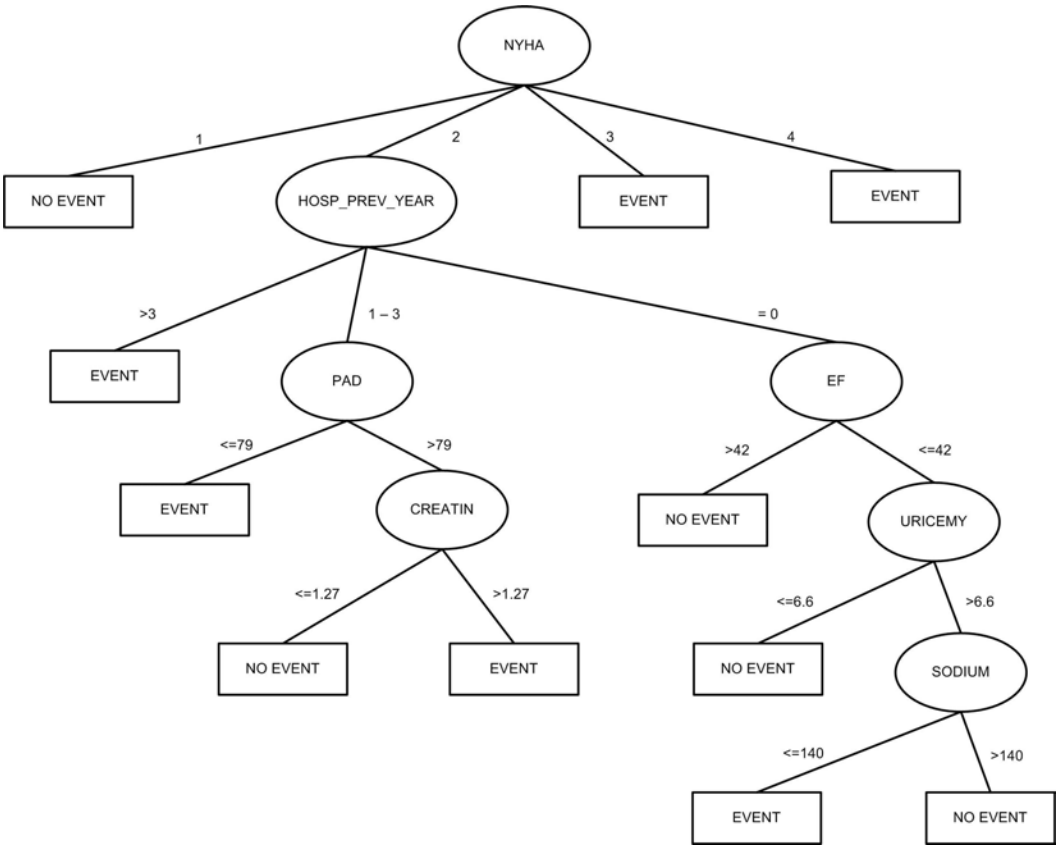

Figure A2-2. The best performing decision tree for the outcome "H" (the patient is rehospitalized after the first visit for any cause or died) with a follow up limit of six months.

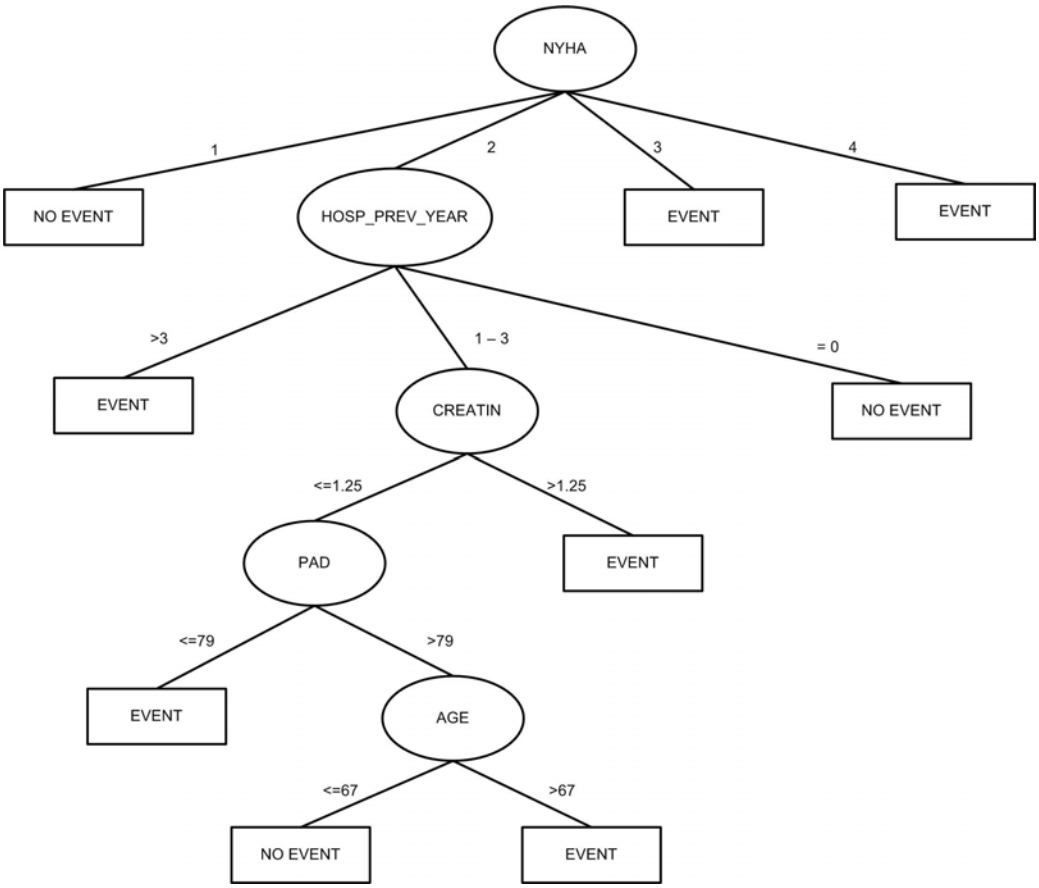

Figure A2-3. The best performing decision tree for the outcome "H" (the patient is rehospitalized after the first visit for any cause or died) with a follow up limit of nine months.

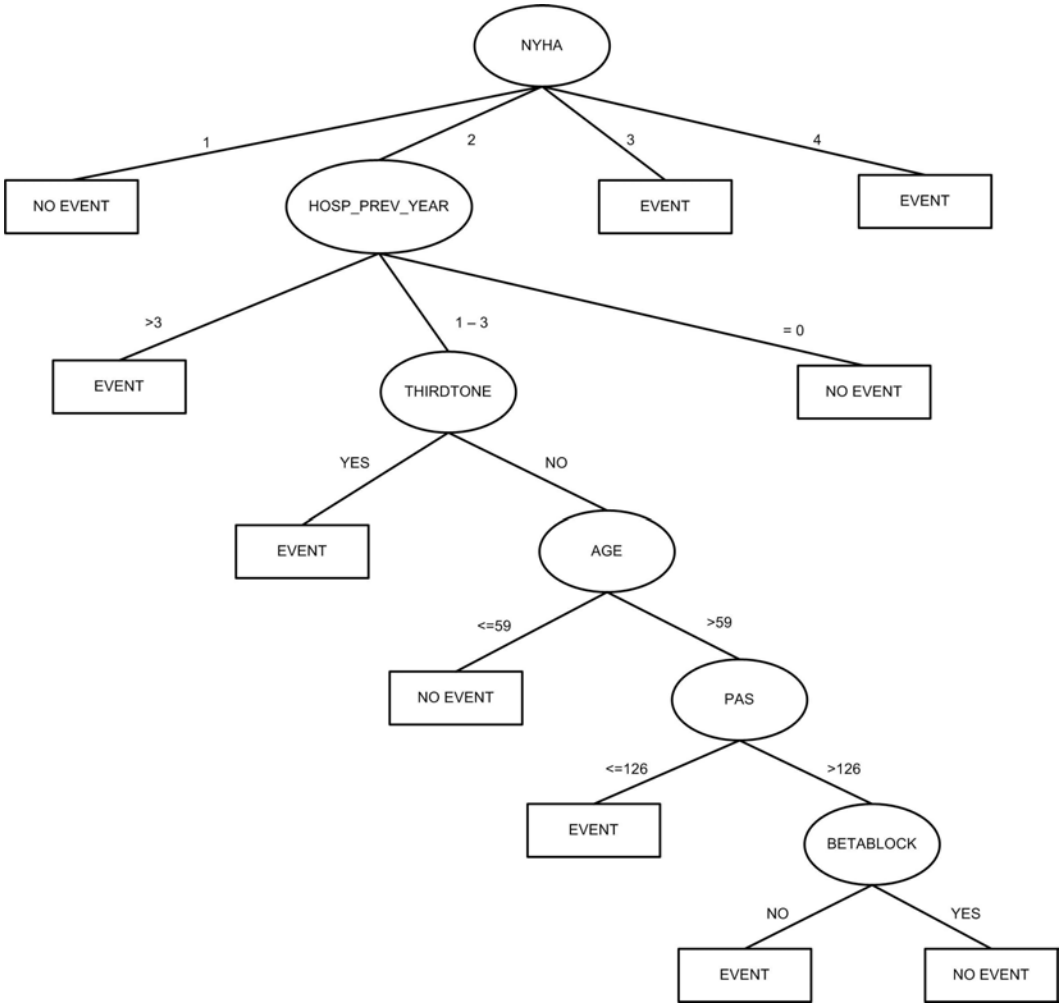

Figure A2-4. The best performing decision tree for the outcome "H" (the patient is rehospitalized after the first visit for any cause or died) with a follow up limit of twelve months.

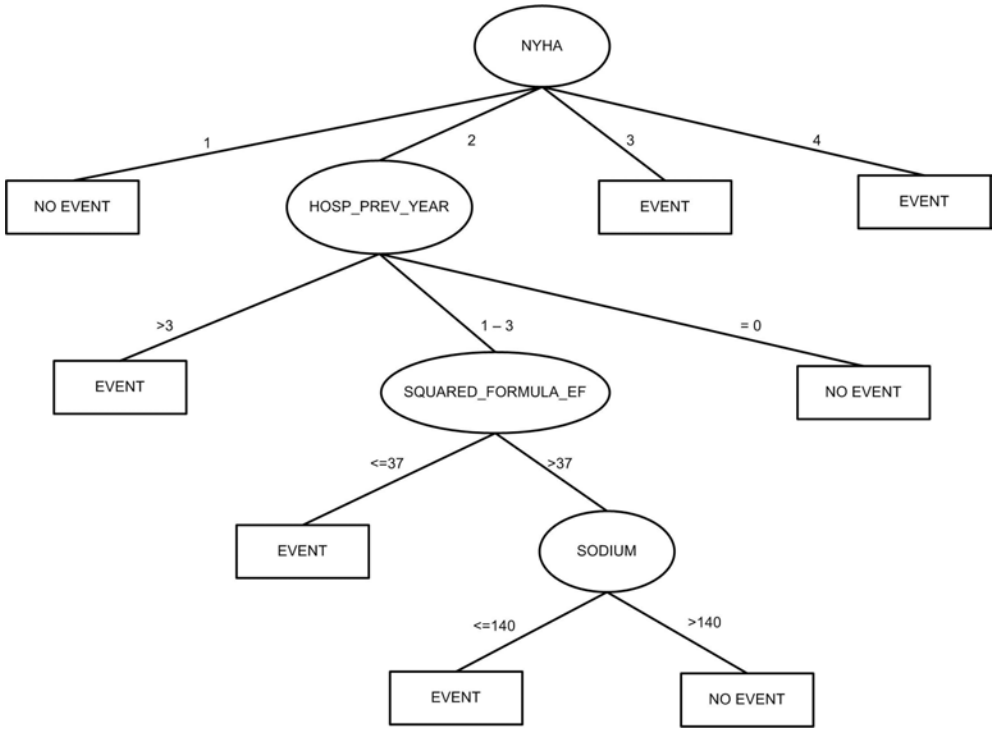

Figure A2-5. The best performing decision tree for the outcome "G" (the patient experiences an adverse event) with a follow up limit of three months.

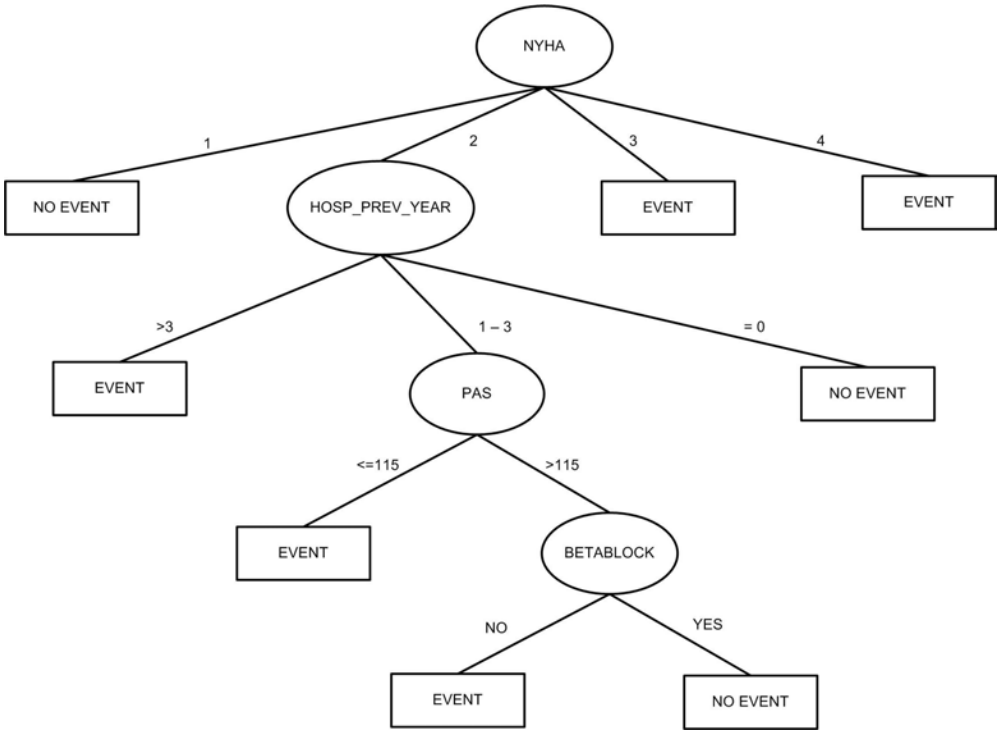

Figure A2-6. The best performing decision tree for the outcome "G" (the patient experiences an adverse event) with a follow up limit of six months.

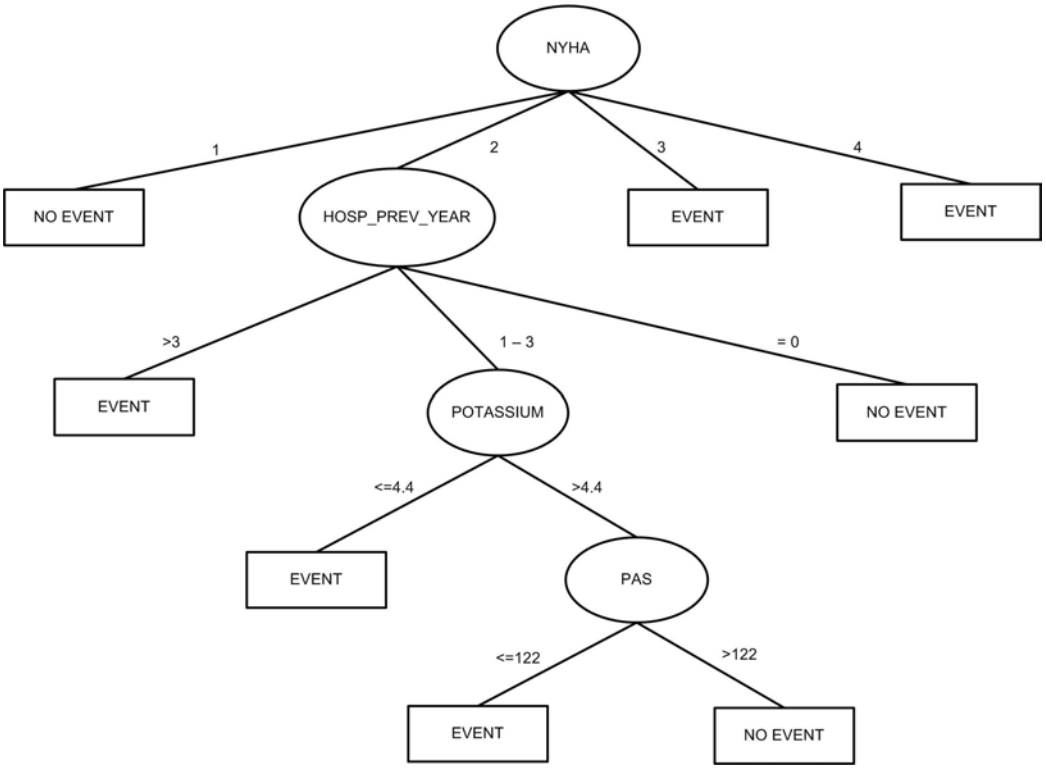

Figure A2-7. The best performing decision tree for the outcome "G" (the patient experiences an adverse event) with a follow up limit of nine months.

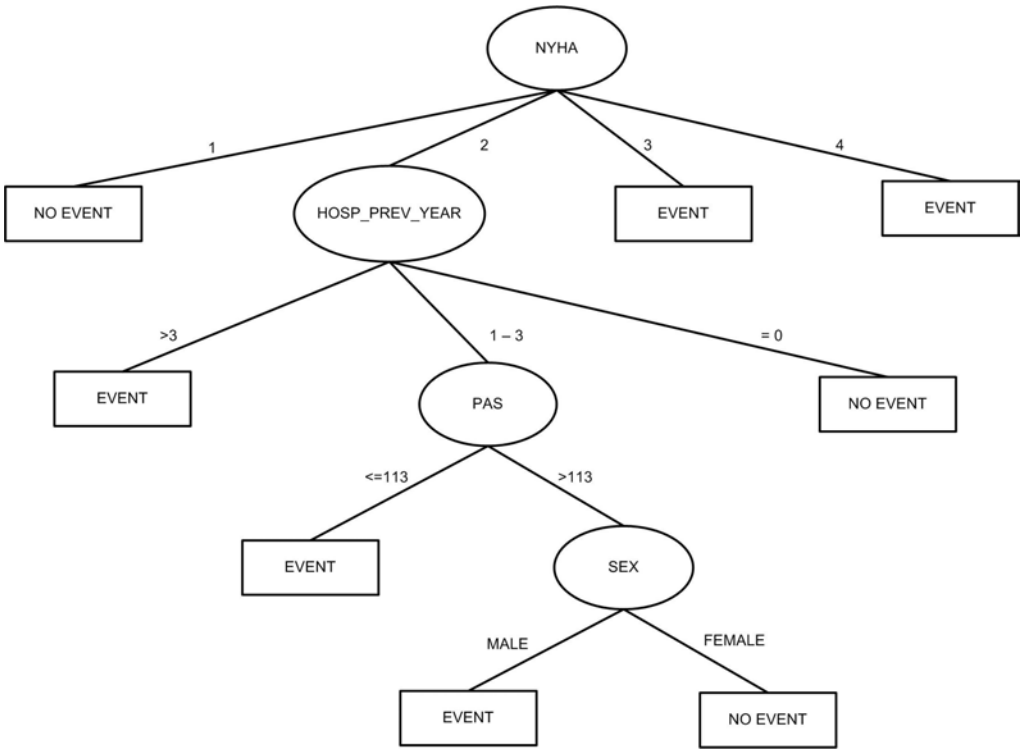

Figure A2-8. The best performing decision tree for the outcome "G" (the patient experiences an adverse event) with a follow up limit of twelve months.

Table A2-2. A pairwise comparison between variables selected by decision tree algorithm J48 and the best-performing SVM model using the HITON-PC algorithm for variable selection. Presented are only 10 the most significant variables. The classification task is to predict whether event "H: the patient is rehospitalized after the first visit for any cause or died" will occur before the time threshold of three months. The variables selected by HITON-PC are ranked according to their pairwise association with the class variable. Variables selected by both methods are shown in bold-face. AUC is performance on the test set for the best model trained with the selected variables.

Variables (Decision Tree)	**Variables (HITON PC + SVM)**
HOSP_PREV_YEAR **NYHA** **PAS** **CREATIN** URICEMY **CF** POTASSIUM HEIGHT	**NYHA** **HOSP_PREV_YEAR** HCOD=H60001 **PAS** **CREATIN** EF **CF** HEMOGLOB SODIUM CHOLEHDL
AUC: 64.3	*AUC: 64.88*

Table A2-3. Same as Table A2-2 but for the classification task for event "H: the patient is rehospitalized after the first visit for any cause or died" will occur before the time threshold of six months.

Variables (Decision Tree)	Variables (HITON PC + SVM)
NYHA **HOSP_PREV_YEAR** PAD **EF** **URICEMY** **CREATIN** **SODIUM**	**NYHA** **HOSP_PREV_YEAR** **CREATIN** HCOD=H60001 PAS **EF** HEMOGLOB **URICEMY** CF **SODIUM**
AUC: 65.3	*AUC: 62.77*

Table A2-4. Same as Table A2-2 but for the classification task for event "H: the patient is rehospitalized after the first visit for any cause or died" will occur before the time threshold of nine months.

Variables (Decision Tree)	Variables (HITON PC + SVM)
NYHA **HOSP_PREV_YEAR** **CREATIN** PAD AGE	**NYHA** **HOSP_PREV_YEAR** HCOD=H60001 **CREATIN** EF PAS HEMOGLOB URICEMY SODIUM BETABLOCK
AUC: 66.5	*AUC: 68.52*

Table A2-5. Same as Table A2-2 but for the classification task for event "H: the patient is rehospitalized after the first visit for any cause or died" will occur before the time threshold of twelve months.

Variables (Decision Tree)	Variables (HITON PC + SVM)
NYHA **HOSP_PREV_YEAR** THIRDTONE **AGE** **PAS** **BETABLOCK**	**NYHA** **HOSP_PREV_YEAR** HCOD=H60001 EF CREATIN **PAS** HEMOGLOB URICEMY **AGE** **BETABLOCK**
AUC: 66.4	*AUC: 73.55*

Table A2-6. Same as Table A2-2 but for the classification task for event "G: the patient experiences any adverse event after the first visit" will occur before the time threshold of three months.

Variables (Decision Tree)	Variables (HITON PC + SVM)
NYHA **HOSP_PREV_YEAR** SQUARED_FORMULA_EF SODIUM	**NYHA** **HOSP_PREV_YEAR** THIRDTONE CF PAS SODIUM_M URICEMY CHOLETOT CUBIC_FORMULA_EF_M HCOD=H60001
AUC: 70.8	*AUC: 72.75*

Table A2-7. Same as Table A2-2 but for the classification task for event "G: the patient experiences any adverse event after the first visit" will occur before the time threshold of six months.

Variables (Decision Tree)	Variables (HITON PC + SVM)
NYHA **HOSP_PREV_YEAR** **PAS** BETABLOCK	**NYHA** **HOSP_PREV_YEAR** EF CF **PAS** HCOD=H30130 CREATIN HCOD=H30065 SODIUM HCOD=H100001
AUC:71.00	*AUC: 74.81*

Table A2-8. Same as Table A2-2 but for the classification task for event "G: the patient experiences any adverse event after the first visit" will occur before the time threshold of nine months.

Variables (Decision Tree)	Variables (HITON PC + SVM)
NYHA **HOSP_PREV_YEAR** POTASSIUM **PAS**	**NYHA** **HOSP_PREV_YEAR** DIURETICS EF CF HCOD=H30130 **PAS** CREATIN HCOD=H30065 HCOD=H30046
AUC: 70.2	*AUC: 74.59*

Table A2-9. Same as Table A2-2 but for the classification task for event "G: the patient experiences any adverse event after the first visit" will occur before the time threshold of twelve months.

Variables (Decision Tree)	**Variables (HITON PC + SVM)**
NYHA **HOSP_PREV_YEAR** **PAS** SEX	**NYHA** **HOSP_PREV_YEAR** EF DIURETICS CF CREATIN **PAS** HCOD=H30046 HCOD=H30130 HCOD=H30065
AUC: 68.6	*AUC: 74.06*

A3 Prognostic models based on survival analysis for HF related worsening and hospitalization

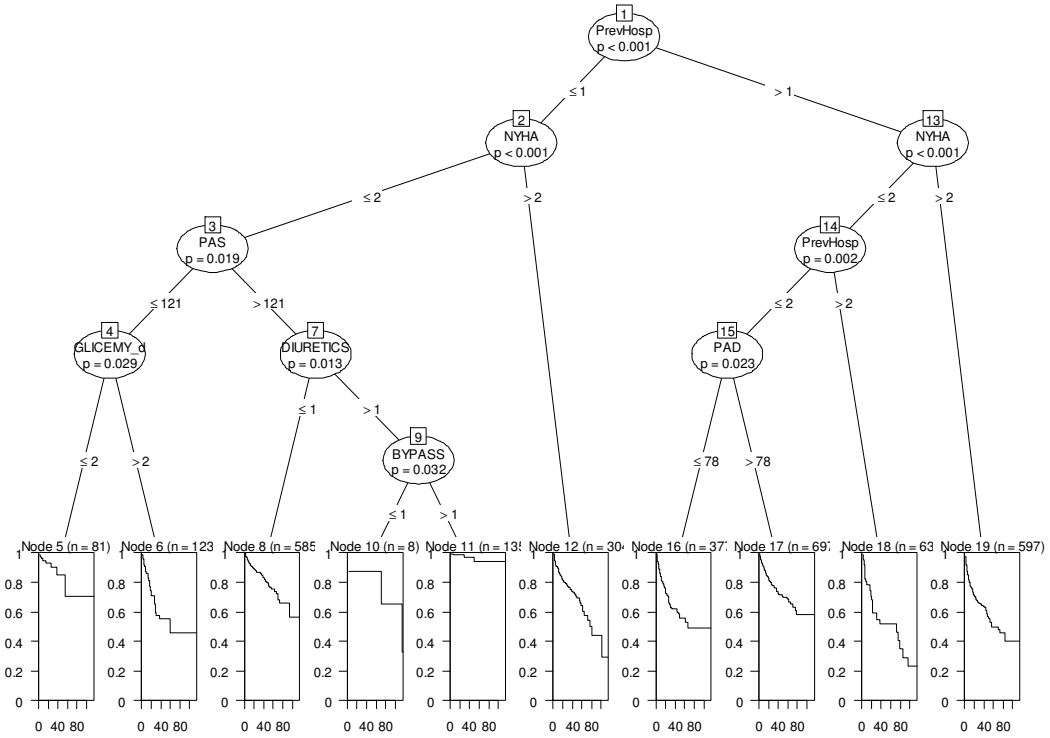

Figure A3-1 The CP (survival analysis) decision tree model for the worsening event for the female patients over 65 years old (ANMCO dataset). In this decision tree besides NYHA classification and the previous hospitalization record, other variables (PAS, PAD, GLICEMY, DIURETICS) are important for the splitting into distinct worsening risk groups (fixed parameters for splitting: p<0.001, maximum depth=4).

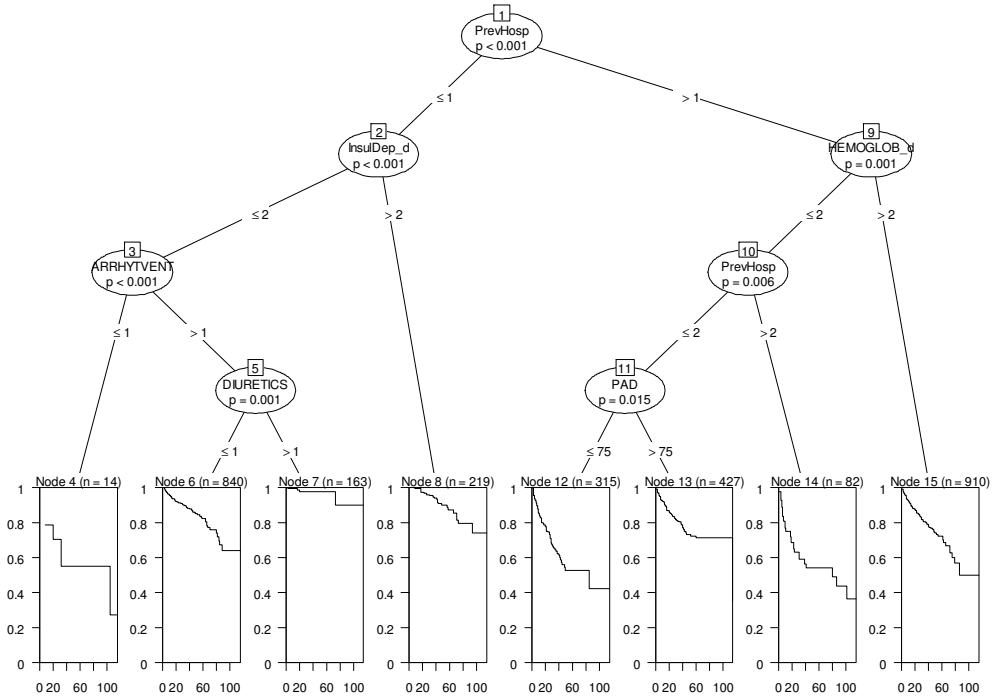

Figure A3-2. The CP (survival analysis) decision tree model for the hospitalization event for the female patients over 65 years old (ANMCO dataset). Similarly to the worsening and death event CP decision trees, most important variables are NYHA and PrevHosp (NYHA classification and the previous hospitalization record). In comparison to worsening event, new splitting parameters are HEMOGLOB and ARRHYTEVENT (fixed parameters for splitting: p<0.001, maximum depth=4).

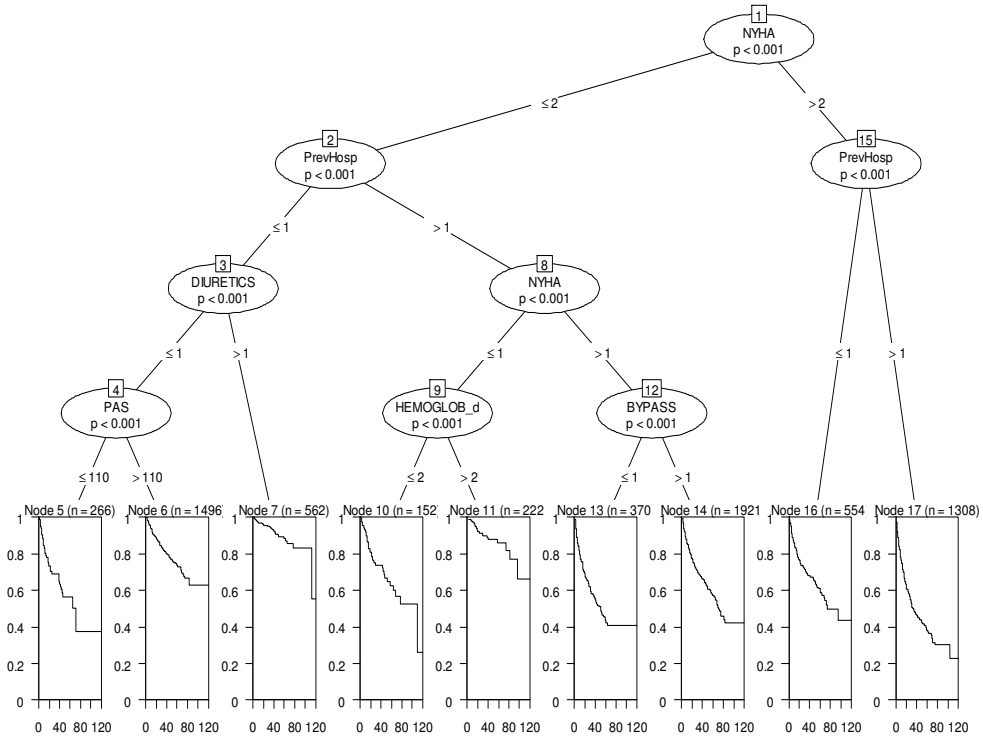

Figure A3-3. The CP (survival analysis) decision tree model for the worsening event for the male patients over 65 years old (ANMCO dataset). In this decision tree besides NYHA classification and the previous hospitalization record, other important variables are (DIURETICS, PAS, HEMOGLOB, BYPASS) are important for the splitting into distinct risk probability groups (fixed parameters for splitting: p<0.001, maximum depth=4)

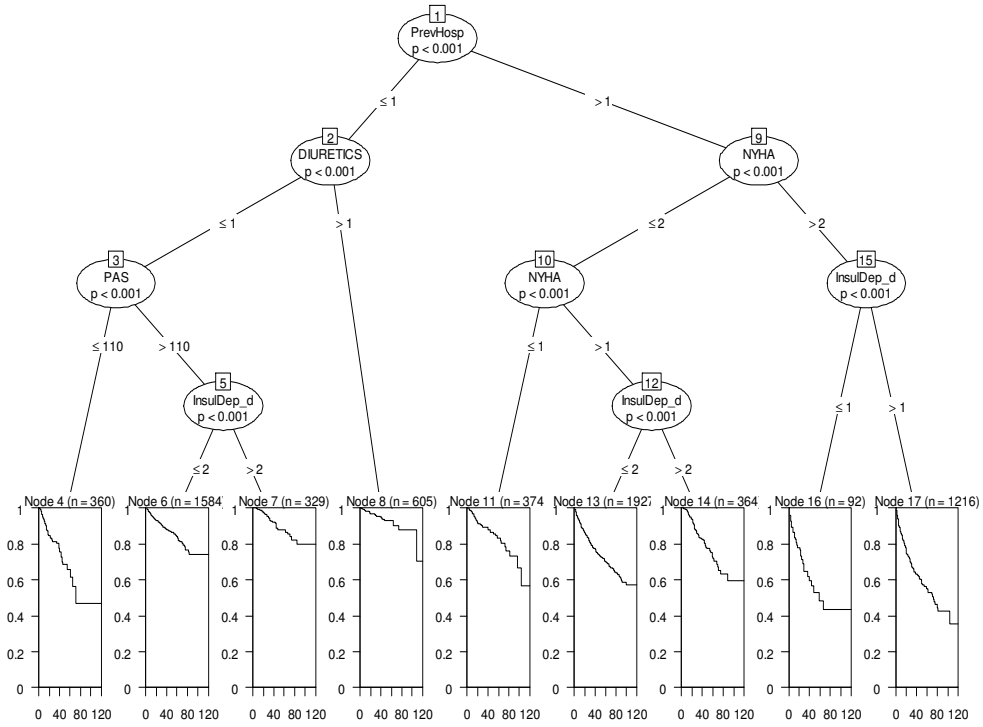

Figure A3-4. The CP (survival analysis) decision tree model for the hospitalization event for the male patients over 65 years old (ANMCO dataset). In this decision tree besides NYHA classification and the previous hospitalization record, DIURETICS, PAS and Insulin Dependence are important for the splitting into distinct risk probability groups (fixed parameters for splitting: p<0.001, maximum depth=4)