Licentiate Thesis

Predictive Techniques and Methods for Decision Support in Situations with Poor Data Quality

Rikard König

Technology

University of Borås
School of Business and Informatics

University of Skövde
Informatics Research Center

University of Örebro
School of Science and Technology
Abstract

Today, decision support systems based on predictive modeling are becoming more common, since organizations often collect more data than decision makers can handle manually. Predictive models are used to find potentially valuable patterns in the data, or to predict the outcome of some event. There are numerous predictive techniques, ranging from simple techniques such as linear regression, to complex powerful ones like artificial neural networks. Complex models usually obtain better predictive performance, but are opaque and thus cannot be used to explain predictions or discovered patterns. The design choice of which predictive technique to use becomes even harder since no technique outperforms all others over a large set of problems. It is even difficult to find the best parameter values for a specific technique, since these settings also are problem dependent. One way to simplify this vital decision is to combine several models, possibly created with different settings and techniques, into an ensemble. Ensembles are known to be more robust and powerful than individual models, and ensemble diversity can be used to estimate the uncertainty associated with each prediction.

In real-world data mining projects, data is often imprecise, contain uncertainties or is missing important values, making it impossible to create models with sufficient performance for fully automated systems. In these cases, predictions need to be manually analyzed and adjusted. Here, opaque models like ensembles have a disadvantage, since the analysis requires understandable models. To overcome this deficiency of opaque models, researchers have developed rule extraction techniques that try to extract comprehensible rules from opaque models, while retaining sufficient accuracy.

This thesis suggests a straightforward but comprehensive method for predictive modeling in situations with poor data quality. First, ensembles are used for the actual modeling, since they are powerful, robust and require few design choices. Next, ensemble uncertainty estimations pinpoint predictions that need special attention from a decision maker. Finally, rule extraction is performed to support the analysis of uncertain predictions. Using this method, ensembles can be used for predictive modeling, in spite of their opacity and sometimes insufficient global performance, while the involvement of a decision maker is minimized.

The main contributions of this thesis are three novel techniques that enhance the performance of the purposed method. The first technique deals with ensemble uncertainty estimation and is based on a successful approach often used in weather forecasting. The other two are improvements of a rule extraction technique, resulting in increased comprehensibility and more accurate uncertainty estimations.
# Contents

1 Introduction .................................................................................................................................. 1

1.1 Key Observations .................................................................................................................. 4

1.2 Problem Statement .................................................................................................................. 4

1.3 Thesis Objectives .................................................................................................................... 4

1.4 Main Contributions .................................................................................................................. 4

1.5 Thesis outline .......................................................................................................................... 5

2 Information Fusion .................................................................................................................. 7

2.1 OODA-Loop ............................................................................................................................ 7

2.1.1 The OODA-loop in Relation to Predictive Modeling ...................................................... 8

3 Knowledge Discovery in Databases ......................................................................................... 11

3.1 Data Mining ............................................................................................................................ 12

3.1.1 Data Mining Tasks ............................................................................................................. 12

3.1.2 Machine Learning vs. Data Mining ................................................................................... 14

3.1.3 Statistics versus Data Mining .......................................................................................... 14

3.2 Predictive Modeling ................................................................................................................. 14

3.3 Data ......................................................................................................................................... 15

3.3.1 Missing Values .................................................................................................................. 16

3.3.2 Outliers ............................................................................................................................. 17

3.4 Preprocessing .......................................................................................................................... 18

3.4.1 Normalization .................................................................................................................... 18

4 Evaluation of Predictive Models ............................................................................................... 19

4.1 Holdout Sample ....................................................................................................................... 19

4.2 Random Subsampling ............................................................................................................. 19

4.3 Cross Validation ....................................................................................................................... 19

4.4 Comparing Models .................................................................................................................. 20

4.4.1 Wilcoxon Signed Rank Rest ............................................................................................ 20

4.4.2 Friedman Test .................................................................................................................... 21

4.4.3 Nemenyi Test .................................................................................................................... 22

4.5 Error Metrics for Classification Tasks .................................................................................... 23
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5.1</td>
<td>Accuracy (ACC)</td>
<td>23</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Area Under ROC-curve (AUC)</td>
<td>23</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Brier Score (BRI)</td>
<td>24</td>
</tr>
<tr>
<td>4.6</td>
<td>Error Metrics for Estimation Tasks</td>
<td>25</td>
</tr>
<tr>
<td>4.6.1</td>
<td>Mean Absolute Deviation (MAD)</td>
<td>25</td>
</tr>
<tr>
<td>4.6.2</td>
<td>Coefficient of Determination ($r^2$)</td>
<td>26</td>
</tr>
<tr>
<td>4.6.3</td>
<td>Root Mean Square Error (RMSE)</td>
<td>26</td>
</tr>
<tr>
<td>4.6.4</td>
<td>Geometric Mean of the Relative Absolute Error (GMRAE)</td>
<td>26</td>
</tr>
<tr>
<td>5</td>
<td>Data Mining Techniques</td>
<td>29</td>
</tr>
<tr>
<td>5.1</td>
<td>Linear Regression</td>
<td>29</td>
</tr>
<tr>
<td>5.2</td>
<td>Decision Trees</td>
<td>29</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Creation</td>
<td>30</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Pruning</td>
<td>31</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Probability Estimation</td>
<td>31</td>
</tr>
<tr>
<td>5.3</td>
<td>Artificial Neural Networks</td>
<td>32</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Basics of a Neural Network</td>
<td>32</td>
</tr>
<tr>
<td>5.4</td>
<td>Evolutionary Algorithms</td>
<td>35</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Genetic Algorithms</td>
<td>36</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Genetic Programming</td>
<td>36</td>
</tr>
<tr>
<td>5.5</td>
<td>Ensembles</td>
<td>39</td>
</tr>
<tr>
<td>5.5.1</td>
<td>Calculating Member Weights</td>
<td>39</td>
</tr>
<tr>
<td>5.5.2</td>
<td>Ensemble Creation</td>
<td>40</td>
</tr>
<tr>
<td>5.5.3</td>
<td>Member Selection</td>
<td>40</td>
</tr>
<tr>
<td>5.5.4</td>
<td>Estimating Ensemble Uncertainty</td>
<td>41</td>
</tr>
<tr>
<td>5.6</td>
<td>Rule extraction</td>
<td>42</td>
</tr>
<tr>
<td>5.6.1</td>
<td>Rule Size</td>
<td>44</td>
</tr>
<tr>
<td>6</td>
<td>Previous work - Genetic Rule Extraction (G-REX)</td>
<td>47</td>
</tr>
<tr>
<td>6.1</td>
<td>Fitness Functions</td>
<td>47</td>
</tr>
<tr>
<td>6.2</td>
<td>Representation Languages</td>
<td>48</td>
</tr>
<tr>
<td>6.3</td>
<td>Empirical Results</td>
<td>49</td>
</tr>
</tbody>
</table>
Motivation of Case Studies .............................................................. 51
Data sets Used for Evaluation........................................................ 53
8.1 Classification Data sets ............................................................. 53
8.2 Estimation Data sets ................................................................. 56
Case Studies ..................................................................................... 59
9.1 Increasing Rule Extraction Comprehensibility ....................... 59
9.1.1 Summary ............................................................................. 59
9.1.2 Background ........................................................................ 59
9.1.3 Method ................................................................................. 60
9.1.4 Experiments ........................................................................ 62
9.1.5 Results .................................................................................. 63
9.1.6 Discussion ............................................................................ 66
9.1.7 Conclusions ......................................................................... 67
9.1.8 Considerations ..................................................................... 67
9.2 Using GP to increase Rule Quality ........................................ 69
9.2.1 Summary ............................................................................. 69
9.2.2 Background ......................................................................... 69
9.2.3 Experiments ......................................................................... 72
9.2.4 Results .................................................................................. 73
9.2.5 Conclusions ......................................................................... 75
9.2.6 Considerations ..................................................................... 76
9.3 Instance Ranking using Ensemble Spread .............................. 77
9.3.1 Summary ............................................................................. 77
9.3.2 Background ......................................................................... 77
9.3.3 Method ................................................................................. 78
9.3.4 Experiments ......................................................................... 82
9.3.5 Results .................................................................................. 82
9.3.6 Conclusions ......................................................................... 85
9.4 Inconsistency – Friend or Foe .................................................. 87
9.4.1 Summary ............................................................................. 87
List of Figures

Figure 1 - The role of predictive modeling tasks in the OODA-loop ............... 8
Figure 2 - Phases of CRISP-DM........................................................................ 11
Figure 3 - Possible model of the balloon data set in Table 1............................ 15
Figure 4 - Predictive Modeling.......................................................................... 15
Figure 5 - Gaussian distribution with mean of 0 and STD of 1 ....................... 17
Figure 6 - Sample ROC curve .......................................................................... 24
Figure 7 - Simple Decision Tree ......................................................................... 30
Figure 8 - Correct classifications / instances reaching each node .................... 32
Figure 9 - A simple MLP .................................................................................... 33
Figure 10 - GA crossover .................................................................................. 36
Figure 11 - GP crossover ................................................................................... 38
Figure 12 - Rule extraction ................................................................................. 42
Figure 13 - Pedagogical rule extraction .............................................................. 44
Figure 14 - Transparent but not comprehensible model .................................. 45
Figure 15 - Complex-IF rule evolved by G-REX ............................................. 48
Figure 16 - IF-REG rule evolved by G-REX ..................................................... 49
Figure 17 - Simple rule with introns ................................................................. 60
Figure 18 - The simplification process ............................................................... 61
Figure 19 - Extracted unsimplified rule for CLE ............................................. 66
Figure 20 - Simplified rule for CLE ................................................................. 66
Figure 21 - Bins affected by outliers .................................................................. 80
Figure 22 - Trimmed bins ................................................................................ 80
Figure 23 - Gain chart for fold 1 of SER ......................................................... 84
Figure 24 - Sample rule for LINREG ............................................................... 97
Figure 25 - Sample rule for FREE ..................................................................... 98
List of Tables

Table 1 - The balloons data set ................................................................. 16
Table 2 - Critical values of the Wilcoxon test at 0.05 significance level ....... 21
Table 3 - F-distribution Critical values for P=0.05 ................................... 22
Table 4 - Critical values for the Nemenyi test ........................................... 22
Table 5 - Critical values for the Bonferroni-Dunn test ................................ 23
Table 6 - BNF for Boolean representation language .................................. 37
Table 7 - BNF for the IF-ELSE representation language .......................... 38
Table 8 - BNF for the Complex-IF representation language ....................... 48
Table 9 - BNF for the IF-REG representation language ............................. 49
Table 10 - Contribution of case studies .................................................... 52
Table 11 - Characteristics of classification data sets ................................. 56
Table 12 - Characteristics of estimation data sets ....................................... 57
Table 13 - Accuracy on individual data sets (CD=1.09) ............................... 64
Table 14 - Size of extracted rules (CD=1.09) ........................................... 64
Table 15 - Rule size after simplification (CD=0.792) .................................. 65
Table 16 - G-REX settings used in all experiments ..................................... 72
Table 17 - BNF for the Adjusted IF-ELSE representation language ............. 73
Table 18 - Average ACC over 10-folds (CD=1.21) ...................................... 74
Table 19 - Average AUC over 10-folds (CD=1.21) ..................................... 74
Table 20 - Average BRE over 10-folds (CD=0.879) .................................... 75
Table 21 - Ensemble MSE ........................................................................ 83
Table 22 - PDE using mode scores (CD = 2.15) ......................................... 83
Table 23 - PDE using support scores (CD = 2.15) ...................................... 84
Table 24 - G-REX Settings ..................................................................... 89
Table 25 - Intra-model consistency (ZOO) .................................................. 91
Table 26 - Inter-model consistency (PID) .................................................... 91
Table 27 - Average consistency over all pairs and folds .............................. 91
Table 28 - Accuracy comparison between G-REX and CART ..................... 92
Table 29 - Comparison between methods CD = 1.38 ................................. 92
Table 30 - BNF for LINREG .................................................................... 97
Table 31 - Free representation (FREE) ..................................................... 98
Table 32 - GMRAE on individual data sets CD=1.97 ................................. 100
List of publications

Journal papers


International Conference Papers


Acknowledgments

First of all I want to thank my supervisor Professor Lars “The dark horse” Niklasson, University of Skövde, for all help, support and encouragement that made this thesis possible.

Special thanks to my assistant supervisor Dr. Ulf “Hawkeye” Johansson, who motivated me to start my PhD studies and then supervised my daily work with keen eyes. This thesis would definitely not have happened without your help and support. Are you sure that it wasn’t your eye, there on top of Barad-dûr?

Obviously I’m very grateful to the University of Borås, University of Skövde, ICA Handlarnas AB and the Knowledge Foundation, for funding my PhD studies. Especially, Rolf Appelqvist, and Mikael Lind at University of Borås, and Lars Niklasson, Mattias Strand at University of Skövde, who together made my studies possible. At ICA Handlarna AB, I especially want to thank Thomas Khanestad, Johan Petterson, Christer Nyman and Anna Byberg for interesting discussions and valuable insights about problems in the “real world”. With some luck, my PhD thesis will contain some solutions that will make your lives a bit easier.

I especially want to thank Cecilia Sönströd who helped proooofreading this thesis, I will buy you a set of new red pens, since yours surely must be out of ink. I also want to thank Andrées Stråhle (who helped giving birth to G-REX), Marcus Hedblom (who showed me that this was possible) and Tuve Löfström (who is a good discussion partner and a superb spiritual guide).

Finally I want to thank my family, Lorena, Birgitta, Leif, Susanna and Olof who supported and believed in me without having a clue of what I was doing. I wouldn’t recommend you to read this thesis, as it probably will bore you to death and make everything even more confusing. Special thanks to Lorena who kicked me out of bed every morning, and encouraged me when I needed it the most.
1 Introduction

Today decision makers often have access to numerous information sources containing vast amounts of data and information; i.e. processed data. Often the data is collected using several sensors to gain as much information as possible about activities related to a decision. [BPR02] argue that this approach is not necessarily the best one to support a decision maker. First, all information may not be required to adequately execute the task. Secondly, the resources required to process all this information might exceed the human information processing capabilities. This problem is also recognized by [HL01] and [SHB+03] as they point out that a decision maker will need an automated or semi automated decision support system (DSS) to make use of large quantities of data, especially if the data is of heterogeneous nature. A DSS needs to be flexible, easy to use and robust enough to perform well over a large set of problems to be useful in the ever changing environment of today.

Information fusion (IF) is a research area that focuses on achieving synergetic effects by combining different information sources. There are numerous definitions of IF, for examples see [HL01], [CGC+05], [Das01], [HW01] and [GB05] but what they all have in common is the emphasis on maximizing the useful information content acquired from heterogeneous sources. IF techniques are sometimes implemented in a fully automatic process or in a human-aiding process for analysis and/or decision support [HL01].

A typical situation that requires a DSS is when the result of some future situation needs to be predicted based on a large set of examples. In information Fusion and the related area of Data Mining this task is called predictive modeling, i.e., the task of predicting an unknown (often future) value of a specific variable based on a model learned from known examples.

A predictive model is created from variables that are thought to provide information of the behavior of the dependent (predicted) variable. These variables are called independent or causal variables and are assumed to have known constant values. Predictive modeling techniques try to create models that maximize a score function which most often is designed to optimize the performance metric used for evaluation.

For predictive modeling the primary performance goal is to achieve high accuracy, i.e. a low error between the predicted value and the real value, when the model is applied to novel data. For certain problems, a simple model, e.g. a linear regression model, is enough to achieve sufficient level of accuracy. Other problems require a more complicated model, e.g. a non-linear neural network or an ensemble. It is well known that ensembles, i.e. a combination of different
models, are able to improve both accuracy and robustness of predictions compared to single models [KV95]. Another advantage of ensembles techniques is that they can eliminate a lot of parameter tweaking which is often needed for creating and finding the best single model.

The choice of a highly complex model as an ensemble, however, entails that the model is hard or impossible to interpret, a dilemma often called the accuracy versus comprehensibility tradeoff. Neither the model as a whole nor the reasons behind a single prediction can be explained when using a complex opaque model.

Sometimes it is impossible to obtain a sufficient performance even with a complex model since no model can be better than the data it was created from. Data can contain uncertainties, low precision, noise or simply lack relevant data making it impossible to create a model with sufficient accuracy. In these cases it is crucial that the model is comprehensible so manual adjustment can be done in a rational way [Goo02]. Even if a complex opaque model would be better than a transparent model with regards to accuracy, it may not be good enough to be trusted in practice. Without any explanation of an uncertain prediction it cannot be adjusted or verified and thus cannot be trusted and used for decision support.

Other motivations for manual adjustments and therefore comprehensible models are that all relevant data cannot always be presented in a structured manner, all data is not always available at prediction time or that a drastic environmental change may have occurred. Furthermore, it can be crucial to be able to explain the underlying rules of a prediction for legal reasons or for making it possible for decision maker to learn from the found relationships [Goo02]. Finally most decision makers would require at least a basic understanding of a predictive model to use it for decision support [Goo02], [DBS77], [Kim06].

Rule extraction (RE), which is one approach to reducing the accuracy versus comprehensibility tradeoff, is the process of transforming the opaque model into a comprehensible model while retaining high accuracy. Rule extraction is based on the assumption that an opaque model can remove noise from the original data set and thus simplify the prediction problem. The extracted model can be used to explain the behavior of the opaque model or be used to make the actual predictions. Any rule representation could be used for the extraction as long as it is understandable for the decision maker.

Another issue in predictive modeling is that accuracy is not always the most suitable performance metric due to several reasons as pointed out by [Faw06]. Some techniques for example, associate a probability estimate to its predictions which accuracy takes no account for. This is a serious drawback as [MF04] notes
that probability estimates are central in decision theory since they allow a
decision maker to incorporate costs/benefits of evaluating alternatives. Another
use for probability estimates is to judge if the model can be used for some specific
instances even if the overall accuracy is insufficient for a fully automatic system.

In any case there is no single performance metric that is suitable for all
problems and it is thus important to be able to use a score function that actually
optimizes the sought performance. Yet, most techniques only have a predefined
score function optimizing accuracy due to underlying algorithmic issues. Thus,
the choice of technique most often leads to optimization of a predefined error
metric when it would be more natural to choose the performance metric best
suited to the problem at hand.

A similar problem applies to the model representation since most techniques
only optimize models of a predefined structure with specific operators. Again the
choice of technique dictates the representation when it would be more suitable to
tailor both the structure and operators to the current problem.

In line with the reasoning above, [Goo02] argues that DDS research should
focus on designing methods and support systems that work with judgmental
forecasters, enabling them to use their judgments effectively, but also encouraging
them to make full use of statistical methods, where this is appropriate.
1.1 Key Observations
The previous description leads to the following key observations:

- Ensembles techniques for predictive modeling are central for decision support as they are powerful, robust and easy to use.
- Ensembles are opaque and give no explanation of the model or the reasons behind a single prediction.
- Poor data quality can make it impossible to create a model with sufficient performance.
- Manual adjustment of model predictions can only be made in a rational way if the model is comprehensible.
- Global model performance is not always sufficient for evaluating model quality and usefulness.
- Many predictive techniques are rigid and cannot be adapted to give a better decision support for a specific problem.
- Rule extraction can provide explanations of opaque models using different representation languages.

1.2 Problem Statement
When opaque models with insufficient performance are used for decision support there is a need for techniques and methods that enable model analysis and provide explanation of predictions.

1.3 Thesis Objectives
With the problem statement and the key observations in mind the following thesis objectives have been selected.

1. Implement and evaluate novel components in a rule extraction framework for improved comprehensibility.
2. Implement and evaluate novel techniques for estimating uncertainty in predictions.
3. Evaluating use of alternative score functions and metrics in predictive modeling

1.4 Main Contributions
This thesis suggests a method for handling situations where poor data quality prevents creation of predictive models with sufficient performance. The idea is to use ensembles as predictive technique, since they are powerful, robust and can be used to estimate prediction uncertainty. After that, rule extraction is performed to facilitate decision support for manual adjustments of predictions with high
uncertainty. In this way the ensemble can be used in spite of insufficient global performance and the need of manual adjustments are minimized.

The main contributions of this thesis are three novel techniques that can be used to enhance the result of the purposed method. The first technique concerns ensemble uncertainty estimations, while the other techniques are improvements of a rule extraction technique called G-REX (Genetic Rule EXtraction):

- A novel technique for ensemble uncertainty estimation based on an approach used in weather forecasting. (Study 9.3)
- A novel technique that enhances the comprehensibility of rules extracted using G-REX. (Study 9.1)
- A novel technique that uses G-REX’ inconsistency to improves uncertainty estimations of extracted rules, thus improving confidence. (Study 9.4).

In addition to these techniques, the advantage of three GP abilities, are demonstrated using G-REX. These abilities should of course also be regarded as advantages for a rule extraction technique used in the purposed method.

- Ability to adapt the representation language to a decision maker’s preferences and to domain specific requirements, thus increasing performance and comprehensible. (Study 9.5).
- Ability to increase rule quality, by optimizing alternative score functions, designed for problem dependent requirements. (Study 9.2).
- Ability to create alternative explanations to help analyzing troublesome predictions (Study 9.4).

1.5 Thesis outline

In the following sections the theoretical background will be presented starting with a more detailed description of information fusion. Next the fundamental concepts of knowledge discovery are introduced in chapter 3 which is followed by a discussion of different error metrics and statistical evaluation techniques in chapter 4. Relevant predictive modeling techniques are described in chapter 5 and some previous work is addressed in chapter 6. Note that there is no special chapter for related work, which is instead presented in respective case study. In chapter 7 cases studies are motivated and related to the thesis objectives. The data sets used in these cases studies are presented in chapter 8, which is followed by the cases studies in chapter 9. Each case study starts with a short summary which
is followed by theoretical background, a description of the method, experimental
details, results and conclusions. More general conclusions for all studies are
presented in chapter 10. Finally, future work is described in chapter 11.
2 Information Fusion

Information fusion is about fusing information from different sources in order to facilitate understanding or providing knowledge that is not evident from the individual sources. Closely related terms (or research areas) are data mining and knowledge discovery, two terms commonly used from an artificial intelligence (AI) perspective for the extraction of regularities from large data sets, [ANO05]. Similarly, decision support constitutes another research area whose subject matter largely overlaps with information fusion, although with a stronger focus on the perspective of the human decision maker. Uncertainty management, i.e., the reduction of overall uncertainty through the combination information sources, is a very central task in IF.

Information fusion not only deals with actual fusion processes, but also with how the results of these processes can be used to improve decision-making. Decisions can be made based on fusion of information from past and present behavior as well as on projections of future behavior. [ANO05]

2.1 OODA-Loop

Boyd’s OODA Loop (Orientation, Observe, Decide, Act) has a military origin and is today central to IF and the dominant model for command and control. The loop is an iterative process that contains the four tasks described originally in [Gra05].

- **Orientation**: An interactive process of many-sided implicit cross-referencing projections, empathies, correlations, and rejections that is shaped by and shapes the interplay of genetic heritage, cultural tradition, previous experiences, and unfolding circumstances.
- **Observe**: The process of acquiring information about the environment by interacting with it, sensing it, or receiving messages about it. Observation also receives internal guidance from the Orientation process, as well as feedback from the Decide and Act processes.
- **Decide**: The process of making a choice among hypotheses about the environmental situation and possible responses to it. Decide is guided by internal feed-forward from Orientation, and provides internal feedback to Observe.
- **Act**: The process of implementing the chosen response by interacting with the environment. Act receives internal guidance from the Orientation process, as well as feed-forward from Decide. It provides internal feedback to Observe.
2.1.1 The OODA-loop in Relation to Predictive Modeling

The OODA-loop is relevant for any predictive modeling task as the current situation (examples of the modeled concept and information of the unknown instance) has to be known before a predictive model can be built. When the observe task is finished the predictive model realizes what have been learned and predictions can be made to support upcoming decisions and plans. Finally, actions can be taken and the loop restarts when the prediction model is evaluated to facilitate correction of the made decisions and incorporation of new information into the prediction model.

The ultimate aim of a DSS system is to fully automate the decision process; there is no better decision support than to have the correct decision made for you. A fully automated system has two fundamental requirements. First the system has to have a sufficient performance and secondly the users have to trust the system enough to use it and not to tamper with the predictions. Figure 1 shows how different predictive modeling tasks fits into the OODA-loop.

![Figure 1 - The role of predictive modeling tasks in the OODA-loop](image)

If a DSS can achieve the required accuracy and gain the trust of its users it could be possible to fully automate the orientation and decide steps (the white area of the loop). This is the ultimate goal (surely a utopia) of this research. Observe is already semi-automated and will always require some manual work as it impossible to control all external variables. Domain experience and some initial work have shown that we are far from this goal today, mostly due to poor data quality and lack of information. Until the data quality is greatly improved, a DSS
cannot be fully automated and can only give well-founded advice. It should be noted that current DSS systems are surely comparable or better than the average forecaster hunch, but DSS of today generally lack the necessary trust of its’ users. Therefore, gaining trust and improving accuracy should be main goals of future research. Accuracy could be achieved by increasing data quality or improving methods for how to best use machine learning techniques as neural networks or ensembles. Trust can only be gained by understanding why a certain action should be taken. It is therefore crucial that each prediction is accompanied by an explanation. This will also work as a way to educate the decision makers about important factors affecting the modeled concept. Users could in turn contribute with business rules and initial knowledge to reduce learning time and increase accuracy. User contributions would also work as a way to increase the general trust level of the system.
3 Knowledge Discovery in Databases

Knowledge Discovery in Databases (KDD) is an interactive, iterative procedure that attempts to extract implicit, previously unknown useful knowledge from data [RG03]. There are several slightly different models of the KDD process but one of the more frequently used is CRISP DM [CCK+08] which is a cross industry standard for KDD and defines a cycle of six phases.

- **Business understanding.** This initial phase focuses on understanding the project objectives and requirements from a business perspective, then converting this knowledge into a data mining problem definition and a preliminary plan designed to achieve the objectives.
- **Data understanding.** The data understanding phase starts with an initial data collection and proceeds with activities in order to get familiar with the data. Here the aim is to identify data quality problems, to discover first insights into the data or to detect interesting data subsets to form hypotheses for hidden information.
- **Data preparation.** The data preparation phase covers all activities to construct the final data set. Data preparation tasks are likely to be performed multiple times and not in any prescribed order. Tasks include instance and attribute selection as well as transformation and cleaning of data for modeling tools.

![Figure 2 - Phases of CRISP-DM](image-url)
• **Modeling.** In this phase, various data mining techniques are selected and applied and their parameters are calibrated to optimal values. Some techniques have specific requirements on the form of the data. Therefore, stepping back to the data preparation phase is often necessary.

• **Evaluation.** An analysis of the developed model to ensure that it achieves the business objectives. At the end of this phase, a decision on the use of the data mining result should be reached.

• **Deployment.** The deployment phase can be as simple as generating a report or as complex as implementing a repeatable data mining process across the enterprise.

KDD and data mining have sometimes been used as synonyms, but over the last few years KDD has been used to refer to a process consisting of many steps, while data mining is only one of these steps [Dun03]. According to this definition, the data mining process is performed within the modeling step of CRISP-DM.

### 3.1 Data Mining

Data mining is often defined as the automatic or semi automatic process of finding meaningful patterns in large quantities of data. The process needs to be automatic due to the very large amounts of available data and the patterns found needs to be meaningful. Data mining is the core of the modeling phase of CRISP-DM but is also strongly connected to the preparation and evaluation phases. The patterns found during a data mining session are given as a model which is also called a concept description since the model is supposed to describe a real world concept based on a set of examples of this concept. All data mining techniques are thus induction based since they try to form general concept definitions by observing specific examples of the concept to be learned [RG03]. A concept is a real phenomenon such as weather, customer behavior, house prices etc. Customer behavior could maybe be described by the attributes income, family size, weekday etc. An example is a recorded instance of the concept where the attributes have specific constant values.

#### 3.1.1 Data Mining Tasks

Data mining problems are usually broken down to a number of tasks that can be used to group techniques and application areas. Even if the number of tasks and the names of the tasks vary slightly, they all cover the same concept, [BL04] defines the following data mining tasks:
- **Classification.** The task of training some sort of model capable of assigning a set of predefined classes to a set of unlabeled instances. The classification task is characterized by well-defined classes, and a training set consisting of preclassified examples.

- **Estimation.** Similar to classification but here the model needs to be able to estimate a continuous value instead of just choosing a class.

- **Prediction.** This task is the same as classification and estimation, except that the instances are classified according to some predicted future behavior or estimated future value. *Time series prediction,* also called *forecasting,* is a special case of prediction based on the assumption that values are dependent of previous values in the time series. Times series methods are either univariate (one variable is forecasted based on its past realizations) or multivariate (when several variables are used to predict the dependent variable).

- **Affinity Grouping or Association Rules.** The task of affinity grouping is to determine which things go together. The output is rules describing the most frequent combinations of the objects in the data.

- **Clustering.** The task of segmenting a heterogeneous population into a number of more homogeneous subgroups or *clusters.*

- **Profiling or Description.** The task of explaining the relationships represented in a complex database.

These tasks can further be divided into the two major categories *predictive* and *descriptive* tasks [BL04]. Classification, estimation and prediction are all predictive in their nature, while affinity grouping, clustering and profiling can be seen as descriptive.

- **Predictive tasks.** The objective of predictive tasks is to predict the value of a particular attribute based on values of other attributes. The attribute to be predicted is commonly known as the *target* or the *dependent variable,* while the attributes used for prediction are known as *explanatory* or *independent variables.*

- **Descriptive tasks.** Here the objective is to derive patterns (correlations, trends, clusters, trajectories and anomalies) that summarize the underlying relationships in the data. Descriptive data mining tasks are often exploratory in nature and frequently require postprocessing techniques to validate and explain results.
Prediction is the most important task seen from a business perspective as it is the only task dealing with future values. However, prediction is nothing else but a special case of classification and estimation and most of the techniques defined for these tasks can also be applied to prediction. Hence, this chapter will only discuss predictive tasks. Univariate methods are not considered as most situations where decision support is needed contain strong explanatory dependencies.

3.1.2 Machine Learning vs. Data Mining
Most data mining techniques come from the field of Machine Learning (ML). According to [Mit97] the field of machine learning (ML) is concerned with the question of how to construct computer programs that automatically improve with experience. The algorithms and techniques used within ML exploit ideas from many fields including statistics, artificial intelligence, philosophy, information theory, biology, cognitive science, computational complexity and control theory. ML also has many other applications than data mining, e.g., information filtering and autonomous agents etc. However, as this research is focused on how to combine different data sources and how to find meaningful patterns, only ML techniques applicable to data mining tasks will be covered.

3.1.3 Statistics versus Data Mining
There is no sharp line between statistics and data mining, some concepts such as decision trees (see chapter 5.2) have even been evolved independently in both fields. However there are slightly different underlying approaches where statistical methods foremost are designed for hypothesis testing while data mining methods are more focused on searching for the best among all possible hypothesis. [WF05]

3.2 Predictive Modeling
Predictive modeling is the task of building a concept model that expresses the target variable as a function of the explanatory variables. The goal of predictive modeling is to minimize the difference between the predicted and real values.

A model representation consists of a set of parameters (attributes, operators and constants) arranged in some kind of structure. Predictive modeling is the process of tuning or training the parameters of the model using a data mining algorithm to fit a set of instances of the concept as well as possible. The instances that are used to build the model are consequently called the training set. A model can have a predefined static structure or it can be developed dynamically during training.
How well a model fits a specific training set is defined as the difference or the error between the predicted values $\hat{y}$ and the real value $y$ and is calculated for the training set by the use of an error metric. When the training error is sufficiently small, the training stops and the model can be used to make predictions for unseen novel data. Normally some of the training instances are set aside prior to training, thus creating a test set which is used to evaluate how good the model will perform on novel data.

![Figure 3 - Possible model of the balloon data set in Table 1](image)

3.3 Data
A trained model is never better than the data used for training and it is therefore important to ensure that the right data is available before modeling starts. If the domain is unfamiliar it is important to consult domain experts to ensure that no potentially important attributes are missing, since no technique can model relationships that are not present in the data. In general it is better to include
more data than less as the data mining techniques are supposed to be able to find the important relationships and disregard superfluous attributes [BL04].

As mentioned above a data set contains a set of instances where each instance consists of specific values of the attributes thought to describe the targeted concept. Table 1 shows a data set containing ten instances of the concept of inflating a balloon (a slightly modified version of the balloon data set in the UCI repository [BM98]). There are four input attributes; color, size, start condition, age and one target attribute, outcome which is the attribute subject to prediction.

<table>
<thead>
<tr>
<th>Color</th>
<th>Size</th>
<th>Start</th>
<th>AGE</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>YELLOW</td>
<td>SMALL</td>
<td>STRETCH</td>
<td>25</td>
<td>INFLATED</td>
</tr>
<tr>
<td>BLUE</td>
<td>SMALL</td>
<td>STRETCH</td>
<td>42</td>
<td>INFLATED</td>
</tr>
<tr>
<td>YELLOW</td>
<td>SMALL</td>
<td>STRETCH</td>
<td>5</td>
<td>DEFLATED</td>
</tr>
<tr>
<td>BLUE</td>
<td>SMALL</td>
<td>DIP</td>
<td>34</td>
<td>DEFLATED</td>
</tr>
<tr>
<td>YELLOW</td>
<td>SMALL</td>
<td>DIP</td>
<td>7</td>
<td>DEFLATED</td>
</tr>
<tr>
<td>?</td>
<td>LARGE</td>
<td>STRETCH</td>
<td>52</td>
<td>INFLATED</td>
</tr>
<tr>
<td>YELLOW</td>
<td>LARGE</td>
<td>STRETCH</td>
<td>23</td>
<td>INFLATED</td>
</tr>
<tr>
<td>BLUE</td>
<td>LARGE</td>
<td>?</td>
<td>3</td>
<td>DEFLATED</td>
</tr>
<tr>
<td>YELLOW</td>
<td>LARGE</td>
<td>DIP</td>
<td>?</td>
<td>DEFLATED</td>
</tr>
<tr>
<td>BLUE</td>
<td>LARGE</td>
<td>DIP</td>
<td>7</td>
<td>DEFLATED</td>
</tr>
</tbody>
</table>

Table 1 - The balloons data set

There are two main types of attributes; numerical which describe attributes that can take a value within a certain range and categorical (also called nominal) where only a set of predefined values is allowed. If the order of the categorical values has a meaningful interpretation it is seen as a special type of attributes called ordinal attributes [TSK06] divide numerical attributes into interval where the differences between the values are meaningful and ratio where both the difference and ratios are meaningful. Temperature recorded in the Celsius scale is an interval attribute as ratios is not meaningful; e.g. 2 degrees Celsius is not twice as hot as 1 degree. Examples of ratio attributes are length, weight, age and monetary quantities.

In Figure 3 Color, Size, Start and Outcome are nominal attributes while Age is numerical, or more precise, a ratio attribute.

3.3.1 Missing Values

As is common in many real world data sets, the data set in Table 1 contains several instances that have missing values (marked with ?) for some attribute. Data can be missing for several reasons; the value was not recorded, the attributes were not present for the instance, or the value could be missing due to some other
error. Whatever the cause, it is important to realize that missing values have impact on the final model depending on how they are handled. Most data mining algorithms are based on the assumption that no data is missing and require that missing values are removed prior to modeling.

There are several strategies that could be used to handle missing values. Instances with missing values could be removed, missing values can be replaced with a certain value not present in the data or they can be replaced with a value that is representative for the data set. However all strategies have their own flaws and which one to choose has to be decided from case to case. A common method for continuous attributes is to replace the missing value with the mean value of instances with no missing values. In the same way nominal missing values can be replaced with the mode value (the most common class).

3.3.2 Outliers
Outliers are values that differ greatly from the normal distribution of an attribute. A common statistical approach is to model an attribute by fitting a Gaussian probability function to the data [TSK06]. A Gaussian model is fitted to a particular attribute by calculating the mean $\mu$ and the standard deviation $\sigma$ of the attribute. The probability of $x$ belonging to that distribution is then calculated by (1) [WF05].

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \hspace{1cm} (1)$$

Figure 5 shows the probability that a value belongs to a Gaussian distribution with a mean of 0 and a standard deviation of 1. The scale of the x-axis is in standard deviations and the y-axis shows the probability. Normally an outlier is defined as a value more than three standard deviations from the mean which correspond to 0.27% of all values.
Outliers can be removed, marked or replaced by a representative value. However it should be clear that an outlier treated in this way is some kind of error and not just a rare value.

3.4 Preprocessing
As seen above, there are several aspects related to the data that have to be handled before modeling can begin. All processes concerning data modification before modeling are known as preprocessing. Removing missing values and outliers are often followed by some kind of transformation of the data according to the need of the data mining technique. One common technique is normalization of the data to remove differences in scale of the attributes and another is dividing the data into training and test sets.

3.4.1 Normalization
Many data mining technique are sensitive to the scale of the attributes. If not handled, attributes with a larger scale will implicitly be given a higher importance during modeling. This is easily handled by normalizing (2) the attributes to a certain range, often 0 to 1. If the max and min values are not known, standardizing (3) can be used to transform all attributes so they get zero mean and unit variance. Below, $X$ is the attribute to be transformed, $x'$ the new value of $x$, $\mu$ is the mean and $\sigma$ the standard deviation

$$x' = \frac{x - \min(X)}{\max(X) - \min(X)} \quad (2)$$

$$x' = \frac{x - \mu}{\sigma} \quad (3)$$
4 Evaluation of Predictive Models

In predictive modeling a model is trained to minimize the error on a training set. However, often this error is a bad estimation of future performance [WF05]. Each prediction is given by some probability function created from a training set which in practice rarely contain all possible instances. Therefore the probability function will not be 100% representative and thus the model cannot be 100% representative. For this reason the training error will always be optimistic in regards to the actual error for the whole population. A better approach is to use a holdout sample or test set to estimate the error on unseen data and compare classifiers. However, the test set will not be representative, but it will be unbiased and can therefore be used to estimate the true error.

4.1 Holdout Sample
A holdout sample is created prior to training by randomly removing a subset of the training instances and placing them in the test set. Normally two-thirds are used for training and one-third for testing. A downside of this technique is that there is less data available to train the model which can result in an inferior model. On the other hand if too little data is used for the test set it will contain a large variance making the estimated error less reliable. Another problem is that the class distribution can differ between the data sets and thus affect both the model and the estimation negatively.

4.2 Random Subsampling
An extension of the holdout sample technique is called random subsampling which simply repeats the process several times. The overall error is calculated as the average error for all models. A problem with this technique is that there is no control of how many times an instance is used for test and training.

4.3 Cross Validation
Cross validation is a more systematic approach to random subsampling. Instead of choosing the holdout set randomly several times, each instance is randomly assigned to one of $k$ equal sized subsets or folds before training. Next, $k$ models are trained each using one of the folds as test set and the other $k-1$ folds as training so that each fold is used as a test set once. In this way cross validation utilizes all instances and each instance is used once for testing and $k-1$ times for training. However, the folds can still be unrepresentative as they are chosen randomly. One timeconsuming approach to this problem is a special case of cross validation called leave one out, where each fold contains a single instance. Leave one out uses the maximum amount of data for training the models and each test
is mutually exclusive, but the method is rather unpractical for most problems. A better approach is to ensure that all folds have a representative class distribution when they are created. Stratification is a simple technique that does this by selecting instances (in a random manner) in a way that ensures that each fold will get an equal number of instances of each class. For cases where the number of folds and instances of a certain class do not add up, a copy of an already selected instance can be used, or the fold can be left as it is. Most often, 10-fold cross validation with stratification is used to evaluate classifiers.

4.4 Comparing Models

Sadly there are no free lunches in data mining, i.e. no technique can outperform all others over the space of all possible learning tasks [Sch94]. Hence, when performing data mining it is normal to create several models using different techniques and settings. However, this means that the models need to be compared in a structured manner to ensure that the best model is actually selected. There has been much research on the topic of the best way of comparing classifiers and numerous tests are available. [Dem06] have reviewed the current practice and examined several suitable tests both theoretically and empirically. Two non-parametric tests are recommended, the *Wilcoxon* signed ranks test for comparison of two classifiers and the *Friedman test* with the corresponding post-hoc tests for comparison of several classifiers over multiple data sets.

4.4.1 Wilcoxon Signed Rank Test

If two models are compared over several data sets [Dem06] recommends the *Wilcoxon signed rank* test [Wil45], which is based on the ranked difference between the models. First the absolute value of the difference in performance $d_i$ on each data set is ranked from the lowest to the highest. In case of equal performance, the ranks of these data sets are split evenly, if there is an odd number one is ignored. Next the rankings are summed for the cases which gives a positive difference $R^+$ and a negative difference $R^-$ according to (4).

$$R^+ = \sum_{d_i > 0} \text{rank}(d_i) + \frac{1}{2} \sum_{d_i = 0} \text{rank}(d_i)$$

$$R^- = \sum_{d_i < 0} \text{rank}(d_i) + \frac{1}{2} \sum_{d_i = 0} \text{rank}(d_i)$$

If the smallest of the sums $T = \min(R^+, R^-)$ are less or equal to the corresponding critical value ($CD$) in Table 2, the hypothesis that the models are equal in performance can be rejected.
### Table 2 - Critical values of the Wilcoxon test at 0.05 significance level

For comparison containing more than 25 data set (5) can be used instead of Table 2. Here the hypothesis can be rejected at a significance level of 0.05 if $z$ is less than -1.96.

\[
z = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}}
\]  

\[\text{(5)}\]

#### 4.4.2 Friedman Test

A Friedman test [Fri37] is a nonparametric test for comparing a number of models over several data sets using their rank. Ranking is done by giving the best a rank of 1, the second best 2, etc. In case of a tie, the average rank of the tied models is assigned. When a model is ranked after a tie it is assigned a rank according to how many models that been ranked so far plus 1. The null-hypothesis for the Friedman test is that all models are equal in their performance and hence also their ranks $R_j$. The Friedman statistic is distributed according to $\chi^2$ with $k - 1$ degrees of freedom when $N>10$ and $k>5$. In cases with fewer models and fewer data sets exact critical values can be found in [Shes00].

\[
\chi^2_F = \frac{12N}{k(k+1)} \left[ \sum R_j^2 - \frac{k(k+1)^2}{4} \right]
\]  

\[\text{(6)}\]

However, the Friedman test has been shown to be unnecessary restrictive and hence modified according to (7) with $k - 1$ (numerator) and $(k-1)(N-1)$ degrees of freedom (denominator).

\[
F_j = \frac{(N-1)\chi^2_F}{N(k-1)-\chi^2_F}
\]  

\[\text{(7)}\]

Some sample critical values for the F-distribution are presented in Table 3. The significance level is 0.05 and the numerator is given by the column and the denominator by the row.
If only two models are compared the ANOVA test has more power if the requirements of an ANOVA test are fulfilled. However, the ANOVA requirements are not always met and it has also been shown that ANOVA and the Friedman test in practice most often give the same results [Dem06].

4.4.3 Nemenyi Test

If the null-hypothesis of the Friedman test is rejected [Dem06] recommends the use of the Nemenyi post hoc test [Nem63] to analyze which of the models that differ significantly from each other. When all \( k \) models are compared to each other over \( N \) data sets, a difference in average rank greater than the critical difference \( CD \), see (8) is significant.

\[
CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}
\]  

(8)

Table 4 gives the value of \( q_\alpha \) depending on the significance level \( \alpha \) and the number of models \( (k) \).

<table>
<thead>
<tr>
<th>#Models ((k))</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_{0.05} )</td>
<td>1.960</td>
<td>2.343</td>
<td>2.569</td>
<td>2.728</td>
<td>2.850</td>
<td>2.949</td>
<td>3.031</td>
<td>3.102</td>
<td>3.164</td>
</tr>
<tr>
<td>( q_{0.10} )</td>
<td>1.645</td>
<td>2.052</td>
<td>2.291</td>
<td>2.459</td>
<td>2.589</td>
<td>2.693</td>
<td>2.780</td>
<td>2.855</td>
<td>2.920</td>
</tr>
</tbody>
</table>

Table 4 - Critical values for the Nemenyi test

If only a single model needs to be compared to a set of models, a more powerful post hoc test such as the Bonferroni-Dunn test [Dun61] should be used. The only difference from the Nemenyi test is the values of \( q_\alpha \) in equation (8) see Table 5.

As seen in the table, the difference between the two tests gets greater with the number of classifiers.
<table>
<thead>
<tr>
<th>#Models(k)</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>(q_{0.05})</td>
<td>1.960</td>
<td>2.241</td>
<td>2.392</td>
<td>2.498</td>
<td>2.576</td>
<td>2.638</td>
<td>2.690</td>
<td>2.724</td>
<td>2.773</td>
</tr>
<tr>
<td>(q_{0.10})</td>
<td>1.645</td>
<td>1.960</td>
<td>2.128</td>
<td>2.241</td>
<td>2.326</td>
<td>2.394</td>
<td>2.450</td>
<td>2.498</td>
<td>2.539</td>
</tr>
</tbody>
</table>

Table 5 - Critical values for the Bonferroni-Dunn test

4.5 Error Metrics for Classification Tasks
Error metrics are different mathematical functions for calculating the error or difference between the predicted value \(\hat{y}\) and the real value \(y\). Error metrics are used to select the best of several models. Since all metrics have slightly different properties it is important to ensure that the chosen metric actually measure the right thing. Accuracy is by far the most common error metric used for classification. However, [PTR98] argues that using accuracy for evaluating classifiers has two serious flaws; it is presumed that the true class distribution is known and that the misclassification cost is equal for all classes. This is regarded as a problem as these assumptions rarely are true for real world problems. Instead other metrics such as Receiver Operating Characteristic (ROC) curve or Brier score are suggested as a more appropriate metric for comparing classification performance. The following chapter will therefore describe these metrics in detail.

4.5.1 Accuracy (ACC)
Accuracy is simply the percentage of all instances that are classified correctly.

\[
ACC = \frac{\sum_{i=1}^{n} j_i}{n} \quad where \quad j_i = \begin{cases} 
1 & \text{if } \hat{y}_i = y_i \\
0 & \text{if } \hat{y}_i \neq y_i 
\end{cases}
\]

As mentioned above, accuracy assumes that the class distribution is known for the target environment. If this distribution is not known accuracy is a poor error metric as it is not possible to claim that the evaluated model is indeed maximizing accuracy for the problem from which the data set was drawn [PTR98].

4.5.2 Area Under ROC-curve (AUC)
ROC curves measure the relationship between hit rate and false alarm rate and are based on estimations of the probability that an instance belongs to a certain class. ROC has an especially attractive property in that they are insensitive to changes in class distributions. ROC graphs are commonly used in medical decision making, and have in recent years been used increasingly in machine learning and data mining research; see e.g. [Faw06].

In a ROC-curve diagram, the y-axis expresses the number of true positives and the false positives are shown on the x-axis. Both axes show the percentage of all
true and false positives to give the unit square area the sum of 1. To draw a ROC-curve the instances are first ranked according to the probability that they belong to the positive class. Next the curve is drawn by moving one step up for each instance that are a true positive and taking one step right if it is a false positive.

![Sample ROC curve](image)

**Figure 6 - Sample ROC curve**

For a detailed description of how ROC curves are constructed see [Faw06]. To compare classifiers using ROC curves, the curve has to be converted into a single scalar value, normally the area under the ROC curve (AUC). Since ROC curves are projected onto the unit square, AUC will always be bounded by 0 and 1. An AUC of 1 means that the predictive model is perfect, while random guessing corresponds to an AUC of 0.5 (the dotted line in Figure 6).

4.5.3 Brier Score (BRI)
A prediction made by a certain model can often be related to a probability estimation. [SK02] points out that for a single instance a probability estimation is neither correct nor wrong, probability estimations are verified by analyzing the joint (statistical) distribution of estimations and observations. While AUC only evaluates how good the produced probability estimates are for ranking the instances the Brier score suggested in [Bri50] evaluates how good these estimations really are. Equation (10) shows the definition of the Brier score (BRI) where \( n \) is the number of instances, \( r \) is the number of possible outcomes (classes), \( f_{ij} \) gives the probability estimate that the instance \( i \) belongs to class \( j \). \( E_{ij} \) takes the value 1 if the event has occurred and 0 if it hasn’t.
\[ BRI = \frac{1}{n} \sum_{j=1}^{r} \sum_{i=1}^{n} (f_{ij} - E_{ij})^2 \]  

A lower score is better and the lowest possible score is 0 which occurs when all classifications are done correctly with 100% certainty. Random guessing from a binary problem with an equal class distribution will result in a score of 0.5.

4.6 Error Metrics for Estimation Tasks

Some error measures are more useful than others. However after examination [ARM01] notes that depending on the situation several error metrics could be relevant and argues for the benefit of presenting all relevant metrics. When comparing different estimation models over several data sets, it is important to use an appropriate error metric. For example if models are going to be compared over several data sets the error metric has to be relative. This is true for both estimations tasks [LZ05] and for time series prediction [ARM01]. A relative metric is a metric which is relative to some reference. Usually some simple straightforward model like the mean value of the target attribute is used for reference. A relative metric is also important since values of the target variable can differ greatly in magnitude between data sets, which of course will affect the magnitude of the error. The reference also accounts for the difficulty of the data set which also is of great concern in a comparison. It is not always worse to achieve a higher error on a difficult task then a slightly lower error on a simple task.

4.6.1 Mean Absolute Deviation (MAD)

Mean absolute deviation is a metric simple to interpret and calculate. A problem with MAD is that it is sensitive to the scale of the predicted variable and thus is unsuitable when comparing models over several data sets.

\[ MAD = \frac{1}{n} \sum_{i=1}^{n} |\hat{y}_i - y_i| \]  

(11)
4.6.2 Coefficient of Determination ($r^2$)
The coefficient of determination $r^2$ is the proportion of the variability in the data that is captured by the evaluated model. $r^2$ is bounded between 0 and 1 where 1 is a perfect correlation and 0 signifies no correlation. It should be noted that $r^2$ takes no account for the bias between the predicted and the actual value, i.e. even if a model has a perfect $r^2$, its prediction can still differ greatly in scale from the actual value [ARM01].

\[ r^2 = \left( \frac{\sum_{i=1}^{n} ((\hat{y}_i - \bar{y}_i)(y_i - \bar{y}_i))^2}{\left( \sum_{i=1}^{n} (\hat{y}_i - \bar{y}_i)^2 \sum_{i=1}^{n} (y_i - \bar{y}_i)^2 \right)^{1/2}} \right)^2 \]  

(12)

4.6.3 Root Mean Square Error (RMSE)
RMSE is a very common metric, which is a bit surprising since it is well known that it is a bad choice for several reasons; it is poorly protected from outliers, has a low reliability and low construct validity [AC92]. It is not always inappropriate to use RMSE, but care has to be taken, especially when forecasting or comparing methods across series with different scale.

\[ RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n}} \]  

(13)

4.6.4 Geometric Mean of the Relative Absolute Error (GMRAE)
In [AC92] several error measures were evaluated in regards of reliability, construct validity, sensitivity to small changes, protection against outliers, and their relationship to decision making. Geometric Mean of the Relative Absolute Error (GMRAE) was presented as one of the strongest measures. GMRAE is based on Relative Absolute Value (RAE) which is defined in (14) where $\hat{y}_{s,m}$ is the prediction (forecast) made by model $m$ on series $s$. RAE is relative to the random walk ($rw$) with zero drift, i.e. the most recent known actual value. For series $s$, $rw$ predicts $y_{s-1}$.

\[ RAE_i = \frac{\frac{\hat{y}_{s,m} - y_s}{\hat{y}_{s,rw} - y_s}}{y_{s,rw} - y_s} \]  

(14)

Random walk is often used as a reference point since it is straightforward and easily interpreted. In addition, random walk most often outperforms the mean.
prediction for time series. To handle extreme values, the RAE is trimmed using Winsorizing according to (15)

\[
WRAE = \begin{cases} 
0.01 & \text{if } RAE_s < 0.01 \\
RAE_s & \text{if } 0.01 \leq RAE_s \leq 10 \\
10 & \text{if } RAE_s > 10
\end{cases}
\]  

(15)

GMRAE summarizes the WRAE using the geometric mean. The arithmetic mean is not suitable as any arithmetic average will be dominated by its large terms. In other words, good small errors should be counted but would essentially be ignored. It might be reasonable to expect that a large error should be balanced by a sufficiently small error which is the case for the geometric average but not for the arithmetic averages [LZ01].

\[
GMRAE = \left[ \prod_{i=1}^{n} WRAE_i \right]^{1/n}
\]  

(16)
5 Data Mining Techniques

5.1 Linear Regression
Linear regression is a simple technique suitable for numeric prediction that is frequently used in statistical application. The idea is to find the amount of how much each of the attributes \( a_1, a_2, \ldots, a_k \) in a data set contributes to the target value \( x \). Each attribute is assigned a factor \( w_i \) and one extra factor is used to constitute the base level of the predicted attribute.

\[
x = w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_k a_k
\] (17)

The aim of linear regression is to find optimal weights for the training instances by minimizing the error between the real and the predicted values. As long as the data set contains more instances than attributes this is easily done using the least square method [WF05]. Linear regression is quite intuitive and easily understood but the downside is that it handles non-numerical attributes poorly and that it can’t handle more complex nonlinear problems [Dun03].

5.2 Decision Trees
Decision trees (DT) are machine learning models most suited for classification task but also applicable to regression problems. In the data mining community, decision trees have become very popular as they are relatively fast to train and produce transparent models. A decision tree can be seen as a series of questions, arranged in a tree structure, leading to a set of predefined classes. Most often a decision tree consists of nodes containing either a prediction or a boolean condition (the question) with two corresponding child nodes.

The conditions are chosen to split the data set used for training into smaller more pure set of records, i.e. sets of instances which are dominated by a single target class. When it’s impossible to find a condition in a node that will make the resulting sets purer it’s marked as a leaf node and labeled to predict the majority class of the instances reaching the node.

When used for classification, the root node is first evaluated followed by the left or the right node depending on outcome of the condition. This process is repeated for each resulting node until a leaf node with a prediction is reached. A path from the root to a leaf can be seen as a rule consisting of simple if else conditions.
5.2.1 Creation
A decision tree is built to minimize the classification error on a training set. The creation of the tree is done recursively by splitting the data set on the independent variables. Each possible split is evaluated by calculating the purity gain it would result in if it was used to divide the data set $D$ into the new subsets $S = \{D_1, D_2, \ldots, D_s\}$. The purity gain is the difference in purity between the original data set and the subsets as defined in (18) where $P(D_i)$ is the proportion of $D$ that are placed in $D_i$. The split resulting in the highest purity gain is selected and the procedure is then repeated recursively for each subset in this split.

$$
    \text{gain}(D, S) = \text{purity}(D) - \sum_{i=1}^{s} P(D_i) \text{purity}(D_i) \tag{18}
$$

There are several different DT algorithms, such as ID3 [Qui86], C4.5 [Qui93], CART [BFO+83] and CHAID [Kas80] which all uses slightly different purity functions.

ID3 use information gain as purity function which is based on the entropy metric that measures the amount uncertainty in a set of data. Information gain is calculated for all possible conditions for all independent variables and the entropy $E$ is defined in (19), where $p_i$ is the probability of randomly choosing an instances of class $p_i$ in the data set $S$.

$$
    E(S) = \sum_{i=1}^{c} p_i \log \left( \frac{1}{p_i} \right) \tag{19}
$$

Entropy ranges between 0 and 1 and reaches a maximum when all class probabilities are equal.
CART, uses the gini diversity index (GDI) that measures the class impurity in the node $t$, given the estimated class probabilities $p^*(j | t), j = 1, \ldots, J$ (for $J$ classes). GDI is given by (20).

$$GDI = 1 - \sum_j p^2(j | t) \quad (20)$$

5.2.2 Pruning
When a DT is fully grown it is optimized on the training set which often leads to an over fitting of the model and thereby a high generalization error. Pruning is a method that analyses a decision tree to make it more general by removing weak branches. Removing parts of a tree will of course result in a decrease in training accuracy but the idea is that it will make the tree more general and perform better on new unseen data.

There are two approaches to pruning; prepruning that tries to stop the growth of the tree before weak branches occur and postpruning where a fully grown tree is first created and then pruned. [WF05] notes that postpruning has some favorable advantages as for example a series of conditions can be powerful together even when they all are weak by themselves.

In general postpruning algorithms generates, candidate subtrees which are evaluated on the training data or a new validation set containing previously unseen instances. Exactly how these candidate subtrees are created differ between algorithms, but they all apply subtree replacement and/or subtree raising in some fashion. Subtree replacement starts in the leaves of the tree and replaces the selected subtrees with single leaves. Subtree raising moves a subtree to a position higher up in its branch deleting intermediate nodes. During pruning, a large number of candidate subtrees can be created and the tree with the best performance on the validation data is selected as the final tree.

5.2.3 Probability Estimation
Although the normal operation of a decision tree is to predict a class label based on an input vector, decision trees can also be used to produce class membership probabilities; in which case they are referred to as probability estimation trees (PETs). The easiest way to obtain a class probability is to use the proportion of training instances corresponding to a specific class in each leaf. In Figure 8, 6 training instances reach the lower right leaf, and 4 of those belong to class Positive. The assigned class probability for class Positive, in that leaf, would become $\frac{4}{6} = 0.67$. Consequently, a future test instance classified by that leaf would be classified as class Positive, with the probability estimator 0.67.
Normally, the relative frequencies are not used directly for probability estimations, since they do not consider the number of training instances supporting a classification. Instead the Laplace estimate is commonly used to produce calibrated probability estimates based on the support [MD02]. Equation (21) shows how probability estimates $p$ are calculated when using Laplace. $N$ is the total number of instances, $C$ is the number of classes and $k$ is the number of training instances supporting the predicted class $A$.

$$p_{classA} = \frac{k + 1}{N + C}$$

In Figure 8 above, the probability estimate for the lower right node would be calculated as $4/6 = 0.67$ without Laplace and $((4 + 1)/(6 + 2)) = 0.63$ using Laplace. It should be noted that the Laplace estimator introduces a prior uniform probability for each class; i.e. before any instances have reach a leaf ($k=N=0$), the probability for each class is $1/C$.

5.3 Artificial Neural Networks
Artificial neural networks (ANN) is a ML technique loosely based on the function of the human brain. ANNs are extremely powerful in the sense that they are universal functional approximators, i.e., they can approximate any function to any desired accuracy [Kec01]. Multi-layer perceptrons (MLP) which is one of the most common types of ANNs have been used to solve a wide variety of problems and are frequently used for both classification and regression tasks due their inherent capability of arbitrary input output mapping [ZPH98].

5.3.1 Basics of a Neural Network
An MLP consists of a set of units (neurons) which are arranged in layers and connected by weighted links which pass signals between the units. The units in the first layer, the input layer, are connected to the input variables and receive inputs signals accordingly. When a signal is sent through a link, it is multiplied with the weight of the link which and thus decides the impact on the receiving unit. All incoming signals are summarized and an activation function is applied to
calculate the responding output. Units in the input layer are connected to units in a hidden layer which can be connected to another hidden layer or to the output layer, see Figure 9.

![Figure 9 - A simple MLP](image)

**Hidden Layers and Hidden Units**
The number of hidden layers and hidden nodes in an ANN is a crucial design choice as they allow the network to detect the features and capture the pattern in the data by performing complicated nonlinear mappings between input and output variables. One hidden layer is sufficient for ANNs to approximate any continuous function the desired degree of accuracy [Hor91]. Two hidden layers can sometimes be preferable as it can achieve the same result with less hidden units and approximate discontinuous functions. A network with fewer units will be faster to train and will in principle generalize better forecasts [ZPH98].

The most common way to determine the number of hidden nodes is via initial experiments. Several rules of the thumb have been proposed but none of these work well for all problems.

**Output Layer and Output Units**
Selecting the number of output units is another important factor for the performance of the network. For regression problems, a single output unit is sufficient but classification tasks are most often designed with a one of C coding where each class is represented by an output unit. Classification is done according to the associated class for the unit with highest output signal.

**Activation Functions**
The activation function determines the relationship between the inputs and outputs of a node and a network. In general, the activation function introduces a
degree of nonlinearity that is valuable for most ANN applications. The most common activation functions are the squashing sigmoidal functions: the *unipolar logistic function* (22) and the *bipolar sigmoidal function* (23), i.e. see [Kec01].

\[
f(x) = \frac{1}{1 + e^{-x}} \quad (22)
\]

\[
f(x) = \frac{2}{1 + e^{-x}} - 1 \quad (23)
\]

Different activation functions can be used in a network but usually only one type is used in each layer. In [ZPH98], the *linear activation function* (24) is recommended for continuous target variables.

\[
f(x) = x \quad (24)
\]

**Training of a Neural Network**

A trained ANN is basically a mapping between input and output variables of the data. Training of the ANN is an optimization process of the connection weights. The most commonly used training algorithm *back propagation* is a gradient steepest descent method. A step size often called *learning rate* and a momentum parameter must be set before training can begin. These parameters play a critical role in the performance of the final network but there is no way to calculate the optimal learning rate and momentum for a given problem. Several experiments with different setting are often performed before selecting the final values.

Broyden-Fletcher-Goldfarb-Shanno (BFGS) and Levenberg-Marquardt [ZPH98] are two alternatives to the back propagation algorithm that are more robust and have achieved significant improvements in training time and accuracy. Another advantage of these methods is that the parameters used by the algorithms can be set automatically.
5.4 Evolutionary Algorithms

Evolutionary algorithms (EA) are a collection of algorithms and techniques inspired by Darwin’s principles of natural selection and survival of the fittest. EAs are essentially methods for optimization based on the natural process which occurs frequently in nature when a species adapts to changes in its environment. There are four main EA paradigms: Genetic Algorithms (GA), Evolutionary Strategies (ES), Genetic Programming (GP) and Evolutionary Programming (EP). However, this chapter will not cover ES as it is similar to GA but less straightforward to apply. EP will neither be described as it is similar to GP but aimed at evolving artificial life.

The main idea in EAs is to solve a problem using a population of competing individuals. The problem defines the constrictions of an environment in which the individuals (candidate solutions) are going to compete. If no prior knowledge exists about the nature of the environment the initial population is created randomly, as this will ensure an approximate average performance of the population as a whole. Each individual is then evaluated and assigned a score (fitness) that correlates to how well they fit the environment. Finally, individuals are selected to be part of a mating pool where they will be exposed to genetic operations to create offspring for a new stronger generation. There are several different strategies for how the individuals are selected to be part of the mating pool, two of the more common are roulette wheel and tournament selection.

In roulette wheel selection each individual is selected with a probability proportional to its fitness. This can be pictured by a roulette wheel where each individual gets a slot where the size of the slot corresponds to its relative fitness. The structure of wheel stays the same for each selection allowing an individual to be selected several times (reselection).

Tournament selection is performed by selecting $k$ individuals randomly and then selecting the one with the highest fitness. A higher $k$ will put a higher weight on the fitness value while a $k$ of one will disregard the fitness and choose individuals randomly.

When creating a new population, there are three fundamental genetic operations; reproduction, crossover and mutation. Reproduction makes a copy of an individual; crossover splits each of the two parents into two parts and recombines them into two new individuals. Mutation makes a random change to an individual. Most EAs implementations are done in a so-called steady state fashion where the genetic operations are applied to individuals in the mating pool until the new population has the same size as the current population. New
generations are continuously evolved in this manner until a certain number of
generations are reached or until a good enough solution is found.

The main difference between GA and GP is the representation of the individuals
and how the genetic operations are applied; this will be described in detail in the
following sections.

5.4.1 Genetic Algorithms
In Genetic Algorithms, first presented in [Hol75], an individual or chromosome
is represented by a bit string. Normally the size of the string is fixed to facilitate
simple crossover operations. The initial population is created by randomly
assigning 1 or 0 to each bit of each individual. A population can contain
hundreds or even thousands of individuals depending on the problem and how
computationally expensive the evaluation of an individual is. To evaluate an
individual the bit string is first translated into a candidate solution and then
evaluated by a fitness function.

In GA a crossover is performed on two parents which are selected randomly
from the mating pool. Next, both parents are split into two parts at the same
randomly decided crossover point. Two new offspring are finally created by
combining one part from each parent in an orderly fashion which preserves the
placement of each part, i.e. see Figure 10

![Figure 10 - GA crossover]

Mutation is simply done by changing a random bit and reproduction makes an
exact copy of the selected individual.

5.4.2 Genetic Programming
In genetic programming, introduced by [Koz92], an individual or program is
represented using a tree structure similar to a decision tree, containing predefined
functions and terminals. Koza originally used the functions defined in the LISP
programming language but there are no restrictions of which functions and
terminals that could be used. However it is important that the function and
terminals sets are chosen with the problem in mind, as GP only can find solutions containing these elements. Selecting the wrong functions and terminals may result in a search space that doesn’t include the optimal solution while superfluous elements will result in an unnecessarily large search pace.

For data mining problems, where one important aspect is comprehensibility, simple representation languages as Boolean or If Else rules are most often used. For Boolean rules, the functions set $F$ consists of and, or and the relational operators greater, less, and equals while the terminal set $T$ is defined by the attributes of the current data set and random constants. Table 6 shows the function set, the terminal set and the BNF for simple Boolean rules such as the ones presented in Figure 11 on the next page. Each row represents a construction (terminal or function), consisting of a name and a set of placeholders separated with bold lines. Thin lines are used to separate optional constructions for a placeholder and optional sets of placeholders (rows) for a construction.

<table>
<thead>
<tr>
<th>Function set</th>
<th>AND</th>
<th>OR</th>
<th>&lt;</th>
<th>&gt;</th>
<th>==</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal set</td>
<td>ConV</td>
<td>CatV</td>
<td>Double</td>
<td>Category</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Boolean rule</th>
<th>Condition</th>
<th>BinaryOp</th>
<th>BinaryOp</th>
<th>BinaryOp</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinaryOp</td>
<td>Boolean rule</td>
<td>AND</td>
<td>OR</td>
<td>Boolean rule</td>
<td></td>
</tr>
<tr>
<td>Condition</td>
<td>ConV</td>
<td>&gt;</td>
<td>&lt;</td>
<td>Double</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CatV</td>
<td>==</td>
<td>Category</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ConV</th>
<th>Continuous variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>CatV</td>
<td>Categorical variable</td>
</tr>
<tr>
<td>Double</td>
<td>Double in range of associated ConV</td>
</tr>
<tr>
<td>Category</td>
<td>Category of associated CatV</td>
</tr>
</tbody>
</table>

Table 6 - BNF for Boolean representation language

Table 7 shows BNF for a basic IF-ELSE representation and the tree displayed in Figure 8 is an example of this representation; if an if-statement is true, the left subtree is evaluated, if not the right. An instance is classified as the value of the leaf it reaches, following the if-statements.
<table>
<thead>
<tr>
<th>Function set</th>
<th>If</th>
<th>&lt;</th>
<th>&gt;</th>
<th>==</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal set</td>
<td>ConV</td>
<td>CatV</td>
<td>Double</td>
<td>Category</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>If Condition</th>
<th>Class</th>
<th>else If Condition</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConV</td>
<td>&gt;</td>
<td>&lt; Double</td>
<td>== Category</td>
</tr>
<tr>
<td>CatV</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Double</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Condition</th>
<th>Class</th>
<th>else Condition</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConV</td>
<td></td>
<td>&gt; Double</td>
<td>== Category</td>
</tr>
<tr>
<td>CatV</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Double</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Continuous variable*

*Categorical variable*

*Double in range of associated ConV*

*Category of associated CatV*

*Class of target variable*

**Table 7 - BNF for the IF-ELSE representation language**

There are three main strategies for how the initial population is created in GP. *Full* creates trees by randomly selecting functions until a certain depth is reached, when terminals are selected for all functions. A tree created with *Full* is therefore always balanced, i.e., all terminals are defined at the same depth. *Grow* randomly selects functions and terminals and can thereby create unbalanced trees. The last strategy *Ramped half and half* creates half of the population using *Grow* and half using *Full* while at the same time varying the maximum depth. *Ramped half and half* is the most commonly used creation strategy since it introduces the most diverse population.

In GP, the crossover operation is performed by randomly selecting a crossover point in each tree and recombining the parts into two new individuals as seen in Figure 11. The crossover point can be different in each tree, but the resulting trees have to be two legal programs. If two incompatible points are chosen, the resulting programs are discarded and the procedure is performed again.

![Figure 11 - GP crossover](image-url)
Mutation is done by randomly choosing a mutation point and deleting everything below this point. Next the Grow creation strategy is applied to create new part of the program starting at the same mutation point. Reproduction is done in the same way as in GA, i.e. by simply copying the program into the next generation.

5.5 Ensembles
An ensemble is a combination of the predictions made by several different classification or regression models. It is well known that a combination of several models can improve the accuracy of the prediction on unseen data, i.e. decrease the generalization error. At the same time, an ensemble is by definition a set of models, making it very hard to express the overall relationship found in original variables. The option to use an ensemble thus is a choice of a black-box model prioritizing accuracy. Models can be combined by simply using majority voting (for classification) and by averaging (for regression) or by weighting in some fashion. [KV95]

A combination of models is only meaningful if they disagree to some extent. If all models produce the exact same output no new information could be gained by combining them. \( \hat{A} \) the disagreement or the ambiguity of the ensemble member models \( m_1 \ldots m_n \) for the instances \( x_1 \ldots x_n \), can be calculated using (25) where \( \omega_m \) is the weight for model \( m \) and \( V \) is the weighted ensemble average. [KV95]

\[
\hat{A} = \sum_{i=1}^{n} \left( \sum_m \omega_m (V^m(x_i) - \bar{V}(x_i))^2 \right)
\]  

(25)

If \( \bar{E} \) is the weighted average of the generalization error (\( \bar{E} = \sum_m \omega_m E^m \)), the generalization error of the ensemble \( E \) follow (26).

\[
E = \bar{E} - \hat{A}
\]

(26)

This means that the accuracy of the ensemble will increase if the ambiguities can be increased without increasing the generalization error.

5.5.1 Calculating Member Weights
There is no analytic way to calculate the absolute weights for the models in the ensemble. It is important to realize that a model can have such poor generalization that its optimal weight should be zero. In the forecast community simple averaging of the models is normally used. A problem with simple averaging is that it assumes that all models are equally good [HSY94]. One approach to overcome this problem is to calculate the optimal linear combination
(OLC) of the ensemble members through their weights. Although various loss functions could be used, the squared-error loss is most often used in practice. The objective is then to minimize the MSE using least square regression, which is why the technique often is called MSE-OLC. For an overview of MSE-OLC see [HSY94]. Since each member is trained on the same data they are prone to colinearity problems, which reduce the robustness of the ensemble. An alternative approach is to allow one model to learn an adequate strategy to combine the ensemble models [LCV02].

5.5.2 Ensemble Creation
If the ensemble members are ambiguous but still have a similar accuracy, calculating the optimal weights becomes unnecessary, as simple averaging will work. This is the assumption for all ensemble creation techniques which try to create good ensembles. Two of the most common techniques are bagging [Bre96] and boosting [Sch89].

**Bagging**
In bagging the ensemble members are created from different training sets formed by making *bootstrap* replicas of the original set. The bootstrap replicas are constructed by randomly selecting instances (with reselection) from the original training set. All replicas have the same number of instances as the original data set but may contain duplicates of some instances. To make a prediction for a regression problem the ensemble members are simply averaged.

**Boosting**
Boosting also creates different data set replicas for each ensemble member but the creation of the replicas is not independent. All replicas are copies of the original training set but have different weights for each instance. An ensemble is created sequentially by adding one new member at a time. The weights are calculated depending on the performance of the current ensemble by giving instances wrongly classified a high weight and thereby a high importance for the new member while the correctly classified instances receive a low weight. When added to the ensemble a classifier is usually weighted in some way that is related to its accuracy. Boosting does not always help the training if the training data contains a lot of noise [Dun03].

5.5.3 Member Selection
Adding more ensemble members to increase accuracy will only work up to a certain point. The reason is that when the size of an ensemble increases the probability that a new member will increase the ensemble ambiguity decreases.
Eventually adding more members will only result in unnecessary computation. Adding new poor members could in some cases even harm the ensemble.

An alternative is to select a subset of candidate ensemble members. This could be achieved by considering the difference in MSE, of the current ensemble with \( K \) members and a new candidate ensemble with \( K+1 \) members. The members are tried in order of their MSE starting with the member with the lowest error. A new member should only be added to the ensemble if the following inequality is satisfied [LCV02]:

\[
(2K+1)MSE[f_k] > 2 \sum_{i=\text{new}} E[m_{\text{new}}m_i] + E[m_{\text{new}}^2]
\] (27)

\( MSE[f_k] \) is the MSE of the ensemble with \( K \) members, \( E[\ ] \) refers to the mathematical expectation operator and \( m_{\text{new}} \) is the error that should be produced by the new member. If this criterion is not satisfied the candidate member is discarded and the next candidate member in the ordered sequence is tested.

The ensemble members could also be selected to maximize the diversity of the ensemble following [KV95]. Several different diversity measures exist and there are many different techniques for constructing the ensembles. For a survey of the most important and common measures and techniques see [Joh07].

5.5.4 Estimating Ensemble Uncertainty

Estimating the uncertainty associated with a prediction is crucial if the prediction is to be used in a decision making situation [RS03]. For ensembles the variety (or ambiguity) of the individual solutions is called the ensemble spread and in a way represents the forecast uncertainty [Zhu05]. The spread is mathematically defined as the standard deviation of the ensemble members from the ensemble mean [Zhu04]. A perfect ensemble should have a spread equal to the ensemble mean error (or high correlation between the ensemble spread and the ensemble mean error) [Zhu05], but there is no exact translation of what a certain spread will correspond to in uncertainty. This depends on the size of the ensemble error and the ensemble spread.

Since diversity is related to ensemble uncertainty, it could be used to rank the confidence in predictions made for instances the test set. If the ensemble members disagree on a certain instance, this should be interpreted as a higher uncertainty for the ensemble prediction on that instance.
5.6 Rule extraction

The most powerful predictive techniques such as ANN or ensembles of different models are hard or impossible to understand for humans, i.e. they are opaque. Transparent models like decision trees are easily understood but often have an inferior accuracy. This becomes a problem when the task requires an accurate transparent model. As described above, transparency or comprehensibility could be motivated by legal reasons, safety issues [ADT95], user acceptance, [CS96] or the necessity of making rational forecast adjustments [Goo02]. The of using an opaque model with high accuracy or an inferior transparent model is often called the accuracy vs. comprehensibility tradeoff. One approach to overcome this tradeoff is rule extraction; i.e. the process of transforming an accurate opaque model to a transparent model while retaining high accuracy.

![Figure 12 - Rule extraction](image)

A totally different motivation for rule extraction could be that an opaque model exists and an explanation for its behavior is needed.

A decision maker can choose if he wants to keep the prediction from the opaque model and only use the extracted model as an explanation or if he wants to make new predictions with the extracted model. In general, the opaque model will still have slightly higher accuracy but using its prediction will result in a less transparent solution as the extracted model only will explain the prediction to a certain degree. For this reason the extracted model is also most often used for making the predictions as the extracted model should have retained a high accuracy and that the rule extraction was motivated by a need of comprehensibility in the first case.
Rule extraction is a quite mature field where much research has been performed and several different methods have been proposed. To facilitate comparison of rule extraction techniques and to set guidelines for further research [CS99] suggested five criteria that should be fulfilled by good rule extraction algorithms listed below. These criteria were designed for rule extraction from ANNs but only need small adjustments to be applicable to arbitrary opaque model.

- **Comprehensibility**: The extent to which extracted representations are comprehensible for humans.
- **Fidelity**: The extent to which extracted representations accurately model the networks from which they were extracted.
- **Accuracy**: The ability of extracted representations to make accurate predictions on previously unseen cases.
- **Scalability**: The ability of the model to scale, e.g., ANNs with large input spaces and large numbers of units and weighted connections.
- **Generality**: The extent to which the method requires special training regimes or restrictions on network architecture.

Some authors add the criterion *consistency*; see e.g. [TS93]. A rule extraction algorithm is consistent if rules extracted for a specific problem are similar between runs; i.e. consistency measures how much extracted rule sets vary between different runs. The motivation is that if extracted rule sets differ greatly, it becomes hard to put a lot of faith in one specific rule set.

There are two basic categories of rule extraction techniques *decompositional* and *pedagogical*, and a third *eclectic* that combines the two basic approaches [ADT95]. Decompositional techniques try to extract rules by analyzing the internal structure of the opaque model (usually the hidden and output units of an ANN). An example of a decompositional technique is RULEX, for details see [AG95]. Pedagogical techniques treat the model as a black box and thus are applicable for all kinds of models. Pedagogical techniques convert the rule extraction problem to a learning task from the original inputs to the models outputs, i.e. the original target variable is replaced by the predicted value. Two examples of pedagogical techniques are VIA[Thr93] and TREPAN [CS96]. Most pedagogical techniques could of course be applied directly on the original data set as a normal classifier. However the underlying idea for this type of rule extraction is that it is assumed that the opaque model contributes in the rule creation process by cleaning the original data set from noise. A data set with less noise should be a better foundation for rule induction.
When rule extracting from ensembles, only pedagogical rule extraction meets the
generality criteria; i.e. it would be impossible to create a decompositional rule
extraction technique general enough to work on an arbitrary ensemble. As
ensembles in general are more accurate and robust than single techniques only
pedagogical rule extraction techniques will be considered in this research.

5.6.1 Rule Size
If the guidelines given by [CS99] are to be followed it, is not enough that the
extracted model is transparent, it also needs to be comprehensible. A transparent
model is a model that can be read by a human and understood in its parts. A
comprehensible model on the other hand needs to be understandable as a whole
which is another matter. Figure 14 shows a Boolean rule written in prefix
notation that is transparent but not comprehensible. It does not help that each
condition can be understood when there are too many conditions. There is no a
hard line deciding whether a rule is comprehensible or not, but when comparing
two rules with the same representation, the simpler (smaller) one is most often
considered more comprehensible. This is also in accordance to the old theorem
called *Occam’s razor* (other things being equal smaller theories are preferable to
larger ones) and the minimum description principle (MDL)[Ris78].

Two rules created with the same representation can be compared in a
straightforward way by simply counting the number of elements each rule
contains. If both rules have the same accuracy the one with fewer elements should
be chosen.
Figure 14 - Transparent but not comprehensible model
6 Previous work - Genetic Rule Extraction (G-REX)

G-REX (Genetic Rule EXtraction) is a previously suggested rule extraction technique first presented in [JKN03]. G-REX is in its foundation a GP framework design to induce prediction rules for data mining tasks. As a rule extraction technique, G-REX follows the methodology of pedagogical rule extraction algorithms closely.

Several distinctive features in G-REX make it very suitable as a rule extraction algorithm. G-REX main strength is that it was especially designed to bridge the accuracy versus comprehensibility gap and to facilitate tailoring of the rule representations to the problem at hand. G-REX has been shown to fulfill the rule extraction criteria comprehensibility, fidelity, accuracy, generality, consistency and to some small extent scalability.

6.1 Fitness Functions

According to Occam’s razor and common sense, a good prediction rule should be accurate and as simple as possible. This idea is translated straight into the G-REX fitness function for classification, by giving a reward for each correct prediction a rule makes and a small punishment for each element it contains as seen in (28). A parameter \( p \) is used to tune how much shorter rules should be favored. During evolution, this will mean that each element in a rule will be weighed against how much accuracy it adds to the rule. \( E(i_k) \) is the ensemble prediction for instance \( i_k \) and \( R, (i_k) \) is the prediction made for the same instance by the extracted rule. \( size_r \) is the number of elements, (operators, variables and constants), used in the rule.

\[
\text{fitness}_r = \sum_{k=1}^{n} (E(i_k) == R, (i_k)) - size_r * p
\]  

(28)

For estimation tasks, G-REX uses a fitness function based on the total absolute error made by the rule. To produce short comprehensible rules, G-REX also uses a punishment for longer rules. The length punishment is calculated in the same way as for classification tasks and the absolute error is negated as a higher fitness value usually corresponds to a fitter individual.

\[
\text{regfitness}_r, = -\sum_{k=1}^{k_{ua}} |E(i_k) - R, (i_k)| - size_r * p
\]  

(29)
6.2 Representation Languages

The basic classification model used by G-REX is a typical decision tree with binary splits based on a single condition, as defined in Table 7. Optionally, G-REX also allows complex conditions where several variables can be combined using conjunctions and disjunctions; for a BNF see Table 8. Figure 15 shows a sample complex G-REX rule, where boxes with thick borders signify class predictions.

<table>
<thead>
<tr>
<th>Function set</th>
<th>If</th>
<th>&lt;</th>
<th>&gt;</th>
<th>==</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal set</td>
<td>ConV</td>
<td>CatV</td>
<td>Double</td>
<td>Category</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>If</th>
<th>BinaryOP</th>
<th>Condition</th>
<th>then</th>
<th>If</th>
<th>Class</th>
<th>else</th>
<th>if</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinaryOp</td>
<td>BinaryOP</td>
<td>Condition</td>
<td>AND</td>
<td>OR</td>
<td>BinaryOp</td>
<td>Condition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Condition</td>
<td>ConV</td>
<td>&gt;</td>
<td>&lt;</td>
<td>Double</td>
<td>ConV</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CatV</td>
<td>==</td>
<td>Category</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| ConV            | Continuous variable |
| CatV            | Categorical variable |
| Double          | Double in range of associated ConV |
| Category        | Category of associated CatV |
| Class           | Class of target variable |

Table 8 - BNF for the Complex-IF representation language

Figure 15 - Complex-IF rule evolved by G-REX

Similar to most regression trees, a G-REX regression model consists of a normal decision tree with constant values as leaves. This model can also use complex conditions and optionally allows comparisons between attributes. The G-REX representation language for regression is called IF-REG and is presented in Table 9. Figure 16 is an example rule IF-REG rule evolved by G-REX.
### Table 9 - BNF for the IF-REG representation language

<table>
<thead>
<tr>
<th>Function set</th>
<th>If</th>
<th>&lt;</th>
<th>&gt;</th>
<th>==</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal set</td>
<td>ConV</td>
<td>CatV</td>
<td>Double</td>
<td>Category</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Condition</th>
<th>then</th>
<th>If</th>
<th>Prediction</th>
<th>else</th>
<th>If</th>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConV</td>
<td></td>
<td>&gt;</td>
<td>&lt;</td>
<td></td>
<td>Double</td>
<td>ConV</td>
</tr>
<tr>
<td>CatV</td>
<td></td>
<td>==</td>
<td>Category</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **ConV**: Continuous variable
- **CatV**: Categorical variable
- **Double**: Double in range of associated ConV
- **Category**: Category of associated CatV

**Double in range of target variable.**

![IF-REG rule evolved by G-REX](image)

6.3 **Empirical Results**

For classification tasks, G-REX has been evaluated thoroughly in several studies e.g. [JKN03], [JSK+03], [JSN04], [JNK04], [JKN04] and [JKN05]. The opaque models used as targets for rule extraction have most often been ensembles of neural networks, but have also included ensembles made of a mix of different models. Overall, these studies show that G-REX consistently achieves a higher or comparable accuracy compared to standard decision trees algorithms like CART, C4.5 and Tree Fit as implemented in MatLab R2007\(^1\). G-REX also outperformed another popular rule extraction algorithm called TREPAN. However the main result was not that G-REX was slightly more accurate than these algorithms but that it outperformed them with significantly more comprehensible (smaller) rules.

Although the rules produced by G-REX are often rather compact, they still contain unnecessary elements on occasions. These elements can be parts of a rule that is never executed (introns) or simply logical nonsense. [Koz92] argues that introns are an important part of genetic programming, since they help to protect valuable parts from being destroyed by the crossover operation. While introns are

\(^1\) www.mathworks.com
necessary in the evolutionary process they add meaningless complexity to the final rule.

G-REX has not been thoroughly evaluated for estimation tasks but initial experiments demonstrating the concept were performed with promising results in [JKN04]. In that study G-REX used a simple representation language with constant values as terminals.
7 Motivation of Case Studies

This chapter will relate the cases studies in chapter 9 to the thesis objectives based on theory and related work. Table 10 summarizes this chapter by briefly describing how each study contributes to the thesis objectives.

1. **Implement and evaluate novel components in a rule extraction framework for improved comprehensibility.**

   Only pedagogical techniques should be considered when extracting rules from ensembles since they are the only kind that is general enough to deal with an ensemble of arbitrary members. VIA, TREPAN and G-REX are three pedagogical techniques that could be considered. However, G-REX has previously been shown to outperform TREPAN on accuracy and especially on comprehensibility. Finally the source code for G-REX is available and can easily be extended to evaluate new components. Rules extraction techniques based on GP, (like G-REX), sometimes extract rules that contain unnecessary parts such as introns. If rules could be simplified by removing unnecessary parts it would further improve the comprehensibility of the extracted rules. Since G-REX is designed to extract rules with arbitrary representations a simplification technique also needs to be able to simplify arbitrary representations. Study 9.1 presents and evaluates a novel technique to handle this problem.

   Previous studies of G-REX have focused on classification tasks. Only a few experiments have been performed on estimation tasks and then using a simple standard representation applied to a single domain. More complex representations could possibly improve the accuracy but could also make the rules less comprehensible. Hence study 9.5 implements and evaluates different G-REX representations languages for a wide range of estimation tasks.

2. **Implement and evaluate novel techniques for estimating uncertainty in predictions.**

   Since G-REX is based on GP, it is inherently inconsistent, i.e. different runs can often produce slightly different rules with similar accuracy. Some authors consider inconsistency as a serious disadvantage for rule extraction techniques. Study 9.4 however, presents a novel technique that uses inconsistency to improve the probability estimates of extracted rules.

   When using ensembles for regression, uncertainty estimations are usually based on the ensemble spread, i.e. the standard deviation of the ensemble members from the ensemble mean. However, much work on ensembles has been done previously, especially in the field of weather forecasting. A typical
component of weather forecasting techniques, which differs from normal techniques, is the use of statistical databases to calibrate estimations. Study 9.3 investigates how a successful weather forecasting technique can be adapted and used for domains lacking extensive statistical data.

3. Evaluating use of alternative score functions and metrics in predictive modeling

Normally predictive techniques use a score function that optimizes a predefined performance metric. Obviously, it would be better to be able to choose which metric that should be optimized, according to the problem and situation at hand. Study 9.2 shows how G-REX can be used to optimize alternative performance metrics and that this could improve the quality of the extracted rules. Study 9.2 further demonstrates G-REX flexibility evaluating special score functions for estimation tasks.

| Case study 9.1 | Presents a novel technique for increasing comprehensibility of extracted rules. | Demonstrates how optimization of alternative performance metrics can improve rule quality. |
| Case study 9.2 | Improved ensemble uncertainty estimation, using a modification of a weather forecasting technique. | |
| Case study 9.3 | Using rule extraction inconsistency to improve the uncertainty estimates of extracted models. | |
| Case study 9.4 | Demonstrates how adaptation of the representation language can improve both accuracy and comprehensibility. | Evaluation of score functions for estimation tasks, using suitable performance metrics. |

Table 10 - Contribution of case studies
8 Data sets Used for Evaluation

The experiments in this study are performed on data sets from a wide range of domains. All data sets are publicly available from the UCI Repository [BM98] and are frequently used to evaluate machine learning algorithms. The motivation for using these benchmark data sets is that they are all gathered from real domains and therefore will contain problems that a DSS may face in the real world. Below each data set used in the experiments will be described briefly.

8.1 Classification Data sets

- **Bupa liver disorder (BLD).** The problem is to predict whether or not a male patient has a liver disorder based on blood tests and alcohol consumption. The first five attributes are all blood tests which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption and the sixth is the number of alcoholic beverages drunk per day. Each instance constitutes the record of a single male individual.

- **Cleveland heart disease (CHD).** Prediction of whether a patient has a heart disease or not. The attributes includes age, gender and measurements of eight relevant characteristics of the patient’s heart.

- **Contraceptive method choice (CMC).** This data set is a subset of the 1987 National Indonesia Contraceptive Prevalence Survey. The samples are married women who were either not pregnant or did not know if they were at the time of interview. The problem is to predict the current contraceptive method choice (no use, long-term methods, or short-term methods) of a woman based on her demographic and socio-economic characteristics.

- **Credit card application (CRA).** All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data.

- **Congressional Voting Records Database (CVR).** This data set includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the CQA. The task is to classify if a congressman is a democrat or a republican depending on these votes.

- **Final settlements (FSL).** Final settlements in collective agreements reached in the business and personal services sector for organizations with at least 500 members (teachers, nurses, university staff, police, etc) in Canada in 1987 and first quarter of 1988.
• **German Credit data** (GCD). Data related to 1000 German credit card applications which among other things contains information about credit history, purpose of credit, amount, employment, age, etc.

• **Glass Identification Database** (GLA). This study was motivated by criminological investigation and concerns classification of glass using the proportions of the ingredients.

• **Horse Colic database** (HCD). Retrospective study concerning if the lesion for cases of horse colic was surgical or not. Attributes contain the age of the horses, data from different medical diagnose, at which clinic they were treated and if the horse survived.

• **Hepatitis Domain** (HED). Prediction of the survival of patients with hepatitis depending on treatment and disease related measurements.

• **Iris Plants Database** (IPD). The data set contains 3 classes of 50 instances each where each class refers to a type of iris plant. One class is linearly separable from the other two but are not linearly separable from each other.

• **Johns Hopkins Ionosphere** (JHI). This radar data was collected by a system in Goose Bay, Labrador. This system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. The idea is to predict good radar returns; i.e. those showing evidence of some type of structure in the ionosphere.

• **Lymphography Domain** (LYD). Detection of cancer using x-ray of lymph nodes and lymphatic vessels made visible by the injection of a special dye. This lymphography domain was obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia.

• **Pima Indians Diabetes Database** (PID). Prediction of whether or not a patient shows signs of diabetes according to the World Health Organization criteria. All patients are females at least 21 years old of Pima indian heritage.

• **Protein Localization Sites** (PLS). A biological data set concerning classification of the localization site of a protein. Attributes consist of measurements of cytoplasm, persiplasm and six membrane properties.

• **StatLog (Heart Disease)** (SHD). Similar to the Cleveland heart disease data set but in a slightly different form.

• **Sonar, Mines vs. Rocks** (SMR). This data set contains instances obtained by bouncing sonar signals off a metal cylinder at various angles and under various conditions. The transmitted sonar signal is a frequency-modulated
chirp, rising in frequency. Each attribute represents the energy within a particular frequency band, integrated over a certain period of time.

- **Thyroid Disease (THD).** The task for this data set is to diagnose whether a patient is sick with hyperthyroid or not.
- **Wisconsin Breast Cancer (WBC).** Prediction of whether a detected case of breast cancer is malignant or benign.
- **Waveform Database Generator (Version 2) Data set (WDG).** This is an artificial three-class problem based on three waveforms.
- **Vehicle Silhouettes (VES).** The problem is to classify a given silhouette as one of four types of vehicle, using a set of features extracted from the silhouette. The vehicle may be viewed from one of many different angles.
- **Wine (WIN).** Recognition of wine based on chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.
- **Zoo (ZOO).** A database containing 17 Boolean-valued animal attributes. The "type" to be predicted is: mammal, bird, reptile, fish, amphibian, insect or invertebrate.

For a summary of the characteristics of the data sets see Table 11. *Size* is the number of instances in the data set. *Classes* are the number of output classes in the data set. *Num.* is the number of numeric attributes, *Nom.* is the number of nominal attributes and *Naïve* is the proportion of the majority class, i.e. the accuracy achieved by only predicting the majority class. *Naïve* is presented a reference point for each data set and shows the worst possible performance that a reasonable classifier should achieve.
<table>
<thead>
<tr>
<th>Data set</th>
<th>Size</th>
<th>Classes</th>
<th>Num.</th>
<th>Nom.</th>
<th>Naïve</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLD</td>
<td>345</td>
<td>2</td>
<td>6</td>
<td>0</td>
<td>58%</td>
</tr>
<tr>
<td>CHD</td>
<td>303</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>54%</td>
</tr>
<tr>
<td>CMC</td>
<td>1473</td>
<td>3</td>
<td>2</td>
<td>7</td>
<td>43%</td>
</tr>
<tr>
<td>CRA</td>
<td>690</td>
<td>2</td>
<td>6</td>
<td>9</td>
<td>56%</td>
</tr>
<tr>
<td>CVR</td>
<td>435</td>
<td>2</td>
<td>0</td>
<td>16</td>
<td>61%</td>
</tr>
<tr>
<td>FSL</td>
<td>57</td>
<td>2</td>
<td>8</td>
<td>8</td>
<td>65%</td>
</tr>
<tr>
<td>GCD</td>
<td>1000</td>
<td>2</td>
<td>7</td>
<td>13</td>
<td>70%</td>
</tr>
<tr>
<td>GLA</td>
<td>214</td>
<td>7</td>
<td>9</td>
<td>0</td>
<td>36%</td>
</tr>
<tr>
<td>HCD</td>
<td>368</td>
<td>2</td>
<td>7</td>
<td>15</td>
<td>63%</td>
</tr>
<tr>
<td>HED</td>
<td>155</td>
<td>2</td>
<td>6</td>
<td>13</td>
<td>79%</td>
</tr>
<tr>
<td>IPD</td>
<td>150</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>33%</td>
</tr>
<tr>
<td>JHI</td>
<td>351</td>
<td>2</td>
<td>34</td>
<td>0</td>
<td>64%</td>
</tr>
<tr>
<td>PID</td>
<td>768</td>
<td>2</td>
<td>8</td>
<td>0</td>
<td>65%</td>
</tr>
<tr>
<td>PLS</td>
<td>336</td>
<td>8</td>
<td>7</td>
<td>0</td>
<td>43%</td>
</tr>
<tr>
<td>SHD</td>
<td>270</td>
<td>2</td>
<td>5</td>
<td>8</td>
<td>56%</td>
</tr>
<tr>
<td>SMR</td>
<td>208</td>
<td>2</td>
<td>60</td>
<td>0</td>
<td>53%</td>
</tr>
<tr>
<td>THD</td>
<td>151</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>92%</td>
</tr>
<tr>
<td>WBC</td>
<td>699</td>
<td>2</td>
<td>9</td>
<td>0</td>
<td>66%</td>
</tr>
<tr>
<td>WDG</td>
<td>5000</td>
<td>3</td>
<td>21</td>
<td>0</td>
<td>34%</td>
</tr>
<tr>
<td>VES</td>
<td>846</td>
<td>4</td>
<td>18</td>
<td>0</td>
<td>26%</td>
</tr>
<tr>
<td>WIN</td>
<td>178</td>
<td>3</td>
<td>13</td>
<td>0</td>
<td>40%</td>
</tr>
<tr>
<td>ZOO</td>
<td>101</td>
<td>7</td>
<td>0</td>
<td>17</td>
<td>41%</td>
</tr>
</tbody>
</table>

Table 11 - Characteristics of classification data sets

8.2 Estimation Data sets

- **Auto Price (AUT).** The data set consists of car specifications from which the price of the cars should be predicted. There are three types of attributes that describe an instance: car specifications, assigned insurance risk rating and its value losse compared to other cars.

- **Boston Housing Data (BHD).** The inputs variables are statistics related to the living standard in the different suburbs of Boston. The target variable is the median value of owner-occupied homes in a certain suburb.

- **Computer Hardware (COH).** The problem concerns predicting the relative CPU performance based on six CPU properties such as cycle time and memory size, etc.
• **Diabetes Numeric (DIN).** The objective is to investigate the dependence of the level of serum C-peptide on the various other factors in order to understand the patterns of residual insulin secretion. The response measurement is the logarithm of C-peptide concentration (pmol/ml) at the diagnosis, and the predictor measurements age and base deficit, a measure of acidity.

• **Pharynx (PHA).** A data set consisting of patients with squamous carcinoma of 3 sites in the mouth and throat (in the oropharynx). The objective of the study was to compare the two treatment policies with respect to patient survival time.

• **Sleep (SLE).** Includes brain and body weight, life span, gestation time, predation and danger indices for 62 mammals. The target is the total time the mammal spends sleeping.

• **Veteran's Admin. Lung Cancer Trial (VET).** The data set contains data about veterans with lung cancer. The input variables consist of information about the patients, the type of cancer and the treatment. The target variable is the patient’s survival time.

• **Wisconsin (WIS).** Each record in the data set represents follow-up data for a breast cancer study. The target is the recurrence time of the cancer and the input variables are measures for the cell nucleus.

Table 12 shows the general properties of the data sets with the same abbreviations that were used for the classification data sets. The mean value (Mean), the standard deviation (Std), the maximum value (Max), the minimum value (Min) and the number of outliers (Outl.) are all calculated for the target variable.

<table>
<thead>
<tr>
<th>Size</th>
<th>Num.</th>
<th>Nom.</th>
<th>Mean</th>
<th>Std</th>
<th>Max</th>
<th>Min</th>
<th>Outl</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUT</td>
<td>159</td>
<td>14</td>
<td>1</td>
<td>11446</td>
<td>5878</td>
<td>35056</td>
<td>5118</td>
</tr>
<tr>
<td>BHD</td>
<td>506</td>
<td>13</td>
<td>1</td>
<td>22.5</td>
<td>9.2</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>DIA</td>
<td>43</td>
<td>2</td>
<td>0</td>
<td>4.8</td>
<td>0.72</td>
<td>6.6</td>
<td>3</td>
</tr>
<tr>
<td>MAC</td>
<td>209</td>
<td>6</td>
<td>0</td>
<td>105.6</td>
<td>161</td>
<td>1150</td>
<td>6</td>
</tr>
<tr>
<td>PHA</td>
<td>195</td>
<td>2</td>
<td>10</td>
<td>555.5</td>
<td>422</td>
<td>1823</td>
<td>0</td>
</tr>
<tr>
<td>SLE</td>
<td>62</td>
<td>7</td>
<td>0</td>
<td>8.5</td>
<td>5.8</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>VET</td>
<td>137</td>
<td>3</td>
<td>4</td>
<td>121.6</td>
<td>158</td>
<td>999</td>
<td>1</td>
</tr>
<tr>
<td>WIS</td>
<td>194</td>
<td>32</td>
<td>0</td>
<td>46.9</td>
<td>34.5</td>
<td>125</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 12 - Characteristics of estimation data sets
9 Case Studies

9.1 Increasing Rule Extraction Comprehensibility
This chapter is based on the paper “Increasing rule extraction comprehensibility”, [KJN06]. Note that some G-REX settings have been converted fit the latest version of G-REX.

9.1.1 Summary
In this chapter an extension to the rule extraction algorithm G-REX is suggested and evaluated. As noted in chapter 6, rules produced by G-REX sometimes contain introns that add meaningless complexity to the extracted rule. In this study a novel extension to G-REX, that simultaneously removes both introns and any other unnecessary code elements from the extracted rules, is suggested.

When trying to simplify an extracted rule, G-REX starts by replacing the original target variable with the corresponding prediction from the extracted rule. This new data set is then used by G-REX to evolve a new population of rules. Here, however, the fitness function is altered. A rule that fails to classify all training examples exactly as the initial rule is given a very large penalty, making it extremely unlikely that a faulty rule is going to survive to even the next generation. Another, much smaller, penalty is used to encourage shorter rules. This fitness function all but guarantees that this second round of G-REX will produce a shorter rule. To improve performance the rules in the initial population is created by mutating the extracted rule. Experiments on 16 data sets from the UCI repository shows that the novel simplification feature of G-REX succeeded in simplifying an extracted rule by removing unnecessary parts. As a matter of fact, even rather short rules could often be further simplified.

Hence, the contribution of this study is a novel G-REX component that increases the comprehensibility of extracted rules. It should be noted that the proposed method obviously can be used on a rule set with arbitrary syntax since it uses exactly the same procedure as the original rule extraction.

9.1.2 Background
G-REX has been evaluated in several studies [JKN03], [JSK+03], [JNK04], [JKN04], [JKN05] and [JLK+06] which all shows that G-REX can extract short accurate rules. The produced rules are significantly shorter compared to standard transparent techniques like CART and C5.0, while having a higher or comparable accuracy. However, as mentioned in chapter 6, rules produced by G-REX sometimes contain introns. Even if it is argued in [Koz92] that introns are an
important part of genetic programming since they help to protect valuable parts, they add meaningless complexity to the final rule. Obviously it would be possible to (automatically or manually) remove introns and logical nonsense from the extracted rule after evolution has finished. A more subtle issue is, however, the fact that rules often include perfectly logical and semantically correct parts that still do not affect the predictions from the extracted model. In this study, a novel extension to G-REX is suggested, that simultaneously removes both kinds of unnecessary code elements in the extracted rules.

9.1.3 Method

Two main experiments are performed to evaluate two different approaches to eliminate introns. In the first experiment, an approach that tries to simplify the final extracted rule by removing introns is evaluated. The second experiment investigates if a continuous fitness function will hamper the creation of introns.

As G-REX is a rule extraction algorithm, an opaque model is needed as the target of the rule extraction. This study uses an ensemble consisting of models created with five different data mining techniques; CART [BFO+83], two different multi-layer perceptron (MLP) neural networks [RM86], a radial basis function (RBF) neural network [BL88], and a probabilistic neural network (PNN) [Spe90].

Simplification

G-REX standard fitness function is presented in chapter 6 in (28) where size is the number of elements in the rule and $p$ is a parameter that controls the amount of punishment a rule gets according to its size. The number of the introns that can appear in a rule is in a way related to the value of parameter $p$. A low $p$ value allows bigger rules (and more introns) since the rules only get a relative small punishment compared to the reward of a correct prediction.

Introns can be removed automatically by simply noting which parts of the rule that are actually used when the rule is evaluated on the training set and then removing the unused parts. Logical nonsense is harder to remove as there are
many different constellations that would have to be handled and it would be almost impossible to define all possible cases beforehand. However, in this study, it is argued that G-REX could simplify the extracted rule automatically by performing a second evolutionary process, with the aim of removing both introns and logical nonsense. In this evolution, the fitness function should reward rules that are shorter than the extracted rule but still makes the same predictions. To facilitate the use of a fitness function with these properties, a copy of the data set used for the rule extraction is created with the exception that the predictions of the opaque model are replaced with the predictions of the extracted rule. This new data set is then used in combination with a low value of $p$ (0.01) to evolve a new simplified rule.

As the fitness function used during simplification (Equation (30)) has very small size punishment, fidelity towards the extracted rule is the highest priority. $R(i_k)$ is the prediction made by the extracted rule $R$ and $S_r(i_k)$ is the prediction made by the simplified rule.

$$\text{simplificationFitness}_r = \sum_{k=1}^{k=n} (R(i_k) \Rightarrow S_r(i_k)) - \text{size}_r \times 0.01 \quad (30)$$
When using the simplification fitness function, the size punishment is in reality only effective when comparing rules with the same fidelity. To improve the performance and speed of the simplification process, the initial population is created using the extracted rule. More specifically; the population is divided into five equally sized parts, each subjected to a different level of mutation. One part starts with exact copies of the original rule, while the remaining parts start with from one to four mutation operations on each individual. In practice this all but guarantees that the outcome of the simplification will be 100% faithful to the extracted rule.

**Continuous fitness**

A different approach is to try to reduce the number of introns present in the evolved population. If fewer introns are present during the evolution, the extracted rule would of course contain less introns. The standard fitness function for G-REX, presented in (28) above, the parameter $p$ was used to control the size of the rules. However, when G-REX rules are analyzed more closely they still often contain some introns. A solution to this problem could be to use a fitness function with a more direct connection between the fidelity reward and the size penalty. Papagelis and Kalles suggested such a continuous penalty for a fitness function in [PK01]; see (31):

$$fitness_r = \sum_{k=1}^{n} (E(i_k) \Rightarrow R_r(i_k))^2 \frac{x}{size_r^2 + x}$$

(31)

where $x$ is an arbitrarily, large number. A smaller $x$ means a bias towards smaller rules while a larger $x$ prioritizes accuracy. In the study, Papagelis and Kalles used $x$-values of 1000 and 10000. Here, this penalty function is evaluated again when used in G-REX.

**9.1.4 Experiments**

In the following chapter, the outline for the experiments is described in detail. 16 publicly available data sets from the UCI repository [BM98] were used for evaluation. For a description of the data sets see chapter 8.

First each data set is randomly divided into five partitions. Each partition is divided into 75% training instances and 25% test instances. G-REX is executed in a batch of ten consecutive runs and the rule with highest fidelity towards the ensemble is selected as the final rule. Three different fitness functions were tested, all rewarding high fidelity towards the ensemble but using different size punishments. Two of the fitness functions, Small and Large, used G-REX’ normal fitness function, see equation (28). *Small* was defined to produce very short rules
and hence had $p = 0.33$, Large had $p = 0.0125$ to allow longer rules. A third fitness function Cont. used the continuous penalty described in chapter 9.1.3 with the parameter $x$ set to 1000. Each fitness function was applied to each data set and the produced rules were simplified using the procedure described in chapter 9.1.3 above.

Comparing sizes between rules extracted by G-REX and CART is not entirely trivial. For problems with more than two classes, the representation is identical (see Table 7), but for binary problems, G-REX uses a slightly different representation. The difference is that G-REX evolves Boolean rules, (as defined in Table 6), while CART creates decision trees. Regarding the overall size, this has no major implication. As an example; a test like $X_1 < 5.2$ would require one node for CART but three for G-REX. On the other hand, the classes are implicit in the G-REX representation while they constitute the leaves in the CART tree. From this it was deemed accurate enough to measure the size for both G-REX and CART as the total number of nodes.

9.1.5 Results

Table 13 shows the results for the different fitness functions on each data set. Results for the ensemble are omitted as the focus is to compare comprehensible models. The ensemble clearly outperformed CART, seen over all data sets which is not surprising and according to theory. Note that the last row gives the mean result over all data sets. A Friedman test ($p=0.089$) shows that no technique is significant better than all others. Small has the highest mean accuracy and best rank, Cont. is clearly the weakest technique in regards of both accuracy and rank.

Table 14 shows the size of the original extracted rules and Table 15 shows the length of the same rules after simplification. When evaluating the original rules in terms of comprehensibility (size), it is obvious that Small and Cont clearly outperforms CART and Large. This is also confirmed by a Friedman test ($p=2.5607 \times 10^{-5}$) and that the critical rank difference for this experiment is 1.09. Both Small and Cont produce significantly smaller rules than CART and Large but there were no significance in the difference between Small and Cont.
<table>
<thead>
<tr>
<th>Data set</th>
<th>CART ACC Rank</th>
<th>Large ACC Rank</th>
<th>Small ACC Rank</th>
<th>Cont. ACC Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLD</td>
<td>62.1 2</td>
<td>61.6 3</td>
<td>62.5 1</td>
<td>61.4 4</td>
</tr>
<tr>
<td>CLE</td>
<td>76.3 1</td>
<td>74.4 3</td>
<td>76.1 2</td>
<td>72.8 4</td>
</tr>
<tr>
<td>CMC</td>
<td>49.1 4</td>
<td>49.7 2.5</td>
<td>50.5 1</td>
<td>49.7 2.5</td>
</tr>
<tr>
<td>CRA</td>
<td>80.2 4</td>
<td>85.7 2</td>
<td>87.0 1</td>
<td>85.0 3</td>
</tr>
<tr>
<td>CVR</td>
<td>95.6 3</td>
<td>96.1 1</td>
<td>95.8 2</td>
<td>95.4 4</td>
</tr>
<tr>
<td>GLA</td>
<td>65.1 1</td>
<td>59.6 3.5</td>
<td>61.1 2</td>
<td>59.6 3.5</td>
</tr>
<tr>
<td>IPD</td>
<td>91.5 4</td>
<td>94.7 1</td>
<td>92.1 3</td>
<td>92.6 2</td>
</tr>
<tr>
<td>JHI</td>
<td>89.9 2</td>
<td>90.5 1</td>
<td>89.5 3</td>
<td>88.9 4</td>
</tr>
<tr>
<td>PID</td>
<td>67.0 4</td>
<td>72.5 3</td>
<td>74.3 1</td>
<td>72.8 2</td>
</tr>
<tr>
<td>PLS</td>
<td>78.7 1</td>
<td>76.9 3</td>
<td>78.6 2</td>
<td>76.0 4</td>
</tr>
<tr>
<td>SHD</td>
<td>74.0 2</td>
<td>73.0 4</td>
<td>75.9 1</td>
<td>73.1 3</td>
</tr>
<tr>
<td>THD</td>
<td>99.5 1</td>
<td>96.0 3</td>
<td>96.2 2</td>
<td>95.0 4</td>
</tr>
<tr>
<td>WBC</td>
<td>94.1 4</td>
<td>96.4 2</td>
<td>96.4 2</td>
<td>96.4 2</td>
</tr>
<tr>
<td>WDG</td>
<td>69.2 1</td>
<td>65.7 4</td>
<td>68.3 2.5</td>
<td>68.3 2.5</td>
</tr>
<tr>
<td>VES</td>
<td>67.1 1</td>
<td>62.4 4</td>
<td>63.1 3</td>
<td>63.5 2</td>
</tr>
<tr>
<td>WIN</td>
<td>96.5 1</td>
<td>93.3 3.5</td>
<td>93.3 3.5</td>
<td>95.1 2</td>
</tr>
<tr>
<td>MEAN</td>
<td>78.5 2.25</td>
<td>78.0 2.72</td>
<td>78.8 2.00</td>
<td>77.9 3.03</td>
</tr>
</tbody>
</table>

Table 13 - Accuracy on individual data sets ($CD=1.09$)

<table>
<thead>
<tr>
<th>Data set</th>
<th>CART Size R</th>
<th>Large Size R</th>
<th>Small Size R</th>
<th>Cont. Size R</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLD</td>
<td>81.8 3</td>
<td>129 4</td>
<td>10.2 1.5</td>
<td>10.2 1.5</td>
</tr>
<tr>
<td>CLE</td>
<td>58.2 3</td>
<td>203 4</td>
<td>18.2 1</td>
<td>27.8 2</td>
</tr>
<tr>
<td>CMC</td>
<td>522.2 4</td>
<td>176 3</td>
<td>33.0 2</td>
<td>22.0 1</td>
</tr>
<tr>
<td>CRA</td>
<td>33.1 3</td>
<td>45 4</td>
<td>15.3 1</td>
<td>16.3 2</td>
</tr>
<tr>
<td>CVR</td>
<td>95.6 4</td>
<td>53 3</td>
<td>11.8 1.5</td>
<td>11.8 1.5</td>
</tr>
<tr>
<td>GLA</td>
<td>59.0 3</td>
<td>144 4</td>
<td>20.0 1</td>
<td>44.0 2</td>
</tr>
<tr>
<td>IPD</td>
<td>9.8 1</td>
<td>45 4</td>
<td>12.0 2</td>
<td>18.0 3</td>
</tr>
<tr>
<td>JHI</td>
<td>37.4 3</td>
<td>73 4</td>
<td>15.0 2</td>
<td>14.2 1</td>
</tr>
<tr>
<td>PID</td>
<td>156.6 3</td>
<td>192 4</td>
<td>22.2 2</td>
<td>19.8 1</td>
</tr>
<tr>
<td>PLS</td>
<td>51.0 3</td>
<td>183 4</td>
<td>25.0 1</td>
<td>42.0 2</td>
</tr>
<tr>
<td>SHD</td>
<td>45.8 3</td>
<td>96 4</td>
<td>14.2 1</td>
<td>19.0 2</td>
</tr>
<tr>
<td>THD</td>
<td>19.2 3</td>
<td>46 4</td>
<td>7.3 1</td>
<td>13.0 2</td>
</tr>
<tr>
<td>WBC</td>
<td>94.1 3</td>
<td>104 4</td>
<td>15.0 1</td>
<td>22.0 2</td>
</tr>
<tr>
<td>WDG</td>
<td>69.2 3</td>
<td>83 4</td>
<td>40.0 2</td>
<td>27.0 1</td>
</tr>
<tr>
<td>VES</td>
<td>67.1 3</td>
<td>202 4</td>
<td>53.0 2</td>
<td>52.0 1</td>
</tr>
<tr>
<td>WIN</td>
<td>96.5 3</td>
<td>123 4</td>
<td>20.0 1</td>
<td>32.0 2</td>
</tr>
<tr>
<td>MEAN</td>
<td>93.5 3.00</td>
<td>118.4 3.88</td>
<td>20.8 1.44</td>
<td>24.4 1.69</td>
</tr>
</tbody>
</table>

Table 14 - Size of extracted rules ($CD=1.09$)
The average size of the simplified rules in Table 15 is notably smaller for Small and Large when compared to the original rules. This proves that G-REX could remove large amounts of introns and unused parts since the simplified rules still had the same fidelity as the original rules. Even the rules produced by the fitness function Small, which supposedly produces the shortest rules could be further simplified. Rules extracted using Cont. however, could not be simplified at all and had exactly the same size as the extracted rules. A Friedman test yields $p = 3.2231 \times 10^{-6}$ which indicates that there are statistical differences in size between the techniques also after simplification. When compared using average ranks Small is significantly better than both Large and Cont. as the critical difference for this experiment is 0.792. Rules extracted with small were actually smaller than the simplified rules of large and Cont. but could still be made smaller when simplified.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Large</th>
<th>Small</th>
<th>Cont.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Size</td>
<td>R</td>
<td>Size</td>
</tr>
<tr>
<td>BLD</td>
<td>23.8</td>
<td>3</td>
<td>8.6</td>
</tr>
<tr>
<td>CLE</td>
<td>26.2</td>
<td>2</td>
<td>14.2</td>
</tr>
<tr>
<td>CMC</td>
<td>67.0</td>
<td>3</td>
<td>23.0</td>
</tr>
<tr>
<td>CRA</td>
<td>30.3</td>
<td>3</td>
<td>14.5</td>
</tr>
<tr>
<td>CVR</td>
<td>52.6</td>
<td>3</td>
<td>11.8</td>
</tr>
<tr>
<td>GLA</td>
<td>76.0</td>
<td>3</td>
<td>18.0</td>
</tr>
<tr>
<td>IPD</td>
<td>22.0</td>
<td>3</td>
<td>12.0</td>
</tr>
<tr>
<td>JHI</td>
<td>44.6</td>
<td>3</td>
<td>14.2</td>
</tr>
<tr>
<td>PID</td>
<td>65.4</td>
<td>3</td>
<td>15.8</td>
</tr>
<tr>
<td>PLG</td>
<td>63.0</td>
<td>3</td>
<td>17.0</td>
</tr>
<tr>
<td>SHG</td>
<td>38.2</td>
<td>3</td>
<td>11.0</td>
</tr>
<tr>
<td>THG</td>
<td>12.3</td>
<td>2</td>
<td>7.3</td>
</tr>
<tr>
<td>WBC</td>
<td>103.8</td>
<td>3</td>
<td>15.0</td>
</tr>
<tr>
<td>WDG</td>
<td>67.6</td>
<td>3</td>
<td>22.0</td>
</tr>
<tr>
<td>VES</td>
<td>90.0</td>
<td>3</td>
<td>32.0</td>
</tr>
<tr>
<td>WIN</td>
<td>51.0</td>
<td>3</td>
<td>17.0</td>
</tr>
<tr>
<td>MEAN</td>
<td>52.1</td>
<td>2.88</td>
<td>15.8</td>
</tr>
</tbody>
</table>

Table 15 - Rule size after simplification (CD=.792)
Figure 19 shows a rule extracted for the CLE data set using the *Small* fitness function. The size of the rule is 31 which is a bit longer than the average rule extracted with *Small*. It is not obvious that the rule contains introns as it seems logical when manually inspected. In reality, the grey elements are unnecessary for predicting the training instances. They could of course be used to predict new unseen instances, but as these elements lack support in the training set there is no rational motivation for including them in the rule.

After simplification of the rule in Figure 19, G-REX outputs the rule in Figure 20, which makes the exact same predictions for the training set. When the introns have been removed the rule has a size of 11 while having the same performance as the extracted rule.

9.1.6 Discussion

*Cont.* was clearly the least accurate technique, but there were only significant differences between the techniques in terms of size. Even if the rules produced by *Large* could be greatly simplified they were still bigger than rules produced by *Small* and *Cont*. This is probably due to the fact that the fitness function allows bigger rules with larger amount of introns. More complex rules, should be able to achieve a higher fidelity, but they also greatly increase the search space, making it
harder to find the best solution. Hence, as the number of generations was the same for all experiments, it is not surprising that Large is slightly worse in both fidelity and size.

Before simplification, Cont and Small performed comparable regarding both accuracy and size. Small was slightly better on both metrics, but the difference was not statistically significant. Using the simplification process, the smaller rules produced by Small could be made even smaller, while Cont’s rules could not be simplified at all. This is probably the result of Cont’s fitness function where every extra rule part results in a high penalty, making it crucial that every new part adds extra fidelity to the rule. Cont’s search for fit individuals seems to be less efficient than Small as it produces both longer and less accurate rules, possibly due to the fact that it hampers the creation of introns.

Compared to CART, even the original extracted rules were significantly shorter while having a better or comparable accuracy. It should be noted that the size of the CART trees could be limited to a certain size by controlling the allowed depth of the created tree. On the other hand, this parameter has to be set before execution and does not in any way balance accuracy against the size of the rule.

9.1.7 Conclusions
G-REX succeeded in simplifying extracted rules by removing unnecessary parts. As a matter of fact, even rather short rules could often be further simplified. It should be noted that this process obviously can be used on a rule set with arbitrary syntax since it uses exactly the same procedure as the original rule extraction.

Allowing larger rules had negative effect, in this case especially considering that even the simplified Large rules were longer than the original rules produced by the other techniques. If larger, more complex, rules are needed, care must be taken that the resulting larger search space is explored during a sufficient number of generations.

Cont and Small had comparable size and accuracy when used for rule extraction, but the simplification process greatly reduced the size of the Small rules. After simplification, the rules produced with Small were significantly smaller than the rules produced with Cont, which could not be simplified at all.

9.1.8 Considerations
Care has to be taken so that the number of generations matches the size of the search space. If comprehensibility is of importance, simplification should always be done immediately after rule extraction.
9.2 Using GP to increase Rule Quality
This chapter is based on the article “Using Genetic Programming to Increase Rule Quality”, [KJN08].

9.2.1 Summary
This study has two objectives, to evaluate two new fitness functions for G-REX and to show how G-REX can be used as a rule inducer. The fitness functions are designed to optimize two score functions containing alternative quality measures, area under ROC curves and new a metric designed for the recently suggested comprehensibility criteria brevity. Rules with good brevity classify typical instances with few and simple tests and use complex conditions only for atypical examples. Experiments using thirteen publicly available data sets show that the two novel fitness functions succeeded in increasing brevity and area under the ROC curve without sacrificing accuracy. When compared to a standard decision tree algorithm, G-REX achieved slightly higher accuracy, but also added additional quality to the rules by increasing their AUC or brevity significantly.

The main contribution of this study is that it demonstrates the advantage of the ability to optimize alternative performance metrics, and that that G-REX has this ability.

9.2.2 Background
As described in chapter 5.6 comprehensibility is an important criteria when evaluating rule extraction techniques. In most studies, comprehensibility is measured as the size of the extracted representation. This is an obvious simplification, chosen with the motivation that it makes comparisons between techniques fairly straightforward. In addition, size could be calculated in a number of ways. In [SJK07] it is argued that comprehensibility should be broken down to interpretability, i.e., that the model should express conditions in a way humans tend to use and brevity, that a model should classify as many instances as possible with few conditions. Even if brevity is well described in [SJK07] there is no definition of how brevity could be captured in a quantitative performance metric.

Most rule inducers producing comprehensible models only optimize accuracy. This is a fundamental design choice that is built into the algorithms, making it hard or impossible to change the optimization criteria. Intuitively, it would be an advantage to be able to optimize the criteria most important for the problem at hand, e.g., if a comprehensible model is required a comprehensibility metric should be a part of the optimized score function.
The main purpose of this study is to demonstrate the advantages of a rule inducer with the ability to optimize alternative score functions. Any GP based technique could be used for this purpose but here G-REX is used an adapted to work as a rule inducer. A subordinate purpose is to suggest a performance metric for the brevity criteria and show how it could be optimized using GP.

Method
In the experiments G-REX is used as a rule inducer to optimize two alternative performance measures, area under ROC-curve and the suggested brevity metric presented below. The only difference to the normal use of G-REX is that the target of the GP is the actual target value instead of predictions of an opaque model. To optimize Brevity and Area under ROC-curve two new fitness functions $BREfitness$ and $AUCfitness$ are defined for G-REX.

During experimentation, comparisons are made against both the standard G-REX fitness function $ACCfitness$, primarily optimizing accuracy and against MatLab’s decision tree algorithm Treefit. All three fitness functions are described in detail in the following sections.

ACCfitness
G-REXs original fitness function (equation(28) in 6.1) had a disadvantage in that the parameter $p$ used to control the size of the extracted rules had to be set individually for each data set. This problem occurred because the size punishment was related to the number of correctly classified instances. Two rules with the same accuracy would be affected differently by the size punishment depending on the size of each data set. When using G-REX as a rule inducer, this would have to be considered a drawback since the decision maker would have yet another parameter to tune. To counter this problem the fitness function was modified by dividing the reward from the number of correct classified instances with the total number of instance i.e. the accuracy of the rule, see(32).

$$ACCfitness, = \frac{\sum_{k=1}^{k=n} (E(i_k) == R(i_k))}{n} - \text{size, } * \ p \quad (32)$$

AUCfitness
Area under ROC-curve has been suggested to be a more appropriate metric than accuracy for comparing classification performance, see chapter 4.5.2. The main difference is that AUC is based on the evaluated model’s probability estimates for the predictions, while accuracy only considers the results of the actual
predictions. Since G-REX uses GP to evolve rules, it does not, just like decision trees, provide explicit probability estimates for the predictions. Hence when employing AUCFitness, G-REX uses Laplace estimates, (see chapter 5.2.3) for ranking the instances during the ROC analysis. Overall, AUCfitness balances the AUC and the length of the rule in the same way that ACCFitness balances accuracy and length.

\[
AUC_{fitness} = AUC - size \times p
\]  

**BREfitness**

The fitness function BREfitness is aimed at optimizing rule brevity. Instead of just using the length of the rules as a base for punishment, BREfitness evaluates how the rule is actually used when classifying instances. The idea is that a tree must not only be relatively small but also should classify as many instances as possible using only a few tests, i.e. the rule brevity should be high. The overall principle as defined in [SJK07] is that a typical instances should be classified using a few simple tests, while more complex rules only should be used for more atypical patterns. Although brevity could be measured using different metrics, the most natural should be calculate it as the average number of tests needed to be checked in order to classify the whole data set, i.e. the brevity for a rule set \( r \) when classifying the instances \( k=1..n \) is:

\[
BRE_r = \sum_{k=1}^{n} \frac{\# conditions_k}{\# instances}
\]  

\( \# conditions_k \) is the number of conditions that have to be checked to classify instance \( k \). If the counts in Figure 8 are calculated on test instances, the corresponding brevity is: \((59*1+3*2+15*2)/(59+3+15)=1.2\). Note that using this definition, brevity does not consider accuracy at all. A rule with the best (lowest) possible brevity (1.0) could classify all instances incorrectly. Brevity should therefore only be used to evaluate rules with comparable accuracy. Because of this, BREfitness functions uses accuracy as a reward and brevity as a punishment (a lower BRE means that the rule has better brevity). The size of the punishment is adjusted, again using a parameter \( p \).

\[
BRE_{fitness} = \frac{\sum_{k=1}^{n} \left(E(i_k) \Rightarrow R_r(i_k)\right)}{n} - BRE_r \times p
\]
**Treefit**
The *Treefit* algorithm in MatLab is a decision tree algorithm for regression and classification which is based on CART [BFO+83], see chapter 5.2.1. Treefit produces a binary tree with splits based on the input variables. For continuous variables the relational operators $<$ and $>$ are used. Categorical variables are split using \( \text{variable} = \text{category list} \), which is true if the current variable is one of the listed categories.

### 9.2.3 Experiments
This study uses 13 binary data sets from the UCI repository which are described in detail in chapter 8. Missing values for continuous variables are replaced with the mean value of all non missing values for the variable. Categorical missing values are, similarly, replaced with the mode value of the variable. When missing values have been handled, each data set is divided into ten folds which are stratified to ensure that each fold will have a representative class distribution. For evaluation, standard 10-fold cross validation is used.

**G-REX Setting**
In this study, G-REX executes in a batch of ten runs for each fold. The rule with the highest fitness is selected as the winner and is then evaluated on the test set. All fitness functions use $p=0.05$, which should produce short and comprehensible rules. G-REX used the GP settings presented in Table 16.

<table>
<thead>
<tr>
<th>Number of generations</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
<td>1000</td>
</tr>
<tr>
<td>Crossover probability</td>
<td>0.8</td>
</tr>
<tr>
<td>Mutation probability</td>
<td>0.001</td>
</tr>
<tr>
<td>Creation type</td>
<td>Ramped half and half</td>
</tr>
<tr>
<td>Maximum creation depth</td>
<td>6</td>
</tr>
<tr>
<td>Length punishment ($p$)</td>
<td>0.05</td>
</tr>
<tr>
<td>Batch size</td>
<td>10</td>
</tr>
</tbody>
</table>

*Table 16 - G-REX settings used in all experiments*

In the experiments the IF-ELSE representation, (see Table 7), was used with a small adjustment allowing comparisons between continuous variables in the conditions, see Table 17.
<table>
<thead>
<tr>
<th>Function set</th>
<th>If</th>
<th>&lt;</th>
<th>&gt;</th>
<th>==</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal set</td>
<td>ConV</td>
<td>CatV</td>
<td>Double</td>
<td>Category</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>If</th>
<th>Condition</th>
<th>then</th>
<th>If</th>
<th>Class</th>
<th>else</th>
<th>If</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConV</td>
<td>Continuous variable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CatV</td>
<td>Categorical variable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Double</td>
<td>Double in range of associated ConV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Category of associated CatV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td>Class of target variable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 17 - BNF for the Adjusted IF-ELSE representation language

**Evaluation**

Accuracy, AUC and brevity are calculated for all rules, regardless of technique and optimization criterion. The results are averaged over all folds and reported individually for each data set. Overall performance of the different techniques is compared using a Friedman-test the 0.05 significance level. The Nemenyi test is used to decide if the performance of two techniques differs significantly different.

**9.2.4 Results**

In the following chapter the results of the experiments are presented. Each table shows a different metric for the same rule set; i.e. accuracy, AUC, and brevity are calculated for one rule set per technique and data set, but the results are presented in three different tables. All results are averaged values for ten stratified folds. Next to the evaluated metric, the rank among the evaluated techniques is presented. Table 18 on the next page shows the achieved accuracy for the different techniques on each data set. As seen in the table, all techniques produce comparable results. A Friedman test confirms that the differences in accuracy is not statistical significant since it yields $p=0.210$.

Table 19 shows the AUC values for the same rule sets that were evaluated on accuracy in Table 18. AUCfitness clearly outperforms the other techniques using this metric, achieving the highest AUC on 9 of 13 data sets. The second best technique is ACCFitness followed by Treefit and BREFitness. A $p$-value of $8.06 \times 10^{-5}$ and a $CD$ value of 1.21 which shows that both AUCFitness and ACCFitness are significantly better than Treefit and BREFitness.
<table>
<thead>
<tr>
<th>Data set</th>
<th>Tree Fit</th>
<th>ACCEfit</th>
<th>BREfit</th>
<th>AUCfit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ACC R</td>
<td>ACC R</td>
<td>ACC R</td>
<td>ACC R</td>
</tr>
<tr>
<td>BLD</td>
<td>66.1 2</td>
<td>65.5 3</td>
<td>67.8 1</td>
<td>62.6 4</td>
</tr>
<tr>
<td>CHD</td>
<td>75.8 2</td>
<td>78.8 1</td>
<td>72.5 4</td>
<td>74.9 3</td>
</tr>
<tr>
<td>CRA</td>
<td>85.5 1.5</td>
<td>84.6 4</td>
<td>85.5 1.5</td>
<td>85.2 3</td>
</tr>
<tr>
<td>CVR</td>
<td>95.4 3</td>
<td>94.9 4</td>
<td>95.6 1.5</td>
<td>95.6 1.5</td>
</tr>
<tr>
<td>FSL</td>
<td>74.7 4</td>
<td>78.3 3</td>
<td>80.3 2</td>
<td>81.7 1</td>
</tr>
<tr>
<td>GCD</td>
<td>74.0 1</td>
<td>71.5 2</td>
<td>70.7 3.5</td>
<td>70.7 3.5</td>
</tr>
<tr>
<td>HCD</td>
<td>85.3 1</td>
<td>82.4 3</td>
<td>78.8 4</td>
<td>84.8 2</td>
</tr>
<tr>
<td>HED</td>
<td>76.9 4</td>
<td>81.9 1</td>
<td>81.4 2</td>
<td>78.7 3</td>
</tr>
<tr>
<td>JHI</td>
<td>89.7 2</td>
<td>93.2 1</td>
<td>88.0 3</td>
<td>87.7 4</td>
</tr>
<tr>
<td>PID</td>
<td>75.6 1</td>
<td>75.0 2</td>
<td>73.7 3.5</td>
<td>73.7 3.5</td>
</tr>
<tr>
<td>SHD</td>
<td>77.4 3</td>
<td>80.4 1.5</td>
<td>80.4 1.5</td>
<td>72.6 4</td>
</tr>
<tr>
<td>SMR</td>
<td>72.5 1</td>
<td>71.1 3</td>
<td>72.1 2</td>
<td>68.7 4</td>
</tr>
<tr>
<td>WBC</td>
<td>94.3 2.5</td>
<td>96.3 1</td>
<td>94.3 2.5</td>
<td>94.1 4</td>
</tr>
<tr>
<td>MEAN</td>
<td>80.2 2.15</td>
<td>81.1 2.27</td>
<td>80.1 2.46</td>
<td>79.3 3.11</td>
</tr>
</tbody>
</table>

Table 18 - Average ACC over 10-folds (CD=1.21)

<table>
<thead>
<tr>
<th>Data set</th>
<th>Treefit</th>
<th>ACCEfit</th>
<th>BREfit</th>
<th>AUCfit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AUC R</td>
<td>AUC R</td>
<td>AUC R</td>
<td>AUC R</td>
</tr>
<tr>
<td>BLD</td>
<td>64.2 3</td>
<td>65.3 1</td>
<td>65.2 2</td>
<td>64.0 4</td>
</tr>
<tr>
<td>CHD</td>
<td>78.0 3</td>
<td>83.7 2</td>
<td>72.4 4</td>
<td>83.8 1</td>
</tr>
<tr>
<td>CRA</td>
<td>86.2 3.5</td>
<td>89.1 2</td>
<td>86.2 3.5</td>
<td>92.2 1</td>
</tr>
<tr>
<td>CVR</td>
<td>95.0 4</td>
<td>96.8 2</td>
<td>95.9 3</td>
<td>97.7 1</td>
</tr>
<tr>
<td>FSL</td>
<td>74.0 4</td>
<td>80.6 2</td>
<td>78.1 3</td>
<td>83.3 1</td>
</tr>
<tr>
<td>GCD</td>
<td>71.9 1</td>
<td>66.9 3</td>
<td>53.4 4</td>
<td>70.5 2</td>
</tr>
<tr>
<td>HCD</td>
<td>85.3 2</td>
<td>81.9 3</td>
<td>77.7 4</td>
<td>88.3 1</td>
</tr>
<tr>
<td>HED</td>
<td>49.6 4</td>
<td>75.5 2</td>
<td>56.7 3</td>
<td>78.1 1</td>
</tr>
<tr>
<td>JHI</td>
<td>90.2 3</td>
<td>92.0 1</td>
<td>87.6 4</td>
<td>91.8 2</td>
</tr>
<tr>
<td>PID</td>
<td>72.0 3</td>
<td>72.4 2</td>
<td>68.1 4</td>
<td>79.8 1</td>
</tr>
<tr>
<td>SHD</td>
<td>77.1 3</td>
<td>84.4 1</td>
<td>76.2 4</td>
<td>83.4 2</td>
</tr>
<tr>
<td>SMR</td>
<td>72.5 4</td>
<td>74.5 2</td>
<td>73.7 3</td>
<td>79.1 1</td>
</tr>
<tr>
<td>WBC</td>
<td>93.3 4</td>
<td>96.2 2</td>
<td>94.7 3</td>
<td>97.2 1</td>
</tr>
<tr>
<td>MEAN</td>
<td>77.6 3.19</td>
<td>81.5 1.92</td>
<td>75.8 3.42</td>
<td>83.8 1.46</td>
</tr>
</tbody>
</table>

Table 19 - Average AUC over 10-folds (CD=1.21)
When evaluating the rule sets against brevity (Table 20), the results for Treefit is not reported as the representation language is slightly different.

<table>
<thead>
<tr>
<th>Data set</th>
<th>ACCfit BRE</th>
<th>AUCfit BRE</th>
<th>BREfit BRE</th>
<th>AUCfit R</th>
<th>BREfit R</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLD</td>
<td>2.48</td>
<td>1.58</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CHD</td>
<td>7.65</td>
<td>1.18</td>
<td>10.73</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CRA</td>
<td>6.56</td>
<td>3</td>
<td>7.03</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CVR</td>
<td>5.87</td>
<td>3.00</td>
<td>9.77</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FSL</td>
<td>7.67</td>
<td>2.23</td>
<td>6.54</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>GCD</td>
<td>6.6</td>
<td>1.06</td>
<td>12.55</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>HCD</td>
<td>7.51</td>
<td>3.09</td>
<td>9.61</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>HED</td>
<td>5.44</td>
<td>1.14</td>
<td>9.77</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>JHI</td>
<td>4.37</td>
<td>1.77</td>
<td>4.82</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>PID</td>
<td>2.04</td>
<td>1.11</td>
<td>3.95</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>SHD</td>
<td>2.91</td>
<td>1.89</td>
<td>3.69</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>SMR</td>
<td>3.06</td>
<td>1.62</td>
<td>3.84</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>WBC</td>
<td>3.04</td>
<td>1.54</td>
<td>2.83</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>MEAN</td>
<td>5.02</td>
<td>2.15</td>
<td>1.86</td>
<td>1.00</td>
<td>6.86</td>
</tr>
</tbody>
</table>

Table 20 - Average BRE over 10-folds ($CD=0.879$)

From Table 20 it is clear that $BRE_{fitness}$ produces rules with much lower brevity. It is worth noting that the same rule sets also performed rather well when evaluated using accuracy. It is natural that $BRE_{fitness}$ does not perform as well when evaluated against AUC, since it produces very short rule sets leading to quite rough probability estimations. A Friedman test yields $p=1.2278 \times 10^{-5}$ and a Nemenyi test gives a $CD=0.879$ which shows that $BRE_{fitness}$ produces rules with significantly higher brevity than the other techniques. Another interesting result is that $AUC_{fitness}$ is significantly worse than all other techniques when compared on brevity. This could probably be explained using the same argument as above; i.e. to achieve high AUC it is necessary to have good probability estimations, which can only be produced by more complex rule sets.

9.2.5 Conclusions

This study had two objectives, to evaluate two new fitness functions for the G-REX rule extraction algorithm and to show the advantages of using GP as a rule inducer.

Both fitness functions perform well as they retain the same accuracy as the original fitness function while optimizing yet another other criteria. $AUC_{Fitness}$ produces rules that have high accuracy and a higher AUC than $Treefit$ and the other evaluated fitness functions. $BRE_{Fitness}$ uses the proposed comprehensibility
metric brevity to create rules which classifies the typical instances with simple conditions and only uses complex condition for atypical instances. The fitness function outperforms the $AUCFitness$ and $ACCFitness$ regarding brevity without significant loss of accuracy.

The experiments show that when G-REX is used as a rule inducer it is a good alternative to Treefit and similar decision trees algorithms. G-REX achieves a level of accuracy that is comparable to Treefit, but adds additional quality to the rules by increasing their AUC or brevity. A decision maker could certainly benefit from this extra rule quality, especially by directing G-REX to produce rules optimized on the quality metric best suited for the problem at hand. Another strength of GP based rule inducers is that the representation language can be tailored for the problem and thereby possibly increase comprehensibility, accuracy or both.

Overall the experiments show that G-REX is well suited for classification tasks and should certainly be considered when a high quality transparent model is needed.

9.2.6 Considerations
G-REX should definitely be considered when choosing a rule inducer as it can use a tailored representation language and optimize more than one performance metric.
9.3 Instance Ranking using Ensemble Spread
This chapter is based on the paper “Instance Ranking using Ensemble Spread”, [KJN07].

9.3.1 Summary
This chapter investigates a technique for estimating ensemble uncertainty originally proposed in the weather forecasting domain. Ensembles are used since they are accurate, reliable and can provide case dependent estimates of forecast uncertainty. The diversity of the predictions made by the ensemble members is called the forecast spread or ensemble spread and in a way represents the forecast uncertainty. Spread is mathematically defined as the standard deviation of the ensemble members from the ensemble mean.

The overall purpose is to find out if the technique can be modified to operate on a wider range of regression problems. The main difference, when moving outside the weather forecasting domain, is the lack of extensive statistical knowledge readily available for weather forecasting. In this chapter, four different modifications to the original technique are suggested. In the experiments, the modifications are compared to each other and to two straightforward techniques, using ten publicly available regression problems. Three of the techniques show promising results, especially one modification based on genetic algorithms. The suggested modification can accurately determine whether the confidence in ensemble predictions should be high or low.

9.3.2 Background
The domain of weather forecasting is closely related to information fusion. Data is collected from multiple sources and forecasting systems have to deal with a high degree of uncertainty. The forecasts also have to be constantly updated as conditions change rapidly. A lot of research has been performed in the weather forecasting domain during the last decades, often with great success. The research has been focused on ensemble forecasting, which today is an integral part of numerical weather prediction [TCM01]. Using ensembles for forecasting is both more accurate and more reliable compared to single model forecasts. Another key advantage for ensemble forecasting is that it can provide case dependent estimates of forecast uncertainty [Ehr97]. This is considered a requirement for weather forecasts, but it is clearly an advantage in all decision making, since a forecast with associated uncertainties is more informative.

The techniques used when performing weather forecasting are often a combination of machine learning algorithms and statistical knowledge of weather phenomena. One promising approach is presented in [TCM01], where
predictions of the ensemble members are sorted into a number of bins, according to a fixed set of continuous ranges. The principle is that it should be equally probable that the predicted value would fall into any bin. In [TCM01] the bins are calculated using statistical weather databases. It should be noted that the statistics are only used to pick the boundaries and not to train the ensemble. Each instance has its own set of bins, but the ranges are the same for all instances. The distribution of the predictions over the bins is calculated for each instance and each bin gets a score according to the number of predictions it contains.

Two different approaches are used to rank the instances according to the bin scores. The first approach scores the instances according to the ensemble mode, i.e. the highest bin score for each instance. A high mode corresponds to a compact ensemble where many ensemble members predict similar values. A lower mode indicates a diverse ensemble with little agreement between the ensembles. A higher value should presumably correspond to a lower prediction error. The second approach is member support, i.e. the number of members who fall into the same bin as the ensemble prediction.

The purpose with this study is to investigate how the technique purposed in [TCM01], could be adapted to a wider range of regression problems. As mentioned above, the main difference when moving outside the weather forecasting domain is that most regression problems lack the extensive statistical knowledge readily available for weather forecasting. Hence it is not possible to calculate a priori probabilities needed to select the bin boundaries beforehand.

9.3.3 Method
In the experiments, four alternative approaches of calculating bin boundaries are suggested. Evaluation is done against three techniques that do not use binning. All techniques will be described in the following sections, starting with the binning techniques called Equal bins, Naïve bins, Trimmed bins and Genetic bins, which are followed by the non binning techniques, Best, Variance and ANN.

Evaluation
Mean absolute error (MAE) is used to measure the performance of the ensemble. The performance of each probability ranking technique on a certain data set is calculated as the MAE of the ensemble for the 50% of the instances that the technique ranked as having the lowest error. In the original study, see [TCM01], the technique was used to select the worst 10% and the best 10% instances, but here only the result for the top 50% instances are evaluated. This is required to get a sufficient number of instances when using 10-fold cross validation as some data sets are quite small.
As the data sets differ greatly in scale, MAE cannot be used to compare the techniques over all data sets. It is widely accepted that a unit free metric must be used if forecast methods are compared over several data sets [LZ05]. To make the metric unit free, the percentage decrease in error (PDE) compared to the ensemble MAE is calculated.

\[
PDE(T, D) = 1 - \frac{\text{MAE}(T, D)}{\text{MAE}(E, D)}
\]

Here, \( T \) is the evaluated technique and \( D \) is the data set and \( E \) is the ensemble.

**Equal bins (EQU)**

In the weather forecasting domain, the climatologic probability used to select bin boundaries, is calculated from historic data. Note that this data is more extensive then the data used to train the ensemble. For regression problems without extensive historical data the bin boundaries have to be estimated from the training data. Calculations based on a smaller subset (the training set) will of course be less reliable. Bins with equal probability are created using (37), i.e. by sorting the instances after the target variable and then selecting equally spaced instance values as boundary values. This will results in bins that are equally probable for the training instances.

\[
b_i = \text{sort}(f(x)); j = \text{floor}\left( i \times \frac{\#\text{instances}}{\#\text{bins}} \right)
\]

In the equation above \( b_i \) is the boundary, \#instances is the number of instances in the training set and \#bins is the number of bins to calculate.

**Naïve bins (NAI)**

The simplest way to calculate the bin boundaries is to divide the spread of the predictions equally among them.

\[
b_i = \min(E(x)) + \frac{(\max(E(x)) - \min(E(x)))}{\#\text{bins}} \times i
\]

Here, \( E(x) \) is the ensemble prediction for the instances being ranked. This naïve way to calculate the boundaries is sensitive to outliers, since outliers could draw some of the bins to a range where very few predictions fall. Figure 21 shows an example of an ensemble with five members and their predictions. All instances have predicted values between 40 and 60, except two outliers. The outliers
(instances 5 and 15) affect the calculation of the boundaries (the horizontal lines are the bin boundaries) in such a way that in reality only two of the bins are used for the ranking of the other instances.

![Figure 21 - Bins affected by outliers](image1)

**Trimmed bins (TRI)**

A value is normally defined as an outlier if it is more than two standard deviations from the mean. The sensitivity to outliers can be reduced by trimming the max and min using equation (39).

$$b_i = \text{mean}(f(x)) - 2 \times \text{std}(f(x)) + \frac{4 \times \text{std}(f(x))}{\# \text{bins}} \times i$$  \hspace{1cm} (39)

Figure 22 shows that the trimmed bins are a bit more suited to the problem than in Figure 21.

![Figure 22 - Trimmed bins](image2)

**Genetic bins (GA)**

Another approach is to choose boundaries that are optimal for the currently known data, i.e. the training set, hoping that they also will perform well for novel instances.
Here the optimization is done using genetic algorithms. The bin boundaries are coded in a chromosome as a set of values (genes) representing the size of each bin. When evaluated a chromosome is translated into the real bin boundaries by formula (40), which are then used to calculate ensemble bin scores for each instances.

\[
b_1 = \min(f(x)) + (\max(f(x)) - \min(f(x))) \frac{\sum_{i=1}^{n} \text{gene}_i}{n} \]

\[
b_i^{2-n} = b_{i-1} + (\max(f(x)) - \min(f(x))) \frac{\sum_{i=1}^{n} \text{gene}_i}{n} \]

Since the evaluation criterion is defined as finding the 50% of the instance with the lowest ensemble error, the fitness function is defined as the MAE of the ensemble for the 50% of the instances with the highest bin scores. The population is evolved using the Genetic algorithm toolbox in MatLab, using standard settings for mutation, crossover and reproduction. The initial population contains 300 randomly generated chromosomes and evolution is performed over 100 generations using roulette wheel selection. Mutations are done by adding a random number, or mutation, chosen from a Gaussian distribution, to each gene of the parent chromosome.

**Variance (VAR)**

The most straightforward and naïve way to rank the ensemble predictions according to error probability is to use the ensemble variance for each instance [TB03]. The ensemble variance is the variance of the predictions of the ensemble members. A lower variance indicates a more compact ensemble prediction and should correspond to a lower prediction error. When used each instance is assigned a score equal to the ensemble variance for that instance.

**ANN**

Here an ANN is trained to estimate a function describing the error made by the ensemble. This ANN is also trained on the training set, but the individual predictions from the ensemble members are the inputs and the corresponding error made by the ensemble is the target. When the ANN is fully trained, it is applied to the test data, and the ANN output is noted for each instance. Finally the MAE is calculated on the ensemble error for the 50% instances for which the ANN has estimated the lowest ensemble error.
Best

Best is the optimal result that could be achieved in the current experimental setup, i.e. the ensemble MAE for the 50% instances for which the ensemble had the lowest prediction error. It should be noted that this technique cheats by looking at the test instances and is only used for comparisons.

9.3.4 Experiments

The study is evaluated using 10-fold cross validation on 10 data sets from the UCI repository. The ensembles used in the experiments consist of 20 ANNs. Each ANN is trained on an individual subset of the training set. The subset is created by randomly picking 80% of the training instances without replacement. The number of input features is also reduced by selecting 80% of the features randomly. The purpose of this is, of course, to increase diversity. Each ANN is initiated with random weights and with a random number of hidden units between 0 and 20. The training algorithm used to train the ANNs is MatLab’s `traingdx`, which is a gradient descent backpropagation algorithm, using a momentum term and adaptive learning rate. A validation set is used to decide when to stop the training. The validation sets is created by removing 20% of the training instances and assigning them to the validation set.

9.3.5 Results

All results presented in this chapter are the mean result over all ten folds, if it is not otherwise noted. In the following sections the result for the ensemble are first presented. Next the mode, (the highest bin score for each instance), and the member support, (the number of members who fall into the same bin as the ensemble prediction) are presented for each technique and data set.

**Ensemble result**

Table 21 presents the MSE for the ensemble on each data set. These results are only presented as a background to the following sections as the results there show how much this error could be reduced by the different techniques.
Result using mode scores
Table 22 shows the average PDE over all ten folds for each data set when the ensemble mode is used as scoring function. The PDE shows to which extent the ensemble error presented in Table 22 could be reduced by only choosing the top 50% ranked instances.

<table>
<thead>
<tr>
<th>Data set</th>
<th>NAI</th>
<th>TRI</th>
<th>EQU</th>
<th>GA</th>
<th>VAR</th>
<th>ANN</th>
<th>BEST</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PDE</td>
<td>R</td>
<td>PDE</td>
<td>R</td>
<td>PDE</td>
<td>R</td>
<td>PDE</td>
</tr>
<tr>
<td>AUO</td>
<td>47</td>
<td>4</td>
<td>53</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>48</td>
</tr>
<tr>
<td>BHD</td>
<td>17</td>
<td>5</td>
<td>24</td>
<td>4</td>
<td>-24</td>
<td>6</td>
<td>30</td>
</tr>
<tr>
<td>HUN</td>
<td>43</td>
<td>4</td>
<td>48</td>
<td>2</td>
<td>40</td>
<td>5</td>
<td>49</td>
</tr>
<tr>
<td>IPD</td>
<td>68</td>
<td>4</td>
<td>69</td>
<td>3</td>
<td>38</td>
<td>6</td>
<td>70</td>
</tr>
<tr>
<td>MAC</td>
<td>52</td>
<td>2.5</td>
<td>52</td>
<td>2.5</td>
<td>26</td>
<td>6</td>
<td>53</td>
</tr>
<tr>
<td>PHA</td>
<td>12</td>
<td>4</td>
<td>15</td>
<td>3</td>
<td>7</td>
<td>5</td>
<td>28</td>
</tr>
<tr>
<td>SER</td>
<td>59</td>
<td>2.5</td>
<td>56</td>
<td>5</td>
<td>44</td>
<td>6</td>
<td>59</td>
</tr>
<tr>
<td>SLE</td>
<td>-7</td>
<td>6</td>
<td>17</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>VET</td>
<td>16</td>
<td>5</td>
<td>19</td>
<td>1.5</td>
<td>18</td>
<td>3</td>
<td>14</td>
</tr>
<tr>
<td>WIS</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>6</td>
<td>9</td>
<td>3.5</td>
<td>17</td>
</tr>
<tr>
<td>MEAN</td>
<td>30.9</td>
<td>4.2</td>
<td>35.4</td>
<td>3.0</td>
<td>15.9</td>
<td>5.1</td>
<td>37.8</td>
</tr>
</tbody>
</table>

Table 22 - PDE using mode scores (CD = 2.15)

GA is the technique that achieves the highest PDE on average. It is 2% better on average than the second best technique VAR, and 29% below the theoretically optimal result. All techniques except EQU perform reasonably well on average, but only GA and VAR achieves an error reduction on all data sets. NAI, EQU and ANN increase the error for at least one of the data sets.

In the main experiments, the performance of the techniques is only measured on the top 50%. To show the general performance of the GA technique, Figure 23...
displays how the total error increases when choosing the instances in ascending order to their mode score. The solid line describes the error when the instances are selected randomly. Note that the increase in % of total MAE decreases when the mode increases (the increase in error is correlated to the angle of the GA line).

Figure 23 - Gain chart for fold 1 of SER

**Results using member support**

Table 23 shows the results when member support was used as scoring function. Note that the results for VAR, ANN and BEST are the same as in Table 22, since these techniques do not use the binned approach.

All binned techniques here perform worse than VAR, when compared over all data sets. In addition they are clearly worse than the result achieved by using the mode scoring function. EQU performance is not affected by support scoring but it is still far worse than the VAR technique.

<table>
<thead>
<tr>
<th>Data set</th>
<th>NAI</th>
<th>TRI</th>
<th>EQU</th>
<th>GA</th>
<th>VAR</th>
<th>ANN</th>
<th>BEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUO</td>
<td>36</td>
<td>4</td>
<td>41</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>39</td>
</tr>
<tr>
<td>BHD</td>
<td>15</td>
<td>5</td>
<td>19</td>
<td>4</td>
<td>-24</td>
<td>6</td>
<td>27</td>
</tr>
<tr>
<td>HUN</td>
<td>21</td>
<td>6</td>
<td>26</td>
<td>5</td>
<td>40</td>
<td>3</td>
<td>45</td>
</tr>
<tr>
<td>IPD</td>
<td>64</td>
<td>3.5</td>
<td>64</td>
<td>3.5</td>
<td>38</td>
<td>6</td>
<td>68</td>
</tr>
<tr>
<td>MAC</td>
<td>49</td>
<td>3</td>
<td>39</td>
<td>5</td>
<td>26</td>
<td>6</td>
<td>53</td>
</tr>
<tr>
<td>PHA</td>
<td>-7</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>7</td>
<td>3</td>
<td>21</td>
</tr>
<tr>
<td>SER</td>
<td>5</td>
<td>5</td>
<td>-19</td>
<td>6</td>
<td>44</td>
<td>4</td>
<td>59</td>
</tr>
<tr>
<td>SLE</td>
<td>-21</td>
<td>6</td>
<td>-11</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>-8</td>
</tr>
<tr>
<td>VET</td>
<td>-16</td>
<td>6</td>
<td>0</td>
<td>5</td>
<td>18</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>WIS</td>
<td>0</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>9</td>
<td>3.5</td>
<td>11</td>
</tr>
<tr>
<td>MEAN</td>
<td>14.6</td>
<td>4.95</td>
<td>16.3</td>
<td>4.45</td>
<td>15.9</td>
<td>4.15</td>
<td>32.3</td>
</tr>
</tbody>
</table>

Table 23 - PDE using support scores (CD = 2.15)
9.3.6 Conclusions
The most obvious result of this study is that the mode scoring function clearly outperforms the function using member support. The mode function reduces the error on the top 50% twice as much as when member support is used, and does this for all but one technique (EQU). The difference is also statistically significant for all techniques except EQU.

Only GA, TRI and VAR succeed in reducing the ensemble error on the top 50% for all data sets. These techniques also have a better performance in general, compared to the other techniques. GA is the most promising of the techniques, and is on average 2% better than the second best technique VAR, but this difference is not statistically significant. GA achieves an error reduction of 38% on average which has to be recognized as a good result. The optimal result achieved by BEST when it sorts the instances after the actual ensemble error is 67%. GA’s result is a substantial decrease of the ensemble error and this information could be very useful for any decision maker.

It is surprising that EQU, which is the technique most similar to the original technique proposed in [TCM01] performed by far the worst. The main reason for this is probably that in the original study, the bounds were calculated from extensive statistical knowledge which the data sets in this study lacked. The method used by EQU to produce equally likely bins could possibly work for larger data sets, but the data sets used in this study were apparently too small to allow calculation of a priori probabilities.

Even if EQU did not perform as well as expected, the GA version of the original technique did produce very promising results. This should be seen as a successful example of how techniques from a very mature domain closely related to information fusion, could contribute to fusion goals. Even if the domain does not match exactly, small adjustments can be done to techniques to make them compatible.

Most important is the fact that the GA technique combined with the mode scoring function clearly outperformed the frequently used method of ranking by ensemble variance.
9.4 Inconsistency – Friend or Foe
This chapter is based on the paper “Inconsistency – Friend or Foe”, [JKN07].

9.4.1 Summary
When evaluating rule extraction algorithms, one frequently used criterion is consistency; i.e. the algorithm must produce similar rules every time it is applied to the same problem. Rule extraction algorithms based on evolutionary algorithms are, however, inherently inconsistent, something that is regarded as their main drawback. In this study, it is argued that consistency is an overvalued criterion, and that inconsistency can even be beneficial in some situations. The study contains two experiments, both using publicly available data sets, where rules are extracted from neural network ensembles. In the first experiment, it is shown that it is normally possible to extract several different rule sets from an opaque model, all having high and similar accuracy. The implication is that consistency in that perspective is useless; why should one specific rule set be considered superior? Clearly, it should instead be regarded as an advantage to obtain several accurate and comprehensible descriptions of the relationship. In the second experiment, rule extraction is used for probability estimation. More specifically, an ensemble of extracted trees is used in order to obtain probability estimates. Here, it is exactly the inconsistency of the rule extraction algorithm that makes the suggested approach possible. This is the main contribution, as the technique is novel improvement of G-REX, and that it proves that inconsistency does not need to be regarded as a disadvantage for rule extraction techniques.

9.4.2 Background
As mentioned in chapter 5.6 some authors suggest that consistency is an important property of rule extraction techniques. Since G-REX (a rule extraction technique presented in chapter 6) uses GP to optimize the extracted representation, consecutive runs, even using the same opaque model, will normally result in slightly different extracted rules. In other words, G-REX is inherently inconsistent. This is also considered to be the main drawback for G-REX (and other rule extractors based on evolutionary algorithms) in the survey reported by Huysmans, Baesens and Vanthienen; see [HBV06]. However in this chapter, it is argued that the criterion consistency is insignificant. In addition, it is showed how the fact that a rule extraction algorithm is inconsistent can be utilized in order to perform probability estimation.
Probability estimation trees

Algorithms inducing decision trees are very popular, mostly because their ability to produce transparent and rather accurate models. In addition, decision trees are consistent, (i.e. deterministic), work very well with large data sets and require a minimum of parameter tweaking. However, decision trees, i.e PETs are known to produce poor probability estimates. Provost and Domingo in [PD03] try to overcome this disadvantage by combining PETs based on the Laplace estimator, described in chapter 5.2.3, with the ensemble technique bagging, described in chapter 5.2.1. More specifically, an ensemble consisting of several, independently trained, PETs was created and final probability estimates were computed by averaging the individual probability estimates from all ensemble members. To obtain some diversity (inconsistency in the produced models), each PET was trained using only part of the training data, here a bootstrap sample. In order to obtain what they call well-behaved PETs, Provost and Domingo changed the C4.5 algorithm by turning off both pruning and the collapsing mechanism. This obviously led to much larger trees, which, in fact, turned out to be much better PETs; for details see the original paper [PD03].

9.4.3 Method

The empirical study consists of two main experiments. The purpose of the first experiment is to investigate what level of consistency to expect from G-REX. In addition, G-REX accuracy is also compared to CART. The purpose of the second experiment is to evaluate whether G-REX can be used for probability estimation. The performance is again compared to CART. In both experiments, G-REX extracts rules from ANN ensembles. Each ensemble consists of 20 independently trained ANNs. All ANNs are fully connected feed-forward networks and a localist representation is used. Of the 20 ANNs, five have no hidden layer; ten have one hidden layer and the remaining five have two hidden layers. The exact number of units in each hidden layer is slightly randomized, but is based on the number of features and classes in the current data set. For an ANN with one hidden layer the number of hidden units is determined from (41).

\[ h = \lceil 2 \times \text{rand} \sqrt{(v \times c)} \rceil \]  

(41)

\textit{v} is the number of input features and \textit{c} is the number of classes. \textit{rand} is a random number in the interval [0, 1]. For ANNs with two hidden layers, the number of units in the first and second hidden layer (\( h_1 \) and \( h_2 \)) are determined using equation (42).
\[ h_1 = \frac{\sqrt{(v^* c)}}{2} + 4 \cdot \text{rand} \cdot \frac{\sqrt{(v^* c)}}{c} \]

\[ h_2 = \left( \text{rand} \cdot \frac{\sqrt{(v^* c)}}{c} + c \right) \]

Each ANN is, in an attempt to increase diversity, trained using only 80% (randomized without replacement) of available training instances.

In this study, G-REX will be compared to CART, so the representation language is chosen to mimic CART; see the IF-ELSE grammar presented using Backus-Naur form in Table 7 in chapter 5.4.2. The GP settings used for G-REX in this study are given in Table 24.

<table>
<thead>
<tr>
<th>Number of generations</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
<td>1000</td>
</tr>
<tr>
<td>Crossover probability</td>
<td>0.8</td>
</tr>
<tr>
<td>Mutation probability</td>
<td>0.01</td>
</tr>
<tr>
<td>Creation type</td>
<td>Ramped half and half</td>
</tr>
<tr>
<td>Maximum creation depth</td>
<td>6</td>
</tr>
<tr>
<td>Length punishment (p)</td>
<td>0.01</td>
</tr>
<tr>
<td>Batch size</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 24 - G-REX Settings

Experiment 1
The purpose of Experiment 1A is to evaluate the consistency of G-REX. More specifically, the algorithm is run several times to extract a number of trees. The trees are then compared in order to determine whether they are similar or not. In this experiment, similarity is measured only based on the predictions made by the different extracted trees. This means that the actual meaning of the rules is ignored. The evaluation is divided in two parts; intra-model and inter-model. Intra-model compares trees extracted from a mutual opaque model, while inter-model compares trees extracted from different opaque models, trained on identical data. The metric used is the proportion of instances classified identically by the two extracted trees. It should be noted that this metric is blind to how accurate the rules are, it measures only whether the two predictions are identical.

In Experiment 1B, G-REX accuracy is compared to CART. The CART implementation used here is the one in Clementine. For the comparison, standard 10-fold cross validation is used. To obtain G-REX accuracy on a specific fold, ten G-REX trees are individually extracted from the ANN ensemble. Then the single tree having highest fitness is applied to the test set. This is a standard procedure
when using G-REX, although the number of extracted trees to choose from is normally rather two or three than 10. But in order to investigate the consequences of G-REX inconsistency, all ten G-REX trees are also compared to the single CART tree.

**Experiment 2**

The purpose of the second experiment is to investigate whether G-REX trees can be used for probability estimation. Here, three separate experiments are conducted. In Experiment 2A, G-REX and CART are again both run normally; i.e. just performing classification. In Experiment 2B, G-REX produces ten trees for each individual fold, and the tree with the highest fitness is applied to the test set. CART here uses all available data and consequently produces one tree per fold. Both G-REX and CART use the Laplace estimator to produce probability estimates based on the training instances. To determine the quality of the probability estimate, each tree ranks all test instances according to the estimates and discards the worst 20%. The accuracy is finally calculated for each tree on the remaining instances (Top80). In Experiment 2C, the G-REX trees are used as an ensemble with 10 members. The tree having the highest fitness is still used for prediction, but probability estimation is made by combining all ten trees; as described above. CART here first trains one tree normally; i.e. the parameter setting favors generalization accuracy. After that, ten additional CART trees are trained, prioritizing accuracy on the training data and using no pruning. These trees are consequently quite large. In order to introduce some diversity, each tree uses only 70% of the available training data. CART then uses the first tree for prediction, but the entire ensemble for probability estimation, again as described above. Using these techniques, it is very important to recognize that both G-REX and CART still produce one, transparent, model. For the evaluation, the same procedure as in Experiment 2B is used; i.e. 80% of the test set instances are chosen for prediction, using the Laplace estimator. For all experiments, 10-fold cross validation is used. The CART implementation used here is the one in MatLab; i.e. the ‘Tree Fit’ function in the statistics toolbox. It should also be noted that in this experiment, all data sets were stratified, which is the main reason for the higher accuracies overall.

**9.4.4 Results**

Table 25 shows the intra-model results for one fold from the Zoo problem. In this fold, the average consistency is 0.910 for the training set and 0.892 for the test set. The interpretation is that a pair of extracted trees, on average, agrees on 91.0% of all training set instances, and on 89.2% of all test instances.
<table>
<thead>
<tr>
<th>TRAIN</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
<th>Test</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>0.920</td>
<td>0.920</td>
<td>0.960</td>
<td>0.970</td>
<td>R1</td>
<td>0.880</td>
<td>0.880</td>
<td>0.960</td>
<td>0.880</td>
</tr>
<tr>
<td>R2</td>
<td>0.970</td>
<td>0.880</td>
<td>0.920</td>
<td></td>
<td>R2</td>
<td></td>
<td>0.920</td>
<td>0.880</td>
<td>0.880</td>
</tr>
<tr>
<td>R3</td>
<td>0.893</td>
<td>0.907</td>
<td></td>
<td></td>
<td>R3</td>
<td>0.880</td>
<td></td>
<td></td>
<td>0.880</td>
</tr>
<tr>
<td>R4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.880</td>
<td>R4</td>
<td></td>
<td>0.880</td>
</tr>
</tbody>
</table>

Table 25 - Intra-model consistency (ZOO)

Similarly, Table 26 shows the inter-model results for one Diabetes fold. Here the average consistency is 0.910 for the training set and 0.874 for the test set.

<table>
<thead>
<tr>
<th>TRAIN</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
<th>Test</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>0.899</td>
<td>0.891</td>
<td>0.905</td>
<td>0.908</td>
<td>R1</td>
<td>0.885</td>
<td>0.859</td>
<td>0.880</td>
<td>0.901</td>
</tr>
<tr>
<td>R2</td>
<td>0.943</td>
<td>0.922</td>
<td>0.915</td>
<td></td>
<td>R2</td>
<td></td>
<td>0.901</td>
<td>0.911</td>
<td>0.870</td>
</tr>
<tr>
<td>R3</td>
<td>0.917</td>
<td>0.896</td>
<td></td>
<td></td>
<td>R3</td>
<td>0.854</td>
<td>0.844</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R4</td>
<td></td>
<td>0.903</td>
<td>R4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.833</td>
<td></td>
</tr>
</tbody>
</table>

Table 26 - Inter-model consistency (PID)

Table 27 shows the overall results from Experiment 1A. On average, two extracted rules agree on approximately 89% of all instances. Somewhat surprising, this holds for both training and test sets, as well as intra-model and inter-model.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Intra-model</th>
<th>Inter-model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRAIN TEST</td>
<td>TRAIN TEST</td>
</tr>
<tr>
<td>BLD</td>
<td>0.805 0.751</td>
<td>0.829 0.793</td>
</tr>
<tr>
<td>CHD</td>
<td>0.893 0.845</td>
<td>0.912 0.886</td>
</tr>
<tr>
<td>PID</td>
<td>0.904 0.893</td>
<td>0.904 0.881</td>
</tr>
<tr>
<td>SMR</td>
<td>0.914 0.898</td>
<td>0.896 0.876</td>
</tr>
<tr>
<td>WBC</td>
<td>0.986 0.971</td>
<td>0.987 0.967</td>
</tr>
<tr>
<td>ZOO</td>
<td>0.897 0.903</td>
<td>0.911 0.858</td>
</tr>
<tr>
<td>MEAN</td>
<td>0.900 0.877</td>
<td>0.907 0.877</td>
</tr>
</tbody>
</table>

Table 27 - Average consistency over all pairs and folds

Table 28 shows the comparison between CART and G-REX in Experiment 1B. The first two columns contain the overall test set accuracies obtained using 10-fold cross validation. The last three columns compare all extracted G-REX trees to CART. The numbers tabulated are the number of wins, ties and losses for G-REX, when comparing the ten extracted trees on every fold to the corresponding CART tree.
Table 28 - Accuracy comparison between G-REX and CART

Although it interesting to note that G-REX outperforms CART on 8 of the 10 data sets, this is not the major point here. The most important observation is instead that a huge majority of all individual G-REX trees are at least as accurate as their CART counterpart. Only 13.2% of all GREX trees have lower accuracy compared to the corresponding CART tree. Table 29 shows the results from Experiment 2. Note on the last four columns can be compared using the Top 80 approach described above.

Table 29 - Comparison between methods CD = 1.38

There are several interesting results in Table 29. First of all, it should be noted that a single G-REX tree most often outperformed the corresponding CART tree; both in Experiment 2A and 2B. In addition, it is obvious that both techniques could utilize the probability estimates obtained in order to rank instances based
on how hard they are to predict. Even if the differences in ranks are not significant between all techniques, (CD = 1.38), the G-REX ensemble approach clearly obtained a superior probability estimate.

9.4.5 Conclusions
Consistency is a hard criterion to assess. Although it must be considered important to have at least fairly high consistency, it is not obvious exactly what level this translates to. As comparison; tree inducers like CART will produce identical trees every time, so their consistency would be 100%. With this in mind, the consistency obtained by G-REX initially raised some questions. As seen in Experiment 1, on some data sets, more than 15% of all instances are, on average, classified differently by two G-REX runs. Is it really possible to put faith in one specific extracted rule when another run might produce a different rule, disagreeing on 15% of all instances? The answer to this question is not obvious. Intuitively, most data miners would probably want one accurate and comprehensible model. On the other hand, most individual G-REX runs do produce exactly that; i.e. a compact yet accurate tree. As shown in Experiment 1, any single G-REX tree is most often more accurate than the corresponding tree induced by CART. Apparently there are several descriptions (models) of the relationship that all have similar accuracy. Using a tree inducer you get one specific model, but why is that better than obtaining several, slightly different, models, each having high accuracy? Based on this, it can be argued, that for rule extraction algorithms, low consistency, should not be considered a major problem. Inconsistency could even be seen as measure of the probability that the created model describes the true underlying concept.

Regarding the experiment with probability estimation, it is obvious that G-REX is more than capable of producing good probability estimates. Especially the results where G-REX used all extracted trees in order to obtain a probability estimate are promising. Clearly, it is the inherent inconsistency of G-REX that makes this approach possible, without having to use separate parts of the data set for each ensemble member. Considering the relatively small amount of extra work needed to extract several trees instead of one, this is an encouraging result for both G-REX and rule extraction in general.

9.4.6 Discussion and future work
The ultimate goal of rule extraction is of course one accurate and comprehensible model. At the same time, it is well-known that sufficiently diverse ensembles are more accurate than single models. One interesting question is therefore whether extracted trees could be fused together to increase the performance. Another
option is to keep all trees, and somehow try to visualize them together in order to increase the understanding of the underlying relationship. In this study, G-REX extracted rules from an ANN ensemble. It would be interesting to investigate whether probability estimates based on the ensemble could be used in some way to improve the probability estimation performed by the G-REX tree. G-REX has previously been used to produce different types of regression trees. With this in mind, one obvious study is to try to utilize G-REX’s inconsistency, in order to produce probability estimates for regression problems such as time series. This should be compared to the standard way of producing probability estimates for times series; i.e. to post-process the output from each ensemble member statistically (Ensemble-MOS).

9.4.7 Considerations
Most data sets in the experiments have a skewed distribution (last column in Table 11) but no correction for this is done in the experiments. It is possible that a correction towards the true prior probabilities could further improve the results.
9.5 Genetic Programming – A Tool for Flexible Rule Extraction
This chapter is based on the paper “Genetic Programming – A Tool for Flexible Rule Extraction”, [KJN07]

9.5.1 Summary
G-REX, a rule extraction algorithm based on genetic programming, has previously been evaluated in several studies. One key property of G-REX, due to the use of genetic programming, is the possibility to use different representation languages. In this study G-REX is applied to estimation tasks. More specifically, three representation languages are evaluated using eight publicly available data sets. The quality of the extracted rules is compared to two standard techniques producing comprehensible models; multiple linear regression and the decision tree algorithm CART. The results show that G-REX outperforms the standard techniques, but that the choice of representation language is important. This is not a problem for G-REX, since it can use any representation language. In addition, it is easy to perform initial experiments in order to choose the most suitable. The study also shows that it is often beneficial to guide the GP search in some way. In the most successful technique called LINREG, GP was used to divide the data set in subsets on which standard linear regression was applied. All representations could actually be used to construct the rules produced with LINREG but the GP search seldom reached these solutions due to the enormous search space. From the results it is clear that the right representation language can improve both accuracy and comprehensibility. It is also shown how different representations can be optimized using G-REX. Another important contribution is the method of guiding the GP process using the least square method, which clearly improves the performance of the evolved rules.

9.5.2 Background
In the only previous estimation study [JKN04], G-REX used a simple representation language described using Backus-Naur in Table 9. Furthermore, the use of G-REX for estimation tasks, was only tested in the market response modeling domain. With this in mind, the overall purpose of this study is to evaluate G-REX using two new representation languages for estimation.
9.5.3 Method
This study evaluates if GP, and more specifically G-REX, can be used to extract accurate and comprehensible rules, (for estimation tasks), from an ensemble of ANNs. The extracted rules are evaluated against the tree algorithm CART, and multiple linear regression. The following sections will describe the experimentation.

**Ensemble Creation**
The ensembles used in the experiments consist of 20 fully-connected ANNs, each having one hidden layer. Each ANN is trained on an individual subset of the training set. The subset is created by randomly picking 80% of the training instances without replacement. The number of input variables is also reduced by selecting 80% of the features randomly. The purpose of this is, of course, to increase the ensemble diversity. Each ANN is initiated with random weights, and with a random number of hidden units which is calculated by (43), where \( \text{rand} \) is a random number between 0 and 1.

\[
\# \text{hidden} = \lfloor \text{rand} \times 20 \rfloor \tag{43}
\]

MatLab’s training algorithm traingdx is used for training and a validation set, (20\% of the training set) is used to stop the training.

**Representation languages**
To evaluate the importance of representation language, Three different representations languages are evaluated in this study.

The IF-REG representation language, (Table 9 in chapter 6.2), is a simple representation language for a regression tree that uses the same set of functions and terminals as CART. The only difference is that CART’s syntax only allows comparisons between a variable and a constant value in a condition, while IF-REG also allows comparisons between two variables. IF-REG is the original representation language used for G-REX estimation in [JKN04].

The LINREG, (see Table 30), representation extends IF-REG by allowing linear regression as terminals.
<table>
<thead>
<tr>
<th>Function set</th>
<th>If</th>
<th>&lt;</th>
<th>&gt;</th>
<th>==</th>
<th>*</th>
<th>+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal set</td>
<td>ConV</td>
<td>CatV</td>
<td>Double</td>
<td>Category</td>
<td>LinC</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>If</th>
<th>Condition</th>
<th>then</th>
<th>If</th>
<th>LinReg</th>
<th>else</th>
<th>If</th>
<th>LinReg</th>
</tr>
</thead>
<tbody>
<tr>
<td>LinReg</td>
<td>ConV</td>
<td>*</td>
<td>LinC</td>
<td>+</td>
<td>LinC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Condition</td>
<td>ConV</td>
<td>&gt;</td>
<td>&lt;</td>
<td>Double</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ConV</td>
<td>CatV</td>
<td>==</td>
<td>Category</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CatV</td>
<td>Continuous variable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Double</th>
<th>Double in range of associated ConV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>Category of associated CatV</td>
</tr>
<tr>
<td>LinC</td>
<td>Double value calculated using the least square method on the training instance reaching the associated LinReg</td>
</tr>
</tbody>
</table>

Table 30 - BNF for LINREG

Using this grammar, the representation language becomes more powerful, since a terminal can predict different values for each of the instances it is associated with. The predicted value \( \hat{y}_i \) for instance \( i \) is calculated using (44); where \( z_i \) is the value of one of the input variables, \( K \) and \( M \) are constants found by the linear regression.

\[
\hat{y}_i = K \times z_i + M \tag{44}
\]

When a linear regression expression is created, \( K \) and \( M \) are calculated by the least square method. The calculations are performed on the training instances that are associated with the linear regression terminal when the rule is first created. During evolution, mutation and crossover are only applied to the if-statements and its conditions, which will affect how the instances are associated with the terminals but not the linear regressions which will remain the same. Mutation can introduce new linear regression terminals during the evolution. Figure 24 shows an example of a rule created with LINREG.

![Figure 24 - Sample rule for LINREG](image-url)
The last representation FREE presented below is similar to LINREG with two exceptions. First, each subgroup of instances can be associated with a combination of several linear regressions, making it possible to base the prediction on several input variables. Secondly the linear regressions are created randomly and are affected by the genetic operations in normal fashion.

<table>
<thead>
<tr>
<th>Function set</th>
<th>If</th>
<th>&lt;</th>
<th>&gt;</th>
<th>==</th>
<th>*</th>
<th>+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal set</td>
<td>ConV</td>
<td>CatV</td>
<td>Double</td>
<td>Category</td>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Condition</th>
<th>Exp</th>
<th>then</th>
<th>Exp</th>
<th>else</th>
<th>Exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConV</td>
<td>If</td>
<td>&gt;</td>
<td>&lt;</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>CatV</td>
<td>+</td>
<td></td>
<td></td>
<td>category</td>
<td></td>
</tr>
<tr>
<td>Term</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>C</td>
</tr>
<tr>
<td>Term</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td>C</td>
</tr>
</tbody>
</table>

| ConV | Continuous variable |
| CatV | Categorical variable |
| Double | Double in range of associated ConV |
| Category | Category of associated CatV |
| C    | Double |

Table 31 - Free representation (FREE)

The linear regressions are combined by addition, in the same way as in multiple linear regression. This is in essence a way to evolve different multiple linear regressions for different parts of the data sets.

![Figure 25 - Sample rule for FREE](image)

**Fitness Function**

In previous studies where G-REX has been used to extract regression rules, Equation (29) has been used to calculate the fitness of a rule. The factor $p$ is used
to balance accuracy against comprehensibility, but it will have different effect depending on the magnitude of the error. Before G-REX is applied to a new data set, initial experiments have to be performed to decide an appropriate size of the factor. To achieve a better balancing the fitness function below is used.

\[
Regfitness_i = 1 - \frac{\sum_{k=1}^{n} (|E(i_k) - R_i(\hat{y}_k)|)}{C} - size_i * p
\] (45)

C is initially set to the sum of the target variable and is then updated after each generation according to

\[
C = \begin{cases} 
C*1.5 & \text{if } Regfitness_{best} > 0.9 \\
C & \text{if } 0.1 \geq Regfitness_{best} \geq 0.9 \\
C*0.5 & \text{if } Regfitness_{best} < 0.1 
\end{cases}
\] (46)

C is used to normalize the error made by the rule, in a way that makes the balancing of the length of the rule independent of the magnitude of the error.

**Evaluation**

To evaluate the benefit of using rule extraction, G-REX is evaluated against algorithms that produce comprehensible models. In this study, G-REX is evaluated against multiple linear regression (MREG) and CART [BFO+83]. Both techniques produce transparent models, normally regarded as comprehensible. The experiments were performed on eight data sets from the UCI repository using standard 10-fold cross validation with stratification to ensure that each fold was representative. Details regarding the data sets used in this study can be found in chapter 8.2.

This study focuses on estimation problems where the error metric GMRAE recommended in chapter 4.6.4 cannot be applied as it is relative to random walk. But GMRAE can still be used by simply exchanging the reference to the \( rw \) in (14) with the mean of the currently known actual values, as done in (47).

\[
RAE_i = \frac{|\hat{y}_i - y_i|}{|\bar{y} - y_i|}
\] (47)
9.5.4 Results

The results presented in Table 32 are the average GMRAE of the ten folds of each data set. The results are relative to the result of always predicting the mean value of the target variable, i.e., a value less than one, means that the result is better than the mean prediction. R is the ascending rank of the technique for each data set.

<table>
<thead>
<tr>
<th></th>
<th>IF-REG</th>
<th>FREE</th>
<th>LINREG</th>
<th>MREG</th>
<th>CART</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GMRAE R</td>
<td>GMRAE R</td>
<td>GMRAE R</td>
<td>GMRAE R</td>
<td>GMRAE R</td>
</tr>
<tr>
<td>AUT</td>
<td>0.351  5</td>
<td>0.341  4</td>
<td>0.262  1</td>
<td>0.317  3</td>
<td>0.303  2</td>
</tr>
<tr>
<td>DIA</td>
<td>1.054  4</td>
<td>3.186  5</td>
<td>0.818  1</td>
<td>0.942  2</td>
<td>0.971  3</td>
</tr>
<tr>
<td>BHD</td>
<td>0.852  5</td>
<td>0.734  4</td>
<td>0.488  1</td>
<td>0.506  3</td>
<td>0.501  2</td>
</tr>
<tr>
<td>MAC</td>
<td>0.499  5</td>
<td>0.403  4</td>
<td>0.289  1</td>
<td>0.383  3</td>
<td>0.320  2</td>
</tr>
<tr>
<td>PHA</td>
<td>0.952  4</td>
<td>0.946  3</td>
<td>0.725  2</td>
<td>0.672  1</td>
<td>1.049  5</td>
</tr>
<tr>
<td>SLE</td>
<td>0.856  4</td>
<td>0.777  3</td>
<td>0.665  2</td>
<td>1.104  5</td>
<td>0.238  1</td>
</tr>
<tr>
<td>VET</td>
<td>0.848  2</td>
<td>0.992  4</td>
<td>0.772  1</td>
<td>0.947  3</td>
<td>1.034  5</td>
</tr>
<tr>
<td>WIS</td>
<td>0.961  4</td>
<td>0.886  1</td>
<td>0.902  2</td>
<td>0.903  3</td>
<td>1.008  5</td>
</tr>
<tr>
<td>MEAN</td>
<td>0.796  4.1</td>
<td>1.033  3.7</td>
<td>0.615  1.3</td>
<td>0.722  2.9</td>
<td>0.678  3</td>
</tr>
</tbody>
</table>

Table 32 - GMRAE on individual data sets CD=1.97

A Friedman test yields (p=0.0029) which signifies a statistical difference between the techniques. LINREG has clearly the lowest rank but is not is significant better than MREG and CART as the critical difference rank is 1.97. However, LINREG is clearly the best alternative as it only loose once against MREG and once against CART.

Conclusions

The results show that G-REX outperforms the standard regression techniques, but that the choice of representation language is important. All three G-REX representations differs in performance, which could be expected since they differ in complexity.

In view of the fact that FREE and LINREG can represent the same type of relationships it is interesting to note that LINREG performs significantly better. Since both representations results in similar search spaces, for the GP process, it is obvious that LINREG benefits of the use of the least square method.

However, as LINREG is a more complex representation than IF-REG, it is also slightly less comprehensible, than an IF-REG rule of similar size. A more complex representation could on the other hand capture certain relationships with smaller rules. In any case, this study shows that it can be favorable, both in terms of performance and comprehensibility, to evaluate different representations languages when extracting comprehensible model. Hence, the freedom of
choosing representation language should be considered when evaluating rule extraction techniques.

9.5.5 Considerations
MREG and CART both use quite simple representations while G-REX’s FREE and LINREG are slightly more complex. Hence the comparison is not totally fair, but since G-REX produces very short rules they are still deemed to be at least as comprehensible as the other representations. G-REX could of course have been compared against techniques such as Cubist that produce more complex expressions but these rules are often too complex to be considered comprehensible.
10 Conclusions

In this chapter the more general conclusions of the case studies will be presented and related to the thesis objectives. In section 10.1 these conclusions will also be put into the context of the problem statement and related to information fusion.

1. Implement and evaluate novel components in a rule extraction framework for improved comprehensibility.
   - A novel technique that simplifies extracted G-REX rules, by significantly reducing the size while retaining a high accuracy, is presented in study 9.1. Since the smaller simplified rules are constructed using the same representation language as the original rules, they are clearly more comprehensible. It should be noted that the rules that were simplified already were significantly shorter than rules produced by standard techniques.
   - G-REX is based on GP which enables optimization of arbitrary representation languages. Study 9.5 clearly demonstrates G-REX flexibility when optimizing rules of different representation languages. This is an important property as it allows decision makers to create representation languages with functions and terminals that are natural for the problem domain, thus making the extracted rules more comprehensible. The study also shows that the choice of representation can have a significant effect on the performance of the extracted rules.

2. Implement and evaluate novel techniques for estimating uncertainty in predictions.
   - The binning method presented in study 9.3 seems to be a very promising approach to ensemble uncertainty estimation. When the uncertainty estimates of the best technique, (which was based on GA), was used to rank the predictions; the top 50% (the easy instances) had a mean error that was only 62% of the mean error for all instances.
   - Study 9.4 shows that inconsistency, (often considered a drawback for rule extraction techniques), could be used to greatly improve uncertainty estimates for an extracted model.
Since all rules that were extracted from the same ensemble, in study 9.4, had similar accuracy they can be seen as different possible explanations. Hence the inconsistency of nondeterministic techniques like G-REX can be seen as a measure of the probability that the created model describes the true underlying concept.

3. Evaluating use of alternative score functions and metrics in predictive modeling

- Study 9.2 shows that it is better to optimize the evaluated metric than something else, i.e. if AUC is sought AUC should be optimized. A rather intuitive result, but this should be seen in the light of that most techniques are designed to only optimize accuracy.
- For both alternative score functions presented in 9.2, i.e. optimizing AUC and the performance metric for brevity, G-REX produced rules that where clearly better when evaluated against the optimized metric. The rules also had an accuracy that was comparable to the rules created when optimizing accuracy. Hence both score functions added an extra quality to the produced rules.
- Study 9.5 shows that G-REX can extract estimation rules of different representation languages, using a score function specially adapted for estimation tasks. The best rules have superior performance compared to standard techniques producing transparent models, i.e. MREG and CART.

10.1 Concluding remarks
The results of this thesis can be combined into a simple method for predictive modeling in situations with poor data quality. First ensembles should be used as the predictive technique since they in general are more powerful, more robust and are less sensitive to design choices than single techniques. In cases where the overall ensemble performance is insufficient, ensemble uncertainty estimation techniques, like the ones presented in 9.3, can be used to single out predictions that need special attention, thus minimizing the work of a decision maker.

When predictions with high ensemble uncertainty are analyzed, rule extraction can produce explanations for the reason behind the ensemble predictions, thus facilitating rational adjustments. The extracted rules can of course also be used as decision support for other situations, e.g. when a prediction needs to be adjusted manually due to some sudden external event, not captured by the ensemble.
The studies 9.1, 9.2, 9.4, 9.5 all show that G-REX is well suited to extract rules for both classification and estimation tasks due to several reasons demonstrated in the case studies:

- Extraction of accurate rules that are more comprehensible than rules created with standard techniques producing transparent models (study 9.1).
- Representations languages can be adapted to domain specific requirements, making them more natural and comprehensible for decision makers (study 9.5).
- Optimization of arbitrary score functions enables creation of rules that better fits the problem (study 9.2).
- Inconsistency enables enhanced uncertainty estimations which increase confidence of the extracted rules (study 9.4).
- Creation of alternative explanations which can help analyzing troublesome predictions (study 9.4).

If incorporated in the OODA-Loop, the proposed techniques would be used during the orientation step, but the results would also affect both the observe and decide steps. Ensemble uncertainty estimations will facilitate that some decision could be made automatically even when the global performance of the ensemble is insufficient. Furthermore, they may increase the trust and acceptance of the system by letting the user know that it is needed.

When manual adjustments are required, the decision maker will be supported by the extracted rules. Obviously, the extracted rules can be used to explain any prediction thus increasing the level of trust in the system. The explanations could also give insight into which variables that are extra important for the predictions. By monitoring these variables closely, during the observe phase, sudden drastic environmental changes could be caught early handled by manual adjustments.
11 Future Work

Information Fusion
A general problem for all ML-techniques is how rare and new events should be handled. As ML requires several examples to find a pattern it is impossible to do this automatically and instead domain experts have to make manual adjustments. A comprehensible model is crucial as a rational base for such adjustments but previous research have also shown that the final adjustments between domain experts and the ML-model are best done mechanically. However it is not clear how the fusion of the ML-model and the Expert should be done in the best way and this is therefore an interesting task for future research.

A domain experts could either be allowed to adjust a prediction or make an own prediction that could be fused with the original prediction. Another option is to let the domain expert continuously judge the importance of event and use these estimations as an input when creating the model. It is also unclear how new trends should be incorporated if there is not enough evidence of the trend in the data.

Parameter evaluation
Another area for use of predictive models is parameter evaluation. One example is marketing mix models where the ML-model is used to estimate the effect of different marketing efforts. Here it is also crucial that domain knowledge can be incorporated if maximal performance is to be achieved. An interesting difference is that in this case the domain expert can be used to give retroactive explanations of events that were not captured by the model. These explanations could then be used together with the original data to create a new model with higher performance thus fusing the expert knowledge with the data.

Utilize inconsistency
As seen in study 9.4 inconsistency of a rule inducer could actually be used to create improved probability estimates. It would be interesting to continue this work and evaluate if the inconsistency could be used in other ways. One example could be to create a metric that would give a decision maker a way to realize the level of ambiguity present in a data set. Thus it would be possible to better judge which confidence that should be put in models created from the data set. It could also be possible relate the ambiguity to a certain part of the found relationship. This could possibly lead to new insight of which data that needs to be added to decrease the ambiguity and improve the performance of the extracted model.
Another way to utilize the inconsistency is to simply create ensembles of accurate inconsistent models. Here the inconsistency would be a way of creating diversity without modifying the data with weights as in boosting or creating bootstraps as in bagging.
12 References


