Robust rules for prediction and description

Manuel Proenca, H.

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.

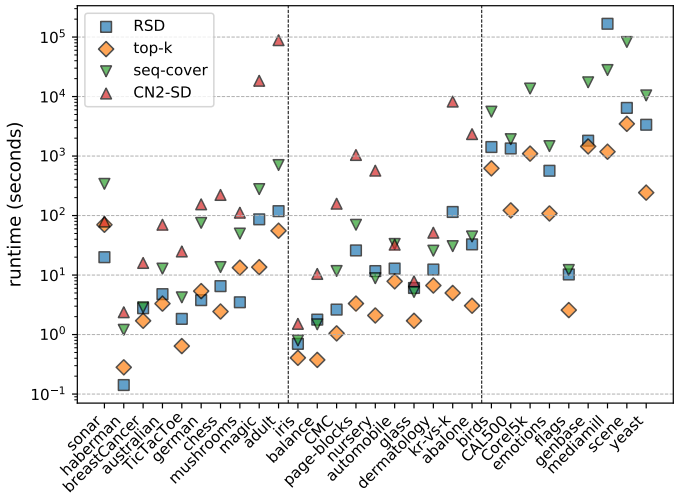
5.3.4 Numeric target results

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.

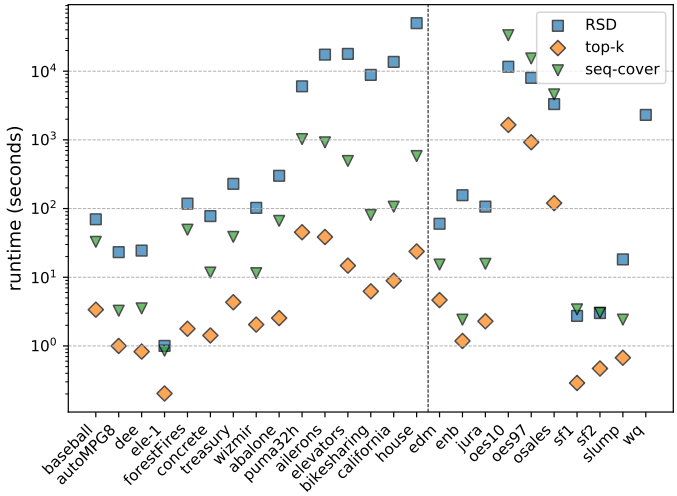
Comparing SWKL results for top-k with seq-cover and RSD shows that irrespective of dispersion-aware (RSD) or not (seq-cover), covering different regions of the data increases the quality of the list in terms of SWKL, validating the use of our measure. It should be noted that both top-k and seq-cover 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.

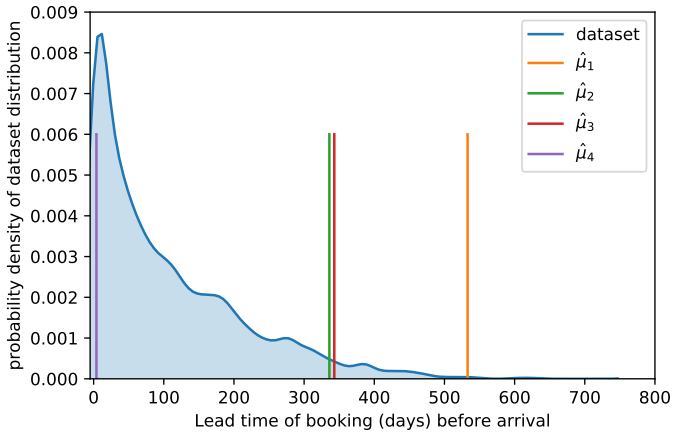


Figure 5.4: *Hotel bookings* kernel density estimation of the dataset distribution and location (not density) of the mean value of the first 4 subgroups obtained with RSD and normalized gain.

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-

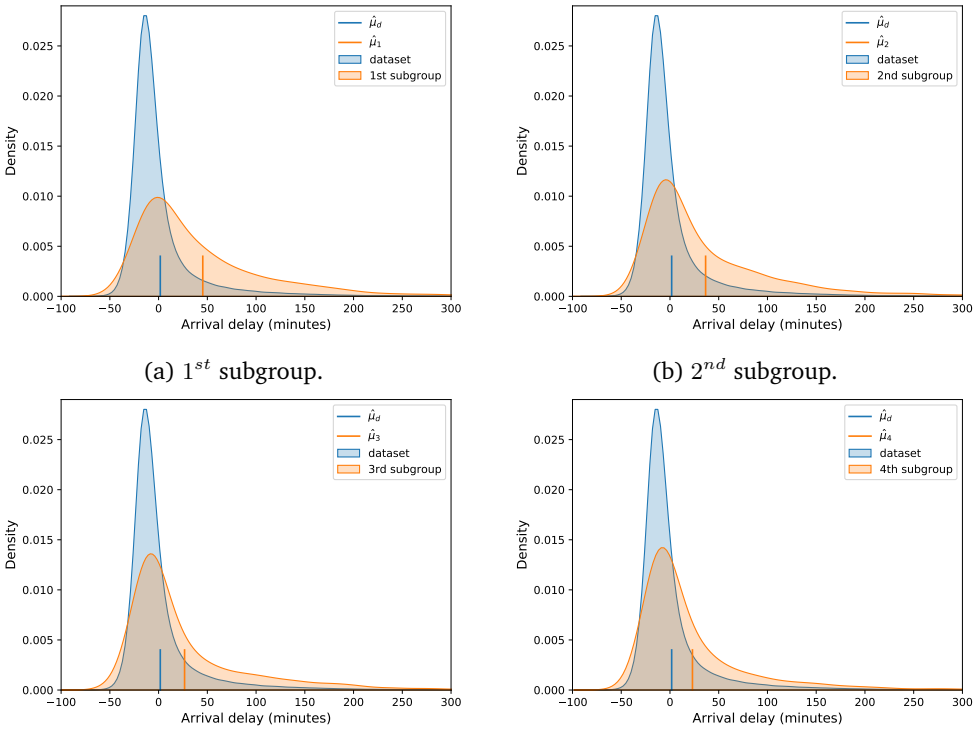
ures 5.6a, 5.6b, 5.6c, and 5.6d.

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.

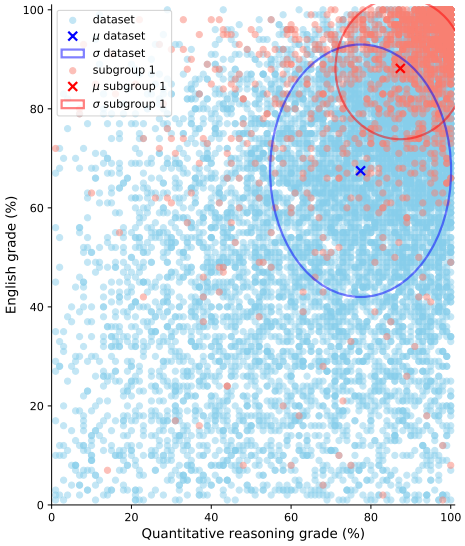
s	**description** of a student socioeconomic background	n_s	Quant.(%)	English(%)
1	household_income ≥ 5 min. wage & public_school = no & edu_mother $>$ high_school & Microwave = yes	1676	87 ± 16	88 ± 14
2	household_income < 5 min. wage & stratum < 5 & public_school = yes	4031	72 ± 25	54 ± 26
3	gender = 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 $= 0.41$

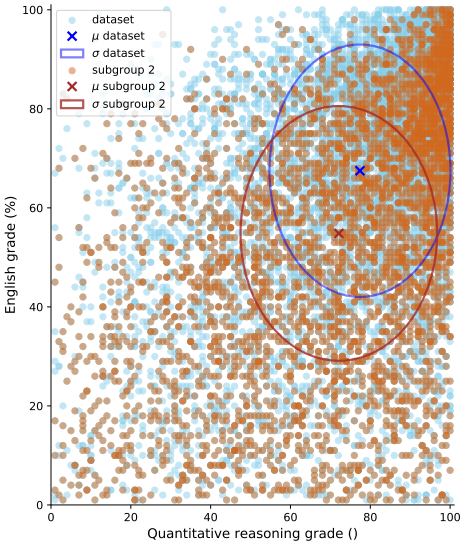
s	**description** of a student socioeconomic background	n_s	Quant.(%)	English(%)
1	household_income ≥ 5 min. wage & gender = M & household_size < 3 & edu_father $>$ high-school & mobile = yes	39	96 ± 5	92 ± 6
2	household_income ≥ 5 min. wage & school_type = academic & occ._mother = retired & edu_father $\geq$ 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 $= 0.52$

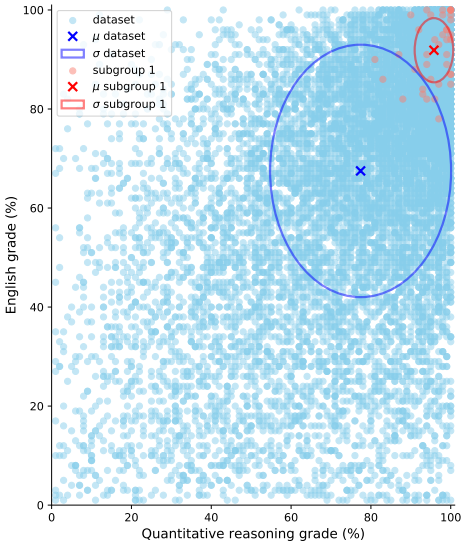
Figure 5.7: *Colombia engineering students* performance in Quantitative Reasoning and English exams. The results of Fig. 5.7a and 5.7b were obtained by RSD with *absolute gain* ($\beta = 0$) and *normalized gain* ($\beta = 1$). The dataset contains two numeric target variable *Quantitative Reasoning* 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.



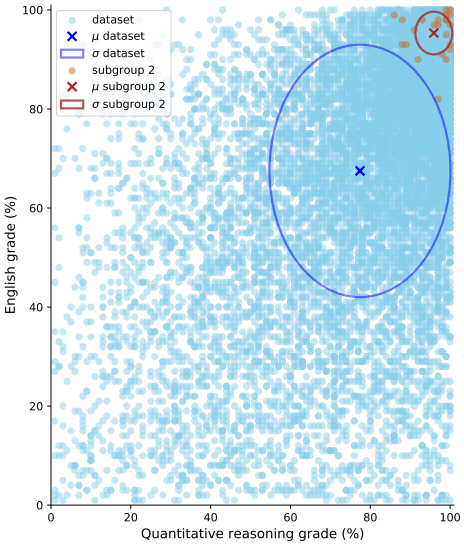
(a) 1^{st} subgroup with *absolute* gain.



(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.