Universiteit
Leiden
The Netherlands

Robust rules for prediction and description

Manuel Proenca, H.

Citation

Manuel Proenca, H. (2021, October 26). *Robust rules for prediction and description*. *SIKS Dissertation Series*. Retrieved from https://hdl.handle.net/1887/3220882

Version:	Publisher's Version
License:	Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from:	https://hdl.handle.net/1887/3220882

Note: To cite this publication please use the final published version (if applicable).

Robust rules for prediction and description

Hugo Manuel Proença

Keywords machine learning · data mining · rule lists · subgroup lists · subgroup discovery · pattern mining · interpretability · the Minimum Description Length (MDL) principle · Bayesian statistics.

Universiteit Leiden

SIKS Dissertation Series No. 2021-23
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

Copyright © Hugo Manuel Proença orcid.org/0000-0001-7315-5925, 2021
All rights reserved

ISBN: 978-94-6332-792-3
This work is part of the research programme Indo-Dutch Joint Research Programme for ICT 2014 with project number 629.002.201, SAPPAO, which is (partly) financed by the Netherlands Organisation for Scientific Research (NWO), in collaboration with IIT Roorkee and GE Global Research Bangalore.

Printed by: GVO Drukkers & Vormgevers B.V.
Cover: Kimber McLaughlin @pixelatedpeach

Typeset using LaTeX, diagrams generated using MATPLOTLIB and SEABORN.

Robust rules for prediction and description

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof.dr.ir. H. Bijl,
volgens besluit van het college voor promoties
te verdedigen op dinsdag 26 oktober 2021
klokke 10.00 uur

door

Hugo Manuel Proença

geboren te Hong Kong
in 1990

Promotiecommissie

Promotor:	Prof.dr. T.H.W. Bäck
Co-promotor:	Dr. M. van Leeuwen
Overige leden:	Prof.dr. A. Plaat
	Prof.dr.ir. N. Mentens
	Prof.dr. P.D. Grünwald (CWI, The Netherlands)
	Prof.dr. A.P.J.M. Siebes (Universiteit Utrecht, The Netherlands)
	Prof.dr. J. Vreeken (CISPA Helmholtz Center for Information Security, Germany)

aos meus pais

"You look at where you're going and where you are and it never makes sense, but then you look back at where you've been and a pattern seems to emerge. And if you project forward from that pattern, then sometimes you can come up with something."

Robert M. Pirsig in *Zen and the Art of Motorcycle Maintenance*

Contents

List of symbols **iv**

List of acronyms **viii**

1 Introduction **1**
1.1 Predictive rule lists . . . 3
1.2 Subgroup lists . . . 4
1.3 Research question and contributions . . . 6
1.4 Outline of this dissertation . . . 8
1.5 Publications . . . 9

2 Preliminaries **11**
2.1 Introduction to rules . . . 11
2.2 Supervised data . . . 14
2.3 Association rules, predictive rules and subgroups . . . 18
2.3.1 Interpretation as probabilistic rule . . . 19
2.3.2 Maximum likelihood estimation . . . 20
2.4 Rule lists, predictive rule lists, and subgroup lists . . . 21
2.5 Classification performance measures . . . 23
2.6 Subgroup discovery measures . . . 27
2.6.1 Top-k quality measures . . . 28
2.6.2 Weighted Kullback-Leibler divergence . . . 28
2.7 Subgroup set discovery measures . . . 30

3 MDL for rule lists **33**
3.1 The Minimum Description Length (MDL) principle 36
3.2 Model encoding 36
3.3 Data encoding 38
3.3.1 Two types of data encoding 40
3.4 Data encoding: nominal target variables 41
3.4.1 Encoding categorical distributions with *known* parameters . . . 42
3.4.2 Encoding categorical distributions with *unknown* parameters . . 42
3.4.3 Relationship of MDL-optimal subgroup lists to WKL-based SD . 44
3.4.4 Relationship of MDL-optimal subgroup lists to Bayesian testing 46
3.5 Data encoding: numeric target variables 46
3.5.1 Encoding normal distributions with *known* parameters 47
3.5.2 Encoding normal distributions with *unknown* parameters 48
3.5.3 Relationship of MDL-optimal subgroup lists to WKL-based SD . 50
3.5.4 Relationship of MDL-optimal subgroup lists to Bayesian testing 52
3.6 A new measure for subgroup sets: the sum of WKL divergences 52
3.7 Theoretical difference between subgroup list and predictive rule list . . 53

4 Discovering predictive rule lists with CLASSY **57**
4.1 Related work 59
4.1.1 Rule-based classifiers 60
4.1.2 Pattern mining 61
4.1.3 MDL-based data mining 61
4.2 The CLASSY algorithm 62
4.2.1 Separate-and-conquer greedy search 62
4.2.2 Compression gain 63
4.2.3 Candidate generation 65
4.2.4 Finding good rule lists 65
4.2.5 Time and space complexity 66
4.3 Empirical evaluation 67
4.3.1 Compression versus classification 70
4.3.2 Candidate set influence 71
4.3.3 Classification performance 74
4.3.4 Interpretability 75
4.3.5 Statistical significance testing 77
4.3.6 Overfitting 79
4.3.7 Runtime 79
4.3.8 Discussion 79
4.4 Conclusions 83

5 Discovering subgroup lists with RSD **85**
5.1 Related work . . . 87
5.1.1 Subgroup discovery . . . 87
5.1.2 Pattern mining . . . 90
5.1.3 MDL in pattern mining . . . 91
5.1.4 Algorithmic comparison in the literature . . . 91
5.2 The RSD Algorithm . . . 93
5.2.1 Algorithm high-level description . . . 93
5.2.2 Compression gain . . . 94
5.2.3 Statistical testing interpretation of compression gain . . . 95
5.2.4 Beam search for subgroup generation . . . 96
5.2.5 The Robust Subgroup Discoverer algorithm . . . 98
5.2.6 Time and space complexity . . . 99
5.3 Empirical evaluation . . . 101
5.3.1 Influence of RSD hyperparameters . . . 102
5.3.2 Setup of the subgroup quality performance comparisons . . . 102
5.3.3 Nominal target results . . . 105
5.3.4 Numeric target results . . . 106
5.3.5 Runtime comparison . . . 107
5.4 Case Study: Hotel Bookings . . . 111
5.5 Case study: flight delay analysis . . . 112
5.5.1 Analysis of subgroups obtained with RSD . . . 113
5.6 Case study: socioeconomic background and university performance . . . 117
5.6.1 Analysis of subgroups obtained with RSD . . . 117
5.7 Conclusions . . . 122

6 Conclusions **123**
6.1 Summary . . . 124
6.2 Discussion . . . 125
6.3 Future Work . . . 127
6.3.1 Short and medium-term research . . . 127
6.3.2 Long-term research . . . 128

Appendices **131**

Appendix A Kullback-Leibler divergence between two normal distributions 133

Appendix B Prequential plug-in encoding for rule lists with categorical distributions 135

Appendix C Normalized Maximum Likelihood for rule lists with categorical distributions 139

Appendix D Bayesian encoding of a normal distribution with mean and standard deviation unknown 143

Appendix E Bayesian encoding convergence to BIC for large n 147

Appendix F Datasets used for classification experiments 149

Appendix G RSD supplementary empirical evaluation 151
G.1 Datasets used for subgroup discovery experiments 151
G.2 Analysis of RSD compression gain hyperparameter 154
G.3 Analysis of RSD beam search hyperparameters 156
G.4 Results of non-sequential subgroup set discovery algorithms 159

Bibliography 161

Samenvatting 173

Summary 175

Resumo 177

List of publications 179

Acknowledgements 181

Titles in the SIKS dissertation series since 2011 185

Curriculum Vitae 207

List of symbols

Supervised Dataset

D	Labelled dataset.
$\mathbf{X}$	Dataset of explanatory variables of D.
X	An explanatory variable of $\mathbf{X}$.
$\mathcal{X}$	Domain of X.
$\mathbf{x}$	A explanatory variables sample of $\mathbf{X}$.
x	The value of sample $\mathbf{x}$ for variable X.
$\mathbf{Y}$	Dataset of target variables of D.
Y	An target variable of $\mathbf{Y}$.
$\mathcal{Y}$	Domain of Y.
$\mathbf{y}$	A target variables sample of $\mathbf{Y}$.
y	The value of sample $\mathbf{y}$ for variable Y.
i	Index for subsetting by row.
j	Index for subsetting by column.
v	A generic explanatory variable.

k Number of classes of a nominal target variable.

n Number of examples in dataset D.

m Number of explanatory variables.

t Number of target variables.

d Subscript associated with dataset distribution or defaul rule.

Model classes

M Generic model (either a rule list or a subgroup list).

RL Rule list model (including rules R and default rule).

R Rules in model RL.

r A rule.

SL Subgroup list model (including subgroups S and default rule).

S Subgroups in model SL.

s A subgroup.

ω Number of rules/subgroups in M.

a Description of a subgroup s or a rule r.

a_i Description of the i^{th} rule/subgroup in model M.

$D^a = \{\mathbf{X}^a, \mathbf{Y}^a\}$ Samples of dataset D covered by description a.

$D^i = \{\mathbf{X}^i, \mathbf{Y}^i\}$ Samples of dataset D covered by the i^{th} description in model M.

$Dist(\Theta)$ Generic probability distribution with parameters Θ.

$\mathcal{N}(\mu; \sigma)$ Normal probability distribution with parameters μ and σ.

$Cat(p_1, \cdots, p_k)$ Categorical probability distribution with p_i probability per category.

μ Mean value parameter.

σ Standard deviation parameter.

δ Effect size (ratio of μ and σ).

$\hat{\theta}$ Maximum likelihood estimation of parameter θ.

Subgroup Discovery

$q(a)$ Subgroup discovery quality measure.

$Q(S)$ Subgroup set discovery quality measure.

$f(\hat{\Theta}^a, \hat{\Theta}^d)$ Function of differences between distribution $\hat{\Theta}^a$ and $\hat{\Theta}^d$.

α Tradeoff between subgroup coverage and distribution difference.

KL Kullback-Leibler divergence general form.

KL_{Cat} Kullback-Leibler divergence for categorical distributions.

KL_{μ} Kullback-Leibler divergence for location distributions.

$KL_{\mu,\sigma}$ Kullback-Leibler divergence for normal distributions.

WKL Weighted Kullback-Leibler divergence general form.

SWKL Sum of Weighted Kullback-Leibler divergences.

MDL

$L(\cdots)$ Length of encoding.

ℓ Log-likelihood.

$L_{\mathbb{N}}$ Universal code of integers.

$L_{NML}(Y_j^i)$ Normalized Maximum Likelihood length of encoding of data Y_j^i.

$\mathcal{C}(n_a, k)$ Multinomial distribution complexity with n_a points and k categories.

L_{Bayes} Bayesian length of encoding with improper priors.

$Y^{i|2}$ The two points that make the Bayesian encoding proper.

$L_{Bayes2.0}$ Bayesian length of encoding made proper with first 2 points.

$\Gamma(n)$ Gamma function, the extension of the factorial to real numbers.

Algorithm

$\Delta_\beta L(D, M \oplus a)$ Compression gain of adding description a to model M.

β Level of normalization of the compression gain.

ζ Set of all items (possible single conditions) in $\mathbf{X}$.

$stats$ Statistics of a subgroup.

d_{max} Beam search maximum depth of search.

w_b Beam search beam width.

n_{cut} Number of cut points for numeric discretization.

Acronyms

AUC Area Under the receiver operator Curve.

BIC Bayesian Information Criterion.

BTS Bureau of Transportation Statistics.

CART Classification And Regression Trees.

CORELS Certifiably Optimal RulE ListS.

CRS Computerized Reservation System.

DSSD Diverse Subgroup Set Discovery.

EDA Exploratory Data Analysis.

EWR NEWaRk liberty international.

FN False Negative.

FP False Positive.

FPR False Positive Rate.

FSSD Fast and efficient algorithm for Subgroup Set Discovery.

FURIA Fuzzy Unordered Rule Induction Algorithm.

IDS Interpretable decision sets.

KDD Knowledge Discovery from Data.

KL Kullback-Leibler divergence.

MCMC Markov Chain Monte Carlo.

MCTS Monte Carlo Tree Search.

MCTS4DM Monte Carlo Tree Search for Data Mining.

MDL Minimum Description Length.

NML Normalized Maximum Likelihood.

RIPPER Repeated Incremental Pruning to Produce Error Reduction.

RSD Robust Subgroup Discoverer.

RSS Residual Sum of Squares.

SaC Separate and Conquer.

SAPPAO a Systems APproach towards data mining and Prediction in Airlines Operations.

SBRL Scalable Bayesian Rule Lists.

SD Subgroup Discovery.

SISD Subjectively Insteresting Subgroup Discovery.

SSD Subgroup Set Discovery.

SVM Support Vector Machine.

SWKL Sum of Weighted Kullback-Leibler divergences.

TN True Negative.

TP True Positive.

TPR True Positive Rate.

UA United Airlines.

WKL Weighted Kullback-Leibler divergence.

WRAcc Weighted Relative Accuracy.

1 Introduction

Rules are an essential part of what makes us humans. They are prime methods of information storage and sharing, employed every day to assimilate complex ideas into more manageable chunks of information. Their use can be found everywhere, from the mental note "if I do not put the alarm, then I will not wake up" to the intricate system of clinical diagnosis rules employed by physicians. A simple example is a clinical rule for diagnosing the flu, given by "if a person has either a fever or sore throat (between others), then she has the flu.".
A rule does not need to be always correct, but in most cases, it should; otherwise, it would not contain the essential information about the problem. Nonetheless, their most compelling property is that they are easy to understand, i.e., interpretable. Thus, it should not come as a surprise that researchers have long used them to describe the world, be it in machine learning or data mining.

While machine learning is concerned with finding a representation from data that can predict current and future events, data mining is concerned with extracting interesting information from data. Even though both tasks are intertwined and rule utilization is extensive in both fields, they stem from different intentions; hence, they arrive at different outcomes. In machine learning, the combination of several rules forms a model that makes predictions about future events. In data mining, sets of rules describe patterns in data that are worth seeing.
In the wake of deep learning successes, one can question if rule-based models still have a place; after all, deep learning models seem to make better predictions. Nevertheless, the complex nature of deep learning from which its success stems is also its main limitation as its models are inscrutable and not accountable. For this reason, it is imperative to find algorithms that can learn high-quality rule-based models from

data that are competitive in prediction while at the same time interpretable.

Even though research on rule-based models started more than half a century ago, many questions remain to be answered, such as: What is an optimal set of rules? What is the relationship between rules in data mining and machine learning? Can we guarantee that the models are statistically robust before seeing future data?
To answer such questions, we apply the Minimum Description Length (MDL) principle to rule-based models, which objectively quantifies the quality of models and guarantees statistical robustness. Based on information theory, the principle states that the best model is the simplest that describes the data well. This idea is a formal restatement of Occam's Razor, the law of parsimony that directly relates to the notion that a good rule, but now a set of rules, should only describe what matters most in the data for a particular task.

More specifically, we focus on ordered rule sets, i.e., rule lists. These are the first rule-based model invented, and—compared to their unordered counterparts—they have appealing mathematical properties that allow for a suitable formulation according to the MDL principle. This dissertation establishes a better understanding of rules and rule lists in machine learning and data mining. To distinguish between both, rule lists are called predictive rule lists in machine learning and subgroup lists in data mining. Our focus in machine learning is on supervised learning and in data mining on subgroup discovery. For the less acquainted with the last topic, subgroup discovery is the task of finding descriptions of data subsets—rules in tabular data—that deviate from "normal behaviour" for a target variable. In both cases, the MDL principle formalizes their optimality for a given dataset.

Motivation. The research conducted in this dissertation was in part motivated by the real-world problem of flight delays, and in specific by the SAPPAO (a Systems APproach towards data mining and Prediction in Airlines Operations) project. Its objective was to integrate flight delay predictions in optimizing airplane and crew schedules to reduce fuel and crew costs and decrease unnecessary CO2 emissions. In our part of the project, we focused on the characterization of subgroups of flights with above-average delays. We show an example of utilizing our theory and algorithms in publicly available datasets in Section 5.5.

1.1 Predictive rule lists

Interpretable machine learning has recently witnessed a strong increase in attention [26], both within and outside the scientific community, driven by the increased use of machine learning in industry and society. This is especially true for applications domains where decision making is crucial and requires transparency, such as in health care [81, 68] and societal problems [67, 126].

While it is of interest to investigate how existing 'black-box' machine learning models can be made transparent [104], the trend towards interpretability also offers opportunities for data mining, or *Knowledge Discovery from Data* (KDD), as this field traditionally has a stronger emphasis on intelligibility.

In recent years several interpretable approaches have been proposed for supervised learning tasks, such as classification and regression. Those include approaches based on prototype vector machines [95], generalized additive models [84], decisions sets [69, 122], and predictive rule lists [81, 125]. Restricting our focus to classification, we make two important observations. First, we observe that state-of-the-art algorithms [69, 122, 81, 125, 5] are designed for binary classification; no interpretable methods specifically aimed at multiclass classification have been proposed, despite being a common scenario in practice. Multiclass classification is more challenging because of 1) the increased complexity in model search, due to the uncertain consequences of favouring one class over the others, and 2) the lack of possibilities to prune the search such as commonly used when finding, e.g., decision lists [5] or Bayesian rule lists [125] for binary classification. Our second observation is that although current methods based on rules [81, 125] and decision sets [69, 122] are effective, they tend to have 1) a fair number of hyperparameters that need to be fine-tuned and 2) limited scalability. Especially the need for hyperparameter tuning can be problematic in practice, as it requires significant amounts of computation power and data (i.e., not all data can be used for training, as a substantial part has to be reserved for validation).

To address these shortcomings, *we introduce a novel approach to finding interpretable, probabilistic multiclass classifiers that requires very few hyperparameters and results in compact yet accurate classifiers*. In particular, we will show that our method naturally provides a desirable trade-off between model complexity and classification performance without the need for hyperparameter tuning, which makes the application of our approach very straightforward and the resulting models both adequate classifiers and easy to interpret.

We use probabilistic rule lists, as both the antecedent of a rule (i.e., a *pattern*) and its consequent (i.e., a probability distribution) is interpretable [81]. Using a probabilistic model has the additional advantage that one cannot only provide a crisp prediction,

but also make a statement about the (un)certainty of that prediction. Note that, given a set of ordered patterns, we can trivially estimate the corresponding consequent probability distributions from the data. The remaining question, then, is how to select a set of patterns that together form an interpretable rule list.

Interpretable rule list discovery. Informally, the problem of finding interpretable rule lists for prediction is: how to select a *compact* set of rules that together define a predictive rule list that is *accurate* yet it does not *overfit*. Overfitting is not only important to ensure generalizability beyond the observed data, but it also aligns with keeping the models as compact as possible: larger models are harder to interpret by a human analyst [56] and more prone to overfit. Another layer of interpretability that we consider is that the algorithm used to find these rule lists does not have many hyperparameters, and thus does not require much human intervention to obtain good and reliable models.

Recent optimization [69] and Bayesian [125] approaches to obtain interpretable rule lists for classification heavily rely on hyperparameters to achieve this, but those need to be tuned by the analyst and we specifically aim to avoid this.
To accomplish this, the solution that we propose is based on the MDL principle [107, 48].

1.2 Subgroup lists

Exploratory Data Analysis (EDA) [118] aims at enhancing its practitioner's natural ability to recognize patterns in the data being studied. The more she explores the more she discovers, but also the higher the risk of finding interesting results arising out of coincidences, as, e.g., spurious relations between variables that have no connection in the real world. Intuitively this corresponds to testing multiple hypothesis without realizing it. This duality of EDA requires a thorough analysis of results and highlights the need for statistically robust techniques that allow us to explore the data in a responsible way. While EDA encompasses all techniques referring to data exploration, *Subgroup Discovery* (SD) [63, 8] is the subfield concerned with discovering interpretable descriptions of subsets of the data that stand out with respect to a given target variable, i.e., *subgroups*. In this dissertation, we aim at improving the discovery of subgroup lists, i.e., ordered sets of subsets, that describe different regions of the data while being statistically robust at an individual level and as a whole.

Subgroup discovery (SD) can be seen as the exploratory counterpart to rule learning or

association rule mining, where the targets/consequent of the rules are fixed, and rules are ranked according to quality measures combining subgroup size and deviation of the target variable(s) with respect to the overall distribution in the data. In its traditional form, subgroup discovery is also referred to as top-k subgroup mining [8], which entails mining the k top-ranking subgroups according to a *local* quality measure and a number k selected by the user. Since its conception, subgroup discovery has been developed for various types of data and targets, e.g., nominal, numeric [45], and multi-label [72] targets. SD has been applied in a wide range of different domains [52, 8], such as identifying the properties of materials [43], unusual consumption patterns in smart grids [60], identifying the characteristics of delayed flights [98], and understanding the influence of pace in long-distance running [23].

Even though SD appeals to several domains, top-k mining traditionally suffers from three main issues that make it impractical for many applications: $1)$ poor efficiency of exhaustive search for more relevant quality measures [12]; $2)$ *redundancy* of mined subgroups, i.e., the fact that subsets with the highest deviation according to a certain *local* quality measure tend to cover the same region of the dataset with slight variations in their description of the subset [75]; $3)$ lack of generalization or statistical robustness of mined subgroups [77]. In this dissertation, we focus on the last two issues together: lowering *redundancy* by finding small lists of subgroups that describe the differences in the data well; and obtaining *statistically robust* subgroups. First, we define what an optimal subgroup list is using the MDL principle. Second, we propose a greedy algorithm that finds good subgroup lists using a *local* objective that is equivalent to maximizing Bayesian one-sample proportions, multinomial or t-test between each subgroup's distribution and the dataset marginal distribution, for binary, nominal or numeric data, respectively, plus a penalty for multiple hypothesis testing.

In recent years both issues have been partially addressed, mostly independent of each other; we next briefly discuss recent advances and limitations.

In terms of *redundancy*, the first main limitation of existing works is their focus on one type of target variables, such as binary targets [14, 10], nominal targets [71], or numeric targets [83], where only DSSD focuses on univariate and multivariate nominal and numeric targets [75]. The second main limitation is the lack of an optimality criterion for subgroup sets or lists, where the only exception is FSSD [10]. It is important to emphasize that some works aim to find sequential subgroups or subgroup *lists*, while others aim to find unordered sets or subgroup *sets*. Subgroup lists are akin to predictive rule lists [96] in the sense that each subgroup needs to be interpreted sequentially and they are not allowed to overlap, while subgroup sets are allowed to overlap. In this chapter, we focus solely on subgroup lists, and although previous works often did not use this term, we retroactively rename those models that are in

fact subgroup lists.
In terms of *statistical robustness*, most existing approaches consider first mining the top-k subgroups and then post-processing them in terms of a statistical test to find if the discovered subgroups are statistically significant [30, 77].

Robust subgroup discovery. Informally the problem of robust subgroup discovery is to define and find the *globally* optimal set or list (i.e., an ordered set) of non-redundant subgroups that together explain the most relevant *local* deviations in the data with respect to specified target variables. As finding the optimal set or list will typically be practically infeasible, the secondary problem is to construct an algorithm that efficiently mines "good" subgroup sets or lists from the data that retains as much from the *global* formulation's statistical properties as possible.

In this dissertation we restrict our focus to finding *subgroup lists*, because 1) they were one of the first model classes proposed for subgroup set discovery [71]; 2) they allow for an optimal formulation based on the MDL principle due to its property of unambiguously partitioning the data into non-overlapping parts; and 3) finally, they allow an ordered interpretation of the subgroups, i.e., from most to least relevant discovered subgroup.

1.3 Research question and contributions

This dissertation attempts to answer one overarching research question:

How to learn robust and interpretable rule-based models from data for machine learning and data mining, and define their optimality

In pursuit of valid answers to this question, this dissertation presents contributions on five topics: 1) predictive rule lists; 2) subgroup lists; 3) MDL learning theory; and 4) the difference between predictive rules and subgroup discovery rules.

Our contributions on **predictive rule lists** and **machine learning** are the following:

1. *Interpretable predictive rule lists using MDL* (Chapter 3) – We define optimal predictive rule lists for single- and multi-target classification and regression using the MDL principle. For classification, we derive two optimal encodings: the prequential plug-in; and the Normalized Maximum Likelihood (NML) (Section 3.4.3). For regression we use a Bayesian encoding with non-informative priors (Section 3.5.3).

2. CLASSY *algorithm* (Chapter 4) – We propose a heuristic algorithm for finding good predictive rule list for multiclass classification. The algorithm combines a frequent pattern mining algorithm to mine all the candidate rules with a greedy search to sequentially add rules to a list. Technically, CLASSY only has one hyperparameter, the candidate rules taken as input to find the rule list. It is empirically shown that Classy outperforms RIPPER, C5.0, CART, and Scalable Bayesian Rule Lists (SBRL) [125] when it comes to the combination of classification performance and interpretability.

Our contributions on **subgroup lists** and **subgroup discovery** are the following:

3. *Subgroup list model class* (Chapter 2) – We define the subgroup list model class over a tabular dataset in general, providing a *global* probabilistic formulation for the problem of sequential subgroup mining, and in particular for univariate and multivariate, nominal and numeric targets.

4. *Robust subgroup lists using MDL* (Chapter 3) – We define optimal subgroup lists using the MDL principle, where we resort to the optimal Normalized Maximum Likelihood (NML) encoding for nominal targets (Section 3.4) and the Bayesian encoding with non-informative priors for numeric targets (Section 3.5). Notably, we show that this problem formalization is equivalent to the standard definition of top-1 subgroup discovery with Weighted Kullback-Leibler (WKL) divergence as quality measure for the case of a subgroup list with one subgroup (Section 3.4.3 for nominal targets and Section 3.5.3 for numeric targets).

5. *RSD algorithm* (Chapter 5) – We propose the *Robust Subgroup Discoverer* (RSD) algorithm, which combines beam search to find subgroups with greedy search to iteratively add the best found subgroup to the subgroup list (Section 5.2). We show that the greedy objective is equivalent to a one-sample Bayes proportions, multinomial, or t-test (for binary, nominal or numeric targets, respectively) plus a penalty to compensate for multiple hypothesis testing (Section 3.4.4 for binary and nominal targets, Section 3.5.4 for numeric targets, and Section 5.2.3 for the greedy objective of RSD).

The contributions on **MDL learning theory** are the following:

6. *Prequential plug-in code for partition models* – Derivation of the prequential plug-in asymptotically optimal encoding, a refined MDL encoding, for model classes that partition the data for nominal target variables—subgroup lists, rule lists, trees, etc. (presentation in Section 3.4 and full derivation in Appendix B).

7. *Normalized Maximum Likelihood for partition models* – Derivation of the Normalized Maximum Likelihood (NML) optimal encoding, a refined MDL encoding,

for model classes that partition the data for nominal target variables—subgroup lists, rule lists, trees, etc. (presentation in Section 3.4 and full derivation in Appendix C).

8. *Bayesian encoding of normal distributions for partition models* – Derivation of a Bayesian optimal encoding of normal distributions with non-informative priors for numeric targets (presentation in Section 3.5 and full derivation in Appendix D). It is shown that for large number of instances it converges to the BIC (Appendix E). Similarly to the prequential and NML encodings, it can be used by any model class that unambiguously partitions the data, such as subgroup lists, rule lists, trees, etc.

9. *Greedy MDL algorithms maximize local statistical test* (Chapter 5) – We show that the greedy gain commonly used in the MDL for pattern mining literature can be interpreted as an MDL equivalent to a *local* Bayesian hypothesis test, a.k.a. Bayesian factor, on the likelihood of the data being better fitted by the greedy extended model versus the current model, plus a penalty for the extra model complexity (Section 5.2.3).

Finally, our contribution on the difference between **predictive rules** and **subgroup discovery rules**:

10. *Subgroups discovery versus rule-based prediction* – We demonstrate the difference between the formal objectives for subgroup discovery and predictive rule models, such as classification rule lists, from the perspective of our MDL-based approach (Section 3.7).

1.4 Outline of this dissertation

The structure of this dissertation is as follows. Chapter 2 presents the fundamental problem definitions and mathematical notation necessary to understand later chapters. It starts with a gentle introduction of association rules in rule-based classifiers, subgroup discovery, and subgroup set discovery, posteriorly formalizing these tasks for supervised data. Moreover, it presents the rule list model class and specializes this generic model class to the predictive rule list in machine learning and the subgroup list in subgroup discovery. Then, it shows how to empirically measure the quality of classification models, subgroups, and subgroup sets.
Chapter 3 presents how to encode rules, predictive rule lists, and subgroup lists using the MDL principle for univariate and multivariate nominal and numeric target variables. Then, it proceeds to prove the equivalence of MDL-based subgroup lists with

one subgroup and the standard definition of subgroup discovery—top-1 mining—with the Weighted Kullback-Leibler (WKL) divergence as a quality measure. Finally, we use our MDL formulation of predictive rule lists and subgroup lists to find the similarities and differences between rule-based prediction and subgroup discovery.
Chapter 4 introduces CLASSY, a heuristic algorithm based on the MDL principle to find good predictive rule lists for multiclass classification. Then, extensive empirical comparisons validate our proposed MDL formulation and algorithm in terms of classification performance, interpretability, overfitting, and runtime. They show that CLASSY is competitive in classification performance against state-of-the-art algorithms that produce rule-based models (or trees) while usually finding simpler models.
In Chapter 5, we propose the Robust Subgroup Discoverer (RSD) algorithm finds good subgroup lists based on our MDL formulation. It combines beam search for candidate generation with greedy search to add one subgroup at a time. Moreover, this greedy gain equals an MDL equivalent of Bayesian testing. Then, our MDL formulation and RSD show that they obtain high-quality subgroup lists on 54 datasets compared to state-of-the-art algorithms. In the end, we conduct three case studies to show how RSD works on real-world problems.
Finally, Chapter 6 presents the main conclusions of this dissertation and possible future work directions.

1.5 Publications

The chapters of this thesis are based on the following publications:

- H. M. Proença and M. van Leeuwen. Interpretable multiclass classification by mdl-based rule lists. *Information Sciences*, 512:1372–1393, 2020
- H. M. Proença, P. Grünwald, T. Bäck, and M. van Leeuwen. Discovering outstanding subgroup lists for numeric targets using mdl. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 19–35. Springer, 2020
- H. M. Proença, T. Bäck, and M. van Leeuwen. Robust subgroup discovery. *Data Mining and Knowledge Discovery (preprint available in arXiv:2103.13686)*, submitted

Other publications

- H. M. Proença, R. Klijn, T. Bäck, and M. van Leeuwen. Identifying flight delay patterns using diverse subgroup discovery. In *2018 IEEE SSCI*, pages 60–67. IEEE, 2018

2 Preliminaries

In this chapter predictive rule lists and subgroup lists are presented. To that end, we give a gentle introduction to association rules, what constitutes a rule-based classifier and subgroup discovery, and how to measure the quality of a rule-based model in classification and subgroup discovery.

This chapter is divided as follows. First, in Section 2.1 we give a high-level introduction of association rules, rule-based classifiers, subgroup discovery, and subgroup set discovery. Next, in Section 2.2 the notation for supervised structured *i.i.d.* data is presented together with a formal definition of prediction, subgroup discovery, and subgroup set discovery tasks. Then, in Section 2.3 association rules and their characteristics are introduced. After that, in Section 2.4 the rule list model class is defined in general plus the specific case of the predictive rule lists and the subgroup list. Then, in Sections 2.5, 2.6, and 2.7 performance measures for classification, quality measures for subgroup discovery, and quality measures for subgroup set discovery are introduced, respectively.

2.1 Introduction to rules

The main topics of this thesis, rule-based classification and subgroup discovery, are two paradigms arising from related fields, machine learning and data mining, respectively. Both topics share the fact that they are supervised tasks on structured data that resort to *association rules* to construct their models. Thus, we will now informally introduce what each of these tasks encompasses, starting from what they have in common, and finalizing with their differences.

Association rule. An association rule $a \mapsto b$ is an assertion of a possible relationship between the antecedent a and consequent b, which can be read in the form of "*If* a appears in the data *then* b usually also appears" with a certain level of confidence [2]. A classic example from market basket analysis is that people who buy bread and butter (antecedent), usually, also buy milk (consequent) [2]. In this case, the association rule takes the form of: $\{bread = yes\}\&\{butter = yes\} \mapsto milk = yes$. A probabilistic extension of these rules, deemed a probabilistic association rule [81], associates a parametric probability distribution to the consequent, thus, instead of having a crisp decision, it has a probability associated with each possible case:

$$a \mapsto b \sim Dist(\Theta), \tag{2.1}$$

where Θ are the parameters of the distribution $Dist$ that describe the consequent. In the case of the previous example, where the consequent is a binary variable, this could take the form of: $\{bread = yes\}\&\{butter = yes\} \mapsto milk \sim Bernouilli(p_{yes} = 0.60; p_{no} = 0.40)$; where p_{yes} is the probability of having bought milk, and $p_{no} = 1 - p_{yes}$ the probability of not having bought it. A rule is said to be active in a region of the data D if for a data instance $\mathbf{x} \in D$ its antecedent is present, such as in our example $\{bread = yes\}\&\{butter = yes\}$.

Rule-based classifiers. Classification is the task of predicting an unseen outcome y of a discrete target variable from an instance of explanatory variables $\mathbf{x}$ [36]. In order to learn the relationship between the variables, a classification model is learned from a supervised dataset $D = \{\mathbf{X}, Y\}$, which is composed of paired examples $(\mathbf{x}, y)$. Note that we only talk about rule-based classifiers and not regressors because, to the best of our knowledge, there are no competitive rule-based models for regression.

Rule-based classifiers aggregate several rules together in order to perform classification. Combining rules in different ways leads to different rule-based models, of which two stand out [39]: $1)$ *rule list or sequential activation* [109, 85]—the activation of the rules for prediction follows a pre-determined order of the form *if rule 1 then* $Dist(\Theta^1)$... *else if rule 2 then* $Dist(\Theta^2)$, finishing with a default *else* $Dist(\Theta^m)$ that captures all the data not covered by any of the previous rules; $2)$ *rule set or overlapping rules* [21]—an unordered set of rules where several individual rules can be activated at the same time, overlapping. The key difference between both models is that rule lists are ordered and only one rule is active for one data sample $\mathbf{x}$, while rule sets are unordered and multiple rules can be active for one data sample $\mathbf{x}$.
The objective of a rule-based classifier is to maximize a performance measure, thus each association rule should contribute to that *global* goal, i.e., each rule should

take into account other rules to maximize the overall quality of the classifier. Looking back at our example, we see that $\{bread = yes\}\&\{butter = yes\} \mapsto milk \sim Bernouilli(p_{yes} = 0.60; p_{no} = 0.40)$ does not seem particularly good for prediction as it does not distinguish very well between both classes. On the other hand, the rule $\{yoghurt = yes\} \mapsto milk \sim Bernouilli(p_{yes} = 0.90; p_{no} = 0.10)$ seems well suited for prediction. A note should be made in relation to decision lists, which have the same format as rule lists, but instead of combining probabilistic association rules, they are composed of decision rules, with a crisp decision as in our example $\{bread = yes\}\&\{butter = yes\} \mapsto milk = yes$, and appeared first in the literature [109].

Subgroup Discovery (SD). Subgroup discovery is the data mining task of finding subgroups that stand out with respect to some given target variable(s). The definition of standing out, also known as interestingness, is quantified by a quality measure, which depends on the task at hand [123, 63]. In general, these measures quantify quality by how different the target variable distribution of a subgroup is from what is defined as 'normal' behavior in a dataset. In the case of structured data, a subgroup generally takes the form of an association rule, and the 'normal' behavior is usually measured by the average behavior of the target variable of that dataset [8]. Going back to the market basket analysis example, let us consider a dataset made up of the shopping baskets of different clients, and that has as target variable if a client bought milk (or not). 'Normal' behavior can be given by the percentage of people that buy milk over the whole dataset, and let us assume that this value is 90%. Thus, the subgroup defined by $\{bread = yes\}\&\{butter = yes\} \mapsto milk \sim Bernouilli(p_{yes} = 0.60; p_{no} = 0.40)$ seems interesting, as compared with normal behavior, people that buy bread and butter buy milk 33% times fewer times than an average client. This is in clear contrast with rules for prediction, as subgroups that are interesting do not have to divide well between classes: they need to stand out with respect to what is 'normal' behavior in the data. Sometimes, depending on the dataset and task at hand, a good subgroup will also be a good predictive rule, but both tasks arise from different goals and should thus not be confused. In its standard form, subgroup discovery is called top-k mining, as the goal is to find the k top subgroups that maximize a user-defined quality measure. As the quality measures only quantify the individual quality of a subgroup, top-k mining is a *local* paradigm, as it is only concerned with the independent performance of the k subgroups on the respective data covered by each of their descriptions. Top-k subgroup discovery usually finds subgroups that cover the same region of the data, hence it returns redundant subgroups for many datasets. As a solution to this, *subgroup set discovery* was proposed.

Subgroup Set Discovery (SSD). The task of finding a non-redundant set of subgroups that are individually and collectively interesting at the same time is called Subgroup Set Discovery (SSD) [75]. Contrary to a predictive paradigm, the objective is that the subgroups still abide by the standard subgroup discovery principle of *locally* standing out with respect to the 'normal' behavior, while at the same time, *globally* describing different regions of the dataset. To extend subgroup discovery to its set form, two main models exist: 1) *subgroup lists or ordered sets* [71]—a set of subgroups that should be interpreted sequentially and where no subgroup is allowed to overlap in the same region of data as another, take the form of *if subgroup 1 then* $Dist(\Theta^1)$... *else if subgroup 2 then* $Dist(\Theta^2)$, etc.; and 2) *subgroup sets or overlapping sets* [74]—a set of subgroups where each subgroup can be interpreted individually and overlap is allowed according to a definition of overlap interaction. Both extensions have their advantages and disadvantages: while subgroup lists are less interpretable, they have the advantage of a clear definition of the relevance of each subgroup and which subgroup explains each data point. On the other hand, subgroup sets allow for a (semi)independent interpretation of the subgroups, but they require an extra definition that favors non-redundant sets together with a definition of the interaction of subgroups in the region where they overlap, e.g., as a mixture model.

Rule-based classifiers *versus* Subgroup Set Discovery. As was shown throughout this section, predictive rules and subgroups share a lot of the same characteristics. Rule-based classifiers aggregate association rules to maximize a *global* objective of a good overall classification, while subgroup sets balance both a *local* definition of quality with respect to the 'normal' behavior of the dataset and a *global* objective of covering different regions of the data. It is natural that for some datasets good subgroups will be good predictive rules and vice versa, but this is not always the case and it should be distinguished. Throughout this work, we will be referencing them separately to emphasize the different paradigms: *1) predictive rule* will refer to an association rule as used in rule-based models for classification in machine learning; *2) subgroup* to descriptive rules in subgroup discovery; and *3) association rule* or just *rule* to an association rule in general, i.e., when it refers to either a predictive rule or a subgroup. All their idiosyncrasies may not be apparent yet, but as we progress we will continue to emphasize their similarities and differences.

2.2 Supervised data

Consider a dataset $D = (\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}^1, \mathbf{y}^1), (\mathbf{x}^2, \mathbf{y}^2), ..., (\mathbf{x}^n, \mathbf{y}^n)\}$ of n *i.i.d.* instances. Each instance $(\mathbf{x}^i, \mathbf{y}^i)$ is composed of a vector of explanatory variable values $\mathbf{x}^i$ and a

vector of target variable values $\mathbf{y}^i$.
Each observed explanatory vector has m values $\mathbf{x} = [x_1, ..., x_m]$, one for each variable $X_1, ..., X_m$. The domain of a variable X_j, denoted $\mathcal{X}_j$, can be one of two types: nominal or numeric. Similarly, each observed target vector is composed of t values $\mathbf{y} = [y_1, ..., y_t]$, one for each target variable $Y_1, ..., Y_t$, with associated domains $\mathcal{Y}_j$. The target variables can be one of two types: nominal, or numeric. In the nominal case it is $\mathcal{Y}_j = \{1, \cdots, k\}$, with $\mathcal{Y}_j$ the set of k classes/categories of variable Y_j, and in the numeric, the domain is $\mathcal{Y}_j = \mathbb{R}$.
Note that we use subscripts on the dataset variables ($D, \mathbf{X}, \mathbf{Y}, X, Y, x, y$) to indicate column subsets and overscripts to subset over rows. In the case of other notation, such as number of elements n or statistics μ, σ we will not use the superscript as it can be confused with the exponentiation of that value. Also, X_i (resp. Y_i) refers to both the properties of the i^{th} explanatory (resp. target) variable and to all the values of this variable for a specific column. When the dataset only contains one target variable $\mathbf{Y}$ is substituted by Y.

Prediction In statistical learning, the task of prediction is to infer unseen values of a target variable from a set of explanatory variables through the use of past evidence that shows the relationship between target and explanatory variables [36]. Formally, this means that we want to find the best mapping g, from a space of possible hypotheses $\mathcal{G}$, between explanatory data $\mathbf{X}$ to target data Y (in the univariate case and without loss of generality). This mapping can be summarized as $g : \mathcal{X}_1 \times \cdots \times \mathcal{X}_m \rightarrow \mathcal{Y}$; and in the case of a probabilistic predictor, such as ours, this mapping is just a conditional probability $g(x) = \Pr(y \mid \mathbf{x} = x)$, and by abuse of notation $g(\mathbf{X}) = \{g(x^1), \cdots, g(x^n)\}$. Assuming that we are dealing with probabilistic mappings, we can now start making predictions $\hat{y}$ for the target variable values for each instance $\mathbf{x}$, by returning the outcome with the largest probability

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} \Pr(y \mid \mathbf{x}) \tag{2.2}$$

The characteristics of a good mapping are: 1) capture the properties in $\mathbf{X}$ that allow predicting Y; and 2) generalize well on previously unseen data $D_{new} = \{\mathbf{X}_{new}, Y_{new}\}$. In order to choose the best possible mapping, we need to introduce a performance measure $meas$ that empirically quantifies the quality of our mappings for a given dataset, formally $meas : \mathcal{Y}^n \times \mathcal{Y}^n \rightarrow \mathbb{R}_{\geq 0}$. Thus the problem of finding the best mapping g in a dataset $D = \{\mathbf{X}, Y\}$ reduces to:

$$g^* = \arg\max_{g \in \mathcal{G}} meas(Y, g(\mathbf{X})), \tag{2.3}$$

but then another, D_{new} is required for evaluation, as this takes into account generalization and avoids overfitting. Some examples of measures $meas$ for classification

are the accuracy or the AUC, described in Section 2.5, or the Mean Squared Error for regression.
Several variations exist, such as using only predictions $\hat{y}$ instead of $g(\mathbf{x})$ or structural measures that add an extra term to $meas$ to penalize for the structural complexity of the mapping [119]. E.g., in the case of nested mappings such as a polynomial regression, the use of higher-order polynomials is "more complex" than lower-order ones, as they have extra terms. The Minimum Description Length (MDL) principle used throughout this dissertation, is a type of probabilistic structural error minimization principle and this mapping g is called a model M or point hypothesis in it [47].

Subgroup discovery Subgroup discovery is the data mining task of discovering *unkown* patterns in the data that stand out with respect to a target variable [116]. In mathematical terms the objective is to find a mapping between descriptions a of the explanatory data $\mathbf{X}$ and the target variable Y (for the univariate case without loss of generality) that stand out in relation to the 'normal' behavior of the target variable Y. Formally, a description is a function $a : \mathcal{X}_1 \times \cdots \times \mathcal{X}_m \mapsto \{false, true\}$. And in our specific case, a description a is a conjunction of conditions on $\mathbf{X}$, each specifying a value or interval on a variable. The domain of possible conditions depends on the type of a variable: numeric variables support *greater and less than* $\{\geq, \leq\}$; nominal support *equal to* $\{=\}$. E.g., from Figure 2.2, where for the *Car import* dataset, a description can be "weight = heavy & consumption-city ≤ 8 km/L", where the variable weight is conditioned to one value (norminal variable) and $consumption - city$ is conditioned to one interval (numeric variable). As the dataset is made of pairs $(\mathbf{x}^i, y^i)$, for each description a there is an associated subset of data $D^a = \{\mathbf{X}^a, Y^a\}$ with $n_a = D^a$ instances, and an associated empirical parameter distribution of the target Y^a given by $\hat{\Theta}^a$—where the parameters depend on the distribution selected by the user. Thus, in the case of *i.i.d.* data, a subgroup is an association rule $s : a \mapsto y \sim Dist(\hat{\Theta}^a)$.
To quantify how interesting a subgroup s with description a is, we need to define a quality measure $q(n_a, \hat{\Theta}^a, \hat{\Theta}^d)$ that is a function of the subgroup empirical distribution $\hat{\Theta}^a$ and the dataset empirical marginal distribution $\hat{\Theta}^d$—'normal' behavior of the dataset.
Formally the best subgroup, or top-1 subgroup, is given by

$$s^* = \underset{s=(a,\hat{\Theta}^a)\in\mathcal{A}}{\arg\max}\ q(n_a, \hat{\Theta}^a, \hat{\Theta}^d), \tag{2.4}$$

where in the case of top-k subgroup discovery, we return the k top ranking subgroups that maximize q. An example of a quality measure for binary targets in the Weighted Relative Accuracy (WRAcc) or the Weighted Kullback-Leibler (WKL) divergence presented in Section 2.6. Contrary to *prediction*, SD does not aim at performing

well on *unseen* data, but on discovering interesting patterns in the *seen* data.

Subgroup set discovery The objective of subgroup set discovery (SSD) is to find a set of subgroups S that are both (individually) high-quality and non-redundant, i.e., cover different regions of the data [75]. Thus, it uses a *local* idea of quality measure from subgroup discovery plus a *global* concept of covering different regions of the dataset. Given this vague trade-off, SSD objectives have only been defined heuristically in different works, either by sequentially covering or by weighting instances [71, 75].
In the sequential approach one iteratively finds subgroups by: $1)$ discovering the top-1 subgroup according to quality measure as in Eq. (2.4); $2)$ removing the data covered by that subgroup—or in some cases, only the data of a certain class in binary SSD[71]—from the dataset, thus getting $D^{\neg a} = D \setminus D^a$; and $3)$ repeating 1 and 2 until the desired number is reached or no more subgroups can be found. The weighting approach follows the same iterative approach as the sequential one, but instead of removing the whole data in step 2, it reweighs each instance if it was already covered by subgroups selected in previous iterations. Formally, SSD can be defined as a dependent system of equations:

$$\begin{aligned} s_1 &= \underset{s=(a,\hat{\Theta}^a)\in\mathcal{A}}{\arg\max}\ q(n_a, \hat{\Theta}^a, \hat{\Theta}^d), \\ s_2 &= \underset{s=(a,\hat{\Theta}^a)\in\mathcal{A}|(s_1)}{\arg\max}\ \tilde{q}(n_a, \hat{\Theta}^a, \hat{\Theta}^d, s_1), \\ &\vdots \\ s_k &= \underset{s=(a,\hat{\Theta}^a)\in\mathcal{A}|(s_1,\cdots,s_{k-1})}{\arg\max}\ \tilde{q}(n_a, \hat{\Theta}^a, \hat{\Theta}^d, s_1, \cdots, s_{k-1}), \end{aligned} \tag{2.5}$$

where $\mathcal{A}|(s_1, \cdots, s_{k-1})$ represents that the space of possible subgroups and their empirical distributions depend on the subgroups found so far, $\tilde{q}$ means that the quality measure can be slightly modified by a weighting, given previous selected subgroups.
As there are no SSD global quality measures, in Section 2.7 we describe what the characteristics of a SSD quality measure $Q(S, Y)$ over a whole dataset should be, and in Section 3.6 we propose the Sum of Weighted Kullback-Leibler (SWKL) divergences as an SSD measure for sequential subgroup sets—that could in the future be adapted for non-sequential sets.

Tasks. Depending on the type (nominal or numeric) and number of targets (one or multiple), and the task at hand—rule-based prediction or subgroup discovery—the type of problem for each task can be divided into *four* categories.
First, for the case of rule-based prediction, it can be divided as: *1) classification:*

univariate nominal target; *2) regression:* univariate numeric target; *3) multi-label or multi-target classification:* multivariate binary or nominal targets, respectively; and *4) multi-target regression:* multivariate numeric targets.
Second, for the case of subgroup discovery the names are themselves self explanatory: *1) single-nominal*; *2) single-numeric*; *3) multi-nominal*; and *4) multi-numeric*.

2.3 Association rules, predictive rules and subgroups

Association rules are the shared building block of rule-based classification and subgroup discovery. To distinguish the tasks, when we mention and association rule r, we are talking about its general form and it can refer to both a predictive rule ρ and a subgroup s.

An association rule r, henceforth, called rule, consists of a *description* (also intent) that defines a *cover* (also extent), i.e., a subset of dataset D. Examples of association rules are given in Figures 2.1 and 2.2

		$\mathbf{Pr}(\boldsymbol{animaltype} = \cdots \mid \boldsymbol{a})$ in %						
description of animal	n	Mammal	Fish	Invert.	Bug	Reptile	Amph.	Bird
backbone = no	18	0	0	56	44	0	0	0

Figure 2.1: Example of one rule from the *Zoo* dataset with coverage n and one nominal target variable characterized by a categorical distribution with parameters p_i for each class in {Mammal; Fish; Invert.; Bug; Reptile; Amph.; Bird}.

		$\boldsymbol{price}$ (K)	
description of automobile specifications	n	μ	σ
weight = heavy & consumption-city $\leq$ 8 km/L	11	35	8

Figure 2.2: Example of one rule from the *Automobile import 1985* with a numeric target variable characterized by a normal distribution with coverage n, mean μ, and standard deviation σ.

Rule description: A description a is a Boolean function over all explanatory variables X. Formally, it is a function $a : \mathcal{X}_1 \times \cdots \times \mathcal{X}_m \mapsto \{false, true\}$. In our case, a description a is a conjunction of conditions on $\mathbf{X}$, each specifying a specific value

or interval on a variable. The domain of possible conditions depends on the type of a variable: numeric variables support *greater and less than* $\{\geq, \leq\}$; nominal support *equal to* $\{=\}$. The size of a description a, denoted $|a|$, is the number of conditioned variables it contains.

Example 1: In Figure 2.2, the rule description size is $|a| = 2$, with one condition on a nominal variable: {weigth $=$ heavy}; and another on a numeric variable: {consumption-city $\leq 8km/L$}.

Rule cover: The cover is the bag of instances from D where the rule description holds true. Formally, it is defined by:

$$D^a = \{(\mathbf{x}, \mathbf{y}) \in D \mid a \sqsubseteq \mathbf{x}\} = \{X_1^a, \cdots, X_m^a, Y_1^a, \cdots, Y_t^a\} = \{\mathbf{X}^a, \mathbf{Y}^a\}, \tag{2.6}$$

where we use $a \sqsubseteq \mathbf{x}$ to denote $a(\mathbf{x}) = true$. Further, let $n_a = |D^a|$ denote the coverage of the subgroup, i.e., the number of instances it covers.
Example 2 (continuation): In Figure 2.2, the rule covers 11 instances in the dataset which can be found by the instances in the data where its description is $true$, and thus its coverage is 11.

2.3.1 Interpretation as probabilistic rule

As D^a encompasses both the explanatory and target variables, the effect of a on the target variables can be interpreted as a probabilistic rule. In this thesis, we will assume that the target variables are *independent* as this simplifies the problem and is a common approach in, e.g., multi-label classification [53]. Thus, the general form of a rule is

$$a \mapsto y_1 \sim Dist(\hat{\Theta}_1^a), \cdots, y_t \sim Dist(\hat{\Theta}_t^a), \tag{2.7}$$

where y_j is a value of variable Y_j, $Dist$ is a probability distribution (defined later) and $\hat{\Theta}_j^a$ is the shorthand for the maximum likelihood estimation of the parameters of $Dist$ over values Y_j^a, i.e., $\hat{\Theta}_j^a = \hat{\Theta}_j(Y^a)$. Thus, $y_j \sim Dist(\hat{\Theta}_j^a)$ tells us that the values of variable Y_j are distributed according to a distribution $Dist$ with parameters $\hat{\Theta}_j^a$ estimated over the values Y_j^a. The vector of all parameter values of a rule is denoted by Θ^a. In our case, $Dist$ is a *categorical* or *normal* distribution for the nominal or numeric target case, respectively. For numeric targets other distributions could have been chosen, however, the *normal* distribution incorporates some of the most relevant information of the data through the mean and variance of the data, it is well studied for regression problems [36], and can be solved in a closed form from a Bayesian [58] and MDL [48] perspective. For an analysis on the direct use of the numeric empirical distribution in subgroup discovery please refer to Meeng et al. [89]. In the numeric

case, the normal distribution is represented as $\mathcal{N}(\hat{\mu}, \hat{\sigma})$, where $\hat{\mu}$ and $\hat{\sigma}$ are the mean and standard deviation of the distribution estimated from the data. In the nominal case, the distribution is $Cat(\hat{p}_1, \cdots, \hat{p}_k)$, where k is the number of classes (or categories) of the corresponding variable and $\hat{p}_c$ the estimated probability for class c.

The categorical distribution is a natural choice for describing the probabilities of classes [81] and the normal distribution captures two properties of interest in numeric variables, i.e., center and spread, while being robust to cases where the data violates the normality assumption [48].
Example 3 (continuation): Revisiting the *Automobile import* example list in Figure 2.2, the description and corresponding statistics are $a = \{\text{weight} = \text{heavy \& consumption-city} \leq 8\,\text{km/L}\}$ and $\hat{\Theta}^a = \{\hat{\mu} = 35; \hat{\sigma} = 8\}$, respectively, where the units are thousands of dollars (K). This corresponds to the following normal probability distribution:

$$\text{price (K)} \sim \mathcal{N}(\hat{\mu} = 35; \hat{\sigma} = 8)$$

Example 4 (continuation): In the case of the *Zoo* rule of Figure 2.1, the description is $a = \{\text{backbone} = \text{no}\}$, and its corresponding statistics are $\hat{\Theta}^a = \{\hat{p}_1 = 0; \hat{p}_2 = 0; \hat{p}_3 = 0.56; \hat{p}_4 = 0.44; \hat{p}_5 = 0; \hat{p}_6 = 0; \hat{p}_7 = 0\}$, where the class labels $1, ..., 7$ correspond to the animal types in the order of Figure 2.1. The target variable follows the following categorical distribution:

$$\text{animal_type} \sim Cat(\hat{p}_1 = \hat{p}_2 = \hat{p}_5 = \hat{p}_6 = \hat{p}_7 = 0.00; \hat{p}_3 = 0.56; \hat{p}_4 = 0.44)$$

2.3.2 Maximum likelihood estimation

The most common way to estimate the parameters of a probability distribution from a dataset is by using the Maximum Likelihood (ML) estimator [93]. In later chapters we also use other estimators, but the ML is still an important building block of these more complex methods.

As shown previously, each description a uniquely defines a subset D^a given by its cover Eq. (2.6). Next, we will show how to estimate the parameters for each type of target variable.
In the **nominal** case, the parameters of the distribution $Cat(\Theta^a)$ are the probabilities associated with each class c, i.e., $\Theta^a = \{p_{c=1|a}, \cdots, p_{c=k|a}\}$, for a domain $\mathcal{Y} = \{1, \cdots, k\}$. Note that we use $\cdot|a$ as a shorthand for conditional on a, as e.g., $p_{c=1|a} = \Pr(c = 1 \mid a \sqsubseteq \mathbf{x})$. Thus, for each class label c, we need to find its subset of

the data $D^{c|a}$, formally given by:

$$D^{c|a} = \{(\mathbf{x}, y) \in D^a \mid y = c\}. \tag{2.8}$$

which allows us to compute the usage over each class $n_{c|a} = |D^{c|a}|$. Now, we are in a position to use the maximum likelihood estimator for the parameters $\hat{\Theta}^a$ of each categorical distribution as:

$$\hat{p}_{c|a} = \frac{n_{c|a}}{n_a}, \tag{2.9}$$

where n_a is the total number of instances and $n_{c|a}$ is the number of instances of class c in the dataset subset D^a.
In the **numeric** case the parameters of the distribution $\mathcal{N}(\Theta^a)$ are the mean and standard deviation, i.e., $\Theta^a = \{\mu, \sigma\}$. They can be directly estimated from the target data Y^a:

$$\hat{\mu}_a = \frac{1}{n_i} \sum_{y \in Y^a} y, \tag{2.10}$$

$$\hat{\sigma}_a^2 = \frac{1}{n_i} \sum_{y \in Y^a} (y - \hat{\mu}_a)^2, \tag{2.11}$$

where $\hat{\sigma}_a^2$ is the biased estimator such that the estimate times n_a equals the Residual Sum of Squares, i.e., $n_a \hat{\sigma}_a^2 = \sum_{y \in Y^a} (y - \hat{\mu}_a)^2 = RSS_a$.

2.4 Rule lists, predictive rule lists, and subgroup lists

Sequentially aggregating rules for prediction and subgroup discovery seamlessly leads to *predictive rule lists* and *subgroup lists*, respectively. They have the same structural format and share the same model class, dubbed *rule list* model class, which takes the format of Figure 2.3, with the *only difference* being how the parameters of the last rule, also known as default rule, are chosen.

The *rule list* is an ordered set of rules, and it contains two parts: $1)$ the part of the rule list that contains the ω ordered rule descriptions $\{a_1, \cdots, a_\omega\}$, which is denoted by R for predictive rule lists and S for subgroup lists; and $2)$ the default rule r_d. Together, both parts form the whole model M. In a rule list, only one rule is activated for each instance, hence each rule only activates in a unique part (subset) of the dataset. If no rule gets activated, that instance will activate the default rule. As an example, to characterize an instance $\mathbf{x}$ with a rule list, one starts by checking the first rule and verify if $a_1 \sqsubseteq \mathbf{x}$ is $true$ or $false$. In case it is $true$, $\mathbf{x}$ belongs to that rule. In case it is $false$, we proceed to check the second rule and so forth, until finding one that returns

1:	IF	$a_1 \sqsubseteq \mathbf{x}$	THEN	$y_1 \sim Dist(\hat{\Theta}_1^1)$	$\cdots$	$y_t \sim Dist(\hat{\Theta}_t^1)$
		$\vdots$				
ω:	ELSE IF	$a_\omega \sqsubseteq \mathbf{x}$	THEN	$y_1 \sim Dist(\hat{\Theta}_1^\omega)$	$\cdots$	$y_t \sim Dist(\hat{\Theta}_t^\omega)$
default:	ELSE			$y_1 \sim Dist(\hat{\Theta}_1^d)$	$\cdots$	$y_t \sim Dist(\hat{\Theta}_t^d)$

Figure 2.3: Generic rule list model M with ω rules and t (number of target variables) distributions per rule.

$true$. In case no rule is $true$, that instance activates the default rule.

Cover of a rule in a rule list. We observe that for any given rule list of the form of Figure 2.3, *any individual instance* $(\mathbf{x}^i, \mathbf{y}^i)$ *can only be 'covered' by one rule*. That is, the cover of a_i, denoted D^a, depends on the order of the list and is given by the instances where its description occurs minus those instances covered by previous descriptions:

$$D^i = \{\mathbf{X}^i, \mathbf{Y}^i\} = \{(\mathbf{x}, \mathbf{y}) \in D \mid a_i \sqsubseteq \mathbf{x} \wedge \left(\bigwedge_{\forall_{i'<i}} a_{i'} \not\sqsubseteq \mathbf{x} \right)\}. \tag{2.12}$$

Next, let $n_i = |D^i|$ be the number of instances covered by the i^{th} description (also known as *usage*). In case an instance $(\mathbf{x}^i, \mathbf{y}^i)$ is not covered by any rule *then it is 'covered' by the default rule*. The instances covered by the default rule D^d are the ones not covered by any rule (hence the name default rule), formally defined as:

$$D^d = \{\mathbf{X}^d, \mathbf{Y}^d\} = \{(\mathbf{x}, \mathbf{y}) \in D \mid \forall_{a_i \in M} a_i \not\sqsubseteq \mathbf{x}\}. \tag{2.13}$$

Maximum Likelihood estimator. Given the partition property of the rule lists, it is straightforward to see that the ML estimators of Eq. (2.9), (2.10), and (2.11) still hold if D^a is replaced by D^i.

What predictive rule lists and subgroup lists have in common is that they are interpreted in order and that each predictive rule or subgroup distribution parameter, with the exception of the default rule, is estimated in their respective subsets D^i.

The difference between predictive rule lists and subgroup lists is how the *default rule* distributions parameter are estimated! In the case of a *predictive rule list*, the default rule is just an ordinary rule that characterizes its subset, thus its parameters $\hat{\Theta}_1^d \cdots \hat{\Theta}_t^d$ are estimated in that subset. In the case of a *subgroup list*, the default rule is fixed to the marginal distribution of the target and it is constant for a dataset. If the mean and standard deviation of a numeric target in a dataset are 18 and 13, respectively, the subgroup list default rule will be fixed at those values, independently

of the (number of) subgroups in the list. This may seem like a subtle difference, but it represents a radical difference in what defines an optimal predictive rule list or subgroup list.

The intuition is the following: in a predictive rule list, the goal is to predict as best as possible an unseen data point, thus each rule should represent homogeneous subsets of the data, and the way the default rule predicts best is if its distribution represents well its subset of the data. In a subgroup list, the goal is to find subsets of the data that have different distributions than the marginal distribution of the dataset, and the default rule covers and represents well all data that follows the marginal distribution. This, in turn, incentivizes the optimal subgroup list to have subgroups that follow distributions different than the default rule distribution, as instances well represented by the default will not be covered by subgroups. Structurally both models look very similar, but by having different definitions of optimality, each model type will favor different types of association rules.

As an *example*, one can look at Figures 2.4a and 2.4b. Given that the dataset has few distinguishing variables and few samples both predictive rules and subgroups are mostly the same, as good predictive rules are also good subgroups in this case. Nonetheless, one can see that the default rules are different. In the predictive rule list, the default rule is clearly predictive, while in the subgroup list it is the original distribution of the dataset.

2.5 Classification performance measures

Classification is a *global* predictive paradigm, thus its measures quantify the quality of a model over the whole dataset [36]. As classification is expected to generalize to unseen points, the quality of a classifier should be measured in data that is different than the one used for training and choosing the classifier. In order to achieve this and have some statistical guarantees of the generalization, each dataset is usually divided into different parts, using some of its parts to train and the rest to test. The most well-known of these techniques is called k-fold cross-validation, where one randomly divides the data into k (approximately) equal parts. Then, one uses $k-1$ parts to train the model, and the 1 part left to test the performance of that trained model, repeating the process k times, until all data was used to test. In the end, the performance over k folds is averaged out.

Classification measures can broadly be divided into two types: $1)$ aggregators of classifier decisions, such as accuracy, precision, or recall; $2)$ measures of how a classifier discriminates between classes, taking into account the confidence with which it classifies different classes, such as Area Under the receiver operator Curve (AUC) or

			$\mathbf{Pr}(\boldsymbol{animaltype} = \cdots \mid \boldsymbol{r})$ in %						
ρ	**description**	n_ρ	Mammal	Fish	Invert.	Bug	Reptile	Amph.	Bird
1	backbone = no	18	0	0	56	44	0	0	0
2	breathes = no	14	0	93	0	0	7	0	0
3	feathers = yes	20	0	0	0	0	0	0	100
4	milk = no	8	0	0	0	0	50	50	0
default rule		41*	100	0	0	0	0	0	0

(a) **Predictive rule list** for zoo dataset. Default rule with distribution estimated from its subset.

			$\mathbf{Pr}(\boldsymbol{animaltype} = \cdots \mid \boldsymbol{s})$ in %						
s	**description**	n_s	Mammal	Fish	Invert.	Bug	Reptile	Amph.	Bird
1	backbone = no	18	0	0	56	44	0	0	0
2	breathes = no	14	0	93	0	0	7	0	0
3	feathers = yes	20	0	0	0	0	0	0	100
4	milk = no	8	0	0	0	0	50	50	0
5	feathers = no	41	100	0	0	0	0	0	0
dataset distribution		0*	41	13	10	8	5	4	2

(b) **Subgroup list** for zoo dataset. Dataset rule with distribution equal to the marginal distribution of the dataset.

Figure 2.4: Illustrative example of a predictive rule list and subgroup list with the *Zoo* dataset obtained with our method. *Zoo* contains one nominal target variable with 7 classes, 101 instances, and 15 binary and 1 numeric variables. n_i refers to the number of instances covered by i^{th} predictive rule or subgroup defined by 'description'. $\Pr(animaltype = * \mid r)$ denotes the estimated probability (in %) of each class label occurring within the subgroup. *Zoo* is a highly structured dataset, thus both rule list and subgroup list found mostly the same descriptions, as in these case good subgroups are also good predictive rules. However, it is important to check that the 'default rules' are indeed different and thus incite the method to find different types of rules. $*$ concerns instances not covered by any of the five subgroups. For illustrative purposes the probabilities displayed correspond to the empirical probabilities in the data, not to the probabilities as would be obtained using the appropriate estimators.

likelihood. In this thesis, we will focus on accuracy, from the first group, as it gives an overall idea of how the classifier takes decisions and AUC from the second group. It should be noted that likelihood is more related to the MDL theory we use, but its interpretation is not as easy, and so it will not be used for ranking classifiers.

The building blocks for describing the measures are the terms defined by the intersection of the true class label and the predicted class label. We will start with the binary setting, when just two classes exist, the *positive* class, also known as the class of interest, and the *negative* class. With two classes, there are four possible characterizations of decisions based on which class it is and if it is correctly predicted by the classifier or not:

- *True Positives (TP)* - Number of instances (correctly) predicted as positive that are positive.
- *True Negatives (TN)* - Number of instances (correctly) predicted as negative that are negative.
- *False Positives (FP)* - Number of instances (incorrectly) predicted as positive that are negative.
- *False Negatives (FN)* - Number of instances (incorrectly) predicted as negative that are positive.

In the multiclass scenario, we can define all these terms per class c, where the positive class represents the class of interest c and the negative class, either represents all the other classes in a *one-versus-all* or another class in a *one-versus-one* setting. In *one-versus-all*, the performance of each class is measured by creating two classes, positive class (class of interest), and negative class, where the negative class is all classes except the positive one. In the second case, *one-versus-one* requires comparing each class against each class. In this work, we will focus on *one-versus-all* as it is the most common and simpler to understand [106]. Thus, for class c we can define *True Positive of* c (TP_c), *True Negative of* c (TN_c), *False Positives of* c (FP_c), and *False Negatives of* c (FN_c).

Accuracy. Given the previous definitions, Accuracy can be immediately defined as the ratio of correctly identified points to all points, formally:

$$Accuracy = \frac{\sum_{c \in \mathcal{Y}} TP_c}{\sum_{c \in \mathcal{Y}} TP_c + FN_c}, \tag{2.14}$$

where for the binary case we have $TP+TN$ in the numerator and $TP+TN+FP+FN$ in the denominator. Even though accuracy gives us an idea of how the classifier makes

predictions on the data, it has one main problem: if the original data is *imbalanced* it can give an erroneous idea of the quality of the classifier. As an example, if the target class is binary, and the majority class is made of 90% of the points, it is straightforward to see that an accuracy of 90% just requires us to choose all the points as the majority class, and that is not a very interesting classifier. In order to correct this, balanced accuracy was introduced [17].

Balanced Accuracy. Contrary to accuracy, balanced accuracy takes into account the classification of each class separately, by giving the same importance to each class of the positive correctly predicted ratio, True Positive Rate (TPR), or recall. Formally, it is given by:

$$bAcc = \frac{1}{|\mathcal{Y}|} \sum_{c \in \mathcal{Y}} \frac{TP_c}{TP_c + FN_c} \tag{2.15}$$

Area under the ROC Curve (AUC). The area under the receiver operator curve, which is only properly defined for binary problems, is given by the two-dimensional plot of True Positive Rate (TPR) against the False Positive Rate (FPR), as one varies the threshold T of classification. The FPR is just $FP/(FN + TP)$, and the threshold is a value above which a point is classified as belonging to the positive class, i.e., $\mathbf{x}$ is classified as positive $c = 1$ if $\Pr(1 \mid \mathbf{x}) > T$. AUC is not restricted to probabilistic classifiers but for ease of presentation we only consider these. Formally, the AUC is the area under the curve of TPR and FPR as a function of the threshold, which is given by the integral:

$$AUC = \int_0^1 TPR(T)FPR(T)\,\mathrm{d}T = \Pr(\mathbf{x}_{pos.} > \mathbf{x}_{neg.}), \tag{2.16}$$

where $T \in [0, 1]$ as we only consider probabilistic classifiers and $\Pr(\mathbf{x}_1 > \mathbf{x}_0)$ is the probability that a randomly selected positive class example will rank higher than a randomly selected negative class example in terms of [33]. An easier way to interpret the AUC is through the Wilcoxon-Mann-Whitney statistic [20], an unbiased estimator given by:

$$AUC = \frac{\sum_{\mathbf{x}_0 \in D_0} \sum_{\mathbf{x}_1 \in D_1} \mathbb{1}[\Pr(1 \mid \mathbf{x}_0) < \Pr(1 \mid \mathbf{x}_1)]}{|D_0| \cdot |D_1|}, \tag{2.17}$$

where D_0 and D_1 are the negative and positive labeled examples in D, respectively, and $\mathbb{1}$ is the indicator function that is 1 if $\Pr(1 \mid \mathbf{x}_0) < \Pr(1 \mid \mathbf{x}_1)$ and zero otherwise. Contrary to accuracy, and similarly to balanced accuracy, AUC gives the same importance to both classes. Nonetheless, in its current format, AUC is not suited for multiclass.

Multiclass AUC. To extend binary AUC to multiclass two things have to be taken into account: how to compare different classes, and how to average performance per class. To compare different classes, the two most common approaches are *one-versus-all* and *one-versus-one*. As mentioned before, in this work, we will focus on *one-versus-all*. Regarding how to average the performance per class, three methods exist: *micro* average; *macro* average; and *weighted* average. *Micro* average takes into account each example, and in the case of *one-versus-all* transforms the dataset into a binary problem, where 1 is the positive class of interest, and 0 otherwise, and computes the AUC for the whole data. *Macro* average computes one AUC per class by, for each class, transforming the dataset into a positive class versus negative class (all other classes), and then computing the average of all AUCs. *Weighted* average computes one AUC per class like the macro average, but then averages all AUCs weighted with the percentage of class examples in the data. For *macro* and *weighted* AUC (easier to present), the formulas are:

$$AUC_{macro} = \frac{1}{|\mathcal{Y}|} \sum_{c \in \mathcal{Y}} AUC(c), \tag{2.18}$$

and

$$AUC_{weighted} = \sum_{c \in \mathcal{Y}} AUC(c) \frac{|D^c|}{|D|}, \tag{2.19}$$

where $AUC(c)$ is the one-versus-all AUC for class label c and $\frac{|D^c|}{|D|}$ is the frequency of that same class label.

2.6 Subgroup discovery measures

As shown before, subgroup discovery can broadly be divided into two categories: its classic form, also known as top-k mining; and Subgroup Set Discovery (SSD). In the first, only the individual quality of a subgroup is measured, hence quality is quantified independently and *locally* for each subgroup. In the second, SSD takes into account the *local* quality of individual subgroups while also taking into a account how well they cover the whole dataset.

Contrary to classification and prediction in general, the goal of subgroup discovery is to describe the dataset well and not to measure the prediction quality on unseen data points. Thus, the quality of the subgroups or subgroup sets is traditionally measured in the dataset where the model is trained. We will first introduce the quality measures for top-k subgroup discovery, and then in Section 2.7 proceed to generalize for subgroup sets.

2.6.1 Top-k quality measures

Top-k stands for finding the k subgroups that maximize a certain quality measure [8]. To assess the quality (or interestingness) of a subgroup description a, a measure that scores subsets D^a needs to be chosen. The measures used vary depending on the target and task, but in general they have two components: 1) representativeness of the subgroup in the data, based on coverage $n_a = |D^a|$; and 2) a function of the difference between statistics of the empirical target distribution of the pattern, $\hat{\Theta}^a = \hat{\Theta}(\mathbf{Y}^a)$, and the overall empirical target distribution of the dataset, $\hat{\Theta}^d = \hat{\Theta}(\mathbf{Y})$. The latter corresponds to the statistics estimated over the whole data, e.g., in the case of the *Automobile import* subgroup list of Figure 3.3 it is $\hat{\Theta}^d = \{\hat{\mu} = 13; \hat{\sigma} = 8\}$ and it is estimated over all 197 instances of the dataset.
The general form of a quality measure to be maximized is

$$q(a) = (n_a)^{\alpha} f(\hat{\Theta}^a, \hat{\Theta}^d),\ \alpha \in [0, 1], \tag{2.20}$$

where α allows to control the trade-off between coverage and the difference of the distributions, and $f(\hat{\Theta}^a, \hat{\Theta}^d)$ is a function that measures how different the subgroup and dataset distributions are. As an example, the most commonly adopted quality measure for single-numeric targets is Weighted Relative Accuracy (WRAcc)[70], with $\alpha = 1$ and $f(\hat{\Theta}_a, \hat{\Theta}_d) = \hat{\mu}_a - \hat{\mu}_d$ (the difference between subgroup and dataset averages).

2.6.2 Weighted Kullback-Leibler divergence

Another commonly adopted measure is the Weighted-Kullback Leibler divergence (WKL) [74]. This is also the measure that we consider throughout this dissertation because of its: 1) flexibility in terms of (number and types of) supported target variables; and 2) relationship to the MDL principle (see Chapter 3).
WKL is defined as the Kullback-Leibler (KL) divergence [66] between a subgroup's and dataset target distribution $KL(\hat{\Theta}^a; \hat{\Theta}^d)$ linearly weighted by its coverage. Revisiting Eq. (2.20) this corresponds to $f(.) = KL(.)$ and $\alpha = 1$. The definition of WKL for a univariate target variable Y is given by:

$$WKL(\hat{\Theta}^a; \hat{\Theta}^d) = n_a KL(\hat{\Theta}^a; \hat{\Theta}^d), \tag{2.21}$$

where $KL(\hat{\Theta}^a; \hat{\Theta}^d)$ is the Kullback-Leibler divergence between subgroup and dataset for target Y. The KL divergence in Eq. (2.21) depends on the probabilistic model chosen to describe the target variables. In its general form the KL divergence can be defined as

$$KL(\hat{\Theta}^a_j; \hat{\Theta}^d_j) = \sum_{y \in Y^a} \Pr(y \mid \hat{\Theta}^a_j) \log \left(\frac{\Pr(y \mid \hat{\Theta}^a_j)}{\Pr(y \mid \hat{\Theta}^d_j)} \right), \tag{2.22}$$

where the logarithm is to the base two (like all logarithms used in this thesis). Thus the choice of the distribution used to describe the target is of great importance and should reflect what the user deems interesting. Now, depending of the type of target we will see show how to compute $WKL(\hat{\Theta}^a; \hat{\Theta}^d)$. It is easy to see that for multivariate targets we either use a multivariate distribution, e.g., a multivariate normal distribution, or assume that they are *independent* target variables, where the total WKL turns out to be just the sum the WKL for each target variable.

We will now provide the definitions of WKL for univariate categorical and normal distributions.

Weighted Kullback-Leibler for categorical distributions. In the case of a univariate *nominal target* Y, the distribution can be uniquely described by a categorical distribution with the probability of each category $\hat{\Theta}^a = \{\hat{p}_{1|a}, ..., \hat{p}_{k|a}\}$, so that the $KL(\hat{\Theta}^a; \hat{\Theta}^d)$ of Eq. (2.21) takes the form of

$$KL_{Cat}(\hat{\Theta}^a; \hat{\Theta}^d) = \sum_{c \in \mathcal{Y}} \hat{p}_{c|a} \log \left(\frac{\hat{p}_{c|a}}{\hat{p}_c} \right), \tag{2.23}$$

where $\hat{p}_{c|a} = \Pr(c \mid a)$ is the maximum likelihood estimate of the conditional probability of the target c given the subgroup a, and $\hat{p}_c$ is the marginal probability for that category.

Weighted Kullback-Leibler for normal distributions. In the case of a univariate *numeric* target Y, many distributions could be used for modelling. We resort to the normal distribution for its robustness and analytical properties, as mentioned before. Nonetheless, still two possibilities remain: a location distribution $\hat{\Theta}^a = \{\mu_a\}$ that only accounts for the mean, or a 'complete' normal distribution $\hat{\Theta}^a = \{\mu_a, \sigma_a\}$ that accounts for the mean and the variance. With the location distribution $KL(\hat{\Theta}^a; \hat{\Theta}^d)$ equals[1]:

$$KL_{\mu}(s) = \frac{(\hat{\mu}_d - \hat{\mu}_a)^2}{\hat{\sigma}_d}, \tag{2.24}$$

while with the normal distribution one obtains:

$$KL_{\mu,\sigma}(s) = \left[\log \frac{\hat{\sigma}_d}{\hat{\sigma}_a} + \frac{\hat{\sigma}_a^2 + (\hat{\mu}_a - \hat{\mu}_d)^2}{2\hat{\sigma}_d^2} \log e - \frac{\log e}{2} \right]. \tag{2.25}$$

Note that since $\hat{\sigma}_d$ is a constant for each dataset, there is a strong resemblance between $WKL_{\mu}(s)$ and WRAcc, where the only difference is the square of the difference of the means. Also notice that $WKL_{\mu,\sigma}$ directly takes into account the variance of a subgroup and penalizes for a larger variance, while $WKL_{\mu}(s)$ (and also

[1]The derivations of these formulas can be found in Appendix A.

WRAcc) does not take into account the variance, and thus fail to give importance to the spread of subgroup values. This is a key point as this makes a quality measure like $WKL_{\mu,\sigma}(s)$ *dispersion-aware*, while measures like $WKL_{\mu}(s)$ and $WRAcc$ are not [12].

2.7 Subgroup set discovery measures

Subgroup set discovery [75] is the task of finding a set of high-quality, non-redundant subgroups that together describe all substantial deviations in the target distribution. That is, given a quality function Q for subgroup sets and the set of all possible subgroup sets $\mathcal{S}$, the task is to find that subgroup set $S^* = \{s_1, \ldots, s_k\}$ given by $S^* = \arg\max_{S \in \mathcal{S}} Q(S, \mathbf{Y})$. Note that Q should not only take into account the individual quality of subgroups $q(a)$, but also the overlap of their coverages D^a and quantify the contribution of each instance only once, as opposed to top-k mining where only their individual qualities are taken into account, and thus there is no *global* definition of the quality of a set.

Ideally, a quality measure for subgroup sets Q should: 1) *be global*, i.e., for a given dataset it should be possible to compare subgroup set qualities regardless of subgroup set size or coverage; 2) *maximize the individual qualities* of the subgroups; and 3) *minimize redundancy* of the subgroup set, i.e., the subgroups covers should overlap as little as possible while ensuring the previous point.

Subgroup sets quality has mostly only been defined heuristically, by iteratively finding one subgroup at the time and after each discovery removing/weighting the instances covered by these [71, 75].

Nonetheless, there have been attempts to formally define the quality of a subgroup set, although, they are usually not universal, i.e., independent of the number of subgroups, and usually rely on some heuristic definitions. For example, in Knobbe and Ho [64], the authors propose to first mine the top-k patterns with k very large and then filter out a subset k' according to some measure that takes into account overlap and individual quality. This is an effective approach to find a non-redundant set, but by using top-k in the first step and fixing k', they are first, biasing the search to a certain region of the data that is highly redundant, and second, defining optimality dependent on k'. In van Leeuwen and Ukkonen [76], the authors avoid defining one measure, by treating the problem of finding a diverse set as a multi-objective, and thus using two measures instead of one, one for quality and another for diversity of subgroup set of size k. In the case of Belfodil et al. [10], the authors define a subgroup set as a disjunction of subgroups and based on that, propose a *global* measure

for this type of set. However, this subgroup set definition does not match with previous works, as a disjunction of subgroups is just a subgroup that uses both logical conjunctions and disjunctions to form a description. Thus, each set describes only the global behaviour of this definition of subgroup.

Based on this limitation, in Section 3.6 we propose a new subgroup set measure called the Sum of Weighted Kullback-Leibler (SWKL) divergences that is straightforward to compute for a subgroup list. To extend it to subgroup sets, however, it would require defining the overlap of subgroups distributions in a probabilistic format, such as through a mixture model.

3

MDL for rule lists

In this chapter[1] we formalize the task of finding predictive rule lists and subgroup lists as a model selection problem using the Minimum Description Length (MDL) principle [107, 48, 47].

In the previous chapter, we defined the rule list model, which we recall here in Figure 3.1. Now, the remaining question is how to select adequate models. For that, we resort to the MDL principle, which can be paraphrased as "*induction by compression*" and roughly states that the best model is the one that best compresses the data. The idea of compression can seem unintuitive at first. Still, one should notice that it is intimately connected to the concept of probability, i.e., the model that has the highest probability given the data is the same that maximizes compression. This idea was first formally stated by Shannon [114], which tells us that the optimal length of the encoding for an event A—smaller length corresponds to higher compression—equals the negative logarithm of the probability of that event, thus

$$L(A) = -\log \Pr(A), \tag{3.1}$$

where $L(A)$ is the length of the encoding for the event. To be objective, the MDL principle attempts to make the minimum number of assumptions about the model class. At this point, we should recall that the models we are trying to select from the data are rule lists and that, depending on the type of task, we use predictive rule lists and subgroup lists for machine learning or data mining respectively. Both models have the same model structure (that of Figure 3.1) and only differ in how the parameters of the default rule are estimated.

[1] Parts of this chapter are based on Proença and van Leeuwen [96], Proença et al. [99, 100]

In the case of a subgroup list, the default rule is fixed to the marginal distribution of each target, making its parameters *known* and *fixed* for a certain dataset [99, 100]. In the case of a predictive rule list, however, the last rule is 'free' in the sense that it depends on the estimate of its subset $\mathbf{Y}^d$ [96].

$$
\begin{array}{lllllll}
1: & \text{IF} & a_1 \sqsubseteq \mathbf{x} & \text{THEN} & y_1 \sim Dist(\hat{\Theta}_1^1) & \cdots & y_t \sim Dist(\hat{\Theta}_t^1) \\
 & & \vdots & & & & \\
\omega: & \text{ELSE IF} & a_\omega \sqsubseteq \mathbf{x} & \text{THEN} & y_1 \sim Dist(\hat{\Theta}_1^\omega) & \cdots & y_t \sim Dist(\hat{\Theta}_t^\omega) \\
\text{default:} & \text{ELSE} & & & y_1 \sim Dist(\hat{\Theta}_1^d) & \cdots & y_t \sim Dist(\hat{\Theta}_t^d)
\end{array}
$$

Figure 3.1: Generic rule list model M with ω rules and t (number of target variables) distributions per rule.

This may seem like a subtle difference, but for subgroup lists, it allows to find subgroups that always differentiate themselves from the dataset marginal distribution. In contrast, for predictive rule lists, it will enable finding predictive rules that maximize predictive performance. A theoretical proof of their difference, from an MDL perspective, is given in Chapter 3.7.

Nonetheless, all the data encodings developed in this chapter can be used for both predictive rule lists and subgroup lists. In the case of predictive rule lists, the data encodings were only empirically tested in the classification setting. In contrast, subgroup lists were tested for all the settings, i.e., univariate and multivariate nominal and numeric targets. To not burden the reader, we here present two simple examples of subgroup lists in Figures 3.2 and 3.3, which will be used throughout this chapter to exemplify the MDL encodings.

Structure of the chapter. This chapter is organized as follows. First, in Section 3.1 the MDL principle for supervised datasets is introduced. Next, in Section 3.2 the encoding of the model structure is shown. Then, in Section 3.3 the high-level encoding of the data given the model is presented. After that, the specific encodings of the data given a model for categorical and normal distributions are given in Sections 3.4 and 3.5, respectively. Then, in Section 3.6 a new subgroup set discovery measure is presented. Finally, in Chapter 3.7 the theoretical difference between rule lists and subgroup lists is studied through the MDL lens.

			$\mathbf{Pr}(\boldsymbol{animaltype} = \cdots \mid \boldsymbol{s})$ in %						
s	**description**	n_s	Mammal	Fish	Invert.	Bug	Reptile	Amph.	Bird
1	backbone = no	18	0	0	56	44	0	0	0
2	breathes = no	14	0	93	0	0	7	0	0
3	feathers = yes	20	0	0	0	0	0	0	100
4	milk = no	8	0	0	0	0	50	50	0
5	feathers = no	41	100	0	0	0	0	0	0
dataset distribution		0^*	41	13	10	8	5	4	2

Figure 3.2: *Zoo* dataset subgroup list obtained by RSD algorithm (presented in Chapter 5). *Zoo* contains one nominal target variable with 7 classes, 101 instances, and 15 binary and 1 numeric variables. n_s refers to the number of instances covered by subgroup 's' defined by 'description'. $\Pr(animaltype = * \mid s)$ denotes the estimated probability (in %) of each class label occurring within the subgroup. The bottom row shows the marginal probability distribution of the dataset. $*$ concerns instances not covered by any of the five subgroups. For illustrative purposes the probabilities displayed correspond to the empirical probabilities in the data, not to the probabilities as would be obtained using the appropriate estimator.

			price (K)	
s	**description** of automobile specifications	n_s	$\hat{\mu}$	$\hat{\sigma}$
1	weight = heavy & consumption-city $\leq$ 8 km/L	11	35	8
2	fuel-type = gas & consumption-city $\geq$ 13 km/L	45	7	1
3	weight = light & wheel-base = low	35	9	1
4	length = medium & 13 $\leq$ consumption-city $\leq$ 15 km/L	27	10	2
5	peak-rpm = medium	49	16	3
6	engine-size = medium	12	26	7
dataset overall distribution		18^*	13	8

Figure 3.3: *Automobile import 1985* subgroup list obtained with RSD algorithm (presented in Chapter 5). The dataset contains *price* as numeric target variable, 197 examples, and 17 variables. The dataset was modified, some variables removed and others discretized, for ease of presentation. n_s refers to the number of instances covered by subgroup 's' defined by 'description', $\hat{\mu}$ and $\hat{\sigma}$ its estimated mean and standard deviation for the target variable in thousands of dollars (K). $*$ concerns instances not covered by any of the five subgroups.

3.1 The Minimum Description Length (MDL) principle

As we are interested in finding compact yet good models that are statistically robust, we resort to the Minimum Description Length (MDL) [107, 48] principle. The problem of selecting a concrete model from a large space of possible models is a *point hypothesis selection* problem, for which we should use a two-part code [48].
In contrast to existing pattern-based modeling approaches (e.g., [120, 78]), we deal with a *supervised* setting in which the goal is to learn a mapping from instances to target variables. This implies that we are not looking for structure *within* instance data $\mathbf{X}$, but for structure in $\mathbf{X}$ that helps to *explain* (subgroup lists) or *predict* (predictive rule lists) $\mathbf{Y}$.
That is, to induce a mapping from instances to target variables, we should consider the instance data $\mathbf{X}$ to be given as 'input' to the model and *only encode the target variables* $\mathbf{Y}$. Clearly, this corresponds to the rule lists that we introduced in the previous chapter. Then, given the complete space of models $\mathcal{M}$, uniquely specified by all ordered sets of patterns over $\mathcal{X}$, the optimal model is the model $M \in \mathcal{M}$ that minimizes a two-part code [48], i.e.,

$$M^* = \arg\min_{M \in \mathcal{M}} L(D, M) = \arg\min_{M \in \mathcal{M}} \left[L(\mathbf{Y} \mid \mathbf{X}, M) + L(M) \right], \tag{3.2}$$

where $L(\mathbf{Y} \mid \mathbf{X}, M)$ is the encoded length, in bits[2], of target variables data $\mathbf{Y}$ given explanatory data $\mathbf{X}$ and model M, $L(M)$ is the encoded length, in bits, of the model, and $L(D, M)$ is the total encoded length and the sum of both terms. Note that this definition holds up for different models, such as rule list RL or subgroup list SL. Intuitively, the best model M^* is the model that results in the best trade-off between how well the model compresses the target data and the complexity of that model—thus minimizing redundancy and automatically selecting the best list size. This formulation is similar to that previously used for two-view association discovery [73].

3.2 Model encoding

The next step is to define the two length functions; we start with $L(M)$. Following the MDL principle [48], we need to ensure that: 1) all models in the model class, i.e., all rule lists for a given dataset, can be distinguished; and 2) larger code lengths are assigned to more complex models. To accomplish the former, we encode all elements of a model that can change, while for the latter, we resort to two different codes: when a larger value represents a larger complexity we use the universal code for integers

[2]To obtain code lengths in bits, all logarithms in this paper are to the base 2.

[108], denoted[3] $L_{\mathbb{N}}$; and when we have no prior knowledge but need to encode an element from a set, we choose the uniform code. Note that as predictive rule lists and subgroup lists have the same structure, their model can be defined in the same way, as given all the rules in M, the default rule subset is completely defined.
Specifically, the encoded length of a model M over variables in $\mathbf{X}$ is given by

$$L(M) = L_{\mathbb{N}}(|M|) + \sum_{a_i \in M} \left[L_{\mathbb{N}}(|a_i|) + \log \binom{m}{|a_i|} + \sum_{v \in a_i} L(v) \right], \tag{3.3}$$

where we first encode the number of antecedents $|M|$, which can symbolize predictive rules $|R|$ or subgroups $|S|$, using the universal code for integers, and then encode each rule description individually. For each description, first, the number $|a_i|$ of variables used is encoded, then the set of variables using a uniform code over the set of all possible combinations of $|a_i|$ from all explanatory variables, and finally the specific condition for a given variable. As we allow variables of two types, the latter is further specified by

$$L(v) = \begin{cases} \log |\mathcal{X}_v| & \text{if } v \text{ is nominal} \\ L_{\mathbb{N}|2}(n_{op}) + \log N(n_{op}, n_{cut}) & \text{if } v \text{ is numeric} \end{cases} \tag{3.4}$$

where the code for each variable type assigns code lengths proportional to the number of possible parts the variable's domain can partition the dataset. Note that this seems justified, as having more parts implies more potential spurious associations with the target that we would like to avoid. For `nominal` variables, this is given by the size of the domain, i.e., the number of categories in a nominal variable. For `numeric` variables, it equals the number of operators used $n_{op}|$[4] plus the possible number of outcomes $N(n_{op}, n_{cut})$ given the operators and n_{cut} cut points. The number of operators for numeric variables can be one or two, as there can be conditions with one (e.g., $x \leq 2$) or two operators (e.g., $1 \leq x \leq 2$), which is a function of the number of possible subsets generated by n_{cut} cut points. Note that we here assume that equal frequency binning is used, which means that knowing X and n_{cut} is sufficient to determine the cut points.

Example 5 (continuation): Let us assume that the subgroup list of the *Automobile* example of Figure 3.3 is composed of only the first subgroup. In that case the rule list only has one subgroup with description: {weight = heavy & consumption-city ≤ 8

[3] $L_{\mathbb{N}}(i) = \log k_0 + \log^* i$, where $\log^* i = \log i + \log \log i + \ldots$ and $k_0 \approx 2.865064$.

[4] Note that we use $L_{\mathbb{N}|2}$, which is how we denote the universal code for integers with codes restricted to $n = 1$ or 2. This can be obtained by applying the maximum entropy principle to $L_{\mathbb{N}}$ when it is known that it cannot take values of $n > 2$.

km/L }. Taking into account that the dataset has 17 variables, $|\mathcal{X}_{weight}| = 3$ and only 3 cut points were used for numeric attributes, the model length is given by:

$$\begin{aligned} L(M) &= L_{\mathbb{N}}(1) + L_{\mathbb{N}}(2) + \log \binom{17}{2} + \log |\mathcal{X}_{weight}| + \left[L_{\mathbb{N}|2}(1) + \log 2n_{cut} \right] \\ &= 1.52 + 2.52 + 7.09 + 1.59 + 0.77 + 2.59 \\ &= 16.08 \text{ bits} \end{aligned}$$

It is important to note that the length of the model can (and should) be a real number, as we are only concerned with the idea of compression, not with materialising and transmitting the actually encoded data [48].

3.3 Data encoding

The remaining length function is that of the target data given the explanatory data and model, $L(\mathbf{Y} \mid \mathbf{X}, M)$. In this section, we show how to encode the target data $\mathbf{Y}$ by dividing it into smaller subsets that can be encoded individually and then summed together, and why there are different types of data encoding for each of the subsets. The specifics of encoding nominal and numeric targets are described in Sections 3.4 and 3.5, respectively.

Cover of a rule in a rule list. Let us recall from Chapter 2.4 that for any given rule list of the form of Figure 3.1, *any individual instance* $(\mathbf{x}, \mathbf{y})$ *can only be 'covered' by one rule or subgroup*. That is, the cover of a description in a list a_i, denoted D^i, depends on the order of the list and is given by the instances where its description occurs minus those instances covered by previous descriptions, i.e., $a_j, \forall_{j<i}$.
In case an instance $(\mathbf{x}, \mathbf{y})$ is not covered by any pattern $a \in M$ *then it is 'covered' by the default rule*. The number of instances covered by the default rule D^d are the ones not covered by any description (hence the name default rule). The instances covered by a description, also called *usage*, are denoted by $n_i = |D^i|$, and those covered by the default rule, $n_d = |D^d|$
As every description defines an individual subset, one can estimate the parameters of its target variable distributions using the maximum likelihood estimator described in Section 2.3.2.
Note that this shows us that a rule or subgroup is fully defined by its description a_i in a dataset D, and we will interchangeably refer to rules by their descriptions and to its elements (statistics, parameters, distributions, etc.) by its index i when obvious from context.

As the default rule is the only difference between a rule list and a subgroup list, it is also the difference in their encoding. As a rule list induces a *partition of the data*, the total length of the encoded data can be given by the sum of its *non-overlapping parts*. For a **predictive rule list**, the data encoding is given by:

$$L(\mathbf{Y} \mid \mathbf{X}, M) = L(\mathbf{Y}^d) + \sum_{r_i \in R} L(\mathbf{Y}^i), \tag{3.5}$$

while for a **subgroup list** it is given by:

$$L(\mathbf{Y} \mid \mathbf{X}, M) = L(\mathbf{Y}^d \mid \mathbf{\Theta}^d) + \sum_{s_i \in S} L(\mathbf{Y}^i), \tag{3.6}$$

where $\mathbf{\Theta}^d$ is the vector of parameters for each variable $\Theta_1^d, \ldots, \Theta_t^d$ for the marginal distribution of the target variables. Observe that we dropped $\mathbf{X}^a$ as these are not necessary to encode $\mathbf{Y}^a$ but only to generate the partition of the data, and also dropped the parameters $\mathbf{\Theta}^i$ of the rules and default predictive rule as we do not know what are their parameters until we see the data. This last part will be clarified at the end of this section, where we describe how to encode subsets without knowing their parameters. As can be seen, the difference between the predictive rule list and subgroup list is that the default rule is either encoded as a regular rule, or using the dataset distribution. This amounts to a difference in optimality between predictive rule lists and subgroup lists, which emphasizes the discovery of different types of descriptions for each model class.

As a side-note, note that Eq. (3.5) concerns the encoding of any supervised partition of the data, which allows to directly quantify the quality of any tree learning method—each such tree induces a partition of the data.

Encoding data of t (assumed) independent target variables. As each target variable is assumed independent from each other, the encoding of target data is given by the sum of their individual encodings:

$$L(\mathbf{Y} \mid \mathbf{X}, M) = -\log\left(\prod_{j=1}^{t} \Pr(Y_j \mid \mathbf{X}, M)\right) = \sum_{j=1}^{t} L(Y_j \mid \mathbf{X}, M). \tag{3.7}$$

Joining (3.5) and (3.7), one obtains for **predictive rule lists**:

$$L(\mathbf{Y} \mid \mathbf{X}, M) = \sum_{j=1}^{t} \left(L(Y_j^d) + \sum_{s_i \in S} L(Y_j^i)\right) \tag{3.8}$$

and joining (3.6) and (3.7), one obtains for **subgroup lists**:

$$L(\mathbf{Y} \mid \mathbf{X}, M) = \sum_{j=1}^{t} \left(L(Y_j^d \mid \Theta_j^d) + \sum_{s_i \in S} L(Y_j^i)\right) \tag{3.9}$$

3.3.1 Two types of data encoding

Data encoding can be separated into two different categories: 1) with *known parameters*; and 2) with *unknown parameters*. In our case, *known parameters* correspond to the default rule of a subgroup list, while *unknown parameters* correspond to the predictive rules, subgroups, and default rule of a predictive rule list.

1**) Known parameters**: when the parameters of a distribution are *known*, one can encode the data points directly using the probability for those points given by the distribution with the known parameters. Thus, the encoding of points Y_j^i (j^{th} variable and i^{th} subgroup) is equal to the negative logarithm of their probability given by known parameters $\hat{\Theta}_j^i$:

$$L(Y_j^i \mid \hat{\Theta}_j^i) = \sum_{y \in Y_j^i} -\log \Pr(y \mid \hat{\Theta}_j^i), \tag{3.10}$$

which is just the minus log-likelihood of parameter $\hat{\Theta}_j^i$ given observed data Y_j^i. This type of code is used in the case of the default rule of a subgroup list, as the parameters $\hat{\Theta}_j^d$ are equal to the marginal distribution of variable Y_j and are constant for each dataset. Note that this is the *key difference between a subgroup list and a predictive rule list*: the last rule of a subgroup list is fixed to the marginal distribution, while in the predictive rule list its parameters are unknown and depend on the subset D^d.

2**) Unknown parameters**: when the parameters are *unknown* we need to encode both the parameter values and the data points. We have two possibilities: 1) crude MDL, i.e., encoding the probabilities using a suboptimal probability distribution and then applying the Shannon-Fano code, i.e., the logarithm of the empirical probability [114]; or 2) employ an optimal encoding of both parameters of the distribution and data points together [48]. In this work, we employ optimal encoding of parameters, as it guarantees optimality in the sense that the encoding is the best possible in the worst-case scenario, i.e., in case the sample of the data is not representative of the population. Three types of optimal encodings exist, which are, in increasing order of optimality guarantees: 1) *prequential plug-in*; 2) *Bayesian*; 3) *Normalized Maximum Likelihood (NML)*. While the first two are asymptotically optimal, the NML encoding is optimal for fixed sample sizes.
Depending on the target type, we employ the best encoding possible while being computationally feasible, i.e., we require adequate run-time for our algorithm. For nominal targets, we present a prequential plug-in and an NML encoding for both the probabilities of each class and the data points in Section 3.4, where the second is a theoretical improvement over the first. We resort to a Bayesian encoding for numeric targets as the NML code is not computationally feasible for that case.

3.4 Data encoding: nominal target variables

When the data have one or more nominal targets, the target distributions of the probabilistic rules (2.7) are categorical distributions $Cat(\Theta)$, each with a set of parameters $\Theta = \{p_1, \cdots, p_k\}$ representing the k classes:

$$\Pr(y = c \mid p_1, \cdots, p_k) = p_c, \text{ subject to } \sum_{c=1}^{k} p_c = 1. \tag{3.11}$$

This implies a probabilistic rule of the form:

$$a \mapsto y_1 \sim Cat(p_1, \cdots, p_k), \cdots, y_t \sim Cat(p_{1'}, \cdots, p_{k'}),$$

where k and k' are the number of classes Y_1 and Y_t, respectively. To simplify the introduction of concepts we will assume we only have one target variable in $\mathbf{Y}$, and then generalize the results to multiple variables at the end. Also in line with this simplification, we will only refer to association rules, and then, specialize in the end for both predictive rule lists and subgroup lists. Thus, throughout this section $\mathbf{Y}$ becomes Y, and the parameters of each rule r_i become $\hat{\Theta}^i = \{p_{1|i}, \cdots, p_{k|i}\}$ as there is only one variable with k classes, where $p_{1|i}$ is the probability of class 1 for subgroup i, i.e., $\Pr(c = 1 \mid a_i)$. The general form of a rule list with one nominal target takes the form of Figure 3.4.

$$\begin{array}{rllll}
r_1\text{:} & \text{IF} & a_1 \sqsubseteq \mathbf{x} & \text{THEN} & y \sim Cat(\hat{p}_{1|1}, \cdots, \hat{p}_{k|1}) \\
 & & \vdots & & \\
r_\omega\text{:} & \text{ELSE IF} & a_\omega \sqsubseteq \mathbf{x} & \text{THEN} & y \sim Cat(\hat{p}_{1|\omega}, \cdots, \hat{p}_{k|\omega}) \\
\text{default:} & \text{ELSE} & & & y \sim Cat(\hat{p}_{1|d}, \cdots, \hat{p}_{k|d})
\end{array}$$

Figure 3.4: Generic rule list model M with ω rules $\{r_1, ..., r_\omega\}$ and a single nominal target Y with k categories.

In the following sections, we will derive the data encoding with categorical distributions. First, in Section 3.4.1, it is shown how to encode a categorical distribution when its parameters are known, which is the case for the default rule of a subgroup list. After that, in Section 3.4.2 it is shown how to encode a categorical distribution when the parameters of the distribution are unknown. Then, in Section 3.4.3 the equivalence between MDL-based subgroup lists with only one subgroup and standard (top-1) subgroup discovery with WKL as a quality measure is proven. Finally, in Section 3.4.4, we show the data encoding of subgroup lists is equivalent to a Bayesian test. Note that for the next section we will also use the maximum likelihood expressions of Section 2.3.2.

3.4.1 Encoding categorical distributions with *known* parameters

To encode target values with *known parameters*—as is the case for the default rule of a subgroup list—we can directly use Eq. (3.10) with given parameter estimates $\hat{\Theta}^d = \hat{p}_{1|d}, \cdots, \hat{p}_{k|d}$ (marginal distribution over the whole dataset):

$$L(Y^d \mid \hat{p}_{1|d}, \cdots, \hat{p}_{k|d}) = \sum_{c \in \mathcal{Y}} -n_{c|d} \log \hat{p}_{c|d} = -\ell(\hat{\Theta}^d \mid Y^d), \tag{3.12}$$

where $\ell(\hat{\Theta}^d \mid Y^d)$ is the log-likelihood of the parameter set $\hat{\Theta}^d$, and $n_{c|d}$ denotes the number of points associated with each class c covered by default rule Y^d.
Note that we exemplified this code using the dataset marginal distribution parameters as these are the only *known* parameters used throughout this thesis, however, this encoding can be used with any known parameters.

3.4.2 Encoding categorical distributions with *unknown* parameters

To encode target values for which the parameters are *unknown*—as is the case for each predictive rule, subgroup, and predictive default rule—we need to encode parameters and data together. For that, we have developed two types of codes: 1) the *prequential plug-in code* that is asymptotically optimal; and 2) the *Normalized Maximum Likelihood (NML)* code that is "optimal in the sense that it achieves the minimax optimal codelength regret" [48, Part II]. The prequential plug-in code was developed earlier [96], as it is easier to use and compute. Nonetheless, the NML code enjoys better theoretical properties, and thus should be preferred when possible.

Prequential plug-in encoding. The main idea of the *prequential plug-in* code is to treat each subset of labels Y^i as sequential data and then predict each label as it arrives, starting with no knowledge about their distribution and updating it each time one receives a label. To achieve that, it requires the use of a smoothed version of the ML estimator, as before receiving any point we already need to have a probability distribution

$$\Pr_{\text{plug-in}}(y^u = c \mid Y^{|u-1}) := \frac{|\{y \in Y^{|u-1} \mid y = c\}| + \epsilon}{\sum_{c' \in \mathcal{Y}} |\{y \in Y^{|u-1} \mid y = c'\}| + \epsilon}, \tag{3.13}$$

where $Y^{|u-1}$ represents the ordered sequence of $u-1$ class labels, ϵ the pseudocount which allows us to have probabilities before seeing any label.
Intuitively, this means that one starts with a pseudocount ϵ for each possible element, constructs a code using these pseudocounts, starts encoding/sending/decoding messages one by one, and then *updates the count of each element after sending/receiving*

each individual message. The *prequential plug-in code* is asymptotically optimal even without any prior knowledge of the probabilities [48].
Taking into account that the rule list creates a partition of the data, and applying Eq. (3.13) to each class label in part, we obtain for each part[5] Y^i:

$$\begin{aligned} L_{\text{plug-in}}(Y^i) &= -\log\left(\prod_{u=1}^{n_i} \Pr_{\text{plug-in}}(y^u \mid Y^{i|u-1})\right) \\ &= -\log\left(\frac{\prod_{c=1}^{k}\prod_{u=0}^{n_{c|i}-1}(u+\epsilon)}{\prod_{j=0}^{n_i-1}(u+k\epsilon)}\right) \\ &= -\log\left(\frac{\prod_{c=1}^{k}(n_{c|i}-1+\epsilon)!/(\epsilon-1)!}{(n_i-1+k\epsilon)!/(k\epsilon-1)!}\right) \\ &= -\log\left(\frac{\prod_{c=1}^{k}\Gamma(n_{c|i}+\epsilon)/\Gamma(\epsilon)}{\Gamma(n_i+k\epsilon)/\Gamma(k\epsilon)}\right), \end{aligned} \tag{3.14}$$

where $Y^{i|u}$ is a sequence of class labels of length u in part D^i, and $n_i = |D^i|$ and $n_{c|i} = |D^{c|i}|$. Further, Γ is the gamma function, an extension of the factorial to real and complex numbers that is given by $\Gamma(u) = (u-1)!$. The most common values for ϵ, which takes the role of a prior in the Bayesian literature [125], are the Jeffrey's prior of 0.5 or the uniform prior of 1. For simplicity in our experiments, the value of $\epsilon = 1$ was used as it allows us to obtain natural factorials instead of gamma functions. It is interesting to note two things: 1) we started with a sequential idea, but the final encoding of Eq. (3.14) is independent of the order in which the data is processed; and 2) for the case of categorical and multinomial distributions the prequential plug-in code is equivalent to a Bayesian code with a Dirichelet prior [48, Chapter 9]

NML encoding. The expression of the NML code can be daunting, but its intuition is very clear [65], i.e., the NML code is equivalent to first encoding all maximum likelihood estimates of sequences Z of n_i points based on their likelihoods, and then encoding data Y^i with its maximum likelihood estimate $\hat{\Theta}^i$ as in Eq. (3.12). Formally, the NML code length of the subset Y^i is given by[6]:

$$\begin{aligned} L_{NML}(Y^i) &= -\log \frac{\prod_{y\in Y^i}\Pr(y \mid \hat{\Theta}^i)}{\sum_{Z\in\mathcal{Y}^{n_i}}\prod_{z\in Z}\Pr(z \mid \hat{\Theta}^Z)} \\ &= \sum_{c\in\mathcal{Y}} -n_{c|i}\log\hat{p}_{c|i} + \log\sum_{Z\in\mathcal{Y}^{n_i}}\prod_{z\in Z}\Pr(z \mid \hat{\Theta}^Z) \\ &= -\ell(\hat{\Theta}^i \mid Y^i) + \mathcal{C}(n_i, k) \end{aligned} \tag{3.15}$$

[5]For full details and intuition on the derivations of the prequential plug-in code check Appendix B.
[6]For details on the derivation of Eq. (3.15), please see Appendix C.

where $\mathcal{Y}^{n_i}$ is the space of all possible sequences of n_i points with cardinality $k = |\mathcal{Y}|$ (possible values per point), $\hat{\Theta}^Z$ is the maximum likelihood estimate over Z, $\mathcal{C}(n_i, k)$ is the complexity—as it is called in MDL literature[48]—of the multinomial distribution over n_i points and k categories. Note that this term can be efficiently computed in sub-linear time $\mathcal{O}(\sqrt{dn_i} + k)$ if approximated by a finite floating-point precision of d digits [92].

Predictive rule list encoding. The total data encoding of a predictive rule list, using the NML encoding, is obtained by inserting (3.12) and (3.15) in (3.8):

$$L(\mathbf{Y} \mid \mathbf{X}, M) = \sum_{j=1}^{t} \left(L_{NML}(Y_j^d) + \sum_{\rho_i \in R} L_{NML}(Y_j^i) \right), \tag{3.16}$$

where for the total data encoding using the prequential plug-in code, substitute $L_{NML}(\cdots)$ by $L_{\text{plug-in}}(\cdots)$ of Eq. (3.14).

Subgroup list encoding. The total data encoding of a subgroup list, using the NML encoding, is obtained by inserting (3.12) and (3.15) in (3.9):

$$L(\mathbf{Y} \mid \mathbf{X}, M) = \sum_{j=1}^{t} \left(L(Y_j^d \mid \hat{\mathbf{\Theta}}^d) + \sum_{s_i \in S} L_{NML}(Y_j^i) \right), \tag{3.17}$$

where $\mathbf{\Theta}^d$ is the dataset marginal parameters, and for the total data encoding using the prequential plug-in code, substitute $L_{NML}(\cdots)$ by $L_{\text{plug-in}}(\cdots)$ of Eq. (3.14).

Example 6 (continuation): Let us revisit the *Zoo* subgroup list example of Figure 3.2 and compute the length encoding of the first subgroup subset Y^1 using the NML encoding. To compute it we just need to get the probabilities associated with each category ($\{0; 0; 0.56; 0.44; 0; 0; 0\}$), the number of samples covered by each of them ($\{0; 0; 10; 8; 0; 0; 0\}$), and the total number of categories $k = |\mathcal{Y}| = 7$. Given these, the length of encoding of the data Y^1 is given by:

$$\begin{aligned} L_{NML}(Y^1) &= (-10 \log 0.56 - 8 \log 0.44) + \mathcal{C}(18, 7) \\ &= 17.84 + 10.42 \\ &= 28.26 \text{ bits.} \end{aligned}$$

3.4.3 Relationship of MDL-optimal subgroup lists to WKL-based SD

We now investigate the relationship between finding an MDL-optimal subgroup list and WKL-based top-k subgroup discovery. Remember that WKL is the weighted Kulback-

Leibler (WKL) divergence, an existing subgroup discovery measure [72] that can be seen as an information-theoretic instance of the general form of a subgroup discovery measure as given in Eq. (2.20); we described it in more detail in Section 2.6.2.
Assume that we have a single target variable (Y instead of $\mathbf{Y}$) and a subgroup list consisting of just one subgroup s with description a (and the default rule). Next, let us turn the MDL minimization problem into a maximization problem by multiplying Eq. (3.2) by minus one and adding a constant (for each dataset) $L(Y \mid \mathbf{\Theta}^d)$ to obtain:

$$s^* = \arg\max_{s \in \mathcal{M}} \Big[L(Y^d \mid \mathbf{\Theta}^d) - L(Y \mid \mathbf{X}, M) - L(M) \Big].$$

In the case of a subgroup list with *one subgroup* and one target, the data encoding of Eq. (3.17) can be substituted by $L(Y \mid \mathbf{X}, M) = L(Y^d \mid \mathbf{\Theta}^d) + L_{NML}(Y^a)$. Also, note that Y^d is given by all the points not covered by the subgroup description a, i.e., $Y^{\neg a}$. Thus, we can further develop the maximization problem to:

$$\begin{aligned}
L(Y \mid \hat{\Theta}^d) - L(Y \mid \mathbf{X}, M) - L(M) = \\
&= L(Y^a \mid \hat{\Theta}^d) + \cancel{L(Y^{\neg a} \mid \hat{\Theta}^d)} - L_{NML}(Y^a) - \cancel{L(Y^{\neg a} \mid \hat{\Theta}^d)} - L(M) \\
&= \sum_{y \in Y^s} \log \frac{\hat{p}_{y|a}}{\hat{p}_{y|d}} - \mathcal{C}(n_a, k) - L(M) \\
&= n_a \sum_{c \in \mathcal{Y}} \hat{p}_{c|a} \log \left(\frac{\hat{p}_{c|a}}{\hat{p}_{c|d}} \right) - \mathcal{C}(n_a, k) - L(M) \\
&= n_a KL(\hat{\Theta}^a; \hat{\Theta}^d) - \mathcal{C}(n_a, k) - L(M),
\end{aligned} \tag{3.18}$$

where $n_a KL(\hat{\Theta}^a; \hat{\Theta}^d)$ is the Weighted Kulback-Leibler divergence from $\hat{\Theta}^a$ to $\hat{\Theta}^d$. *This result shows that finding the MDL-optimal subgroup is equivalent to finding the subgroup that maximizes WKL, plus two extra terms: one that defines the complexity of the distribution $\mathcal{C}(n_a, k)$, and another that defines the complexity of the subgroup $L(M)$.*
When we consider subgroup lists having more than one subgroup, Eq. (3.18) simply expands to:

$$\begin{aligned}
L(Y \mid \hat{\Theta}^d) - L(Y \mid \mathbf{X}, M) - L(M) &= \sum_{a_i \in S} n_i KL(\hat{\Theta}^i; \hat{\Theta}^d) - \sum_{a_i \in S} \mathcal{C}(n_i, k) - L(M) \\
&= \text{SWKL}(S) - \sum_{a_i \in S} \mathcal{C}(n_i, k) - L(M),
\end{aligned}$$

where SWKL(S) is the Sum of Weighted Kulback-Leibler divergences of subgroup set S, a measure for subgroup set quality that we propose later in Section 3.6, and the other terms penalize the complexity of the subgroup list. The fact that the MDL-based objective for the optimal subgroup list can be formulated as subgroup set quality

minus two terms for model complexity demonstrates that our formalization naturally aims for subgroup lists of high quality while penalizing complexity.

3.4.4 Relationship of MDL-optimal subgroup lists to Bayesian testing

We will now show how our MDL criterion is related to Bayesian testing. The Bayesian alternative to statistical testing is the Bayesian factor, denoted here by K [58, 61]. The Bayesian factor compares two models (hypotheses) through the division of the likelihood of the data given each model $\Pr(D \mid M_1)/\Pr(D \mid M_2)$, where the more likely model dominates. Notice that the form that we arrived at in the term $n_a KL(\hat{\Theta}^a; \hat{\Theta}^d) - \mathcal{C}(n_a, k) - L(M)$ of Eq. (3.18) (for a list consisting of one subgroup) is very similar to the logarithm of a Bayes factor, and indeed it can be decomposed into:

$$\begin{aligned} L(Y \mid \hat{\Theta}^d) - L(Y \mid \mathbf{X}, M) - L(M) &= \log\left(\frac{\Pr(Y \mid \mathbf{X}, M)}{\Pr(Y \mid \hat{\Theta}^d)}\right) L(M) \\ &= \log K + L(M), \end{aligned}$$

where we use the Shannon-Fano code [114] to transform code length in bits $L(\cdots)$ to probabilities $\Pr(\cdots)$. In practice, taking into account $L(M)$ (or $\Pr(M)$) is equivalent to using the posterior distributions instead of just the Bayes factor, and in our case amounts to a penalty for multiple hypothesis testing. This tells us that when finding the first subgroup we are indeed maximizing an MDL version of a Bayesian factor, and thus, doing an equivalent Bayesian proportions test (with a binary target) or a multinomial test (with a nominal target). When we consider the problem of finding a subgroup beyond the first, it is straightforward to observe that we are testing each subgroup in S against the marginal distribution of the dataset.

3.5 Data encoding: numeric target variables

When we have one or more numeric target variables, the consequents of probabilistic rules as in Eq. (2.7) are now normal distributions $\mathcal{N}(\Theta)$ with parameters $\Theta = \{\mu, \sigma\}$, and take the following form:

$$\Pr(y \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right),$$

where we use $\Pr(y \mid \mu, \sigma)$ to denote the probability density function (pdf), which is a slight abuse of notation that we admit to unify the whole work.

This translates to a probabilistic rule of the form:

$$a \mapsto y_1 \sim \mathcal{N}(\hat{\mu}_{a1}, \hat{\sigma}_{a1}), \cdots, y_t \sim \mathcal{N}(\hat{\mu}_{at}, \hat{\sigma}_{at}) \tag{3.19}$$

To simplify the introduction of concepts we will again assume we only have one target variable in $\mathbf{Y}$, and then generalize the results to multiple variables at the end. Also in line with this simplification, we will only refer to association rules, and then, specialize in the end for both predictive rule lists and subgroup lists. Thus, throughout this section $\mathbf{Y}$ becomes Y, and the parameters of each rule r_i become $\Theta^i = \{\mu_i, \sigma_i\}$ as there is only one variable. The general form of a rule list with normal target distribution is given in Figure 3.5.

$$\begin{array}{rlll} r_1\text{:} & \text{IF} & a_1 \sqsubseteq \mathbf{x} \quad \text{THEN} & y \sim \mathcal{N}(\hat{\mu}_1, \hat{\sigma}_1) \\ & & \vdots & \\ r_\omega\text{:} & \text{ELSE IF} & a_\omega \sqsubseteq \mathbf{x} \quad \text{THEN} & y \sim \mathcal{N}(\hat{\mu}_\omega, \hat{\sigma}_\omega) \\ \text{dataset:} & \text{ELSE} & & y \sim \mathcal{N}(\hat{\mu}_d, \hat{\sigma}_d) \end{array}$$

Figure 3.5: Generic rule list model M with ω rules $\{r_1, ..., r_\omega\}$ and a single numeric target Y.

In the following subsections, we will derive the data encoding with normal distributions. First, in Section 3.5.1 we show how to encode a normal distribution when its parameters μ and σ are known, such as is the case for the default rule of a subgroup list. After that, in Section 3.5.2 we show how to encode a normal distribution using an uninformative prior when the parameters of the distribution are unknown. Then, in Section 3.5.3 the equivalence between MDL-based subgroup lists with only one subgroup and standard (top-1) subgroup discovery with WKL as a quality measure is proven. Finally, in Section 3.5.4, we show the data encoding and corresponding criterion are equivalent to a Bayesian test. Note that for the next section we will also use the maximum likelihood expressions of Section 2.3.2.

3.5.1 Encoding normal distributions with *known* parameters

To encode target values with *known parameters*—as is the case for the default rule of a subgroup list—we can directly use Eq. (3.10) with given parameter estimates

$\hat{\Theta}_d = \{\hat{\mu}_d, \hat{\sigma}_d\}$ (marginal distribution over the whole dataset):

$$\begin{aligned} L(Y^d \mid \hat{\mu}_d, \hat{\sigma}_d) &= -\log\left[\prod_{y\in Y^d} \frac{1}{\sqrt{2\pi\hat{\sigma}_d^2}} \exp\left(\frac{(y-\hat{\mu}_d)^2}{2\hat{\sigma}_d^2}\right)\right] \\ &= \frac{n_d}{2}\log 2\pi + \frac{n_d}{2}\log\hat{\sigma}_d^2 + \left(\frac{1}{2\hat{\sigma}_d^2}\sum_{y\in Y^d}(y-\hat{\mu}_d)^2\right)\log e \\ &= -\ell(\hat{\Theta}^d \mid Y^d), \end{aligned} \tag{3.20}$$

where $\ell(\hat{\Theta}^d \mid Y^d)$ is the log-likelihood of the parameter set $\hat{\Theta}^d$. The first two terms are normalization terms of a normal distribution, while the last term represents the Residual Sum of Squares (RSS) normalized by the variance of the data. Note that when $Y_d = Y$, i.e., the whole dataset target, RSS is equal to $n_d\sigma_d$, and the last term reduces to $n_d/2\log e$.
Note that we exemplified this code using the dataset marginal distribution parameters as these are the only *known* parameters used throughout this thesis, however, this encoding can be used with any known parameters.

3.5.2 Encoding normal distributions with *unknown* parameters

In contrast to the previous case, here *we do not know a priori the statistics defining the probability distribution corresponding to the rule*, i.e., $\hat{\mu}$ and $\hat{\sigma}$ are not given by the model, and thus both need to be encoded. For this, we resort to the Bayesian encoding of a normal distribution with mean μ and standard deviation σ unknown, which was shown to be asymptotically optimal [48]. The optimal code length is given by the negative logarithm of a probability, and the optimal Bayesian probability for Y^i is given by

$$\begin{aligned} &L_{Bayes}(Y^i) = \\ &-\log\int_{-\infty}^{+\infty}\int_{0}^{+\infty}(2\pi\sigma)^{-\frac{n_i}{2}}\exp\left(-\frac{1}{2\sigma^2}\sum_{y\in Y^i}(y-\mu)^2\right)w(\mu,\sigma)\,\mathrm{d}\mu\,\mathrm{d}\sigma, \end{aligned} \tag{3.21}$$

where $w(\mu, \sigma)$ is the prior on the parameters, which needs to be chosen.

Choosing the prior. The MDL principle requires the encoding to be as unbiased as possible for any values of the parameters, which leads to the use of uninformative priors. The most uninformative prior is Jeffrey's prior, which is $1/\sigma^2$ and therefore constant for any value of μ and σ, but unfortunately its integral is undefined, i.e., $\int\int \sigma^{-2}\,\mathrm{d}\sigma\,\mathrm{d}\mu = \infty$. Thus, we need to make the integral finite, which we will do next.

It should be noted that when using normal distributions with Bayes factors—Bayesian equivalent to traditional statistical testing—the authors tend to also add a normal prior on the effect size, as e.g., $\delta = \mu/\sigma \sim \mathcal{N}(0, \tau)$ [58, 44, 110]. Nonetheless, this prior gives a higher probability to values of μ closer to zero, which is a bias that we do not want to impose. Thus we only use Jeffrey's prior, which converges[7] to the Bayes Information Criterion (BIC) for large n.

Now, given the our prior $w(\mu, \sigma) = \frac{1}{\sigma^2\sqrt{2\pi}}$—where $\sqrt{2\pi}$ was added for normalization reasons—the remaining question is how we can make the integral finite. The most common solution, which we also employ, is to use u data points from Y^i, denoted $Y^{i|u}$, to create a proper conditional prior $w(\mu, \sigma \mid Y^{i|u})$. As there are only two unknown parameters, we only need two points hence $u = 2$ [48]; for more on the interpretation of such "priors conditional on initial data points", see [47]. Consequently, we first encode $Y^{i|2}$ with a non-optimal code that is readily available—i.e., the dataset distribution of Eq. (3.20)—and then use the Bayesian rule to derive the total encoded length of Y^i as

$$\begin{aligned} L_{Bayes2.0}(Y^i) &= -\log \frac{P_{Bayes}(Y^i)}{P_{Bayes}(Y^{i|2})} P(Y^{i|2} \mid \mu_d, \sigma_d) \\ &= L_{Bayes}(Y^i) + L_{cost}(Y^{i|2}), \end{aligned} \tag{3.22}$$

where $L_{cost}(Y^{i|2}) = L(Y^{i|2} \mid \mu_d, \sigma_d) - L_{Bayes}(Y^{i|2})$ is the extra cost incurred by encoding two points non-optimally. After some re-writing[8] we obtain the encoded length of the y values covered by a subgroup Y^i as

$$\begin{aligned} &L_{Bayes2.0}(Y^i) = L_{Bayes}(Y^i) + L_{cost}(Y^{i|2}) \\ &= 1 + \frac{n_i}{2}\log\pi - \log\Gamma\left(\frac{n_i}{2}\right) + \frac{1}{2}\log(n_i) + \frac{n_i}{2}\log n_i\hat{\sigma}_i^2 + L_{cost}(Y^{i|2}), \end{aligned} \tag{3.23}$$

where Γ is the Gamma function that extends the factorial to the real numbers ($\Gamma(n) = (n-1)!$ for integer n) and $\hat{\mu}_i$ and $\hat{\sigma}_i$ are the statistics of Eqs. (2.10) and (2.11), respectively. Note that for $Y^{i|2}$ any two unequal values (otherwise $\hat{\sigma}_2 = 0$ and $L_{Bayes}(Y^{i|2}) = \infty$) can be chosen from Y^a, thus we choose them such that they minimize $L_{cost}(Y^{i|2})$.

Predictive rule list encoding. The total data encoding of a predictive rule list, is obtained by inserting Eq. (3.20) and (3.23) in (3.8):

$$L(\mathbf{Y} \mid \mathbf{X}, M) = \sum_{j=1}^{t} \left(L(Y_j^d \mid \mathbf{\Theta}^d) + \sum_{s_i \in S} L_{Bayes2.0}(Y_j^i) \right).$$

[7] See proof in Appendix E.

[8] The full derivation of the Bayesian encoding and an in-depth explanation are given in Appendix D.

Subgroup list encoding. The total data encoding of a subgroup list, is obtained by inserting Eq. (3.20) and (3.23) in (3.9):

$$L(\mathbf{Y} \mid \mathbf{X}, M) = \sum_{j=1}^{t} \left(L(Y_j^d \mid \mathbf{\Theta}^d) + \sum_{s_i \in S} L_{Bayes2.0}(Y_j^i) \right),$$

where $\mathbf{\Theta}^d$ is the dataset marginal parameters.

Example 7 (continuation): We revisit the *Automobile* subgroup list of Figure 3.3 and find the length of the $Bayes2.0$ encoding (Eq. (3.23)) of the first subgroup. To compute it we need to get the statistics of the subgroup ($\hat{\Theta}^1 = \{\hat{\mu}_1 = 35; \hat{\sigma}_1 = 8\}$), the number of samples it covers ($n_1 = 11$), the dataset statistics ($\hat{\Theta}^d = \{\hat{\mu}_d = 13; \hat{\sigma}_d = 8\}$), and the two points closest to the dataset mean $Y^{1|2} = \{14; 31\}$ that make the encoding proper (and which are not available in the example information). Assuming that $L_{cost}(Y^{i|2}) = 0.69$ bits for simplicity, the length of the encoding of Y^1 is given by:

$$\begin{aligned} L_{Bayes2.0}(Y^1) =& 1 + \frac{11}{2}\log\pi - \log\Gamma\left(\frac{11}{2}\right) + \frac{1}{2}\log(11+1) + \frac{11}{2}\log 11 \cdot 8^2 \\ &+ L_{cost}(Y^{i|2}) \\ =& 58.06 + 0.69 \\ =& 58.75 \text{ bits.} \end{aligned}$$

3.5.3 Relationship of MDL-optimal subgroup lists to WKL-based SD

As in Section 3.4 we next investigate the relationship between finding an MDL-optimal subgroup list and WKL-based top-1 subgroup discovery, but now for the numeric case.

First, we show that Eq. (3.23)—with mean and variance unknown—converges, for large n, to Eq. (3.20)—with mean and variance known—plus an additional term. Using the Stirling approximation of $\Gamma(n+1) \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$ leads to[9]

$$L_{Bayes2.0}(Y^a) \sim \frac{n_a}{2}\log 2\pi + \frac{n_a}{2}\log\hat{\sigma}_a^2 + \frac{n_a}{2}\log e + \log\frac{n_a}{e}, \tag{3.24}$$

where $\log\frac{n}{e}$ is equal to the penalty term of BIC and similar to the usual MDL complexity of a distribution [48].

Now, we can show that minimizing our MDL criterion is equivalent to maximizing a subgroup discovery quality function of the form of Eq (2.20). Focusing on the case

[9]The complete derivation can be found in the Appendix E

where $M = \{s\}$ contains only one subgroup with description a and statistics $\hat{\Theta}^a = \{\hat{\mu}_a, \hat{\sigma}_a\}$, we start with $L(Y \mid X, M)$ (Eq. (3.2)), multiply it by minus one to make it a maximization problem, and add a constant $L(Y \mid \hat{\mu}_d, \hat{\sigma}_d)$, i.e., the encoded size of the whole target Y using the overall distribution dataset. We then get

$$s^* = \arg\max_{s \in \mathcal{M}} \Big[L(Y^d \mid \mathbf{\Theta}^d) - L(Y \mid \mathbf{X}, M) - L(M) \Big].$$

Developing this further, the subgroup s that maximizes this expression is equivalent to the one that maximizes

$$\begin{aligned}
&L(Y \mid \hat{\Theta}^d) - L(Y \mid X, M) \\
&= L(Y^a \mid \hat{\Theta}^d) - L_{Bayes2.0}(Y^a \mid \mathbf{X}^a) - L(M) \\
&\sim \frac{n_a}{2} \log \frac{\hat{\sigma}_d^2}{\hat{\sigma}_a^2} + \left[\frac{1}{2\hat{\sigma}_d^2} \sum_{y^i \in Y^a} (y^i - \hat{\mu}_d)^2 \right] \log e - \frac{n_a}{2} \log e - \log n_a - L(M) \\
&= \frac{n_a}{2} \log \frac{\hat{\sigma}_d^2}{\hat{\sigma}_a^2} + \left[\frac{\sum_{y^i \in Y^a} (y^i)^2 - n\hat{\mu}_a^2 + n\hat{\mu}_a^2 - 2n\hat{\mu}_a\hat{\mu}_d - \hat{\mu}_d)^2}{2\hat{\sigma}_d^2} \right] \log e \\
&\quad - \frac{n_a}{2} \log e - \log n_a - L(M) \\
&= n_a \left[\log \frac{\hat{\sigma}_d}{\hat{\sigma}_a} + \frac{\hat{\sigma}_a^2 + (\mu_a - \mu_d)^2}{2\hat{\sigma}_d^2} \log e - \frac{\log e}{2} \right] - \log(n_a) - L(M) \\
&= n_a KL(\hat{\Theta}^a; \hat{\Theta}^d) - \log n_a - L(M),
\end{aligned} \tag{3.25}$$

where $n_a KL(\hat{\Theta}^a; \hat{\Theta}^d)$ is the usage-weighted Kullback-Leibler divergence between the normal distributions specified by the respective parameter vectors. Similar to the result for the nominal target in Section 3.4.3, this shows that *finding the MDL-optimal subgroup is equivalent to finding the subgroup that maximizes the weighted Kullback-Leibler (WKL) divergence*, an existing subgroup discovery quality measure [72], *plus two terms*. The first defines the complexity of the subgroup distribution with two parameters, the second compensates for multiple hypothesis testing (i.e., the number of possible subgroups). When we have a list with multiple subgroups, Eq. (3.18) expands to

$$\begin{aligned}
L(Y \mid \hat{\Theta}^d) - L(Y \mid \mathbf{X}, M) - L(M) &\sim \sum_{a_i \in S} n_i KL(\hat{\Theta}^i; \hat{\Theta}^d) - \sum_{a_i \in S} \log(n_i) - L(M) \\
&= \text{SWKL}(S) - \sum_{a_i \in S} \log(n_i) - L(M),
\end{aligned}$$

where SWKL(S) is the measure of subgroup set qualities that we proposed in Section 3.6, and the other terms penalize the complexity of the subgroup list.

Dispersion-correction quality measure. Importantly, we can observe from Eq. (3.18) that the measure based on the Kullback-Leibler divergence of normal distributions is part of the family of *dispersion-corrected* subgroup quality measures, as it takes into account both the centrality and the spread of the target values [12].

3.5.4 Relationship of MDL-optimal subgroup lists to Bayesian testing

When we have only one subgroup s in a subgroup list, the data encoding for numeric targets of Eq. (3.5.2) is equivalent to the negative logarithm of a Bayes factor [44, 110]. Indeed, the choice of the prior was based on the Bayesian one-sample t-test by Gönen et al. [44], and we effectively perform a one-sample t-test (including two extra terms) for each subgroup. Formally—and similar to the nominal case as described in Section 3.4.4—a Bayes factor K [58, 61] is given by the division of the likelihoods of the data given each hypothesis: $\Pr(D \mid M_1)/\Pr(D \mid M_2)$. If we use the maximization equivalent of Eq. (3.25),

$$\begin{aligned} L(Y \mid \hat{\Theta}^d) - L(Y \mid \mathbf{X}, M) - L(M) &= \log\left(\frac{\Pr(Y \mid \mathbf{X}, M)}{\Pr(Y \mid \hat{\Theta}^d)}\right) L(M) \\ &= \log K + L(M), \end{aligned}$$

we can see that we have the Bayes factor plus the model encoding. To transform code lengths in bits $L(\cdots)$ to probabilities $\Pr(\cdots)$ we used the Shannon-Fano code [114], which states that the best encoding is given by the negative logarithm of its probability for an event A, i.e., $L(A) = -\log \Pr(A)$. Our MDL-based criterion aims at maximizing a one-sample t-test for numeric targets between the subgroup distribution and the marginal distribution of the dataset while taking into account $L(M)$, which is equivalent to using the posterior distribution and penalizes for multiple hypothesis testing. When we aim to find subgroups beyond the first, it is trivial to see that we are testing each subgroup in S against the marginal distribution of the dataset.

3.6 A new measure for subgroup sets: the sum of WKL divergences

As discussed in Section 2.7, there is no SSD measure, to the best of our knowledge, that takes into account the individual quality of subgroups and their global quality over the whole dataset. Therefore, based on the results of Section 3.4.3 and 3.5.3, it is natural to extend the flexible WKL measure in subgroup discovery (described in Section 2.6.2) to subgroup sets.

That is, we propose the *Sum of Weighted Kullback-Leibler divergences* (SWKL), which can be interpreted as the sum of weighted KL divergences for the individual subgroups:

$$\text{SWKL}(S) = \frac{\sum_{i=1}^{\omega} n_i KL(\hat{\Theta}_j^i; \hat{\Theta}_j^d)}{|D|}, \tag{3.26}$$

where i is the index of each subgroup in a subgroup list, ω is the number of subgroups in S, and $|D|$ is the number of instances in D. The latter is used to normalize the measure and make values comparable across datasets. In the case of multiple target variables, the normalization could also include the number of targets, but that is not used in this thesis. The SWKL measure assumes that the data is partitioned per subgroup and that subgroups can be interpreted sequentially as a list, i.e., the second subgroup is interpreted as the description of the second subgroup is active, while the one of the first *is not* active.
An advantage of the SWKL measure is that it can be used for any type of target variable(s), as long as they are described by a probabilistic model. Note that computing SWKL is straightforward for subgroup lists, but not for subgroup *sets* as instances can be covered by multiple subgroups. For subgroup sets, it would be necessary to explicitly define the type of probabilistic overlap, e.g., additive or multiplicative mixtures of the individual subgroup models.

It should be noted that this measure only quantifies how well a list of subgroups capture the deviations in a given dataset and is prone to overfitting: the higher the number of subgroups, the easier it is to obtain a higher value as there is no penalty for the number of subgroups (or their complexities, for that matter). As such, SWKL can be seen as a measure for 'goodness of fit' for subgroup lists. This turns out to not be an issue for our approach though, as our MDL-based criterion naturally penalizes for multiple hypothesis testing and complexity of the individual subgroups. Further, it is neither an issue in our empirical comparisons in Section 5.3, as the number of subgroups found was similar for most algorithms, rendering the SWKL-based comparison valid.

3.7 Theoretical difference between subgroup list and predictive rule list

In this section, we show the difference between the objectives for subgroup discovery and predictive rules. We do this through the comparison of the equivalent maximization MDL scores for subgroup lists and classification rule lists [96] with only one rule—without loss of generality for greater sizes or regression tasks. To differentiate

both model classes SL and RL will be used for subgroup lists and classification rule lists, respectively.
First, let us recall the form of a subgroup list SL as given in:

$$\text{subgroup } 1 : \text{IF } a \sqsubseteq \mathbf{x} \text{ THEN } y \sim Cat(\hat{\Theta}^a)$$
$$\text{dataset} : \text{ELSE } y \sim Cat(\hat{\Theta}^d)$$

where $\hat{\Theta}^a$ are the estimated parameters of subgroup 1 and $\hat{\Theta}^d$ are the estimated parameters of the marginal distribution of the dataset and are thus constant for each dataset. On the other hand, the model form of a classification rule list RL takes the following form:

$$\text{predictive rule } 1 : \text{IF } a \sqsubseteq \mathbf{x} \text{ THEN } Cat(\hat{\Theta}^a)$$
$$\text{default} : \text{ELSE } y \sim Cat(\hat{\Theta}^{\neg a})$$

where $\hat{\Theta}^{\neg a}$ was used to emphasize that the default rule of a predictive rule list is not fixed, and is equivalent to the 'not rule 1'. This is the key difference between these two types of models: for subgroup lists the default rule is fixed to the marginal distribution of the dataset, while for predictive rule lists the default rule has the distribution of the negative set of the rules in the list. It should be noted that there are many definitions of rule lists for classification that use a fixed default rule, however having a variable default rule that maximizes the prediction quality is the best representative of predictive rule lists and of the objective of finding the best machine learning model, i.e., returning the best partition of the data with the smallest error possible. Note that a decision tree also belongs to this family of models, as any path starting at the root of the tree to one of its leaves also forms a rule, and thus, a decision tree is equivalent to a set of disjoint rules, i.e., none of the rules described in this way overlap on a dataset. For the type of classification rule lists defined above, the encoding of the first rule and default rule is given by Eq. (3.15) as for both rules the parameters are unknown.
Thus the MDL score of a predictive rule list can be rewritten as:

$$L(D, RL) = L(Y^a \mid \mathbf{X}^a) + L(Y^{\neg a} \mid \mathbf{X}^{\neg a}) + L(RL), \tag{3.27}$$

and note that the model encoding $L(RL) = L(SL)$ when having the same association rules.
Following the same steps as in Section 3.4.3, turning the MDL score objective from a minimization to maximization by multiplying by minus one and adding the constant $L(Y^d \mid \mathbf{\Theta}^d)$, we obtain the same objective as in Eq. (3.4.3):

$$\rho^* = \underset{s \in \mathcal{M}}{\arg\max} \left[L(Y^d \mid \mathbf{\Theta}^d) - L(Y \mid \mathbf{X}, RL) - L(RL) \right],$$

where ρ is the classification rule that maximizes the objective. Working out this equation, maximization objective of a *classification rule list* for a target variable of k class labels is given by:

$$\begin{aligned}
&L(Y \mid \hat{\Theta}^d) - L(Y \mid \mathbf{X}, M) - L(RL) \\
&= L(Y^a \mid \hat{\Theta}^d) + L_{NML}(Y^{\neg a} \mid \hat{\Theta}^d) - L(Y^a \mid X_a) - L_{NML}(Y^{\neg a} \mid \mathbf{X}^{\neg a}) - L(RL) \\
&= n_a KL(\hat{\Theta}^a; \hat{\Theta}^d) - \mathcal{C}(n_a, k) + n_{\neg a} KL(\hat{\Theta}^{\neg a}; \hat{\Theta}^d) - \mathcal{C}(n_{\neg a}, k) - L(RL),
\end{aligned} \tag{3.28}$$

This should be contrasted with the maximization objective of *subgroup list* of Eq. 3.18, which is given by:

$$\begin{aligned}
&L(Y \mid \hat{\Theta}^d) - L(Y \mid \mathbf{X}, M) - L(SL) = \\
&n_a KL(\hat{\Theta}^a; \hat{\Theta}^d) - \mathcal{C}(n_a, k) - L(SL).
\end{aligned}$$

Comparing the two previous equations, we can notice the most important distinction between subgroup discovery and classification: the *local* nature of subgroup discovery and the *global* nature of the classification task. In other words, subgroup discovery aims at finding subgroups that locally maximize their quality, independently of the rest of the dataset, and even though classification rules try to maximize their *local* quality also, they have to take into account the quality of their negative set, i.e., a classification rule cannot be considered by its quality alone, it has to be considered in terms of its *global* impact in the dataset.

On the other hand, this result also shows the similarity between both tasks and where the confusion sometimes arises, i.e., in particular cases the best subgroup can also be the best predictive rule. An example of this would be a very large dataset (relatively to the number of observations covered by the rule), and the best rule would cover a small number of observations compared to the rule formed by the negative set of that rule, i.e., $D^{\neg a}$, as a similar distribution to $\hat{\Theta}^d$, making $\hat{\Theta}^{\neg a} \sim \hat{\Theta}^d$.

4

Discovering predictive rule lists with CLASSY

In this chapter[1], we propose the CLASSY algorithm based on the MDL formulation of predictive rule lists given in Chapter 3. This algorithm uses a greedy heuristic to find good predictive rule lists and can be applied to supervised tabular datasets with univariate nominal targets, i.e., multiclass classification. In machine learning, a predictive rule list is an instance of interpretable machine learning models, as long as the number of rules is reasonably small. To validate our approach, we conduct an empirical comparison on 17 datasets against state-of-the-art-algorithms.

Note that in Chapter 3 we presented a formulation of predictive rule lists for single and multi-nominal classification and single and multi-regression problems. However, we only developed CLASSY for *discrete* explanatory variables and regular classification, hence, we only present results for this scenario in this chapter. For regression the greedy algorithm would require some changes in order to find good regression rule lists.

Recapitulation of predictive rule list definition and MDL encoding. In the previous chapters we defined what a predictive rule list is and gave its definition of optimality according to the MDL principle. We will now restate those definitions here.

First, let us recall from Chapter 2.4 the predictive rule lists model in Figure 4.1.

The best predictive rule list RL according to the MDL principle is the one that, given

[1] Parts of this chapter are based on Proença and van Leeuwen [96]

$$
\begin{array}{rlll}
s_1\text{:} & \text{IF} & a_1 \sqsubseteq \mathbf{x} \quad \text{THEN} & y \sim Cat(\hat{p}_{1|1}, \cdots, \hat{p}_{k|1}) \\
 & & \vdots & \\
s_\omega\text{:} & \text{ELSE IF} & a_\omega \sqsubseteq \mathbf{x} \quad \text{THEN} & y \sim Cat(\hat{p}_{1|\omega}, \cdots, \hat{p}_{k|\omega}) \\
\text{dataset:} & \text{ELSE} & & y \sim Cat(\hat{p}_{1|d}, \cdots, \hat{p}_{k|d})
\end{array}
$$

Figure 4.1: Generic rule list model RL for classification with ω rules $R = \{r_1, ..., r_\omega\}$ and one target variable distribution per rule. Note that the parameters of the default rule of a rule list are not fixed (in contrast to those of a subgroup list) and just describes the subset covered by it, i.e., D^d.

the dataset D, minimizes the two-part code defined in Chapter 3.1:

$$RL^* = \underset{RL \in \mathcal{RL}}{\arg\min}\, L(D, RL) = \underset{RL \in \mathcal{RL}}{\arg\min} \left[L(\mathbf{Y} \mid \mathbf{X}, RL) + L(RL) \right],$$

where $L(RL)$ is the length of encoding the predictive rule list model RL, and $L(\mathbf{Y} \mid \mathbf{X}, RL)$ is the length of encoding the target variables data given the predictive rule list RL and the explanatory variables $\mathbf{X}$. The *model encoding* is the same for predictive rule lists and subgroup lists, as they only differ in how the default rule encodes the data, and was defined in Chapter 3.2. Nonetheless, at the time these experiments were developed [96], the formulation was only done for discrete explanatory variables, and it is suboptimal compared to that of Eq. (3.3). Thus, the model encoding throughout this chapter is:

$$L(RL) = L_{\mathbb{N}}(|R|) + \sum_{a_i \in R} \left[L_{\mathbb{N}}(|a_i|) + |a_i| \log m \right],$$

where R is the list of predictive rules in RL, i.e., the model excluding the default rule. Compared with Eq. (3.3), we here used a uniform code for the variables in a_i, i.e., $|a_i| \log m$, which is suboptimal for unordered sets compared to $\binom{m}{|a_i|}$. Also, we do not require $\sum_{v \in a_i} L(v)$ for the different types of variables, as only binary explanatory variables were considered, and only their positive value, i.e., $x = 1$ is encoded.
Then, in the case of *classification*, where there is only one target variable Y, we use a categorical distribution and the Prequential Plug-in encoding defined in Chapter 3.4:

$$L(Y \mid \mathbf{X}, RL) = L_{\text{plug-in}}(Y^d) + \sum_{\rho_i \in R} L_{\text{plug-in}}(Y^i),$$

which is asymptotically optimal. Even though the Normalized Maximum Likelihood encoding is optimal for finite n, and thus, theoretically better than the prequential plug-in, we did not find very significant differences in preliminary experiments, except for a few edge cases.

Structure of the chapter. This chapter is organized as follows. First, in Section 4.1 the most relevant related work is covered, together with the main differences to our approach. After that, in Section 4.2 the CLASSY algorithm, a heuristic algorithm to mine predictive rule lists is defined. Then, in Section 4.3 we empirically validate our proposed method on 17 datasets when compared against the state-of-the-art algorithms for classification. Finally, in Section 4.4 the main conclusions are presented.

4.1 Related work

We start by comparing the most important features of our algorithm to those of state-of-the-art algorithms and then provide a brief overview of the most relevant literature, grouped into three topics: 1) rule-based models; 2) similar approaches in pattern mining; and 3) MDL-based data mining. For an in-depth overview of interpretable machine learning, we refer to Molnar [91].

Table 4.1 compares the most important features of our proposed approach, called CLASSY, to those of other rule-based classifiers, which will be described in the next subsections. Classical methods, such as CART [15], C4.5 [103], and RIPPER [22], lack a global optimisation criterion and thus rely on heuristics and hyperparameters to deal with overfitting. Fuzzy rule-based models [3, 59], here represented by FURIA [55] use rule sets instead of rule lists and lack probabilistic predictions. Recent Bayesian methods [122, 69, 122] are limited to small numbers of candidate rules and binary classification, limiting their usability, and are here represented by SBRL [125] (which is representative for all of them). A recent approach also using MDL and probabilistic rule lists (MRL) [7] is aimed at describing rather than classifying and cannot deal with multiclass problems or a large number of candidates. Interpretable decision sets (IDS) [69] and certifiable optimal rules (CORELS) [5] use similar rules but do not provide probabilistic models or predictions.

Note that methods that explain black-box models [104, 105], typically denoted by the term *explainable machine learning*, also aim to make the decisions of classifiers interpretable. However, they mostly focus on sample-wise (local interpretation) explanations, while we focus on explaining the whole dataset (global interpretation) using a single model. As these goals lead to clearly different problem formulations and thus different results, it would not be meaningful to empirically compare our approach to explainable machine learning methods [111].

Table 4.1: Our approach, CLASSY, does *Multiclass* classification, makes *Probabilistic* predictions, has a global optimisation *Criterion*, can handle large numbers of *cand*idate rules, and does not need hyperparameter *tuning*. "Others" denotes classical algorithms such as CART, CBA, C4.5, and RIPPER.

Method	Multiclass	Probabilistic	Criterion	$\gg$1K cand	No tuning
CLASSY	✓	✓	✓	✓	✓
IDS[69]	✓	-	✓	✓	-
CORELS [5]	-	-	✓	-	-
MRL[7]	-	✓	✓	-	✓
SBRL[125]	-	✓	✓	-	-
FURIA[55]	✓	-	-	✓	-
Others	✓	✓	-	✓	-

4.1.1 Rule-based classifiers

Rule lists have long been successfully applied for classification; RIPPER is one of the best-known algorithms [22]. Similarly, decision trees, which can easily be transformed to rule lists, have been used extensively; CART [15] and C4.5 [103] are probably the best-known representatives. These early approaches represent highly greedy algorithms that use heuristic methods and pruning to find the 'best' models.

Fuzzy rule-based models. Fuzzy rules have been extensively studied in the context of classification and interpretability. Several approaches to construct fuzzy rule-based models have been proposed, such as transforming the resulting model of another algorithm into a fuzzy model and posteriorly optimizing it [55], using genetic algorithms to combine pre-mined fuzzy association rules [3], and doing a multiobjective search over accuracy and comprehensibility to find Pareto-optimal solutions [59]. Although these approaches are related, the rules are aggregated in a rule set, i.e., a set of independent *if* $\cdots$ *then* $\cdots$ rules that can be activated at the same time to classify one instance, contrary to one rule at the time for rules lists. This makes the comparison between both types of models difficult. Also, these fuzzy rule-based models do not provide probabilistic predictions.

Global optimization approaches. Over the past years, rule learning methods that go beyond greedy approaches have been developed, i.e., using probabilistic logic programming for independent rule-like models [11], greedy optimization of submodular problem formulation, or simulated annealing in the case of decision sets [69, 122], Monte-Carlo search for Bayesian rule lists [81, 125], and through branch-and-bound

with tight bounds for decision lists [5]. Even though in theory these approaches could be easily extended to the multiclass scenario, in practice their algorithms do not scale with the higher dimensionality that arises from the search in multiclass space with optimality criteria. Also, only Bayesian rule lists [81, 125] and Bayesian decision sets [122] provide probabilistic predictions.

All previously mentioned algorithms share some similarities with CLASSY. In particular Bayesian rule lists [81, 125] are closely related as they use the same type of models, albeit with a different formulation, based on Bayesian statistics. This difference leads to different types of priors—for example, we use the universal code of integers [108]—and therefore to different results; we will empirically compare the two approaches. Certifiable optimal rules [5] have a similar rule structure but do not provide probabilistic models or predictions. Decision sets [69] share the use of rules, but as opposed to (ordered) lists they consider (unordered) sets of rules.

4.1.2 Pattern mining

Association rule mining [2], a form of pattern mining, is concerned with mining relationships between itemsets and a target item, e.g., a class. One of its key problems is that it suffers from the infamous *pattern explosion*, i.e., it tends to give enormous amounts of rules. Several classifiers based on association rule mining have been proposed. Best-known are probably CBA [85] and CMAR [82], but they tend to lack interpretability because they use large numbers of rules. Ensembles of association rules, such as Harmony [121] or classifiers based on emergent patterns [41], can increase classification performance when compared to the previous methods, however, they can only offer local interpretations.

Supervised pattern set mining [128]. The key difference is that these methods do not automatically trade-off model complexity and classification accuracy, requiring the analyst to choose the number of patterns k in advance.
Note that subgroup discovery, and specifically subgroup lists, are also related as they share the same model structure as predictive rule lists. For more details on that, we refer the interested reader to Chapter 4.

4.1.3 MDL-based data mining

In data mining, the MDL principle has been used to summarize different types of data, e.g., transaction data [120, 18], and two-view data [73].

In prediction, it has been previously used to deal with overfitting [22, 103] and in the selection of the best compressing pattern [127].
RIPPER and C4.5 [22, 103] use the MDL principles in their post-processing phase as a criterion for pruning, while we use it in a holistic way for model selection. Although Krimp has been used for classification [120], it was not designed for this: it outputs large pattern sets, one for each class, and does not give probabilistic predictions. DiffNorm [18] creates models for combinations of classes and also uses the prequential plug-in code, but was designed for data summarization. Aoga et al. recently also proposed to use probabilistic rule lists and MDL [7], but 1) we propose a vastly improved encoding, which is tailored towards prediction (instead of summarization), 2) our solution does multiclass classification, and 3) our algorithm has better scalability.

4.2 The CLASSY algorithm

Given our model class—predictive rule lists—and its corresponding MDL formulation, what remains is to develop an algorithm that—given the training data—finds the best model according to our MDL criterion. To this end, in this section, we present CLASSY, a greedy search-based algorithm that iteratively finds the best predictive rules to add to a rule list. This section is structured as follows. First, a brief description of separate-and-conquer greedy search is given. Then compression gain, i.e., the measure that uses compression to score candidate rules, is described. After that, the CLASSY algorithm is defined. Then, it is explained how individual rules—candidates for the model—are generated from the data. Finally, we analyse CLASSY's time and space complexity.

4.2.1 Separate-and-conquer greedy search

Greedy search is very commonly used for learning decision trees and predictive rule lists [103, 22, 39], as well as for pattern-based modelling using the MDL principle [120, 18, 73]. A few recent approaches use optimization techniques [125], but these have the limitation that the search space must be strongly reduced, providing an exact solution to an approximate problem (as opposed to an approximate solution to an exact problem).
Global heuristics, such as evolutionary algorithms, have been extensively applied to fuzzy rule-based model learning [34], and although they could also be applied here, we found that the arguments in favor of a local search approach were stronger: 1) local heuristics have often been successfully applied for pattern-based modelling using the MDL principle, making it a natural approach to consider; 2) local heuristics are typically faster than global heuristics, as much fewer candidates need to be evaluated;

3) global heuristics typically require substantially more (hyper)parameters that need to be tuned (e.g., population size, selection and mutation operators, etc.), while local heuristics have very few.

Given the arguments presented here the algorithm that we propose is based on greedy search. More specifically, it is a heuristic algorithm that, starting from a rule list with just a default rule equal to the priors of the class labels in the data, adds rules according to the well-known *separate-and-conquer* strategy [39]: 1) iteratively find and add the rule that gives the largest change in compression; 2) remove the data covered by that rule; and 3) repeat steps 1-2 until compression cannot be improved. This implies that *we always add rules at the end of the list, but before the default rule*.

4.2.2 Compression gain

The proposed heuristic is based on the compression gain that is obtained by adding a rule $\rho = (a, \hat{\Theta}^a)$ to a rule list RL, which will be denoted by $RL \oplus \rho$. Note that for categorical distributions $\hat{\Theta}^a = \{\hat{p}_{1|a}, \cdots \hat{p}_{k|a}\}$, which are just the conditional probability for each class label $c \in \{1, \cdots, k\}$, given the description a. We will argue—and demonstrate empirically later—that for the current task it is better to consider *normalized* gain rather than the typically used *absolute* gain. Note that the gains are defined as positive if adding a rule represents a compression improvement, and negative vice-versa.

Absolute compression gain, denoted $\Delta L(D, RL \oplus \rho)$, is defined as the difference in code length before and after adding a rule ρ to R. The gain can be divided into two parts: *data gain*, $\Delta L(Y \mid \mathbf{X}, RL \oplus \rho)$, and *model gain*, $\Delta L(RL \oplus \rho)$. Together this gives

$$\begin{aligned} \Delta L(D, RL \oplus \rho) &= L(D, RL) - L(D, RL \oplus \rho) \\ &= \underbrace{L_{\text{plug-in}}(Y \mid \mathbf{X}, RL) - L_{\text{plug-in}}(Y \mid \mathbf{X}, RL \oplus \rho)}_{\Delta L(Y|\mathbf{X}, RL \oplus \rho)} \\ &+ \underbrace{L(R) - L(RL \oplus \rho)}_{\Delta L(RL)}, \end{aligned} \tag{4.1}$$

where $L_{\text{plug-in}}(\cdots)$ was used to refer that we use the prequential plug-in encoding of Eq. (3.14) in this chapter (instead of the Normalized Maximum Likelihood). Using Eq. (4) we show the *model gain* as:

$$\begin{aligned} \Delta L(RL \oplus \rho) = &L_{\mathbb{N}}(|R|) - L_{\mathbb{N}}(|R| + 1) \\ &- L_{\mathbb{N}}(|a|) - |a| \log m. \end{aligned} \tag{4.2}$$

Note that the model gain is always negative, as adding a rule adds additional complexity to the model.
In the case of the *data gain*, it should be noted that adding rule ρ to RL only activates the part of the data previously covered by the default rule, as new rules are only added after the previous ones and before the default rule. This search strategy of adding rules assumes that the previous rules already cover their subset well and that improvements only need to be made where no rule is activated, which corresponds to the region of the dataset covered by the default rule. Hence, we only need to compute the difference in length of using the previous default rule ρ_d and the combination of the new pattern $a \in \rho$ with the new default rule ρ'_d. Using Equation (B.3) we obtain

$$\begin{aligned}\Delta L(Y \mid \mathbf{X}, RL \oplus \rho) = \overbrace{\cancel{\sum_{i=1}^{\omega} L_{\text{plug-in}}(Y^i)} + L_{\text{plug-in}}(Y^d)}^{L_{\text{plug-in}}(Y|\mathbf{X},RL)} \\ \underbrace{-\cancel{\sum_{i=1}^{\omega} L_{\text{plug-in}}(Y^i)} - L_{\text{plug-in}}(Y^{d'}) - L_{\text{plug-in}}(Y^a)}_{L_{\text{plug-in}}(Y|\mathbf{X},RL\oplus\rho)},\end{aligned} \tag{4.3}$$

where $Y^{d'}$ is the subset of the data covered by the new default rule (after ρ is added to RL) and $\omega = |R|$.
Normalized compression gain, denoted $\delta L(D, RL \oplus \rho)$, is defined as the absolute gain normalized by the number of instances that are activated by pattern $a \in r$, which can be obtained by dividing absolute gain by the usage of a:

$$\delta L(Y \mid \mathbf{X}, RL \oplus \rho) = \frac{\Delta L(Y \mid \mathbf{X}, RL \oplus \rho)}{n_a} \tag{4.4}$$

By normalizing for the number of instances that a predictive rule covers, normalized gain *favors rules that cover fewer instances but provide more accurate predictions* compared to absolute gain. When greedily covering the data, it is essential to prevent choosing large but moderately accurate rules in an early stage; this is likely to lead to local optima in the search space, from which it could be hard to escape. As this is bound to happen when using absolute gain, we hypothesize that normalized gain will lead to better predictive rule lists. We will empirically verify if this is indeed the case.

Note that the *absolute* and *normalized* gains are specific cases of the β-gain of Eq. (5.1) presented later in Chapter 5, where for these specific cases, they correspond to $\beta = 0$ and $\beta = 1$, respectively.

4.2.3 Candidate generation

Candidates are probabilistic rules of the form $\rho = (a, \hat{\Theta}^a)$ that are considered for addition to a predictive rule list for a dataset D. The candidates are generated by first mining a rule antecedent/pattern a using a standard frequent pattern mining algorithm, e.g., FP-growth [13], and then finding the corresponding consequent categorical distribution $\hat{\Theta}^a$ given the dataset, i.e., using the maximum likelihood estimate of Eq. (2.9). In practice, these mining algorithms have only two parameters: the minimum support threshold $n_{min.}$ and the maximum length d_{max} of a pattern.

Mining frequent patterns can be done efficiently due to the anti-monotone property of their support, i.e., given a pattern a and b, if a has fewer conditions then b, i.e., $a \subset b$, implies that $n_a \geq n_b$. This property is also used to **remove strictly redundant rules** in CLASSY.
Given all candidates from the frequent pattern mining algorithm, if antecedent a is a strict subset of antecedent b, i.e., $a \subset b$, and they have equal support, $n_a = n_b$, we say that antecedent b is redundant and will never be selected. This is a consequence of their encoding, i.e., as $Y^a = Y^b \implies L_{plug-in}(Y^a) = L_{plug-in}(Y^b)$ in the case they are being considered for the same position, and that the model encoding length of b will always be larger than a, i.e., $L(a) < L(b)$. From this, we can conclude that b will never be preferred over a during the model search, as the gain of a will always be greater.

4.2.4 Finding good rule lists

We are now ready to introduce CLASSY, a greedy algorithm for finding good solutions to the MDL-based multiclass classification problem as formalized in Section 3.4. The algorithm, outlined in Algorithm 4.1, expects as input a (supervised) training dataset D and a set of candidate patterns, e.g., a set of frequent itemsets mined from D, and returns a predictive rule list.
The first step of our algorithm is to remove the strictly redundant patterns as mentioned in Section 4.2.3 (Ln 1). After that, we initialize the predictive rule list with the default rule (Ln 2), which acts as the baseline model to start from. Then, while there is a predictive rule that improves compression (Ln 7), we keep iterating over three steps: 1) we select the best rule to add (Ln 4)—we here use normalized gain for ease of presentation, but this can be trivially replaced by absolute gain; 2) we add it to the rule list (Ln 5); and 3) we update the usage, and gain of the candidate list (Ln 6). To update the usage of a candidate it is necessary to remove from its usage the instances that it has in common with the previously added rule, and then the gain of adding the candidate can be updated. When no rule improves compression (negative gain)

the while loop stops and the rule list is returned.

Algorithm 4.1 The CLASSY algorithm

Input: Dataset D, candidate set $Cands$
Output: Multiclass probabilistic rule list R

1: $Cands \leftarrow RemoveRedundancy(Cands)$
2: $RL \leftarrow [\varnothing]$
3: **repeat**
4: $\quad \rho \leftarrow \arg\max_{\forall r' \in Cands} : \delta L(D, RL \oplus \rho')$
5: $\quad RL \leftarrow RL \oplus \rho$
6: $\quad \text{UpdateCandidates}(D, RL, Cands)$
7: **until** $\delta L(D, RL \oplus \rho') \leq 0, \forall \rho' \in Cands$
8: **return** RL

4.2.5 Time and space complexity

In this section we analyze the time and space complexity of CLASSY. In terms of time complexity, CLASSY can be divided into two parts: 1) an initialization step, and 2) an iterative loop where one rule is added to a predictive rule list in each iteration.

Initialization step. The time complexity of the initialization step is dominated by sorting the candidates (ascending by length) obtained after running the frequent pattern mining algorithm, and the computation of their instance ids, i.e., the indexes of the instances where each candidate is present. Sorting all candidates takes $\mathcal{O}(|Cands| \log |Cands|)$ time. To compute the instance ids of the candidates, CLASSY first computes the presence of each singleton condition, i.e., $x_i = 1$ is tested for each variable, in each instance, and then stores them as a bitset in a hash table. As this is done for the whole dataset, it takes $\mathcal{O}(|D||V|)$ time. Then, for candidates of size equal or greater than two and given a sorted array of candidates, it sequentially computes the instance ids of each candidate a based on its decomposition in two candidates of one less condition, i.e., it computes the ids of a based on two candidates b_1 and b_2 for which $b_1 \cup b_2 = a$ of length $|b_1| = |b_2| = |a| - 1$. The ids of a are obtained by the intersection of the sets of instance ids of the smaller length candidates and has a complexity of $\mathcal{O}(|(b_1)_{ids}| + |(b_2)_{ids}|)$. As this is done for each class, in the worst case, it would cover the whole dataset, and it would take $\mathcal{O}(|D| + |D|)$. Doing this for all candidates gives $\mathcal{O}(|Cands||D|)$.

Iterative loop. After the initialization step, CLASSY iteratively finds the best rule to add for a total of $\omega = |R|$ runs, where RL is the predictive rule list that CLASSY

outputs at the end. The time complexity of this loop is dominated by the removal of the instance ids that all candidates have in common with the last added rule. Using again the fact that the intersection of instance ids is upper bounded by the dataset size $|D|$, the removal of instance ids takes at most $\mathcal{O}(|R||Cands||D|)$ time. Given that the rule list can grow at most to the size of the dataset, an upper bound on this complexity is $\mathcal{O}(|Cands||D|^2)$. Joining everything together, CLASSY has a worst-case time complexity of

$$\mathcal{O}(|Cands||D|^2),$$

which is really a worst-case scenario, because, in general, MDL will obtain predictive rule lists that are much smaller than the dataset size, i.e., $|R| \ll |D|$, making it possible to treat it as a constant. Making this assumption, we obtain a more realistic worst case time complexity of

$$\mathcal{O}(|Cands| \log |Cands| + |Cands||D|).$$

Note that the time complexity associated with the Gamma function used in the computation of lengths (3.14) and gains (4.4) of data encoding is not problematic when compared with the other terms. This is due to its recursive computation for $|D|$ values, which can be stored in a dictionary. In total, this takes $\mathcal{O}(multi(|D|) + |D|)$ time, where $multi(*)$ is the complexity of the multiplication used; in the case of our Python implementation, this is the Katsuraba multiplication. From then on, the lookup of a value only takes $\mathcal{O}(1)$ time.

Memory complexity. In terms of memory complexity, CLASSY has to store for each candidate for each class: their instance ids $\mathcal{O}(|(a)_{ids}||\mathcal{Y}|)$, their support $\mathcal{O}(|\mathcal{Y}|)$, and their score $\mathcal{O}(|\mathcal{Y}|)$. It is easy to see that $|(a)_{ids}||\mathcal{Y}|$ is upper bounded by the dataset size $|D|$ and that all other memory requirements will be dominated by this part. Also, the storage of the gamma function for each integer up to $|D|$ is only $\mathcal{O}(|D|)$, which gets dwarfed by the instance storage, thus obtaining a worst-case memory complexity of

$$\mathcal{O}(|D||Cands|).$$

4.3 Empirical evaluation

In this section we empirically evaluate our approach[2], first in terms of its sensitivity to the candidate set provided and the relationship between compression and classification performance, and second in comparison to a set of representative, state-of-the-art baselines in terms of classification performance, interpretability, overfitting, and

[2]For the reproducibility of the experiments, please check `https://github.com/HMProenca/MDLRuleLists`

runtime.

Data. We use 17 varied discretised datasets (see Table F.1) from the LUCS/KDD[3] repository, all of which are commonly used in classification papers. They were selected to be diverse, ranging from 150 to 48 842 samples, from 16 to 157 Boolean variables, and from 2 to 18 classes.

Candidate rules generation. Frequent pattern mining algorithms generate different candidate sets by setting different values for the minimum support per class threshold n_{min} and maximum pattern length d_{max}. To demonstrate that CLASSY is insensitive to the exact settings of these parameters, we fix a single set of parameter values for all experiments on all datasets (except when we investigate the influence of the candidate set). Specifically, we use $d_{max} = 4$ and $n_{min} = 5\%$, to obtain a desirable trade-off between candidate set size, convergence, and runtime.

These values were objectively derived based on two criteria: making each run finish within 10 minutes while demonstrating that CLASSY can deal with large candidate set sizes. First, we chose $d_{max} = 4$ because this potentially results in very large candidate sets with many redundant rules (i.e., rules that are very similar / strongly overlapping). We then fixed n_{min} by requiring the runs for all datasets to strictly finish in under 10 minutes and for most datasets even under 1 minute, to be comparable to $CART$, $C5.0$, and $JRip$ in runtime, and also to have attained (empirical) convergence in terms of compression ratio on the training set—further lowering n_{min} would not increase compression—as can be seen from the vertical dashed lines in Figure 4.3.

Candidate patterns are mined using Borgelt's implementation of the well-known frequent pattern mining algorithm FP-growth [13]. The same candidate set was used for all experiments except when assessing its influence on CLASSY in Section 4.3.2. For that experiment we fixed $d_{max} = 4$ and varied the minimum support threshold per class from $n_{min} = \{0.1\%, 0.5\%, 1\%, 2\%, 5\%, 10\%, 15\%, 20\%, 25\%\}$.

Evaluation criteria. We evaluate and compare our approach based on classification performance, overfitting, interpretability, and runtime. Besides, we assess the influence of the candidate set on our algorithm and whether better compression corresponds to better classification. All results presented are averages obtained using 10 times repeated 10-fold cross-validation (with different seeds).

MDL criterion quantification. To quantify how well a predictive rule list compresses the class labels and be able to compare the MDL score across datasets, we define

[3] http://cgi.csc.liv.ac.uk/~frans/KDD/Software/LUCS-KDD-DN/DataSets/dataSets.html

relative compression as

$$L\% = \frac{L(D, R)}{L(D \mid \hat{\Theta}^d)}, \tag{4.5}$$

where $L(D \mid \hat{\Theta}^d)$ is the encoding length of the data given the predictive rule list with only a default rule, i.e., with only the dataset priors for each class. We measure relative compression on the training data, as we use that for model selection.

Classification performance. Classification performance is measured using three measures presented in Section 2.5: accuracy; balanced accuracy [17]; and *Area Under the ROC Curve* (AUC). Each measure portrays different aspects of the classifier performance. Accuracy shows the total number of correct classifications. Balanced accuracy, or averaged class accuracy, takes into account the imbalance of class distributions in the dataset and gives the same importance to each class.
AUC, on the other hand, is not based on a fixed threshold and takes into account the probabilities associated with each prediction. In the case of multiclass datasets, we use *weighted AUC* [102], as it takes into account the class distribution in the dataset.

Interpretability. For interpretability, we follow the most commonly used measure, i.e., that smaller models are easier to understand [26]. With this in mind, we assess: the number of rules and the number of conditions per rule; in all cases, fewer is better. When analyzing decision trees, the number of leaves is given as the number of rules (which includes the default rule), and the average depth of the leaves (except for the longest—assumed the default rule) is given as the number of conditions per rule. Although predictive rule lists derived from decision trees can often be simplified, we here choose not to do this because these directly measures how it would be read by humans.

Overfitting. Overfitting is measured in terms of the absolute difference between the AUC performance in the training set and in the test set.

Runtime. For runtime, wall clock time in minutes is measured; no parallelization was used.

Note on standard deviations. Given that the number of values reported both in figures and tables is large, and that standard deviations are usually $2+$ orders of magnitude smaller than their corresponding averages, we choose not to report them—to avoid unnecessarily cluttering the presentation. We did analyse them though and explicitly comment on the few cases where relevant.

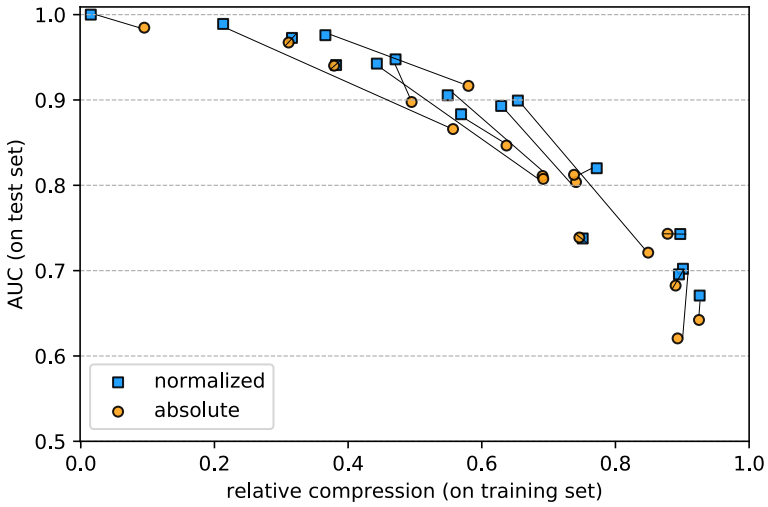


Figure 4.2: Relation between compression and AUC; better compression on the training set (lower relative compression) corresponds to better classification on the test set (higher AUC). Results obtained with CLASSY using normalized (squares) and absolute (circles) gain, on all 17 datasets; each point represents the 10 times repeated 10-fold average for one dataset with one type of gain; each connected pair represents the same dataset, for the two types of gain.

4.3.1 Compression versus classification

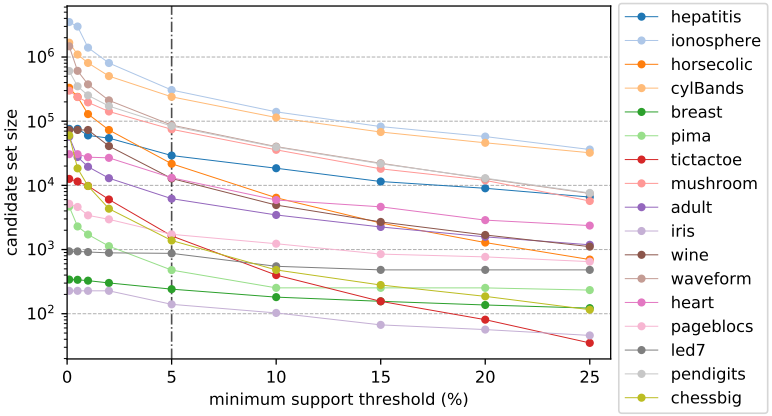
We first investigate the effect of using absolute (4.1) or normalized gain (4.4). To this end, Figure 4.2 depicts how the two heuristics perform for relative compression (on the training set) and AUC (on the test set).

The first observation is that better compression of the training data corresponds to better classification performance on the test data. This is backed by a correlation of -0.92 and a corresponding p-value lower than 0.0001 for the independence test between both variables for the normalized gain data. This is a crucial observation, as it constitutes an independent, empirical validation of using the MDL principle for predictive rule list selection. Moreover, it also shows that MDL successfully protects against overfitting: using normalized gain leads to models that not only compress the training data better but also provides accurate predictions on the test data.

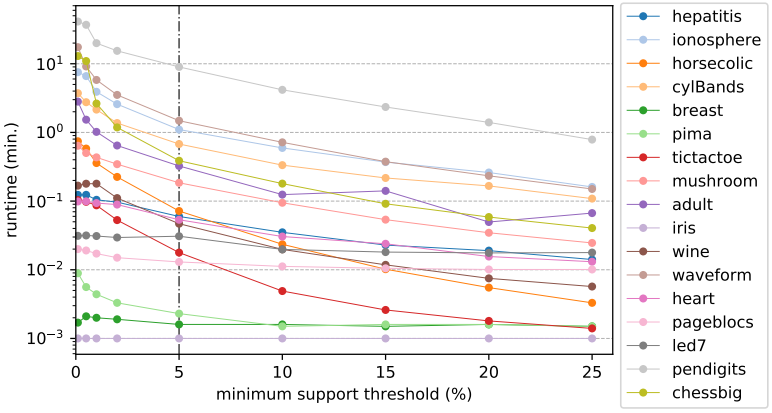
The second observation is that normalized gain performs better overall than absolute gain: AUC is higher in 15 out of 17 cases and relative compression is lower or equal in 11 out of 17 times. This confirms that normalized gain is, as we hypothesized, the best choice. We will therefore use *normalized gain* for the remaining experiments.

4.3.2 Candidate set influence

In this set of experiments, we study the influence of the candidate set on CLASSY, which technically is its only "hyperparameter", as it is the only part that can influence its output given the same dataset. In order to vary the candidate set objectively, the minimum support threshold ranges over $n_{min.} = \{0.1\%, 0.5\%, 1\%, 2\%, 5\%, 10\%, 15\%, 20\%, 25\%\}$ and the maximum pattern length was fixed at $d_{max} = 4$, allowing the generation of large candidate sets.

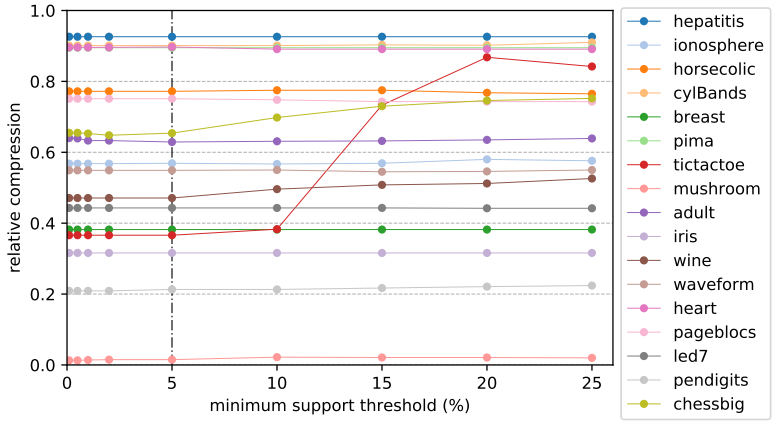


(a) candidate set size

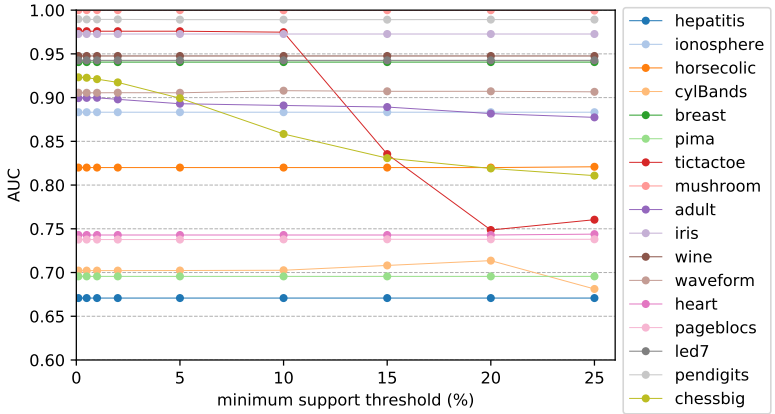


(b) runtime

The results can be seen in the set of Figures 4.3, which show the influence of the can-



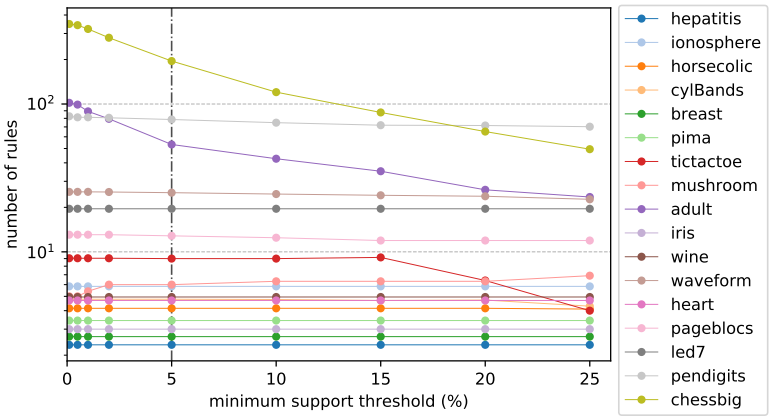
(c) compression in training set



(d) AUC in test set

didate set on CLASSY through: the size of candidates mined in Figure 4.3a; runtime in Figure 4.3b; compression on the training set in Figure 4.3c; AUC in the test set in Figure 4.3d; and the number of rules in a rule list in Figure 4.3e.

Minimum support. Figure 4.3a shows the growth of the candidate set size with the minimum support threshold used, and that, as expected, its growth is exponential with the change in minimum support. Figure 4.3b shows that in general, the runtime increases at a rate similar to the increase in candidate size of Figure 4.3a. This follows our analysis of time complexity in Section 4.2.5, which tells us that the time complexity of CLASSY grows proportionally to the dataset size times the candidate set size,



(e) number of rules

Figure 4.3: Influence of the minimum support threshold on {candidate set size; runtime (in minutes); relative compression on the training set; AUC in the test set; number of rules} for a maximum rule length of 4 and a minimum support threshold per class of $n_{min.} = \{0.1\%, 0.5\%, 1\%, 2\%, 5\%, 10\%, 15\%, 20\%, 25\%\}$. The values were averaged over 10 times repeated 10-fold crossvalidation and each dataset is connected by a line to aid visualization. The vertical dashed line represents the selected minimum support of 5% used in the experiments section for predictive rule lists (Section 4.3).

thus, given a fixed dataset size it becomes proportional only to the candidate set size.

Compression and classification. Figures 4.3c and 4.3d show how CLASSY performs in classification in terms of compression in the training set and AUC in the test set, respectively. The values for both plots remain constant for most cases, and when a value deteriorates in terms of compression (increase in compression ratio) for smaller candidate sets, it also deteriorates accordingly in terms of AUC (decrease in AUC) in the test set.

We make two important observations: 1) the minimum in compression is achieved at the minimum support used for 13 out of 17 datasets, and in the cases where it does not happen the difference in relative compression is below 1%, which tells us that CLASSY can find a good description of the data using large candidate sets, without too greedily using rules that only cover few instances; 2) the minimum in compression and maximum in AUC are achieved for the same support value for 12 out 17 cases, and in the other cases, the difference is usually smaller than 2% in both measures, revealing the robustness of our MDL formulation at obtaining models that generalize well. The main exception is *ionosphere*, where the best AUC is found at the minimum

support threshold of 10%, while the lowest threshold finds a predictive rule list with 3% lower AUC, without almost any change in compression. This can be explained by the relatively small number of examples of *ionosphere* (351 instances) combined with its peculiar structure.

Number of rules. Figure 4.3e shows the number of rules selected based on the candidate set. As expected, the number of rules selected only decreases or remains constant with the candidate set, except for *mushroom*. Upon closer inspection, we observe that this is due to the disappearance of a rule with good performance but low coverage from the candidate set, which has to be replaced by a combination of other rules. The cases where many more rules are selected for lower minimum support thresholds, such as *chessbig* and *adult*, have lower compression and higher AUC values for these large number of rules, which makes these selections sustainable.

4.3.3 Classification performance

We now compare the classification performance of CLASSY to Scalable Bayesian Rule Lists (SBRL) [125], JRip[4], FURIA[4], CART[5], C5.0[6], and Support Vector Machines[7] (SVM). These methods are state-of-the-art classifiers, and SBRL, CART, C5.0, JRip, and FURIA—a fuzzy unordered rule induction algorithm—are clearly related to our approach. C5.0 is a newer version of C4.5, and JRip is a Java-implementation of RIPPER.

Hyperparameters tuning. CLASSY has no hyperparameters apart from the candidate set, which was generated using FP-growth with $n_{min} = 5\%$ and $d_{max} = 4$ for each dataset (as described at the beginning of Section 4.3). We tuned CART by selecting the best performing model on the training set from the models generated with the following complexity parameters: $\{0.001; 0.003; 0.01; 0.03;\ 0.1\}$. The same was done for C5.0, with confidence factors: $\{0.05; 0.15; 0.25; 0.35; 0.45\}$. The SVM, with the radial kernel, was tuned using 3-fold cross-validation and a grid search on $\gamma = \{2^{-6:0}\}$ and $c = \{2^{-4:4}\}$ within the training set. JRip and FURIA were tuned by setting their hyperparameters to 3 folds, a minimum weight of 2, and 2 optimization runs.
SBRL was trained using the guidelines provided by the authors [125]: the number of chains was set to 25; iterations to 5000; η, representing the average size of patterns in a rule, to 1; and λ, representing the average number of rules, to 5. The algorithm was first run on the training set and then re-run with λ changed to the number of

[4] `https://cran.r-project.org/package=RWeka`
[5] `https://cran.r-project.org/package=rpart`
[6] `https://cran.r-project.org/package=C50`
[7] `https://cran.r-project.org/package=e1071`

predictive rules obtained. In an attempt to follow their guidelines to use around 300 candidate rules, minimum and maximum itemset length were set to 1 and 2 (or 3 if possible) respectively, while the minimum threshold for the negative and positive classes was set to one of $\{5\%, 10\%, 15\%\}$. Note that we initially attempted a fair comparison by using the same candidates for SBRL as for CLASSY, but due to the limitations on the number of rules that SBRL could practically handle this, unfortunately, turned out to be infeasible.

Analyse of results. The results are presented in Table 4.2. The SVM models achieve the best ranking overall, but they do not belong to the class of interpretable models.
CLASSY performs on par with most tree- and rule-based models in terms of accuracy and balanced accuracy, worst than FURIA for these two measures, and better than these in terms of AUCs for multiclass datasets. The better performance of FURIA can be explained by the fact that it uses fuzzy rule sets rather than probabilistic rule lists; this allows for multiple rules to be activated and aggregated for a single classification, which improves predictive performance but makes interpretability less straightforward. This also means that the number of rules and conditions cannot be directly compared: a FURIA rule set consisting of 5 rules translates to up to 32 unique rules in the rule list setting. Further, FURIA does not provide probabilistic predictions, unlike our approach.
Comparing to other predictive rule list models, such as SBRL and JRip, CLASSY performs better for most of the measures used. When viewed against the tree-based models, we can see that our method performs on par with CART for most measures and slightly worse than C5.0, except for AUC in the multiclass scenario. Also, as we will show later, C5.0 tends to obtain equivalent rule lists that are much bigger than the ones produced by CLASSY, which makes them perform better in general (but not always).

4.3.4 Interpretability

The results are shown in Table 4.3, where we use AUC, the number of rules, and the number of conditions to compare the trade-off between AUC and model complexity of the tree- and rule-based models. Note that we choose AUC for predictive performance as it agrees with our goal of using the probabilities output of CLASSY to explain the decisions made. Also, note that we intentionally removed FURIA from the rankings of the number of rules and conditions as its models are rule sets—not rule lists.
For binary datasets, CLASSY is in a middle-ranking, better than SBRL and JRip, and worst than CART, C5.0, and FURIA. On the other hand, in multiclass datasets, it achieves a much lower (=better) ranking than all the other algorithms.

Table 4.2: Classification performance average results (10 times repeated 10-fold cross-validation) measured through {accuracy; balanced accuracy; Area Under the ROC Curve (AUC) (weighted AUC for multiclass datasets)}, per dataset {binary; multiclass} for each algorithm. At the bottom of the binary and multiclass datasets the average rank of the algorithm for each is presented. The rank of 1 is given for the best value (highest classification performance) and 7 and 6 for the worst in binary and multiclass, respectively. Note that SBRL cannot do multiclass classification, hence only being present in the binary ranking and the blank spaces.

	Accuracy							Balanced accuracy							AUC						
datasets	CLASSY	SBRL	JRip	CART	C5.0	FURIA	SVM	CLASSY	BRL	JRip	CART	C5.0	FURIA	SVM	CLASSY	BRL	JRip	CART	C5.0	FURIA	SVM
hepatitis	0.83	0.78	0.79	0.79	0.79	0.79	0.83	0.66	0.59	0.65	0.67	0.65	0.61	0.70	0.67	0.62	0.64	0.72	0.68	0.70	0.85
ionosphere	0.89	0.88	0.90	0.91	0.90	0.90	0.92	0.87	0.86	0.89	0.89	0.89	0.88	0.91	0.88	0.88	0.89	0.92	0.92	0.91	0.96
horsecolic	0.80	0.84	0.81	0.83	0.83	0.83	0.84	0.78	0.82	0.80	0.81	0.82	0.81	0.82	0.82	0.83	0.81	0.85	0.85	0.84	0.88
cylBands	0.70	0.68	0.73	0.73	0.74	0.76	0.81	0.65	0.65	0.72	0.71	0.73	0.74	0.80	0.70	0.73	0.74	0.78	0.78	0.79	0.88
breast	0.93	0.94	0.93	0.93	0.94	0.94	0.94	0.94	0.95	0.94	0.94	0.95	0.95	0.95	0.94	0.95	0.96	0.95	0.95	0.95	0.96
pima	0.73	0.74	0.73	0.73	0.73	0.74	0.74	0.66	0.69	0.68	0.67	0.67	0.68	0.67	0.70	0.69	0.68	0.71	0.69	0.68	0.75
tictactoe	0.98	0.81	0.98	0.92	0.94	0.99	0.99	0.98	0.75	0.97	0.91	0.93	0.98	0.99	0.98	0.86	0.97	0.97	0.98	1.00	1.00
mushroom	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
adult	0.85	0.85	0.85	0.85	0.85	0.80	0.86	0.75	0.75	0.74	0.75	0.76	0.74	0.76	0.89	0.88	0.74	0.88	0.87	0.76	0.86
rank	4.4	4.6	4.8	4.9	4.1	3.3	1.9	4.9	4.3	4.4	4.7	3.6	3.8	2.3	4.6	4.8	5.4	3.9	3.8	3.8	1.7
iris	0.95		0.94	0.93	0.93	0.93	0.94	0.95		0.94	0.93	0.93	0.93	0.94	0.97		0.97	0.97	0.97	0.95	0.99
wine	0.89		0.88	0.87	0.89	0.93	0.95	0.90		0.89	0.87	0.90	0.92	0.95	0.95		0.93	0.93	0.95	0.96	1.00
waveform	0.75		0.77	0.76	0.76	0.78	0.80	0.75		0.77	0.76	0.76	0.78	0.80	0.91		0.87	0.90	0.90	0.86	0.94
heart	0.56		0.54	0.57	0.54	0.57	0.59	0.30		0.22	0.31	0.30	0.27	0.30	0.74		0.54	0.76	0.72	0.69	0.84
pageblocs	0.93		0.93	0.93	0.92	0.92	0.93	0.51		0.52	0.52	0.48	0.52	0.52	0.74		0.72	0.74	0.69	0.73	0.72
led7	0.74		0.72	0.75	0.75	0.74	0.76	0.74		0.72	0.75	0.75	0.74	0.76	0.94		0.92	0.94	0.94	0.88	0.95
pendigits	0.92		0.95	0.92	0.96	0.97	0.98	0.92		0.95	0.92	0.96	0.97	0.99	0.99		0.98	0.99	1.00	0.99	1.00
chessbig	0.50		0.52	0.49	0.78	0.62	0.93	0.44		0.61	0.41	0.81	0.67	0.92	0.90		0.81	0.87	0.96	0.67	1.00
rank	4.0		3.4	3.2	2.6	1.6	1.1	4.1		4.4	4.1	4.1	3.2	1.3	2.6		5.1	3.8	3.4	4.6	1.5

CLASSY tends to find more compact models, with a similar number of rules and fewer logical conditions in total, than C5.0, CART, and JRip, that are as accurate or better than these. This can be seen by its average rank of 2 and 1.9 for rules and 1.7 and 1.5 for the total number of conditions, for binary and multiclass datasets respectively. It also can be seen that for most datasets it obtained the lowest number of conditions of all tree- and rule-based classifiers. Although SBRL also finds very compact rule lists, with a small number of rules and conditions, the low variance between the reported values for the different datasets suggests that this strongly depends on the hyper-parameter settings, which penalize too strongly the number of rules not around the user-defined expected average number of rules. Indeed, the compact rule lists exhibit subpar classification performance for some datasets (i.e., *hepatitis* and *tictactoe*). This suggests that without additional (computation-intensive) tuning of these hyperparameters, the recommended procedure for SBRL may lead to underfitting. As expected, C5.0, with its tendency to maximize the classification performance as much as possible, tends to create overgrown models, such as the almost 3000 rules for *chessbig*, that do not necessarily generalize well, such is the case in *adult*, where it obtained the same number of rules as CLASSY but with a 2% lower AUC, and for *pendigits* were it obtained a number of rules around 4 times higher than CLASSY and CART for the same performance.

4.3.5 Statistical significance testing

To analyze whether the results of Tables 4.2 and 4.3 are statistically different [25], we use two non-parametric multiple hypothesis tests, namely Friedman's test [37] and Iman and Davenport's test [57], on the rankings of the algorithms.
The results can be seen in the left side of Table 4.4, which divides the datasets into two groups, for binary and multiclass datasets respectively. The results show that there are significant differences for most measures (significance level 0.05). The only exceptions are balanced accuracy in the binary case, AUC of rule-based models in the binary case, and the number of rules for the multi-class case.

For those cases where the null hypothesis—stating that the algorithms perform on par—is rejected we proceed with a post-hoc Holm's test [54] for pairwise comparisons with CLASSY as control algorithm.
The results of these pairwise comparisons can be seen in the right side of Table 4.4. For most of these tests the null hypothesis—stating that CLASSY and its competitor perform on par—can not be rejected. This can be mostly explained by the relatively small number of datasets; the power of the tests is not very high. We therefore cannot draw strong conclusions from these results, but this is not necessarily a negative

Table 4.3: Interpretability performance average results (10 times repeated 10-fold cross-validation) of tree- and rule-based models measured through {Area Under the ROC Curve (AUC) (weighted AUC for multiclass datasets); number of rules; number of conditions }, per dataset {binary; multiclass} for each algorithm. Rank gives the average rank of each algorithm for binary and multiclass datasets. The rank of 1 is given for the best value (highest AUC or lowest number of rules/conditions). Note that SBRL cannot do multiclass classification, hence only being present in the binary ranking and the blank spaces.
∗ The number of rules and conditions used by FURIA are presented as a reference, as they are not directly comparable to those of the other methods as they form rule sets (and cannot be trivially translated to rule lists). FURIA was therefore also not included in the rankings of those criteria.

	AUC						Number of rules						Number of conditions					
datasets	CLASSY	SBRL	JRip	CART	C5.0	FURIA	CLASSY	SBRL	JRip	CART	C5.0	FURIA	CLASSY	SBRL	JRip	CART	C5.0	FURIA
hepatitis	0.67	0.62	0.64	0.72	0.68	0.70	2	2	3	4	9	4*	1	2	5	6	38	7*
ionosphere	0.88	0.88	0.89	0.92	0.92	0.91	6	3	6	5	12	7*	4	4	9	10	58	12*
horsecolic	0.82	0.83	0.81	0.85	0.85	0.84	4	2	4	6	16	6*	3	2	7	10	76	9*
cylBands	0.70	0.73	0.74	0.78	0.78	0.79	5	3	7	20	59	11*	4	4	17	111	897	18*
breast	0.94	0.95	0.96	0.95	0.95	0.95	3	3	4	5	6	5*	3	5	11	13	15	13*
pima	0.70	0.69	0.68	0.71	0.69	0.68	3	2	3	10	11	2*	3	3	3	23	47	2*
tictactoe	0.98	0.86	0.97	0.97	0.98	1.00	9	6	10	24	42	10*	21	15	27	66	254	17*
mushroom	1.00	1.00	1.00	1.00	1.00	1.00	6	5	5	8	9	8*	7	8	7	25	35	17*
adult	0.89	0.88	0.74	0.88	0.87	0.76	53	13	17	22	52	21*	114	25	81	106	630	34*
rank	3.8	4.0	4.4	3.0	2.9	2.9	2.6	1.1	2.8	3.7	4.9	*	1.7	1.7	2.7	3.9	5.0	*
iris	0.97		0.97	0.97	0.97	0.95	3		3	3	3	2*	2		2	3	3	3*
wine	0.95		0.93	0.93	0.95	0.96	5		5	5	8	4*	4		7	6	24	6*
waveform	0.91		0.87	0.90	0.90	0.86	25		23	46	81	15*	65		108	125	683	33*
heart	0.74		0.54	0.76	0.72	0.69	5		3	11	39	2*	4		6	28	299	2*
pageblocs	0.74		0.72	0.74	0.69	0.73	13		8	10	9	3*	13		9	20	37	4*
led7	0.94		0.92	0.94	0.94	0.88	20		19	29	28	3*	46		74	43	138	6*
pendigits	0.99		0.98	0.99	1.00	0.99	78		107	66	266	74*	221		439	30	2934	133*
chessbig	0.90		0.81	0.87	0.96	0.67	195		418	118	2874	281*	483		2688	25	48826	803*
rank	1.8		4.3	2.9	2.4	3.8	2.3		2.0	2.2	3.5	*	1.5		2.4	2.1	4.0	*

outcome: we aimed at showing that CLASSY performs as well as other rule- and tree-based algorithms while obtaining simpler models. The results show that CLASSY does use significantly fewer conditions than C5.0 for both multiclass and binary datasets, and then CART for the binary case. Also, as expected the SVM obtained always better results than CLASSY except for multiclass AUC. FURIA was better in terms of accuracy but worse in terms of AUC.

4.3.6 Overfitting

To study overfitting, we compared the averages of the absolute difference between the AUC values in the training and test set over 10 times repeated 10-folds for each algorithm. The results can be seen in Table 4.5. In general, CLASSY together with SVM, seem to be the most consistent algorithms in obtaining the lowest values. The usual performance of CLASSY is 5% or lower, 12 out of 17 times, except in the case of *hepatitis* where it got 13%, which was the best value after SVM. SBRL is very consistent, clearly achieving the lowest values for binary datasets, however, this can be explained by its more conservative choice of rules and thus lower AUC on the test set as shown in Table 4.2. Comparing with all rule- and tree-based models, CLASSY obtained the lowest ranking for multiclass datasets, being, from these ones, the algorithm that less overfits overall.

4.3.7 Runtime

All runtimes are averages over ten times repetitions of ten folds, run on a 64-bit Windows Server $2012R2$, with Intel Xeon E5-2630v3 CPU at 2.4GHz and 512GB RAM. Runtimes include parameter tuning where applicable and candidate mining for CLASSY and SBRL.
The results are depicted in Figure 4.4. CART, C5.0, JRip, and FURIA are the fastest, with most runtimes under 1 minute with CLASSY being at a maximum one order of magnitude slower. Comparing to SBRL, CLASSY is 10 times faster, even though it considers around 100 times more candidates than this and performs better in terms of AUC. The worst runtimes were obtained for SVM, due to its costly grid search.
It should be noticed that reducing the candidate set size of CLASSY would have an exponential reduction in its runtimes without much deterioration of its classification performance, as can be seen in Figures 4.3b and 4.3d.

4.3.8 Discussion

From the classification and interpretability results of Table 4.2 and 4.3, it can be seen that CLASSY can provide a good trade-off between classification performance and rule

Table 4.4: Statistical significance testing of differences between the algorithms. Results for Friedman (χ^2 statistic), and Iman and Davenport (F statistic) tests for all the measures presented in Table 4.2 and 4.3 for binary and multiclass datasets with a significance level of 0.05. In case the null hypothesis for the differences is rejected, the post-hoc Holm's procedure is used for pairwise comparisons with CLASSY as control. AUC_{all} is the AUC comparison with all algorithms (SVM included) of Table 4.2 and AUC_{rules} is the AUC comparison of all tree- and rule-based algorithms (SVM excluded) of Table 4.3. p represents the p-value obtained for each specific test, **R** the rejection of $\mathcal{H}_0$—null hypothesis. Note that not all algorithms can be tested for all measures, thus k shows the number of algorithms tested for each measure.

			Difference tests						Holm's post-hoc procedure (CLASSY as control)											
			Friedman			Iman and Davenport			SBRL		JRip		CART		C5.0		FURIA		SVM	
Measures	Classes	k	χ^2	p	$\mathcal{H}_0$	F	p	$\mathcal{H}_0$	p	$\mathcal{H}_0$	p	$\mathcal{H}_0$	p	$\mathcal{H}_0$	p	$\mathcal{H}_0$	p	$\mathcal{H}_0$	p	$\mathcal{H}_0$
Acc	binary	7	13.36	0.038	**R**	2.63	0.028	**R**	0.827	—	0.703	—	0.585	—	0.743	—	0.300	—	0.014	**R**
	multi	6	17.34	0.004	**R**	5.36	<0.001	**R**			0.504	—	0.385	—	0.142	—	0.009	**R**	0.002	**R**
bAcc	binary	7	8.94	0.177	—	1.59	0.171	—												
	multi	6	15.71	0.008	**R**	4.53	0.003	**R**			0.738	—	1.000	—	1.000	—	0.350	—	0.003	**R**
AUC_{all}	binary	7	16.00	0.014	**R**	3.37	0.007	**R**	0.827	—	0.445	—	0.300	—	0.413	—	0.413	—	0.005	**R**
	multi	6	20.00	0.001	**R**	7.00	<0.001	**R**			0.008	**R**	0.229	—	0.423	—	0.033	—	0.229	—
AUC_{rules}	binary	6	5.70	0.337	—	1.16	0.346	—												
	multi	5	13.10	0.011	**R**	4.85	0.004	**R**			0.002	**R**	0.155	—	0.429	—	0.011	**R**		
Rules	binary	5	28.18	<0.001	**R**	28.82	<0.001	**R**	0.053	—	0.766	—	0.136	—	0.002	**R**				
	multi	4	6.64	0.084	—	2.68	0.073	—												
Conditions	binary	5	29.40	<0.001	**R**	35.64	<0.001	**R**	1.000	—	0.205	—	0.011	**R**	<0.001	**R**				
	multi	4	16.35	0.001	**R**	14.96	<0.001	**R**			0.175	—	0.333	—	<0.001	**R**				

Table 4.5: Overfitting average results (10 times repeated 10-fold cross-validation) using the absolute difference between AUC performance in training and test sets as a measure, per fold, for each algorithm and each dataset. Rank gives the average rank of each algorithm for binary and multiclass datasets. Note that SBRL does not have values for multiclass datasets.

	$\lvert AUC_{train} - AUC_{test} \rvert$						
datasets	CLASSY	SBRL	JRip	CART	C5.0	FURIA	SVM
hepatitis	0.13	0.16	0.19	0.14	0.21	0.22	0.13
ionosphere	0.06	0.05	0.07	0.04	0.05	0.08	0.04
horsecolic	0.06	0.06	0.08	0.06	0.09	0.08	0.10
cylBands	0.07	0.06	0.09	0.11	0.18	0.13	0.12
breast	0.02	0.02	0.02	0.02	0.02	0.02	0.02
pima	0.06	0.05	0.05	0.06	0.07	0.05	0.05
tictactoe	0.01	0.03	0.01	0.02	0.02	0.00	0.00
mushroom	0.00	0.00	0.00	0.00	0.00	0.00	0.00
adult	0.01	0.00	0.01	0.00	0.01	0.01	0.03
rank	3.6	2.7	4.4	4.2	5.2	4.3	3.6
iris	0.02		0.02	0.02	0.02	0.04	0.01
wine	0.03		0.05	0.04	0.04	0.03	0.00
waveform	0.02		0.02	0.02	0.03	0.02	0.02
heart	0.05		0.04	0.07	0.15	0.04	0.06
pageblocs	0.00		0.00	0.00	0.00	0.00	0.00
led7	0.01		0.01	0.01	0.01	0.01	0.01
pendigits	0.00		0.01	0.00	0.00	0.01	0.00
chessbig	0.00		0.02	0.00	0.02	0.00	0.00
rank	2.4		4.8	4.4	4.1	3.6	1.8

list size. Particularly, in the case of multiclass datasets, such as *chessbig*, where classical algorithms like JRip find a model with double the number of rules, or in the case of *mushroom*, where CART and C5.0 find more complex models with the same performance. It is interesting to notice that CLASSY performs better in terms of AUC than accuracy. This shows that when it makes a wrong prediction it does so with a small probability, which is reassuring. Moreover, CLASSY has only one hyperparameter—its candidate set—which its tuning is hardly needed as the algorithm has no problem in dealing with large numbers of candidates. This is quite different from the extensive tuning done for the other methods. It is important to observe that *all methods except for* CLASSY *were tuned*.

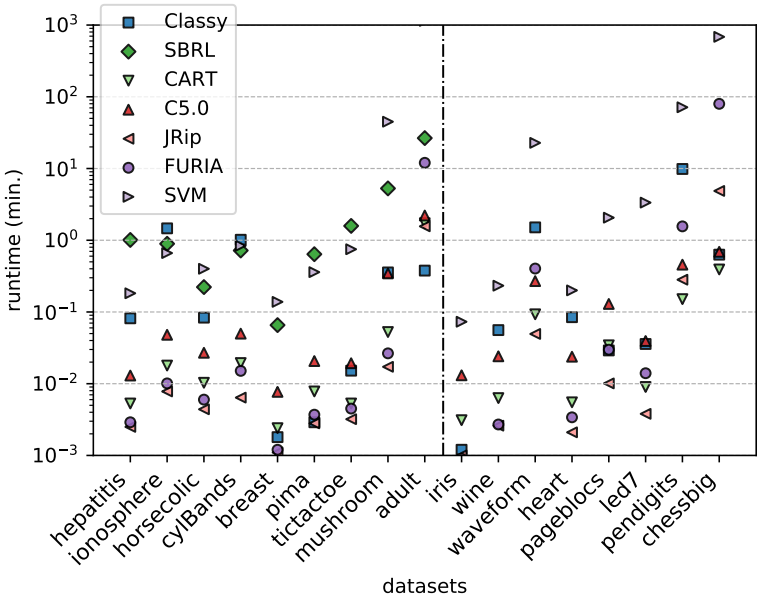


Figure 4.4: Average runtime per fold in minutes for each algorithm and each dataset. The datasets are ordered first by the number classes and then by number of samples (ascending). The vertical dashed line separates binary (to the left) from multiclass datasets. Note that SBRL does not have a runtime for multiclass datasets.

Candidate set influence. In the set of Figures 4.3, it is shown that larger candidate sets do not result in worse models, as our formalization in terms of the MDL principle is well-suited to avoid overfitting without the need for cross-validation and/or parameter tuning. In other words, CLASSY is insensitive to its only hyperparameter—its candidate set—making it virtually parameter-free. This is a big advantage, as one can simply run CLASSY on all training data with as many candidates as possible, without worrying about any parameters. It also means that all training data can be used for training, which is important in the case of small data: no data needs to be reserved for validation.

Compression and classification. From Figure 4.2 we can observe that better compression corresponds to better classification, which is a strong empirical validation of our formalization. As expected, the normalized gain is the best heuristic to use in combination with our greedy rule selection strategy, as it results in better classifiers for 88% of the datasets.

Runtime. From the runtimes of Figure 4.4, it can be seen that CLASSY runtimes are slower by an order of magnitude than other (fast) algorithms, such as C5.0, CART, and JRip, and similar to SBRL. This is expected for the size of candidate sets used in our experiments, as can be seen in Table F.1.

Classification and interpretability. In terms of classification and interpretability, comparing the average ranking with other rule- and tree-based methods in Table 4.3, it is shown that CLASSY performs equally well while also able to find rule lists with fewer conditions, *without any parameter tuning*. CART creates models with fewer rules that have more conditions per rule, while C5.0 has a high AUC at the expense of over-complex rules. FURIA has a better performance in terms of both standard and balanced accuracy, and worst in terms of AUC, which is expected as it is not a probabilistic classifier. Also, it is hard to compare its interpretability as all its rules can interact with each other, generating a much larger equivalent rule list than CLASSY. SBRL on the other hand seems to be able to find simple models that underperform in terms of AUC compared with CLASSY, which can be either a result of its formalization or because it cannot use larger candidate sets.
The experiments also revealed that the Poisson distribution used as prior in SBRL, for the number of conditions per rule and the number of rules, creates tight constraints from which the results hardly deviate. Our results suggest that if the 'optimal' values for these hyperparameters are not known in advance, the best model may not be found. An indicative example of this is the *tictactoe* dataset in Table 4.2, a deterministic dataset for which SBRL can only find the right amount of rules and logical conditions per rule when given these exact values in advance. The results obtained with CLASSY demonstrate that using the universal prior for integers alleviates this strong dependence on hyperparameter tuning.

Overfitting. In terms of overfitting, Table 4.5 shows that CLASSY tends to select models that generalize well and that are not overconfident in the training set. It obtains low differences between training and test compared with the other rule- and tree-based models.

4.4 Conclusions

We proposed CLASSY, a heuristic algorithm that finds good probabilistic rule lists for multiclass classification by greedily approximating our MDL-based formulation of the problem. CLASSY naturally trades off model complexity with predictive accuracy,

effectively avoiding overfitting with very few hyperparameters.
We empirically demonstrated, on a variety of datasets, that CLASSY finds predictive rule lists that perform on par with state-of-the-art interpretable classifiers for predictive accuracy, even though some form of hyperparameter tuning is done for all methods except for CLASSY. Moreover, the models found by our approach are more compact than those obtained by the other methods, which we expect to make them more understandable in practice. Finally, we show that compression strongly correlates with predictive accuracy, which can be regarded as an empirical validation of the MDL-based selection criterion.

Limitations. The CLASSY algorithm is restricted to discretized input variables and multiclass problems. Most machine learning problems in tabular data involve a mix of binary, nominal and numeric input variables. Nonetheless, the formulation for those problems is available in Chapter 3, and one could directly use the RSD algorithm of Chapter 5 for multiclass problems. However, extending it to regression would require adaptations of the greedy gain used, as the problem is not as well defined as classification. In terms of statistical properties, and contrary to RSD, each rule added to the list per iteration does not minimize a specific statistical test, except decreasing the overall MDL score.

5

Discovering subgroup lists with RSD

In this chapter[1], we propose the Robust Subgroup Discoverer (RSD) algorithm based on the MDL formulation of subgroup lists proposed in Chapter 3. This algorithm uses a greedy heuristic to finding good subgroups and can be applied to supervised tabular datasets with univariate and multivariate, nominal and numeric targets. To validate it, we conduct an empirical comparison on 54 datasets against state-of-the-art-algorithms. This is complemented with two case studies of subgroup lists: 1) to describe the characteristics of hotel customers based on the time in advance they make reservations; and 2) to describe how the social-economic background of Colombia Engineering students is associated with their performance in university national exams.

Recapitulation of subgroup lists and their MDL formulation. In previous chapters we defined what a subgroup list and its definition of optimality according to the MDL principle are. We will now restate those definitions here. First, let us recall from Chapter 2.4 the subgroup lists model in Figure 5.1.
The best subgroup list SL according to the MDL principle is the one that given the dataset D minimizes the two-part code defined in Chapter 3.1:

$$SL^* = \underset{SL \in \mathcal{SL}}{\arg\min}\, L(D, SL) = \underset{SL \in \mathcal{SL}}{\arg\min} \left[L(\mathbf{Y} \mid \mathbf{X}, SL) + L(SL) \right],$$

where $L(SL)$ is the length of encoding the subgroup list model SL, and $L(\mathbf{Y} \mid \mathbf{X}, SL)$ is the length of encoding the target variables data given the subgroup list SL and the explanatory variables $\mathbf{X}$. The *model encoding* is the same for predictive rule lists and

[1] Parts of this chapter are based on Proença et al. [99, 100]

$$
\begin{array}{llllll}
s_1\text{:} & \text{IF} & a_1 \sqsubseteq \mathbf{x} & \text{THEN} & y_1 \sim Dist(\hat{\Theta}_1^1) \quad \cdots & y_t \sim Dist(\hat{\Theta}_t^1) \\
 & & \vdots & & & \\
s_\omega\text{:} & \text{ELSE IF} & a_\omega \sqsubseteq \mathbf{x} & \text{THEN} & y_1 \sim Dist(\hat{\Theta}_1^\omega) \quad \cdots & y_t \sim Dist(\hat{\Theta}_t^\omega) \\
\text{default:} & \text{ELSE} & & & y_1 \sim Dist(\hat{\Theta}_1^d) \quad \cdots & y_t \sim Dist(\hat{\Theta}_t^d)
\end{array}
$$

Figure 5.1: Generic subgroup list model SL with ω subgroups $S = \{s_1, ..., s_\omega\}$ and t (number of target variables) distributions per subgroup. Note that the parameters of the default rule of a subgroup list $\hat{\mathbf{\Theta}}^d = \{\hat{\Theta}_1^d, \cdots, \hat{\Theta}_t^d\}$ are fixed to the marginal distribution of the dataset, i.e., the overall category prior for categorical variables and the dataset mean and standard deviation for numeric targets.

subgroup lists, as they only differ on how the default rule encodes the data and was defined in Chapter 3.2 as

$$
L(SL) = L_{\mathbb{N}}(|S|) + \sum_{a_i \in S} \left[L_{\mathbb{N}}(|a_i|) + \log \binom{m}{|a_i|} + \sum_{v \in a_i} L(v) \right],
$$

where S is the list subgroups in SL, i.e., the model excluding the default rule. Then, depending on the type of target data the *data encoding* can vary. In the case of *nominal* target variables, we use categorical distribution and the Normalized Maximum Likelihood encoding defined in Chapter 3.4:

$$
L(\mathbf{Y} \mid \mathbf{X}, SL) = \sum_{j=1}^{t} \left(L(Y_j^d \mid \mathbf{\Theta}^d) + \sum_{s_i \in S} L_{NML}(Y_j^i) \right).
$$

In the case of *numeric* target variables, we use categorical distribution and the Normalized Maximum Likelihood encoding defined in Chapter 3.5:

$$
L(\mathbf{Y} \mid \mathbf{X}, SL) = \sum_{j=1}^{t} \left(L(Y_j^d \mid \mathbf{\Theta}^d) + \sum_{s_i \in S} L_{Bayes2.0}(Y_j^i) \right).
$$

Structure of the chapter. This chapter is divided as follows. First, in Section 5.1 the most relevant related work is covered, together with the main differences to our approach. After that, in Section 5.2 RSD, a heuristic algorithm to mine subgroup lists is defined, as well as its statistical guarantees and time complexity. Then, in Section 5.3 we show the empirical results of our proposed method when compared against the state-of-the-art algorithms for univariate and multivariate nominal and numeric targets over 54 datasets. After that, in Section 5.4 we show a simple case

study of RSD applied to hotel bookings. Then, in Section 5.5 we apply RSD to discover flight delays in an airline dataset. After that, in Section 5.6 we apply robust subgroup discovery to find how descriptions of the socioeconomic background affect the grades of engineering students in Colombia. Finally, in Section 5.7 the main conclusions are presented.

5.1 Related work

In this section we cover work related to our proposed MDL subgroup lists, in three categories: *subgroup discovery*; *pattern mining*; *MDL for pattern mining*; and *algorithmic implementations*. The relevance of each topic is as follows: subgroup discovery directly relates to the task at hand; pattern mining and association rule mining are generalizations of subgroup discovery; MDL for pattern mining shares the same theory for formalizing the problem; and lastly we go over most of the same works but from an algorithm implementation perspective to justify our algorithmic choices. Note that predictive rule lists are also related as they share the same model structure as subgroup lists and for more details on that we refer the interested reader to Chapter 4.

5.1.1 Subgroup discovery

In its traditional form, subgroup discovery, also referred to as top-k subgroup mining [8], entails the mining of the k top-ranking subgroups according to a quality measure and a number k selected by the user. As mentioned in the introduction, this formulation suffers from three main issues that make it impractical for most applications: $1)$ poor *efficiency of exhaustive search* for more relevant quality measures [12]; $2)$ *redundancy of subgroup sets* mined, i.e., the fact that subsets with the highest deviation according to a certain quality measure tend to cover the same region of the dataset with slight variations in their description of the subset [75]; $3)$ lack of *statistical guarantees* and generalization of mined subgroups [77]. We will now go over the recent contributions for these three issues, with special emphasis for the last two, redundancy and statistical guarantees, which our work proposes to solve.

Efficient exhaustive search. In the last years, several developments have been made towards more efficient algorithms for mining the top-k subgroups. Lemmerich et al. [80] proposed an efficient exhaustive search algorithm for numerical targets, Belfodil et al. [9] proposed to mine over numeric attributes with guarantees, and Boley et al. [12] proposed an algorithm that exhaustively mines subgroups that take into account the dispersion (deviation) of the subgroups target distribution. Subgroup discovery extension from deviations of distributions of target variables to deviations between

models is also called Exceptional model mining [79, 32], and can be applied to models such as Bayesian Networks [31] or non-parametric spatio-temporal patterns [28]. Comparing to our approach these works do not take into account the redundancy of the subgroups found, and thus, the subgroups found tend to overlap in the same region of the dataset.

Redundancy of subgroup sets. To address redundancy among subgroups most previously proposed approaches encompass supervised pattern set mining [16], and methods based on relevance [46] and diversity [74, 75]. Unlike diversity-based methods, the supervised pattern set mining objective is to find a fixed number of patterns, which has to be chosen in advance, while relevance is limited to non-numeric targets. It is this last group, the diversity-based methods that share most similarities to our work, i.e., the area of *Subgroup Set Discovery*.

The main approaches in *Subgroup Set Discovery* are CN2-SD [71], Diverse Subgroup Set Discovery (DSSD) [75], *Skylines* of subgroup sets [76], Monte Carlo Tree Search for Data Mining (MCTS4DM) [14], Subjectively Interesting Subgroup Mining (SISD) [83], and FSSD [10]. The differences between Subgroup Set Discovery methods are summarized in Table 5.1, with RSD representing our approach and where all methods are compared in terms of: if they use a list or a set; the target variables they support; if they have statistical guarantees; if they have automatic stopping criteria (not defined by the user); and if they have a global definition of a subgroup set or list.

Considering the methods in more detail, CN2-SD [71] was one of the first methods to deal with redundancy and is a direct adaptation of CN2, a classical rule learner, and can be applied to nominal target variables. It uses a sequential approach, wherein each iteration adds one subgroup to the set, and then removes the data covered by that subgroup until no more data can be covered in this way. DSSD [75] developed a technique based on a novel measure of overlap between subgroups, to iteratively find a set of subgroups. It can be applied to single-and-multi-target nominal and numeric variables, with different types of quality measures. Skylines of subgroup sets [76] proposed to directly account for quality-diversity trade-off, to find the Pareto optimal subgroup sets of size k. MCTS4DM [14] uses Monte Carlo tree search to improve the quality of the subgroups found, although it can only be applied to binary target variables, and attributes of the same type (all numeric or all nominal). Subjectively interesting Subgroup Discovery [83] finds the subjectively most interesting subgroup for numeric target variables with regard to the prior knowledge of the user, based on an information-theoretic framework for formalizing subjective interestingness. By successively updating the prior knowledge based on the found subgroups, it iteratively mines a diverse set of subgroups that are also dispersion-aware. FSSD [10] is a more

recent approach that considers the 'union' of all subgroups as a single pattern by forming a disjunction of subgroups and evaluating its quality and can only be applied to binary target variables. This approach is similar to a sequential approach for mining subgroups although the individual contributions of each subgroup are dissolved in the 'new' subgroup formed by the disjunction of all subgroups.

Table 5.1: Comparison of Subgroup Set Discovery methods in terms of their key properties. From left to right: model class (list or set); types of supported target variables: binary, nominal, numeric and multi-target; *statistical* guarantees of the subgroups mined; automatic *stopping* criterion (not defined by the user); *global* formulation of a subgroup set/list.

		Target variables						
Method	Model	binary	nom.	num.	multi	Statistical	Stopping	Global
RSD	list	✓	✓	✓	✓	✓	✓	✓
CN2-SD[71]	list	✓	✓	-	-	-	-	-
DSSD[75]	set	✓	✓	✓	✓	-	-	-
Skylines[76]	set	✓	✓	-	-	-	-	✓
MCTS4DM[14]	set	✓	-	-	-	-	-	-
SISD[83]	set	-	-	✓	✓	✓	-	-
FSSD[10]	list	✓	-	-	-	-	✓	✓

Statistical guarantees. In terms of statistical guarantees to subgroup discovery, most approaches consider first mining the top-k subgroups and then post-processing them in terms of a test to find subgroups that are statistically significant [30, 77]. Duivesteijn and Knobbe [30] proposed to use random permutations of the target variable for a quality measure to evaluate how the discovered subgroups compare against the null hypothesis generated by those permutations. Later, van Leeuwen and Ukkonen [77] discussed the concept of significance for subgroup discovery and concluded that p-values should be used with caution as not all false discoveries can be removed in this way, as there will always be random subsets with large effect sizes. An exception to this is the work of Lijffijt et al. [83] (already mentioned in the last section), which uses the maximum entropy principle to iteratively find subgroups that are subjectively interesting against a user's prior knowledge. Our approach strongly deviates from the first two, as our method tests for statistical guarantees during the mining process, it is parametric, as we use categorical and normal distributions to model the targets, and also, through the use of MDL-based model encoding, we take into account the concept of a list of subgroups and penalize for all the possible subgroup lists that could be discovered in the dataset. Regarding the last approach, even though they

also mine subgroups iteratively, they lack a definition of an optimal subgroup set, and their goal is to model the user's subjective knowledge and find regions in the data that the user does not know much about.

5.1.2 Pattern mining

Pattern mining and association rule mining [2] are concerned with mining items that co-occur together, i.e., itemsets or patterns, and relationships between itemsets and a target item, e.g., a class, respectively. A key problem is that they suffer from the infamous *pattern explosion*, i.e., they tend to return enormous amounts of patterns/rules. To solve this problem, many approaches were proposed, but two stand out concerning our work, namely, association rule classifiers and statistical rule mining.

Association rule classifiers. A simple way to reduce the number of rules returned is by aggregating association rules in a set used for classification and using a performance measure to choose the best set. It is relevant to notice that classifiers based on association rule mining have a similar structure to predictive rule lists and subgroup lists, as they tend to order the rules sequentially. The best-known techniques are CBA [85] and CMAR [82], but they tend to obtain large numbers of rules. Similar to predictive rule lists, these methods aim to maximize the classification performance, and not to describe the deviations in the data. Another important difference is that these methods tend to return crisp decisions instead of probabilities and can in general only be applied to nominal targets.
A similar class of methods is that of *supervised pattern set mining* [128]. The key difference is that these methods do not automatically trade-off model complexity and classification accuracy, requiring the analyst to choose the number of patterns k in advance.

Statistical rule mining. The idea of mining rules with statistical guarantees is appealing as it increases the users' trust in the patterns found while at the same time reducing the number of rules returned by a miner [51]. The concept of statistical rule mining progressed by incrementally adding more statistical guarantees. Webb [124] proposed for the first time mining of statistically significant patterns, then Hämäläinen [49] proposed KingFisher, an efficient algorithm to mine dependent rules, i.e., rules that show a dependency with respect to a target in terms of a dependency test like Fisher's exact test. After that, Hämäläinen and Webb [50] added extra procedures to remove spurious relations from the miner findings. Lastly, the criteria under which causal rules can be mined was defined and an efficient algorithm to mine them was proposed [19]. All these methods focus on mining all the possible *individual* statistic-

ally significant (or causal) rules and not on finding a non-redundant set, as is the case of Subgroup Set Discovery. In this chapter, we aim to accomplish both at the same time, finding the best *global* subgroup list while assuring *local* statistically robust subgroups.

5.1.3 MDL in pattern mining

In the past, for models similar to our subgroup lists, the MDL principle has mostly been embedded in small parts of predictive algorithms to solve the problem of overfitting. Prominent examples of this are C4.5 [103] and RIPPER [22], which use the MDL principle to prune overfitting models, and help generalization.
In data mining, Krimp [120] was the first method to apply the MDL principle holistically, i.e., for the whole model selection process, unlike previously mentioned approaches that only used it for a subset of the model selection process. This seminal work used a version of crude MDL, i.e., a not completely optimal 'two-part' encoding of the data, to find the pattern list that compressed a transaction dataset best, to address the *pattern explosion* issue in pattern mining. Recent works have aimed at improving the encoding through the use of refined MDL for encoding the data, i.e., an encoding that enjoys optimal properties at least in expectation [48]. The first of such approaches was DiffNorm [18], which used a prequential plug-in code to improve the encoding of transaction data and recently MINT was proposed to mine real-valued pattern sets with a similar encoding[86]. Although Krimp, DiffNorm, and MINT are used to describe data, they aim at finding regularities—not deviations—and do not consider a target variable. For an in-depth survey of MDL in pattern mining please refer to the survey by Galbrun [40].
MDL has been used to find optimal sets of association rules for two-view data [73] and tabular data [35]. The latter is the most related to our work, as it aims to find rule sets that describe the data well. Similar to Krimp it aims at finding all associations in the data though, not at identifying deviations as we do, and no specific target variable(s) are defined.

5.1.4 Algorithmic comparison in the literature

Our proposed algorithm RSD (presented in Section 5.2) is based on a combination of beam search for candidate generation and greedy search for iteratively adding subgroups to the subgroup list. Both techniques have been widely employed for similar problems.
Greedy search has been often used for learning decision trees and predictive rule lists [103, 22, 39, 96], as well as for pattern-based modeling using the MDL principle

[120, 18, 73]. *Beam search* has been commonly used for candidate generation in subgroup discovery [87], including for finding subgroup sets [71, 75]. We next provide the motivation for our algorithmic choices, and describe key similarities and differences compared to algorithms in the most related literature: $1)$ predictive rule list learning; $2)$ subgroup set discovery; and $3)$ evolutionary algorithms for rule learning.

Algorithms for finding predictive rule lists. The common way to finding a good predictive rule list is through heuristic search [22, 39, 96], however recent works have proposed to find optimal models for binary classification under specific conditions [125, 5]. Belong to the former category, Proença and van Leeuwen [96] use a Separate and Conquer (SaC) technique to greedily add rules, together with Frequent Pattern Mining for candidate generation. In this chapter, the beam search for candidate generation does not require a discretized dataset, is faster, and without large loss in the quality of the subgroups found due to discretization [88]. In the latter category, of optimal predictive rule list discovery, the algorithms were only developed for binary classification, and either require a simplification of the rules in the list to decision rules—with true or false instead of probabilities as consequent—combined with a simple objective function, such as accuracy, that allows for efficient branch and bound [5], or it requires the dataset to be sparse and small, with large minimum supports for the rules (above 10%) and using convergence to an optimal algorithm such as Monte Carlo sampling [125]. Neither of these approaches can deal with a variety of target variables as our proposed approach can.

Algorithms for subgroup set discovery. Both beam search and greedy search are commonplace in subgroup set discovery [71, 75], due to their efficiency and flexibility in being applied to different types of targets. More recently, Monte Carlo Tree Search (MCTS) was proposed for mining sets of subgroups [14], although it can only be applied to binary targets and specific types of explanatory variables. In the classical case of mining top-k subgroups without incorporating diversity, exhaustive search is feasible [12], but again it is only efficient for specific types of quality measures or targets and does not scale well for finding the best set [75]. Together with the fact that the loss in quality of using beam search is almost negligible [88], exact algorithms are rarely used in MDL-based data mining, because it is infeasible [120, 18, 35, 96].

Evolutionary algorithms. Global heuristics, such as evolutionary algorithms, have been applied to fuzzy rule-based model learning [34], and although they could also be applied here, we found that the arguments in favor of a local search approach were stronger: $1)$ local heuristics have often been successfully applied for pattern-based modeling using the MDL principle, making it a natural approach to consider;

2) local heuristics are typically faster than global heuristics, as much fewer candidates need to be evaluated; 3) global heuristics typically require substantially more (hyper)parameters that need to be tuned (e.g., population size, selection and mutation operators, etc.), while local heuristics have very few.

5.2 The RSD Algorithm

In this section we propose the *Robust Subgroup Discoverer* (RSD), a heuristic algorithm to find good subgroup lists based on the proposed MDL formulation. As the problem of finding an optimal subgroup list is NP-hard [90] we propose a heuristic based on the Separate-and-Conquer (SaC) [38] strategy of iteratively adding the local best subgroup to the list, combined with beam search for candidate subgroup generation. The use of greedy heuristic approaches is common practice in MDL-based pattern mining [120, 96] and rule-based learning [39], and beam-search is widely adopted for its efficient generation of subgroups in subgroup discovery [71, 87, 75].

This section is divided as follows. First, in Section 5.2.1 we give a high-level description of our proposed algorithm and motivate our choices. After that, in Section 5.2.2 the quality measure used to iteratively add rules—compression gain—is presented, together with its relationship with subgroup discovery quality measures. Then, in Section 5.2.3 the statistical testing interpretation of the compression gain is given. After that, in Section 5.2.4 the beam search for candidate subgroup generation is presented in detail. Then, in Section 5.2.5 the Separate-and-Conquer RSD algorithm is presented. Finally, in Section 5.2.6 the time and space complexity of the overall algorithm is given.

5.2.1 Algorithm high-level description

The algorithm we propose is a heuristic composed of two parts: we greedily add one subgroup at a time to the subgroup list, for which candidates are generated using beam search. More specifically, the greedy search algorithm starts from an empty list, with just a default rule equal to the priors in the data, and adds subgroups according to the well-known *separate-and-conquer* strategy [39]: 1) iteratively find and add the subgroup that gives the largest improvement in compression; 2) remove the data covered by that rule; and 3) repeat steps 1-2 until compression cannot be improved. This implies that *we always add subgroups at the end of the list, but before the default rule*. Beam search is used for candidate generation at each iteration to find the best candidate to add. Given a beam width w_b and maximum search depth d_{max} it consists of: 1) find all items, i.e., all conditioned variables such as $x_1 < 5$ or $x_2 = category$,

and add the best w_b items according to compression gain (Eq. (5.2.2)) as subgroups of size 1 to the beam; 2) refine all subgroups in the beam with all items and add the best w_b to a new empty beam; 3) repeat 2 and 3 until the maximum depth d_{max} of the beam is reached and return the best subgroup—according to the compression score—found in all iterations. The beam search algorithm is described in detail in Section 5.2.4 and the greedy search algorithm RSD in Section 5.2.5.

The main reasons for using greedy search and adding one subgroup at a time are its computational simplicity and transparency, as it adds at each iteration the locally best and most statistically significant subgroup found by the beam search. Further, in the context of subgroup discovery beam search was empirically shown to be very competitive in terms of quality when compared to a complete search, while it demonstrates a considerable speed-up [88]. Also, its straightforward implementation allows flexibility to easily extend this framework to other types of targets in the future.

5.2.2 Compression gain

To quantify the quality of annexing a subgroup s at the end (after all the other subgroups and before the default rule) of subgroup list SL, denoted $SL \oplus s$, we employ the *compression gain*:

$$s^* = \arg\max_{s \in \int} \Delta_\beta L(D, SL \oplus s) = \arg\max_{s \in \int} \left[\frac{L(D, SL) - L(D, SL \oplus s)}{(n_s)^\beta} \right], \ \beta \in [0, 1] \tag{5.1}$$

where β weighs the level of the normalization, and $\Delta_\beta L(D, SL \oplus s)$ should be greater than zero for a decrease in the encoded length from $L(D, SL)$ to $L(D, SL \oplus s)$. Considering the extremes, with $\beta = 1$ we have the *normalized gain* first introduced for the classification setting by Proença and van Leeuwen [96], and for $\beta = 0$ we have the *absolute gain* which is just the regular gain used in the greedy search of previous MDL-based pattern mining [120].

Developing Eq. (5.1) further shows that the compression gain only depends on the added subgroup s, as in the specific case of a subgroup list the default rule is fixed and it is the same for M and $M \oplus s$:

$$\begin{aligned} \Delta_\beta L(D, SL \oplus s) &= \frac{L(\mathbf{Y} \mid \mathbf{X}, SL) - L(\mathbf{Y} \mid \mathbf{X}, SL \oplus s)}{(n_s)^\beta} + \frac{L(SL) - L(SL \oplus s)}{(n_s)^\beta} \\ &= \Delta_\beta L(\mathbf{Y} \mid \mathbf{X}, SL \oplus s) + \Delta_\beta L(SL \oplus s), \end{aligned}$$

where $\Delta_\beta L(\mathbf{Y} \mid \mathbf{X}, SL \oplus s)$ and $\Delta_\beta L(SL \oplus s)$ are the data and model compression gain, respectively.

Furthermore, if we note that maximizing the gain in Eq. (5.1) is equivalent to maximizing the subgroup discovery equivalent objective of Eq. (3.18) for nominal targets

and Eq. (3.25) for numeric targets, this means that finding the subgroup that maximizes the compression gain is the same as finding the subgroup that maximizes the subgroup discovery equivalent objective:

$$\begin{aligned} s^* &= \arg\max_{s \in \mathcal{S}} \Delta_\beta L(SL \oplus s) \\ &= \arg\max_{s \in \mathcal{S}} \frac{n_s KL(\hat{\Theta}_s; \hat{\Theta}_d)}{(n_s)^\beta} - \frac{\texttt{COMP}(n_s, \#param)}{(n_s)^\beta} + \Delta_\beta L(SL \oplus s) \end{aligned}$$

where $n_s KL(\hat{\Theta}_s; \hat{\Theta}_d)$ has the general form of a subgroup discovery measure of Eq. (2.20), $\texttt{COMP}(n_s, \#param)$ is the complexity associated with each target probability distribution (normal or categorical), and $\Delta_\beta L(M \oplus s)$ the added model complexity of adding s.

Interpretation of hyperparameter β. The hyperparameter β represents a tradeoff between finding many subgroups that cover few instances or few subgroups that cover many instances[2]. In the general form of a subgroup quality measure of Eq. (2.20), β is just given by $\beta = 1 - \alpha$. We empirically show later that the *normalized gain* ($\beta = 1$) usually achieves a better MDL score than other β values; this was already known for other measures from rule learning theory [39]. Nonetheless, the main objective of subgroup discovery is to *locally* describe regions in the data that strongly deviate from a certain target. Thus, it is up to the user to specify what one is looking for in the data: either a more granular and detailed perspective (β close to one) or a more general and high-level one (β close to zero). Note that, for comparison to other algorithms *we will always use the normalized gain* ($\beta = 1$) except when explicitly stated.

5.2.3 Statistical testing interpretation of compression gain

The gain of Eq. (5.2.2) shares the same expression of the weighted Kullback Leibler divergence that was shown in Sections 3.4.3 and 3.5.3 to be equivalent to a Bayesian one-sample proportions/multinominal test and t-test, respectively. Thus, it too guarantees individual "significance" for each subgroup according to these tests. We will now look at this in more detail.

A Bayesian factor is an alternative to frequentist statistical testing and is given by the likelihood of both hypotheses generating the data [61]:

$$\log K = \log \frac{\Pr(D \mid M_1)}{\Pr(D \mid M_2)},$$

where M_1 and M_2 are two models that we are comparing. Values of $\log K$ above zero tell us that there is more evidence in favour of model M_1, while negative values tell us

[2]For details on the empirical analysis of different β values please refer to Appendix G.2

the opposite [61]. If we look back at the expression of the greedy gain in Eq. (5.1) for a general model M (instead of SL) and convert the encoding $L(\cdots)$ to probabilities $\Pr(\cdots)$ using the Shannon-Fano code: $L(A) = -\log \Pr(A)$ [114]; we can see that it takes the same form plus some extra terms:

$$\Delta_\beta L(M \oplus s) = \log\left(\frac{\Pr(\mathbf{Y} \mid \mathbf{X}, M \oplus s)}{\Pr(\mathbf{Y} \mid \mathbf{X}, M)} \frac{\Pr(M \oplus s)}{\Pr(M)}\right) \frac{1}{(n_s)^\beta} = \frac{\log\left(K \cdot K_M\right)}{(n_s)^\beta},$$

where $K_M = \Pr(M \oplus s)/\Pr(M)$ represents the division of the model's likelihood (called a prior in Bayesian statistics). Thus, we obtain an expression with three terms: the first, K, gives us an MDL equivalent to a Bayesian factor that weighs how likely the data is given each model (M or $M \oplus s$); the second, K_M, gives the likelihood of each model; and the third is a normalizing term to be able to compare the contribution of different subgroups given how much data they cover.

The first conclusion that we can draw from this is that the subgroup that maximizes the compression gain is the one that *locally* maximizes this statistical test, i.e., it is the *mode* of this distribution. In the specific case of subgroup lists, the factor term K of the compression gain corresponds to a proportion/multinomial or t-test depending on the type of target variable. Second, the term K_M can be seen as a multiple hypothesis testing correction, as the way in which $L(M)$ was developed puts more weight on model structures that can generate more variants. Also, it should be noted that the encoding of $L(M)$ is more subjective than $L(D \mid M)$, but it will be an upper bound on the perfect encoding for M, and can be taken as a more 'conservative' test. Third, if the compression gain is positive for a subgroup, it means that there is more evidence in favor of adding that subgroup than not. Fourth, the normalizing term allows us to adjust the weight that is given to the data covered by each subgroup.

In summary, we can say that the greedy gain based on the compression gain, a common heuristic for MDL in pattern mining, is maximizing the test statistic of a hypothesis test and only adds that subgroup for which most evidence is available.

5.2.4 Beam search for subgroup generation

The *beam search algorithm* for subgroup generation is shown in Algorithm 5.1. It starts by discretizing all variables depending on their subsets, i.e., nominal with the operator *equal to* ($=$) and numeric by generating all subsets with n_{cut} points. At each iteration, the w_b subgroups that maximize the selected gain (Eq. (5.1)) are chosen and will be expanded with all discretized variables until the maximum depth d_{max} of the description is achieved.

The algorithm accepts as inputs the dataset $D = (\mathbf{X}, \mathbf{Y})$, the number of cut points n_{cut} used for equal frequency binning of numeric variables, the beam width w_b, the max-

imum depth of search or number of variables in a subgroup description d_{max}, and the indexes of the data covered by the subgroups present in the subgroup list, $coverage_S$. The algorithm is initialized by filling the $beam$ and $subgroup$ with an empty subgroup of size zero (Ln 2 and Ln 3, respectively). The algorithm is composed of three nested loops. In short, the first (outer) loop goes over each depth of subgroups generated, the second loop goes over each candidate to extend for a fixed depth, and the third (interior) loop goes over each item used to extend the candidates. Now we will go into more detail over each loop.

In the *first loop*, the depth is increased by one (Ln 6), $candidates$ is initialized with the $patterns$ of the $beam$ from the previous iteration (Ln 7), and after that, all $patterns$ are removed from the $beam$ (Ln 8). The *second loop* iterates over all $candidates$ (Ln 9) and expands each of them in the third loop with all the $items$ generated from the explanatory variables $\mathbf{X}$ (Ln 11). An $item$ is a subgroup of size one that can be generated by logical conditions on one variable $X_j \in \mathbf{X}$. If variable X_j is `nominal`, each item is a condition given by the equality operator ($=$) on each category, e.g., $feathers = yes$ from Figure 2.4. If the variable is `numeric`, equal frequency binning with open and closed intervals is used to generate all possible items (further explained at the end of this paragraph). Expanding a candidate $cand$ to generate a $subgroup_new$ (Ln 15) requires computing three properties: 1) its coverage of the data through a bitwise AND (Ln 12); 2) its description (Ln 15); and 3) its statistics Θ_{new} (Ln 15). Its score is computed according to Eq. (5.1) (Ln 16). Then if the score is higher than the pattern with a minimum score in the $beam$, the latter is replaced by the higher-scoring one. Finally, if the score is higher than the score of $subgroup$, this is replaced. The algorithm terminates when the maximum search depth of the subgroups is reached and $subgroup$ is returned, to be added to the subgroup list (Ln 21).

Numeric discretization. Suppose a numeric variable X_j, and a number of cut points n_{cut}. The $items$ generated from this numeric variable are all valid subsets (they must cover at least one instance) given by equal frequency discretization with open and closed intervals for n_{cut} cut points. Open intervals require one operator ($\geq$ or $\leq$), while closed intervals require two ($\geq$ and $\leq$). As an example, in the case of a generic variable X_j and $n_{cut} = 2$, with $cut_point_1 = 10$ and $cut_point_2 = 20$ it generates four $items$ with one operator, i.e., $items_{1op} = \{\ x_j \geq 10,\ x_j \leq 10, x_j \geq 20,\ x_j \leq 20\}$, and one $item$ with two operators, i.e., $items_{2op} = \{10 \leq x_j \leq 20\}$.

Algorithm 5.1 Beam search for subgroup generation

Input: Dataset D, number of cut points n_{cut}, beam width w_b, depth max. d_{max}, and data already covered by other subgroups in SL $coverage_S$.

Output: $subgroup$

1: $(\mathbf{X}, \mathbf{Y}) \leftarrow D$
2: $beam \leftarrow [\varnothing]$
3: $subgroup \leftarrow \varnothing$
4: $d \leftarrow 1$
5: **while** $d \leq d_{max}$ **do**
6: $\quad d \leftarrow d + 1$
7: $\quad candidates \leftarrow beam$
8: $\quad beam \leftarrow empty_list(size = w_b)$
9: $\quad$ **for** $(cand, coverage_cand) \in candidates$ **do**
10: $\quad\quad coverage_cand \leftarrow coverage_pattern$ & $coverage_S$
11: $\quad\quad$ **for** $(item, bitset_item) \in items(\mathbf{X})$ **do**
12: $\quad\quad\quad coverage_new \leftarrow coverage_item$ & $coverage_cand$
13: $\quad\quad\quad cand_new \leftarrow cand \oplus item$
14: $\quad\quad\quad \Theta_{new} \leftarrow statistics(\mathbf{Y}, coverage_new)$
15: $\quad\quad\quad subgroup_new \leftarrow (cand_new, \Theta_{new})$
16: $\quad\quad\quad score \leftarrow \Delta_\beta L(D, SL \oplus subgroup_new)$
17: $\quad\quad\quad$ **if** $score > min_score(beam)$ **then**
18: $\quad\quad\quad\quad beam \leftarrow replace(beam, subgroup_new, min_score)$
19: $\quad\quad\quad$ **if** $score > \Delta_\beta L(D, SL \oplus subgroup)$ **then**
20: $\quad\quad\quad\quad subgroup \leftarrow replace(subgroup, subgroup_new)$
21: **return** $subgroup$

5.2.5 The Robust Subgroup Discoverer algorithm

Algorithm 5.2 presents RSD[3], for *Robust Subgroup Discoverer*, a greedy algorithm that starts with an empty subgroup list and iteratively adds subgroups until no more compression can be gained, where compression is measured in terms of compression gain (Eq. 5.1) of adding a subgroup s.

The algorithm starts by taking as input a dataset D and the beam search parameters, namely the number of cut points n_{cut}, the width of the beam w_b, and the maximum depth of search d_{max}. It initializes the predictive rule list with the default rule, based on the dataset empirical distribution (Ln 1). Then, while the beam search algorithm

[3]Our implementation uses the rulelist package (`https://pypi.org/project/rulelist/`) and can be found on GitHub: `https://github.com/HMProenca/RuleList`

returns subgroups that improve compression (Ln 3), it keeps iterating over two steps: 1) finding the best subgroup from all candidates generated in the beam search (Ln 4); and 2) adding that subgroup to the end of the model, i.e., after all the existing subgroups in the model (Ln 5). The beam search returns the best subgroup on the data not covered by any subgroup already in model M. When no subgroup improves compression (non-positive gain) the while loop stops and the subgroup list is returned. Note that beam search is used at each iteration, instead of only once at the beginning, as it can converge to local optima, and running the candidate search once would thus bias our search to the top-k subgroups instead of the best at each iteration.

Algorithm 5.2 RSD algorithm

Input: Dataset D, number of cut points n_{cut}, beam width w_b, depth max. d_{max} and normalization β

Output: Subgroup list S

1: $M \leftarrow [\Theta_d(Y)]$
2: $subgroup \leftarrow BeamSearch(SL, D, w_b, n_{cut}, d_{max})$
3: **while** $\Delta_\beta L(D, SL \oplus subgroup) > 0$ **do**
4: $\quad subgroup \leftarrow BeamSearch(SL, D, w_b, n_{cut}, d_{max})$
5: $\quad SL \leftarrow SL \oplus subgroup$
6: **return** $S \in SL$

5.2.6 Time and space complexity

In this section we analyze the time and space complexity of RSD as given in Algorithm 5.2. The algorithm can be divided in three parts: 1) preprocessing of the data; 2) the Separate and Conquer (SaC) algorithm; and 3) the beam search. Note that depending on the type of target we have different complexities as each statistic requires different computations.

1) Preprocessing phase. In the preprocessing phase all the coverage bitsets of the items are generated, i.e., the indexes of the instances covered by each item generated from numerical and nominal variables. The set of all items is ζ and its size is given by $|\zeta|$. Thus, we go over the data a maximum of $|\zeta|$ times, obtaining a time complexity of $\mathcal{O}(|\zeta|n)$, and the results are stored in a dictionary for $\mathcal{O}(1)$ access. Also, there are some constants that are cached for a fixed amount the first time they are computed, such as the universal code of integers $L_{\mathbb{N}}(i)$, and $\Gamma(i)$ for the numeric target case, and $\mathcal{C}(i)$ in the categorical case.

2) SaC phase. For the SaC phase, it is clear that the algorithm runs the beam search $|S|$ times, and will thus multiply the time complexity of the beam search by $|S|$.

3) Beam search phase. For the last $d_{max}-1$ iterations of the loop, each of w_b candidates in the beam is refined with all $|\zeta|$ items, which gives a time complexity by itself of $\mathcal{O}(d_{max}w_b|\zeta|)$. Then, for each refinement, the algorithm computes its coverage, statistics and score, where the last two depend on the number and type of target.
The *coverage* of the refinement is the logical conjunction of two bitsets, i.e., the bitset of the candidate b_{cand} and that of the item b_{item}. The computation of this new coverage has a time complexity of $\mathcal{O}(|b_{cand}| + |b_{item}|)$, which in a worst-case equals a run over the dataset $\mathcal{O}(n+n) = \mathcal{O}(n)$. Thus the time complexity of the algorithm is given by

$$\mathcal{O}\left(|S|d_{max}w_b|\zeta|stats\right),$$

where $stats$ is the time complexity associated with computing the statistics for one candidate. Now, we will analyse the specific $stats$ complexity depending on the type of target.

Nominal target variables. The *statistics for categorical distributions* require the computation of the usage for each class for each target of each subgroup rule and the new default rule. Assuming a maximum number of classes k (for all target variables) and t target variables, then the worst case for the coverage gives $\mathcal{O}(tnk)$ from which the likelihood can be directly computed.
The *nominal score* requires the computation of the data and model encoding, from which the data encoding dominates. The data encoding entails the computation of the NML complexity and likelihood for each refinement. In general, the values of the NML complexity are just computed once and then cached, thus in a worst-case where one requires to compute n values for $\mathcal{C}(n_i), \forall_{n_i=1,\dots,n}$. Using the approximation of Mononen and Myllymäki [92] for its computation, with $\mathcal{O}(\sqrt{10n_i}+k)$, gives a worst-case complexity of $\mathcal{O}(tn(\sqrt{n}+k))$. This does not depend on the parameters of the beam, as the lookup of these values is $\mathcal{O}(1)$. The likelihood in general dominates over this term as it is computed for each refinement.
Thus the total time complexity for **nominal targets** is given by:

$$\mathcal{O}\left(|S|d_{max}w_b|\zeta|tnk + tn(\sqrt{n}+k)\right)$$

Numeric target variables. The *statistics for normal distributions* require the computation of the mean and variance (or residual sum of squares) for the refined subgroup

and for the default rule. The mean can be computed in $\mathcal{O}(n)$ and given this the variance can also be computed in $\mathcal{O}(n)$. Thus, for all the targets one obtains $\mathcal{O}(tn)$.
The *numeric score* requires the computation of the data and model encoding, from which the data encoding dominates. The data encoding entails the computation of the gamma function and the direct use of the statistics. Similar to the NML complexity, we compute the values of the gamma function as needed and cache them afterward. In general, the computation of the gamma function is dominated by the other terms as we only compute it at most n times.
Thus the total time complexity for **numeric targets** is given by:

$$\mathcal{O}\left(|S|d_{max}w_b|\zeta|tn\right).$$

Notice that this represents a worst case scenario and that in practice the direct use of bitsets for the computation of the class usages in the nominal case makes it faster than its numeric counterpart for the same dataset size.

Space Complexity. The main memory consumption resources of the algorithm are: 1) the storage of items ζ; 2) the beam; and 3) the cached constants. The item storage requires at most the storage of $|\zeta|$ bitsets, with each bitset taking $\mathcal{O}(n)$, thus it totals $\mathcal{O}(|\zeta|n)$. The beam saves w_b bitsets at a time, thus having a space complexity of $\mathcal{O}(w_b n)$. The cached values make up a total of n values being dominated by the items or beam part. Thus, depending on which part dominates, the space complexity of the algorithm is

$$\mathcal{O}(w_b n + |\zeta|n).$$

5.3 Empirical evaluation

In this section, we will empirically validate our proposed problem formulation and the RSD[4] algorithm. To do this, we will test how varying the hyperparameters of RSD affects the subgroups found, and then we will compare RSD against state-of-the-art algorithms in subgroup set discovery[5].

This section is divided as follows. In Section 5.3.1 we evaluate the effect of changing the different hyperparameters of RSD. Then, in Section 5.3.2 we present the setup for validating our approach, based on algorithms compared against, and datasets and measures used to evaluate them. After that, in Section 5.3.3, the results for univariate

[4]Our implementation uses the rulelist package (`https://pypi.org/project/rulelist/`) and can be found on GitHub: `https://github.com/HMProenca/RuleList`

[5]For replication of the experiments in this chapter please refer to: `https://github.com/HMProenca/RobustSubgroupDiscovery`.

and multivariate nominal targets are presented. Then, in Section 5.3.4 the results for univariate and multivariate numeric targets are shown. Finally, in Section 5.3.5 the runtimes of the algorithms are compared.

5.3.1 Influence of RSD hyperparameters

Here we study the effect of RSD hyperparameters on the discovered subgroup lists. To not overfit our hyperparameters to the datasets and for this reason obtain a better performance than other methods, the values of RSD hyperparameters for the remaining experiments (after this section) are fixed at the standard values of the DSSD implementation for the beam search, i.e., beam width $w_b = 100$, number of cut points $n_{cut} = 5$, and maximum search depth $d_{max} = 5$, and to the compression gain normalization term $\beta = 1$ (normalized gain). These values are assumed to be enough to achieve convergence and to obtain good subgroup lists and are thus taken as the *standard values* of RSD.
Now, to evaluate hyperparameter influence, we vary one hyperparameter value at a time while others remain fixed at their *standard values*. The results of varying the compression gain normalization hyperparameter β can be seen in Appendix G.2; the results of varying the beam search hyperparameters w_b, n_{cut}, and d_{max} can be found in Appendix G.3.

Normalization term β. The results are evaluated in terms of compression ratio, SWKL (presented in Section 3.6), and the number of rules. For compression gain, the results (as shown in Appendix G.2) are similar for a small number of samples but $\beta = 1$ and 0.5 obtain better results for larger datasets. In terms of SWKL, normalized gain ($\beta = 1$) is better. On the other hand, in terms of the number of rules $\beta = 1$ can obtain one order of magnitude more rules than the others, especially for larger datasets.

Beam search hyperparameters w_b, d_{max}, and n_{cut}. The results are evaluated in terms of compression ratio and the average number of conditions per subgroup (for d_{max}). In general, increasing any of the three values result in better models according to relative compression. It is also interesting to note that for maximum depths above 5 it is rare to have an average number of conditions above 4, backing up our decision for the standard value $d_{max} = 5$.

5.3.2 Setup of the subgroup quality performance comparisons

In this section we evaluate the quality of our proposed method by comparing it to the state-of-the-art approaches in subgroup set discovery, which may vary depend-

ing on the type of target variable(s). The comparison takes three dimensions: 1) the *algorithms* used to compare against; 2) *measures* used to evaluate the quality of the subgroups found by each algorithm; 3) the *datasets* in which the algorithms are evaluated. We now discuss the details of each dimension.

1) Algorithms. The algorithms we compared to and their relevant characteristic are listed in Table 5.2. A short description of each is as follows:

1. top-k[6] - standard subgroup discovery miner used as a benchmark.
2. seq-cover[6] - sequential covering as implemented in the DSSD implementation.
3. CN2-SD[7] - the classical sequential covering subgroup discovery algorithm, which is only implemented for nominal targets, and only removes the examples of the class of interest already covered (not all examples covered, as seq-cover does).
4. Diverse Subgroup Set Discovery (DSSD)[6] - diverse beam search for diverse sets of subgroups [75].
5. Monte Carlo Tree Search for Data Mining (MCTS4DM) - an approach to improve on beam search to find better subgroups without getting stuck in local optima [14].
6. FSSD - a sequential approach for subgroup set discovery that defines a set as a disjunction of subgroups [10].

As can be seen in Table 5.2 most algorithms can only be applied to single-target binary problems, and besides RSD only top-k, seq-cover and CN2-SD support the use of Sum of Weighted Kullback-Leibler (SWKL) divergence to measure the quality of the found subgroup set. Thus we only compare against seq-cover and CN2-SD, algorithms that output a subgroup list and can be applied to many target types, and with top-k as a reference of a non-diverse subgroup discovery algorithm. The algorithms that output sets do not have a stopping criterion or global formulation, and underperform in terms of SWKL, thus those comparisons are relegated to Appendix G.4. As an example, DSSD can indeed be applied to all types of target variables, but the fact that it uses weighted sequential covering makes it unsuitable to use the SWKL, making it unfairly underperform and inappropriate for a fair comparison (as shown in the Appendix). Also, note that we do not compare with machine learning algorithms that generate predictive rules for classification or regression, such as RIPPER or CART, as the rules

[6]top-k, seq-cover, and DSSD are available in the implementation of the DSSD algorithm `http://www.patternsthatmatter.org/software.php#dssd/`

[7]Available in the Orange data mining toolkit `https://orangedatamining.com/`

Table 5.2: Algorithms included in the comparison and their functionalities. *Quality* represents the quality measure used to evaluate one single subgroup, *search* is the type of search algorithm supported, *swkl* shows if it supports SWKL to measure the quality of a subgroup set, *output* tells if the subgroups discovered form a list or a set, and '✓' and '–' represent if that type of target variable(s) is supported. MCTS stands for Monte Carlo Tree Search. * Most algorithms only support WKL_μ for numeric targets (Eq. (2.24)), i.e., a Weighted Kullback-Leibler divergency that only takes into account the mean, contrary to the one used by RSD that also uses the variance (Eq. (2.25)). For the nominal target case there is only one WKL (the different WKL measures are explained in Section 2.6.2).

						nominal		numeric	
Algorithm	quality	search	output	swkl	bin.	nom.	multi	single	multi
RSD	WKL	beam	list	✓	✓	✓	✓	✓	✓
top-k	WKL^*_μ	beam	set	✓	✓	✓	✓	✓	✓
seq-cover	WKL^*_μ	beam	list	✓	✓	✓	✓	✓	✓
CN2-SD	entropy	beam	list	✓	✓	✓	-	-	-
DSSD	WKL^*_μ	beam	set	-	✓	✓	✓	✓	✓
MCTS4DM	WKL^*_μ	MCTS	set	-	✓	-	-	-	-
FSSD	$WRAcc$	DFS	list	✓	✓	-	-	-	-

generated aim at making the best prediction possible, and not the highest difference from the dataset distribution, as shown theoretically in Section 3.7.

Quality measures. As the quality of a set is measured using the SWKL, the most appropriate measure to use is the Weighted Kullback-Leibler (WKL) for the algorithms that support it. CN2-SD supports entropy which is related to WKL. FSSD only supports WRAcc at the moment. Note that for the case of numeric targets, except RSD, all use a WKL that only takes into account the mean, given by $WKL_\mu(s) = n_s/\hat{\sigma}_d(\hat{\mu_d} - \hat{\mu_s})^2$, in contrast to the deviation-aware measure of RSD in Eq. 3.25.

Hyperparameters. Most algorithms use beam search, thus only have three main hyperparameters: the maximum depth of search d_{max}; the width of the beam w_b; and the number of cut points to discretize numeric explanatory variables n_{cut}. The larger the values the better the performance but the slower the algorithms become, as time complexity is linear to each of them. To be fair and not over-search the hyperparameters, we selected the default values of the DSSD and seq-cover implementation for all beam-search algorithms: $d_{max} = 5$, $w_b = 100$, $n_{cut} = 5$. For the case of MCTS4DM, which requires a larger set of hyperparameters, only the number of iterations is set, $n_{iter} = 50\,000$, to ensure good convergence, and the rest were set as default. FSSD only requires the maximum depth, which was set at 5.

2) Measures. To ascertain the quality of the subgroup sets we use three different measures. The first is our proposal to measure the overall quality of an ordered set of subgroups, the Sum of Weighted Kullback-Leibler (SWKL), as defined in Eq. (3.26). The other two are the number of subgroups $|S|$ and the average number of conditions per subgroup $|a|$, two commonly used measures for the interpretability/complexity of a set of rules. These two measures follow the law of parsimony and assume that fewer subgroups with fewer conditions are easier to understand by humans, which can be an invalid assumption in some situations. Nonetheless, it is widely used and its simple understanding typically makes for a good proxy [27]. In machine learning, algorithms are tested on their generalization to unseen data, which is achieved by multiple runs using different test sets (e.g., cross-validation). Even though this could be of interest, subgroup discovery is always evaluated on the same dataset, as the goal is to describe the current dataset well. For this reason, and for the fact that existing implementations are not prepared to use a test set, we follow the standard approach in subgroup discovery of only testing on the current dataset.
3) Datasets. For a thorough analysis we use a total of 54 datasets—10-univariate binary; 10 univariate nominal; 9 multivariate nominal; 15 univariate numeric; and 9 multivariate numeric—that are listed in Tables G.1 and G.2 of Appendix G.1. The datasets are commonly used benchmarks of machine learning and subgroup discovery, which are publicly available from the UCI[8], Keel[9], and MULAN[10] repositories. The datasets were selected to be the most varied possible. In the case of the nominal target datasets in Table G.1, the number of targets range from 1 to 374, the classes from 2 to 28, the samples from 150 to $45\,222$, and the variables from 3 to $1\,186$. In the case of the numeric target datasets in Table G.2, the number of targets range from 1 to 16, the samples from 154 to $22\,784$. Note that we used multi-label datasets instead of multi-nominal as the latter are not widely available.

5.3.3 Nominal target results

The results obtained on binary, nominal, and multi-label datasets with sequential subgroup set miners can be seen in Table 5.3, while the results for algorithms that output sets can be found in Table G.3 in Appendix G.4. We can see that overall our algorithm gets 15 out of 29 best results, compared with seq-cover in second place with 13 best results. In terms of SWKL and per type of data, RSD achieves the smallest ranking for binary, seq-cover for nominal, and both are tied for multi-nominal. This small

[8] `https://archive.ics.uci.edu/ml/`
[9] `http://www.keel.es/`
[10] `http://mulan.sourceforge.net/datasets.html`

difference in the results between RSD and seq-cover is important for two reasons. First, it validates SWKL, as it shows that seq-cover is already implicitly maximizing it without knowing it. Second, it shows that RSD can obtain on par or slightly better results than other established approaches. Our non-diverse baseline, top-k, shows that covering different regions of the dataset is important to maximize SWKL.

Regarding the number of found subgroups we can see that in most cases, all algorithms are in the same order of magnitude, with some clear exceptions where RSD obtains many more subgroups (for *adult*, *nursery*, *kr-vs-k*, and *mediamill*). These results can be explained by the use of normalized gain ($\beta = 1$) by RSD, together with the fact that these datasets have a large number of samples, few variables, and/or a large number of categories. First, let us recall that the normalized compression gain of Eq. (5.1) is composed of a data covering part and a model penalization part and that both are normalized by the number of instances covered, which gives an advantage to subgroups that cover less data but are well-covered (only one category, or few categories). When the datasets are larger and the number of variables is reasonably small, like *adult* with $45\,222$ examples and 14 variables, there is a larger chance of finding more statistically "significant" subgroups, as there can be more regions where subgroups only (or almost only) cover one class, and the penalization of the model encoding is small as there are not many variables. On the other hand, subgroups that cover more data can more easily have a larger entropy in the class label distribution. For example, *kr-vs-k*, which is a reasonably large dataset with $28\,056$ and with 18 class labels, a subgroup that only covers one class label, as opposed to covering many class labels, will have a higher chance of being chosen. The number of subgroups found can be large, but it was shown in a classification setting that they generalize well [96]. It is interesting to note that in the case of *corel-5k*, RSD does not find any "significant" subgroup to add.

Regarding the number of conditions per subgroup, the two best-performing algorithms in terms of SWKL, RSD, and seq-cover, tend to have a similar and lower number of conditions than the other algorithms. Top-k, only covering the same region, has a tendency to be close to the maximum depth of 5.

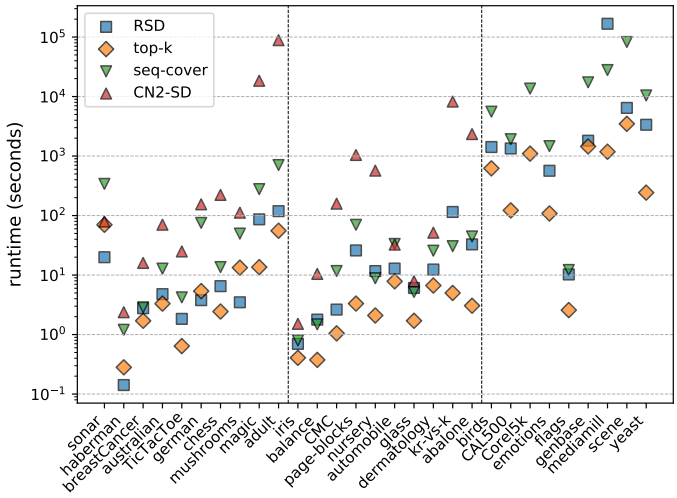
5.3.4 Numeric target results

The results for the single-target and multi-target numeric datasets can be seen in Table 5.4. In general, it can be seen that RSD obtains the best results for 23 out of 25 datasets. This is to be expected as SWKL and RSD take into account the dispersion/deviation of the subgroup target while top-k and seq-cover do not. This is clearly supported by the normalized standard deviation of the first subgroup found, where RSD tends to find subgroups with smaller deviations for 10 out of 15 cases.

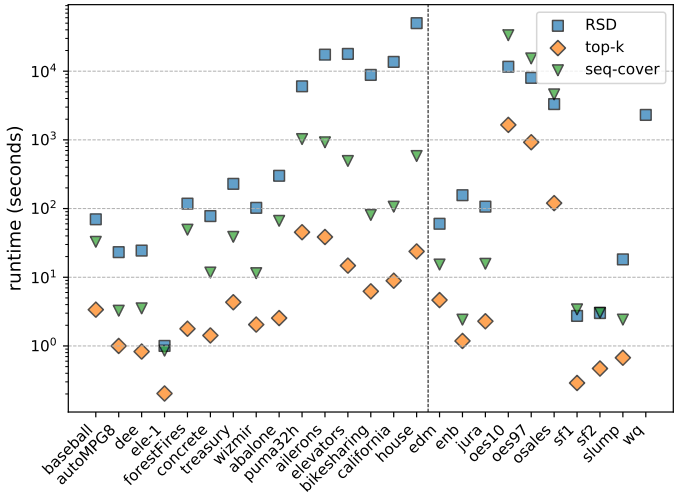
Comparing SWKL results for top-k with seq-cover and RSD shows that irrespective of dispersion-aware (RSD) or not (seq-cover), covering different regions of the data increases the quality of the list in terms of SWKL, validating the use of our measure. It should be noted that both top-k and seq-cover could in practice support taking into account the deviation but that would require several non-trivial modifications in their source code.
Regarding the number of subgroups, seq-cover tends to have more rules than RSD for datasets with less than $5\,000$ examples, while RSD tends to have more for a larger number of examples. This makes sense as there is more evidence to identify possible significant subgroups.
Regarding the number of antecedents, RSD tends to have, on average, one condition fewer than seq-cover for single-target and a similar number for the multi-target case.

5.3.5 Runtime comparison

Runtimes of all algorithms compared, i.e., top-k, seq-cover, CN2-SD, and RSD are shown in Figures 5.2a and 5.2b. In general, it can be seen that the runtime increases with the number of samples in the dataset for a fixed data type. For the nominal datasets, it seems that there is an increase in runtime with the number of target variables, which does not seem to happen for numeric targets. This is because for multivariate numeric targets the number of subgroups found was, in general, smaller.
Comparing the algorithms against each other, as expected, top-k was the fastest algorithm, as it only needs to search for the subgroups once, while the others need multiple iterations.
For nominal targets, CN2-SD was the slowest algorithm, which stems from the use of entropy as a quality measure—experiments with WRAcc proved orders of magnitude faster. RSD seems to perform on par with seq-cover and is often even slightly faster.
For numeric targets, RSD was one order of magnitude slower than seq-cover. One possible reason is the extra time to compute the variance, although this does not explain the difference between both algorithms. It seems that a further study of the numeric implementation could make for an interesting research direction.



(a) Nominal targets.



(b) Numeric targets.

Figure 5.2: Runtime in seconds for all algorithms for each dataset. The black vertical line divides the type of datasets, i.e., from left to right: univariate binary, nominal, and multi-label for nominal targets, and univariate and multivariate for numeric.

Table 5.3: Nominal target results. This includes single-binary, single-nominal, and multi-label, separated by horizontal lines in the table (top to bottom). The properties of the datasets can be seen in Table G.1, and are ordered in ascending number of: 1) target variables; 2) number of classes; and 3) number of samples. The evaluation measures are {quality of the subgroup set swkl; number of subgroups $|S|$; and average number of conditions $|a|$}. 'avg. rank' stands for the average ranking for the respective target variable type, where 1 represents the best rank. Note that CN2-SD does not work for multi-label case and thus the empty values $-$. *as RSD produced no subgroups for corel-5k, seq-cover number of subgroups was used as a reference.

	top-k			seq-cover			CN2-SD			RSD		
datasets	swkl	$\|S\|$	$\|a\|$	swkl	$\|S\|$	$\|a\|$	swkl	$\|S\|$	$\|a\|$	swkl	$\|S\|$	$\|a\|$
sonar	0.24	2	4	**0.96**	9	2	0.67	11	2	0.43	2	3
haberman	0.08	1	5	**0.39**	20	4	0.18	12	4	0.04	1	1
breastCancer	0.37	6	2	0.80	13	2	0.80	11	2	**0.82**	6	2
australian	0.26	5	3	**0.69**	13	3	0.54	24	3	0.55	5	2
tictactoe	0.50	16	3	0.73	18	3	0.21	21	3	**0.87**	16	2
german	0.08	4	5	0.30	22	4	**0.42**	48	4	0.14	4	3
chess	0.25	17	3	0.87	13	2	0.68	51	3	**0.97**	17	2
mushrooms	0.49	12	4	0.92	11	1	**1.00**	36	1	**1.00**	12	1
magic	0.16	69	5	0.38	35	4	0.42	616	3	**0.47**	69	4
adult	0.11	103	5	0.27	79	4	0.43	1230	4	**0.31**	103	4
avg. rank	3.8	1.9	3.8	2.1	2.4	2.2	2.2	3.8	2.5	**1.9**	1.9	1.5
iris	0.53	4	2	**1.45**	5	2	0.96	4	2	1.44	4	1
balance	0.21	9	3	**0.80**	19	3	0.18	3	3	0.69	9	3
CMC	0.07	7	3	**0.30**	38	4	0.27	42	3	0.25	7	2
page-blocks	0.19	21	5	0.45	26	2	0.44	12	4	**0.49**	21	3
nursery	0.92	81	2	1.36	22	3	0.87	8	4	**1.63**	81	3
automobile	0.38	5	4	**1.61**	11	3	**1.54**	7	4	1.25	5	2
glass	1.01	5	2	1.55	5	2	**2.14**	6	2	1.92	5	1
dermatology	0.54	9	2	**2.28**	9	2	2.12	7	3	2.11	9	2
kr-vs-k	0.45	351	5	0.75	43	4	0.20	61	5	**1.83**	351	3
abalone	0.26	16	5	0.62	29	4	0.60	49	3	**0.74**	16	2
avg. rank	3.7	2.4	3.0	**1.6**	3.0	2.2	2.8	2.3	3.4	1.9	2.4	1.4
emotions	0.71	17	5	1.93	22	4	–	–	–	**2.68**	17	3
scene	0.39	49	5	1.85	33	4	–	–	–	**3.05**	49	4
birds	0.49	8	5	**2.02**	20	4	–	–	–	1.57	8	3
flags	0.44	5	4	**2.40**	17	4	–	–	–	1.21	5	2
yeast	0.49	35	5	1.83	55	5	–	–	–	**2.20**	35	5
genbase	0.88	15	2	5.51	12	1	–	–	–	**5.82**	15	1
mediamill	0.43	131	5	1.44	60	5	–	–	–	**2.96**	131	5
CAL500	1.46	1	5	**16.91**	36	4	–	–	–	1.24	1	5
corel5k*	5.81	144	3	**5.39**	144	4	–	–	–	0.00	0	0
avg. rank	2.7	1.9	2.7	**1.7**	2.3	1.9	–			**1.7**	1.8	1.4

Table 5.4: Numeric target results. This includes single-numeric and multi-numeric, separated by a horizontal line in the table (top to bottom). The properties of the datasets can be seen in Table G.2, and are ordered in ascending number of: 1) target variables; 2) number of classes; and 3) number of samples. The evaluation measures are { quality of the subgroup set swkl; number of subgroups $|S|$; normalized standard deviation of the first subgroup $\tilde{\sigma}_{t1}$; and average number of conditions $|a|$}. 'avg. rank' stands for the average ranking for the respective target variable type, where 1 represents the best ranking. Note that $\tilde{\sigma}_{t1}$ is not shown for the multi-numeric case as it is not easy to understand.

	top-k				seq-cover				RSD			
datasets	swkl	$\tilde{\sigma}_{t1}$	$\|S\|$	$\|a\|$	swkl	$\tilde{\sigma}_{t1}$	$\|S\|$	$\|a\|$	swkl	$\tilde{\sigma}_{t1}$	$\|S\|$	$\|a\|$
baseball	0.26	0.82	7	4	1.40	1.22	26	4	**1.86**	0.01	7	2
autoMPG8	0.43	0.54	8	4	1.45	1.85	22	4	**1.57**	0.18	8	2
dee	0.46	0.50	9	4	1.29	2.01	20	4	**1.35**	0.32	9	2
ele-1	0.29	1.06	8	4	1.14	0.94	22	4	**1.22**	1.24	8	2
forestFires	0.61	6.84	22	4	2.73	0.15	57	4	**3.91**	7.57	22	3
concrete	0.28	0.65	18	4	1.27	1.53	35	4	**1.31**	0.21	18	3
treasury	0.43	0.68	31	4	2.74	1.46	21	4	**3.85**	0.01	31	2
wizmir	0.70	0.31	22	4	2.15	3.22	26	4	**2.72**	0.15	22	2
abalone	0.23	0.59	26	4	0.47	1.68	126	5	**0.71**	1.32	26	3
puma32h	0.55	0.59	48	4	1.39	1.68	70	5	**1.44**	0.29	48	3
ailerons	0.24	1.23	98	4	1.04	0.82	105	4	**1.44**	0.98	98	4
elevators	0.23	1.44	158	4	0.83	0.69	150	5	**1.31**	1.40	158	4
bikesharing	0.26	1.09	136	4	1.24	0.92	91	4	**1.70**	0.02	136	4
california	0.19	0.90	174	4	0.69	1.11	116	5	**1.14**	0.00	174	4
house	0.19	1.59	269	4	0.91	0.63	143	5	**2.02**	2.83	269	5
avg. rank	3.0	2.1	1.8	2.0	2.0	2.3	2.3	2.7	**1.0**	1.6	1.8	1.3
edm	0.47	–	5	5	0.81	–	9	2	**1.88**	–	5	2
enb	2.73	–	41	2	3.54	–	19	2	**8.71**	–	41	2
slump	1.38	–	4	5	**2.74**	–	17	4	2.57	–	4	3
sf1	0.16	–	3	5	**2.06**	–	47	4	1.24	–	3	3
sf2	0.86	–	2	5	2.29	–	18	4	**0.91**	–	2	4
jura	0.47	–	15	5	2.38	–	28	4	**3.52**	–	15	3
osales	2.17	–	45	4	18.09	–	48	3	**26.44**	–	45	3
oes97	6.55	–	16	3	30.79	–	19	4	**34.36**	–	16	4
oes10	6.56	–	23	3	29.11	–	27	4	**40.65**	–	23	3
wq	0.87	–	62	5	2.06	–	47	4	**11.14**	–	62	4
avg. rank	3.0	–	1.7	2.4	1.7	–	2.6	1.8	**1.3**	–	1.7	1.8

5.4 Case Study: Hotel Bookings

To assess the usefulness of our method, we apply it to understand the type of clients that make a hotel booking based on how much time in advance (lead time in days) this was done. To this end, we used the "Hotel booking demand dataset" by Antonio et al. [6] and analysed the data of a *resort hotel* in the year of 2016. The first four subgroups of a total of 260 obtained with SSD++ can be seen in Figure 5.3 and its subgroups versus the dataset in Figure 5.4. Only the first 4 subgroups are shown here for clarity, and given that greedy search is used, they are also the 4 most interesting subgroups.

s	**description** of client bookings	n	$\hat{\mu}$	$\hat{\sigma}$	overlap
1	month = 9 & customer_type = Transient-Party & meal = Half Board & country = GBR & adults ≥ 2	22	533 days	34 days	–
2	month $\in [7, 9]$ & market_segment = Groups & weekend_nights = 1 & distribution_channel = Direct	29	336	~ 0	0%
3	month = 9 & week_nights =4 & distribution_channel = Corporate	16	343	3	0%
4	week_nights = 0 & deposit_type = Refundable & repeated_guest = no & adults≥ 2	20	9	~ 0	0%
dataset overall distribution		18 550*	92	99	–

Figure 5.3: First 4 subgroups of a subgroup list obtained by RSD with normalize gain ($\beta = 1$) on the *Hotel booking* dataset with target *lead days*—number of days in advance the bookings were done. *Description* contains information regarding client bookings, n the number of instances covered, $\hat{\mu}$ and $\hat{\sigma}$ are the mean and standard deviation in days, and *overlap* is the percentage of the subgroup description that is covered by subgroups that come before in the list, i.e., how independently can the subgroups be interpreted. The last line represents the dataset overall probability distribution. * The n of the dataset is the total number of instances in the dataset.

The results show us a very detailed picture of the dataset and at first glance, one notices that most subgroups cover a small number of instances. Nevertheless, this is normal as they represent highly defined subgroups, with a very different mean and an almost zero standard deviation, compared with the dataset $\hat{\mu}_d = 92$ and $\hat{\sigma}_d = 99$. As an example, subgroup 1 has an average lead time circa 6 times higher than the dataset distribution, together with a standard deviation that is 3 times smaller. This subgroup seems to represent a group of people that travelled together from Great Britain and all chose the same type of booking, while with some slight days of difference in their bookings. Another interesting subgroup is the 4^{th} which shows that there is a group

of around 20 similar bookings for groups of 2 or more adults done with only 9 days before arrival when the deposit type is refundable. If one would follow the whole subgroup list one would have a complete summary of the bookings done.

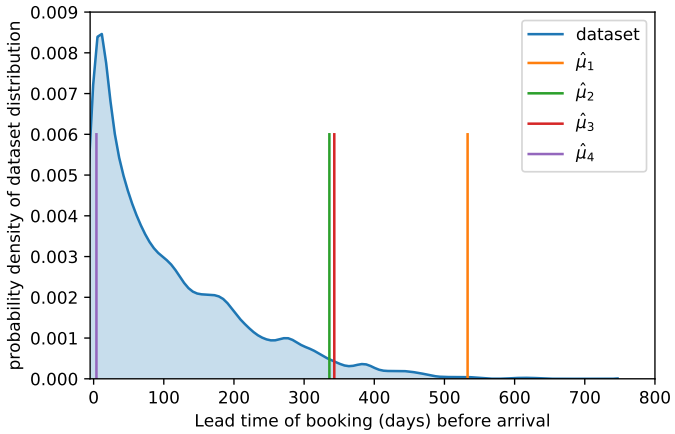


Figure 5.4: *Hotel bookings* kernel density estimation of the dataset distribution and location (not density) of the mean value of the first 4 subgroups obtained with RSD and normalized gain.

5.5 Case study: flight delay analysis

In this section, we apply RSD to the problem of describing flight delays, specifically how to identify subgroups of flights that can (or do not need to) be improved. This case study is part of the SAPPAO (a Systems APproach towards data mining and Prediction in Airlines Operations) project, which aims to combine the prediction of flight delays with the optimization of airplanes and crew scheduling. In our part of the project, we focus on finding sets of flights with an above-average delay, as it is impractical to optimize all flight schedules.

Typically, each airline needs to schedule their airplanes and crews 6 and 2 months in advance of the actual flight departure, respectively [98], and they would like to minimize the number of delayed flights scheduled. However, at the time of planning, the only variables available are the origin and destination airports and the proposed schedule. Thus, the variables highly correlated with delays, such as weather data, are not available. Nonetheless, as flight delays are additional expenses that airlines would like to reduce [94], they would still like to identify some of the characteristics of those flights. For the identified subgroups to impact, they should cover a sizeable amount

of their annual flights.

Dataset. Historical data of flights in the United States is freely available through the Bureau of Transportation Statistics (BTS) [1]. Specifically, we restrict our analysis to a single airline and a single year. The reason for this is that airlines have different strategies when choosing the scheduled time for their routes, and these can also change with a periodicity of six months [62]. Also, the results should reveal which schedules are associated with more delays so that an airline can improve, and using multiple airlines with different strategies can muddle these insights. The restriction to the busiest airports comes from the fact that they are more prone to chronic delays than smaller ones while also covering most airline flights. We selected the year 2017, a typical year of airline operation compared with 2007 to 2008 and 2020 to 2021 where the recession and the COVID-19 pandemic have affected flight operations [1]. After that, we selected one of the major US airlines with a high percentage of delayed flights: United Airlines (UA), with 16% of flights delayed [1].

After cancelled and diverged flights have been removed, the acquired data totals $577\,213$ samples. The variables used—with original names from the BTS—are: date; CRS (Computerized Reservation System) departure time; CRS arrival time; CRS elapsed time; origin airport; destination airport; and arrival delay. The date was transformed into six variables: the meteorological season, the respective month, day of the week, and weekday (or weekend).

Hyperparameters. The goal is to find the characteristics of a large number of delayed flights. Thus, there are two main ways to use RSD to find subgroups that cover large numbers of flights: 1) using the absolute-normalize trade-off hyperparameter; 2) using minimum support threshold. The first approach is flexible as the number of flights covered depends on the quality of the subgroups, while the second is rigid, as no subgroup with coverage below the threshold will be admitted. In this case, we choose to use a strict threshold to simulate an airline engineer looking for a precise number of flights to influence. Also, we combine the minimum support with normalized gain, as it will favour finding subgroups with coverage similar to the threshold. As the "sizeable" number of flights depends on the engineer, airline, and budget to make changes, we fixed it to 1%, a value we thought reasonable to demonstrate the abilities of RSD.

5.5.1 Analysis of subgroups obtained with RSD

The first four subgroups with normalized gain and a minimum support of 1% are shown in Figure 5.5 and their respective probability density functions (pdf) in Fig-

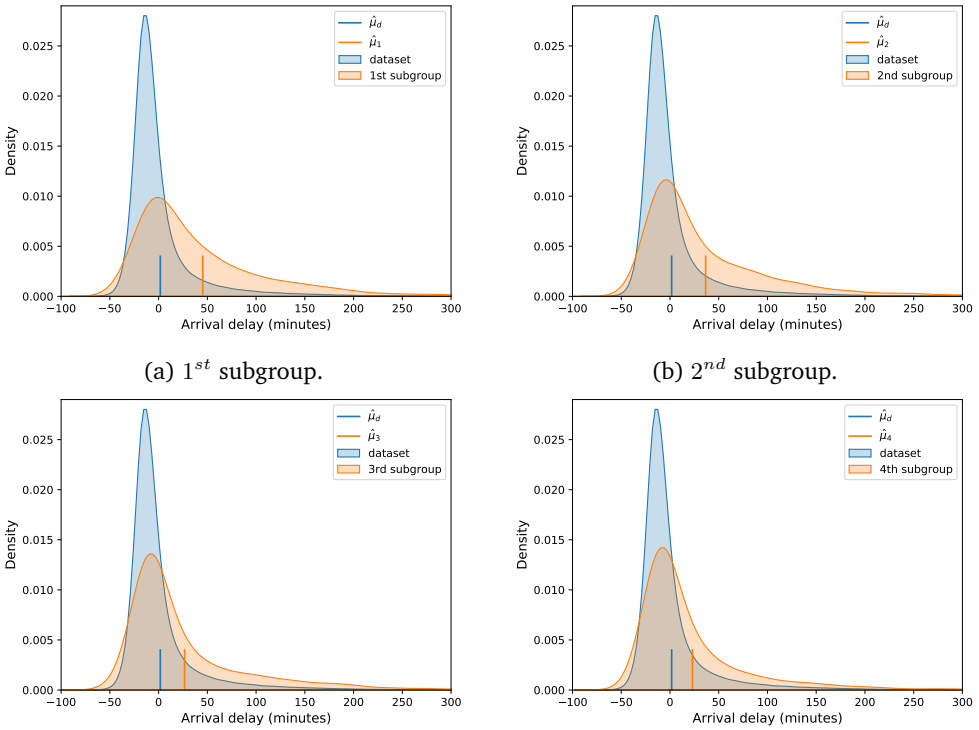
ures 5.6a, 5.6b, 5.6c, and 5.6d.

Interpretation of the results. As expected, the subgroups' pdfs follow a similar shape as the dataset distribution, which makes sense given that the variables used do not have a strong association. Also, the number of flights covered is around the value of our minimum support, which agrees with our use of *normalized* gain. Nonetheless, the four subgroups represent arrival delays above the dataset's average, which could mean chronic delays for those routes. We can see that the pdf of the first subgroup is the most different, and as we progress towards the least, their pdfs resemble more the dataset pdf. In terms of the descriptions, it is interesting to notice that all the four subgroups have either an origin or destination, months, and an arrival or departure time. Not surprisingly, this seems to point out that chronic delays are associated with certain airports at specific times of the year. Specifically, three subgroups contain Newark Liberty International (EWR) airport as origin or destination airport, making it an airport with a tendency for above-average delay. According to these subgroups, it would be interesting to investigate flights from or to this airport in the earlier months of the year.

Violation of the model assumptions. Figures 5.6a, 5.6b,5.6c, and 5.6d show that the pdfs are right-tailed distributions. Although the means do not describe the pdf completely, they still obtained what we were looking for, subgroups different from the dataset distribution. RSD users should be cautious of this and always compare the presented statistics with the subgroup's original distribution to avoid wrong conclusions.

s	**description** of a flight route	n	Arr. Delay
1	dest. = EWR & month $\in [March; Sep.[$ & dep. $\in [13:00; 19:00[$ & arrival $\geq 15:22$ & travel_time $< 03:36$	5259	45 ± 90 min.
2	origin = EWR & month $\in [May; Sep.[$ & weekday = yes & departure $\geq 16:03$	5047	37 ± 75
3	dest = SFO & month $\in [Jan.; May.[$ & departure $\geq 16:03$	5448	27 ± 76
4	dest = EWR & month $\in$ [Jan.; Sep.[& departure $\in [10:11; 13:00]$	5243	23 ± 77
	$\vdots$		
dataset distribution		556 215*	2 ± 47

Figure 5.5: *United Airlines (UA) arrival flight delay* analysis. The results were obtained by RSD with *normalized gain* ($\beta = 0$) and a minimum support of 5000. *UA* dataset contains one numeric target variable *arrival delay* in minutes. The dataset is made of all 2017 flights of United Airlines that were not cancelled , totalling 577 213 flights. *Description* contains information regarding flight routes and schedules, n the number of instances covered, and Arr. Delay the arrival delay in minutes. EWR $-$ Newark Liberty international airport; SFO $=$ San Francisco international airport airport. * The n of the dataset is the total number of instances in the dataset.



(a) 1^{st} subgroup.

(b) 2^{nd} subgroup.

(c) 3^{rd} subgroup.

(d) 4^{th} subgroup.

Figure 5.6: Probability density plot of the United Airlines arrival delay for the whole dataset and for the first four subgroups in the subgroup list. The subgroups were obtained with RSD *normalized* gain and a minimum support of $5\ 000$. The vertical lines represent the mean of the dataset and subgroup arrival delays.

5.6 Case study: socioeconomic background and university performance

In this section we apply RSD to a real use case to assess its usefulness and limitations. To this end, we aim at understanding how socioeconomic factors affect the grades of engineering university students in Colombia on their national exams. The dataset used to study this is fully described by Delahoz-Dominguez et al. [24]. It contains socioeconomic variables and grades in national exams done at high school and university level for engineering students in Colombia. For this specific case study, we have selected two of their exam grades at the university for two reasons. First, the relationship between socioeconomic variables and university grades is weaker (than for high school grades), thus more interesting to see if we can find relations, and second, only having two exam grades improves the visualization of the results.

Dataset. The dataset used is composed of $12\,412$ samples, 22 explanatory variables, and 2 numeric target variables. The explanatory variables refer to the socioeconomic background of the students at the time of high school, and they are made of variables such as parents level of education, the household income, which type of high school they attended, the utilities available at home (e.g., internet and television), and their neighborhood stratum[11]. The numeric targets represent their grades, from 0% to 100%, in two national university-level exams, namely quantitative reasoning and English.
An additional reason for selecting this dataset is that it violates two of our model assumptions: $1)$ the target variables values are truncated between 0 and 100, thus violating the use of a continuous normal distribution to describe them; and $2)$ the target variables are not independent, as suggested by a correlation of 53%. If our approach is shown to work despite these violations, we may consider this is a good result.

5.6.1 Analysis of subgroups obtained with RSD

The first four subgroups with absolute ($\beta = 0$) and normalized ($\beta = 1$) gain can be seen in Figures 5.7a and 5.7b, respectively. The distributions of the first two subgroups for both gains can be seen in Figures 5.8a, 5.8b, 5.8c, and 5.8d. The two extreme gains were used to show that from a user perspective it can be interesting to

[11] Stratum is a classification system unique to Colombia, where districts are ranked based on their affluence level from 1 to 6, where 1 is the lowest level `https://www.dane.gov.co/index.php/servicios-al-ciudadano/servicios-informacion/estratificacion-socioeconomica` (Accessed on 19 April 2021).

use different gains depending on the goal of the data exploration, i.e., coarse versus fine-grained perspective.

Comparison of absolute and normalized gain. Overall, with absolute and normalized gain our method finds 7 and 34 subgroups that cover a total of 84% and 92% of the data, respectively. Looking at Figures 5.8a, 5.8b, 5.8c, and 5.8d, it can be seen that normalized gain favors smaller and compact subgroups that deviate more from the dataset distribution, while absolute gain favors larger subgroups that deviate less from the dataset distribution. These conclusions can be verified by noting that normalized gain subgroups tend to have a smaller standard deviation, between 5% and 9%, while absolute gain has values in the same order of magnitude of the dataset distribution, i.e., around 23%.

Interpretation of the results. Both normalized and absolute gain results point to the fact that having a 'better' socioeconomic background is associated with higher grades on average in both types of exams, and the contrary is associated with lower grades. This is clearer in the absolute gain case, as each subgroup covers more data. It is noticeable in Figure 5.8b that a subgroup with a standard deviation similar to the dataset leads to subgroups that are spread throughout the whole range of values. Nonetheless, that subgroup covers more regions with lower grades than the dataset, making it a relevant result to understand the dataset better.

In general, it can be seen that some conditions appear often in the subgroups, such as $household_income$ above and below 5 minimum wages and education of one of the parents parents equal or above high school. It seems that the presence or absence of these variables is highly associated with above or below-average performance, respectively.

Looking at specific subgroups, it is interesting to see that in the 4^{th} subgroup of the absolute gain, the Quantitative reasoning grade is equal to the average behavior of the dataset (77%), while the English grade is 8% above average. Looking at the subgroups with normalized gain, we see that there are only slight variations of their descriptions and that they belong to a similar socioeconomic macro group but with slight differences in their descriptions, which corresponds to small differences in their grades distribution.

Violation of the model assumptions. Here we can observe how our method behaves when some modeling assumptions are violated. Regarding the truncated values, it seems that the normalized gain is affected by grades around 100 (as seen in Figures 5.8c and 5.8d) as most of its subgroups capture these students, which increases the average and lowers the standard deviation, making them rank higher. Our method

was not developed for highly stratified target values, but the results seem to show that it does not seem prohibitive to the use of RSD in these cases as long as the stratification is mild and the user takes into account this fact.
Regarding the independence assumption, it seems that the subgroups found are still relevant although both grades are almost always taken into account together, i.e., as the values are positively correlated it is more likely to find subgroups with mean values that are high or low for both exams, but not high for one and low for the other. This is expected as the encoding of independent normal distributions does not take into account the covariance between target variables, and thus that case is not deemed a deviation by the current model formulation.

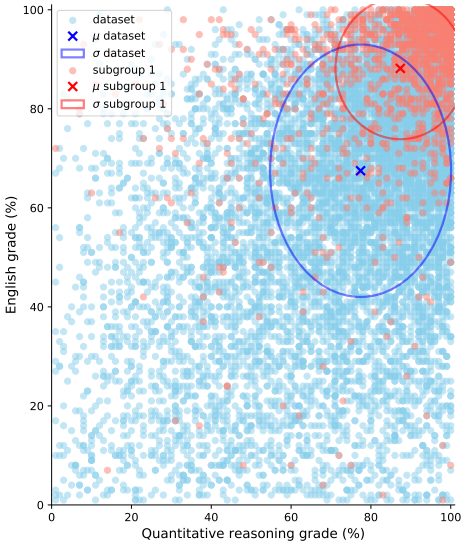
s	**description** of a student socioeconomic background	n_s	Quant.(%)	English(%)
1	household_income ≥ 5 min. wage & public_school = no & edu_mother $>$ high_school & Microwave = yes	1676	87 ± 16	88 ± 14
2	household_income < 5 min. wage & stratum < 5 & public_school = yes	4031	72 ± 25	54 ± 26
3	gender = M & edu_father $\geq$ high_school & social_support = None & stratum > 3 & public_school = no	1478	85 ± 17	78 ± 20
4	social_support = None & edu_father $>$ high_school & public_school = no & internet = yes & mobile = yes	997	77 ± 22	76 ± 19
	$\vdots$			
dataset distribution		1945^*	77 ± 23	68 ± 26

(a) Subgroup list with absolute gain ($\beta = 0$). First 4 subgroups of a total of 7 and swkl $= 0.41$

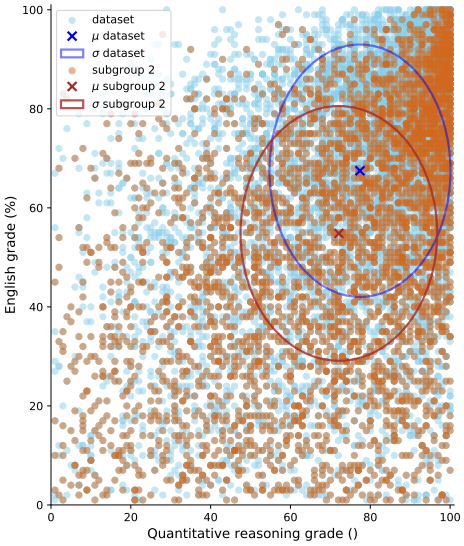
s	**description** of a student socioeconomic background	n_s	Quant.(%)	English(%)
1	household_income ≥ 5 min. wage & gender = M & household_size < 3 & edu_father $>$ high-school & mobile = yes	39	96 ± 5	92 ± 6
2	household_income ≥ 5 min. wage & school_type = academic & occ._mother = retired & edu_father $\geq$ Undergrad	23	96 ± 5	95 ± 4
3	household_income ≥ 5 min. wage & job_mother = independent & stratum ≥ 4 & gender = M & job_father = independent	30	96 ± 5	93 ± 6
4	job_mother = executive & stratum ≥ 4 & mobile = yes & job_father = independent & public_school = no	32	93 ± 9	94 ± 6
	$\vdots$			
dataset distribution		942^*	77 ± 23	68 ± 26

(b) Subgroup list with normalized gain ($\beta = 1$). First 4 subgroups of a total of 34 and swkl $= 0.52$

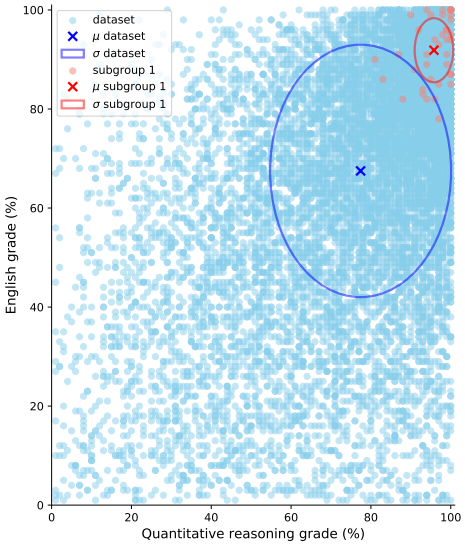
Figure 5.7: *Colombia engineering students* performance in Quantitative Reasoning and English exams. The results of Fig. 5.7a and 5.7b were obtained by RSD with *absolute gain* ($\beta = 0$) and *normalized gain* ($\beta = 1$). The dataset contains two numeric target variable *Quantitative Reasoning* and *English* exams in a 0-100% scale. The dataset represents 12 412 engineering students in Colombia, their grades in university national exams and their social-economic background. *Description* contains information regarding students socio-enconomic background, n_s the number of instances covered, Quant. and English the average grade and standard deviation in the respective exams. * The n of the dataset is the total number of instances in the dataset.



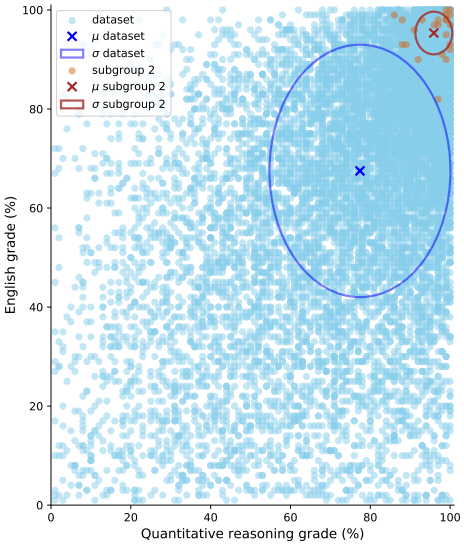
(a) 1^{st} subgroup with *absolute* gain.



(b) 2^{nd} subgroup with *absolute* gain.



(c) 1^{st} subgroup with *normalized* gain.



(d) 2^{nd} subgroup with *normalized* gain.

Figure 5.8: Scatter plot of the grades of students for Quantitative Reasoning and English exam, together with the grades associated with the descriptions of the 1^{st} and 2^{nd} with *absolute* and *normalized* gain.

5.7 Conclusions

We showed that finding good subgroup lists (ordered sets) that are both non-redundant and statistically robust, i.e., *robust subgroup discovery*, is computationally feasible. To achieve this, we proposed a heuristic algorithm dubbed RSD that approximates our MDL-based formulation of the problem using a greedy search that adds the subgroup that locally minimizes the MDL criteria to the list in each consecutive iteration. This approximation was shown to be equivalent to a Bayesian test (factor) between subgroup and dataset marginal target distributions plus a penalty for multiple hypothesis testing, which guarantees that each subgroup added to the list is statistically sound.
These assertions are supported by empirical evidence obtained on a varied set of 54 datasets. In the case of nominal targets, our method performed on par in terms of subgroup list quality, while obtaining smaller lists with fewer conditions. In the case of numeric targets and through the use of a deviation-aware measure, our method dominated in 92% of the cases.
Through a case study relating the socioeconomic background and national exams grades for Colombia engineering university students, we showed that RSD can be flexibly adapted to different goals of the user. In particular, it can change from a *fine-grained* perspective of the data that finds many subgroups that cover small parts of the data well, to a *coarse* perspective that finds few subgroups that cover large parts of the data. Also, it was shown that our method is robust to mild violations of our model assumptions.

Limitations. Even though the RSD algorithm has some appealing local statistical properties, we do not know how far the found models are from the optimal subgroup lists as defined by the global MDL criteria we proposed. Also, it does not scale very well for numeric targets, which was to be expected from the time complexity analysis. At the moment, multiple target variables are assumed to be independent, which can produce erroneous results when this assumption is violated. Preliminary experiments show that for moderately correlated variables (e.g., with a correlation of 0.5) this does not seem to be an issue, but there is no quantification of its implications. Similarly, for numeric targets, we use a normal distribution, and several datasets violate this assumption, either by behaving like a multi-modal or truncated distribution.

6

Conclusions

As machine learning and data mining permeate everyday life, the questions sought should be as much about algorithms as they should be about society itself. Algorithms increasingly affect the lives of individuals everywhere; thus, the pertinent questions are not only purely algorithmic but also about how they can help society solve systematic issues such as discrimination and social inequality.

For this reason, in recent years, we can find research on both classic topics such as algorithmic efficiency and statistical guarantees, and on newer issues such as privacy, fairness, and accountability. It is at the intersection of several of these topics, namely, algorithmic efficiency, statistical guarantees, and accountability, that we pose our main research question: "*How to learn robust and interpretable rule-based models from data for machine learning and data mining, and define their optimality?*".

In an honest attempt to answer it, we selected rule lists as models and the MDL principle as model selection theory. The former confers interpretability by design as humans can easily understand rule lists. At the same time, the latter allows for an objective formulation of learning rule lists from data that combines the performance and complexity of the model in one. Together, these allowed us to propose efficient algorithms that approximate our optimal formulation and achieve state-of-the-art performance while finding simpler models.

Nonetheless, this dissertation is just one step further in answering this research question. Its main limitation is our assumption that interpretability is associated with simplicity, but this is not always the case in reality. Interpretability is subjective, and it depends on the human that will act on or be acted upon by the model. Although

our MDL-based formulation attempts to have the least amount of assumptions, in some cases, it is necessary to add extra input from the human user to guarantee true interpretability.
Moreover, in Section 6.1 we present an overview of the main conclusions by chapter. Then, in Section 6.2 we discuss the strong and weak points of our proposal. Finally, in Section 6.3 we show possible directions of future work.

6.1 Summary

Chapter 1 introduced the scientific background and motivation for this dissertation.

Chapter 2 introduced the necessary mathematical background. In particular, it formally presented the tasks of rule-based prediction, subgroup discovery, and subgroup set discovery. Then, association rules, the standard component of these tasks, is promptly defined. Based on the previous definitions, we presented rule lists, i.e., the model class made of an ordered set of association rules. Furthermore, we distinguish predictive rule lists for machine learning and subgroup lists for data mining. Finally, it shows how to measure model quality in the classification and subgroup discovery setting.

Chapter 3 proposed an optimal formulation of predictive rule lists and subgroup lists for univariate and multivariate, nominal, and numeric target variables based on the Minimum Description Length (MDL) principle. Three new optimal data encodings for models that partition the data—rule lists, trees, clusters, etc.—are presented. In specific, these codes are: 1) the prequential plug-in code for nominal variables; 2) the Normalize Maximum Likelihood (NML) code for nominal variables; and 3) an objective Bayesian code with improper priors for numeric variables. We show that MDL-based subgroup lists with one subgroup are equivalent to top-1 subgroup discovery with weighted Kullback-Leibler divergence as a quality measure, thus validating subgroup lists as a valid generalization of subgroup discovery. Moreover, the best subgroup to add according to the MDL criteria maximizes an MDL equivalent to a Bayesian proportion, multinomial, or t-test plus a multiple hypothesis testing. In the end, we show the difference between predictive rules and subgroups through our MDL formulation of both problems.

Chapter 4 proposed CLASSY, a heuristic algorithm based on the MDL formulation of predictive rule lists for multiclass classification. Experiments show that it finds good predictive models that are also compact without hyperparameter tuning. CLASSY is composed of a frequent pattern mining algorithm to pre-mine all candidate rules and

then iteratively adds one rule at a time to the rule list. It effectively only has one hyperparameter, the pre-mined set of candidate rules. If this set is made large enough to accommodate all possible rules in the data, it can find good models independently of any hyperparameters—at the expense of computational budget. The empirical tests show state-of-the-art performance on classification, interpretability, and overfitting.

Chapter 5 proposed the *Robust Subgroup Discoverer* (RSD), a heuristic algorithm based on our MDL formulation that finds good subgroup lists for univariate and multivariate nominal and numeric target variables. Experiments over 54 datasets show that it outperforms state-of-the-art subgroup set discovery algorithms regarding the quality of sets found, especially for numeric targets. The algorithm iteratively uses a beam search to find candidates and then adds the one that locally minimizes the MDL optimal formulation. This approximation is equivalent to a Bayesian test (factor) between subgroup and dataset marginal target distributions plus a penalty for multiple hypothesis testing. Thus, we guarantee the statistical robustness of each subgroup in the list.

Chapters 4 and 5. The algorithms of both chapters share the greedy adding of rules, although CLASSY pre-mines all association rules, and RSD uses beam search at each iteration. The algorithms can be interchanged for both tasks in practice, although given their historical development, they do not overlap. Nonetheless, CLASSY reflects the intention of having few hyperparameters that we aimed for classification, and RSD demonstrates the flexibility necessary for data exploration, making them appropriate for their respective chapters.

6.2 Discussion

In this section we discuss the advantages and disadvantages of our proposals. To make this section consistent with the previous, we organize the discussion per chapter that proposes new work, i.e., Chapters 2, 3, 4, and 5.

Chapter 2. The only new proposal of this chapter is the *subgroup list* model class. Sequential subgroup set discovery was always defined heuristically, and each subgroup was interpreted individually without considering the previously found ones. We propose the first *global* dataset formulation of the problem of subgroup set selection that is equivalent to top-1 subgroup discovery in the case of a subgroup list with only one subgroup, and that also fits some of the previous heuristic definitions of sequential selection. Its main limitation is that subgroups far down in the list are hard to inter-

pret for large lists as they require considering previous ones. Also, this is not the only possible generalization of subgroup discovery to sets.

Chapter 3. In this chapter, we presented the MDL formulation of predictive rule lists and subgroup lists.

In the case of predictive rule lists, it allows using a single measure—the MDL score—to measure the bias-variance trade-off, one of the core problems in learning models from data. Even though predictive rule lists have long been used in machine learning, they have solely focused on classification and mostly on univariate target classification. We have proposed a theoretical formulation for classification and multi-target classification and regression. Compared with its Bayesian counterpart for rule lists for classification, our MDL formulation tries to make fewer assumptions, which we believe makes it more robust against overfitting. The main limitation of our formulation is that it assumes that a parsimonious model is interpretable, which is not always the case [26].

In the case of subgroup lists, it formulates a *global* perspective of sequential subgroup discovery. It generalizes the original problem of subgroup discovery to lists and gives it a balance between the complexity of the list and the quality of the descriptions.

Chapter 4. CLASSY is a heuristic algorithm with very few hyperameters that is competitive against state-of-the-art algorithms. It finds models with similar classification performance that are more compact and overfit less. It can also be made independent of its hyperparameters at a computational expense. The main drawback of our approach is that it is limited to binary input variables and single-target multiclass problems. Also, compared with RSD of Chapter 5 it does not provide local statistical guarantees for each of the added rules except that it improves the global score.

Chapter 5. RSD is a heurisitic algorithm that can find subgroup lists for univariate and multivariate nominal and numeric targets. Contrary to CLASSY it can deal with both nominal and numeric input variables. In the case of numeric targets, it uses a dispersion-aware measure to find subgroups with smaller standard deviations in the target values. Its normalization hyperparameter can change the granularity of the search from very specific to more general subgroups. One of its disadvantages is that we do not know how close we are to the global optimum. Also, when the normalization is used to the maximum (normalized gain), the algorithm is susceptible to noise in the data, i.e., it would find a different model if small variations are added to the data.

6.3 Future Work

This dissertation focuses on predictive rule lists for machine learning and subgroup lists for subgroup discovery based on the MDL principle. We decided to divide future work into technical developments that can be achieved soon —short and medium-term research—and the vision of the role rule-based models can take in machine learning and data mining— long-term research.

6.3.1 Short and medium-term research

Given the main topics of this dissertation, we will divide the technical advances into five different lines of research: 1) the MDL formulation of rule lists; 2) predictive rule lists; 3) subgroup lists; 4) search algorithms, and 5) rule sets.

1) The MDL formulation of rule lists can be extended to different types of target variables or distributions. First, it is straightforward to combine nominal and numeric targets through independent categorical and normal distributions using the MDL principle. It gives us an objective measure of both in bits. Then, instead of assuming independence between target variables, one can accommodate dependencies using multivariate numeric distributions. Finally, other types of distributions that can be more appropriate in different scenarios can be used, such as a Poisson distribution.

2) Predictive rule lists were only empirically tested for multiclass classification. Chapter 3 already defines the optimal predictive rule list for regression and multi-target classification and regression; thus, only the algorithm would need to be extended to these cases.

3) Subgroup lists, and similarly to the MDL future work, could accommodate non-independent distributions for multivariate targets and propose extensions to RSD find them.

4) Search algorithms. At the moment, only a greedy separate-and-conquer search is proposed. It would be essential to test the feasibility of optimal search algorithms such as branch-and-bound or Markov Chain Monte Carlo (MCMC).

5) Rule sets. In this dissertation, we only study rule lists (ordered rule sets). Extending the MDL theory and algorithms to overlapping rule sets would be a considerable development.

6.3.2 Long-term research

In terms of long-term research, the main directions we envision are related to better approximations of the real problems with fewer assumptions about the ideal behavior of the data. We will divide the topics into four groups: 1) interpretability; 2) rule-based models for sequential data; 3) rule-based models for image data; and 4) causal analysis.

1) Interpretability. As mentioned before, interpretability is subjective, and one cannot expect to have one universal formulation that works for everyone. Also, every person has a unique background that will make, e.g., some variables in the data easier to understand than others. For this reason, it is necessary to insert the human in the learning loop by having an algorithm that takes into consideration both objective concepts and the subjective nature of each individual. A path towards this end would be to start with an MDL formulation of a problem similar to ours, representing a *tabula rasa* or the minimum level of assumptions possible. Then, build upon the subjective characteristics of the user. This last part is crucial, and there are many options for it. It can be done at the beginning in the form of something similar to Bayesian priors or iteratively by presenting the user with a model and querying her about what they prefer (or not). At no point should the model overfit the data, and for that, the tabula rasa represents a baseline of the best formulation with minimum assumptions.

2) Rule-based models for sequential data. Even though there is already research on this direction, it tends to be composed of heuristics that lack statistical robustness. It would be interesting to conjugate the MDL principle with rule-based models for sequential data that formally consider dynamic learning and concept drift.

3) Rule-based models for image data. Rule-based models are shallow learners because they take the input variables as they come and do not transform them into more complex features. For this reason, to make rule-based models appropriate for image analysis, it is necessary that they either make their transformations or that they couple with other tools, such as classic computer vision techniques or neural networks that return human-understandable macro structures. A specific case of interest would be to use an image segmentation tool coupled with subgroup discovery to identify regions in the data that stand out with respect to a particular target, e.g., to describe areas in satellite image data with more pollution than the average.

4) Causal analysis. In supervised learning, it is usually assumed that any variable present in the data can be used to predict or describe the target variable. However, not

all variables have the same relationship with the target, as some can cause the target, be caused by it, or be independent. Also, input variables can be caused by other input variables. If one pays attention to the causal relationships when learning predictive models, it allows one to find robust models that generalize to distributions different from the training data. Also, considering the causal relationships in the data allows asking questions about counterfactuals or "What if something would have happened differently than we see on the data?".

Appendices

A

Kullback-Leibler divergence between two normal distributions

Let us assume two normal probability distributions, $p(x) \sim \mathcal{N}(\mu_p, \sigma_p)$ and $q(x) \sim \mathcal{N}(\mu_q, \sigma_q)$. The Kullback-Leibler divergence of q from p is:

$$\begin{aligned} KL_{\mu,\sigma}(p;q) &= \int_{-\infty}^{+\infty} p(x) \log p(x) \, \mathrm{d}x - \int_{-\infty}^{+\infty} p(x) \log q(x) \, \mathrm{d}x \\ &= \mathbb{E}_p \left[\log p(x)\right] - \mathbb{E}_p \left[\log q(x)\right] \\ &= -\frac{1}{2}\left(\log e + \log 2\pi\sigma_p^2\right) + \frac{1}{2}\log 2\pi\sigma_q^2 + \mathbb{E}_p\left[\frac{(x-\mu_q)^2}{2\sigma_q^2}\log e\right] \\ &= -\frac{\log e}{2} + \log\frac{\sigma_p}{\sigma_q} + \mathbb{E}_p\left[\frac{x^2 - 2x\mu_q + \mu_q^2}{2\sigma_q^2}\log e\right] \\ &= -\frac{\log e}{2} + \log\frac{\sigma_q}{\sigma_p} + \frac{\sigma_p^2 + \mu_p^2 - 2\mu_p\mu_q + \mu_q^2}{2\sigma_q^2}\log e \\ &= -\frac{\log e}{2} + \log\frac{\sigma_q}{\sigma_p} + \frac{\sigma_p^2 + (\mu_p - \mu_q)^2}{2\sigma_q^2}\log e. \end{aligned} \tag{A.1}$$

Note that in the specific case where the Kullback-Leibler divergence only takes into account the means and assumes both standard deviations equal, i.e., $p(x) \sim \mathcal{N}(\mu_p, \sigma)$ and $q(x) \sim \mathcal{N}(\mu_q, \sigma)$ one obtains:

$$KL_{\mu}(p;q) = \frac{(\mu_p - \mu_q)^2}{2\sigma^2}\log e, \tag{A.2}$$

and the weighted version of this KL_μ, i.e., $WKL_\mu = nKL_\mu(p;q)$, is similar to the most common subgroup discovery quality functions used for numeric targets that do

not take into account the dispersion of the subgroup, such as the weighted relative accuracy or the mean-test [75], which uses the square root of KL_{μ}. We will call this measure the Weighted Kullback-Leibler without dispersion.

B

Prequential plug-in encoding for rule lists with categorical distributions

For this section, let us assume that we have a dataset $D = \{\mathbf{X}, Y\}$, Y has $k = |\mathcal{Y}|$ class labels and a model M that forms a partition over the whole data. The model M divides the data D in ω parts, of the form $\{(\mathbf{X}^1, Y^1), \cdots, (\mathbf{X}^\omega, Y^\omega)\}$. Each part has an associated categorical distribution with estimated parameters $\hat{\Theta}^i$ over the target part Y^i (as defined in Section 2.4).

Before introducing the *prequential plug-in* code it is necessary to introduce one main building block, the smoothed maximum likelihood estimator for a subset i:

$$\hat{p}_{c|i} = \frac{n_{c|i} + \epsilon}{n_i + |\mathcal{Y}|\epsilon}. \tag{B.1}$$

Unlike the regular maximum likelihood estimator, this smoothed variant—known as Laplace smoothing—adds a (small) pseudocount ϵ to each class-specific usage even when that class has no counts. This avoids zero probabilities for any class label and corresponds in Bayesian statistics to using a symmetric Dirichlet prior ϵ for each class [42].

Now, the main idea of the *prequential plug-in* code is to sequentially predict the points in a subset, starting with no knowledge about their distribution and updating it each time it receives a point using the Equation (B.1). Intuitively, this means that one starts with a pseudocount ϵ for each possible element, constructs a code using these pseudocounts, starts encoding/sending/decoding messages one by one, and then *updates the count of each element after sending/receiving each individual message*. The *prequential plug-in code* is asymptotically optimal even without any prior knowledge on the

probabilities [48].
Applying this idea to encode the class labels in Y and ignoring the data partition at the moment, initially each class label has a pseudocount of ϵ. Hence, when sending the first class label, y^1, we effectively use a uniform code, i.e., $-\log \frac{\epsilon}{k\epsilon}$. After that, however, we increase the count of that class label by one. Normalizing the updated counts results in a new categorical probability distribution—hence a new code: $-\log \frac{\epsilon+1}{k\epsilon+1}$. This code is the *best possible code given the data seen so far* and is equal to the smoothed maximum likelihood of Eq. (B.1). Formally, the plug-in code for encoding the class labels is defined as

$$\Pr_{\text{plug-in}}(y^u = c \mid Y^{|u-1}) \coloneqq \frac{|\{y \in Y^{|u-1} \mid y = c\}| + \epsilon}{\sum_{c' \in \mathcal{Y}} |\{y \in Y^{|u-1} \mid y = c'\}| + \epsilon}, \tag{B.2}$$

where $u \in \mathbb{N}$, y^u represents the u^{th} class label in Y, $Y^{|u-1} = \{y^1, ..., y^{u-1}\}$ represents the sequence of the $u-1$ first class labels, and ϵ is the pseudocount necessary for $\Pr_{\text{plug-in}}(y^1 = c \mid Y^{|0}) = \epsilon/k\epsilon = 1/k$ to be valid. The most common values for ϵ, which takes the role of a prior in the Bayesian literature [125], are the Jeffrey's prior of 0.5 or the uniform prior of 1. For simplicity in our experiments, the value of $\epsilon = 1$ was used to obtain natural factorials instead of gamma functions as can be seen next.
We now show how this prequential plug-in code can be used in the encoding of the class labels of a dataset partitioned in ω parts. But assuming no interaction between the parts, the total encoding is equal to the sum of its parts:

$$L_{\text{plug-in}}(Y \mid \mathbf{X}, M) = -\log \prod_{i}^{\omega} \Pr_{\text{plug-in}}(Y^i) = \sum_{i}^{\omega} L_{\text{plug-in}}(Y^i), \tag{B.3}$$

where $L_{\text{plug-in}}(Y^i) = -\log \Pr_{\text{plug-in}}(Y^i)$.
Inserting the prequential plug-in code (B.2) in (B.3) we obtain for each part Y^i:

$$\begin{aligned} L_{\text{plug-in}}(Y^i) &= -\log \left(\prod_{u=1}^{n_i} \Pr_{\text{plug-in}}(y^u \mid Y^{i|u-1}) \right) \\ &= -\log \left(\frac{\prod_{c=1}^{k} \prod_{u=0}^{n_{c|i}-1} (u+\epsilon)}{\prod_{u=0}^{n_i-1} (u+k\epsilon)} \right) \\ &= -\log \left(\frac{\prod_{c=1}^{k} (n_{c|i}-1+\epsilon)!/(\epsilon-1)!}{(n_i-1+k\epsilon)!/(k\epsilon-1)!} \right) \\ &= -\log \left(\frac{\prod_{c=1}^{k} \Gamma(n_{c|i}+\epsilon)/\Gamma(\epsilon)}{\Gamma(n_i+k\epsilon)/\Gamma(k\epsilon)} \right), \end{aligned} \tag{B.4}$$

where $Y^{i|u}$ is a sequence of class labels of length u in part D^i, and $n_i = |D^i|$ and $n_{c|i} = |D^{c|i}|$. Further, Γ is the gamma function, an extension of the factorial to real and complex numbers that is given by $\Gamma(u) = (u-1)!$.

This code starts from sequential data, but as one can see in Eq. (B.4), the order in which one transmits class labels does not matter. In the end, the formulation is order agnostic and only depends on the counts per class label.

C

Normalized Maximum Likelihood for rule lists with categorical distributions

For this section, let us assume that we have a dataset $D = \{\mathbf{X}, Y\}$ and model M that forms a partition over the whole data. Model M divides the data D in ω parts, of the form $\{(\mathbf{X}^1, Y^1), \cdots, (\mathbf{X}^\omega, Y^\omega)\}$. Each part has an associated categorical distribution with estimated parameters $\hat{\Theta}^i$ over the target part Y^i (as defined in Chapter 2.4).
Here we show that the NML encoding of a partition equals the sum of the NML encoding of its parts:

$$L_{NML}(Y \mid \mathbf{X}, M) = \sum_{i=1}^{\omega} L_{NML}(Y^i). \tag{C.1}$$

Note that in the case of a subgroup list, as the default rule does not require NML encoding, the M used in this section represents the subgroups S, and D represents the data covered by these. In the case of a tree or rule list, M represents the model that partitions the data at the leaves and rules (including default rule), respectively, and D the whole dataset. There is no loss of generality for subgroup lists as the separation property allows us to separate the encoding of the default rule.
First, lets recall the definition of the NML probability distribution [115]:

$$L_{NML}(Y \mid \mathbf{X}, M) = -\log\left(\frac{\Pr(Y \mid \mathbf{X}; \hat{M}(Y \mid \mathbf{X}))}{\sum_{Z \in \mathcal{Y}^n} \Pr(Z \mid \mathbf{X}; \hat{M}(Z \mid \mathbf{X}))}\right),$$

where $\mathcal{Y}^n$ is the set of all possible sequences of n points with $k = |\mathcal{Y}|$ categories, $\hat{M}(Y \mid \mathbf{X})$ and $\hat{M}(Z \mid \mathbf{X})$ are the models with parameters estimated according to the maximum likelihood over the data Y and Z, respectively. Taking into account that

our data is independent and identically distributed (*i.i.d.*), and that our model M partitions the data into ω parts, we can further develop the previous formula to:

$$\begin{aligned} L_{NML}(Y \mid \mathbf{X}, M) &\overset{\text{i.i.d.}}{=} -\log\left(\frac{\prod_{i=1}^{n}\Pr(y^i \mid \mathbf{x}^i; \hat{M}(Y \mid \mathbf{X}))}{\sum_{Z\in\mathcal{Y}^n}\prod_{i=1}^{n}\Pr(z^i \mid \mathbf{x}^i; \hat{M}(Z \mid X))}\right) \\ &= -\log\left(\frac{\prod_{i'=1}^{\omega}\Pr(Y^{i'}; \hat{\Theta}(Y^{i'}))}{\sum_{Z\in\mathcal{Y}^n}\prod_{i'=1}^{\omega}\Pr(Z^{i'}; \hat{\Theta}(Z^{i'}))}\right) \\ &= -\log\left(\frac{\prod_{i'=1}^{\omega} l(\hat{\Theta}^{i'} \mid Y^{i'})}{g(Y, X, M)}\right) \\ &= -\log\left(\sum_{i'=1}^{\omega} l(\hat{\Theta}^{i'} \mid Y^{i'})\right) + \log g(Y, X, M), \end{aligned} \tag{C.2}$$

where $l(\hat{\Theta}^{i'} \mid Y^{i'})$ is the likelihood function for each of the ω parts and $g(Y, X, M)$ is a complexity function that depends on these three variables.
The first term is already independent for each part, although the second is not.
Let us now look at $g(Y, X, M)$ when we only have one part in the dataset, i.e., D^1. We will call this term the NML complexity of a multinomial distribution and denote it by $\mathcal{C}(n_1, k)$ of one part $D^1 = \{Y^1, X^1\}$, with $n_1 = |D^1|$ and $k = \mathcal{Y}$

$$\begin{aligned} \mathcal{C}(n_1, k) &= \log\left(\sum_{Z\in\mathcal{Y}^{n_1}}\Pr(Z^1; \hat{\Theta}(Z^1))\right) \\ &= \log\left(\sum_{Z\in\mathcal{Y}^{n_1}}\prod_{i=1}^{n_1}\Pr(z^i; \hat{\Theta}(Z^1))\right) \\ &= \log\left(\sum_{n_{1|1}+n_{2|1}+\ldots+n_{k|1}=n_1}\frac{n_1!}{n_{1|1}!n_{2|1}!\ldots n_{k|1}!}\prod_{c\in\mathcal{Y}}\left(\frac{n_{c|1}}{n_1}\right)^{n_{c|1}}\right) \end{aligned} \tag{C.3}$$

where $n_{c|1}$ is the number of points of category c in Y^1, and the passage from the second equality to the last is a property of multinomial distributions commonly used to make the computation of $\mathcal{C}(n_a, k)$ simpler [48]. It is interesting to note that $\mathcal{C}(n_a, k)$ only depends on the number of points in Y^1 and its cardinality, not on the actual values. This term, i.e., the complexity of a multinomial distribution over n_1 points with k possible values, measures the likelihood of each possible sequence.
Now we must generalize from a part to the partition of the dataset. To illustrate how to do this, let us first look at Table C.1, which shows an example of all the possible sequences in a fixed-length three-part partition of the data. Taking into account those

Table C.1: All possible sequences of a partition of fixed length of the data in three parts. Fixed length means that all possible parts always have the same amount of points, as e.g. $|A_1| = |A_2| = \cdots = |A_a| = n_A$.

Part 1	Part 2	Part 3
A_1	B_1	C_1
A_1	B_1	C_2
$\vdots$	$\vdots$	$\vdots$
A_1	B_2	C_1
$\vdots$	$\vdots$	$\vdots$
A_a	B_b	C_c

three parts, let us look at how the probabilities of all those sequences could be computed:

$$\begin{aligned}\sum_{\forall a,b,c} \Pr(A_a)\Pr(B_b)\Pr(C_c) &= \left(\sum_{\forall a}\Pr(A_a)\right)\cdot\left(\sum_{\forall b,c}\Pr(B_b)\Pr(C_c)\right)\\ &= \left(\sum_{\forall a}\Pr(A_a)\right)\cdot\left(\sum_{\forall b}\Pr(B_b)\right)\cdot\left(\sum_{\forall c}\Pr(C_c)\right),\end{aligned}$$

where this follows naturally from the distributive property of the multiplication. It is easy to see that this generalizes to partitions of any number of parts. Thus, going back to the complexity term $g(Y, X, M)$, we can see that

$$\begin{aligned}\log g(Y,X,M) &= \log \sum_{Z\in\mathcal{Y}^n}\prod_{i'=1}^{\omega}\Pr(Z^{i'};\hat{\Theta}(Z^{i'}))\\ &= \log\prod_{i'=1}^{\omega}\sum_{Z^{i'}\in\mathcal{Y}^{n_{i'}}}\Pr(Z^{i'};\hat{\Theta}(Z^{i'}))\\ &= \sum_{i'=1}^{\omega}\log\sum_{Z^{i'}\in\mathcal{Y}^{n_{i'}}}\Pr(Z^{i'};\hat{\Theta}(Z^{i'}))\\ &= \sum_{i'=1}^{\omega}\log\mathcal{C}(n_{i'},k)\end{aligned} \tag{C.4}$$

Substituting this back into Eq. (C.2), we obtain what we wanted:

$$\begin{aligned} L_{NML}(Y \mid \mathbf{X}, M) &= -\log\left(\sum_{i=1}^{\omega} l(\hat{\Theta}^i \mid Y^i)\right) + \sum_{i=1}^{\omega} \log \mathcal{C}(n_i, k) \\ &= \sum_{i=1}^{\omega} l(\hat{\Theta}^i \mid Y^i) + \mathcal{C}(n_i, k) \\ &= \sum_{i=1}^{\omega} L_{NML}(Y^i) \end{aligned} \tag{C.5}$$

D

Bayesian encoding of a normal distribution with mean and standard deviation unknown

For encoding a sequence of numeric valued i.i.d. observations such as $Y = \{y_1, ..., y_n\}$, the Bayesian encoding takes the following form:

$$P_{Bayes}(Y) = \int_{\Theta} f(Y \mid \Theta) w(\Theta) \, \mathrm{d}\Theta, \tag{D.1}$$

where f is the probability density function (pdf), Θ is the set of parameters of the distribution, and $w(\Theta)$ the prior over the parameters. In the case of a normal distribution $\Theta = \{\mu, \sigma\}$, with μ and σ being its mean and standard deviation, respectively, the pdf $f(Y \mid \Theta)$ over a sequence Y is the multiplication of the individual pdfs, thus:

$$f(Y \mid \mu, \sigma) = \frac{1}{(2\pi)^{n/2}\sigma^n} \exp\left[-\frac{1}{2\sigma^2}\sum_{i}^{n}(y^i - \mu)^2\right], \tag{D.2}$$

In order not to bias the encoding for specific values of the parameters, we choose to use the constant Jeffrey's prior of $1/\sigma^2$ for the unknown parameters μ and σ, and add an extra. Thus, our prior is given by:

$$w(\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma^2}, \tag{D.3}$$

where $1/\sqrt{2\pi}$ was added for normalization reasons.

Putting everything together, one obtains:

$$P_{Bayes}(Y) = \\ = (2\pi)^{-\frac{n+1}{2}} \int_{-\infty}^{+\infty} \int_{0}^{+\infty} \frac{1}{\sigma^{n+2}} \exp\left[-\frac{1}{2\sigma^2}\left(\sum_{i}^{n}(y^i-\mu)^2\right)\right] \mathrm{d}\sigma\, \mathrm{d}\mu. \tag{D.4}$$

The integrals over the whole space of the parameters μ and σ allow to penalize the fact that we do not know the statistics *a priori*, thus penalizing the fact that a distribution over n points could, by chance, have the same statistics as the one found in the data.

Note that using an improper prior requires that we somehow make it proper, i.e., we need to find a way to make the integration over the prior finite $\int\int w(\mu,\sigma) = K$, where K is a constant value. The usual way to make an improper prior finite is to condition on the k minimum number observations $Y^{|k} \in Y$ needed to make the integral proper [48], which in the case of two unknowns (μ and σ) is $k = 2$. Thus, instead of using $w(\mu,\sigma)$ we will in practice be using $w(\mu,\sigma \mid Y^{|2})$, and using the the chain rule and the Bayesian formula returns a total encoding of Y equal to

$$P(Y) = P_{Bayes}(Y \mid Y^{|2})P(Y^{|2}) = \frac{P_{Bayes}(Y)}{P_{Bayes}(Y^{|2})}P(Y^{|2}) \tag{D.5}$$

where $P(Y^{|2})$ is a non-optimal probability used to define $Y^{|2} = \{y^1, y^2\}$ that we will define later and y^1, y^2 chosen in a way that maximizes $P(Y)$. Now that we have all the ingredients to define $P(Y)$ we will start by defining $P_{Bayes}(Y)$ and then choose the appropriate probability for $P(Y^{|2})$.

To solve the first integral of $P_{Bayes}(Y)$ in Eq. (D.4), we integrate in σ and note that the formula is an instance of the gamma function,

$$\Gamma(k) = \int_{0}^{+\infty} z^{k-1}e^{-z}\,\mathrm{d}z, \tag{D.6}$$

with the corresponding variable transformation:

$$z = \frac{A}{2\sigma^2};\ \frac{1}{\sigma} = \frac{2^{1/2}z^{1/2}}{A^{1/2}};\ \mathrm{d}\sigma = -\frac{\sigma}{2z}\,\mathrm{d}z;\ A = \left[\sum_{i}^{n}(y^i-\mu)^2\right], \tag{D.7}$$

Performing the variable transformation and noting that the minus sign of $\mathrm{d}z$ cancels with the reversing of the integral limits, we get:

$$P_{Bayes}(Y) = \\ = \Gamma\left(\frac{n+1}{2}\right) 2^{\frac{n+1}{2}-1}(2\pi)^{-\frac{n+1}{2}} \int_{-\infty}^{+\infty} \left[\sum_{i}^{n}(y^i-\mu)^2\right]^{-\frac{n+1}{2}} \mathrm{d}\mu, \tag{D.8}$$

which reveals that the prior on the effect size ρ, and specifically its standard deviation parameter τ, is equivalent to adding $1/\tau^2$ virtual points to the original data.

To solve the integral in μ we need to introduce the statistics $\hat{\mu}$ and $\hat{\sigma}$ as the values estimated from the data. We define these quantities as:

$$\hat{\mu} = \frac{1}{n}\sum_{i}^{n} y^i;\ \hat{\sigma}^2 = \frac{1}{n}\sum_{i}^{n}(y^i - \hat{\mu})^2\ , \tag{D.9}$$

where $\hat{\mu}$ is the mean estimator over n data points and $\hat{\sigma}^2$ is the estimator of the variance. Note that for the variance the biased version with n was used instead of with $n-1$ as it allows to compute the Residual Sum of Squares (RSS) directly by $RSS = n\hat{\sigma}$.

Focusing now on the interior part of the integral of Eq. D.8 and rewriting it in order to resemble the t-student distribution, we obtain:

$$\begin{aligned}
&\left[\sum_{i}^{n}(y^i - \mu)^2\right]^{-(n+1)/2} = \\
&\left[\sum_{i}^{n}(y^i)^2 - n\hat{\mu}^2 + n\hat{\mu}^2 - 2n\hat{\mu}\mu + n\mu^2\right]^{-(n+1)/2} = \\
&\left[\sum_{i}^{n}(y^i)^2 - n\hat{\mu}^2 + n(\hat{\mu} - \mu)^2\right]^{-(n+1)/2} = \\
&\left[n\hat{\sigma'}^2 + n(\hat{\mu}' - \mu)^2\right]^{-(n+1)/2} = \\
&\left[n\hat{\sigma}^2\right]^{-(n+1)/2}\left[1 + \frac{(\hat{\mu} - \mu)^2}{\hat{\sigma}^2}\right]^{-(n+1)/2} \\
&\left[n\hat{\sigma}^2\right]^{-(n+1)/2}\left[1 + \frac{1}{n}\left(\frac{\hat{\mu} - \mu}{s_s^2}\right)^2\right]^{-(n+1)/2},
\end{aligned} \tag{D.10}$$

where $s_s^2 = \hat{\sigma}^2/n$ is the "sampling" variance. Now, taking into account the fact that the integral of the t-student distribution over the whole space is equal to one, and reshuffling around its terms we get

$$\int_{-\infty}^{+\infty}\left[1 + \frac{1}{n}\left(\frac{\hat{\mu} - \mu}{s_s}\right)^2\right]^{-\frac{n+1}{2}} \mathrm{d}\mu = \frac{\Gamma\left(\frac{n}{2}\right)\sqrt{\pi n}s_s}{\Gamma\left(\frac{n+1}{2}\right)}. \tag{D.11}$$

Inserting this back in Eq. D.4 we obtain:

$$\begin{aligned} &P_{Bayes}(Y) = \\ &= \Gamma\left(\frac{n+1}{2}\right) 2^{\frac{n+1}{2}-1}(2\pi)^{-\frac{n+1}{2}} \frac{\Gamma(\frac{n}{2})\sqrt{\pi n}s_s}{\Gamma(\frac{n+1}{2})} \left[n\hat{\sigma}^2\right]^{-(n+1)/2} \\ &= 2^{-1}\pi^{-\frac{n}{2}}\Gamma\left(\frac{n}{2}\right)\frac{1}{\sqrt{n}}\left[n\hat{\sigma}^2\right]^{-\frac{n}{2}}, \end{aligned} \tag{D.12}$$

Returning to the the conditional probability of Eq. (D.5), we see that we still need to define $P(Y^{|2})$, the non-optimal probability of the first two-points. As in the case of our model class we assume that the dataset overall statistics are known, i.e., $\Theta = \{\hat{\mu}_d, \hat{\sigma}_d\}$, we will use this distribution to find the probability of the points $Y^{|2} = \{y^1, y^2\}$ as :

$$P(Y^{|2}) = \log 2\pi + \log \hat{\sigma}_d + \left[\frac{1}{2\hat{\sigma}_d^2}\sum_i^2 (y^i - \hat{\mu}_d)^2\right]\log e. \tag{D.13}$$

Finally, applying the minus logarithm base 2 to all the terms in Eq (D.5) to obtain the total code length in bits,

$$\begin{aligned} &L_{Bayes2.0}(Y) = -\log P_{Bayes}(Y) + \log P_{Bayes}(Y^{|2}) - \log P(Y^{|2}) \\ &= 1 + \frac{n}{2}\log\pi - \log\Gamma\left(\frac{n}{2}\right) + \frac{1}{2}\log n + \frac{n}{2}\log\left(n\hat{\sigma}_n^2\right) \\ &- 1 - \frac{2}{2}\log\pi + 0 - \frac{1}{2} - \log\left(\sum_i^2 (y^i - \hat{\mu}_2)^2\right) \\ &+ \frac{2}{2}\log\pi + \log\hat{\sigma}_d + \left[\frac{1}{2\hat{\sigma}_d^2}\sum_i^2 (y^i - \hat{\mu}_d)^2\right]\log e \\ &= \frac{n}{2}\log\pi - \log\Gamma\left(\frac{n}{2}\right) + \frac{1}{2}\log n + \frac{n}{2}\log\left(n\hat{\sigma}_n^2\right) + L_{cost}(Y^{|2}), \end{aligned} \tag{D.14}$$

where $\hat{\mu}_2$ is the estimated mean of y^1, y^2 and $L_{cost}(Y^{|2})$ is the extra cost incurred of not being able to use a refined encoding for $Y^{|2}$. Now that the length of the encoding is defined, we just need to choose the two points. i.e., y^1, y^2. Because we want to minimize this length, we notice that there are only two terms that contribute to it in $L_{cost}(Y^{|2})$, and thus by choosing the two observations close to $\hat{\mu}_d$ minimizes both the encoding of $P(Y^{|2})$ and maximize $P_{Bayes}(Y^{|2})$ for most cases. There are exceptions to this, depending on the respective values of μ_d and y^1, y^2 but these are not significant to change the values too much and also requires less computational search to find the points.

E

Bayesian encoding convergence to BIC for large n

This section shows that for a large number of instances n, the Bayesian encoding of Appendix D converges to the Bayesian Information Criterion (BIC). Thus, Eq. (D.14)) converges to the encoding of a normal distribution with mean and standard deviation known plus $\log n$. First, the encoding of a normal distribution with mean and standard deviation known over n *i.i.d.* points is equal to the sum of the individual encodings:

$$L(Y \mid \hat{\Theta}) = \frac{n}{2}\log 2\pi + \frac{n}{2}\log \hat{\sigma}^2 + \left[\frac{1}{2\hat{\sigma}^2}\sum_i^n (y^i - \hat{\mu})^2\right]\log e. \tag{E.1}$$

Second, we need to use the Stirling's approximation of the Gamma function for large n:

$$\begin{aligned} &-\log\Gamma\left(\frac{n}{2}\right) \\ &\sim -\frac{1}{2}\log\pi - \frac{1}{2}\log(n-2) - \left(\frac{n}{2}-1\right)\log\left(\frac{n}{2}-1\right) + \left(\frac{n}{2}-1\right)\log e, \end{aligned} \tag{E.2}$$

and finally we insert it into Eq. (D.14) and assume $\tau = 1$ to obtain:

$$
\begin{aligned}
& L(Y) \sim \\
& \sim 1 + \frac{n-1}{2}\log\pi + \frac{1}{2}\log\left(\frac{n}{n-2}\right) + \frac{n}{2}\log\left(\frac{n\hat{\sigma}^2}{n/2-1}\right) + \left(\frac{n}{2}-1\right)\log e \\
& + \log\left(\frac{n}{2}-1\right) + L_{cost}(Y^{|2}) \\
& \sim \frac{n}{2}\log\pi + \frac{n}{2}\log 2\hat{\sigma}^2 + \left[\frac{1}{2\hat{\sigma}^2}\sum_i^n (y^i-\mu)^2\right]\log e + \log n - \log e + L_{cost}(Y^{|2}) \\
& = L(Y \mid \hat{\Theta}) + \log\frac{n}{e} + L_{cost}(Y^{|2}) \\
& \sim \frac{1}{2}\Big(2L(Y \mid \hat{\Theta}) + 2\log n - 2\log e\Big) \\
& = \frac{1}{2}BIC,
\end{aligned}
\tag{E.3}
$$

where from the second to the third line, we assumed large n, making some of the terms disappear, while the definition $n\hat{\sigma}^2 = \sum_i^n (y^i - \mu)^2$ is used for making the third term of the third expression appear. From the fourth to the fifth expressions, it assumes that $L_{cost}(Y^{|2})$ is negligible, as it is the cost of not being able to encode the first two points optimally. For the Bayes information criterion, we used its standard definition,

$$BIC = -2\ln\ell(\Theta \mid Y) + k\ln n, \tag{E.4}$$

where $\ell(\Theta \mid Y)$ is the likelihood as estimated from the data, and k is the number of parameters, which in our case is 2.

F

Datasets used for classification experiments

The 17 datasets used for classification are shown in Table F.1, and were retrieved from LUCS/KDD[1] repository. The datasets all have *binary* explanatory variables.

[1] http://cgi.csc.liv.ac.uk/~frans/KDD/Software/LUCS-KDD-DN/DataSets/dataSets.html

Table F.1: Dataset properties: number of {samples, binary variables, classes, average number of candidate patterns per fold for CLASSY with $n_{min.} = 5\%$ and $d_{max} = 4$}. The datasets are ordered first by number of classes and then by the number of samples.

Dataset	$\|D\|$	$\|V\|$	$\|\mathcal{Y}\|$	$\|Cands\|$
hepatitis	155	48	2	39137
ionosphere	351	155	2	332560
horsecolic	368	81	2	23552
cylBands	540	120	2	304749
breast	699	14	2	299
pima	768	34	2	543
tictactoe	958	26	2	1907
mushroom	8124	84	2	79602
adult	48842	96	2	7231
iris	150	14	3	144
wine	178	63	3	13439
waveform	5000	96	3	86889
heart	303	46	5	21876
pageblocks	5473	39	5	2902
led7	3200	22	10	2507
pendigits	10992	81	10	107001
chessbig	28056	54	18	1384

G

RSD supplementary empirical evaluation

G.1 Datasets used for subgroup discovery experiments

The datasets selected are commonly used in machine learning and subgroup discovery, and were retrieved from UCI [29], Keel [4], MULAN [117] repositories. The datasets for nominal and numeric targets experiments are in Table G.1 and G.2, respectively.

Table G.1: Nominal targets datasets for subgroup discovery: single-binary, single-nominal and multi-label. Dataset properties: number of {target variables T; target labels $|\mathcal{Y}|$; samples $|D|$; type of variables (nominal/numeric)}.

Dataset	T	$\lvert\mathcal{Y}\rvert$	$\lvert D\rvert$	$V(nom./num.)$
sonar	1	2	208	(0/60)
haberman	1	2	306	(0/3)
breastCancer	1	2	683	(0/9)
australian	1	2	690	(0/14)
TicTacToe	1	2	958	(9/0)
german	1	2	1 000	(13/7)
chess	1	2	3 196	(36/0)
mushrooms	1	2	8 124	(22/0)
magic	1	2	19 020	(0/10)
adult	1	2	45 222	(8/6)
iris	1	3	150	(0/4)
balance	1	3	625	(0/4)
CMC	1	3	1 473	(0/9)
page-blocks	1	5	5 472	(0/10)
nursery	1	5	12 960	(7/1)
automobile	1	6	159	(10/15)
glass	1	6	214	(0/10)
dermatology	1	6	358	(0/34)
kr-vs-k	1	18	28 056	(6/0)
abalone	1	28	4 174	(1/7)
emotions	6	2	593	(0/72)
scene	6	2	2407	(0/294)
flags	7	2	194	(9/10)
yeast	14	2	2417	(0/103)
birds	19	2	645	(/258)
genbase	27	2	662	(1186/0)
mediamill	101	2	43 907	(0/120)
CAL500	174	2	502	(0/68)
Corel5k	374	2	5000	(499/0)

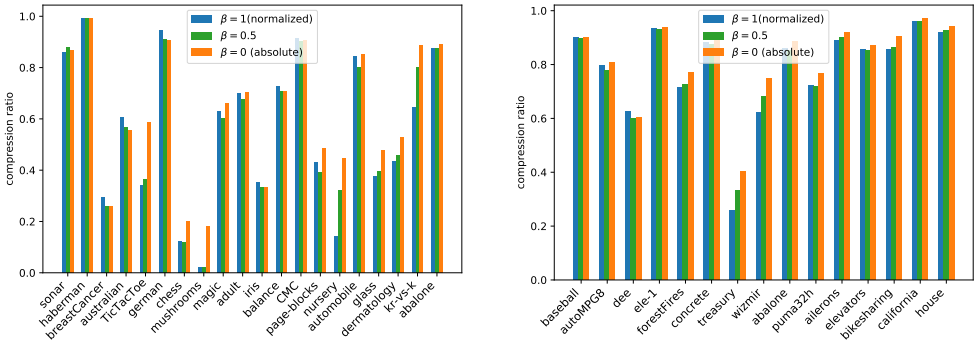
Table G.2: Numeric targets datasets for subgroup discovery: single-numeric and multi-numeric. Dataset properties: {number of target variables T; minimum and maximum target values $[min., max.]$; number of samples $|D|$; number of type of variables (nominal/numeric)}.

Dataset	T	$[min.; max.]$	$\|D\|$	$V(nom./num.)$
baseball	1	$[109; 6100]$	337	$(4/12)$
autoMPG8	1	$[9; 46.6]$	392	$(0/6)$
dee	1	$[0.8; 5.1]$	365	$(0/6)$
ele-1	1	$[80; 7675]$	495	$(0/2)$
forestFires	1	$[0; 1091]$	517	$(0/12)$
concrete	1	$[3; 21]$	1030	$(0/8)$
treasury	1	$[29; 90]$	1049	$(0/15)$
wizmir	1	$[29; 90]$	1461	$(0/9)$
abalone	1	$[1; 29]$	4177	$(0/8)$
puma32h	1	$[-0.0867; 0.0898]$	8192	$(0/32)$
ailerons	1	$[-0.0036; 0]$	13750	$(0/40)$
elevators	1	$[0.012; 0.078]$	16599	$(0/18)$
bikesharing	1	$[1; 977]$	17379	$(2/10)$
california	1	$[14999; 500001]$	20640	$(0/8)$
house	1	$[0; 500001]$	22784	$(0/16)$
edm	2	$[-1; 1]$	154	$(0/16)$
enb	2	$[6.01; 48.03]$	768	$(0/8)$
slump	3	$[0; 78]$	103	$(0/7)$
sf1	3	$[0; 4]$	323	$(0/10)$
sf2	3	$[0; 8]$	1066	$(0/10)$
jura	3	$[0.135; 166.4]$	359	$(0/15)$
osales	12	$[500; 795000]$	639	$(0/413)$
wq	14	$[0; 5]$	1060	$(0/16)$
oes97	16	$[30; 48890]$	334	$(0/263)$
oes10	16	$[30; 64560]$	403	$(0/298)$

G.2 Analysis of RSD compression gain hyperparameter

In this section, we present a thorough comparison of the normalization term β of RSD, where $\beta = 1$ is the *normalized* gain and $\beta = 0$ the *absolute* gain. RSD is executed with the same hyperparameters (beam width, number of cut points for numerical variables, and maximum depth of search) as in the experiments section, i.e., $w_b = 100$, $n_{cut} = 5$, $d_{max} = 5$. The different types of gain are compared for all the benchmark datasets described in the paper in terms of their compression ratio (defined later) in Figure G.1, Sum of Weighted Kullback-Leibler divergency (SWKL) in Figure G.2, and number of rules in Figure G.3. The compression ratio is the length of the found model $L(D, M)$ divided by the length of encoding the data with the dataset distribution (a model without subroups) $L(D \mid \hat{\Theta}^d)$

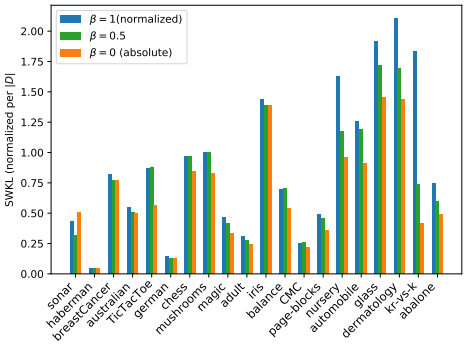
$$L\% = \frac{L(D, M)}{L(D \mid \hat{\Theta}^d)} \tag{G.1}$$



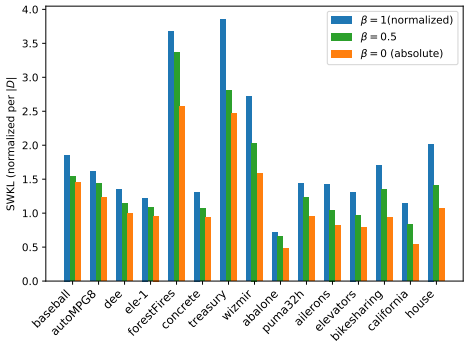
(a) Univariate nominal target.

(b) Univariate numeric target.

Figure G.1: Compression ratio obtained with $\beta = 0$ (absolute gain), $\beta = 0.5$, and $\beta = 1$ (normalized gain).

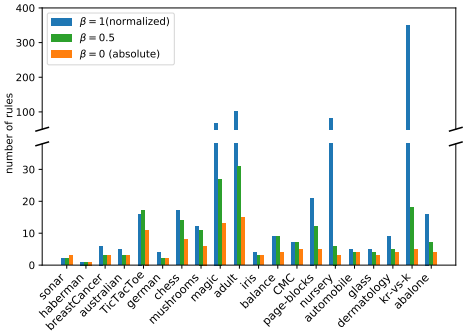


(a) Univariate nominal target.

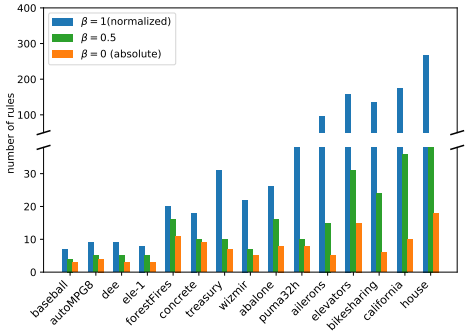


(b) Univariate numeric target.

Figure G.2: Normalized SWKL obtained with $\beta = 0$ (absolute gain), $\beta = 0.5$, and $\beta = 1$ (normalized gain).



(a) Univariate nominal target.



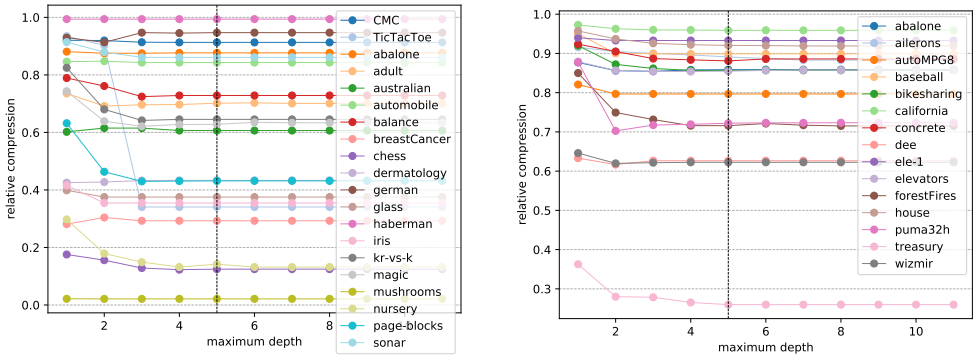
(b) Univariate numeric target.

Figure G.3: Number subgroups obtained with $\beta = 0$ (absolute gain), $\beta = 0.5$, and $\beta = 1$ (normalized gain).

G.3 Analysis of RSD beam search hyperparameters

In this section, we present a thorough comparison of the beam search hyperparameters influence on RSD output. As a complete search over the whole combination of hyperparameters is unfeasible, we present here an exploration over the hyperparameters used for the experimental comparison in the paper ($w_b = 100$, $n_{cut} = 5$, $d_{max} = 5$), i.e., we fix two of the parameters on the aforementioned values and then proceed to change the selected hyperparameter of interest, and we do this for all the 3 parameters. The line between the dots of the same colour does not represent an interpolation and is merely used to aid visualization and suggest trends.

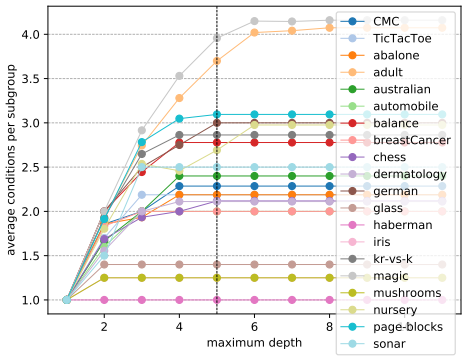
Note on relative compression. It may seem that the values of the relative compression remain constant but that is an illusion due to the scale of the y axis. As the compression ratio is given by the division of large values (usually above the thousands) its value with two decimal digits can be misleading. Nonetheless, in general, when zooming over the figures one can discern a slight improvement (smaller values) for larger values of the hyperparameters.



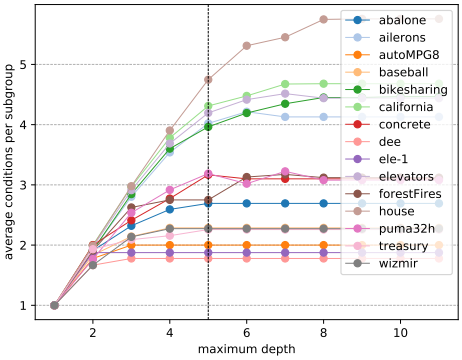
(a) Univariate nominal target.

(b) Univariate numeric target.

Figure G.4: Compression ratio obtained by varying the maximum search depth fixing $w_b = 100$, $n_{cut} = 5$ and $\beta = 1$ (normalized gain). The black vertical line represents the value used in the experiments section for subgroup lists (Section 5.3).

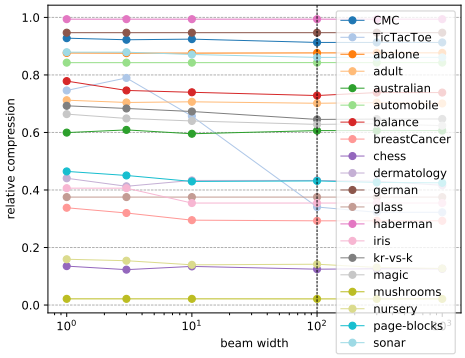


(a) Univariate nominal target.

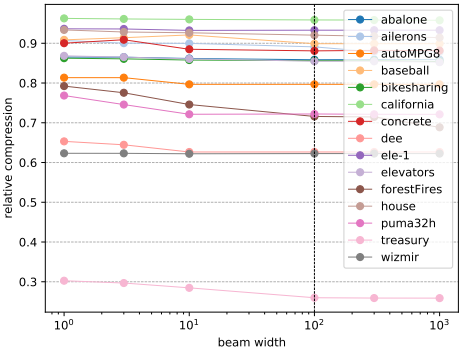


(b) Univariate numeric target.

Figure G.5: Average number of conditions per subgroup obtained by varying the maximum search depth fixing $w_b = 100$, $n_{cut} = 5$ and $\beta = 1$ (normalized gain). The black vertical line represents the value used in the experiments section for subgroup lists (Section 5.3).

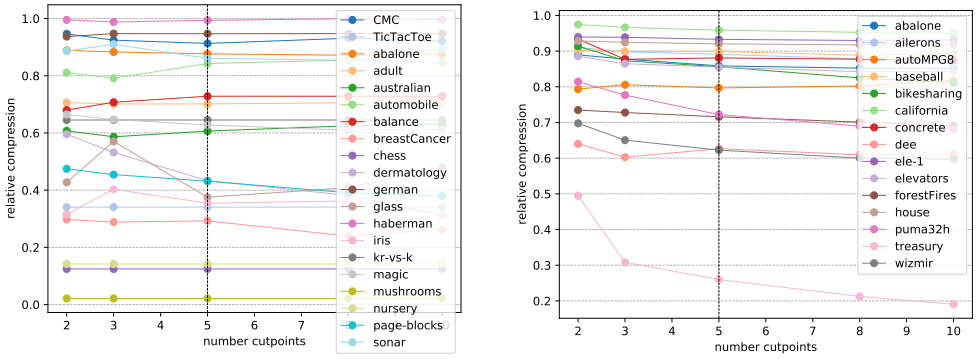


(a) Univariate nominal target.



(b) Univariate numeric target.

Figure G.6: Compression ratio obtained by varying the beam width and fixing $d_{max} = 5$, $n_{cut} = 5$ and $\beta = 1$ (normalized gain). The black vertical line represents the value used in the experiments section for subgroup lists (Section 5.3).



(a) Univariate nominal target.

(b) Univariate numeric target.

Figure G.7: Compression ratio obtained by varying the number of cut points and fixing $w_b = 100$, $d_{max} = 5$ and $\beta = 1$ (normalized gain). The black vertical line represents the value used in the experiments section for subgroup lists (Section 5.3).

G.4 Results of non-sequential subgroup set discovery algorithms

The comparison of RSD with subgroup set discovery algorithms that return sets (and not lists) can be seen in Table G.3.

Table G.3: Single nominal target results for non-sequential methods plus RSD. This includes single-binary, single-nominal, respectively separated by an horizontal line in the table. The properties of the datasets can be seen in Table G.1, and are ordered by number target variables, number of classes, and number of samples, in this order. The evaluation measures are {quality of the subgroup set swkl; number of subgroups $|S|$; and average number of conditions $|a|$}. Note that FSSD does not work for single-nominal case and MCTS4DM only works for datasets with the same type of explanatory variables and thus the empty values $-$. *as DSSD has as stopping criteria the maximum number of subgroups was selected as the number of subgroups found by RSD, and total overlapping subgroups were posteriorly removed.

	DSSD			MCTS4DM			FSSD			RSD		
datasets	swkl	$\|S\|*$	$\|a\|$	swkl	$\|S\|$	$\|a\|$	swkl	$\|S\|$	$\|a\|$	swkl	$\|S\|$	$\|a\|$
sonar	0.33	2	5	–	–	–	0.05	1	43	**0.43**	2	3
haberman	**0.08**	1	4	**0.08**	1	3	0.04	11	3	0.04	1	1
breastCancer	0.79	6	3	0.81	6	4	0.35	6	9	**0.82**	6	2
australian	0.50	3	3	0.54	7	6	0.33	15	12	**0.55**	5	2
tictactoe	0.50	4	3	–	–	–	0.20	5	3	**0.87**	16	2
german	**0.15**	4	5	–	–	–	0.10	6	11	0.14	4	3
chess	0.76	11	4	–	–	–	0.34	4	15	**0.97**	17	2
mushrooms	0.97	3	4	–	–	–	0.40	5	20	**1.00**	12	1
magic	0.30	40	3	–	–	–	0.06	3	10	**0.47**	69	4
adult	0.24	31	5	–	–	–	0.00	1	10	**0.31**	103	4
avg. rank	1.8	1.7	2.0	–	–	–	3.0	1.9	2.9	**1.2**	2.5	1.1
iris	1.44	3	2	**1.45**	4	3	–	–	–	1.44	4	1
balance	0.63	9	3	–	–	–	–	–	–	**0.69**	9	3
CMC	0.18	7	3	0.16	20	4	–	–	–	**0.25**	7	2
page-blocks	0.36	19	3	–	–	–	–	–	–	**0.49**	21	3
nursery	0.92	2	3	–	–	–	–	–	–	**1.63**	81	3
automobile	0.85	5	5	–	–	–	–	–	–	**1.25**	5	2
glass	1.55	3	1	1.12	5	6	–	–	–	**1.92**	5	1
dermatology	1.85	6	3	1.02	9	6	–	–	–	**2.11**	9	2
kr-vs-k	0.62	13	3	–	–	–	–	–	–	**1.83**	351	3
abalone	0.53	14	3	–	–	–	–	–	–	**0.74**	16	2
avg. rank	1.9	1.2	1.7		–	–	–	–	–	**1.1**	1.9	1.3

Bibliography

[1] The Bureau of Transportation Statistics (BTS), 2021. URL `https://www.bts.gov/`.

[2] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *Acm sigmod record*, volume 22, pages 207–216. ACM, 1993.

[3] J. Alcala-Fdez, R. Alcala, and F. Herrera. A fuzzy association rule-based classification model for high-dimensional problems with genetic rule selection and lateral tuning. *IEEE Transactions on Fuzzy systems*, 19(5):857–872, 2011.

[4] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera. Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, 17, 2011.

[5] E. Angelino, N. Larus-Stone, D. Alabi, M. Seltzer, and C. Rudin. Learning certifiably optimal rule lists. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 35–44. ACM, 2017.

[6] N. Antonio, A. de Almeida, and L. Nunes. Hotel booking demand datasets. *Data in brief*, 22:41–49, 2019.

[7] J. O. Aoga, T. Guns, S. Nijssen, and P. Schaus. Finding probabilistic rule lists using the minimum description length principle. In *International Conference on Discovery Science*, pages 66–82. Springer, 2018.

[8] M. Atzmueller. Subgroup discovery. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(1):35–49, 2015.

[9] A. Belfodil, A. Belfodil, and M. Kaytoue. Anytime subgroup discovery in numerical domains with guarantees. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 500–516. Springer, 2018.

[10] A. Belfodil, A. Belfodil, A. Bendimerad, P. Lamarre, C. Robardet, M. Kaytoue, and M. Plantevit. Fssd-a fast and efficient algorithm for subgroup set discovery. In *Proceedings of DSAA 2019*, 2019.

[11] E. Bellodi and F. Riguzzi. Structure learning of probabilistic logic programs by searching the clause space. *Theory and Practice of Logic Programming*, 15(2): 169–212, 2015.

[12] M. Boley, B. R. Goldsmith, L. M. Ghiringhelli, and J. Vreeken. Identifying consistent statements about numerical data with dispersion-corrected subgroup discovery. *Data Mining and Knowledge Discovery*, 31(5):1391–1418, 2017.

[13] C. Borgelt. Efficient implementations of apriori and eclat. In *FIMI'03: Proceedings of the IEEE ICDM workshop on frequent itemset mining implementations*, 2003.

[14] G. Bosc, J.-F. Boulicaut, C. Raïssi, and M. Kaytoue. Anytime discovery of a diverse set of patterns with monte carlo tree search. *Data Mining and Knowledge Discovery*, 32(3):604–650, 2018.

[15] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[16] B. Bringmann and A. Zimmermann. The chosen few: On identifying valuable patterns. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 63–72. IEEE, 2007.

[17] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann. The balanced accuracy and its posterior distribution. In *2010 20th international conference on pattern recognition*, pages 3121–3124. IEEE, 2010.

[18] K. Budhathoki and J. Vreeken. The difference and the norm—characterising similarities and differences between databases. In *Proceedings of ECMLP-KDD'15*, pages 206–223. Springer, 2015.

[19] K. Budhathoki, M. Boley, and J. Vreeken. Discovering reliable causal rules. *arXiv preprint arXiv:2009.02728*, 2020.

[20] T. Calders and S. Jaroszewicz. Efficient auc optimization for classification. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 42–53. Springer, 2007.

[21] P. Clark and T. Niblett. The cn2 induction algorithm. *Machine learning*, 3(4): 261–283, 1989.

[22] W. W. Cohen. Fast effective rule induction. In *Proceedings of the twelfth international conference on machine learning*, pages 115–123, 1995.

[23] A.-W. De Leeuw, L. A. Meerhoff, and A. Knobbe. Effects of pacing properties on performance in long-distance running. *Big Data*, 6(4):248–261, 2018.

[24] E. Delahoz-Dominguez, R. Zuluaga, and T. Fontalvo-Herrera. Dataset of academic performance evolution for engineering students. *Data in Brief*, page 105537, 2020.

[25] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

[26] F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv:1702.08608 [stat.ML]*, 2017.

[27] F. Doshi-Velez and B. Kim. Considerations for evaluation and generalization in interpretable machine learning. In *Explainable and Interpretable Models in Computer Vision and Machine Learning*, pages 3–17. Springer, 2018.

[28] X. Du, Y. Pei, W. Duivesteijn, and M. Pechenizkiy. Exceptional spatio-temporal behavior mining through bayesian non-parametric modeling. *Data Mining and Knowledge Discovery*, 34(5):1267–1290, 2020.

[29] D. Dua and C. Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[30] W. Duivesteijn and A. Knobbe. Exploiting false discoveries–statistical validation of patterns and quality measures in subgroup discovery. In *2011 IEEE 11th International Conference on Data Mining*, pages 151–160. IEEE, 2011.

[31] W. Duivesteijn, A. Knobbe, A. Feelders, and M. van Leeuwen. Subgroup discovery meets bayesian networks–an exceptional model mining approach. In *2010 IEEE International Conference on Data Mining*, pages 158–167. IEEE, 2010.

[32] W. Duivesteijn, A. J. Feelders, and A. Knobbe. Exceptional model mining. *Data Mining and Knowledge Discovery*, 30(1):47–98, 2016.

[33] T. Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8): 861–874, 2006.

[34] A. Fernandez, V. Lopez, M. J. del Jesus, and F. Herrera. Revisiting evolutionary fuzzy systems: Taxonomy, applications, new trends and challenges. *Knowledge-Based Systems*, 80:109–121, 2015.

[35] J. Fischer and J. Vreeken. Sets of robust rules, and how to find them. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 38–54. Springer, 2019.

[36] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[37] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, 32 (200):675–701, 1937.

[38] J. Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54, 1999.

[39] J. Fürnkranz, D. Gamberger, and N. Lavrač. *Foundations of rule learning*. Springer Science & Business Media, 2012.

[40] E. Galbrun. The minimum description length principle for pattern mining: A survey. *arXiv preprint arXiv:2007.14009*, 2020.

[41] M. García-Borroto, J. F. Martínez-Trinidad, and J. A. Carrasco-Ochoa. A survey of emerging patterns for supervised classification. *Artificial Intelligence Review*, 42(4):705–721, 2014.

[42] A. Gelman, H. S. Stern, J. B. Carlin, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 2013.

[43] B. R. Goldsmith, M. Boley, J. Vreeken, M. Scheffler, and L. M. Ghiringhelli. Uncovering structure-property relationships of materials by subgroup discovery. *New Journal of Physics*, 19(1):013031, 2017.

[44] M. Gönen, W. O. Johnson, Y. Lu, and P. H. Westfall. The bayesian two-sample t test. *The American Statistician*, 59(3):252–257, 2005.

[45] H. Grosskreutz and S. Rüping. On subgroup discovery in numerical domains. *Data Min. Knowl. Discov.*, 19(2):210–226, 2009.

[46] H. Grosskreutz, D. Paurat, and S. Rüping. An enhanced relevance criterion for more concise supervised pattern discovery. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1442–1450, 2012.

[47] P. Grünwald and T. Roos. Minimum description length revisited. *International Journal of Mathematics for Industry*, 11(1), 2019.

[48] P. D. Grünwald. *The minimum description length principle*. MIT press, 2007.

[49] W. Hämäläinen. Kingfisher: an efficient algorithm for searching for both positive and negative dependency rules with statistical significance measures. *Knowledge and information systems*, 32(2):383–414, 2012.

[50] W. Hämäläinen and G. I. Webb. Specious rules: an efficient and effective unifying method for removing misleading and uninformative patterns in association rule mining. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 309–317. SIAM, 2017.

[51] W. Hämäläinen and G. I. Webb. A tutorial on statistically sound pattern discovery. *Data Mining and Knowledge Discovery*, 33(2):325–377, 2019.

[52] F. Herrera, C. J. Carmona, P. González, and M. J. Del Jesus. An overview on subgroup discovery: foundations and applications. *Knowledge and information systems*, 29(3):495–525, 2011.

[53] F. Herrera, F. Charte, A. J. Rivera, and M. J. Del Jesus. Multilabel classification. In *Multilabel Classification*, pages 17–31. Springer, 2016.

[54] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70, 1979.

[55] J. Hühn and E. Hüllermeier. Furia: an algorithm for unordered fuzzy rule induction. *Data Mining and Knowledge Discovery*, 19(3):293–319, 2009.

[56] J. Huysmans, K. Dejaeger, C. Mues, J. Vanthienen, and B. Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51(1):141–154, 2011.

[57] R. L. Iman and J. M. Davenport. Approximations of the critical region of the fbietkan statistic. *Communications in Statistics-Theory and Methods*, 9(6):571–595, 1980.

[58] H. Jeffreys. *The theory of probability*. OUP Oxford, 1998.

[59] F. Jiménez, G. Sánchez, and J. M. Juárez. Multi-objective evolutionary algorithms for fuzzy classification in survival prediction. *Artificial intelligence in medicine*, 60(3):197–219, 2014.

[60] N. Jin, P. Flach, T. Wilcox, R. Sellman, J. Thumim, and A. Knobbe. Subgroup discovery in smart electricity meter data. *IEEE Transactions on Industrial Informatics*, 10(2):1327–1336, 2014.

[61] R. E. Kass and A. E. Raftery. Bayes factors. *Journal of the american statistical association*, 90(430):773–795, 1995.

[62] D. Klabjan. Large-scale models in the airline industry. In *Column generation*, pages 163–195. Springer, 2005.

[63] W. Klösgen. Explora: A multipattern and multistrategy discovery assistant. In *Advances in Knowledge Discovery and Data Mining*, pages 249–271. 1996.

[64] A. J. Knobbe and E. K. Ho. Pattern teams. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 577–584. Springer, 2006.

[65] P. Kontkanen, P. Myllymäki, W. Buntine, J. Rissanen, and H. Tirri. An mdl framework for data clustering. *Minimum*, page 323, 2005.

[66] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[67] H. Lakkaraju and C. Rudin. Learning cost-effective and interpretable treatment regimes for judicial bail decisions. *arXiv preprint arXiv:1610.06972*, 2016.

[68] H. Lakkaraju and C. Rudin. Learning cost-effective and interpretable treatment regimes. In *Artificial Intelligence and Statistics*, 2017.

[69] H. Lakkaraju, S. H. Bach, and J. Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of KDD'16*, pages 1675–1684. ACM, 2016.

[70] N. Lavrač, P. Flach, and B. Zupan. Rule evaluation measures: A unifying view. In *International Conference on Inductive Logic Programming*, pages 174–185. Springer, 1999.

[71] N. Lavrač, B. Kavšek, P. Flach, and L. Todorovski. Subgroup discovery with cn2-sd. *Journal of Machine Learning Research*, 5(Feb):153–188, 2004.

[72] M. van Leeuwen. Maximal exceptions with minimal descriptions. *Data Mining and Knowledge Discovery*, 21(2):259–276, 2010.

[73] M. van Leeuwen and E. Galbrun. Association discovery in two-view data. *IEEE Transactions on Knowledge and Data Engineering*, 27(12):3190–3202, 2015.

[74] M. van Leeuwen and A. Knobbe. Non-redundant subgroup discovery in large and complex data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 459–474. Springer, 2011.

[75] M. van Leeuwen and A. Knobbe. Diverse subgroup set discovery. *Data Mining and Knowledge Discovery*, 25(2):208–242, 2012.

[76] M. van Leeuwen and A. Ukkonen. Discovering skylines of subgroup sets. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 272–287. Springer, 2013.

[77] M. van Leeuwen and A. Ukkonen. Expect the unexpected–on the significance of subgroups. In *International Conference on Discovery Science*, pages 51–66. Springer, 2016.

[78] M. van Leeuwen and J. Vreeken. Mining and using sets of patterns through compression. In *Frequent Pattern Mining*, pages 165–198. Springer, 2014.

[79] D. Leman, A. Feelders, and A. Knobbe. Exceptional model mining. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 1–16. Springer, 2008.

[80] F. Lemmerich, M. Atzmueller, and F. Puppe. Fast exhaustive subgroup discovery with numerical target concepts. *Data Mining and Knowledge Discovery*, 30 (3):711–762, 2016.

[81] B. Letham, C. Rudin, T. H. McCormick, D. Madigan, et al. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, 9(3):1350–1371, 2015.

[82] W. Li, J. Han, and J. Pei. Cmar: Accurate and efficient classification based on multiple class-association rules. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 369–376. IEEE, 2001.

[83] J. Lijffijt, B. Kang, W. Duivesteijn, K. Puolamaki, E. Oikarinen, and T. De Bie. Subjectively interesting subgroup discovery on real-valued targets. In *2018 IEEE ICDE*, pages 1352–1355. IEEE, 2018.

[84] Y. Lou, R. Caruana, and J. Gehrke. Intelligible models for classification and regression. In *Proceedings KDD'12*, pages 150–158. ACM, 2012.

[85] B. L. W. H. Y. Ma and B. Liu. Integrating classification and association rule mining. In *Proceedings of the fourth international conference on knowledge discovery and data mining*, 1998.

[86] T. Makhalova, S. O. Kuznetsov, and A. Napoli. Mint: Mdl-based approach for mining interesting numerical pattern sets. *arXiv preprint arXiv:2011.14843*, 2020.

[87] M. Meeng and A. Knobbe. Flexible enrichment with cortana–software demo. In *Proceedings of BeneLearn*, pages 117–119, 2011.

[88] M. Meeng and A. Knobbe. For real: a thorough look at numeric attributes in subgroup discovery. *Data Mining and Knowledge Discovery*, pages 1–55, 2020.

[89] M. Meeng, H. de Vries, P. Flach, S. Nijssen, and A. Knobbe. Uni-and multivariate probability density models for numeric subgroup discovery. *Intelligent Data Analysis*, 24(6):1403–1439, 2020.

[90] T. Mielikäinen and H. Mannila. The pattern ordering problem. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 327–338. Springer, 2003.

[91] C. Molnar. Interpretable machine learning. *A Guide for Making Black Box Models Explainable*, 2018.

[92] T. Mononen and P. Myllymäki. Computing the multinomial stochastic complexity in sub-linear time. In *Proceedings of the 4th European Workshop on Probabilistic Graphical Models*, pages 209–216, 2008.

[93] I. J. Myung. Tutorial on maximum likelihood estimation. *Journal of mathematical Psychology*, 47(1):90–100, 2003.

[94] E. B. Peterson, K. Neels, N. Barczi, and T. Graham. The economic cost of airline flight delay. *Journal of Transport Economics and Policy (JTEP)*, 47(1):107–121, 2013.

[95] I. Polaka, E. Gašenko, O. Barash, H. Haick, and M. Leja. Constructing interpretable classifiers to diagnose gastric cancer based on breath tests. *Procedia Computer Science*, 104, 2017.

[96] H. M. Proença and M. van Leeuwen. Interpretable multiclass classification by mdl-based rule lists. *Information Sciences*, 512:1372–1393, 2020.

[97] H. M. Proença, S. M. Vieira, U. Kaymak, R. J. Almeida, and J. M. Sousa. Optimizing probabilistic fuzzy systems for classification using metaheuristics. In *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1635–1641. IEEE, 2016.

[98] H. M. Proença, R. Klijn, T. Bäck, and M. van Leeuwen. Identifying flight delay patterns using diverse subgroup discovery. In *2018 IEEE SSCI*, pages 60–67. IEEE, 2018.

[99] H. M. Proença, P. Grünwald, T. Bäck, and M. van Leeuwen. Discovering outstanding subgroup lists for numeric targets using mdl. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 19–35. Springer, 2020.

[100] H. M. Proença, T. Bäck, and M. van Leeuwen. Robust subgroup discovery. *arXiv preprint arXiv:2103.13686*, 2021.

[101] H. M. Proença, T. Bäck, and M. van Leeuwen. Robust subgroup discovery. *Data Mining and Knowledge Discovery (preprint available in arXiv:2103.13686)*, submitted.

[102] F. Provost and P. Domingos. Well-trained pets: Improving probability estimation trees. 2000.

[103] J. R. Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[104] M. T. Ribeiro, S. Singh, and C. Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[105] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[106] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141, 2004.

[107] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5), 1978.

[108] J. Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of statistics*, pages 416–431, 1983.

[109] R. L. Rivest. Learning decision lists. *Machine learning*, 2(3):229–246, 1987.

[110] J. N. Rouder, P. L. Speckman, D. Sun, R. D. Morey, and G. Iverson. Bayesian t tests for accepting and rejecting the null hypothesis. *Psychonomic bulletin & review*, 16(2):225–237, 2009.

[111] C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1 (5):206–215, 2019.

[112] C. M. Salgado, C. Azevedo, H. Proença, and S. M. Vieira. Missing data. *Secondary analysis of electronic health records*, pages 143–162, 2016.

[113] C. M. Salgado, C. Azevedo, H. Proença, and S. M. Vieira. Noise versus outliers. *Secondary analysis of electronic health records*, pages 163–183, 2016.

[114] C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[115] Y. M. Shtar'kov. Universal sequential coding of single messages. *Problemy Peredachi Informatsii*, 23(3):3–17, 1987.

[116] A. Siebes. Data surveying: Foundations of an inductive query language. In *KDD*, pages 269–274, 1995.

[117] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, 12: 2411–2414, 2011.

[118] J. W. Tukey. *Exploratory data analysis*, volume 2. Reading, MA, 1977.

[119] V. Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[120] J. Vreeken, M. van Leeuwen, and A. Siebes. Krimp: mining itemsets that compress. *Data Mining and Knowledge Discovery*, 23(1):169–214, 2011.

[121] J. Wang and G. Karypis. Harmony: Efficiently mining the best rules for classification. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pages 205–216. SIAM, 2005.

[122] T. Wang, C. Rudin, F. Velez-Doshi, Y. Liu, E. Klampfl, and P. MacNeille. Bayesian rule sets for interpretable classification. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 1269–1274. IEEE, 2016.

[123] G. I. Webb. Opus: An efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research*, 3:431–465, 1995.

[124] G. I. Webb. Discovering significant patterns. *Machine Learning*, 68(1):1–33, 2007.

[125] H. Yang, C. Rudin, and M. Seltzer. Scalable bayesian rule lists. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3921–3930. JMLR. org, 2017.

[126] J. Zeng, B. Ustun, and C. Rudin. Interpretable classification models for recidivism prediction. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 180(3), 2017.

[127] X. Zhang, G. Dong, and K. Ramamohanarao. Information-based classification by aggregating emerging patterns. In *IDEAL*, pages 48–53. Springer, 2000.

[128] A. Zimmermann and S. Nijssen. Supervised pattern mining and applications to classification. In *Frequent Pattern Mining*. Springer, 2014.

Samenvatting

Regels bieden een eenvoudige vorm voor het opslaan en delen van informatie over de wereld. Als mensen gebruiken we elke dag regels. Een arts die iemand met griep diagnosticeert, gebruikt bijvoorbeeld de regel: ”als een persoon koorts of keelpijn heeft, dan heeft hij of zij griep”. Hoewel een individuele regel alleen eenvoudige gebeurtenissen kan beschrijven, kunnen verschillende, geaggregeerde regels gezamenlijk complexere scenario’s beschrijven, zoals de volledige set diagnostische regels die (impliciet) door een arts worden gebruikt.

Gezien het veelvuldige gebruik van regels in verschillende domeinen, is het geen verrassing dat op regels gebaseerde modellen tot een van de eerstegebruikte technieken behoorden om computers uit te rusten met besluitvormingsmechanismen. Oorspronkelijk voerden mensen regels rechtstreeks in een computersysteem in; met de beschikbaarheid van grote hoeveelheden data verschoof de interesse naar het leren van regels uit data. De elektronische dossiers van een arts, waarin is vastgelegd of zijn patienten wel of geen griep hebben op basis van hun symptomen, kunnen bijvoorbeeld gebruikt worden om het besluitvormingsproces van de desbetreffende arts te achterhalen.

Het gebruik van regels omvat vele gebieden in de informatica; in dit proefschrift richten we ons op de op regels gebaseerde modellen voor machine learning en datamining. Machine learning is erop gericht om op basis van data het model te leren dat toekomstige (niet eerder waargenomen) gebeurtenissen het beste voorspelt. Datamining heeft als doel om interessante patronen te vinden in de beschikbare data. In het bijzonder richten we ons op de volgende onderzoeksvraag: “Hoe leren we robuuste en interpreteerbare op regels gebaseerde modellen van data voor machine learning en datamining, en definiëren we het optimale model?”

Om deze vraag te beantwoorden gebruiken we het Minimum Description Length (MDL) principe, waarmee we de optimaliteit van op regels gebaseerde modellen voor een bepaalde dataset kunnen bepalen. Informeel is het beste model voor een specifieke dataset het eenvoudigste model dat de data goed beschrijft. De specifieke modelklasse waarop we ons concentreren zijn zogenaamde regellijsten, d.w.z. geordende verzamelingen regels die opeenvolgend worden geïnterpreteerd. Helaas is het vinden van een optimaal model in de meeste gevallen computationeel onhaalbaar. Daarom stellen we heuristische algoritmen voor die goede modellen vinden en daarbij enkele garanties geven. We testen onze algoritmen empirisch om onze aanpak te valideren, en laten zien dat ze in de meeste gevallen beter of vergelijkbaar presteren in vergelijking met de state of the art.

Summary

Rules provide a simple form of storing and sharing information about the world. As humans, we use rules every day, such as the physician that diagnoses someone with flu, represented by "if a person has either a fever or sore throat (among others), then she has the flu.". Even though an individual rule can only describe simple events, several aggregated rules can describe more complex scenarios, such as the complete set of diagnostic rules employed by a physician.

Given their abundant use, it is no surprise that rule-based models were some of the first techniques used to equip computers with decision-making capabilities. In the beginning, humans entered rules directly into computer systems; however, with the availability of large amounts of data, the interest shifted to learning rules from data. For example, the records of a physician's diagnoses of patients who either have flu or not based on their symptoms can be used to learn that doctor's decision-making process. The use of rules spans many fields in computer science, and in this dissertation, we focus on rule-based models for machine learning and data mining. Machine learning focuses on learning from data the model that best predicts future (previously unseen) events. Data mining aims to find interesting patterns in the available data. Specifically, we are concerned with the research question: "How to learn robust and interpretable rule-based models from data for machine learning and data mining, and define their optimality?"

To answer such a question, we employ the Minimum Description Length (MDL) principle, which allows us to define the optimality of rule-based models for a particular dataset. Informally, the best model for a specific dataset is the simplest one that describes the data well. The specific model class we focus on is the rule list, i.e., an ordered set of rules that are interpreted sequentially. Nonetheless, finding an op-

timal model is computationally infeasible in most cases. Thus, we propose heuristic algorithms that find good models with some guarantees. We test our algorithms empirically to validate our approach and show that they achieve, in most cases, better or similar performance to the state-of-the-art in machine learning and data mining.

Resumo

Regras são uma forma simples de armazenar e partilhar informação sobre o mundo. Como seres humanos, usamos regras todos os dias, tal como o médico que diagnostica alguém com gripe, representada da forma “se uma pessoa tem febre ou dor de garganta (entre outras aflições), então ela tem gripe.”. Embora uma regra só possa descrever eventos simples, o agregamento de várias regras pode descrever cenários mais complexos, como o conjunto de todas as regras de diagnóstico usadas por um médico.

Dado seu uso abundante, não é surpresa que os modelos baseados em regras tenham sido umas das primeiras técnicas usadas para equipar computadores com poder de decisão. Inicialmente, as pessoas inseriam as regras diretamente nos sistemas dos computadores; no entanto, com a disponibilidade de grandes quantidades de dados, o interesse voltou-se para aprender regras a partir dos dados. Por exemplo, os relatórios de diagnósticos de um médico sobre se os seus pacientes têm ou não gripe com base nos seus sintomas podem ser usados para aprender o processo de tomada de decisão desse mesmo médico. O uso de regras abrange muitos campos da ciência da computação e, nesta dissertação, focamo-nos em modelos baseados em regras para as áreas de aprendizagem de máquina e prospecção de dados. A aprendizagem de máquina dedica-se a aprender o modelo que melhor prevê eventos futuros (não vistos até então) a partir dos dados. A prospecção de dados visa encontrar padrões interessantes nos dados disponíveis. Mais concretamente, estamos preocupados com a seguinte questão científica: “ Como aprender a partir dos dados modelos baseados em regras que sejam robustos e interpretáveis nas áreas de aprendizagem de máquina e prospeção de dados, e definir a sua proximidade ao óptimo? ”

Para responder a essa questão, empregamos o princípio da Descrição de Comprimento

Mínimo (MDL em inglês), que nos permite definir quão óptimos são os modelos baseados em regras para um conjunto de dados especifico. Informalmente, o melhor modelo para um conjunto de dados específico é o modelo mais simples que melhor descreve os dados. A classe de modelo específica em que nos focamos é chamado lista de regras, ou seja, um conjunto ordenado de regras que têm de ser interpretadas sequencialmente. No entanto, encontrar um modelo óptimo é computacionalmente impossível na maioria dos casos. Por isso, propomos heurísticas que descobrem bons modelos mantendo algumas das garantias originais. Os nossos algoritmos foram empiricamente testados, validando desta forma a nossa abordagem e mostrando, na maioria dos casos, que eles atingem um desempenho melhor ou semelhante ao estado da arte em aprendizagem de máquina e prospeção de dados.

List of publications

1. H. M. Proença, S. M. Vieira, U. Kaymak, R. J. Almeida, and J. M. Sousa. Optimizing probabilistic fuzzy systems for classification using metaheuristics. In *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1635–1641. IEEE, 2016

2. C. M. Salgado, C. Azevedo, H. Proença, and S. M. Vieira. Missing data. *Secondary analysis of electronic health records*, pages 143–162, 2016

3. C. M. Salgado, C. Azevedo, H. Proença, and S. M. Vieira. Noise versus outliers. *Secondary analysis of electronic health records*, pages 163–183, 2016

4. H. M. Proença, R. Klijn, T. Bäck, and M. van Leeuwen. Identifying flight delay patterns using diverse subgroup discovery. In *2018 IEEE SSCI*, pages 60–67. IEEE, 2018

5. H. M. Proença and M. van Leeuwen. Interpretable multiclass classification by mdl-based rule lists. *Information Sciences*, 512:1372–1393, 2020

6. H. M. Proença, P. Grünwald, T. Bäck, and M. van Leeuwen. Discovering outstanding subgroup lists for numeric targets using mdl. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 19–35. Springer, 2020

7. H. M. Proença, T. Bäck, and M. van Leeuwen. Robust subgroup discovery. *Data Mining and Knowledge Discovery (preprint available in arXiv:2103.13686)*, submitted

Acknowledgements

Even though this dissertation only bears one name, it is dedicated to all that I have met on my path. I could not have done it alone; thank you to everyone that in some way contributed to this work.

The first praises go to my supervisor, Matthijs van Leeuwen, whose guidance helped me transverse the sinuous paths of research. You not only inspired me as a researcher, but as a person. I could always count on you, both professionally and personally. You have never pre-emptively judged my choices and I am deeply honoured to have been your student.
Secondly, I would like to thank my promotor, Thomas Bäck. Thank you for accepting an unknown Portuguese student as your PhD student. You always trusted me to make the right decisions and allowed me to pursue my research without impediments. Despite your busy schedule, your door was always open when I needed you.

As a part of an Indian-Dutch collaboration, I was able to hone my research skills with international experts and delve into my fascination of the Indian culture. First, I would like to thank Dhish Saxena, Arioli Arumugam, and Rajesh Alla, for welcoming me so warmly. A special thanks to Divyam and Sarang, that besides my colleagues became close friends. Divyam, thank you for guiding me through Roorkee and Bangalore. Sarang Kapoor, thank you for all the spirited research sparring. Finally, thank you Michael Emmerich for all the fond memories from our trips to India.

Throughout my PhD, I had the pleasure to work with amazing researchers. I would like to thank João Sousa and Susana Viera, for their initial support; Peter Grünwald, for always keeping the door open for my questions; Antti Ukkonen, for the brainstorming sessions; Alexandros Agapitos, whose ambitious ideas were contagious; and

David Lynch, for being the best remote colleague.

LIACS is a special place, where welcoming colleagues swiftly become your friends. I would like to thank everyone that was part of NACO, EDA, and the coffee club. In particular, I would like to thank: Ricardo Cachucho, for advising me even before I arrived in Leiden; Marvin, for always being ready to hear; Jan, for bringing me to the infamous gladiator workout; Irene, my first office mate; Daniela, for sharing the hectic organization of a digital conference; Sander, for the rough start that turned out into a good relationship; Theodoros, my faithful companion of existential PhD discussions; Lise, for our nonsensical chats in Dutch-Portuguese; Peter, for the philosophical discussions; Cláudio, for hosting me; and Lincen, for the MDL discussions.

While abroad, your friends are an intrinsic part of your success, and slowly they become part of your family.
I was lucky to have lived at Zoe, the affectionate name we gave our house, for four years. In it everyone shared meals, laughter, and stories. I would especially like to thank: Esther Weigmann, for becoming such a close friend; Justina Gabrielaityte, for all the laughter and potato dinners you brought; Lorenzo Pasqualleto, for turning an ordinary moment into an unforgettable experience; Andreas Pedersen, for the trips where you got sunburned when I did not; and Léa-Rose, for always bringing banal conversations to the next level. Also, as honorary guests of the house, I would like to thank Bart Bezemer, my tea and chess companion; Neda Malkawi, with whom I share fascination for Thai massages; and Alberto Ferraresso, for judging the parking of bicycles with me.
Depois, há o meu grupo de amigos portugueses em Leiden, que sempre me fizeram sentir em casa. Eu gostaria de agradecer ao: Nuno, por estar lá desde o início; Vasco, porque sempre pude bater à tua porta; Inês, por fazeres sempre um pão extra; Ricardo Castro, pela tua honestidade; João Cunha, por não sabermos conversar por pouco tempo; Cátia, por teres energia para mover o mundo; Paulo, o meu conterrâneo; Tiago Jorge; Tiago Cabrita; Bruno; e Helena Morais.
Finalmente, muchas gracias Laura por ser el apoyo que mejoró mi vida. ¡Gracias por organizar esas copas donde te conocí!

Then there are all the friends that even away, were always present. Thank you Joaquim Viegas, for being the best organizer of any event; Cátia, for inviting me to the swimming pool; Cigin, for our yearly meetings; Miguel Reis, for sharing the same passion for growth; António Coutinho, for all the Bayesian deliberations; Madalina, for sharing your best moments; Daniel Matos, for being always there; and Gonçalo Guiomar, for the long conversations.

Finalmente, quero agradecer o apoio incondicional da minha família: aos meus pais, Maria e Francisco, sem os quais nada teria sido possível; a minha irmã Carolina; a minha sobrinha Malu; aos meus tios Ivone, Angélica, e João; e à memória do meu primo Jorge. Por último, obrigado à minha família nos países baixos Marie-Helene, Christophe, Maxence, and Gautier.

Obrigado!

Titles in the SIKS dissertation series since 2011

2011 01 Botond Cseke (RUN), Variational Algorithms for Bayesian Inference in Latent Gaussian Models

02 Nick Tinnemeier (UU), Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language

03 Jan Martijn van der Werf (TUE), Compositional Design and Verification of Component-Based Information Systems

04 Hado van Hasselt (UU), Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference

05 Bas van der Raadt (VU), Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline.

06 Yiwen Wang (TUE), Semantically-Enhanced Recommendations in Cultural Heritage

07 Yujia Cao (UT), Multimodal Information Presentation for High Load Human Computer Interaction

08 Nieske Vergunst (UU), BDI-based Generation of Robust Task-Oriented Dialogues

09 Tim de Jong (OU), Contextualised Mobile Media for Learning

10 Bart Bogaert (UvT), Cloud Content Contention

11 Dhaval Vyas (UT), Designing for Awareness: An Experience-focused HCI Perspective

12 Carmen Bratosin (TUE), Grid Architecture for Distributed Process Mining

13 Xiaoyu Mao (UvT), Airport under Control. Multiagent Scheduling for Airport Ground Handling

14 Milan Lovric (EUR), Behavioral Finance and Agent-Based Artificial Markets
15 Marijn Koolen (UvA), The Meaning of Structure: the Value of Link Evidence for Information Retrieval
16 Maarten Schadd (UM), Selective Search in Games of Different Complexity
17 Jiyin He (UVA), Exploring Topic Structure: Coherence, Diversity and Relatedness
18 Mark Ponsen (UM), Strategic Decision-Making in complex games
19 Ellen Rusman (OU), The Mind's Eye on Personal Profiles
20 Qing Gu (VU), Guiding service-oriented software engineering - A view-based approach
21 Linda Terlouw (TUD), Modularization and Specification of Service-Oriented Systems
22 Junte Zhang (UVA), System Evaluation of Archival Description and Access
23 Wouter Weerkamp (UVA), Finding People and their Utterances in Social Media
24 Herwin van Welbergen (UT), Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior
25 Syed Waqar ul Qounain Jaffry (VU), Analysis and Validation of Models for Trust Dynamics
26 Matthijs Aart Pontier (VU), Virtual Agents for Human Communication - Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots
27 Aniel Bhulai (VU), Dynamic website optimization through autonomous management of design patterns
28 Rianne Kaptein (UVA), Effective Focused Retrieval by Exploiting Query Context and Document Structure
29 Faisal Kamiran (TUE), Discrimination-aware Classification
30 Egon van den Broek (UT), Affective Signal Processing (ASP): Unraveling the mystery of emotions
31 Ludo Waltman (EUR), Computational and Game-Theoretic Approaches for Modeling Bounded Rationality
32 Nees-Jan van Eck (EUR), Methodological Advances in Bibliometric Mapping of Science
33 Tom van der Weide (UU), Arguing to Motivate Decisions

34 Paolo Turrini (UU), Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations
35 Maaike Harbers (UU), Explaining Agent Behavior in Virtual Training
36 Erik van der Spek (UU), Experiments in serious game design: a cognitive approach
37 Adriana Burlutiu (RUN), Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference
38 Nyree Lemmens (UM), Bee-inspired Distributed Optimization
39 Joost Westra (UU), Organizing Adaptation using Agents in Serious Games
40 Viktor Clerc (VU), Architectural Knowledge Management in Global Software Development
41 Luan Ibraimi (UT), Cryptographically Enforced Distributed Data Access Control
42 Michal Sindlar (UU), Explaining Behavior through Mental State Attribution
43 Henk van der Schuur (UU), Process Improvement through Software Operation Knowledge
44 Boris Reuderink (UT), Robust Brain-Computer Interfaces
45 Herman Stehouwer (UvT), Statistical Language Models for Alternative Sequence Selection
46 Beibei Hu (TUD), Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work
47 Azizi Bin Ab Aziz (VU), Exploring Computational Models for Intelligent Support of Persons with Depression
48 Mark Ter Maat (UT), Response Selection and Turn-taking for a Sensitive Artificial Listening Agent
49 Andreea Niculescu (UT), Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality

2012 01 Terry Kakeeto (UvT), Relationship Marketing for SMEs in Uganda
02 Muhammad Umair (VU), Adaptivity, emotion, and Rationality in Human and Ambient Agent Models
03 Adam Vanya (VU), Supporting Architecture Evolution by Mining Software Repositories
04 Jurriaan Souer (UU), Development of Content Management System-based Web Applications
05 Marijn Plomp (UU), Maturing Interorganisational Information Systems
06 Wolfgang Reinhardt (OU), Awareness Support for Knowledge Workers in Research Networks

07 Rianne van Lambalgen (VU), When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions
08 Gerben de Vries (UVA), Kernel Methods for Vessel Trajectories
09 Ricardo Neisse (UT), Trust and Privacy Management Support for Context-Aware Service Platforms
10 David Smits (TUE), Towards a Generic Distributed Adaptive Hypermedia Environment
11 J.C.B. Rantham Prabhakara (TUE), Process Mining in the Large: Preprocessing, Discovery, and Diagnostics
12 Kees van der Sluijs (TUE), Model Driven Design and Data Integration in Semantic Web Information Systems
13 Suleman Shahid (UvT), Fun and Face: Exploring non-verbal expressions of emotion during playful interactions
14 Evgeny Knutov (TUE), Generic Adaptation Framework for Unifying Adaptive Web-based Systems
15 Natalie van der Wal (VU), Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes.
16 Fiemke Both (VU), Helping people by understanding them - Ambient Agents supporting task execution and depression treatment
17 Amal Elgammal (UvT), Towards a Comprehensive Framework for Business Process Compliance
18 Eltjo Poort (VU), Improving Solution Architecting Practices
19 Helen Schonenberg (TUE), What's Next? Operational Support for Business Process Execution
20 Ali Bahramisharif (RUN), Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing
21 Roberto Cornacchia (TUD), Querying Sparse Matrices for Information Retrieval
22 Thijs Vis (UvT), Intelligence, politie en veiligheidsdienst: verenigbare grootheden?
23 Christian Muehl (UT), Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction
24 Laurens van der Werff (UT), Evaluation of Noisy Transcripts for Spoken Document Retrieval
25 Silja Eckartz (UT), Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application
26 Emile de Maat (UVA), Making Sense of Legal Text

27 Hayrettin Gurkok (UT), Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games
28 Nancy Pascall (UvT), Engendering Technology Empowering Women
29 Almer Tigelaar (UT), Peer-to-Peer Information Retrieval
30 Alina Pommeranz (TUD), Designing Human-Centered Systems for Reflective Decision Making
31 Emily Bagarukayo (RUN), A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure
32 Wietske Visser (TUD), Qualitative multi-criteria preference representation and reasoning
33 Rory Sie (OUN), Coalitions in Cooperation Networks (COCOON)
34 Pavol Jancura (RUN), Evolutionary analysis in PPI networks and applications
35 Evert Haasdijk (VU), Never Too Old To Learn – On-line Evolution of Controllers in Swarm- and Modular Robotics
36 Denis Ssebugwawo (RUN), Analysis and Evaluation of Collaborative Modeling Processes
37 Agnes Nakakawa (RUN), A Collaboration Process for Enterprise Architecture Creation
38 Selmar Smit (VU), Parameter Tuning and Scientific Testing in Evolutionary Algorithms
39 Hassan Fatemi (UT), Risk-aware design of value and coordination networks
40 Agus Gunawan (UvT), Information Access for SMEs in Indonesia
41 Sebastian Kelle (OU), Game Design Patterns for Learning
42 Dominique Verpoorten (OU), Reflection Amplifiers in self-regulated Learning
43 Withdrawn
44 Anna Tordai (VU), On Combining Alignment Techniques
45 Benedikt Kratz (UvT), A Model and Language for Business-aware Transactions
46 Simon Carter (UVA), Exploration and Exploitation of Multilingual Data for Statistical Machine Translation
47 Manos Tsagkias (UVA), Mining Social Media: Tracking Content and Predicting Behavior
48 Jorn Bakker (TUE), Handling Abrupt Changes in Evolving Time-series Data

49 Michael Kaisers (UM), Learning against Learning - Evolutionary dynamics of reinforcement learning algorithms in strategic interactions
50 Steven van Kervel (TUD), Ontologogy driven Enterprise Information Systems Engineering
51 Jeroen de Jong (TUD), Heuristics in Dynamic Sceduling; a practical framework with a case study in elevator dispatching

2013 01 Viorel Milea (EUR), News Analytics for Financial Decision Support
02 Erietta Liarou (CWI), MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing
03 Szymon Klarman (VU), Reasoning with Contexts in Description Logics
04 Chetan Yadati (TUD), Coordinating autonomous planning and scheduling
05 Dulce Pumareja (UT), Groupware Requirements Evolutions Patterns
06 Romulo Goncalves (CWI), The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience
07 Giel van Lankveld (UvT), Quantifying Individual Player Differences
08 Robbert-Jan Merk (VU), Making enemies: cognitive modeling for opponent agents in fighter pilot simulators
09 Fabio Gori (RUN), Metagenomic Data Analysis: Computational Methods and Applications
10 Jeewanie Jayasinghe Arachchige (UvT), A Unified Modeling Framework for Service Design.
11 Evangelos Pournaras (TUD), Multi-level Reconfigurable Self-organization in Overlay Services
12 Marian Razavian (VU), Knowledge-driven Migration to Services
13 Mohammad Safiri (UT), Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly
14 Jafar Tanha (UVA), Ensemble Approaches to Semi-Supervised Learning Learning
15 Daniel Hennes (UM), Multiagent Learning - Dynamic Games and Applications
16 Eric Kok (UU), Exploring the practical benefits of argumentation in multi-agent deliberation
17 Koen Kok (VU), The PowerMatcher: Smart Coordination for the Smart Electricity Grid
18 Jeroen Janssens (UvT), Outlier Selection and One-Class Classification
19 Renze Steenhuizen (TUD), Coordinated Multi-Agent Planning and Scheduling

20 Katja Hofmann (UvA), Fast and Reliable Online Learning to Rank for Information Retrieval
21 Sander Wubben (UvT), Text-to-text generation by monolingual machine translation
22 Tom Claassen (RUN), Causal Discovery and Logic
23 Patricio de Alencar Silva (UvT), Value Activity Monitoring
24 Haitham Bou Ammar (UM), Automated Transfer in Reinforcement Learning
25 Agnieszka Anna Latoszek-Berendsen (UM), Intention-based Decision Support. A new way of representing and implementing clinical guidelines in a Decision Support System
26 Alireza Zarghami (UT), Architectural Support for Dynamic Homecare Service Provisioning
27 Mohammad Huq (UT), Inference-based Framework Managing Data Provenance
28 Frans van der Sluis (UT), When Complexity becomes Interesting: An Inquiry into the Information eXperience
29 Iwan de Kok (UT), Listening Heads
30 Joyce Nakatumba (TUE), Resource-Aware Business Process Management: Analysis and Support
31 Dinh Khoa Nguyen (UvT), Blueprint Model and Language for Engineering Cloud Applications
32 Kamakshi Rajagopal (OUN), Networking For Learning; The role of Networking in a Lifelong Learner's Professional Development
33 Qi Gao (TUD), User Modeling and Personalization in the Microblogging Sphere
34 Kien Tjin-Kam-Jet (UT), Distributed Deep Web Search
35 Abdallah El Ali (UvA), Minimal Mobile Human Computer Interaction
36 Than Lam Hoang (TUe), Pattern Mining in Data Streams
37 Dirk Börner (OUN), Ambient Learning Displays
38 Eelco den Heijer (VU), Autonomous Evolutionary Art
39 Joop de Jong (TUD), A Method for Enterprise Ontology based Design of Enterprise Information Systems
40 Pim Nijssen (UM), Monte-Carlo Tree Search for Multi-Player Games
41 Jochem Liem (UVA), Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning
42 Léon Planken (TUD), Algorithms for Simple Temporal Reasoning

43 Marc Bron (UVA), Exploration and Contextualization through Interaction and Concepts

2014 01 Nicola Barile (UU), Studies in Learning Monotone Models from Data
02 Fiona Tuliyano (RUN), Combining System Dynamics with a Domain Modeling Method
03 Sergio Raul Duarte Torres (UT), Information Retrieval for Children: Search Behavior and Solutions
04 Hanna Jochmann-Mannak (UT), Websites for children: search strategies and interface design - Three studies on children's search performance and evaluation
05 Jurriaan van Reijsen (UU), Knowledge Perspectives on Advancing Dynamic Capability
06 Damian Tamburri (VU), Supporting Networked Software Development
07 Arya Adriansyah (TUE), Aligning Observed and Modeled Behavior
08 Samur Araujo (TUD), Data Integration over Distributed and Heterogeneous Data Endpoints
09 Philip Jackson (UvT), Toward Human-Level Artificial Intelligence: Representation and Computation of Meaning in Natural Language
10 Ivan Salvador Razo Zapata (VU), Service Value Networks
11 Janneke van der Zwaan (TUD), An Empathic Virtual Buddy for Social Support
12 Willem van Willigen (VU), Look Ma, No Hands: Aspects of Autonomous Vehicle Control
13 Arlette van Wissen (VU), Agent-Based Support for Behavior Change: Models and Applications in Health and Safety Domains
14 Yangyang Shi (TUD), Language Models With Meta-information
15 Natalya Mogles (VU), Agent-Based Analysis and Support of Human Functioning in Complex Socio-Technical Systems: Applications in Safety and Healthcare
16 Krystyna Milian (VU), Supporting trial recruitment and design by automatically interpreting eligibility criteria
17 Kathrin Dentler (VU), Computing healthcare quality indicators automatically: Secondary Use of Patient Data and Semantic Interoperability
18 Mattijs Ghijsen (UVA), Methods and Models for the Design and Study of Dynamic Agent Organizations
19 Vinicius Ramos (TUE), Adaptive Hypermedia Courses: Qualitative and Quantitative Evaluation and Tool Support
20 Mena Habib (UT), Named Entity Extraction and Disambiguation for Informal Text: The Missing Link

21 Kassidy Clark (TUD), Negotiation and Monitoring in Open Environments
22 Marieke Peeters (UU), Personalized Educational Games - Developing agent-supported scenario-based training
23 Eleftherios Sidirourgos (UvA/CWI), Space Efficient Indexes for the Big Data Era
24 Davide Ceolin (VU), Trusting Semi-structured Web Data
25 Martijn Lappenschaar (RUN), New network models for the analysis of disease interaction
26 Tim Baarslag (TUD), What to Bid and When to Stop
27 Rui Jorge Almeida (EUR), Conditional Density Models Integrating Fuzzy and Probabilistic Representations of Uncertainty
28 Anna Chmielowiec (VU), Decentralized k-Clique Matching
29 Jaap Kabbedijk (UU), Variability in Multi-Tenant Enterprise Software
30 Peter de Cock (UvT), Anticipating Criminal Behaviour
31 Leo van Moergestel (UU), Agent Technology in Agile Multiparallel Manufacturing and Product Support
32 Naser Ayat (UvA), On Entity Resolution in Probabilistic Data
33 Tesfa Tegegne (RUN), Service Discovery in eHealth
34 Christina Manteli (VU), The Effect of Governance in Global Software Development: Analyzing Transactive Memory Systems.
35 Joost van Ooijen (UU), Cognitive Agents in Virtual Worlds: A Middleware Design Approach
36 Joos Buijs (TUE), Flexible Evolutionary Algorithms for Mining Structured Process Models
37 Maral Dadvar (UT), Experts and Machines United Against Cyberbullying
38 Danny Plass-Oude Bos (UT), Making brain-computer interfaces better: improving usability through post-processing.
39 Jasmina Maric (UvT), Web Communities, Immigration, and Social Capital
40 Walter Omona (RUN), A Framework for Knowledge Management Using ICT in Higher Education
41 Frederic Hogenboom (EUR), Automated Detection of Financial Events in News Text
42 Carsten Eijckhof (CWI/TUD), Contextual Multidimensional Relevance Models
43 Kevin Vlaanderen (UU), Supporting Process Improvement using Method Increments
44 Paulien Meesters (UvT), Intelligent Blauw. Met als ondertitel: Intelligence-gestuurde politiezorg in gebiedsgebonden eenheden.

45 Birgit Schmitz (OUN), Mobile Games for Learning: A Pattern-Based Approach
46 Ke Tao (TUD), Social Web Data Analytics: Relevance, Redundancy, Diversity
47 Shangsong Liang (UVA), Fusion and Diversification in Information Retrieval

---

2015 01 Niels Netten (UvA), Machine Learning for Relevance of Information in Crisis Response
02 Faiza Bukhsh (UvT), Smart auditing: Innovative Compliance Checking in Customs Controls
03 Twan van Laarhoven (RUN), Machine learning for network data
04 Howard Spoelstra (OUN), Collaborations in Open Learning Environments
05 Christoph Bösch (UT), Cryptographically Enforced Search Pattern Hiding
06 Farideh Heidari (TUD), Business Process Quality Computation - Computing Non-Functional Requirements to Improve Business Processes
07 Maria-Hendrike Peetz (UvA), Time-Aware Online Reputation Analysis
08 Jie Jiang (TUD), Organizational Compliance: An agent-based model for designing and evaluating organizational interactions
09 Randy Klaassen (UT), HCI Perspectives on Behavior Change Support Systems
10 Henry Hermans (OUN), OpenU: design of an integrated system to support lifelong learning
11 Yongming Luo (TUE), Designing algorithms for big graph datasets: A study of computing bisimulation and joins
12 Julie M. Birkholz (VU), Modi Operandi of Social Network Dynamics: The Effect of Context on Scientific Collaboration Networks
13 Giuseppe Procaccianti (VU), Energy-Efficient Software
14 Bart van Straalen (UT), A cognitive approach to modeling bad news conversations
15 Klaas Andries de Graaf (VU), Ontology-based Software Architecture Documentation
16 Changyun Wei (UT), Cognitive Coordination for Cooperative Multi-Robot Teamwork
17 André van Cleeff (UT), Physical and Digital Security Mechanisms: Properties, Combinations and Trade-offs
18 Holger Pirk (CWI), Waste Not, Want Not! - Managing Relational Data in Asymmetric Memories
19 Bernardo Tabuenca (OUN), Ubiquitous Technology for Lifelong Learners

20 Lois Vanhée (UU), Using Culture and Values to Support Flexible Coordination
21 Sibren Fetter (OUN), Using Peer-Support to Expand and Stabilize Online Learning
22 Zhemin Zhu (UT), Co-occurrence Rate Networks
23 Luit Gazendam (VU), Cataloguer Support in Cultural Heritage
24 Richard Berendsen (UVA), Finding People, Papers, and Posts: Vertical Search Algorithms and Evaluation
25 Steven Woudenberg (UU), Bayesian Tools for Early Disease Detection
26 Alexander Hogenboom (EUR), Sentiment Analysis of Text Guided by Semantics and Structure
27 Sándor Héman (CWI), Updating compressed colomn stores
28 Janet Bagorogoza (TiU), Knowledge Management and High Performance; The Uganda Financial Institutions Model for HPO
29 Hendrik Baier (UM), Monte-Carlo Tree Search Enhancements for One-Player and Two-Player Domains
30 Kiavash Bahreini (OU), Real-time Multimodal Emotion Recognition in E-Learning
31 Yakup Koç (TUD), On the robustness of Power Grids
32 Jerome Gard (UL), Corporate Venture Management in SMEs
33 Frederik Schadd (TUD), Ontology Mapping with Auxiliary Resources
34 Victor de Graaf (UT), Gesocial Recommender Systems
35 Jungxao Xu (TUD), Affective Body Language of Humanoid Robots: Perception and Effects in Human Robot Interaction

---

2016 01 Syed Saiden Abbas (RUN), Recognition of Shapes by Humans and Machines
02 Michiel Christiaan Meulendijk (UU), Optimizing medication reviews through decision support: prescribing a better pill to swallow
03 Maya Sappelli (RUN), Knowledge Work in Context: User Centered Knowledge Worker Support
04 Laurens Rietveld (VU), Publishing and Consuming Linked Data
05 Evgeny Sherkhonov (UVA), Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers
06 Michel Wilson (TUD), Robust scheduling in an uncertain environment
07 Jeroen de Man (VU), Measuring and modeling negative emotions for virtual training
08 Matje van de Camp (TiU), A Link to the Past: Constructing Historical Social Networks from Unstructured Data

09 Archana Nottamkandath (VU), Trusting Crowdsourced Information on Cultural Artefacts
10 George Karafotias (VUA), Parameter Control for Evolutionary Algorithms
11 Anne Schuth (UVA), Search Engines that Learn from Their Users
12 Max Knobbout (UU), Logics for Modelling and Verifying Normative Multi-Agent Systems
13 Nana Baah Gyan (VU), The Web, Speech Technologies and Rural Development in West Africa - An ICT4D Approach
14 Ravi Khadka (UU), Revisiting Legacy Software System Modernization
15 Steffen Michels (RUN), Hybrid Probabilistic Logics - Theoretical Aspects, Algorithms and Experiments
16 Guangliang Li (UVA), Socially Intelligent Autonomous Agents that Learn from Human Reward
17 Berend Weel (VU), Towards Embodied Evolution of Robot Organisms
18 Albert Meroño Peñuela (VU), Refining Statistical Data on the Web
19 Julia Efremova (Tu/e), Mining Social Structures from Genealogical Data
20 Daan Odijk (UVA), Context & Semantics in News & Web Search
21 Alejandro Moreno Célleri (UT), From Traditional to Interactive Playspaces: Automatic Analysis of Player Behavior in the Interactive Tag Playground
22 Grace Lewis (VU), Software Architecture Strategies for Cyber-Foraging Systems
23 Fei Cai (UVA), Query Auto Completion in Information Retrieval
24 Brend Wanders (UT), Repurposing and Probabilistic Integration of Data; An Iterative and data model independent approach
25 Julia Kiseleva (TU/e), Using Contextual Information to Understand Searching and Browsing Behavior
26 Dilhan Thilakarathne (VU), In or Out of Control: Exploring Computational Models to Study the Role of Human Awareness and Control in Behavioural Choices, with Applications in Aviation and Energy Management Domains
27 Wen Li (TUD), Understanding Geo-spatial Information on Social Media
28 Mingxin Zhang (TUD), Large-scale Agent-based Social Simulation - A study on epidemic prediction and control
29 Nicolas Höning (TUD), Peak reduction in decentralised electricity systems - Markets and prices for flexible planning
30 Ruud Mattheij (UvT), The Eyes Have It
31 Mohammad Khelghati (UT), Deep web content monitoring

32 Eelco Vriezekolk (UT), Assessing Telecommunication Service Availability Risks for Crisis Organisations
33 Peter Bloem (UVA), Single Sample Statistics, exercises in learning from just one example
34 Dennis Schunselaar (TUE), Configurable Process Trees: Elicitation, Analysis, and Enactment
35 Zhaochun Ren (UVA), Monitoring Social Media: Summarization, Classification and Recommendation
36 Daphne Karreman (UT), Beyond R2D2: The design of nonverbal interaction behavior optimized for robot-specific morphologies
37 Giovanni Sileno (UvA), Aligning Law and Action - a conceptual and computational inquiry
38 Andrea Minuto (UT), Materials that Matter - Smart Materials meet Art & Interaction Design
39 Merijn Bruijnes (UT), Believable Suspect Agents; Response and Interpersonal Style Selection for an Artificial Suspect
40 Christian Detweiler (TUD), Accounting for Values in Design
41 Thomas King (TUD), Governing Governance: A Formal Framework for Analysing Institutional Design and Enactment Governance
42 Spyros Martzoukos (UVA), Combinatorial and Compositional Aspects of Bilingual Aligned Corpora
43 Saskia Koldijk (RUN), Context-Aware Support for Stress Self-Management: From Theory to Practice
44 Thibault Sellam (UVA), Automatic Assistants for Database Exploration
45 Bram van de Laar (UT), Experiencing Brain-Computer Interface Control
46 Jorge Gallego Perez (UT), Robots to Make you Happy
47 Christina Weber (UL), Real-time foresight - Preparedness for dynamic innovation networks
48 Tanja Buttler (TUD), Collecting Lessons Learned
49 Gleb Polevoy (TUD), Participation and Interaction in Projects. A Game-Theoretic Analysis
50 Yan Wang (UVT), The Bridge of Dreams: Towards a Method for Operational Performance Alignment in IT-enabled Service Supply Chains

---

2017 01 Jan-Jaap Oerlemans (UL), Investigating Cybercrime
02 Sjoerd Timmer (UU), Designing and Understanding Forensic Bayesian Networks using Argumentation
03 Daniël Harold Telgen (UU), Grid Manufacturing; A Cyber-Physical Approach with Autonomous Products and Reconfigurable Manufacturing Machines

04 Mrunal Gawade (CWI), Multi-core Parallelism in a Column-store
05 Mahdieh Shadi (UVA), Collaboration Behavior
06 Damir Vandic (EUR), Intelligent Information Systems for Web Product Search
07 Roel Bertens (UU), Insight in Information: from Abstract to Anomaly
08 Rob Konijn (VU) , Detecting Interesting Differences:Data Mining in Health Insurance Data using Outlier Detection and Subgroup Discovery
09 Dong Nguyen (UT), Text as Social and Cultural Data: A Computational Perspective on Variation in Text
10 Robby van Delden (UT), (Steering) Interactive Play Behavior
11 Florian Kunneman (RUN), Modelling patterns of time and emotion in Twitter #anticipointment
12 Sander Leemans (TUE), Robust Process Mining with Guarantees
13 Gijs Huisman (UT), Social Touch Technology - Extending the reach of social touch through haptic technology
14 Shoshannah Tekofsky (UvT), You Are Who You Play You Are: Modelling Player Traits from Video Game Behavior
15 Peter Berck (RUN), Memory-Based Text Correction
16 Aleksandr Chuklin (UVA), Understanding and Modeling Users of Modern Search Engines
17 Daniel Dimov (UL), Crowdsourced Online Dispute Resolution
18 Ridho Reinanda (UVA), Entity Associations for Search
19 Jeroen Vuurens (UT), Proximity of Terms, Texts and Semantic Vectors in Information Retrieval
20 Mohammadbashir Sedighi (TUD), Fostering Engagement in Knowledge Sharing: The Role of Perceived Benefits, Costs and Visibility
21 Jeroen Linssen (UT), Meta Matters in Interactive Storytelling and Serious Gaming (A Play on Worlds)
22 Sara Magliacane (VU), Logics for causal inference under uncertainty
23 David Graus (UVA), Entities of Interest — Discovery in Digital Traces
24 Chang Wang (TUD), Use of Affordances for Efficient Robot Learning
25 Veruska Zamborlini (VU), Knowledge Representation for Clinical Guidelines, with applications to Multimorbidity Analysis and Literature Search
26 Merel Jung (UT), Socially intelligent robots that understand and respond to human touch
27 Michiel Joosse (UT), Investigating Positioning and Gaze Behaviors of Social Robots: People's Preferences, Perceptions and Behaviors
28 John Klein (VU), Architecture Practices for Complex Contexts

29 Adel Alhuraibi (UvT), From IT-BusinessStrategic Alignment to Performance: A Moderated Mediation Model of Social Innovation, and Enterprise Governance of IT"
30 Wilma Latuny (UvT), The Power of Facial Expressions
31 Ben Ruijl (UL), Advances in computational methods for QFT calculations
32 Thaer Samar (RUN), Access to and Retrievability of Content in Web Archives
33 Brigit van Loggem (OU), Towards a Design Rationale for Software Documentation: A Model of Computer-Mediated Activity
34 Maren Scheffel (OU), The Evaluation Framework for Learning Analytics
35 Martine de Vos (VU), Interpreting natural science spreadsheets
36 Yuanhao Guo (UL), Shape Analysis for Phenotype Characterisation from High-throughput Imaging
37 Alejandro Montes Garcia (TUE), WiBAF: A Within Browser Adaptation Framework that Enables Control over Privacy
38 Alex Kayal (TUD), Normative Social Applications
39 Sara Ahmadi (RUN), Exploiting properties of the human auditory system and compressive sensing methods to increase noise robustness in ASR
40 Altaf Hussain Abro (VUA), Steer your Mind: Computational Exploration of Human Control in Relation to Emotions, Desires and Social Support For applications in human-aware support systems
41 Adnan Manzoor (VUA), Minding a Healthy Lifestyle: An Exploration of Mental Processes and a Smart Environment to Provide Support for a Healthy Lifestyle
42 Elena Sokolova (RUN), Causal discovery from mixed and missing data with applications on ADHD datasets
43 Maaike de Boer (RUN), Semantic Mapping in Video Retrieval
44 Garm Lucassen (UU), Understanding User Stories - Computational Linguistics in Agile Requirements Engineering
45 Bas Testerink (UU), Decentralized Runtime Norm Enforcement
46 Jan Schneider (OU), Sensor-based Learning Support
47 Jie Yang (TUD), Crowd Knowledge Creation Acceleration
48 Angel Suarez (OU), Collaborative inquiry-based learning

2018 01 Han van der Aa (VUA), Comparing and Aligning Process Representations
02 Felix Mannhardt (TUE), Multi-perspective Process Mining
03 Steven Bosems (UT), Causal Models For Well-Being: Knowledge Modeling, Model-Driven Development of Context-Aware Applications, and Behavior Prediction

04 Jordan Janeiro (TUD), Flexible Coordination Support for Diagnosis Teams in Data-Centric Engineering Tasks
05 Hugo Huurdeman (UVA), Supporting the Complex Dynamics of the Information Seeking Process
06 Dan Ionita (UT), Model-Driven Information Security Risk Assessment of Socio-Technical Systems
07 Jieting Luo (UU), A formal account of opportunism in multi-agent systems
08 Rick Smetsers (RUN), Advances in Model Learning for Software Systems
09 Xu Xie (TUD), Data Assimilation in Discrete Event Simulations
10 Julienka Mollee (VUA), Moving forward: supporting physical activity behavior change through intelligent technology
11 Mahdi Sargolzaei (UVA), Enabling Framework for Service-oriented Collaborative Networks
12 Xixi Lu (TUE), Using behavioral context in process mining
13 Seyed Amin Tabatabaei (VUA), Computing a Sustainable Future
14 Bart Joosten (UVT), Detecting Social Signals with Spatiotemporal Gabor Filters
15 Naser Davarzani (UM), Biomarker discovery in heart failure
16 Jaebok Kim (UT), Automatic recognition of engagement and emotion in a group of children
17 Jianpeng Zhang (TUE), On Graph Sample Clustering
18 Henriette Nakad (UL), De Notaris en Private Rechtspraak
19 Minh Duc Pham (VUA), Emergent relational schemas for RDF
20 Manxia Liu (RUN), Time and Bayesian Networks
21 Aad Slootmaker (OUN), EMERGO: a generic platform for authoring and playing scenario-based serious games
22 Eric Fernandes de Mello Araujo (VUA), Contagious: Modeling the Spread of Behaviours, Perceptions and Emotions in Social Networks
23 Kim Schouten (EUR), Semantics-driven Aspect-Based Sentiment Analysis
24 Jered Vroon (UT), Responsive Social Positioning Behaviour for Semi-Autonomous Telepresence Robots
25 Riste Gligorov (VUA), Serious Games in Audio-Visual Collections
26 Roelof Anne Jelle de Vries (UT),Theory-Based and Tailor-Made: Motivational Messages for Behavior Change Technology
27 Maikel Leemans (TUE), Hierarchical Process Mining for Scalable Software Analysis
28 Christian Willemse (UT), Social Touch Technologies: How they feel and how they make you feel

29 Yu Gu (UVT), Emotion Recognition from Mandarin Speech
30 Wouter Beek, The ”K” in ”semantic web” stands for ”knowledge”: scaling semantics to the web

2019 01 Rob van Eijk (UL),Web privacy measurement in real-time bidding systems. A graph-based approach to RTB system classification
02 Emmanuelle Beauxis Aussalet (CWI, UU), Statistics and Visualizations for Assessing Class Size Uncertainty
03 Eduardo Gonzalez Lopez de Murillas (TUE), Process Mining on Databases: Extracting Event Data from Real Life Data Sources
04 Ridho Rahmadi (RUN), Finding stable causal structures from clinical data
05 Sebastiaan van Zelst (TUE), Process Mining with Streaming Data
06 Chris Dijkshoorn (VU), Nichesourcing for Improving Access to Linked Cultural Heritage Datasets
07 Soude Fazeli (TUD), Recommender Systems in Social Learning Platforms
08 Frits de Nijs (TUD), Resource-constrained Multi-agent Markov Decision Processes
09 Fahimeh Alizadeh Moghaddam (UVA), Self-adaptation for energy efficiency in software systems
10 Qing Chuan Ye (EUR), Multi-objective Optimization Methods for Allocation and Prediction
11 Yue Zhao (TUD), Learning Analytics Technology to Understand Learner Behavioral Engagement in MOOCs
12 Jacqueline Heinerman (VU), Better Together
13 Guanliang Chen (TUD), MOOC Analytics: Learner Modeling and Content Generation
14 Daniel Davis (TUD), Large-Scale Learning Analytics: Modeling Learner Behavior & Improving Learning Outcomes in Massive Open Online Courses
15 Erwin Walraven (TUD), Planning under Uncertainty in Constrained and Partially Observable Environments
16 Guangming Li (TUE), Process Mining based on Object-Centric Behavioral Constraint (OCBC) Models
17 Ali Hurriyetoglu (RUN),Extracting actionable information from microtexts
18 Gerard Wagenaar (UU), Artefacts in Agile Team Communication
19 Vincent Koeman (TUD), Tools for Developing Cognitive Agents
20 Chide Groenouwe (UU), Fostering technically augmented human collective intelligence

21 Cong Liu (TUE), Software Data Analytics: Architectural Model Discovery and Design Pattern Detection
22 Martin van den Berg (VU),Improving IT Decisions with Enterprise Architecture
23 Qin Liu (TUD), Intelligent Control Systems: Learning, Interpreting, Verification
24 Anca Dumitrache (VU), Truth in Disagreement - Crowdsourcing Labeled Data for Natural Language Processing
25 Emiel van Miltenburg (VU), Pragmatic factors in (automatic) image description
26 Prince Singh (UT), An Integration Platform for Synchromodal Transport
27 Alessandra Antonaci (OUN), The Gamification Design Process applied to (Massive) Open Online Courses
28 Esther Kuindersma (UL), Cleared for take-off: Game-based learning to prepare airline pilots for critical situations
29 Daniel Formolo (VU), Using virtual agents for simulation and training of social skills in safety-critical circumstances
30 Vahid Yazdanpanah (UT), Multiagent Industrial Symbiosis Systems
31 Milan Jelisavcic (VU), Alive and Kicking: Baby Steps in Robotics
32 Chiara Sironi (UM), Monte-Carlo Tree Search for Artificial General Intelligence in Games
33 Anil Yaman (TUE), Evolution of Biologically Inspired Learning in Artificial Neural Networks
34 Negar Ahmadi (TUE), EEG Microstate and Functional Brain Network Features for Classification of Epilepsy and PNES
35 Lisa Facey-Shaw (OUN), Gamification with digital badges in learning programming
36 Kevin Ackermans (OUN), Designing Video-Enhanced Rubrics to Master Complex Skills
37 Jian Fang (TUD), Database Acceleration on FPGAs
38 Akos Kadar (OUN), Learning visually grounded and multilingual representations

---

2020 01 Armon Toubman (UL), Calculated Moves: Generating Air Combat Behaviour
02 Marcos de Paula Bueno (UL), Unraveling Temporal Processes using Probabilistic Graphical Models
03 Mostafa Deghani (UvA), Learning with Imperfect Supervision for Language Understanding
04 Maarten van Gompel (RUN), Context as Linguistic Bridges

05 Yulong Pei (TUE), On local and global structure mining
06 Preethu Rose Anish (UT), Stimulation Architectural Thinking during Requirements Elicitation - An Approach and Tool Support
07 Wim van der Vegt (OUN), Towards a software architecture for reusable game components
08 Ali Mirsoleimani (UL),Structured Parallel Programming for Monte Carlo Tree Search
09 Myriam Traub (UU), Measuring Tool Bias and Improving Data Quality for Digital Humanities Research
10 Alifah Syamsiyah (TUE), In-database Preprocessing for Process Mining
11 Sepideh Mesbah (TUD), Semantic-Enhanced Training Data AugmentationMethods for Long-Tail Entity Recognition Models
12 Ward van Breda (VU), Predictive Modeling in E-Mental Health: Exploring Applicability in Personalised Depression Treatment
13 Marco Virgolin (CWI), Design and Application of Gene-pool Optimal Mixing Evolutionary Algorithms for Genetic Programming
14 Mark Raasveldt (CWI/UL), Integrating Analytics with Relational Databases
15 Konstantinos Georgiadis (OUN), Smart CAT: Machine Learning for Configurable Assessments in Serious Games
16 Ilona Wilmont (RUN), Cognitive Aspects of Conceptual Modelling
17 Daniele Di Mitri (OUN), The Multimodal Tutor: Adaptive Feedback from Multimodal Experiences
18 Georgios Methenitis (TUD), Agent Interactions & Mechanisms in Markets with Uncertainties: Electricity Markets in Renewable Energy Systems
19 Guido van Capelleveen (UT), Industrial Symbiosis Recommender Systems
20 Albert Hankel (VU), Embedding Green ICT Maturity in Organisations
21 Karine da Silva Miras de Araujo (VU), Where is the robot?: Life as it could be
22 Maryam Masoud Khamis (RUN), Understanding complex systems implementation through a modeling approach: the case of e-government in Zanzibar
23 Rianne Conijn (UT), The Keys to Writing: A writing analytics approach to studying writing processes using keystroke logging
24 Lenin da Nobrega Medeiros (VUA/RUN), How are you feeling, human? Towards emotionally supportive chatbots
25 Xin Du (TUE), The Uncertainty in Exceptional Model Mining

26 Krzysztof Leszek Sadowski (UU), GAMBIT: Genetic Algorithm for Model-Based mixed-Integer opTimization
27 Ekaterina Muravyeva (TUD), Personal data and informed consent in an educational context
28 Bibeg Limbu (TUD), Multimodal interaction for deliberate practice: Training complex skills with augmented reality
29 Ioan Gabriel Bucur (RUN), Being Bayesian about Causal Inference
30 Bob Zadok Blok (UL), Creatief, Creatieve, Creatiefst
31 Gongjin Lan (VU), Learning better – From Baby to Better
32 Jason Rhuggenaath (TUE), Revenue management in online markets: pricing and online advertising
33 Rick Gilsing (TUE), Supporting service-dominant business model evaluation in the context of business model innovation
34 Anna Bon (MU), Intervention or Collaboration? Redesigning Information and Communication Technologies for Development
35 Siamak Farshidi (UU), Multi-Criteria Decision-Making in Software Production

---

2021 01 Francisco Xavier Dos Santos Fonseca (TUD),Location-based Games for Social Interaction in Public Space
02 Rijk Mercuur (TUD), Simulating Human Routines:Integrating Social Practice Theory in Agent-Based Models
03 Seyyed Hadi Hashemi (UVA), Modeling Users Interacting with Smart Devices
04 Ioana Jivet (OU), The Dashboard That Loved Me: Designing adaptive learning analytics for self-regulated learning
05 Davide Dell'Anna (UU), Data-Driven Supervision of Autonomous Systems
06 Daniel Davison (UT), "Hey robot, what do you think?" How children learn with a social robot
07 Armel Lefebvre (UU), Research data management for open science
08 Nardie Fanchamps (OU), The Influence of Sense-Reason-Act Programming on Computational Thinking
09 Cristina Zaga (UT), The Design of Robothings. Non-Anthropomorphic and Non-Verbal Robots to Promote Children's Collaboration Through Play
10 Quinten Meertens (UvA), Misclassification Bias in Statistical Learning
11 Anne van Rossum (UL), Nonparametric Bayesian Methods in Robotic Vision
12 Lei Pi (UL), External Knowledge Absorption in Chinese SMEs
13 Bob R. Schadenberg (UT), Robots for Autistic Children: Understanding and Facilitating Predictability for Engagement in Learning

14 Negin Samaeemofrad (UL), Business Incubators: The Impact of Their Support
15 Onat Ege Adali (TU/e), Transformation of Value Propositions into Resource Re-Configurations through the Business Services Paradigm
16 Esam A. H. Ghaleb (UM), BIMODAL EMOTION RECOGNITION FROM AUDIO-VISUAL CUES
17 Dario Dotti (UM), Human Behavior Understanding from motion and bodily cues using deep neural networks
18 Remi Wieten (UU), Bridging the Gap Between Informal Sense-Making Tools and Formal Systems - Facilitating the Construction of Bayesian Networks and Argumentation Frameworks
19 Roberto Verdecchia (VU), Architectural Technical Debt: Identification and Management
20 Masoud Mansoury (TU/e), Understanding and Mitigating Multi-Sided Exposure Bias in Recommender Systems
21 Pedro Thiago Timbó Holanda (CWI), Progressive Indexes

Curriculum Vitae

Hugo Manuel Proença was born on January 21^{st}, 1990, in Hong Kong. Then, he lived his first two years in Macau, after which he went to Lisbon, Portugal. Hugo conducted the first two years of high school (2004-2006) at Liceu Maria Amália Vaz de Carvalho in Lisbon, after which he finalized the last year at Escola Secundária Campos Melo, Covilhã. After that, he returned to Lisbon to join Instituto Superior Técnico, Universidade de Lisboa. There, he obtained a Bachelor degree in Physics Engineering in 2011, and a Master degree in Mechanical Engineering, specializing in Systems Engineering, in 2015. His Mater thesis, titled "Optimizing Probabilistic Fuzzy Systems For Classification" was supervised by João Sousa, and with the support of Susana Vieira, Rui Almeida, and Uzay Kaymak. During the time of his Master thesis, Hugo conducted an internship as a Mechanical Engineer at Mettler Toledo in Changzhou, China.
Soon after his graduation, he performed a data science internship at the Cybernetics Department of the Czech Technical University in Prague. Then, he returned to Lisbon, where he worked with Susana Vieira, and Cátia Salgado, on two chapters of the MIT critical data book "Secondary Analysis of Electronic Health Records".
In June 2016 he started his PhD in Computer Science at Leiden University as a part of the SAPPAO project and joined both the Natural Computing and the Explanatory Data Analysis group, under the supervision of Thomas Bäck and Matthijs van Leeuwen. The project was a Dutch-Indian collaboration which involved a close collaboration with Dhish Saxena, Sarang Kapoor, and Divyam Aggarwal, at IIT Roorkee, and Rajesh Alla, and Arioli Arumugam, at GE Global Research in Bangalore. Here, he specialized on developing interpretable and statistically robust machine learning and data mining methods. From October 2020, for a period of six months, he was a researcher at Huawei Ireland Research Center.