Karlsruhe Institute of Technology

Comprehensible and Robust Knowledge Discovery from Small Datasets

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Vadim Arzamasov

Tag der mündlichen Prüfung:	22 Juli 2020
Erster Gutachter:	Prof. Dr.-Ing Klemens Böhm
Zweiter Gutachter:	Prof. Dr. Oliver Grothe

Acknowledgements

I want to thank my supervisor, Professor Klemens Böhm, who gave me research freedom and guidance. His continuous feedback on my research activities has helped to improve significantly every piece of my work.

I want to thank my colleagues who had a lot of patience in talking about my research problems, helping me overcome them. In particular, my thank goes to Georg Steinbuß, Edouard Fouché, Yaroslav Akhremtsev, Michael Vollmer, Natalia Arzamasova, Adrian Englhardt, Jens Willkomm, Holger Trittenbach, Pavel Obraztcov. I also want to thank Patrick Jochem, who has put the first stone in the foundation of my research agenda.

I want to thank Bettina Wagner and Barbara Breitenstein for supporting me in dealing with bureaucracy.

I want to thank my family and friends for their faith in me and support.

Abstract

Knowledge Discovery in Databases (KDD) targets extracting useful knowledge from data. Data can represent a set of measurements from a real-world process or be a set of input-output values of a simulation model. Two often conflicting requirements to the knowledge acquired are: (1) summarize the data accurately and (2) be in a comprehensible form for a human. Decision trees or subgroup discovery methods, the topic of this study, yield data summaries in the form of hyper-rectangles, which are human-comprehensible.

To demonstrate the importance of comprehensible data models, we study Decentral Smart Grid Control (DSGC) — a new system that implements demand response in electrical grids without significant changes in the infrastructure. The conventional analysis of this system performed so far was limited to considering identical participants and thus did not reflect the reality sufficiently well. We run many simulations, with diverse input values, apply decision trees to the resulting data, and provide new insights.

Decision trees allow describing the system behavior for all combinations of inputs. Sometimes, one is interested not in partitioning the whole input space, but in isolating large regions which lead to particular output — subgroups. The existing algorithms to discover subgroups usually require big datasets to produce stable and accurate hyper-rectangles. However, the data collection process is often costly. Our main contribution is improving subgroup discovery from small datasets — with few observations.

Subgroup discovery in simulated data is called scenario discovery. A commonly used algorithm for scenario discovery is PRIM (Patient Rule Induction Method). In this work, we propose a new procedure, REDS (Rule Extraction for Discovering Scenarios). We train an intermediate statistical model which generalizes fast and use it to create more data for PRIM. We provide the statistical intuition behind this idea. Experiments show that this method is much better than PRIM on its own. It reduces the number of simulation runs necessary by 75% on average.

With simulated data, one has perfect knowledge on the input distribution — an assumption of REDS. To make REDS applicable to measured data, we combined it with sampling from an estimated joint probability distribution of attributes. We have evaluated the resulting method in combination with various methods to generate data experimentally on a broad range of datasets. We did so for PRIM and BestInterval — another representative subgroup discovery method. In the majority of cases, our methodology enhanced the quality of discovered subgroups.

Zusammenfassung

Die Wissensentdeckung in Datenbanken (“Knowledge Discovery in Databases”, KDD) zielt darauf ab, nützliches Wissen aus Daten zu extrahieren. Daten können eine Reihe von Messungen aus einem realen Prozess repräsentieren oder eine Reihe von Eingabe-Ausgabe-Werten eines Simulationsmodells. Zwei häufig widersprüchliche Anforderungen an das erworbene Wissen sind, dass es (1) die Daten möglichst exakt zusammenfasst und (2) in einer gut verständlichen Form vorliegt. Entscheidungsbäume (“Decision Trees”) und Methoden zur Entdeckung von Untergruppen (“Subgroup Discovery”) liefern Wissenszusammenfassungen in Form von Hyperrechtecken; diese gelten als gut verständlich.

Um die Bedeutung einer verständlichen Datenzusammenfassung zu demonstrieren, erforschen wir Dezentrale intelligente Netzsteuerung — ein neues System, das die Bedarfsreaktion in Stromnetzen ohne wesentliche Änderungen in der Infrastruktur implementiert. Die bisher durchgeführte konventionelle Analyse dieses Systems beschränkte sich auf die Berücksichtigung identischer Teilnehmer und spiegelte daher die Realität nicht ausreichend gut wider. Wir führen viele Simulationen mit unterschiedlichen Eingabewerten durch und wenden Entscheidungsbäume auf die resultierenden Daten an. Mit den daraus resultierenden verständlichen Datenzusammenfassung konnten wir neue Erkenntnisse zum Verhalten der Dezentrale intelligente Netzsteuerung gewinnen.

Entscheidungsbäume ermöglichen die Beschreibung des Systemverhaltens für alle Eingabekombinationen. Manchmal ist man aber nicht daran interessiert, den gesamten Eingaberaum zu partitionieren, sondern Bereiche zu finden, die zu bestimmten Ausgabe führen (sog. Untergruppen). Die vorhandenen Algorithmen zum Erkennen von Untergruppen erfordern normalerweise große Datenmengen, um eine stabile und genaue Ausgabe zu erzielen. Der Datenerfassungsprozess ist jedoch häufig kostspielig. Unser Hauptbeitrag ist die Verbesserung der Untergruppenerkennung aus Datensätzen mit wenigen Beobachtungen.

Die Entdeckung von Untergruppen in simulierten Daten wird als Szenarioerkennung bezeichnet. Ein häufig verwendeter Algorithmus für die Szenarioerkennung ist PRIM (Patient Rule Induction Method). Wir schlagen REDS (Rule Extraction for Discovering Scenarios) vor, ein neues Verfahren für die Szenarioerkennung. Für REDS, trainieren wir zuerst ein statistisches Zwischenmodell und verwenden dieses, um eine große Menge neuer Daten für PRIM zu erstellen. Die grundlegende statistische Intuition beschrieben wir ebenfalls. Experimente zeigen, dass REDS viel besser funktioniert als PRIM für sich alleine: Es reduziert die Anzahl der erforderlichen Simulationsläufe um 75% im Durchschnitt.

Mit simulierten Daten hat man perfekte Kenntnisse über die Eingangsverteilung — eine Voraussetzung von REDS. Um REDS auf realen Messdaten anwendbar zu machen, haben wir es mit Stichproben aus einer geschätzten multivariate Verteilung der Daten kombiniert. Wir haben die resultierende Methode in Kombination mit verschiedenen Methoden zur Generierung von Daten experimentell evaluiert. Wir haben dies für PRIM und BestInterval — eine weitere repräsentative Methode zur Erkennung von Untergruppen — gemacht. In den meisten Fällen hat unsere Methodik die Qualität der entdeckten Untergruppen erhöht.

Contents

1 Introduction

Machine Learning (ML) is an extensive subarea of artificial intelligence that studies the methods of constructing algorithms capable of learning. ML models are widespread. They are used in computer vision, materials science, machine translation, credit scoring, and many other domains. Some machine learning models have a complex internal structure (for instance, artificial neural nets or boosted trees); the others are human-comprehensible (e.g., linear regression, decision trees, classification rules). Those with a more sophisticated structure often offer better accuracy. Thus, the common belief is that there exists an accuracy-comprehensibility trade-off [B+01, Gun16][1]. Below, we explain when one prefers comprehensible ML models. We divide the sources of data — input for ML models, — into *simulated* and *measured*. The former is an output of simulations, also known as computer experiments; the latter represents the set of measurements from physical processes. We intentionally do not use the terms "synthetic" and "real-world" to avoid making the impression that one type is less "real" than another. Both types are used widely for decision making.

1.1 Comprehensible ML Models for Simulated Data

The behavior of many systems, such as electrical grids or climate systems, can be described with differential or difference equations. The resulting model connects a set of input values to the output and can be solved with computer experiments. Analyzing data resulting from simulations has been of interest to the Knowledge Discovery in Databases (KDD) community for a long time. Specifically, after performing simulation runs for different combinations of input values, it is often worth to replace the simulation model with a statistical or machine learning model, a so-called *metamodel*, sometimes referred to as surrogate model, response surface model, replacement model [GCD+10] or model emulator [ULHM15]. Several authors give different reasons for doing so, including

- optimization of the simulated system [Kle15, SPKA01, WS06, GCD+10],
- design space exploration [SPKA01, WS06, GCD+10],
- sensitivity analysis [Kle15, GCD+10],
- model approximation [WS06],
- understanding relationships [SPKA01],
- risk analysis [Kle15].

[1] Rudin [Rud19] argues that this trade-off is a myth resulting from a little attempt of ML practitioners in finding useful features, which will allow a comprehensible model to achieve good accuracy. This concern might be true to a certain extent, as we will discuss in Section 6.2

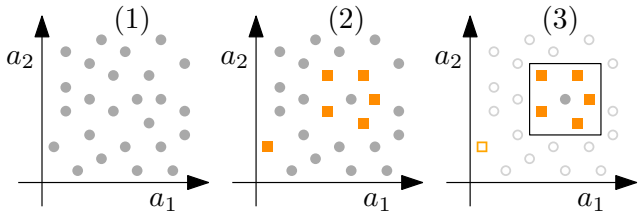


Figure 1.1: Scenario discovery process.

Although the items of this list are not clearly separated and can overlap, one can distinguish between the goals of high accuracy (optimization) and of having metamodels understandable for humans (design space exploration, risk analysis). One example task of the latter group has recently become known under the name *scenario discovery*, as we describe next.

The inputs of a simulation model can be classified into the ones which the user of the model (scientist, engineer, policy maker) can set, so-called *control* variables, and the ones reflecting uncertainty regarding specific conditions in which the modeled phenomena can take place, so-called *environmental* variables [SWN03]. In the decision making literature, a set of values of the control variables is referred to as alternative [HRZC15], policy, or candidate strategy [BL10, SS83]; and the set of values of environmental variables is known as state of the world [HRZC15, BL10], state of nature [SS83] or alternative future [BL10]. The term *scenario* has several definitions in this context. In the narrowest sense, it refers to any single possible state of the world [HRZC15, TGLS16]. Sometimes scenario refers to a set of states of the world where the policy fails to meet its goals [BL10, DHL+13]. According to an even broader definition, a scenario is the regions of particular interest in the space of environmental inputs [KC16, KJ16, IP16], for instance, where the output variable is above or below some threshold or takes a certain value. We use this last definition in this paper. Consequently, *scenario discovery* is the process of finding these interesting regions.

There are various degrees of uncertainty that one can associate with environmental variables [WLK13], see also [ULHM15] for different sources of uncertainties. So-called *deep uncertainty* occurs when their distributions are not known, or the users of a simulation model do not agree on these distributions [HRZC15, BL10]. In this case, one typically performs scenario discovery by

1. running several simulations for different combinations of environmental inputs drawn from a uniform distribution;
2. labeling the outcomes of interest with 1, the rest with 0;
3. applying a machine learning algorithm to find scenarios.

The algorithm used in the last step usually finds regions in the form of hyperboxes. On Figure 1.1 the plots correspond to three steps just described.

1.2 Comprehensible ML Models for Measured Data

Learning a comprehensible ML model for measured data is of particular importance for high-stakes decision support, for instance, in criminal justice or health care [Rud19]. A user

of such a model can explain and validate its logic, thus trusts its recommendations more and can ensure that there is no discrimination involved. This especially important than a recommendation is controversial to the intuition of its operator. In some domains, like credit scoring, providing explanations of decisions is legally enforced [Fre13].

Comprehensible models can be employed as a part of Explanatory Data Analysis [Tuk77] to find associations and designing better features. They can also lead to the formulation of new theories/hypotheses regarding the process which has generated data [Fre13].

Interpretable models can also be useful for identifying confounding variables. These variables usually allow fitting a statistical model that provides accurate predictions when applied to the data coming from the same population. However, the output of such models is often useless, counterintuitive, and may be harmful, as we illustrate shortly. In the example used in [RRM99, CLG+15, Lip18] an ML model has assigned the patients who had pneumonia a lower risk of dying if they had asthma. This counterintuitive behavior resulted from the fact that these patients were treated more aggressively. If one has used the output of such anML model for decision making and stopped treating severely sick patients properly, many of them could have died. Freitas [Fre13] provides another (perhaps, hypothetical) example. An artificial neural net had poor performance when deployed in the field since it was trained to recognize military objects on the dataset, which only contained a picture of tanks taken on a sunny day. That is, the ANN learned to recognize the colors of the sky. Similarly, in the example from [RSG16] an ANN recognizes husky as a wolf because of the snow in the background of a picture. Interpretable, white-box models make such inconsistencies apparent as opposed to complex, black-box models.

1.3 Research Goal

Many researchers agree that decision trees and subgroups — the main focus of this work, — are often a human-comprehensible form of knowledge representation. Both decision trees and subgroups are sets of hyperboxes (rules) in the data space. While this form of representation is flexible, — not restricted to a specific kind of dependency in data, — the algorithms to learn decision trees or discover subgroups require big datasets to produce stable and accurate knowledge summaries. In many cases, datasets are not big enough, since acquiring new examples is impossible (e.g., data for a rare disease) or comes at high financial (e.g., experiments in drug discovery) or computational (e.g., simulations to evaluate various shapes of an aeroplane wing) costs. The main target of this work is to improve knowledge discovery in the form of rules from small datasets.

1.4 Contributions

We have demonstrated the advantage of applying comprehensible models to the datasets resulting from simulations. To this end, we focused on Decentral Smart Grid Control (DSGC) — a novel approach realizing demand response in electrical grids without a need in a centralized IT infrastructure. The key idea of DSGC is to connect the electricity price to the local frequency of an electrical grid. Simulations based on this system can be used to

ensure the stability of the DSGC for different combinations of inputs values which describe the behavior of participants. The original DSGC model is subject to various restrictions. In particular, they include that participants of the grid are homogeneous, e.g., behave in the same way. To get rid of this unrealistic assumption, we have simulated the system for a variety of combinations of input values to figure out whether the system is stable for each particular combination. Using the resulting dataset, we trained a decision tree. This allowed us to reveal new insights regarding DSGC not known from previous studies. For instance, we found that the system can be stable even if some participants adapt their energy consumption to the price changes with a high delay.

Decision tree divides the space of inputs of a simulation model into regions leading to a particular output. In some cases, one may not require to partition the whole space. Instead, one may want a small set of scenarios which are large and pure, that is, are very likely responsible for a particular outcome. This is known as *subgroup discovery* in the KDD community. PRIM, a well-known subgroup discovery algorithm, is a state-of-the-art method for scenario discovery. However, on small datasets, PRIM performs poorly, producing essentially random subgroups. So we proposed method REDS (Rule Extraction for Discovering Scenarios) which adds an intermediate step to the process as follows. Instead of applying PRIM to the data resulting from simulations, we use this data to train an accurate metamodel. We then use this metamodel as a replacement for the simulation model to produce a significantly larger dataset inexpensively. We then use the resulting data as input for PRIM. As one contribution, we provide a statistical intuition behind REDS. In the experiments, we compare our REDS with the state-of-the-art techniques for scenario discovery on a variety of test models as well as on the DSGC simulation model. We find that REDS almost always outperforms conventional methods. For many test models, the results obtained with REDS with 400 computer experiments are comparable to the result obtained with conventional methods for 1600 simulations, meaning $\approx 75\%$ reduction in computational resources required for simulating the system. Moreover, in contrast to PRIM, REDS can benefit from existing active learning techniques to bring down the number of simulations needed to discover scenarios even further.

REDS works well for simulated data since one has control over the simulation process and thus has perfect knowledge of the joint distribution $p(x)$ of inputs x. Having $p(x)$ it is easy to generate any number of input combinations. The metamodel can then cheaply provide estimates $f^{am}(x)$ of the conditional probabilities $f(x) = p(y = 1|x)$. With measured data, one needs a procedure allowing to obtain a sample from $p(x)$. We propose REDS+, a generalization of REDS, to improve scenario discovery from datasets with unknown $p(x)$. We tested REDS+ in combination with various metamodels, methods to generate data, and two subgroup discovery methods — PRIM and BestInterval, — experimentally on a broad range of datasets. The results provide evidence that REDS+ improves subgroup discovery on measured data.

To facilitate the adoption of our results, we publicly release our implementations, experiments, and benchmark data via open-source platforms.

1.5 The Scope of the Work

In this work we consider datasets

1. in which the property of interest can take two values (interesting/uninteresting or 0/1); this is a usual setting in scenario discovery domain and for many subgroup discovery algorithms;
2. with real-valued *attributes* — the factors which have a potential influence on the property of interest.

These restrictions define the domain, where our contributions are justified experimentally. However, many ideas we have proposed can be generalized to more general settings (e.g., regression instead of classification or categorical and mixed attributes) straightforwardly, as we will explain.

1.6 Notations

In the rest of the work, we use the following notations. A dataset D is a $N \times (M+1)$ matrix

$$D = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1M} & y_1 \\ x_{21} & x_{22} & \dots & x_{2M} & y_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{NM} & y_N \end{pmatrix}$$

The first M columns are *attributes* $a_j = (x_{1j}, \dots, x_{Nj})$ and the last column is a *target* variable $y = (y_1, \dots, y_N)$. We refer to the first M elements in each row $x_i = (x_{i1}, \dots, x_{iM})$ as *point*; the whole row $d_i = (x_i, y_i) = (x_{i1}, \dots, x_{iM}, y_i)$ is an *example*.

For instance, if D results from computer experiments, x_i are the inputs' values for a simulation model and y_i — respective output. In this case, the number of rows $N = |D|$ is the number of simulation runs.

We define $N^+ = \sum_{i=1}^{N} y_i$. A *hyperbox*, also referred to as pattern, subgroup description or body of a rule [Atz15, GR09, MNFK12], B is a conjunction of intervals $B = \prod_{j=1}^{M} [a_j^l, a_j^r]$, $a_j^l \in \mathbb{R} \cup -\infty$, $a_j^r \in \mathbb{R} \cup +\infty$. Sometimes the intervals defining the box are open (a_j^l, a_j^r) or half-open $(a_j^l, a_j^r]$, — this will be clear from the context. A *subgroup* (pattern cover) $S_B \subseteq D$ corresponding to a hyperbox B is a set of examples from a dataset D "inside" the hyperbox. That is $d_i \in S_B \iff d_i \in D \wedge x_{ij} \in [a_j^l, a_j^r], \forall j$. We say that the attribute a_j *defines* a subgroup S_B if $a_j^l \neq -\infty$ or $a_j^r \neq +\infty$. The size of the subgroup S_B is $n = |S_B|$; $n^+ = \sum_{i:d_i \in S_B} y_i$ is the sum of values of the target variable in S_B. We usually assume that $y \in \{0,1\}^N$ and call examples with $y_i = 1$ *interesting* or *positive* and with $y_i = 0$ — *not interesting* or *negative*, hence the notations N^+ and n^+.

Further, let U_j be the ordered set of unique values in the j-th column of D, $j \in 1, \dots, M$ — $U_j = \{u_{1j}, \dots, u_{Z_j j}\}$: $x_{ij} \in U_j \ \forall i$ and $\forall i, k, i < k : u_{ij} < u_{kj}$. In what follows, we sometimes omit index i and/or j to simplify notations. For instance, we write a to refer to an attribute, $U = \{u_1, \dots, u_Z\}$ — to its ordered unique values, x — to a point.

Symbol	Interpretation
P^{source}	nominal power of generator/load
I	moment of inertia
K_D	friction coefficient
P^{max}	capacity of the line
t	time
$\delta(t)$	rotor angle (phase)
ω	grid reference frequency (e.g. 50 Hz)
$\theta(t)$	$= \omega t - \delta(t)$, rotor angle relatve to reference frequency
p	price
c_1	proportionality factor defining the price based on $\theta(t)$
c	coefficient proportional to price elasticity
$\hat{P}(p)$	$\approx P_j + c_j \cdot (p_j - p_\omega)$, power consumed/produced at price p
p_ω	electricity price when $d\theta_j/dt \equiv 0$
P	$= (P^{source} - K_D\omega^2)/I$, mechanical power produced/consumed
κ	$= 2K_D/I$, damping constant
K	$= P^{max}/I\omega$, coupling strength, proportional to line capacity
γ	coefficient, proportional to price elasticity
τ	reaction time, the delay between a price change and adaptation to it
T	averaging time, required to measure price signal

Table 1.1: DSGC-specific notations. We omit indexes of system participants

Table 1.2 summarizes the notations just described as well as the ones introduced later. The last column of the table refers to a section containing the first mention of notation. For the DSGC system, we use notations described in Section 4.2 and list them separately in Table 1.1; the lower part contains the main notations and the upper part — auxiliary ones.

1.7 Dissertation Outline

Chapter 2 reviews related work and describes the existing subgroup discovery algorithms we use in subsequent chapters. Chapter 3 features our case study, aiming at demonstrating the utility of human-comprehensible ML models. Chapters 4–5 feature REDS and REDS+, the approaches we developed for improving scenario discovery and subgroup discovery from simulated and measured data correspondingly. Chapter 6 discusses possible future research directions. Chapter 7 concludes.

Symbol	Interpretation	Ref.
a_j	$= (x_{1j}, \ldots, x_{Nj})$, the j-th attribute (column) of D, $j \in \{1, \ldots, M\}$	1.6
$a^{[h]}$	i-th dummy attribute for original attribute a	2.3.1
a_j^l	$\in \mathbb{R} \cup -\infty$, the left end of the interval in j-th attribute defining B	1.6
a_j^r	$\in \mathbb{R} \cup +\infty$, the right end of the interval in j-th attribute defining B	1.6
α	"peeling" parameter of PRIM	2.3.3
B	$= \prod_{j=1}^{M} [a_j^l, a_j^r]$, a hyperbox (pattern, subgroup description)	1.6
$\mathcal{B}$	search space — a set of all possible subgroup descriptions which a subgroup discovery algorithm can handle	2.3.1
β	"pasting" parameter of PRIM	2.3.3
d_i	$= (x_i, y_i) = (x_{i1}, \ldots, x_{iM}, y_i)$, an *example* — the i-th row of D	1.6
D	$N \times (M + 1)$ matrix	1.6
D^{val}	a dataset for validation	2.3.3
$f(x)$	$\mathbb{R}^M \rightarrow \mathbb{R}$: $p(y = 1\|x)$	1.4
$f^{am}(x)$	$\mathbb{R}^M \rightarrow \mathbb{R}$, approximation of $f(x)$ estimated from data D	1.4
k_{max}	maximal number of iteration of PRIM algorithm	2.3.3
L	number of newly generated points with REDS/REDS+ algorithm	4.2
m	the number of attributes defining subgroup	2.1
M	number of attributes in D	1.6
$minpts$	minimal size of subgroup found by PRIM	2.3.3
n	$\|S_B\|$, the number of examples in S_B	1.6
n^+	$= \sum_{i:d_i \in S_B} y_i$	1.6
N	number of rows in D	1.6
N^+	$= \sum_{i=1}^{N} y_i$	1.6
$p(x)$	$\mathbb{R}^M \rightarrow [0, 1]$, probability density function of a random variable X, so that x_i in D are realizations of X	1.4
$p(y\|x)$	conditional probability density function of a target variable Y given X, so that y_i in D are realizations of Y	2.4.3
ϕ	quality measure of a subgroup	2.3.1
q	number of attributes in a subset for PRIM algorithm with bumping	2.3.3
Q	number of iterations in PRIM algorithm with bumping	2.3.3
S_B	$\subseteq D$, a subgroup defined by a hyperbox B, i.e., a set of examples from D which are inside B	1.6
U_j	$= \{u_{1j}, \ldots, u_{Z_j j}\}$, an set of unique values in the j-th column of D; if applicable, sorted in ascending order	1.6
V	$= \{v_1, \ldots, v_z\}$, a set of values — split points for the attribute a sorted in ascending order	2.3.1
x_i	$= (x_{i1}, \ldots, x_{iM})$, a *point* — the first M elements in the i-th row	1.6
y	$= (y_1, \ldots, y_N)$, the target variable — the $(M + 1)$-th column of D	1.6
y_i^{am}	$= f^{am}(x_i)$	4.2.1
Z_j	number of unique values in the j-th column of D	1.6
bs	the beam size	2.3.3

Table 1.2: Overview of general notations

2 Fundamentals and Related Work

This chapter describes jointly the methods we build upon and related ideas. In Section 2.1, we discuss the comprehensibility of machine learning (ML) models. Section 2.2 describes research related to analyzing simulated data with ML models. In Section 2.3, we take a closer look at subgroup discovery methods which are the central part of this dissertation. Finally, Section 2.4 features findings in other domains which inspired our ideas.

2.1 Comprehensibility of ML Models

We first describe which ML models are deemed human-comprehensible and how to measure comprehensibility. An alternative to fitting an interpretable model to data is to fit a sophisticated model and explain its behavior or individual predictions. This approach is beyond the scope of the dissertation; we briefly describe it at the end of this section.

2.1.1 Comprehensible ML Models, Measures of Comprehensibility

Following [Rud19, GMR+19][1], we use the terms *comprehensibility* and *interpretability* interchangeably throughout the document. The model which is complicated for a human to interpret, i.e., not comprehensible, is a *black-box* or *complex*[2] model. Several studies suggest that results produced by decision trees or classification rules are quite comprehensible [Fre13, HDM+11, BSH+10], as compared to other data-mining methods. Ustun et al. [UR16] argue that linear models, — scoring systems — are comprehensible. Additionally, oblique rules [SL97] might also belong to this category, according to [HDM+11, PSLB15][3].

Assessing the comprehensibility of a model (by a human) requires experiments involving humans. Its result might depend on a task (a dataset) at hand [Fre13] and background of participants [HDM+11]. For instance, Pazzani et al. [PMS97] provide evidence that experts find the rules more comprehensible if they comply with their prior knowledge. Many researchers use proxies of interpretability, such as the number of rules produced by the model or the number of attributes involved in defining a single rule. These criteria are often negatively correlated. For example, the decision trees with "M-of-N" splitting criterion [CS95] or oblique rulestend to use more attributes for each rule but produce fewer rules on average [SL97]. Freitas [Fre13] warns against using the model size as a proxy of interpretability and refers to the studies where users found larger models to be more comprehensible than the smaller ones. Additionally, the number of rules

[1] Rudin [Rud19] also suggests two other synonyms — *explainability* and *transparency*

[2] We use this term in line with the literature [Lip18, Rud19, HDM+11]. However, the term *simple* model sometimes has a different meaning than comprehensible model [VI19]

[3] The latter citation examines the standard rules in rotated space — essentially oblique rules.

might be small just because the rules do not cover all points in a dataset D, i.e., not exhaustive [ZJC03, HSBV08]. Comparing an exhaustive set of rules to a not exhaustive one, measuring interpretability through the number of rules is not fair.

Rudin [Rud19] argues that there is no need to come up with a single comprehensibility measure since the notion of interpretability is domain-specific similarly to the notion of model performance or accuracy. The latter includes, for instance, accuracy, an area under the receiver operating characteristic curve, F1-score and many others, e.g., a variety of measures computed from a confusion matrix. This is in line with the recent research of Lipton [Lip18] suggesting that "interpretability is not a monolithic concept". The research on interpretability and the approaches to measure it is ongoing. Therecent findings are in [DVK17, Mil19].

In our work, we mainly focus on subgroup discovery algorithms. These algorithms produce not exhaustive sets of rules. Thus, we measure comprehensibility through the number of attributes involved in the rule definition m (search depth). Lower value of m is preferable. This is in line with scenario discovery literature.

2.1.2 Explaining Complex Models

Instead of training an interpretable model, one can try to explain a complex model to achieve comprehensibility. Black-box models are especially challenging to interpret when they use more than three inputs. Indeed, according to [Dom12] "It's even been said that if people could see in high dimensions machine learning would not be necessary". The examples of complex models are neural networks, trees ensembles, support vector machines. Guidotti et al. [GMR+19] provide a comprehensive and up-to-date review of the methods for explaining these models. According to their taxonomy, one can distinguish between model-specific (designed for a particular black-box model type) or model-agnostic (generally applicable) explanatory techniques. One can also distinguish *global* (model) from *local* (outcome) explanations. The latter do not allow to understand the whole logic of a model but provide an explanation for the reasons of each particular prediction. This is achieved for instance, through estimating "local" linear [RSG16] or rule-based [GMR+18] model, providing examples and counter-examples [GMMP20], or by calculating a gradient of a loss function [BSH+10].

The methods for global explanation usually *rule extraction* algorithms [HBV06]. We review these methods in greater detail in Section 2.4.1 since they have inspired our work.

Rudin [Rud19] speaks in favour of inducing comprehensible models directly, instead of extracting explanations from complex models.

2.2 ML Models for Data from Simulations

A significant part of this work addresses the problem of analyzing simulated data. Our use case, Decentral Smart Grid Control (DSGC), is a novel idea of realizing demand response in an electrical grid. We use simulations to define if the grid is stable for particular input values of DSGC. Later in this document, we analyze the output of DSGC simulations with

comprehensible ML models. That is, we perform scenario discovery. We further propose a methodology to improve the quality of scenario discovery from limited data.

This section presents the respective related work. We first feature the research on using ML models to analyze data from simulations of electrical systems — in Section 2.2.1. Section 2.2.2 is an overview of scenario discovery literature. In the end, we describe existing attempts to improve scenario discovery — competitors of our approach, REDS.

2.2.1 ML Models for Data from Simulations of Electrical Systems

Several studies are applying ML models to analyze the security/stability of power systems. McCalley et al. [MWZ+97] employ a neural network to security problems within a power system in California. System behaviour is studied in 1792 simulation runs. The resulting boundary has been visualized in the form of so-called nomograms. The limitation of this approach is that it considers two inputs at a time and hence does not explicitly support exhaustive insights with more inputs. Jayasekara et al. [JA06] propose modelling a non-linear security boundary by using features formed as monomials of the original input up to a certain degree. They solve the problem of a large number of features with kernel ridge regression. Two systems are simulated: the New England 39-bus system and a larger 470-bus system. Moulin et al. [MDSESM04] apply support vector machines and neural networks to analyze the transient stability of a 2484-bus system. The data is generated in 1242 simulation runs and contains 244 inputs. To reduce the number of inputs, a feature selection mechanism is applied in some cases. This work concludes that a support vector machine with no feature selection achieves a higher accuracy than a neural network with feature selection. In [AR13], decision trees are used to study the transient stability of a toy 9-bus system and of the 1696-bus Iran national grid. The authors then compare the results to those obtained with ANN and SVM models. In all cases, the accuracy is about 99%, although it is not clear whether the out-of-sample data was used for testing. — All these studies target accurate prediction, but not at a general view on the system.

2.2.2 Scenario Discovery

To discover scenarios, one usually induces a comprehensible ML model on the dataset resulting from computer experiments. Although not using the term *scenario discovery*, similar frameworks originated before: Pierreval [Pie92] use the GENREG algorithm, Yoshida and Nakasuka [YN89] use oblique rules, and Berthold and Huber [BH99] propose a fuzzy rule learning algorithm to analyze simulation outputs. Some authors, e.g. [LBB08, HHRK15, ABJ18, KP13] use decision trees trained with the Classification and Regression Trees (CART) algorithm [BFOS84]. Currently, Patient Rule Induction Method (PRIM) [FF99], a subgroup discovery method, dominates in the scenario discovery domain, see [GRS16, BL10, HRZC15, LGPB06, GL07, LBB08, HHRK15, KP13] and many others, cf. [GRS16].

Lempert et al. [LBB08] compare CART and PRIM and conclude that the latter is more interactive and requires less post-processing effort. CART has a high variance [Bre96], meaning that the boxes vary highly for different datasets produced when simulating the same phenomenon, and it might include irrelevant attributes in the definition of the

boxes [Fre13]. This is undesirable according to [KC16, BL10] but also holds for PRIM to some extent, as we will show.

Many listed algorithms target at partitioning the input space into regions and assigning a class (interesting/uninteresting) to each of them. Accuracy is their main goal. Recall and precision of individual hyperboxes are the primary target of PRIM and often are more relevant in scenario discovery.

2.2.3 PRIM Improvements for Scenario discovery

Several improvements of PRIM for scenario discovery were recently proposed. Normally, PRIM targets at maximizing the mean outcome value n^+ within a hyperbox; here, the function "mean" is called peeling criterion. Kwakkel and Jaxa-Rozen [KJ16] study alternative peeling criteria other than simple mean, also proposed in [FF99], and conclude that these alternatives are beneficial with heterogeneous inputs. Kwakkel and Cunningham [KC16] propose using a bagging procedure to increase the quality of PRIM. In fact, the authors use *bumping* [HTF09], also proposed by Friedman and Fisher [FF99], combined with random feature selection. Dalal et al. [DHL$^+$13] combine PRIM with principal component analysis to produce oblique rules (PCA-PRIM). Bryant and Lempert [BL10] complement PRIM with a "quasi p-value" test to exclude insignificant attributes from the box definition.

We will explain PRIM with bumping [KC16] and original PRIM in Section 2.3.3. PCA-PRIM [DHL$^+$13] and different peeling criteria [KJ16] are orthogonal to our study.

2.3 Subgroup Discovery

PRIM algorithm mentioned above is one of the most known representatives of subgroup discovery domain. The central part of our work is improving the quality of scenario discovery from small datasets. We now provide a brief introduction in this domain (Section 2.3.1) and describe existing quality measures (Section 2.3.2) and subgroup discovery algorithms (Section 2.3.3) which we will further use with our approaches.

2.3.1 Definition and Taxonomy

One of the earliest definitions of the subgroup discovery task [Wro97] is the following. Discover the subgroups of the population that are statistically "most interesting", i.e., are as large as possible and have the most unusual statistical (distributional) characteristics with respect to the property of interest. Sometimes this definition is further specified, e.g., to explicitly include the number of subgroups or the length of their description [GR09].

To define a subgroup discovery algorithm, one should propose a search space, search strategy and a quality measure, which will quantify (and often combine) the notions of size and unusualness from the definition above.

Search Space. Given a dataset D, a search space [Atz15] $\mathcal{B}$ (also referred to as hypothesis language [Wro97], pattern language [MNFK12] or description language [HCGdJ11]) is a set of all possible subgroup descriptions which a subgroup discovery algorithm can

Original	Dummy			
a	$a^{[1]}$	$a^{[2]}$	$a^{[3]}$	$a^{[4]}$
1	1	1	1	1
2	0	1	1	1
3	0	0	1	1
4	0	0	0	1
5	0	0	0	0

Original	Dummy			
a	$a^{[1]}$	$a^{[2]}$	$a^{[3]}$	$a^{[4]}$
A	1	0	0	0
G	0	1	0	0
A	1	0	0	0
C	0	0	1	0
T	0	0	0	1

Table 2.1: Attributes $a^{[h]}$ created for a numeric (left) and a categorical attribute a (right).

handle. One can distinguish subgroup discovery algorithms based on the type of acceptable attributes. Some algorithms [AP06, GRW08] assume nominal-valued attributes; subgroup definitions produced by them are conjunctions of conditions $a = u \in U$ (see Section 1.6 for notations). Other methods [FF99, MNFK12] can also include conditions like

$$a \in \tilde{U} \subseteq U. \tag{2.1}$$

Additionally, some methods work with numeric attributes and include intervals in subgroup definition [FF99, GR09, MNFK12], as we explained in Section 1.6. In our work, we focus on the methods of this group. Consequently, we deal with the search space consisting of boxes B defined by a conjunction of intervals. We will discuss selected algorithms in more detail in Section 2.3.3.

The distinction of algorithms based on the type of attributes is not as strict as it might seem. To adapt methods dealing with nominal attributes to a numerical domain, one can discretize them. There is a big variety of discretization methods [GLS+13, AKV19]. The result of discretization procedure for the numeric attribute a is the ordered set of unique split points $V = \{v_1, \ldots, v_z\}$; a is then replaced by a new nominal attribute with domain $\{(-\infty, v_1], \ldots, (v_z, +\infty)\}$. If $V = U$, we call discretization *exhaustive* [GR09][4]. If a subgroup discovery algorithm does not allow condition like (2.1), even exhaustive discretization will not produce as rich search space as many algorithms designed for numerical data. To see this, observe that, e.g., intervals like $(v_i, v_{i+2}]$, $i \leq z - 2$ do not belong to its search space. To overcome this issue, Lavrac and Gamberger propose [LG04] a binarization procedure. In short, the idea to replace each attribute a with a set of attributes $a^{[h]}$, each taking values form $\{0, 1\}$ so that each possible interval or a set of values can be selected by a conjunction of equality conditions $a^{[h]} = 0$ and/or $a^{[h]} = 1$. The number of new binary features is $Z - 1$ for numeric and ordinal and Z for nominal attribute a, where Z is the number of unique values of a in D (see Section 1.6). Table 2.1 illustrates this. For instance, $a \in [2, 4] \iff a^{[1]} = 0 \wedge a^{[4]} = 1$ (left) or $a \in \{C, G\} \iff a^{[1]} = 0 \wedge a^{[4]} = 0$ (right).

Quality Measures. Given a dataset D, a quality measure is a function: $\phi : \mathcal{B} \to \mathbb{R}$ that maps every subgroup description in the search space to a real number. This number reflects the interestingness of subgroup S_B. Various measures exist for binary, nominal and numeric

[4]Do not confuse with exhaustive search, introduced later.

target variables y. For instance, weighted relative accuracy, chi-squared, binomial test, information gain [MNFK12], multi-class weighted relative accuracy, numeric weighted relative accuracy, mean test, weighted Kullback–Leibler divergence, weighted Krimp gain [vLK12], proper scoring rules [SKFK16] and many other [Atz15]. In Section 2.3.2, we review some measures which we will use further.

When more than one subgroup is required, one should access the *joint* quality of multiple subgroups. The naive approach would be to return the demanded number of subgroups which have the highest quality [GR09]. This, however, might result in a redundant subgroup set — containing significantly overlapping subgroups. A covering scheme can reduce redundancy. With sequential covering, the algorithm searches for a single best subgroup description B at each iteration. Then it removes the examples S_B from D and proceeds with the next iteration until the required number of subgroups is obtained [FF99]. A weighted covering algorithm [LKFT04, GL02], modification of sequential covering, downweights the examples from subgroups discovered in previous iterations instead of removing them — this strategy allows subgroups to overlap.

Covering approaches require multiple runs of the algorithm. This can be computationally expensive. Thus, several studies propose to include an explicit diversity measure in the evaluation of the sets of subgroups (see e.g., [vLK12]), which return a set of diverse subgroups after a single run.

Search Strategy. A search strategy is an algorithm to traverse the search space. For the sake of simplicity assume that a single subgroup should be found — this excludes diversity criterion from consideration. A naive search strategy would be to *exhaustively* evaluate the complete search space and report the best subgroup. However, this strategy usually is not computationally affordable due to the high search space cardinality, especially for datasets D with high number M of numeric attributes discretized exhaustively. Different pruning strategies were proposed to speed up the search, including optimistic estimate pruning [GRW08, Wro97] or the strategies explicitly developed for numeric attributes [GR09, MNFK12], see also [Atz15]. Pruning strategies do not affect the search quality; they still return a globally best subgroup from the search space.

Usually, exhaustive search is too expensive even with the use of pruning. In this case, two strategies are conceivable. First one can reduce the search space by defining minimal subgroup size n or maximal subgroup description size m (search depth) — the number of attributes defining subgroup S_B. Second, one can use a heuristic search strategy, for instance, beam search [LKFT04, vLK12] or genetic algorithm [LRRV14]; see also [Atz15, Hel16] for other examples.

2.3.2 Quality Measures

Quality measure ϕ of a subgroup S_B usually depends on the size of the dataset $N = |D|$, the sum of the values of target vector y, N^+, and respective numbers for the subgroup S_B, n and n^+ (see Section 1.6). The Numeric Weighted Relative Accuracy (sometimes referred to as impact [Web01]) measure is calculated as

This work	[BL10]	[FF99]	[HCGdJ11]	Formula
NWRacc	–	coverage	WRAcc	(2.2)
recall	coverage/recall/sensitivity	–	sensitivity	(2.3)
precision	density/precision	box mean	confidence	(2.3)
–	support	support	coverage	n/N
–	–	–	precision Q_c	$n^+ - c \cdot n^-$
–	–	–	precision Q_g	$n^+/(n^- + g)$
–	–	–	support	n^+/N

Table 2.2: Names of quality measures suggested by literature.

$$NWRAcc(D, S_B) = \frac{n}{N}\left(\frac{n^+}{n} - \frac{N^+}{N}\right) \tag{2.2}$$

When y is binary, $y \in \{0,1\}^N$, N^+ and n^+ are counts of interesting examples in D and S_B, respectively, and the measure is called Weighted Relative Accuracy (*WRAcc*).

One usually wants a subgroup to include many examples (be large) and have a high sum n^+ (number of interesting examples, when y is binary). This is equivalent to maximizing *recall* and *precision*:

$$recall(D, S_B) = \frac{n^+}{N^+} \qquad precision(D, S_B) = \frac{n^+}{n} \tag{2.3}$$

We note, that confusion might arise since literature uses different names for the same qualities often mixing them. Table 2.2 reveals some of these variations[5]. For instance, the term "coverage" can stand for three different quantities.

Evaluation of SD Methods. There is no consensus among researches if it is correct to measure the quality of a subgroup description on the same data as was used to discover it. For instance, a cross-validation procedure is used in [SKFK16, CGdJ+11, KLZG05, KLJ03, LRRV13], whereas [GR09, VGBS19, RRRA12, AP06, vLK12] most likely use the same dataset. In this work, we always use independent test data to evaluate the quality of subgroup descriptions.

2.3.3 Algorithms for Data with Numerical Attributes

Below we describe PRIM, its modification — PRIM with bumping, and BestInterval subgroup discovery algorithms.

PRIM. The PRIM algorithm was initially proposed in [FF99]; [HTF09] (pp. 317–320) contains a concise description. The algorithm works in two steps, called *peeling* and

[5] In this table, $n^- = n - n^+$; c, g are some numbers.

Algorithm 1: PRIM.peel (peeling step)

Data: D, D^{val}, α, $minpts$, k_{max} — as described in the text.
Result: sequence of nested hyperboxes

1 $k = 0$;
2 $B_0 = \prod_{j-1}^{M}(a_j^l, a_j^r), \quad a_j^l = -\infty,\ a_j^r = +\infty \quad \forall j$;
3 **while** $|S_{B_k}| > minpts$ & $|S_{B_k}^{val}| > minpts$ & $k < k_{max}$ **do**
4 $m = -1$;
5 **for** $1 \le j \le M$ **do**
6 $D_r = \{d_i \in S_{B_k} | x_{ij} \le t_r\}$, choose t_r so that $|D_r| = (1-\alpha) \cdot |S_{B_k}|$;
7 $D_l = \{d_i \in S_{B_k} | x_{ij} \ge t_l\}$, choose t_l so that $|D_l| = (1-\alpha) \cdot |S_{B_k}|$;
8 **if** $\text{avg}(\{y_i | d_i \in D_r\}) > m$ **then**
9 $m = \text{avg}(\{y_i | d_i \in D_r\})$;
10 $B_{k+1} = \text{set}(B_k, a_j^r = t_r)$;
11 **if** $\text{avg}(\{y_i | d_i \in D_l\}) > m$ **then**
12 $m = \text{avg}(\{y_i | d_i \in D_l\})$;
13 $B_{k+1} = \text{set}(B_k, a_j^l = t_l)$;
14 $k = k + 1$;
15 $m^{val}(i) = \text{avg}(\{y_j | d_j \in S_{B_k}^{val}\})$;
16 $last = \arg\max_i (m^{val}(i))$;
17 **return** $\{B_0, \ldots, B_{last}\}$

pasting. We describe them separately. Algorithm 1 is the peeling step[6]. It starts with the whole dataset D and the M-dimensional box $B_0 = \prod_{j-1}^{M}(-\infty, +\infty)$. Then it repeatedly "peels out" α points from the data with a hyperplane orthogonal to one dimension, so that the mean value of y of the remaining points is maximal (Lines 5–13), and it adjusts the box so that it is a minimal bounding rectangle of these remaining points (Line 14). Here α is the *peeling parameter* [FF99] and denotes the *share* of points. This is done until the stopping criterion is met. In our case, it is the minimum number of points *minpts* of the train set D or validation set D^{val} contained in the box, or the number of iterations k_{max} (Line 2). We introduced the latter condition to restrict the minimum *relative* size of a scenario in our experiments. Finally, the hyperbox with the highest mean on D^{val} (Line 16) is returned together with all preceding boxes (Line 17). The rationale is to let a domain expert choose the one which best suits their needs.

The pasting step (Algorithm 2) works similarly but in the opposite direction. It receives the data D, the initial box containing it — B_0, the pasting parameter β and the box to be expanded — B. It repeatedly expands the box along any dimension so that the mean value of y of the points from D inside the resulting box increases (Lines 8, 11). Let the box be described with the inequalities $B = \prod_{j-1}^{M}[a_j^l, a_j^r]$, $t = 1, \ldots, D$. The operation

[6] In different sources PRIM descriptions slightly vary. For instance, original paper [FF99] proposes to use a validation set D^{val} to select the last returned box. Hastie et al. [HTF09] propose to use cross-validation but do not explain the particular procedure detailed enough.

Algorithm 2: PRIM.paste (pasting step)

Data: D, β, B — as described in the text
Result: expanded hyperbox.

```
1  boxes = B;
2  while boxes <> {} do
3      B = sample.one.box.randomly(boxes);
4      m = avg({y_i | d_i ∈ S_B});
5      boxes = {};
6      for 1 ≤ j ≤ M do
7          B_r = expand(B, a_j^r, β);
8          if avg({y_i | D_i ∈ S_{B_r}}) > m then
9              boxes = append(boxes, B_r);
10         B_l = expand(B, a_j^l, β);
11         if avg({y_i | D_i ∈ S_{B_l}}) > m then
12             boxes = append(boxes, B_l);
13 return B
```

$B_{new} = \text{expand}(B, \xi, \beta)$ changes the value ξ in the box description so that the volume of B_{new} is greater than that of box by $1 + \beta$. The algorithm stops when no further expansion is possible (Line 2).

PRIM with Bumping. The PRIM algorithm with bumping [KC16] produces multiple boxes by varying the dataset D and returns only the ones not dominated by any other box in terms of precision and recall (Section 2.3).

Definition 2.3.1 *For a set of quality measures $\{\phi_1, \ldots, \phi_n\}$, a box b is dominated by a box B if $\forall k \in \{1, \ldots, n\} : \phi_k(S_b) \leq \phi_k(S_B)$ and $\exists k^* : \phi_{k^*}(S_b) < \phi_{k^*}(S_B)$.*

The PRIM algorithm with bumping works as follows.

1. Take a random bootstrap sample D^{bs} from D;
2. take a random subset A of q attributes from $\{a_1, \ldots, a_M\}$;
3. run Algorithm 1 with $D = D^{bs}$ using attributes A;
4. repeat Steps (1)–(3) Q times;
5. return the hyperboxes not dominated on the validation set D^{val}.

Algorithm 3 is a formalization of PRIM with bumping. We observe that the word "bagging" used in [KC16] to describe this method is misleading. Since no averaging over several models takes place, the original term *bumping* is correct [FF99, HTF09].

BestInterval. Mampaey et al. [MNFK12] propose a BestInterval[7] Algorithm 4. It takes a subgroup description B and iteratively refines it considering one dimension at a time. In

[7] In our formalization we have changed originally half-open intervals to closed for consistency. This makes no difference when discretization is exhaustive ($V = U$) as we assume in our work.

Algorithm 3: PRIM with bumping

Data: D, D^{val}, α, *minpts*, q, Q — as described in the text
Result: sequence of hyperboxes

```
1 boxes = {};
2 for 1 < i < Q do
3     D^bs = bootstrap sample from D;
4     A = {a_{j_1}, ..., a_{j_q}} — random subset of q attributes;
5     D^bs_A = D^bs with inputs in A;
6     boxes_i = PRIM.peel(D = D^bs_A, ...);
7     boxes = append(boxes, boxes_i);
8 boxes = non.dominated(boxes);
9 return boxes
```

contrast to PRIM, BestInterval returns the best refinement for a given attribute at each iteration. That is, one can think about BestInterval as a greedy counterpart of PRIM. For a given attribute a the algorithm iterates over the sorted set of points $V = \{v_1, \ldots, v_z\}$ (Line 4), maintaining the best interval found so far (Line 12).

In short, BestInterval uses the following observation. Assume one has two subgroup descriptions B_1 and B_2 different only in the range of the attribute a — it is $[a^{l_1}, a^r]$ for B_1 and $[a^{l_2}, a^r]$ for B_2. Denote by $\tilde{B}_1$ and $\tilde{B}_2$ the hyperboxes which differ from B_1 and B_2 by the upper bound of the same attribute. That is, the respective intervals are $[a^{l_1}, \tilde{a}^r]$ for $\tilde{B}_1$ and $[a^{l_2}, \tilde{a}^r]$ for $\tilde{B}_2$, where $\tilde{a}^r > a^{l_1}, a^{l_2}$. Then the following holds:

$$WRAcc(D, S_{B_1}) - WRAcc(D, S_{B_2}) = WRAcc(D, S_{\tilde{B}_1}) - WRAcc(D, S_{\tilde{B}_2}).$$

The observation is based on the following property (Property 4 in [MNFK12]).

Property 2.3.1 *WRAcc is an additive property, i.e., for any two hyperboxes B_1 and B_2 with $S_{B_1} \cap S_{B_2} = \emptyset$, it holds that*

$$WRAcc(D, S_{B_1} \cup S_{B_2}) = WRAcc(D, S_{B_1}) + WRAcc(D, S_{B_2})$$

To discover a subgroup, following Mampaey et al. [MNFK12], we use a beam-search heuristic [FGL12] as Algorithm 5 describes. It iteratively refines the initial box $\prod_{j=1}^{M}(-\infty, +\infty)$ with BestInterval algorithm. The parameter m regulates the search depth, i.e., the number of iterations, equivalently, the maximal number of attributes describing subgroup (Line 2). The beam size bs defines the number of subgroup descriptions kept at each iteration (Line 6). The function Keep.Best(*BSet*, bs) returns bs hyperboxes with maximal *WRAcc* on D from the set *BSet*.

2.4 Related Ideas

In this section, we present the ideas and findings in other domains which have inspired ours. We explain the similarities and differences between them and our ideas. Section 2.4.1

Algorithm 4: BestInterval

Data: D, B, a, V — as described in the text
Result: a refined hyperbox

$B_{res} = \text{set}(B, a^l = -\infty, a^r = +\infty)$;
$WRAcc_{max} = WRAcc(D, S_B)$;
$h_{max} = v_{max} = -\infty$;
for $1 \leq i \leq z$ **do**
    $B_i = \text{set}(B, a^l = v_i, a^r = +\infty)$;
    $h = WRAcc(D, S_{B_i})$;
    if $h > h_{max}$ **then**
        $h_{max} = h$;
        $v_{max} = v_i$
    $B_i = \text{set}(B, a^l = v_{max}, a^r = v_i)$;
    if $WRAcc(D, S_{B_i}) > WRAcc_{max}$ **then**
        $B_{res} = B_i$;
        $WRAcc_{max} = WRAcc(D, S_{B_i})$
return B_{res}

describes the research where an intermediate statistical model is used to produce a new dataset for training the final model. Section 2.4.2 presents alternative methods to enlarge original data by creating artificial examples. Section 2.4.3 summarizes research related to semi-supervised learning.

2.4.1 Rule Extraction and Knowledge Distillation

Huysmans et al. [HBV06] define the task of rule extraction as follows, "Given an opaque predictive model and the data on which it was trained, produce a description of the predictive model's hypothesis that is understandable yet closely approximates the predictive model's behavior". A wide variety of rule extraction methods has been developed over the past decades. For instance, Trepan [CS95], one of the most known methods, builds m-of-n decision trees (DT) using an opaque model, an artificial neural network (ANN) originally, as an oracle. The resulting tree was found to be more accurate than DT learned directly from the data used for ANN training, on four datasets. The CMM algorithm [Dom97] works similarly, with C4.5 rules instead of DT and an ensemble of C4.5 rules instead of ANN. Fortuny and Martens [dFM15] in a similar study discovered that C4.5-rules and Ripper rule learning algorithms had higher accuracy and comprehensibility when were trained on the datasets augmented by the output of SVM, ANN and random forest rather than using original train data alone.

Rule extraction can be treated as a subdomain of a larger area — model parroting (Section 7.5 in [Set09]) or knowledge distillation [HVD15, XHLL19], where a small and fast model is often used as a student to be faster than the teacher. One exemplary method from this area is model compression [BCN06]. The target is to replace a large model,

Algorithm 5: BestIntervalBS

Data: D, $\{V_j\}$, m, bs — as described in the text
Result: a hyperbox maximizing *WRAcc*

```
BSet = ∏_{j=1}^{M} (−∞, +∞);
repeat m times
    for B ∈ BSet do
        for 1 ≤ j ≤ M do
            BSet = BSet ∪ BestInterval(D, B, a_j, V_j)
    BSet = Keep.Best(BSet, bs)
return Keep.Best(BSet, 1)
```

an ensemble, which is expensive to store and execute but generalizes well, with a much "lighter" and faster model, an ANN, which needs a big labeled dataset to avoid overfitting. The ensemble labels new training data for the ANN. The resulting ANN was more accurate than the one learned from the initial small dataset. A similar methodology is used in [KS13] for object recognition in a semi-supervised setting.

The ideas used above are alike to our idea. However, the distinction lies in the final model to be trained. In the research mentioned, the final model is a "universal" learner, a model which approximates any function with arbitrary accuracy given enough data. Thus, it is natural that the performance of the final model eventually converges to the one of an "intermediate" model. With subgroup discovery, the final model is a hyperbox, and accuracy is no longer the target — WRAcc, recall and precision are of interest.

2.4.2 Data Augmentation

According to [Wu14], "data augmentation is a Markov chain Monte Carlo algorithm for sampling from a Bayesian posterior distribution". This definition has a poor relation to the ideas developed in our work. However, data augmentation has recently acquired a second meaning thanks to the rapid development of artificial neural networks[8]. ANN can achieve excellent performance in various pattern recognition tasks, but often require massive training datasets which are not always available. One can increase the size of a dataset by joining it with examples from another available dataset of the similar domain [GG15] or by creating new (synthetic) examples via applying label-preserving transformations[9] to existing (genuine) ones [CGK15]. For datasets consisting of images, such transformations include zoom, rotation, distortion [BRH19]; for text data, words can

[8] Using the on-line reference for bibliographic information, DBLP [Ley02], one can observe this process of a new meaning emergence. For instance, in 2005, all three entries containing "data augmentation" in their titles use this term with the meaning described above; in 2013, five out of six entries assume this; in 2015, already 11 out of 18 entries imply the second meaning; in the most recent years (20018–2020) hundreds of papers having the term in their titles refer to expanding the training dataset.

[9] This approach can be traced back at least to 2003 [SSP03], but the use of the term "data augmentation" to refer to this method became common only recently.

be replaced by synonyms using embeddings models [WY15]; for speech data — masking a specific frequency channel in the mel spectrogram [PCZ+19].

Our motivation is similar to one of the research just mentioned. However, for the datasets we deal with — tables filled with numeric values, developing a set of label preserving transformations is impossible. For example, observe that with image data augmentation, adding noise most likely will not replace a cat with a dog on a picture. Oppositely, if the dataset results from simulations of system stability, adding noise to inputs of the system may change the stability label. Hence, the methods we use to increase the number of training examples are different.

2.4.3 Semi-Supervised Learning (SSL)

In this work, we deal with the problem of the limited size of labeled data. In Section 5, we focus on real-world (measured) data. There, we apply our method, REDS+, to a semi-supervised problem. This is the case when a dataset contains many points, but only a few of them are labeled. But even beyond semi-supervised setting, we see a certain similarity between REDS/REDS+ and a group of semi-supervised methods called *self-labeling* techniques. These methods refer to exploiting a supervised learning algorithm(s) to obtain enlarged labeled dataset(s) using (most confident) predictions of these algorithms [TGH15]. After a brief definition of semi-supervised learning, we explain two variants of self-labeled techniques — self-training[10] and multi-view learning. We then discuss the similarities and differences between these methods and our idea.

Semi-supervised learning refers to the use of both labeled and unlabeled data for training [Zhu05]. It takes an intermediate place between supervised and unsupervised learning. Indeed, for instance, SSL with generative models can be treated as either classification (supervised learning) with additional information on density or clustering (unsupervised learning) with additional information about the class label of some points [CSZ06]. According to Seeger [See00], semi-supervised learning methods work only when $p(x)$ and $p(y|x)$ share parameters. In the rest of this section, we assume a classification task.

Self-training. Arguably, this is the earliest ([III65]) SSL method, also known as self-learning, decision-directed learning [CSZ06], pseudo-labeling [Lee13, OOR+18], self-teaching or bootstrapping [Zhu05]. With self-training, an ML model is first trained on a set of labeled data. Then this classifier adds some [CSZ06, Zhu05, TGH15] (the most confident) or all [Lee13][11] unlabeled points together with their predicted labels to the training (labeled) data. After it, the classifier is retrained on the enlarged with its own predictions labeled set, and the procedure is repeated. Variations of the procedure just described include, for instance, adding noise [XHLL19].

Multi-view Learning. This method first trains multiple learners (e.g. decision trees, support vector machines, etc.) with the same labeled data. Then these trained models

[10] Chapelle et al. [CSZ06] put equality sign between the terms *self-labeling* and *self-training*.

[11] Oliver et al. [OOR+18] have a different interpretation of this pseudo-labeling algorithm, assuming that only the most confident labels are added.

teach each other using unlabeled data. A similar concept is *co-training* which instead of different learners use different feature sets (views) conditionally independent given the target variable. The concepts of multi-view learning and co-training are often confused in the literature. For instance, Zhu [Zhu05] attributes "tri-training" [ZL05] to co-training, whereas Triguero et al. [TGH15] — to multi-view learning.

Relation to Our Idea. Our idea behind REDS/REDS+ is similar to pseudo-labeling as we create the pseudo-labels for a subgroup discovery algorithm using a machine learning model but is different since (1) we use two different models and (2) our learning procedure is not iterative. Our approaches are also similar to the multi-view learning in a sense that it uses one ML model to teach another, but differ since this learning is directed, e.g. subgroup discovery algorithm is never used to teach an ML model. In general, REDS/REDS+ differ from SSL, since they do not make any assumption on the connection between $p(x)$ and $p(y|x)$. Finally, in a semi-supervised setting, REDS+ may also benefit from SSL methods by using them to train an intermediate ML model, as we will explain.

3 Demonstration — DSGC Analysis

The content of this chapter is based on the following publication.

- Vadim Arzamasov, Klemens Böhm and Patrick Jochem. Towards concise models of grid stability. *In 2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–6, 2018.

We demonstrate the utility of human-comprehensible ML models for understanding the behaviour of a simulated system. We do so by analyzing the data resulting from simulations of Decentral Smart Grid Control — a novel approach for automatic balancing demand and supply in electrical grids. As an additional contribution, we collect explicit and implicit assumptions behind DSGC; this is the most complete list to the best of our knowledge.

Chapter Outline. Section 3.1 describes the research question. Section 3.2 lists the assumptions behind the DSGC model. Section 3.3 describes our methodology. Section 3.4 features the results. Section 3.5 concludes.

3.1 Problem Formulation

Motivation. Electrical grids require a balance between electricity supply and demand to be stable. Conventional systems achieve this balance through demand-driven electricity production. For future grids with a high share of inflexible (i.e., renewable) energy sources however, the concept of demand response is a promising solution. This implies changes in electricity consumption in reaction to electricity price changes. There are different ways to set the price and communicate it to consumers. Conventional approaches, like local electricity auctions, might cause cybersecurity and privacy issues [LXL+12]. The Decentral Smart Grid Control (DSGC) system, proposed recently [SMTW15], [SGA+16], has received much attention. To avoid much of communication it ties the electricity price to the grid frequency so that the price is available to all participants, i.e., all electricity consumers and producers. By that, DSGC introduces real-time pricing, as opposed to, e.g., auctions, where electricity is traded at 15-minutes intervals.

Current models of DSGC come together with assumptions. Some assumptions facilitate simulations of its stability in [SMTW15], [SGA+16], i.e., to infer whether the behaviour of participants in response to price changes destabilizes the grid. To this end, the system is described with differential equations. Following [Kle15], we refer to the variables of the equations as inputs. The current approach to study stability consists of two steps:

1. Assign fixed values to some inputs across all equations and simulation runs.

2. For the other inputs, draw values from a fixed distribution in each experiment (equal across equations).

The result is a set of one-dimensional intervals describing the dependence of stability on tabulated input values [SMTW15], [SGA+16]. For example, the inference from such an analysis carried out in [SGA+16] could look as follows. (We omit some inputs and all units to keep the example simple.)

If $T_j \equiv T = 2$ *and* $\gamma_j \equiv \gamma = 0.25$ *and* $\tau_j \equiv \tau \in [0, 0.7] \cup [2.2, 5]$*, then the system is stable.*

We see two shortcomings of this methodology. First, changing only one input value at a time as in the example, with only input τ varying, leads to the consideration of only a few values of the other inputs (T and γ). This does not facilitate any estimation of interactions among inputs [Kle15]. We refer to this problem as the fixed inputs issue. Second, the assumption of equal input values is not realistic, especially when they are inherent to system participants, e.g., private households, and cannot be regulated externally. Examples of these inputs are price elasticities of energy consumers, i.e., their willingness to change consumption in response to price changes, or the time they need to react to such a change. We call this equality issue.

In this chapter, we demonstrate how scenario discovery can help to solve these issues. We seek a way to analyze the DSGC system for many diverse input values, removing those restrictive assumptions on input values. In other words, our objective is to create metamodels. We want to keep these metamodels simple, hence the term "concise" in the title of this article. — At the same time, we seek new insights regarding the behaviour of the DSGC system.

To deal with the fixed inputs issue, one might allow simultaneous changes of several input values in the simulations. This leads to the question in which form the results of such an analysis should be presented so that they remain comprehensible. An example of this representation for the system could be:

If $T_j \equiv T \in [2.5, 4]$ *and* $\gamma_j \equiv \gamma \in [0, 0.5]$ *and* $\tau_i \equiv \tau \in [1, 3.7]$*, then the system is stable.*

The difference to the first example is that now all conditions are intervals.

A solution to the equality issue would be to allow input values to be different from each other and to be drawn from some reasonable distribution. However, this is not trivial. Think of a system described by N_{eq} equations, each having N_{in} inputs, which already leads to $M = N_{eq} \cdot N_{in}$ degrees of freedom. This means that in general, there will be intervals for all M inputs in the description of stability regions, as follows:

If $T_1 \in [2.5, 4]$ *and* $T_2 \in [1.7, 3]$ *and …, and* $\tau_3 \in [2.2, 4.1]$ *and* $\tau_4 \in [1, 3.7]$*, then the system is stable.*

Clearly, comprehensibility suffers. The problem is more prominent with more inputs.

Contributions. To identify the current limitations of the DSGC, we systematically collect the assumptions behind it. We have already described two, the fixed inputs and the equality issue. But as we will show, there are more. To our knowledge, despite the rising

popularity of this system, a respective comprehensive summary in the literature does not exist.

To deal with the fixed inputs and equality issues, we investigate system stability for different design points and apply a decision tree to the results. To deal with the many inputs and to make the description of stability regions more understandable, we replace the inputs of the system with aggregates, referred to as features. An example rule in the feature space is:

If $\text{avg}(T_j) \in [2.5, 4]$ *and* $\min(\gamma_j) < 0.25$ *and* $\max(\tau_j) > 3$, *then the system is stable.*

We demonstrate that the approach gives way to new insights into the simulated system. For instance, we have learned that fast adaptation generally improves system stability.

3.2 Decentral Smart Grid Control (DSGC)

In this section, we first review the system under consideration proposed in [SMTW15], including the derivation of the physical model and its economic superstructure. Next, we systematically list assumptions and open issues regarding the DSGC. Some assumptions are inherent to the origin of equations describing the system. One cannot remove the premises without changing the equations. Examples are (3.5) and (3.8) below. The other assumptions are ones made when analyzing the system with simulations. An example is that some inputs have specific fixed values. In this study, we want to loosen these restrictions by removing assumptions of the second type.

3.2.1 The Model

The DSGC system consists of two parts. The first, physical part describes the dynamics of generators and loads based on the equations of motion [FNP08], [RSTW12]. The second part is an economic superstructure binding the electricity price to the grid frequency, proposed and studied in [SMTW15], [SGA+16].

Physical Model. Both generators and loads are modelled as rotating machines. Energy conservation laws define the dynamics of a respective system as follows:

$$P^{source} = P^{accumulated} + P^{dissipated} + P^{transmitted} \tag{3.1}$$

That is, the power generated is accumulated in the rotational motion of the generator or dissipated, or transmitted to the loads. Replacing terms in equation (3.1) with respective equations yields:

$$P_j^{source} = \frac{1}{2} I_j \frac{d}{dt} \left(\dot{\delta}_j\right)^2 + K_{Dj} \left(\dot{\delta}_j\right)^2 - \sum_k P_{jk}^{max} \sin\left(\delta_k - \delta_j\right) \tag{3.2}$$

where j is an index of the system participant (load or generator), I is the moment of inertia, K_D is a friction coefficient, P_{jk}^{max} is the capacity of the line connecting Participants j and k. $\delta_j(t)$ is a rotor angle (or phase). [SMTW15] then specifies:

$$\delta_j(t) = \omega t + \theta_j(t) \tag{3.3}$$

where ω is a grid reference frequency (e.g., 50 Hz) and $\theta_j(t)$ is a rotor angle relative to it. Finally, (3.3) is substituted in (3.2) to yield:

$$\frac{d^2\theta_j}{dt^2} = P_j - \kappa_j \frac{d\theta_j}{dt} + \sum_k K_{jk} \sin\left(\theta_k - \theta_j\right) \tag{3.4}$$

where we abbreviate

$$K_{jk} = \frac{P_{jk}^{max}}{I_j \omega}, \qquad \kappa_j = \frac{2K_{Dj}}{I_j}, \qquad P_j = \frac{P_j^{source} - K_{Dj}\omega^2}{I_j \omega}.$$

According to Filatrella et al. [FNP08], for the transition from (3.2) to (3.4) to be correct, the following assumptions must hold:

$$\frac{d\theta_j}{dt} \ll \omega, \qquad \frac{d^2\theta_j}{dt^2} \ll 2K_{Dj}\omega/I_j \tag{3.5}$$

Economic superstructure. The idea of [SMTW15] is binding the electricity price to the grid frequency, with some proportionality factor c_1, and letting participants adjust their production/consumption with price changes. The additional equations are:

$$p_j = p_\omega - c_1 \cdot \int_{t-T_j}^{t} \frac{d\theta_j}{dt}(t - \tau_j)dt \tag{3.6}$$

$$\hat{P}_j(p_i) \approx P_j + c_j \cdot (p_j - p_\omega) \tag{3.7}$$

where p_j is the electricity price for the j-th participant, $\hat{P}_j$ is the power consumed/produced at price p_j, c_j is a coefficient proportional to the price elasticity, p_ω is the electricity price when $d\theta_j/dt \equiv 0$, τ_j is the reaction time, i.e., the time after which one adjusts consumption/production to a price change, T_j is so-called averaging time: The average frequency during Period T_j defines the price. It is assumed that:

$$\sum_j P_j \equiv 0 \tag{3.8}$$

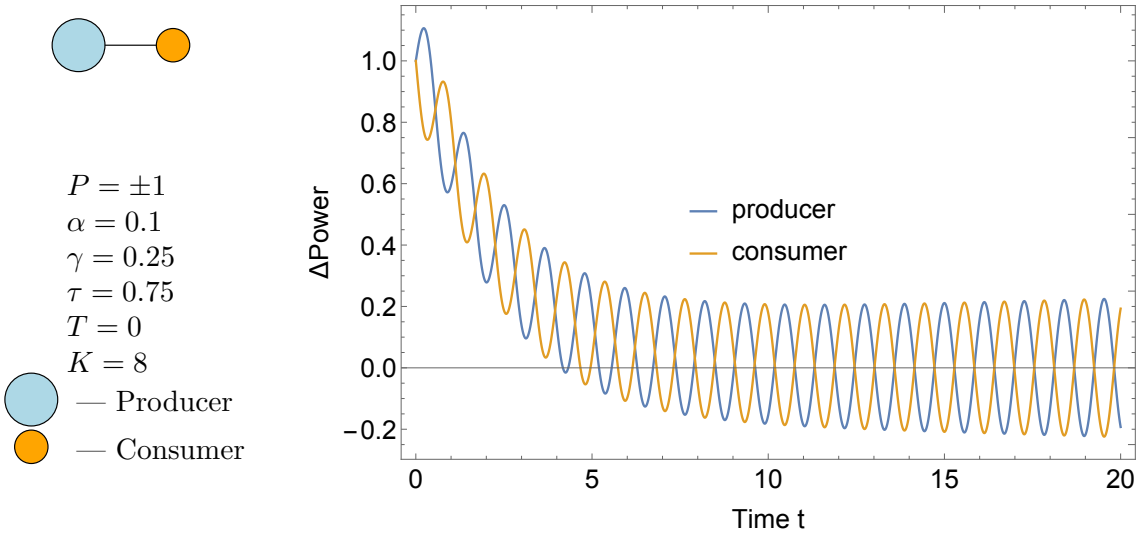
The final equation describing the dynamics of the DSGC results from substituting P_j in (3.4) with $\hat{P}_j$ from (3.7), where p_j is defined as in (3.6), and denoting $\gamma_j = c_1 \cdot c_j$.

$$\frac{d^2\theta_j}{dt^2} = P_j - \kappa_j \frac{d\theta_j}{dt} + \sum_{k=1}^{N} K_{jk} \sin\left(\theta_k - \theta_j\right) - \frac{\gamma_j}{T_j}\left(\theta_j(t - \tau_j) - \theta_j(t - \tau_j - T_j)\right) \tag{3.9}$$

The bottom part of Table 1.1 summarizes the system inputs.

3.2.2 Model Assumptions and Open Questions

In addition to the assumptions in (3.5) and (3.8), we now list the implicit ones. We split the assumptions into three parts, one related to the physical equations, one to the equations describing the economic superstructure and one to the way the system has been studied previously. Additionally, we identify two open questions regarding the system.



(a) System structure

(b) System dynamics after perturbation

Figure 3.1: DSGC system gets destabilized after a perturbation due to overreaction of participants resulting in resonance.

Assumptions behind the Physical Model.

1. Generators are modelled as rotating mass. However, the model is often motivated (cf. [SMTW15], [FNP08], [RSTW12]) by assuming an increasing share of renewables, some of which (PV or wind turbines) do not have any rotational inertia [UBA14].

2. Loads are modelled as rotating mass (synchronous motors)

   a) According to [MWS+14], in power engineering "in many of the applications …passive loads are considered instead of motors…". This is particularly true for modelling households [KBL94].

   b) According to [PTR95], [NM15], "Most of motor loads in the US are indeed induction motors, not synchronous motors."

3. The model completely neglects any so-called control, for instance, active control, as stated in [FNP08].

4. The model assumes constant voltages and constant mechanical power. This limits its validity to short time intervals of around 1s [SPFK13]. However, the model has been explored to study system dynamics during tens of seconds [SMTW15], [SGA+16], [FNP08]. The constant mechanical power also contradicts the intermittent character of wind [Kam14] and solar power plants.

5. Treating variable values as constants in (3.9) implies constant moments of inertia M_j for each participant. However, the inertia in systems with a high penetration of renewables has high variations [UBA14].

Assumptions behind the Economic Superstructure.

1. Adapting the energy consumption to price change does not change the inertia of the system nor the damping constant. This requires further elaboration on how adaptation takes place. For example, if a consumer switches off some device in response to a price change, this generally does have an effect on inertia and the friction in the system.

2. The adaptation of consumption and production happens permanently. That is, load or generation profiles smoothly oscillate with periods of a couple of seconds, see Figure 3.1 or Figure 3 in [SMTW15]. Smooth consumption behaviour again raises the question regarding the mechanism of its adjustment, see the previous Item.

3. Consumers do not learn from the past. This means that after a few oscillations of the price (and grid frequency), the equilibrium level of production or consumption becomes obvious. The better strategy than continuing to adapt to the current price might be to consume or produce at that level. Additionally, resonances often destabilize the system. This means that consumers use more energy when the price is high and less when it is low. This is not rational. See Figure 3.1.

Assumptions Made for Analysis.

1. The values of inputs P_j, γ_j and K_{jk} are fixed.

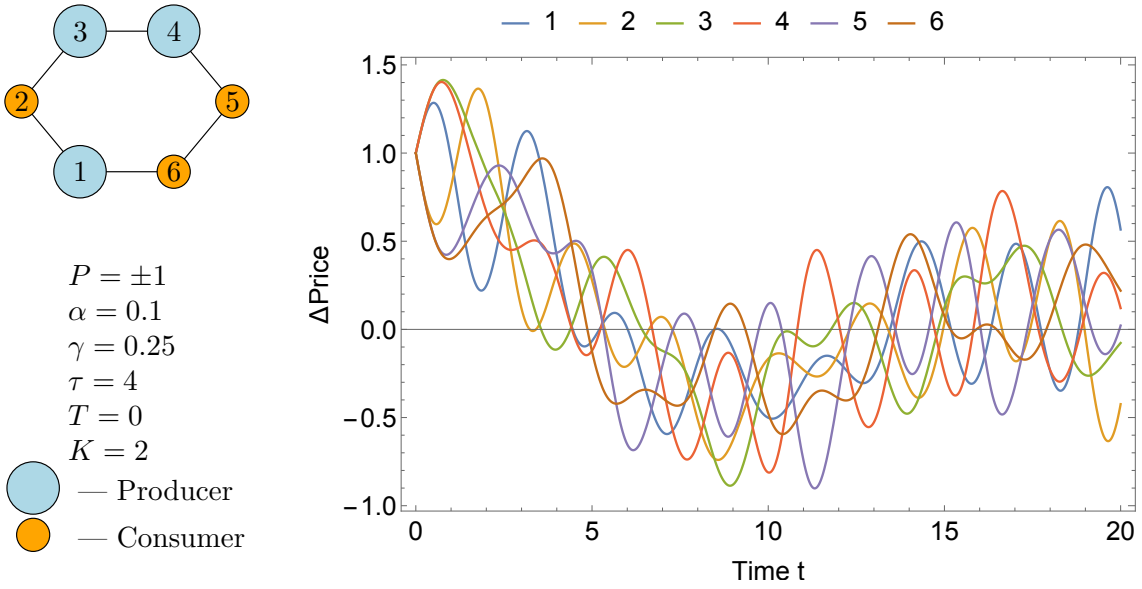
2. The values of inputs are equal for all participants: $X_j = X$, where X_j stands for any input in (3.9), except for P_j, which must satisfy (3.8).

Open Questions.

1. The inventors of the model do not make any statement regarding the scale of model validity. It is unclear to what extent the model is suitable to describe a large country-wise or a small island grid. In other words, should every household be modelled as a separate consumer, or can one consumer represent a bigger unit, e.g., a town?

2. When the reaction of the system to disturbance is analyzed, the dynamics of rotor angles are different for each participant in the stabilization period after disturbance. This means that the prices also are different; see (3.6) and Figure 3.2. Then it is unclear whether the amount of money paid by consumers equals the income of producers.

Equation (3.9) makes sense only if we think of it as a coarse-grained description of the system. But then one must definitely consider a huge heterogeneity of parameters. Thus, in what follows we remove assumptions from paragraph “Assumptions Made for Analysis” and discuss the new insights from this generalization.



(a) System structure

(b) System dynamics after perturbation

Figure 3.2: After perturbation, at each moment of time, price is different at various locations in a DSGC system.

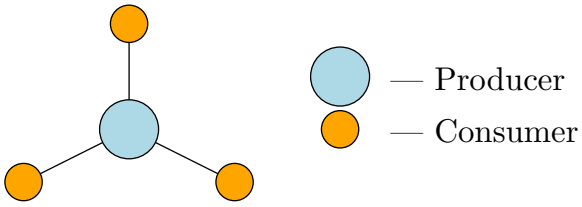


Figure 3.3: System structure

3.3 Methodology

In this section, we first review which input values have been used for DSGC simulations so far and justify our choice of input values, select the data mining model and say how we convert inputs into features. We consider different notions of DSGC stability from literature and select one for further usage.

3.3.1 Input Values

We simulate the four-node star electrical grid with centralized production (Fig. 3.3), studied in [SGA+16] as well. A generator is in the center, and three consumers are connected to it.

DSGC system (3.9) contains six inputs in total. They must be initialized to launch the simulations. The values used in the literature are in Table 3.1. We use the notation $[a, b]$ if the input values are sampled from the interval and a, b if only values a, b are considered.

[1]Power consumed by each load (consumer). Power produced by producer is calculated according to (3.8)

[2]In our experiments below, we specify or choice of the upper bound for τ_j and the range for T_j

Input	[SMTW15], [SGA+16], [RSTW12]	[FNP08]	**Our choice**
P_j[1]	$-1s^{-2}$	$-\{1.5, 2, 2.5\}s^{-2}$	$-[0.5, 2]s^{-2}$
κ_j	$0.1s^{-1}$	$[0.1, 1]s^{-1}$	$0.1s^{-1}$
K_{jk}	$\{4, 8\}s^{-2}$	$\{5, 7, 8, 10, 12\}s^{-2}$	$8s^{-2}$
γ_j	$0.25s^{-1}$	–	$[0.05, 1]s^{-1}$
τ_j[2]	$[0, 10]s$	–	$[0.5, \{5, 10\}]s$
T_j	$\{0, 1, 2, 4\}s$	–	$2s$ or $[0, 4]s$

Table 3.1: Ranges of input variables

According to (3.2)–(3.4), $K_{jk} = K$ implies that $I_j = I$, that is, equal moments of inertia for all participants.

For our analysis, we change the values of some inputs and allow others to take values from a defined range instead of a fixed value. To choose which inputs to vary, we classify them as environmental or control, following [SWN03]. For control inputs, an engineer or system designer can set the values. Here, averaging time T_j, damping constant κ_j, and line capacities K_{jk} are of this type. Environmental (noise) input values depend on the specific user or environment at the time the item is used. Consumed power P_j or reaction time τ_j is environmental. The input γ_j is of mixed type, since in $\gamma_j = c_1 \cdot c_j$ the term c_1 (connecting price to grid frequency) is controllable, and c_j (promotional to price elasticity of participant) is environmental. We fix the values of control inputs and allow the values of the environmental inputs to vary. We also remove the assumption of indistinguishable participants by choosing the input values for each of them distinctly. The nominal power and coefficients γ_j now take values from a range. We have chosen the range of γ_j in line with [Lij07]. Reaction times are no longer equal among participants. The last column of Table 3.1 summarizes this.

3.3.2 Model and Experimental Design

Among other machine learning models, decision trees yield results in the form we target at, exemplified in Section I. Earlier results [Fre13], [HDM+11] confirm that this form of results is comprehensible. To learn decision trees we choose one of the most popular algorithms, CART [BFOS84].

We have defined the ranges of input values and the model. The question arises for which values exactly one should run experiments, i.e., which experimental design to use. Although there is the opinion that one should choose the experimental design together with the statistical model [SPKA01], [WS06], we are not aware of any proof of this being superior for decision trees or classification rules. Intuitively, a space-filling design should be reasonable. We stick to random LHS design [Ste87], [JMK12].

Since the system has symmetries, we hypothesize that a more concise representation of simulation results is feasible based on input aggregates, i.e., features. To create features, we take the minimum, maximum and mean values across all N participants of each input, e.g., $\min(\tau_j)$ for $j = 1, \ldots, N$.

3.3.3 Stability Analysis

There are several types of stability of a system [SGA+16]. We now briefly describe them and list their advantages and limitations.

Stability against Single Perturbations Here one studies the ability of the system to reach an equilibrium state after some perturbation. It can be specified in terms of power when some loads require more power for a short time [FNP08]. This type of stability analysis introduces many new degrees of freedom to the simulations:

- Which nodes of the grid to perturb?
- How exactly does the perturbation look like?
- How long should one observe a system after the perturbation to draw conclusions on its stability?

Basin Stability To study basin stability [MHMK13], one specifies a range of possible perturbations and simulates the system for a set of randomly sampled perturbations from this range. Next, one may estimate the "basin volume" as the ratio of initial conditions converging to a stable operation over the total number of initial conditions [SMTW15], [SGA+16]. Basin stability is more general than stability against perturbation, inheriting all limitations of the latter[3].

Local (Linear) Stability Linear stability analysis explores dynamical stability around the steady-state operation of the grid. It consists of finding roots of the characteristic equation, written as:

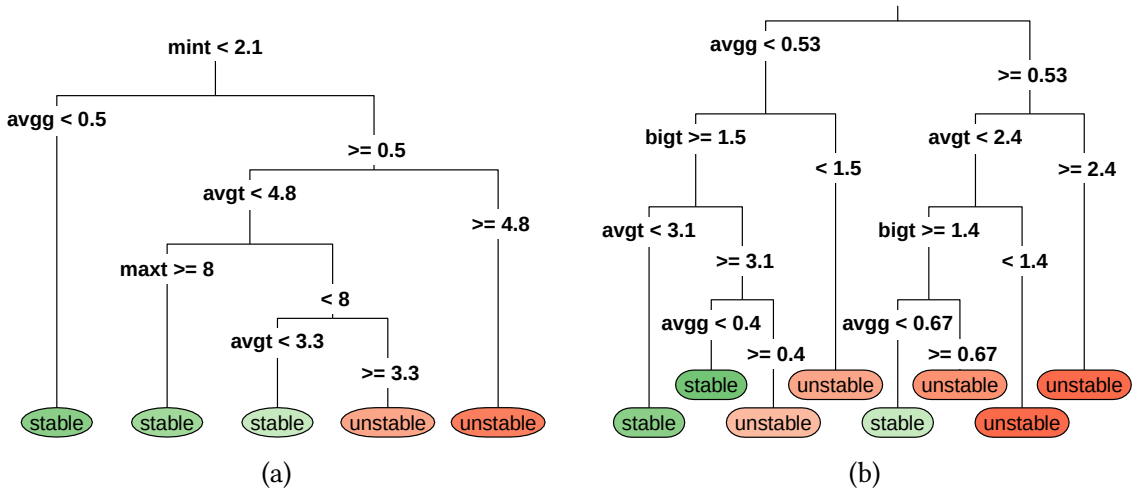
$$\det A = 0 \tag{3.10}$$

A is a $2N \times 2N$ matrix derived from equations of motion (3.9). The equation has infinitely many solutions, but only a finite number of solutions can have a positive real part, and they determine the instability of the system [SGA+16]. To find these roots, a numerical optimization problem is solved. We will use local stability analysis since it is a necessary requirement for all types of stability and does not bring additional degrees of freedom to an already complex system.

3.4 Experimental Results

We now evaluate the usefulness of the proposed approach. To do so, we graph the results of applying the CART algorithm to data from our simulations. This algorithm partitions the input space into regions with higher or lower stability than on average. The paths to the leaves of the decision tree produced by CART define the stable/unstable regions.

[3]We presume that the concept of basin stability is a re-discovery of so-called Starr's domain criterion [Sta63]. According to Herman et al. [HRZC15]: "In practice, this [domain criterion calculation, — author note] is done by calculating the fraction of sampled states of the world in which a solution satisfies one or more performance thresholds". Schneller and Sphicas [SS83] further analyze this criterion and compare with four alternative approaches.



(a) (b)

Figure 3.4: Decision trees on the data from simulations.

To avoid clutter in the plots, we slightly change the notation, replacing γ_j with g and τ_j with t. In the next subsections, we first explore the so-called rebound effect and then show how one can use our approach to set the values for control inputs. At the end of each subsection, we discuss new insights which previous analyses have not revealed.

3.4.1 Rebound Effect

In [SGA+16] the rebound effect has been discovered for a four-node system: For delays $\tau > \tau_c$ ($8s$ when $T = 2s$) the system always is unstable. We investigate whether this effect persists if consumers are heterogeneous. We perform simulations for 10000 design points with the input values specified in the last column of Table 3.1 where we choose $\tau_j \in [0.5, 10]s$ and $T_j = 2s$. Our space of values includes the ones explored in [SGA+16] as a special case[4].

Fig. 3.4a is an excerpt of the decision tree obtained. The path to the second leaf from the left reads as follows: If $\min(\tau_j) < 2.1$ and $\text{avg}(\gamma_j) \geq 0.5$ and $\text{avg}(\tau_j) < 4.8$ and $\max(\tau_j) \geq 8$, then the system is stable. This means that, in a stable grid, a consumer may have a reaction time higher than $\tau_c \approx 8s$ as long as there is a consumer reacting quite fast, and the average reaction time is moderate. Moreover, the presence of a consumer reacting slowly has a positive effect on stability in this case. These results, stemming from our consideration of heterogeneity of participants and discovered with our approach, are new — they have not been insinuated in [SGA+16] in particular.

3.4.2 Defining Values for Control Inputs

According to [SGA+16], there is a trade-off between avoiding the rebound effect with small values of τ_j and increasing the basin stability with high values of τ_j; the suggested value is $\tau \approx 4$. At the same time, higher values of averaging time T have a positive effect

[4]The dataset is available from [DG17]: `https://archive.ics.uci.edu/ml/datasets/Electrical+Grid+Stability+Simulated+Data+`

on stability. We now check to what extent these conclusions hold for the system with diverse participants and sketch a way how our proposed approach could help choosing the values for control inputs. To do so, we allow one control input, T_j, to vary. We assume that $T_j = T$. This is a realistic assumption, since the averaging time can be inherent to the frequency-measuring device, and a policymaker can define it. We simulate the system for 10000 design points with the input values specified in the last column of Table 3.1 with $\tau_j \in [0.5, 5]s$ and $T = [0, 4]s$. As before, this includes the input values explored in [SGA+16] as a special case. Fig. 3.4b graphs the tree. First, the analysis confirms that high values of T are good for stability. Specifically, the stable leaves of the tree only exist for $T \geq 1.4$. This information can be used when designing the system, to, say, regulate the averaging time, prohibiting values lower than 1.5. One can also see that all leaves classified as stable imply that $avg(\gamma_j) < 0.67$. In principle, one could also regulate this by adjusting the parameter connecting price to the grid frequency, c_1, in (3.6). But then the impact on demand response should be assessed. In other words, setting $c_1 = 0$ leads to perfect stability but completely cancels the effect of the economic superstructure.

Our results suggest that values of $\text{avg}(\tau_j)$ smaller than 3.1 contribute to system stability. For $\text{avg}(\tau_j) \geq 3.1$, the stable region (the second leaf), occupies only a small share of the design space. So $\tau \approx 4$ might not be a good value for a system with heterogeneous consumers. We observe that power does not appear in any tree. We speculate that it only has a small or even no impact on stability as long as the physical system is stable. So a takeaway is that the view on the system is more differentiated with the use of decision trees than in [SGA+16].

3.5 Conclusions

Decentral Smart Grid Control, the topic of this article, has been touted as a way to realize demand response. We have collected the assumptions behind it systematically, some of which are restrictive. In order to eliminate some assumptions, while at the same time targeting at simple and insightful models, we have proposed the following:

- Simulate the system for diverse sets of input values. A simulation result has been an inference on whether the system is stable for the specific input values.
- Create features from original inputs to reduce the number of degrees of freedom.
- Apply a decision-tree algorithm to the data resulting from all simulations.

This approach does reveal new insights regarding DSGC, not known from previous studies. For example, we have learned that the system can be stable even if some participants adapt their energy consumption with a high delay, or fast adaptation is preferable for stability under certain conditions.

4 Improving Scenario Discovery — REDS

The content of this chapter is based on the following publication.

- Vadim Arzamasov and Klemens Böhm. 2021. REDS: Rule Extraction for Discovering Scenarios. *In Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21), June 20–25, 2021, Virtual Event, China.* ACM, NewYork, NY, USA, 14 pages. `https://doi.org/10.1145/3448016.3457301`

Decision tree used in the previous chapter partitions the space formed by the inputs of a simulation model into a set of hyperboxes. Often, one does not need to partition the whole space and prefers to have a small number of hyperboxes (called scenarios), but wants each of them to cover a significant share of the interesting examples, minimize the coverage of uninteresting ones and not to restrict the inputs with little effect on the output. The respective metrics are recall, precision (see Section 2.3.2 for formal definitions) and interpretability [BL10]. Patient Rule Induction Method (PRIM) is a conventional approach used to find such scenarios. In this Chapter we deal with the problem of improving scenario discovery with PRIM from small datasets.

Chapter Outline. Section 4.1 introduces the research question. Section 4.2 introduces our approach and justifies it from a statistical point of view. Section 4.3 supports our statistical considerations with experiments. Sections 4.4 and 4.5 describe the quality metrics used and the experimental setup. Section 4.6 features the results. Section 4.7 concludes.

4.1 Problem Formulation

Since simulations often are computationally expensive [WS06], one wants to obtain a good scenario with few simulations. In this chapter, we address the problem of reducing the number of simulation runs to obtain high-quality scenarios or, equivalently, increasing the quality of scenarios discovered from a limited number of runs. As we will show, quality increases slowly with the number of runs with PRIM. It starts from ϕ_0 and approaches some saturation level ϕ_s for a big dataset, forming a *learning curve*. On the other hand, some ML models like random forest can learn a good approximation of a simulated model already from a small dataset but hide their internal logic, i.e., humans cannot easily interpret it. These ML models can be used to inexpensively label more data, which later serves as input to the scenario-discovery method, making the learning curve more concave for small datasets (Figure 4.1). Combining PRIM with a powerful ML model is our main innovation.

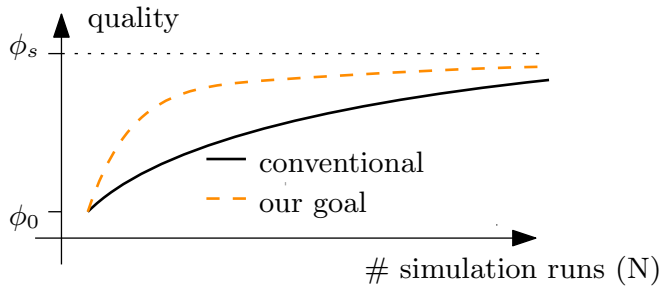


Figure 4.1: Learning curves for scenario discovery tools.

Algorithm 6: Our method: REDS

Data: D, D^{val}, α, B_0, *minpts*, L, AM — described in the text
Result: sequence of nested hyperboxes

```
f^am = train(AM, D);
D^new = L × (M + 1) matrix consisting of rows d_i^new, i ∈ {1, ..., L};
for 1 < i < L do
    x_i = new.point();
    y_i = predict(x_i, f^am);
    d_k^new = (x_i, y_i);
boxes = PRIM.peel(D = D^new, D^val = D, ...);
return boxes
```

Put differently, we propose a new scenario discovery process, REDS (Rule Extraction for Discovering Scenarios), by introducing an intermediate step of estimating an accurate metamodel and using it to obtain a larger dataset to learn an interpretable model. Learning an interpretable model describing a more "complex" one is called *rule extraction* [HBV06]. We then come up with an analysis leading to the expectation that REDS works better than conventional scenario-discovery approaches from a statistical point of view. We compare our method to existing approaches with extensive experiments, using data from simulations of the electrical grid and 32 benchmark functions used in the metamodeling literature.[1] REDS improves scenario discovery significantly. In contrast to PRIM, REDS can benefit from existing active learning techniques to bring down the number of simulations needed to discover scenarios even further, as we will explain.

4.2 Proposed Method: REDS

In this section we describe our method; from a statistical point of view we show why original PRIM has difficulties in learning functional scenarios from small datasets, and how our method, REDS, can overcome them. Then, we discuss the compatibility of REDS with the existing active learning methods.

[1] The code required to reproduce our experiments and figures: `https://github.com/Arzik1987/SDRE`

The novelty of REDS is the introduction of an intermediate step to train a metamodel with low variance and generalization error (Algorithm 6). The proposed process consists of the following steps.

1. Use D to train an accurate metamodel AM (Line 1);
2. uniformly sample L new points from the same area as D (Lines 3–4);
3. label these points using the trained metamodel f^{am} to form a new dataset D^{new} (Lines 5–6);
4. run Algorithm 1 with D^{new} instead of D, and D as D^{val}.

In the rest of the paper, we use random forest [Bre01] as the intermediate metamodel, since it performs well in various classification tasks [WAF16].

4.2.1 Statistical Intuition.

Let $f(x) : \mathbb{R}^M \to \mathbb{R}$ denote the function described by the simulation model; $f(x_i) = p(y_i = 1|x_i)$, where x_i is a vector of inputs' values and y_i — respective model output. In each iteration k of the peeling stage, PRIM compresses the B_k to obtain the box B_{k+1}. It chooses B_{k+1} from candidate boxes — we name them B_{jk}, $j = 1, \ldots, 2M$ — so that the mean value of f in it is maximal. Equivalently, the mean value of f in the box $b_{jk} = B_k \setminus B_{jk}$ which is "peeled out" is minimal. In reality, f is unknown, and its mean μ_{jk} in b_{jk} is estimated from the sample of points contained in b_{jk}. A high error of this estimate may result in cutting off the wrong box — the one which does not maximize the mean value of f in B_{k+1}. That is, our method will make fewer wrong cuts if its error in estimating μ_{jk} is smaller than with the original method.

Let $b \in \{b_{jk}\}$. The mean value of f in b is

$$\mu = \frac{\int_b f(x)p(x)\,\mathrm{d}x}{\int_b p(x)\,\mathrm{d}x}. \tag{4.1}$$

Here $p(x)$ stands for the pdf of the M-dimensional random variable X denoting the point in the input space. In the following analysis, we assume b to be fixed. It contains $n' = \alpha \cdot (1-\alpha)^k \cdot N$ points labeled with y_i, $i = 1, \ldots, n'$ by means of simulations. The estimate of mean μ from the data then is

$$\hat{\mu} = \frac{1}{n'} \sum_{i=1}^{n'} y_i \tag{4.2}$$

The mean squared error (MSE) of this quantity is [FF99]

$$MSE_O = \mathbb{E}[\mu - \hat{\mu}]^2 = (\mu - \mathbb{E}\hat{\mu})^2 + \mathbb{E}[\hat{\mu} - \mathbb{E}\hat{\mu}]^2 = [\mathrm{Bias}(\hat{\mu})]^2 + \mathrm{Var}(\hat{\mu}) \tag{4.3}$$

Here the expectation is taken over all datasets D with $|D| = N$ containing points i.i.d. from $p(x)$. Under deep uncertainty, one accepts the uniform distribution of model inputs, i.e., $p(x) = \mathrm{const}$, and samples x^i i.i.d. from it. Assuming that there is no noise induced by the simulation process, $\hat{\mu}$ is an unbiased estimate of μ. Remember that y_i takes values

from $\{0, 1\}$. Thus, y_i is a Bernoulli random variable with $\Pr(y_i = 1) = \mu$ within the box b. Written formally,

$$\text{Bias}(\hat{\mu}) = 0, \quad \text{Var}(\hat{\mu}) = \frac{\mu(1-\mu)}{n'} = MSE_O \tag{4.4}$$

With our method, a function f^{am} learned with metamodel AM is used to label points. Let μ^{am} be the mean value of f^{am} within b and

$$\hat{\mu}^{am} = \frac{1}{l} \sum_{i=1}^{l} y_i^{am} \tag{4.5}$$

be its estimate where $y_i^{am} = f^{am}(x^i)$, and $l \approx n' \cdot L/N$ is the number of new points inside b; see Section 1.6 for an explanation of L.

Assume first that $y_i^{am} \in \{0, 1\}$. Then $\Pr(y_i^{am} = 1) = \mu^{am}$ (within the box b). In general, $\mu^{am} \neq \mu$. For a fixed function f^{am}, analogously to (4.3)–(4.4), the bias-variance decomposition of the mean squared error is

$$[\text{Bias}(\hat{\mu}^{am})]^2 = (\mu - \mu^{am})^2, \quad \text{Var}(\hat{\mu}^{am}) = \frac{\mu^{am}(1-\mu^{am})}{l} \xrightarrow{l \to \infty} 0 \tag{4.6}$$

where the expectation was taken over all datasets D^{new} (Algorithm 6, Line 2) that are possible with our approach. The MSE with our method using metamodel AM for large L is

$$MSE_{AM} = \mathbb{E}[\mu - \mu^{am}]^2 \tag{4.7}$$

where the expectation is taken over all feasible datasets D as in (4.3), and all possible fits f^{am} of a given metamodel AM obtained on them.

Now we can compare the MSE_O obtained with the original approach (4.4) with MSE_{AM} with our approach (4.7). Assuming that the best scenario is the one discovered with PRIM knowing the true function f, our method will perform superior if, for all possible boxes b, $\mathbb{E}[\mu - \mu^{am}]^2 < \mu(1-\mu)/n'$. Similarly, our method is *likely* to show better performance than original PRIM if the above inequality holds for the *majority* of boxes. Note that the left-hand side of the inequality implicitly depends on N, as increasing the size of a training set typically leads to higher accuracy of f^{am} and lower value of $\mathbb{E}[\mu^{am} - \mu]^2$. Now consider the model AM which outputs class probability estimates instead of classes, i.e., $y_i^{am} \in [0, 1]$. Interestingly, in this case, our approach may outperform the original one even when the size L of the new dataset D^{new} is comparable to the size N of the initial dataset D. Specifically, the following holds.

Proposition 4.2.1 *If $n' = l \wedge \mu = \mu^{am}$, then $Var(\hat{\mu}^{am}) \leq Var(\hat{\mu})$*

Proof: Since $n' = l$, it is sufficient to show that

$$\mathbb{E}[y_i^{am}]^2 - (\mathbb{E}y_i^{am})^2 = \text{Var}(y_i^{am}) \leq \text{Var}(y_i) = \mu(1-\mu) \tag{4.8}$$

Since $\mathbb{E}y_i^{am} = \mu$, (4.8) $\iff$ $\mathbb{E}[y_i^{am}]^2 \leq \mu$. This is true as

$$\mathbb{E}[y_i^{am}]^2 = \int_0^1 (y_i^{am})^2 g(y_i^{am})\, \mathrm{d}y_i^{am} \leq \int_0^1 y_i^{am} g(y_i^{am})\, \mathrm{d}y_i^{am} = \mu. \tag{4.9}$$

Here $g(y_i^{am})$ is the pdf of y_i^{am} implied by the restriction of f^{am} to the box b. The latter inequality holds since $y_i^{am} \in [0, 1]$ and $g(y_i^{am}) \geq 0$. □
However, the condition $\mu = \mu^{am}$ of the above proposition does not hold for all possible boxes b, unless $f \equiv f^{am}$. We experiment with the performance of our approach in case $L = N$ in Section 4.6.2.

Finally, when the simulation process is imperfect and introduces noise, in general, $\Pr(y_i = 1) \neq \mu$. Consequently, $\text{Bias}(\hat{\mu})$ in (4.4) is no longer zero, and the analysis becomes more sophisticated. We evaluate the influence of noise experimentally (Section 4.6.2).

4.2.2 Discussion of the statistical derivations.

To avoid restrictive assumptions on the true function f, we made certain simplifications. First, we assume that the box b and the number of points it contains, n', are fixed simultaneously, while the points in D are sampled at random. In reality, only n' is fixed at each iteration, and the box boundary varies to include exactly n' points (Algorithm 1, Lines 6–7). Allowing the box boundary to vary with different realizations of D would make MSE estimates (4.4) and (4.6) incomparable. Second, one usually uses so-called space-filling designs [SWN03] to form a dataset D rather than "brute force" random sampling., e.g., Latin hypercube sampling [Kle15]. Generally, this would result in lower variance values than estimated with (4.4) or (4.6). With these simplifications, our analysis explains the experimental results sufficiently well. We further illustrate the intuition behind REDS with the experiments in Section 4.3.

4.2.3 REDS and Active Learning

We propose REDS to decrease the number of simulations needed for learning scenarios, i.e., to minimize the labeling effort. There exist techniques with a similar target, known under the names active learning [Set09], adaptive sampling [GCD+10] or selective sampling [BS19]. Active learning allows the learning algorithm to select the instances to be labeled next iteratively. However, existing sampling techniques are not applicable to PRIM in a straightforward manner. This is because they require some measure of uncertainty, which is not apparent in the case of PRIM. This might be the reason why adaptive sampling was only applied to increase the diversity of scenarios so far [IP16], not the quality of PRIM results. REDS in turn is compatible with active learning. This is because several techniques exist for random forests [BS19], support vector machines [TK01] and others [Set09] which one may use as an intermediate metamodel in REDS.

4.3 Intuition behind REDS: Demonstration

We check the validity of our analysis in Section 4.2 and provide a synthetic example where our method yields much better output than the original one. To this end, we use function #3 from [DHL+13]. Here the label y only depends on two inputs (attributes), a_1 and a_2, as $y = 1$ if $a_1 > 0.6$ and $a_2 > 0.8$. In the end, 0.2% of the labels are inverted, imitating noise. The support of the function is $a_j \in [0, 1]$, $j = 1, \ldots, 5$.

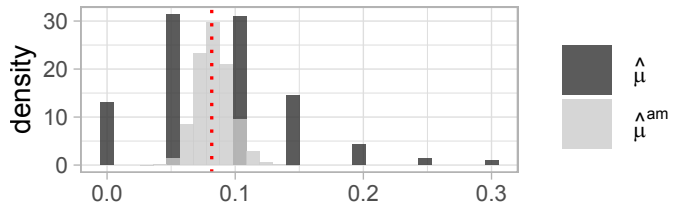


Figure 4.2: The distributions of $\hat{\mu}$ (dark) and $\hat{\mu}^{am}$ (light)

4.3.1 Mean-Squared Error

We describe the general experimental setting and the results obtained in experiments.

MSE: experimental setting. In our experiment, we created independently 200 datasets D of size $N = 400$ labeled with the selected function. For each dataset D_i, $i = 1, \ldots, 200$, we trained one random forest model, f_i^{am} and let it label 100 datasets D_{ij}^{new}. For a fixed box b we calculate MSEs:

$$
\begin{aligned}
MSE_O = \mathrm{Var}(\hat{\mu}) &= \sum_{i=1}^{200} (\hat{\mu}_i)^2/200 - \left(\sum_{i=1}^{200} \hat{\mu}_i/200\right)^2, \\
MSE_{AM} &= \sum_{i=1}^{200} \left(\mathrm{Bias}(\hat{\mu}_i^{am}) + \mathrm{Var}(\hat{\mu}_i^{am})\right) \\
\mathrm{Bias}(\hat{\mu}_i^{am}) &= \mu_{gt} - \sum_{j=1}^{100} \hat{\mu}_{ij}^{am}/100, \\
\mathrm{Var}(\hat{\mu}_i^{am}) &= \sum_{j=1}^{100} (\hat{\mu}_{ij}^{am})^2/100 - \left(\sum_{j=1}^{100} \hat{\mu}_{ij}^{am}/100\right)^2
\end{aligned}
$$

Here we use indexes i for enumerating datasets D_i and j for indexing datasets D_{ij}^{new}, and keep the other notations as in Section 4.2.1. We have also substituted μ in (4.6) with

$$\mu^{gt} = \frac{n_{gt}^+}{n_{gt}}$$

where n_{gt}^+ and n_{gt} are calculated (see Section 1.6) for a subgroup S_b^{gt} defined on a large dataset D^{gt} of 10^6 points labeled with the selected function.

MSE: results We use random forest, which outputs probabilities, as a metamodel. We set $N = 400$ and estimate the MSE of the μ estimate for box b defined as $a_3 > 0.95$ ($\alpha = 0.05$). The expected number of points in this box is $n = \alpha \cdot N = 20$, and $\mu \approx (1-0.6)(1-0.8) = 0.08$. We neglected the noise in the last calculation. The variance of the original method, calculated using (4.4), is $\sigma = 0.08 \cdot (1-0.08)/20 = 3.68 \cdot 10^{-3}$. MSE_O obtained in experiments is $3.54 \cdot 10^{-3} \approx \sigma$, whereas MSE_{AM} are $2.9 \cdot 10^{-3}$ and $0.19 \cdot 10^{-3}$ for $K = 400$ and $K = 10^5$

respectively. That is, $MSE_O > MSE_{AM}$. This result holds for eight (for $K = 400$) and nine (for $L = 10^5$) boxes out of ten candidates for cutting off at the first iteration of the peeling step of PRIM. Figure 4.2 illustrates the experiment. It shows distributions of $\hat{\mu}$ and $\hat{\mu}^{am}$ ($L = 10^5$) for b; the dotted line is the true μ.

MSE: additional experiments We conducted the experiments described above with three more functions: #8 from [DHL+13], "linketal06simple" from [SB13] and "morris" from R package "sensitivity". In all experiments, we calculated MSEs of the mean estimates of labels in $2 \cdot D$ boxes, considered as candidates for cutting off at the first iteration of PRIM. When the metamodel is random forest, which outputs class labels, we have obtained $MSE_O > MSE_{AM}$ (when $L = 10^5$) for the majority of boxes for the first two functions. This is similar to the result described above. For function "morris" for all $2 \cdot M = 40$ boxes the relation was opposite: $MSE_O < MSE_{AM}$. However, as we will show (Tables 4.4–4.7), in this case, our method is by far better than the original one.

Let us index these 40 boxes with r. The detailed analysis reveals that μ_r^{gt} for $r \in \{8, 9, 10\}$ are significantly lower than for other values of r. The share of cases, when $\min_r \hat{\mu}_{ir}$ is reached for some $r^* \notin \{8, 9, 10\}$ is 26%. Whereas, the share of cases, when $\min_r \hat{\mu}_{ijr}^{am}$ is reached for some $r^* \notin \{8, 9, 10\}$ is only 0.3%. That is, despite greater MSE, our method makes less irrelevant cutoffs. This is because, for each f_i^{am}, the bias does not change the ranking of boxes by the mean value of labels of points they contain. All this might indicate, that the mechanism we described in Section 4.2.1 is not the unique one responsible for better performance of our approach.

4.3.2 Comparing Scenarios

For this study, we set $N = 100$. Figure 4.3 graphs the result obtained with original PRIM and with our method (solid thin lines). Four dimensions $\{a_1, \ldots, a_4\}$ are presented. Dashed thick lines represent the true scenario.

The top two plots show the 100 points obtained directly from function #3. White points stand for $y = 0$ and black ones for $y = 1$. The plots reveal that PRIM makes mistakes by cutting the box in the irrelevant space $\{a_3, a_4\}$. It does so for the following reasons:

- There are nine true scenario examples ($y = 1$) — not enough to fill the range of irrelevant attributes $\{a_3, a_4\}$. So there are some areas near axes in these coordinates with no scenario examples, for instance near the axis a_4.

- One point, near the bottom of the left upper plot, is wrongly labeled with $y = 1$ (noisy data). It prevents the algorithm from further restricting the box in this space before all cuts in other dimensions leading to better results are made.

- The algorithm stops when the learned box contains $minpts = 20$ points. Since there are only nine true scenario examples, this box contains at least 11 irrelevant ones. In other words, the box is too big to describe the scenario properly but cannot be further reduced due to a lack of points in D.

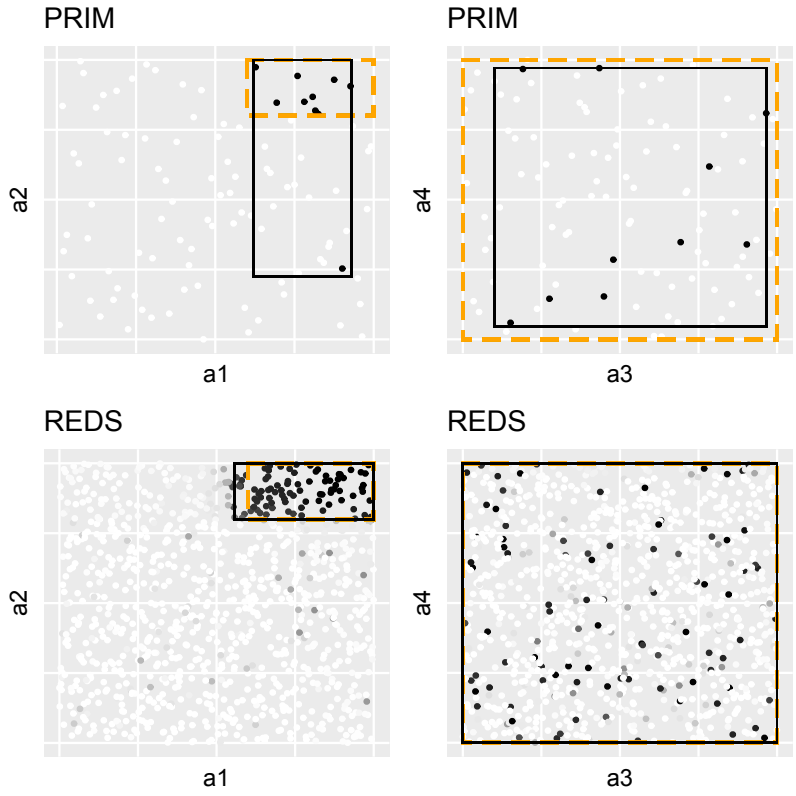


Figure 4.3: The results obtained with original PRIM and with our proposed method on the example dataset.

The bottom two plots graph the 1000 newly generated points labeled with random forest. The color of points stands for their probability of belonging to scenario $y = 1$, changing from white ($\Pr(y = 1) = 0$) to black ($\Pr(y = 1) = 1$). Clearly, this new dataset allows PRIM to learn a better scenario by overcoming the issues of the original dataset just described.

4.4 Quality Metrics

In this section, we describe the quality metrics we use to compare the different algorithms. We assume that to estimate these metrics, one uses a separate data (large) set of points D^{test} labeled by the simulation model which does not overlap with D and D^{val}.

4.4.1 AUpC, precision, Interpretability

As described before, PRIM does not produce a single box. Instead, it outputs a sequence of boxes from which a user can choose one, usually by compromising between precision and recall (see Section 1.6) [BL10]. To exclude this subjective choice from our evaluation, we compute the precision and recall for each box in the sequence. The resulting pairs of values form a curve on the recall-precision plane — the *peeling trajectory* [BL10, FF99]. To rank two algorithms, we compare their curves: *AB* and *AC* on Figure 4.4. If if the peeling

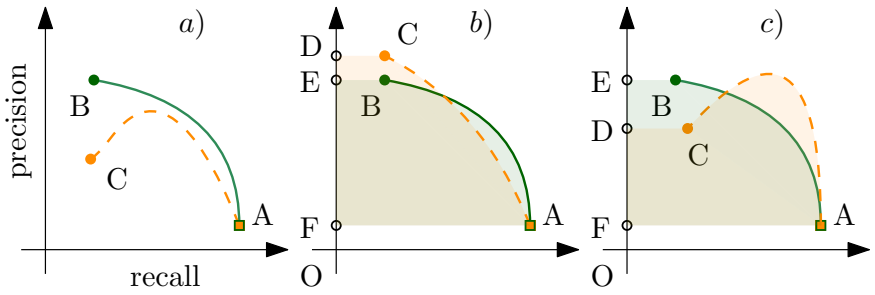


Figure 4.4: Mutual positions of two peeling trajectories

trajectories do not intersect, like in Plot a), the conclusion is clear: the algorithm, which has yielded AB is better. When they cross each other, we propose to use the Area Under the peeling Curve (AUpC) to quantify the quality. This is, we compare the areas covered by figures $ABEF$ and $ACDF$, as shown in Plots b) and c), a larger area corresponds to a better algorithm.

Sometimes one wants to find scenarios as pure as possible, i.e. to maximize the precision while allowing lower recall. Since the test data is not available in reality, this choice can be made using validation data. This corresponds to choosing the last box returned by Algorithm 1. Thus, we will also compare the precision of the last boxes produced by different methods.

Interpretability is another quality metric often used in scenario discovery. A hyperbox is already deemed an interpretable model output [Fre13, HDM+11]. However, one can additionally compare scenarios by the number of dimensions restricted by the hyperboxes defining them [BL10, KC16]. As with precision, we will make such a comparison with the last boxes returned by each algorithm.

4.4.2 Consistency.

Different datasets produced by a simulation model can result in different scenarios discovered. This reduces interpretability [FF99] since one looks for some hidden structure in the model, rather than in the particular data produced by it. So we introduce the fourth quality measure for scenarios — *consistency*.

Definition 4.4.1 *For two datasets D_1 and D_2, $|D_1| = |D_2|$ produced with the same model, let B_1 and B_2 be the scenario descriptions obtained with the same scenario discovery algorithm SDA. Let V_o be the volume of the overlap of these boxes and V_u the one of the union of B_1 and B_2. The consistency of SDA is*

$$consistency = \mathbb{E}\left[\frac{V_o}{V_u}\right]$$

When the definition of any box B_1 or B_2 includes unbounded intervals ($a_j^l = -\infty$ or $a_j^r = +\infty$), consistency calculated with this definition would be either 0 or an indeterminate form, i.e., not meaningful. To overcome this issue, we take advantage of our perfect knowledge of pdf $p(x)$ and replace infinities in the box definitions with minimal and maximal values of the respective attribute as implied by the support of $p(x)$.

Consistency is often used in the rule learning literature, but with different definitions. For instance, [HBV06] defines it as the inherent randomness of the algorithm, not the one resulting from the training data.

4.5 Experimental Setup

For our experiments, we use 32 benchmark functions, most of which are commonly used in the metamodeling domain, and one simulation model, the decentral smart grid control (DSGC). We describe these data sources and the methodology of the experiments.

4.5.1 Data Sources.

We have used different sources of functions. First, we have implemented functions 1–8 and 102 from [DHL+13] using the descriptions in that paper[2]. All these functions are "noisy" with probability 0.2% of a false label.

The functions "morris" and "sobol" are implemented in the R package "sensitivity"[3]. We also used the R implementations of the other functions from [SB13]; they are used to test input variable screening and sensitivity analysis in simulations. We keep the original names of the functions as provided by the implementation. We converted continuous function output to binary by specifying the threshold *thr*, so that $y = 1$ if the output is below *thr* and $y = 0$ otherwise; this is common in scenario discovery [BL10]. Next, we simulate DSGC system (see Section 3.2.1) of a particular structure presented in Figure 4.5. It is a "star"-like system, containing four consumers and one producer — in the center. We fixed the values of some inputs and sampled the others from the given ranges uniformly independently. See Table 4.1. In total, there are 12 inputs that vary: γ, T and τ for each electricity consumer. We also introduced the function "ellipse" as follows:

$$y = \sum_{j=1}^{15} w_j \cdot (a_j - c_j)^2$$

where

$$w \approx \{0.353, 0.434, 0.899, 0.373, 0.278, 0.164, 0.927, 0.769, 0.975, 0.606, 0, 0, 0, 0, 0\},$$
$$c \approx \{0.975, 0.843, 0.772, 0.325, 0.805, 0.945, 0.221, 0.732, 0.289, 0.6, 0, 0, 0, 0, 0\}$$

Table 4.2 lists all 33 functions used in our study. The columns contain the following information. M is the number of inputs. $Infl \leq M$ is the number of "influential" inputs, i.e., those affecting the output. The "reference" column contains the references to the work using the respective function. A big share of functions we use is from [SB13]. The threshold values are in column "thr". The functions which already output $y \in \{0, 1\}$, have the values "na" (not applicable) in this column. Finally, the expected share of outcomes $y = 1$ with uniform sampling o points from the input space is in column "share".

[2]For functions 1, 3, 5–7, we obtained data similar to that presented in Figure 2 in [DHL+13], but for functions 2, 4 and 8 the plots were different. With function 101, we did not get any examples labeled with $y = 1$ and thus excluded it from our analysis.

[3]`https://cran.r-project.org/web/packages/sensitivity/`

Table 4.1: Input values used for DSGC simulations

Input	P	α	K	γ	τ	T
Consumers	-1	0.1	8	[0.05,1]	[0.5,5]	[1,4]
Producer	4			0	–	–

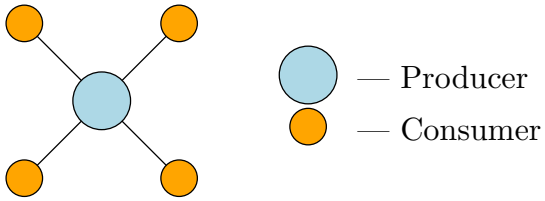


Figure 4.5: DSGC system structure

4.5.2 Hyperparameters.

The algorithms described above — original PRIM, PRIM with bumping and REDS — have certain hyper-parameters in common. We set the peeling parameter to $\alpha = 0.05$, as it is the most common choice in the literature [KC16]. We set the stopping criterion to $minpts = 20$ so that $minpts \cdot \alpha = 1$; and we set $k_{max} = 99$ so that the minimal relative volume of the scenario is at least $(1-\alpha)^{(k_{max}+1)} = 0.6\%$. For all functions, we experimented with datasets of sizes: $N = \{400, 800, 1600\}$. For DSGC the sizes are $N = \{200, 400, 800, 1200, 1600, 2000\}$. We consider two versions of the original PRIM: without pasting — "O", and with the pasting step — "O.p". For "O.p", the pasting parameter is $\beta = 0.01$ and apply the pasting step (Algorithm 2) to each box resulting from "O". The hyper-parameter values for PRIM with bumping were not stated in the respective paper [KC16]. We set the number of iterations to $Q = 50$. We consider two variants of this algorithm which differ in the number of attributes q used in each iteration. "B" uses $q = \lceil\sqrt{M}\rceil$ attributes, the common value used in random forest for classification, where M is the number of attributes (see Section 1.6); "B.all" uses all attributes, i.e., $q = M$. By default, we set the number of new points generated and labeled with REDS to $L = 10^5$, i.e., much larger than any N used. As mentioned, we use random forest as the "complex" metamodel in Algorithm 6. To set its hyper-parameters, we use the default hyperparameter optimization procedure of package "caret"[4]. Random forest can return class labels (0 or 1) or class probabilities for the new L points, yielding two variants of REDS, "RF.l" and "RF.p". Table 4.3 summarizes the above.

4.5.3 Design of Experiments.

To form the datasets $D = D^{val}$, Halton sequence [HS64] for DSGC simulations and Latin hypercube sampling for the other functions. For each function, we generate the test data D^{test} containing 10^4 points. We run the experiment 50 times for each function and each value of N and average the values of AUpC, precision and interpretability. We compute

[4] `http://topepo.github.io/caret/index.html`

Table 4.2: Functions for experimental study

function	M	Infl	reference	thr	share (%)
1	5	2	[DHL+13]	na	47.6
2	5	2	[DHL+13]	na	25.7
3	5	2	[DHL+13]	na	8.2
4	5	2	[DHL+13]	na	18
5	5	2	[DHL+13]	na	8
6	5	2	[DHL+13]	na	8.1
7	5	2	[DHL+13]	na	35
8	5	2	[DHL+13]	na	10.9
102	15	9	[DHL+13]	na	67.2
borehole	8	8	[SB13, AO01]	1000	30.9
dsgc	12	12	Section 4.5	na	53.7
ellipse	15	10	Section 4.5	0.8	22.5
hart3	3	3	[SB13]	−1	33.5
hart4	4	4	[SB13, P+13]	−0.5	30.1
hart6sc	6	6	[SB13, P+13]	1	22.6
ishigami	3	3	[SB13, IH90]	1	25.5
linketal06dec	10	8	[SB13, LBH+06]	0.15	25.3
linketal06simple	10	4	[SB13, LBH+06]	0.33	28.5
linketal06sin	10	2	[SB13, LBH+06]	0	27.2
loepetal13	10	7	[SB13, LWM13]	9	38.9
moon10hd	20	20	[SB13, Moo10]	0	42.1
moon10hdc1	20	5	[SB13, Moo10]	0	34.2
moon10low	3	3	[SB13, Moo10]	1.5	45.6
morretal06	30	10	[SB13, MMM06]	−330	34.5
morris	20	20	[S+00]	20	30.1
oakoh04	15	15	[SB13, OO04]	10	24.9
otlcircuit	6	6	[SB13, BAS07]	4.5	22.5
piston	7	7	[SB13, BAS07]	0.4	36.8
soblev99	20	19	[SB13, SL99]	2000	41.3
sobol	8	8	[S+00]	0.7	39.2
welchetal92	20	18	[SB13, WBS+92]	0	35.6
willetal06	3	2	[SB13, WHG+06]	−1	24.9
wingweight	10	10	[SB13, FSK08]	250	37.8

Table 4.3: Experimental setting

	B	B.all	O	O.p	RF.l/RF.p
α	0.05				
$N = \|D\|$	$\{400, 800, 1600\}$ ($\{200, 400, 800, 1200, 1600, 2000\}$ — for DSGC)				
minpts	20				
β	×	×	×	0.01	×
q	$\lceil\sqrt{M}\rceil$	M	×	×	×
Q	50		×	×	×
L	×	×	×	×	100000

consistency for each pair of last boxes from different runs and average the results, similarly to the approach in [Dom97] to compute stability, a measure akin to consistency.

4.6 Results

We compare the performance of approaches across all functions and further experiment with DSGC to study how the randomness of the sampling forming the dataset D, the values of N and L, and the level of noise, influence the performance of the methods.

4.6.1 Performance across all Functions.

We present the results for the four metrics in a form which we now explain, taking AUpC as an example. First, for each function and each value of $N = |D|$, we averaged AUpC values across 50 experiments. Figure 4.6 plots these averages; each box is based on the 33 functions under consideration. The average AUpC values over 50 experiments and 33 functions for each dataset size, N, are in the row "avg" in Table 4.4. For a rather high number of points, $N = 1600$, the differences in these values between original PRIM and our methods are low. But it remains high for particularly complex models, which often require much time per simulation run — see the last row of the Table, presenting the results for 20-dimensional "morris" function. For each function, we also recorded which methods performed best and second best. The counts of these numbers are in rows "#1" and "#2" in Table 4.4, respectively. These results show that in general, our method ("RF.l" and "RF.p") outperforms the existing ones in terms of AUpC.

The results for precision are presented in Figure 4.7 and Table 4.5. Again, our method performs well in most cases. The minimal precision achieved by it is much higher than that of competing methods.

Figure 4.8 and Table 4.6 feature results regarding the number of restricted dimensions. Smaller numbers, standing for higher interpretability, are better. The values for "B" are slightly lower. However, the comparison with "B" is unfair, since the number of attributes used in the box definition under "B" is "manually" restricted ($t = \lceil\sqrt{D}\rceil$) and usually does not fall together with the number of inputs actually influencing the simulation result.

Table 4.4: AUpC, all functions

N		B	B.all	O	O.p	RF.l	RF.p
400	avg	42.8	42.4	41.3	41.2	48.7	49.1
	# 1	1	0	0	0	8	24
	# 2	0	1	0	0	24	8
800	avg	46.4	46.7	46.4	46.3	50	50.5
	# 1	0	1	0	0	5	27
	# 2	4	0	0	0	24	5
1600	avg	48.3	49.3	49.1	49	50.8	51.2
	# 1	0	1	0	0	5	27
	# 2	6	0	0	0	22	5
m_{1600}	avg	21.8	20.9	21	21	28.7	28.5

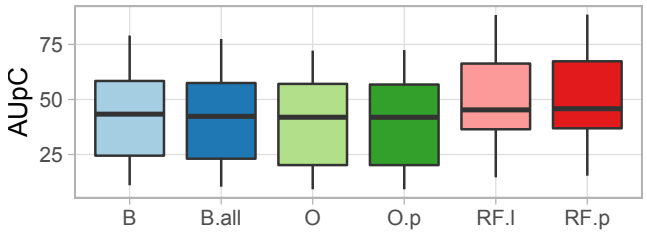


Figure 4.6: AUpC, all functions, $N = 400$

Table 4.8: Number of irrelevant dimensions restricted

N	B	B.all	O	O.p	RF.l	RF.p
400	0.38	2.78	2.78	2.78	0.1	0.37
800	0.29	2.67	2.35	2.36	0.12	0.49
1600	0.28	2.59	2.18	2.19	0.1	0.71

More detailed results are that our method, "RF.l", often approximates the number of inputs affecting the output better than any competitor. To see this, we first selected 19 functions, where the number of influencing inputs $Infl_i$ ($i = \{1, \ldots, 19\}$) is lower than the total number of inputs M_i (see Table 4.2). For each such function, we took the difference between the number of dimensions restricted and $Infl_i$. We then kept only positive values — i.e., irrelevant dimensions are restricted — and averaged the results across functions. See Table 4.8. Clearly, the "RF.l" variant is the leader, providing the fewest irrelevant restrictions.

Finally, Figure 4.9 and Table 4.7 contain results regarding consistency. REDS again is superior. The gap between it and competitors grows with the size of the dataset N.

On average, the levels of AUpC, precision and consistency achieved for $N = 1600$ with conventional methods are comparable to those achieved with $N = 400$ with REDS. That is,

Table 4.5: Precision, all functions

N		B	B.all	O	O.p	RF.l	RF.p
400	avg	80.9	81.7	81.1	80.9	91.4	92.6
	# 1	1	0	0	0	2	30
	# 2	2	1	0	0	28	2
800	avg	86.3	87.1	87	86.9	93.9	95.2
	# 1	0	1	0	0	1	31
	# 2	6	0	0	0	26	1
1600	avg	89.8	91.2	91.1	90.9	95.2	96.6
	# 1	0	1	0	0	1	31
	# 2	10	0	0	0	22	1
m_{1600}	avg	70.3	67.9	69.3	69.2	87.5	92.4

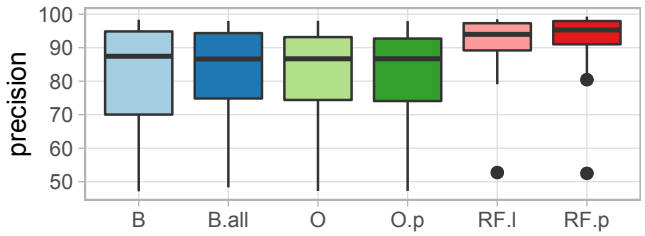


Figure 4.7: Precision, all functions, $N = 400$

our approach can reduce the number of simulation runs by ≈ 75%. The reduction is even more prominent for complex models with many inputs.

4.6.2 Experiments with DSGC

The Variance of the Results and Peeling Trajectories. Figure 4.10 plots quality metrics for DSGC with $N = 400$, obtained in 50 independent experiments with different methods. One can see that REDS yields a *significant* improvement over the alternatives, as measured with AUpC or precision. The method "B" restricts the smallest number of dimensions, as this number is forced to be $q = \lceil \sqrt{M} \rceil = 4$. In the case of DSGC, this restriction is too hard as more attributes do influence the simulation output. I.e., the actual benefit in interpretability with "B" is arguable, while the other quality measures suffer.

Figure 4.11 plots the peeling trajectories for DSGC with $N = 400$ for different methods smoothed across 50 experiments. The curves produced with REDS dominate the ones obtained with competitors — both precision and recall are higher.

Learning Curves. Figure 4.12 shows the learning curves (cf. Figure 4.1) for AUpC, precision and consistency for DSGC. The horizontal axis is logarithmically scaled. AUpC and precision values obtained using REDS are always greater than those of competing methods. For small $N = |D|$, consistency of REDS is a bit lower than that of the original

Table 4.6: Number of restricted dimensions

N		B	B.all	O	O.p	RF.l	RF.p
400	avg	3.3	7.72	7.71	7.71	3.63	4.24
	# 1	15	0	0	0	20	3
	# 2	9	0	0	0	13	7
800	avg	3.25	7.72	7.36	7.37	3.87	4.67
	# 1	15	0	0	0	19	1
	# 2	11	0	0	0	14	6
1600	avg	3.22	7.57	7.14	7.14	3.99	5.04
	# 1	17	0	0	0	17	1
	# 2	13	0	0	0	15	3
m_{1600}	avg	4.98	17.3	16.8	16.9	6.82	10.6

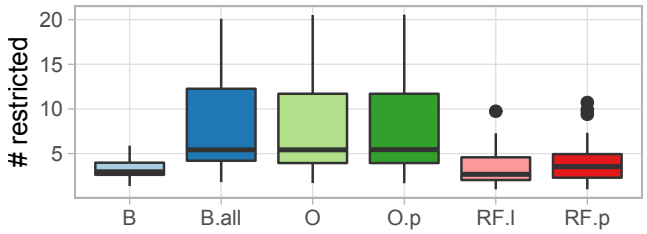


Figure 4.8: Number of restricted dimensions, all functions, $N = 400$

PRIM. This is because the boxes produced by REDS tend to be smaller, resulting in smaller areas of overlap. Generating boxes of very different size is not fair when comparing consistency of the algorithms. Indeed, the most consistent algorithm is one doing nothing but returning the box containing all data. This algorithm is clearly useless. However, with the growth in N, consistency of "RF.l" and "RF.p" grows, while consistency of the other approaches does not for the dataset sizes we consider. For $N > 800$, our approach is better.

Different Values of K. For these experiments, we set $N = 400$ and experimented with different values of the number of newly generated and labeled points L. Note that REDS uses only L newly generated and labeled points, ignoring the initial data. Figure 4.13 graphs the results averaged across 50 runs. For both AUpC and precision, "RF.p" (when random forest outputs probabilities) is better than "RF.l" (when labels $\{0, 1\}$ are predicted) and stops improving at $L \approx 1600$. For "RF.l" the precision continues to improve even for $L > 25000$. Observe that "RF.p" outperforms the other methods even when $200 = L < N = 400$. That is, 200 points labeled with probabilities learned with random forest let PRIM induce a better scenario than 400 points labeled with $\{0, 1\}$ with the original simulation model. This result confirms our statistical analysis.

Different Noise Levels. Finally, we experiment with different levels of noise in the data. We assume that the noise has the form of a random flip of the label value ($0 \longrightarrow 1$ or $1 \longrightarrow 0$),

Table 4.7: Consistency, all functions

N		B	B.all	O	O.p	RF.l	RF.p
400	avg	42.3	41.5	42.6	43.3	53.3	51
	# 1	1	0	0	2	19	11
	# 2	3	0	3	2	10	15
800	avg	42	44.1	45.1	45.5	55.6	52.5
	# 1	0	1	0	1	19	12
	# 2	4	3	2	2	10	12
1600	avg	41.7	46.1	47.2	47.6	58.5	53.8
	# 1	1	2	0	1	19	10
	# 2	5	4	2	0	11	11
m_{1600}	avg	9.9	11.1	14.9	15.1	40.5	28.6

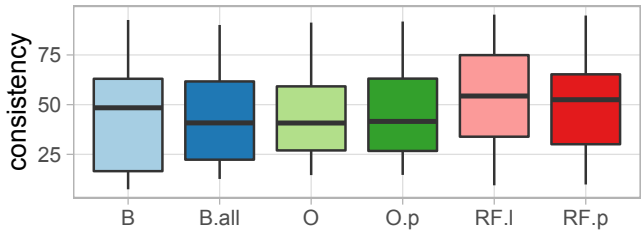


Figure 4.9: Consistency, all functions, $N = 400$

i.e., is independent of the model inputs $\{a_1, \ldots, a_M\}$ and the true label y. The noise level then reflects the share of points chosen at random for which the label is changed. Noise level 0.5 implies completely random data with $\Pr(y = 1) = \Pr(y = 0) = 0.5$. Figure 4.14 contains the results for $N = 400$ averaged across 50 runs. As expected, scenario quality decreases with the level of noise. Our approach provides better results for any level we experimented with, except for 0.5. For random data, when the noise level is 0.5, the AUpC value is positive, and the precision (0.57–0.64) is greater than that on the entire test set d_{test} (0.54). This is due to the best scenario being "inside" the initial box and thus even cutting off any dimension at random results in the increased precision at first.

4.7 Conclusions

Simulations allow studying the behavior of complex systems. Scenario discovery is the process of using simulations to gain interpretable insights regarding this behavior. PRIM is a state-of-the-art method for scenario discovery. It isolates conditions for system behavior of interest (e.g., system instability), in the form of a hyperbox, referred to as a scenario. The disadvantage of this method, as we have shown, is that it requires relatively many simulation runs, particularly in high dimensions, i.e., for systems with many input variables. Since simulations are often computationally hard, this is prohibitive in many cases.

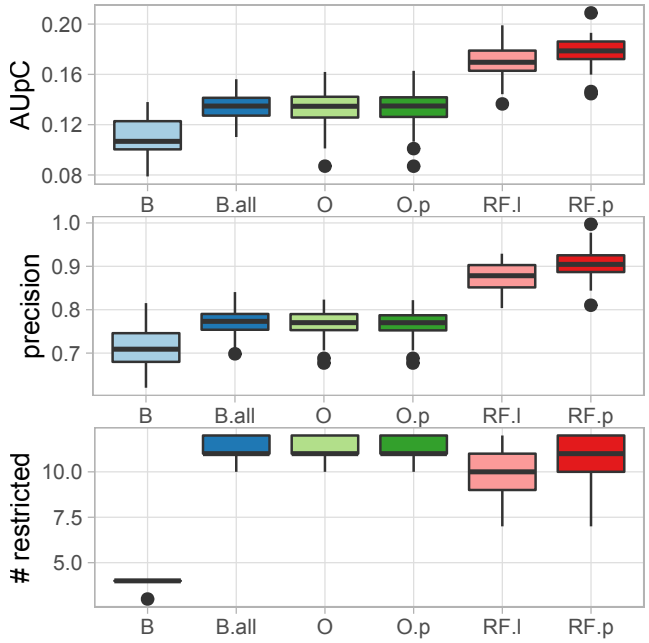


Figure 4.10: Quality metrics for DSGC for $N = 400$

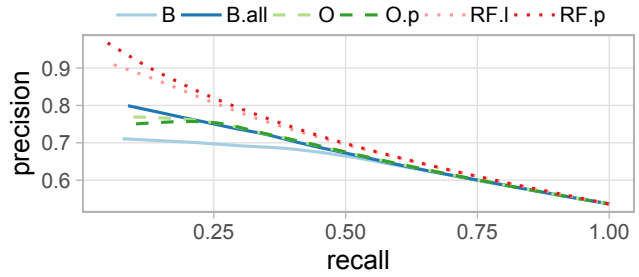


Figure 4.11: Peeling trajectories for DSGC for $N = 400$

We have studied reducing the number of simulations needed to discover good scenarios. Based on data produced with simulations, our method, REDS, first trains a powerful, but a complex statistical model. This model then replaces the simulation model, to label much more data for PRIM. We have justified the plausibility of our approach from a statistical point of view as well as with exhaustive experiments. Experiments show that our method is much better than existing ones. Specifically, it requires 75% fewer points than a conventional one on average to produce scenarios of comparable quality.

Finally, we explained that, in contrast to PRIM, REDS is compatible with active learning, which could further reduce the number of simulations.

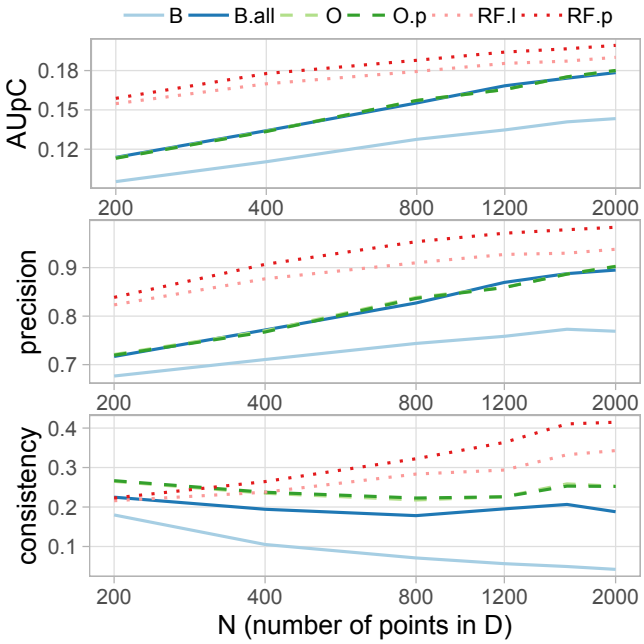


Figure 4.12: Learning curves on DSGC data

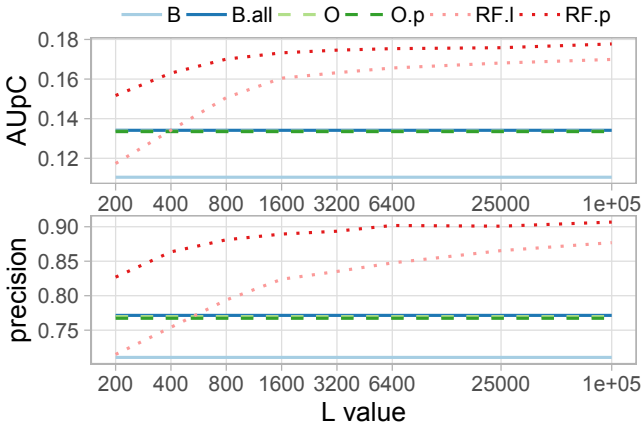


Figure 4.13: DSGC: quality metrics in dependence on K

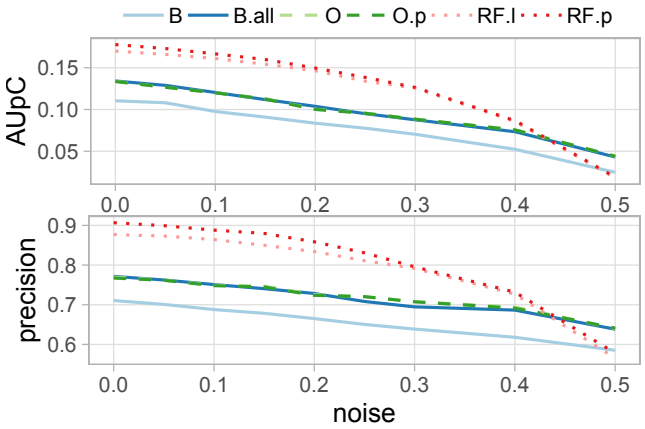


Figure 4.14: Quality metrics for DSGC in dependence on noise level for $N = 400$

5 Improving Subgroup Discovery

In the previous chapter, we proposed an approach, REDS (Rule Extraction for Discovering Scenarios) [AB21], to discover better subgroups (scenarios) from a limited amount of simulated data, when a simulation process is computationally expensive. With measured data, large labeled datasets are also often not available. Labeling additional data is either impossible or associated with high costs, or requires considerable time which cannot be reduced. Naturally, one wants to be able to find high-quality subgroups from small amounts of measured data. As we will show, REDS often fails to solve this task. In this Chapter, we propose REDS+, a method extending REDS methodology to work with measured data. We evaluate REDS+ experimentally with two algorithms for subgroup discovery (SD) from datasets with continuous attributes, PRIM and BestInterval; we found that REDS+ often improves the quality of their results.

Chapter Outline. Section 5.1 explains why REDS is not directly applicable to measured data. Section 5.2 introduces our approach, REDS+. Section 5.3 describes the experimental setup. Section 5.4 features the results. Section 5.4 concludes.

5.1 REDS on Measured Data

In this section, we explain why REDS may perform poorly on measured data. We conduct a small experiment to support our claim.

5.1.1 When REDS Does and Does Not Work

In our analysis in Section 4.2.1, $\hat{\mu}^{am}$ (4.5) approximates μ (4.1). Let us define by $p^g(x)$ the probability distribution of points used to estimate $\hat{\mu}^{am}$; we can write

$$\hat{\mu}^{am} = \frac{1}{l}\sum_{i=1}^{l} y_i^{am} \xrightarrow{l\to\infty} \frac{\int_b f^{am}(x)p^g(x)\,\mathrm{d}x}{\int_b p^g(x)\,\mathrm{d}x} \tag{5.1}$$

With REDS, we know the true distribution $p(x) = const$ and use $p^g(x) = p(x)$; the only source of bias in the estimate of μ is an imperfect approximation of $f(x)$ by f^{am}. With measured data, setting $p^g(x) = const$ may lead to an arbitrary high bias even if $f(x) = f^{am}$

$$[\mathrm{Bias}(\hat{\mu}^{am})]^2 = (\mu - \mu^{am})^2 = \left(\frac{\int_b f(x)p(x)\,\mathrm{d}x}{\int_b p(x)\,\mathrm{d}x} - \int_b f^{am}(x)\,\mathrm{d}x\right)^2.$$

Consequently, the quality of subgroups discovered with REDS can be low.

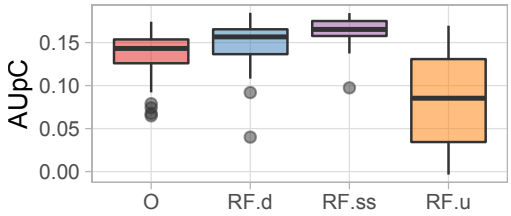


Figure 5.1: DSGC: quality metrics in dependence on K

In one specific case, semi-supervised setting, we can assume that unlabeled points form a representative sample from $p(x)$, and REDS will improve the quality of discovered subgroups. Specifically, we can label these points with f^{am}, add to the labeled data, and use the resulting dataset as an input for a subgroup discovery method. We now provide an example supporting the above considerations.

5.1.2 Small Experiment

We take the dataset "higgs" (see Table 5.1), name it D^h, set $N = 400$, $L = 2500$, and proceed as follows.

1. create a dataset D^i by selecting N examples from D^h;
2. create a dataset D^1 by selecting L points from D^h, which are not in D^i. D^1 represents the unlabeled pool in a semi-supervised setting;
3. create a dataset D^2 by selecting L points i.i.d. from the multidimensional uniform distribution with the support defined by the ranges of attributes' values in D^i, as REDS does;
4. train random forest classifier f^{am} on D^i;
5. create datasets $D^d = D^i$, $D^{ss} = D^i \cup D^1$ and $D^u = D^i \cup D^2$ replacing target variable's values with the probabilities predicted by f^{am};
6. apply PRIM to D^i, D^d, D^{ss}, D^u and call the outputs "O", "RF.d", "RF.ss" and "RF.u" respectively;
7. calculate AUpC (see Section 4.4.1) of each output on $D^h \backslash D_i$ for "O", "RF.d","RF.u", and on $D^h \backslash D^{ss}$ for "RF.ss";
8. repeat the above steps 30 times on different samples D^i, $i = 1, \ldots, 30$, so that $D^i \cap D^j = \emptyset$, $i \neq j$.

Figure 5.1 presents the results. One can see that the box for "RF.d" lies higher than the one for "O". I.e., relabeling points in D^i with probabilities predicted by random forest has improved the quality of PRIM output. This result improves further in a semi-supervised setting ("RF.ss"). However, when the new points are sampled from a uniform distribution ("RF.u"), AUpC is significantly lower than for PRIM on original data D^i ("O"). This experiment shows the importance of $p^g(x)$ being a good approximation of $p(x)$.

Algorithm 7: REDS+

Data: D, D^{val}, α, B_0, *minpts*, L, AM, SM, G, SD — described in the text
Result: Output of the algorithm SD
1 $D^s = \text{scale}(D, SM)$;
2 $f^{am} = \text{train}(AM, D)$;
3 $p^g = \text{train}(G, D)$;
4 $D^{add} = \text{new.points}(L, p^g) + \text{zeros}(L)$;
5 $D' = D^s \cup D^{add}$;
6 **for** $1 < i < (L + N)$ **do**
7 $\quad y_l^{add} = \text{predict}(x_l^{add}, f^{am})$;
8 $D^{new} = \text{scale.back}(D', SM)$;
9 $res = \text{SD}(D = D^{new}, \dots)$;
10 **return** *res*

5.2 Extending REDS

Algorithm 7 describes our proposed method, REDS+, to discover better subgroups from measured data. In contrast to REDS, it uses a method G (we call it data generator) to approximate with $p^g(x)$ the true multidimensional distribution $p(x)$ of points in D, and then samples from $p^g(x)$. We now describe the details.

First, REDS+ scales the attributes in D with a scaling method SM to obtain D^s (Line 1). This is done since some metamodels AM data generators G work better with scaled data. It then fits a metamodel f^{am} and a data generator p^g to data D^s (Lines 2–3). Next, the algorithm generates L additional points using p^g, adds the target vector consisting of zeros to them, and adds the resulting points to D^s (Lines 4–5) to form D'. After this, REDS+ replaces the values of the target variable in D' with predictions of f^{am} and scales the data back to obtain D^{new} (Lines 6–8). Finally, our method applies a subgroup discovery algorithm SD to D^{new} and returns its output (Lines 9–10).

The scaling method used in Line 1 could be, e.g., z-score scaling or scaling to a given range. For instance, if we define $a_j^{max} = \max_i (x_{ij})$, $a_j^{min} = \min_i (x_{ij})$, scaling to the range $[0, 1]$ is

$$x_{ij}^s = \frac{x_{ij} - a_j^{min}}{a_j^{max} - a_j^{min}},$$

and respective formula for scaling back is

$$x_{ij}^{new} = x_{ij}^s \left(a_j^{max} - a_j^{min} \right) + a_j^{min}.$$

When there exists a pool of unlabeled data, the quality of intermediate metamodel f^{am} might be further improved by using existing semi-supervised methods (see Section 2.4.3) in the training procedure (Line 2). We now present the intuition behind REDS+.

5.2.1 Intuition Behind REDS+

Our analysis in Chapter 4 suggests that REDS works if the reduction in the variance of PRIM due to a larger training set exceeds the bias of a metamodel AM. As we explained above, it is essential that the original and additionally generated points come from similar distributions. The detailed analysis depends on a specific subgroup discovery method chosen for REDS+ and may not provide insights going beyond these qualitative considerations.

In other words, our key assumption is that the quality of the output of an SD algorithm increases with the number of examples N in the dataset and approaches some saturation level. In our case, this level cannot be above one, since it is the maximal value of any quality measure we use. We aim at making the learning curve (see Figure 4.1) steeper at the beginning, by generating additional examples. If these examples are as good as the original ones, that is, come from the same distribution, our goal is achieved naturally. REDS+ uses a data generator G to approximate $p(x)$ with p^g; and a metamodel AM to approximate $p(y = 1|x) = f(x)$ with f^{am}. If both p^g and f^{am} are accurate, REDS+ will improve the quality of subgroups found from small datasets. We will check this experimentally.

5.3 Experimental Setting

In this section, we describe our experimental setting. First, we introduce the key components REDS+: data generators G, metamodels AM, and subgroup discovery algorithms SD, which we will use in the experiments. We discuss how we chose the hyperparameter values of these components and give a short name to each combination of a component and its hyperparameter value(s), which we will use further on the plots. Then we list the datasets and their sources and describe the configuration of experiments and quality metrics.

5.3.1 Data Generators

Dummy. This generator does not generate additional points. Dummy generator combined with a metamodel AM in REDS+ means that the points in D are re-labeled with f^{am}, as is the case with "RF.d" in our experiment above. This generator does not have any hyperparameters; we use a short name "d" to refer to it.

Uniform. This is our baseline generator. As in REDS, it samples points from a multivariate uniform distribution with the support $\prod_{j=1}^{M}[\min_i(x_{ij}), \max_i(x_{ij})]$. As we have demonstrated above, augmenting the dataset with uniformly generated points may decrease the quality of discovered subgroups. This generator does not have any hyperparameters; we use a short name "unif" to refer to it.

Normal. This generator samples new points from a multivariate normal distribution with the mean vector $\{\hat{\mu}_1, \ldots, \hat{\mu}_M\}$ and covariance matrix $\mathrm{diag}(\hat{\sigma}_1^2, \ldots, \hat{\sigma}_M^2)$. Where

$$\hat{\mu}_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij}, \qquad \hat{\sigma}_j^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(x_{ij} - \mu_j\right)^2 \tag{5.2}$$

Algorithm 8: MUNGE

Data: D, L, pr, lv — as described in the text
Result: a dataset D^{add} of the size $L \times M$, without target variable

$D^{add} = \emptyset$;
while $|D^{add}| < L$ **do**
  $D' = D$;
  foreach $x'_i \in D'$ **do**
    $x_k \in D$ = nearest neghbor (but not the same point) of x'_i;
    foreach $j \in \{1, \ldots, M\}$ **do**
      with probability pr : $x'_{ij} = \text{norm}(x_{kj}, |x'_{ij} - x_{kj}|/lv)$;
  $D^{add} = D^{add} \cup D'$;
  $D^{add} = \text{clean}(D^{add}, D)$;
$D^{add} = \text{sample.rows}(D^{add}, L)$;
return D^{add}

This generator does not have any hyperparameters, and we use a short name "norm" to refer to it.

Gaussian Mixtures. A more flexible generation, which includes the normal as a particular case, is to sample points from a mixture of H Gaussian distributions:

$$X \sim \sum_{h=1}^{H} \phi_h \mathcal{N}(\mu_h, \Sigma_h^2), \qquad \sum_{h=1}^{H} \phi_h = 1$$

The number of components H, mean vector μ_h, and covariance matrix Σ_h^2 for each component are estimated from data. To prevent overfitting, one can decrease the number of degrees of freedom by constraining covariance matrices to be all diagonal, shared between components or scalar: $\Sigma_h^2 = \sigma_h^2 I$, where I is the identity matrix and $\sigma_h \in \mathbb{R}$.

The Gaussian mixtures ("gm") model is estimated with the expectation-maximization algorithm [DLR77]. We consider all available covariance types and numbers of components from the set $\{1, \ldots, 30\}$; we choose the optimal values via 5-fold cross-validation[1]

Kernel Density Estimate. Another possibility to generate points is to sample them from a multivariate probability density estimated from data with a Gaussian kernel as

$$p(x) = \frac{1}{N\lambda_1 \cdots \lambda_M (2\pi)^{\frac{M}{2}}} \sum_{i=1}^{N} e^{-\frac{1}{2}\sum_{j=1}^{M}\left(\frac{x_{ij}-x}{\lambda_j}\right)^2}.$$

A smoothing parameter λ_j, $j = 1, \ldots, M$ is called bandwidth.

[1] McLachlan and Rathnayake [MR14] present other possibilities to optimize hyperparameters including, widely used Bayesian information criterion [S+78]. We use cross-validation for consistency.

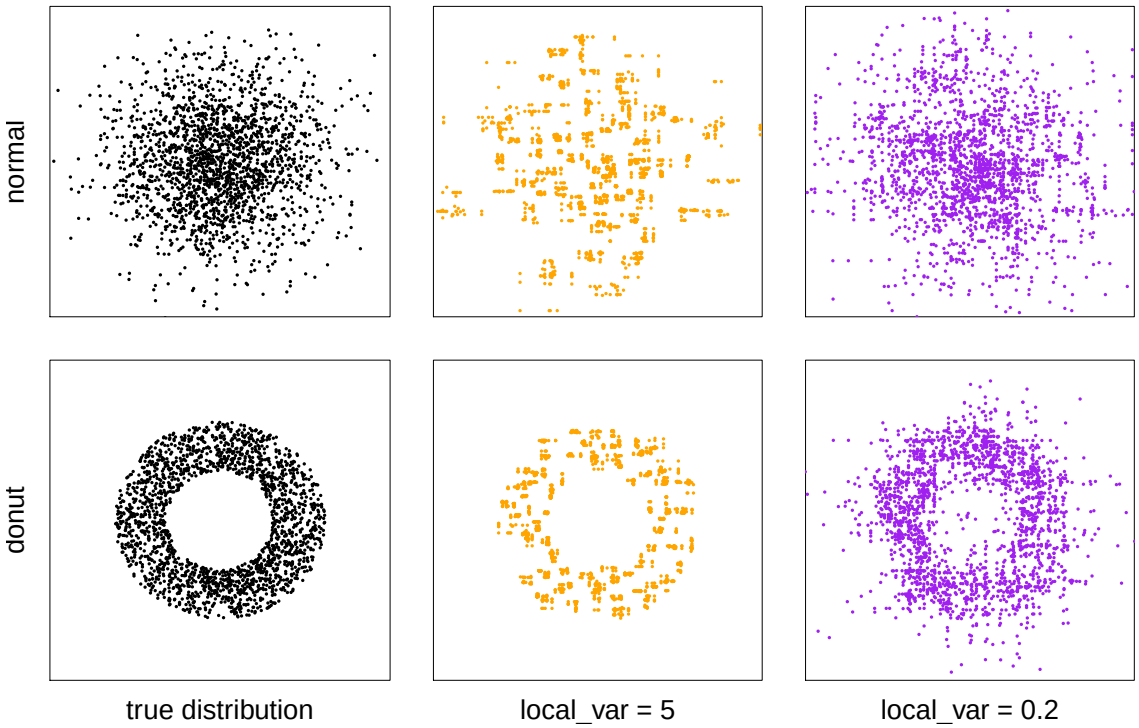


Figure 5.2: Influence of local variance on MUNGE output.

In our experiments, we use the Gaussian kernel. We use 5-fold cross-validation to chose the value of bandwidth from the set of 50 equidistant values $\{0.01, \ldots, 0.5\}$ ("kde.g"); we also try Silverman's [Sil86] rule of thumb ("kde.s"):

$$\lambda_j = \hat{\sigma}_j \left[\frac{4}{(M+2)N} \right]^{\frac{1}{M+4}}, \qquad j = 1, \ldots, M,$$

where the attributes' standard deviations $\hat{\sigma}_j$ are estimated as in (5.2).

We use the kernel density estimation and Gaussian mixtures model implementations from scikit-learn [PVG+11] (version 0.22.2) python[2] module.

MUNGE. This data generation technique is from [BCN06]. Its original description suggests L operations of nearest neighbor search at minimum; each search is in N points from M-dimensional space. The authors do not propose an efficient way of doing this search; neither they provide the reason behind. Their description of MUNGE also does not assume removing duplicated points from the generated data. Thus we present our interpretation of the MUNGE (Algorithm 8); it searches nearest neighbors less frequently than the original description suggests and generates only distinct points.

We have implemented the MUNGE generator ourselves. The original paper [BCN06] does not provide a recommendation regarding the values of its hyperparameters, pr, and lv. In the algorithm implementation which we have found[3], their default values are 0.5

[2] `https://www.python.org/`

[3] `https://github.com/Prometheus77/munge/blob/master/R/munge.R`

and 5, respectively. However, we have experimentally found that the result with $lv = 0.2$ and $pr = 0.5$ looks more reasonable. In Figure 5.2, all plots contain 2000 points. MUNGE was applied to a sample of 200 points from the distributions shown on the leftmost plots. With $lv = 5$ (middle plots), the generated points tend to form too dense clusters.

So we use $p = 0.5$ and $lv = 0.2$ (Figure 5.2, right plots; we call this generator "m0.2") in further experiments and also try $lv = 1$ (we use "m1" to refer to this instantiation). Still, the optimal hyperparameter values might depend on the dataset size N and the number of attributes M. But since this method is nonparametric, hyperparameter optimization through, e.g., cross-validation, is not straightforwardly applicable.

5.3.2 Metamodels

Subgroup discovery algorithms scale super-linearly with an increase in the number of examples as they require sorting the data, which is in $O(N \log(N))$ [CLRS01]. In Chapter 4, we have obtained that labeling with probabilities leads to a more rapid quality improvement of discovered subgroups. Hence, in this chapter, we consider metamodels that output probability estimates. In particular, we will experiment with random forest [Bre01], Gaussian processes classification [RW06], naive Bayes [HTF09] and support vector machines [Bur98].

One needs to calibrate the output of SVM to obtain probabilities [P+99]. Next, while naive Bayes predicts probabilities, this model is usually too confident in its predictions (close to 0 or 1) due to the "naive" assumption on attributes' mutual independence, given the class variable [NC05]. Thus, we calibrate the output of both SVM and naive Bayes with Platt Scaling [P+99] — a recommended technique for small sample sizes [NC05].

All metamodels are implemented in the scikit-learn python module. For Naive Bayes (NB) and Gaussian process (GP) classifiers, we do not alter the hyperparameter values suggested by their default implementation. According to scikit-learn documentation[4], a random forest (RF) classifier is most sensitive to two hyperparameters — the number of trees and the number of randomly selected attributes to consider at each split. More trees usually result in better quality, so we keep default value 100 for this hyperparameter. We choose the optimal value of the number of randomly selected attributes from the set $\{2, \sqrt{M}, M\}$ via cross-validation. For support vector machines (SVM), we choose the optimal values of cost hyperparameter from the set $\{0.1, 1, 10, 100\}$ and kernel hyperparameter from the set $\{0.001, 0.01, 0.1, 1\}$ using cross-validation. The exponentially growing value of the increment in both sequences is in line with the existing recommendations [HCL+03].

5.3.3 Subgroup Discovery Algorithms

Many subgroup discovery (SD) algorithms were developed over the past decades. A significant share of them performs an exhaustive search (see Section 2.3). However, such methods can only cope with small datasets with a low number of continuous attributes, despite using ingenious pruning strategies, see, e.g., [GR09]. Hence, we focus on the methods exploiting heuristic search.

[4] https://scikit-learn.org/stable/modules/ensemble.html#forest

To this end, we select PRIM [FF99] and BestIntervalBS [MNFK12] algorithms, described in Section 2.3. Their search heuristics are, in a certain sense, complementary — "patient" in PRIM and "greedy" in BestInterval. Thus, it is interesting whether both can benefit from the proposed methodology.

We use metamodels that label data with probabilities, whereas BestInterval algorithm was proposed to optimize *WRAcc* measure, i.e., assumes a binary input. We now show that it also optimizes *NWRacc*.

Proposition 5.3.1 *The BestInterval algorithm [MNFK12] originally developed for WRAcc quality measure can be used for NWRAcc measure without modification.*

Proof: The BestInterval algorithm is based on the additive property of *WRAcc* (Property 2.3.1). To show that it holds for *NWRAcc*, we simply repeat the derivation from [MNFK12] using our more general notations. Specifically, we show that for any two hyperboxes B_1 and B_2 with $S_{B_1} \cap S_{B_2} = \emptyset$, it holds that $NWRAcc(D, S_{B_1} \cup S_{B_2}) = NWRAcc(D, S_{B_1}) + NWRAcc(D, S_{B_2})$. Referring to the quantities associated with B_1 and B_2 using respective subscripts (1 and 2), we obtain

$$\begin{aligned} NWRAcc(D, S_{B_1} \cup S_{B_2}) &= \frac{n_1 + n_2}{N} \left(\frac{n_1^+ + n_2^+}{n_1 + n_2} - \frac{N^+}{N} \right) = \frac{n_1^+ + n_2^+}{N} - (n_1 + n_2) \frac{N^+}{N^2} \\ &= \frac{n_1}{N} \left(\frac{n_1^+}{n_1} - \frac{N^+}{N} \right) + \frac{n_2}{N} \left(\frac{n_2^+}{n_1} - \frac{N^+}{N} \right) \\ &= NWRAcc(D, S_{B_1}) + NWRAcc(D, S_{B_2}) \end{aligned}$$

□

Since the results from Chapter 4 indicate a low influence of the pasting step of PRIM, we use only the peeling step in the following experiments. We set PRIM hyperparameters to the same values as before, i.e., $\alpha = 0.5$ and $minpts = 20$; we also assume $D = D^{val}$. For BestIntervalBS ("BI"), we set the beam size to $bs = 1$ and the maximal number of attributes defining a subgroup to $m = 20$, since we do not experiment with datasets with $M \geq 20$ due to relatively high time complexity of the algorithm. We will additionally try different values of bs in the particular experiment.

5.3.4 Datasets

We use 18 datasets to test our approach. We have selected sufficiently large ones to split each into many non-overlapping small datasets in order to reduce random effects in or experiments. Since many of the generators assume continuous attributes, we remove the attributes having few ($Z < 20$) unique values; we also remove irrelevant attributes, e.g., "ID". From the resulting datasets, we remove rows with missing values. Table 5.1 lists the datasets together with their characteristics and corresponding references. Here N^* — the number of examples after removing incomplete ones (i.e., with missing attributes' values); M^* — the number of attributes; M — the final number of attributes (after removing "discrete" and irrelevant); "+" — the positive class; N^{*+}/N^* — the share of positive examples.

Name[5]	N^*	M^*	M	+	N^{*+}/N^*	Reference
occupancy	20560	6	6	1	0.231	[DG17, CF16]
higgs	98050	28	7	1	0.529	[VvRBT13, BSW14]
htru	17898	8	8	1	0.092	[DG17, LSC+16]
shuttle	58000	9	9	1	0.786	[DG17]
avila	20867	10	10	A	0.411	[DG17, SMFdF18]
gamma-telescope	19020	10	10	g	0.648	[DG17]
eeg-eye-state	14980	14	14	1	0.551	[DG17]
credit cards	30000	24	14	1	0.221	[DG17, YL09]
higgs_o	98050	28	17	1	0.529	[VvRBT13, BSW14]
ring	7400	20	20	1	0.505	[OCO+17]
sylva	14395	108	20	1	0.062	[VvRBT13]
jm1	10880	21	21	true	0.193	[VvRBT13, SSM05]
stocks	96320	21	21	1	0.505	[VvRBT13]
sensorless	58509	48	48	1	0.091	[DG17]
bankruptcy	4885	64	64	1	0.022	[DG17, ZTT16]
gas	10310	128	128	1	0.191	[DG17, VVA+12]
clean2	6598	168	168	1	0.154	[OCO+17]
seizure	11500	178	178	1	0.2	[DG17, ALM+01]

Table 5.1: Datasets used for evaluation.

To make REDS+ applicable to datasets with mixed attribute types, one could use suitable data generators. For instance, MUNGE can be extended to mixed attributes as in [BCN06]. Alternatively, one could only use the artificially enlarged dataset when considering continuous attributes and use the original small dataset for discrete ones. This approach is similar to the one proposed in [GR09]. However, respective experiments are beyond the scope of this work.

[5] occupancy — "Occupancy Detection". We keep only time in the first attribute, "date" `http://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+`; higgs — "HIGGS", only "high-level" features `https://www.openml.org/d/4532`; htru — "HTRU2" `https://archive.ics.uci.edu/ml/datasets/HTRU2`; shuttle — "Statlog (Shuttle)" `http://archive.ics.uci.edu/ml/datasets/Statlog+(Shuttle)`; avila — `https://archive.ics.uci.edu/ml/datasets/Avila`; gamma-telescope — "MAGIC Gamma Telescope" `https://archive.ics.uci.edu/ml/datasets/magic+gamma+telescope`; eeg-eye-state — "EEG Eye State" `http://archive.ics.uci.edu/ml/datasets/EEG+Eye+State`; credit cards — "Default of credit card clients" `https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients`; higgs_o — only relevant "low-level" features from "HIGGS"; sylva — "SYLVA" `https://www.openml.org/d/1040`; jm1 — "JM1/Software defect prediction" `https://www.openml.org/d/1053`; stocks — "Encrypted Stock Market Data from Numerai" `https://www.openml.org/d/23517`; sensorless — "Dataset for Sensorless Drive Diagnosis" `http://archive.ics.uci.edu/ml/datasets/Dataset+for+Sensorless+Drive+Diagnosis`; bankruptcy — "Polish companies bankruptcy data" (3-rd file) `https://archive.ics.uci.edu/ml/datasets/Polish+companies+bankruptcy+data`; gas — "Gas Sensor Array Drift Dataset" `https://archive.ics.uci.edu/ml/datasets/Gas+Sensor+Array+Drift+Dataset`; seizure — "Epileptic Seizure Recognition" `https://archive.ics.uci.edu/ml/datasets/Epileptic+Seizure+Recognition`

5.3.5 Quality Metrics

As in Chapter 4, we use AUpC and precision to measure the quality of subgroups produced with PRIM. Since BestInterval is designed to optimize WRAcc, we take this measure to evaluate the algorithm's output. We also compare PRIM and BestInterval, which was not done before to our knowledge. To this end, we compute WRAcc for the box B_R in PRIM output which has maximal *NWRAcc* on the dataset D, see Algorithm 1. We do so since one can show that PRIM maximizes *NWRAcc* up to a certain iteration. Specifically, the following holds.

Proposition 5.3.2 *Let B_w, $w = 0, \ldots, W$ be the sequence of boxes produced by PRIM when it is used with its default target function — mean value of points inside the box. Changing this function to WRAcc (NWRacc) would result in the sequence of boxes B_r, $r = 0, \ldots, R$, $R \leq W$ with $(B_w = B_r | w = r)$.*

Proof: Repeating the reasoning from Section 4.2.1, on the w-th iteration, PRIM selects the next box B_{w+1} with a maximal value of $n^+ = n^+_{w+1}$ for points inside it, from a set of possible boxes B_{jw}, all defining subgroups with the same number of points n. Assume, that changing the target function to *WRAcc* (*NWRAcc*), would have resulted in the selection of another box $B_{r+1} \neq B_{w+1}$, with $n^+ = n^+_{r+1} < n^+_{w+1}$. But this would mean that

$$WRAcc(D, S_{B_{r+1}}) = \frac{n}{N}\left(\frac{n^+_{r+1}}{n} - \frac{N^+}{N}\right) < \frac{n}{N}\left(\frac{n^+_{w+1}}{n} - \frac{N^+}{N}\right) = WRAcc(D, S_{B_{w+1}}),$$

which contradicts the selection criterion. □

5.3.6 Design of Experiments

Every single experiment in our evaluation consists of the following steps.

1. Split a dataset DS^j into two parts $D_i : |D_i| = N$ and $D^{test} = DS^j \backslash D_i$;
2. apply REDS+ to D_i;
3. evaluate quality on D^{test}.

We repeat the above experiment for PRIM, all datasets DS^j ($j = \{1, \ldots, 18\}$) from Table 5.1, all data generators, and metamodels listed above. For each dataset, we experiment with different sizes of D^i: $N = \{200, 400, 800, 1600\}$. To reduce the influence of random effects, for each N value, we do $\min\{30, \lfloor |DS^i|/N \rfloor\}$ splits into D^i and D^{test}, so that sets D^i from different splits do not overlap: $D^i \cap D^j = \emptyset, i \neq j$. We experiment with BestIntervalBS (BI) algorithm in the same way, but with a smaller variety of datasets, data generators, and metamodels since the experiments take longer. To emulate a semi-supervised setting, we remove random 2500 points from D^{test} and use them as unlabeled data. For convenience, we treat this data as produced by a data generator which we call "ss".

For all generators, we set the number of newly generated points $L = 2500$. We chose this value since it is at least as large as N but small enough for BestIntervalBS algorithm so that the experiments take a reasonable time.

We configure both scenario discovery algorithms to output a single subgroup description. It allows us to apply the quality metrics straightforwardly without developing a methodology to evaluate the quality of multiple subgroups. This configuration does not affect the generality of our conclusions since both (and many other) subgroup discovery algorithms follow *covering* procedure when searching multiple subgroups (see Section 2.3). That is, they find subgroup descriptions one after another, removing corresponding examples from train data.

5.4 Results

In this section, we resent the results from experiments with PRIM, followed by the ones from experiments with BestIntervalBS. In the end, we compare these two scenario discovery algorithms.

5.4.1 Experiments with PRIM

Figure 5.3 summarizes the outcome of experiments with PRIM for different sizes N of D^i and different quality metrics. Each cell in this heatmap plot corresponds to a combination of a metamodel and a data generator. Its color reflects the improvement in percent in median (taken over $\min\{30, \lfloor |DS^j|/N \rfloor\}$ experiments averaged over datasets DS^j from Table 5.1). Blue shadows stand for quality improvement, red — for the decrease. The results are consistent across dataset sizes N and quality measures. The best performing metamodel is random forest, the worst one is naive Bayes. REDS+ with uniform data generation is worse than just applying PRIM to D^i, whereas REDS+ with generating from kernel density estimates or Gaussian mixtures improves the quality of subgroups discovered by PRIM. For more detailed insights, we visualize the results separately for each dataset.

Figures 5.4–5.5 provide such visualization for $N = 400$ and $N = 1600$, respectively, and for $AUpC$ quality measure. Each plot stands for a single dataset DS^j; we have shortened some names but retained the datasets order the same as in Table 5.1. Each box is based on $\min\{30, \lfloor |DS^j|/N \rfloor\}$ experiments; their number is given in brackets in the plot title. Box color corresponds to the algorithm used. "O" means original algorithm (here, PRIM); different REDS+ instantiations we abbreviate as "<metamodel>.<generator>" — e.g., "RF.kde.s" stands for random forest metamodel used together with generation from kernel density estimate with bandwidth chosen according to Silverman's rule.

One can see that RF.kde.s (green boxes) often leads to a noticeable improvement in AUpC compared to just PRIM (red boxes). REDS+ in semi-supervised setting — RF.ss (purple boxes) — is usually as good or even better than RF.kde.s. For some datasets (shuttle, avila, sylva clean2) RF.ss performs significantly worse than RF.kde.s. That is, generating points from approximated distribution for these datasets turned out to be better than from the true one. Relabeling existing points with estimated probabilities (RF.d, blue boxes) does not result in consistent improvement of the PRIM output. All this confirms that generating points from the distribution reflecting the true one well is an essential feature of REDS+.

In Figure 5.6, we present learning curves for PRIM and "RF.kde.s". Horizontal axes are logarithmically scaled. The lines are median values from $\min\{30, \lfloor |DS^j|/N \rfloor\}$ experiments;

the area between 25-th and 75-th percentiles is shaded. Only in six (out of $18 \cdot 4 = 72$) cases, the median AUpC obtained with REDS+ is lower than that resulting from PRIM. For instance, for bankruptcy dataset at $N = 1600$. Observe, that PRIM's AUpC at $N = 1600$ is achieved with REDS+ at $N \approx 200$ for gamma-telescope and ring, at $N \approx 400$ for sylva, sensorless, credit cards, i.e., for 4–8 times smaller number of points. Even better results are for jm1, clean2, shuttle, and avila datasets.

5.4.2 Experiments with BestIntervalBS

The conclusions similar to PRIM and AUpC also apply to BestIntervalBS and WRAcc. Figures 5.7–5.8 are analogous to Figures 5.4–5.5; Figure 5.10 presents the learning curves. REDS+ with random forest in a semi-supervised setting ("RF.ss") and with generating from kernel density estimate ("RF.kde.s") improves the result of BestIntervalBS. For avila and credit cards datasets, "RF.kde.s" even beats "RF.ss", i.e., it is better to generate artificial points than to use existing unlabeled ones.

By adding points to D, REDS+ increases the search space for BestIntervalBS. This results in the longer time needed to discover subgroups, which rises with the growth of the number of added points L. Since BestIntervalBS is heuristic, the part of space it searches (consequently, the search time) can be controlled by the beam size hyperparameter bs. Naturally, one may wonder if increasing bs is a viable alternative to REDS+ since both come at the cost of additional computation complexity. To check this, we experiment with "RF.kde.s" and BestIntervalBS with $bs = 1$ and different values of L from one side, and BestIntervalBS with different numbers of bs from another side. We do so with the gamma-telescope dataset. Figure 5.9 presents the results. Each point is the median value over 30 experiments on non-overlapping D^i (see Section 5.3.6), error bars cover the area between 25-th and 75-th percentiles. Changing bs did not allow BestIntervalBS to achieve the performance of REDS+.

5.4.3 Comparing PRIM and BestIntervalBS

To compare PRIM with BestIntervalBS, we plot corresponding learning curves. PRIM produces better subgroups than BestIntervalBS on shuttle and credit cards datasets; it is worse than BestIntervalBS on avila and ring datasets. For the other datasets, the results of these algorithms are comparable Figure 5.11. REDS+ with PRIM is superior to REDS+ with BestIntervalBS in the majority of cases, see Figure 5.12.

These results should be interpreted with care. In the proof of Proposition 5.3.2, we used the assumption of equal sizes of candidate subgroups. However, for PRIM (Algorithm 1), this might not be true if there are attributes with a specific value overrepresented ($\exists u \in \mathbb{R} : |\{a = u\}|/N > 1/N$), which is the case for many datasets we use. Changing the target function of PRIM from mean to WRAcc should result in a more fair comparison, but this is beyond the scope of this work.

5.5 Conclusions

Subgroup discovery methods discover knowledge in a comprehensible form of hyperboxes. However, they require many examples to produce a reliable output. With measured data, large datasets are not often available due to considerable costs of the data collection process. Two cases are possible. In a semi-supervised case, many unlabeled examples (points) are available, and one can label them with a powerful ML model to artificially increase the dataset size. This procedure is almost the same as the one implemented by REDS in Chapter 4. In the second case, when no pool of unlabeled data exists, REDS can fail to produce good results. This is because it generates new points from a multidimensional uniform distribution, which can be a bad approximation of the one data comes from.

In this chapter, we have proposed REDS+, a generalization of REDS, for finding better subgroups from small datasets. Instead of generating new points uniformly at random, it generates unlabeled examples trying to resemble a distribution of points in the data. REDS+ then joins these points with the genuine ones, labels them using an ML model fitted to original data, and applies scenario discovery algorithm to the resulting dataset.

We have experimented with different approaches to generate new points, different ML models, and two subgroup discovery algorithms. We have found that using REDS+ with random forest leads to better results than directly applying a scenario discovery algorithm to labeled data both in the semi-supervised case and when no unlabeled data available. In the latter case, REDS+ exhibits the best performance when new points are generated from a kernel density estimate. For some datasets, our approach requires 4–8 times fewer examples than conventional scenario discovery process to find subgroups of similar quality.

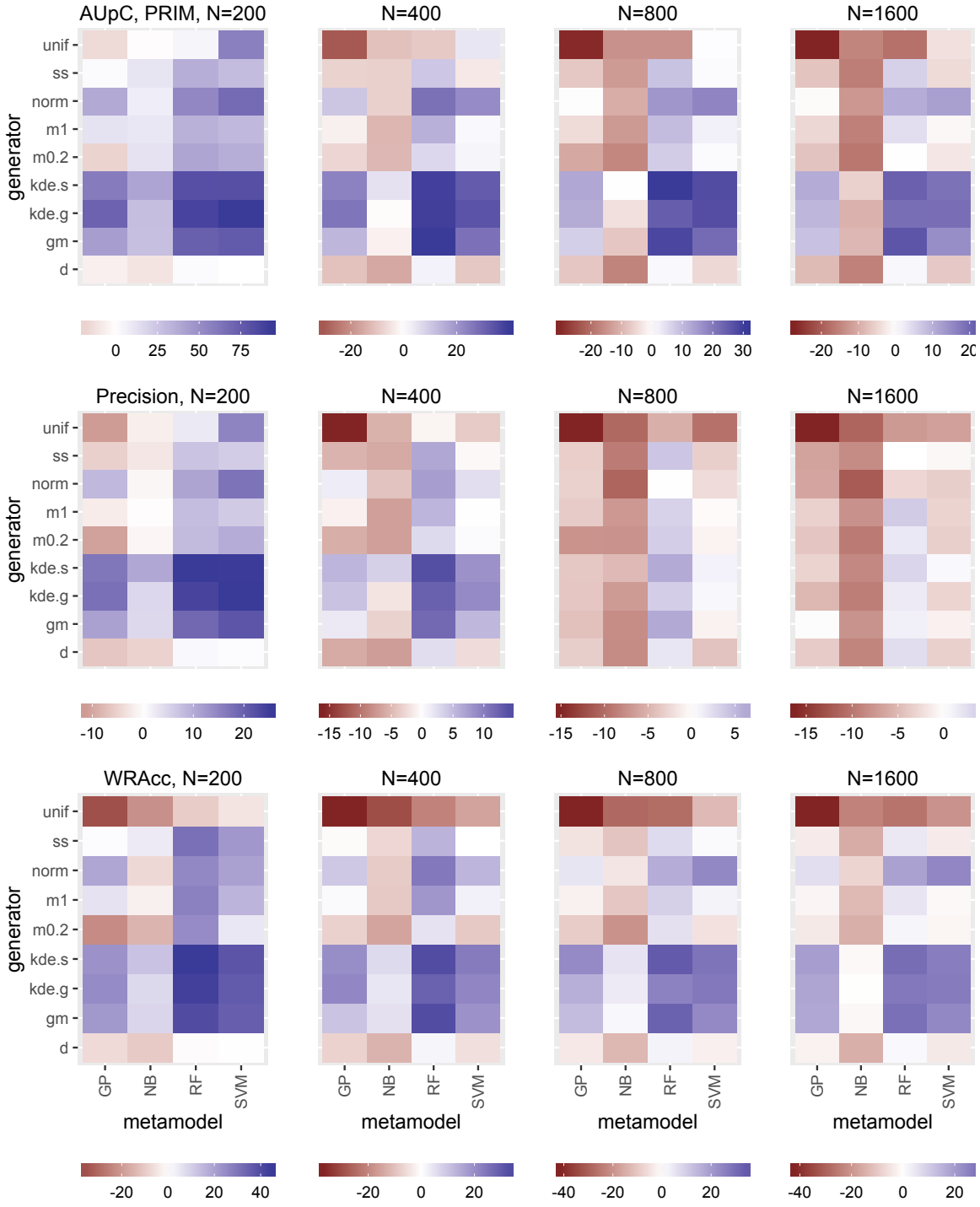


Figure 5.3: Average (over datasets) improvement in median (taken over data splits) quality of subgroups found by REDS+ with PRIM compared to just PRIM in percent.

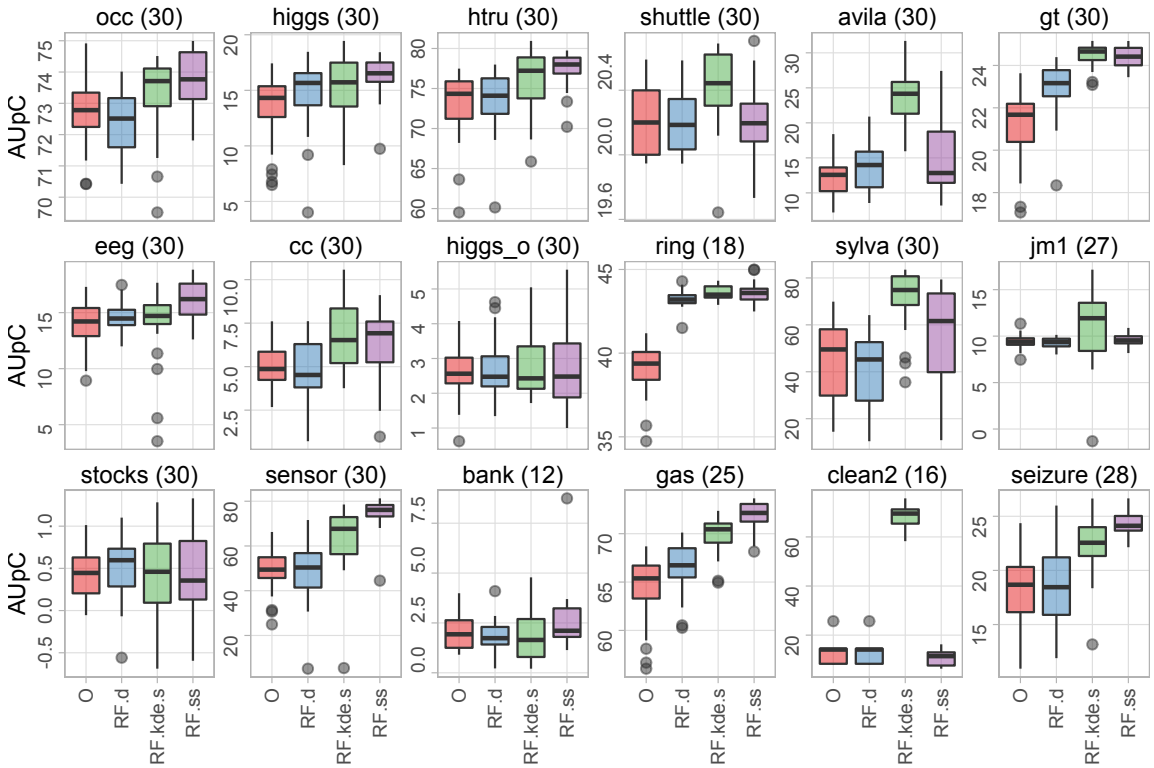


Figure 5.4: AUpC. REDS+ with PRIM. N = 400.

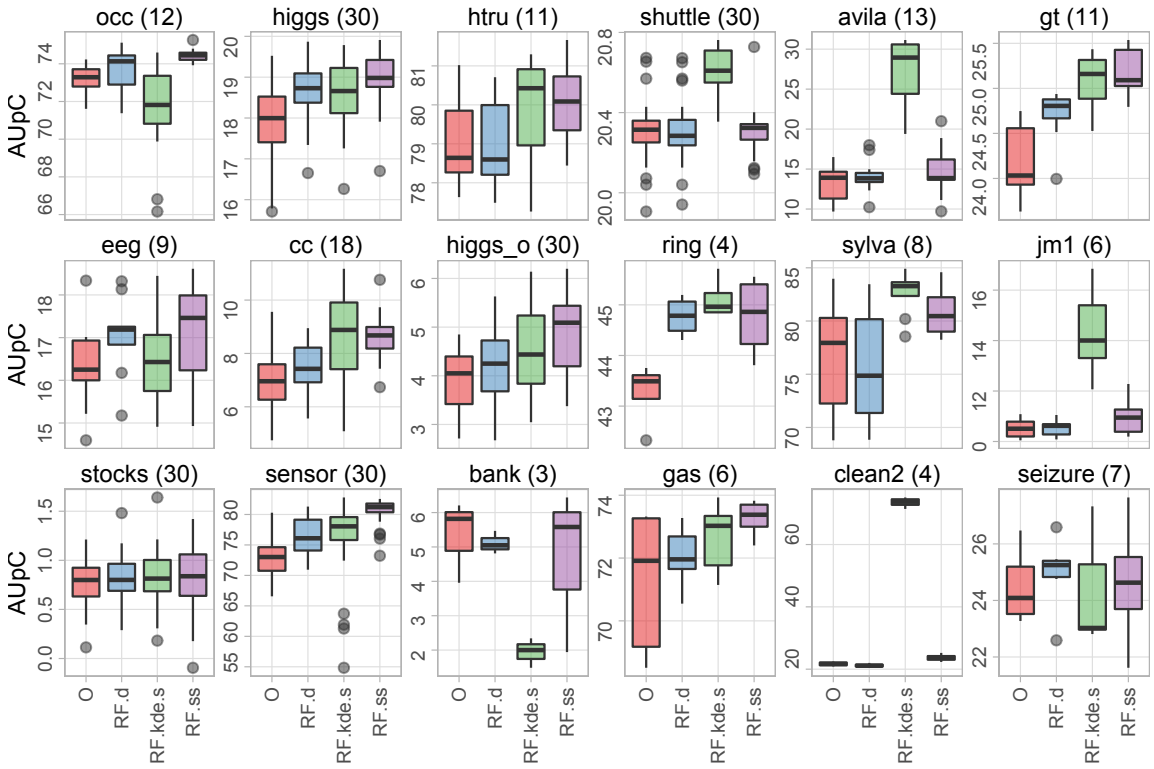


Figure 5.5: AUpC. REDS+ with PRIM. N = 1600.

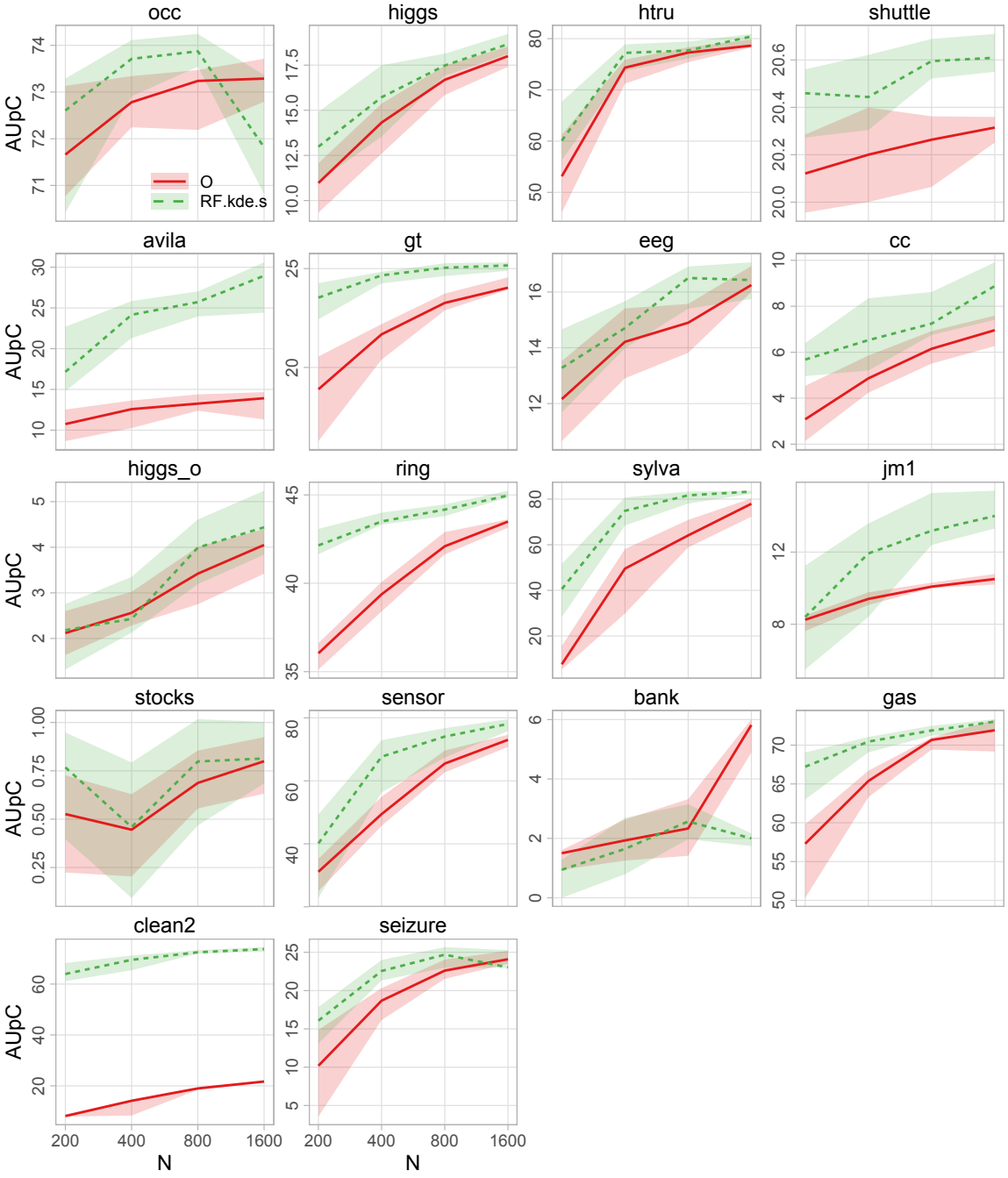


Figure 5.6: AUpC. REDS+ with PRIM learning curves.

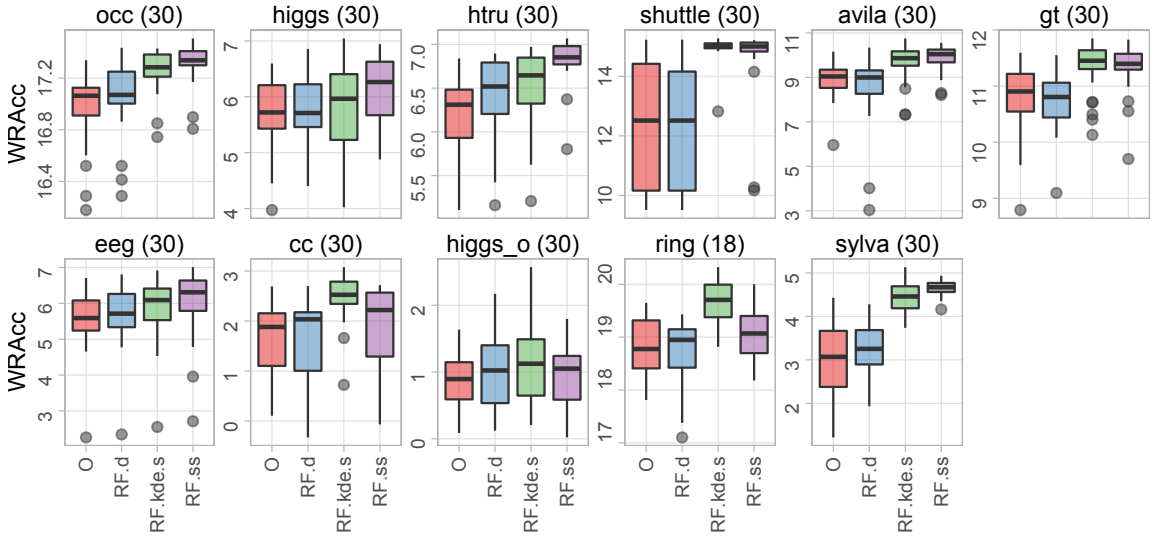


Figure 5.7: WRAcc. REDS+ with BestIntervalBS. N = 400.

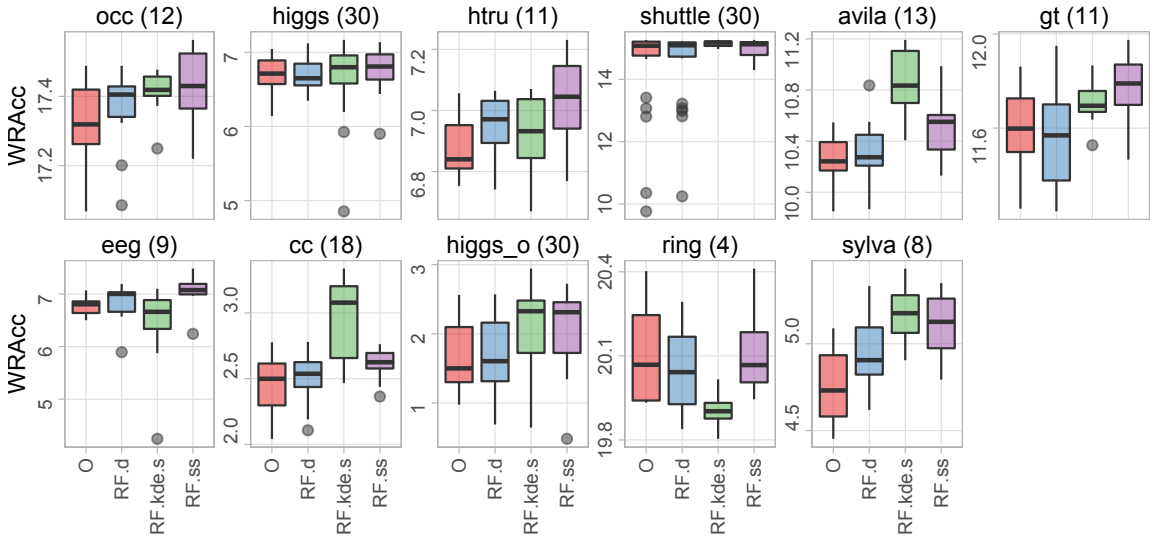


Figure 5.8: WRAcc. REDS+ with BestIntervalBS. N = 1600.

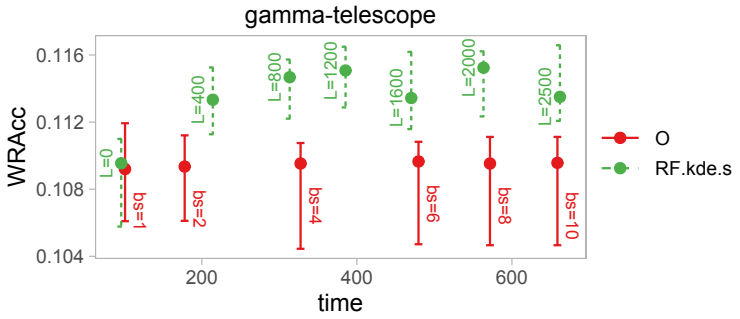


Figure 5.9: REDS+ with BestIntervalBS varying L vs just BestIntervalBS varying bs.

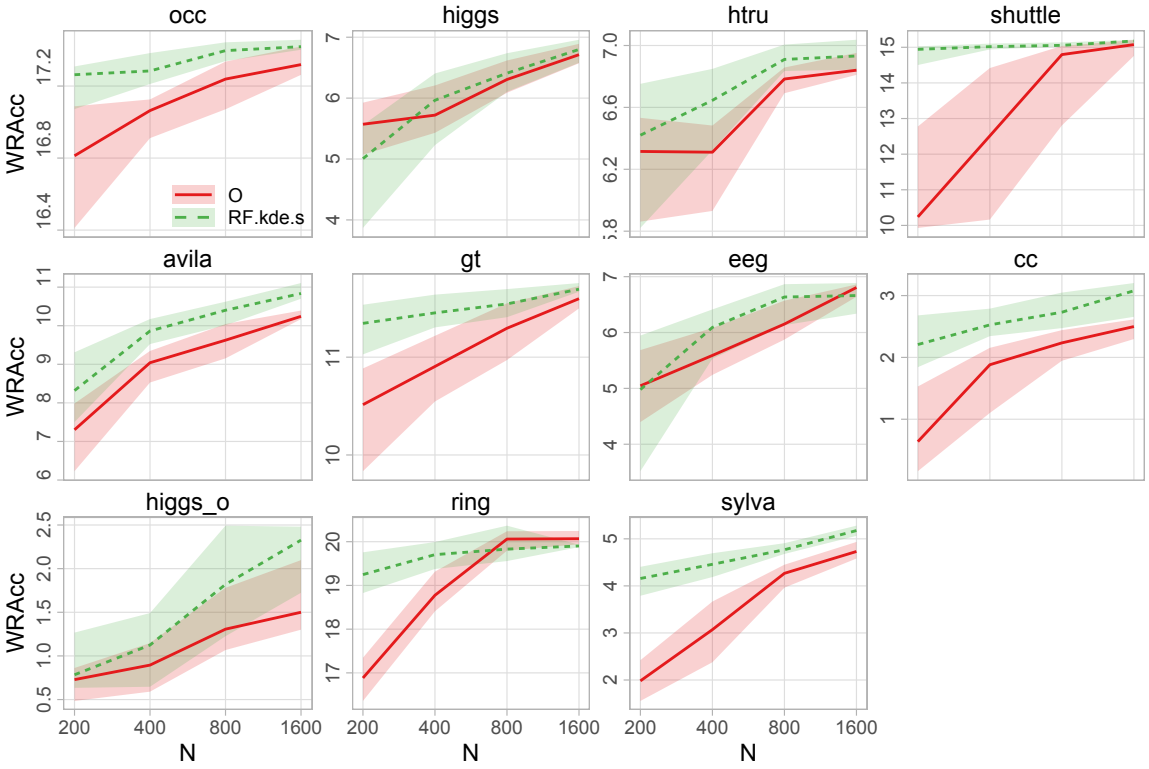


Figure 5.10: WRAcc. REDS+ with BestIntervalBS learning curves.

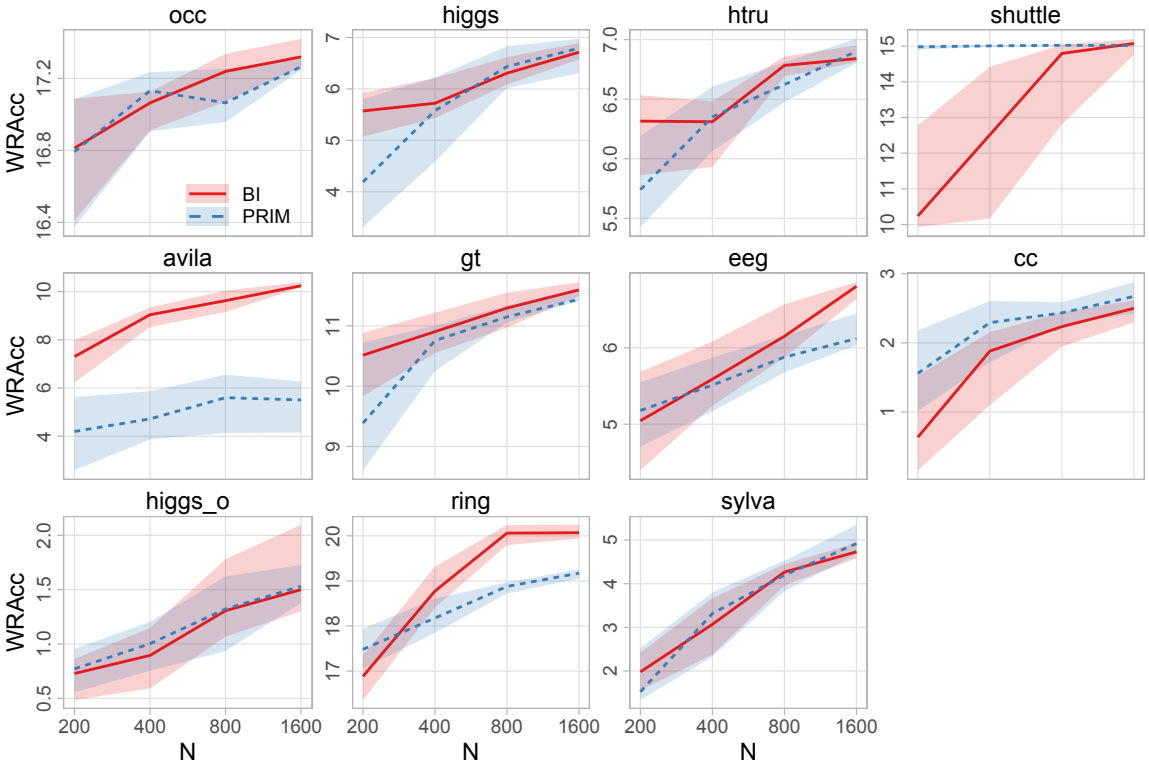


Figure 5.11: WRAcc. PRIM vs BestIntervalBS

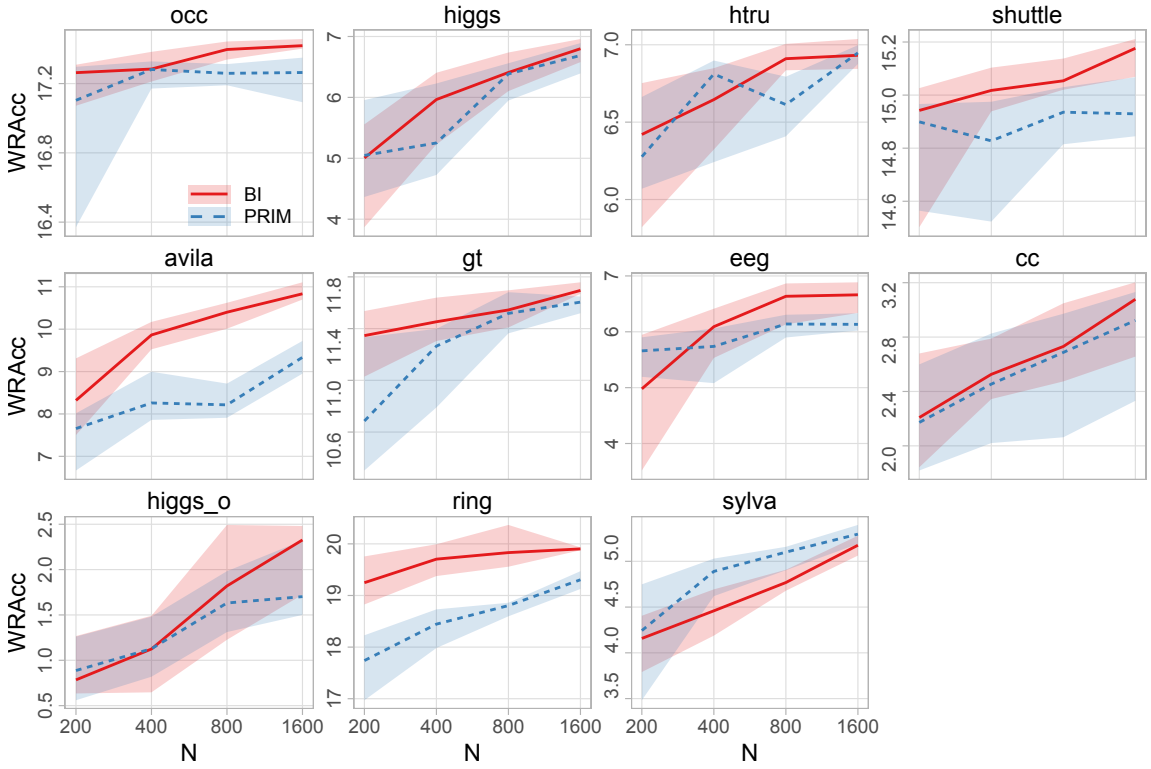


Figure 5.12: WRAcc. REDS+ with PRIM vs REDS+ with BestIntervalBS. Metamodel — "RF", data generator — "kde.s"

6 Future Research Directions

6.1 Scenario Discovery and Subgroup Discovery

Scenario Discovery. PRIM is a notable representative of rule learning [FGL12] or, more precisely, of *supervised descriptive rule discovery*, also known as subgroup discovery [Atz15], contrast set mining, emerging pattern mining [NLW09] or informative data summarization [VGBS19]. PRIM is a popular method for scenario discovery "because it is highly interactive, presents multiple options for the choice of scenarios, and provides visualizations that help users balance among the three measures of scenario quality: coverage, density, and interpretability" [BL10]. However, to the best our knowledge, there exists only a little research justifying the superiority of PRIM experimentally — Lempert et al. [LBB08] compare CART and PRIM and conclude that the latter is more interactive and requires less post-processing effort. An experimental comparison of the existing supervised descriptive rule discovery algorithms to assess their suitability for scenario discovery is an interesting research question. However, such a comparative study would require much expert knowledge, in particular, to evaluate the usefulness of discovered scenario. Additionally, this work first requires a comprehensive comparison study of subgroup discovery methods.

Subgroup Discovery. At a high level, the taxonomy of subgroup discovery methods includes specifying search space, quality function(s), and search strategy. Existing surveys [HCGdJ11, CGdJH14, Atz15] make use of these dimensions when put different subgroup discovery methods together. To our knowledge, the latest and most exhaustive study comparing SD techniques is [Hel16]. Still, it covers only six methods in the experimental part, and many questions remain open.

First, Helal [Hel16] does not describe the experimental setting with enough details. For instance, the exact procedure to discretize continuous attributes is not evident. From what we can see, the author did not separate train and test data and did not use cross-validation. That is, the conclusions were likely drawn from the same data which was used to discover subgroups. Thus, there is no guarantee that the results will hold if one uses the out-of-sample validation procedure.

Next, Helal [Hel16] does discretize numeric attributes. They also do fix the hyper-parameters of different algorithms in experiments. As Herrera et al. [HCGdJ11] write "…the discretization of the continuous variables and its influence in the results of the subgroup discovery task is another open topic. It is unclear how the previous discretization of continuous variables may affect the results of the subgroup discovery process or the advantages of the subgroup discovery algorithms that use continuous variables without any prior discretization". We now take this concern one step further. As we explained in

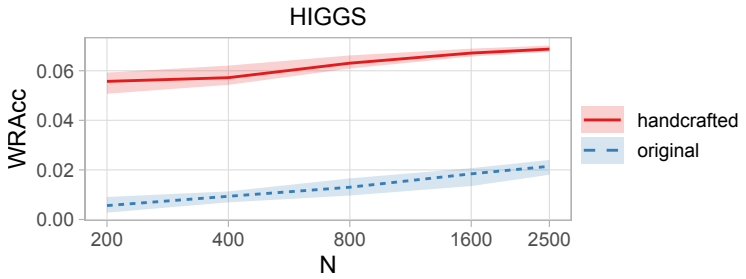


Figure 6.1: Performance of BestIntervalBS using original or handcrafted features.

Section 2.3, one can usually make the search space identical via binarization procedure and that even continuous attributes are essentially always discretized — we call this *exhaustive* discretization. Thus, the main distinction between different methods is whether they perform exhaustive or heuristic search. The former requires considerable time but finds the global optimum; the latter is faster but converge to local optimum. With more time, the solution of heuristic-based methods can be improved via increasing the share of the search space examined. This happens naturally with SD methods using evolutionary optimization as a search strategy. Alternatively, one can increase this share, by using the beam-search strategy (see Section 2.3) and controlling the beam size *bs*, like with BestIntervalBS. Conversely, one can speed up the methods performing an exhaustive search by limiting the search space. Typically, one can either limit the search depth, i.e., the maximal number of restricted dimensions, or switch to more coarse-grained discretization.

All this implies that experimenting with fixed discretization and hyperparameters like beam size or search depth is not very insightful. For instance, which strategy is (likely) the best given a time limitation remains unknown. Thus, the questions raised in [HCGdJ11] can be specified as follows. Given a time restriction, what is the best choice of discretization granularity, beam size (if applicable), search depth, and other parameters (e.g., peeling parameter α in PRIM)? To what extent does a discretization technique [GLS+13] influence the quality of a discovered subgroup? Are heuristic methods generally better than the ones based on exhaustive search? How do the answers depend on the time limitation and the dataset size?

Finally, it would be interesting to evaluate how the methodology proposed in this dissertation affects the results of such study.

6.2 Automatic Feature Construction

Motivation. From our findings as well as from earlier research, it is evident that particular kind of data preprocessing — feature construction, — can improve the results of comprehensible models. For instance, Dalal et al. [DHL+13] used PRIM in rotated space resulting from applying principal component analysis (PCA-PRIM) and found that "PCA-PRIM produces improvements averaging 37 percent [...] over PRIM alone". We now present similar results from our experience.

Example 6.2.1 *HIGGS dataset [VvRBT13, BSW14] comes with 21 original (raw) attributes and seven handcrafted features, derived as functions of the attributes. We apply* BestInter-

VALBS separately to the dataset part including attributes and to the part including features. We experiment in the same way as described in Chapter 5. Figure 6.1 shows the learning curves. As before, shaded are the areas between 25-th and 75-th percentiles in the set of outcomes from 30 experiments. The subgroups found using the handcrafted features have several times higher WRAcc than the subgroups discovered with the original attributes. The median number of restricted features is five, much less than the number of restricted attributes, 13; I.e., the subgroup descriptions are more comprehensible[1].

Example 6.2.2 *For one of the datasets used in Chapter 3*[2] *we trained the CART classifier using (a) original attributes and (b) derived features as explained in that Chapter. The average accuracies from 10-fold cross-validation are* 0.799 ± 0.009 *and* 0.823 ± 0.01*, and the number of leaves in the resulting decision trees is 19 and 9, for (a) and (b) respectively.*

We now briefly discuss the outcome of the latter example and formulate the research question. CART is a decision tree algorithm. It can approximate the "true" (i.e., minimizing given risk functional [Vap99]) decision boundary with arbitrary precision given enough data. Consequently, with a large dataset, the accuracy of the decision tree fitted with original attributes will be at least as good as that of the decision tree learned using the features derived from those attributes. This is because features do not have any new information which is not contained in the attributes. However, one hopes that in feature space, the decision boundary is in some sense "simpler", and a decision tree can adequately capture it from a smaller number of examples and with a simples structure (i.e., fewer leaves). This is the case in Example 6.2.2.

We have created the features using our knowledge of DSGC system symmetry. For an electrical grid of arbitrary size and structure, feature construction is not straightforward. This problem is not specific to our study. Hence, one wants to have an automatic feature construction process that learns useful features from data. For instance, the description of HIGGS dataset used in Example 6.2.1 states: "There is an interest in using deep learning methods to obviate the need for physicists to develop such features manually".

Definitions of Automatic Feature Construction. We briefly researched the literature related to automatic feature generation and identified the following.

First, "feature construction" is a broad term. Often it refers to various transformations of the space defined by attributes of data in a tabular form. Some feature construction methods can also accept other forms of data representation like relational databases [CKG+11, KV15, LTS+17] or a set of sequences like speech, music or natural language [BCV13].

Second, feature construction can be a separate process like in Examples 6.2.1–6.2.2; or it can be a part of an ML algorithm. For instance, kernel methods implicitly map the space of original attributes into a feature space implied by corresponding kernel function [DL18], artificial neural nets automatically discover multiple levels of representation [SRK15, ZWD16]. Even subgroup discovery or decision tree algorithms considered

[1] Adopting the convention that the number of restricted dimensions reflects comprehensibility. See Section 2.1 for the overview of research related to comprehensibility.

[2] Available from [DG17]: `https://archive.ics.uci.edu/ml/datasets/Electrical+Grid+Stability+Simulated+Data+`

in this dissertation can be treated as automatically constructing binary features from the attributes a_j by finding inequalities $a_j^l < a_j$ or $a_j^r > a_j$ [GMP14, FGL12, MR02], or by learning the conjunctions of these inequalities $\prod_{j=1}^{M}[a_j^l, a_j^r]$ [PH90, MR02].

Finally, the notion "(automatic) feature construction" (used in e.g., [SB05, NZJ07, PS09, Son09, FZP+10, NZA12, GMP14, TZX16]) has many synonyms: feature discovery [PH90, DR12], (nonlinear) dimensionality reduction [RS00, BN03, NZJ07], feature generation [MR02, CKG+11, KSS16, KMP17, DZOV18], data representation [BN03], attribute construction [OSFN03], feature learning [AEP08, CNL11, LMS+18], representation learning [BCV13][3], feature extraction [SRK15], feature synthesis [KV15], feature engineering [KNS+16, KTSP16, LTS+17, NSK+17, KST18], identifying best low-dimensional descriptor [OCA+18], manifold learning or distance metric learning [SRK15]. Related approaches also include visualization of multidimensional data [TLZM16]. Moreover, various automated machine learning methods (e.g. [OM16, SL09]) include automatic feature construction component[4].

Open Questions. It is interesting, which automatic feature construction methods can be used to design better spaces for comprehensible ML models like decision trees and subgroups. One expects that the model learned using the constructed features will have higher quality (accuracy, WRAcc, etc.) but will remain comprehensible. I.e., the discovered features should be simple functions of original attributes. Thus, many deep-learning-based feature construction algorithms might not be suitable.

So far, a little work has been done towards a comparison of different automatic feature construction methods that are not based on artificial neural nets. To the best of our knowledge, the most recent and the broadest comparison is made in [KMP17] and covers only four methods. The study does not consider comprehensibility, and the code for the experiments was not made available.

6.3 Mixed Attribute Types, Regression Setting

In the dissertation, we have focused on the classification setting, i.e., binary target variable. We further assumed that all attributes are numeric. The method REDS, which we developed for simulated data, (Chapter 4) does not algorithmically exploit these assumptions. That is, it can accept mixed attributes and numeric target. REDS+ developed in Chapter 5 can also work with a numeric target when it is used with a suitable subgroup discovery method. It does not support mixed attribute types directly; still, the adaptation to work with such data is possible, as we explained in Section 5.3.4. However, the question of how good the proposed methods work in the other settings and if they could be further improved by taking the specifics of each particular setting into account remains open.

[3] According to [SRK15], "representation learning could refer to the entire literature of extracting features from input data, however, in practice it is usually associated with extracting features via multi-layer neural networks and is studied within neural network research community"

[4] This diversity of names often complicates the search of related work. For example, Katz et al. [KSS16] have indicated only a single related research paper, which is arguable given the references to earlier literature we just mentioned.

7 Conclusions

Comprehensible statistical models constitute an essential part of knowledge discovery from databases. Decision trees and subgroups, the main focus of this work, are often a human-comprehensible form of knowledge representation.

To show the utility of metamodels, we studied the model of a novel system, DSGC, realizing demand response in electrical grids. We run many simulations for random combinations inputs' values, constructed features from these inputs and trained a decision tree on the resulting dataset. This allowed us to obtain new insights, not known from conventional analysis of DSGC done before. For example, we discover that the system can be stable even if some participants adapt their energy consumption with a high delay, or that fast adaptation is preferable for system stability. Additionally, we have collected a comprehensive list of current assumptions and open questions behind DSGC.

Decision trees partition the entire input space into regions. Sometimes one does not aim to achieve such partitioning. Instead, one may want to find areas, as large as possible, but likely to be responsible for a particular outcome. Subgroup discovery methods are better suitable for this purpose, but they require large datasets to produce a high-quality outcome. However, large datasets often come at high computational (for simulated data) or financial (for measured data) cost. We developed two approaches, REDS and its generalization REDS+, which can find better subgroups from small datasets of simulated and measured data respectively. We provided a statistical intuition behind our approaches; in the experiments, they outperformed existing subgroup discovery methods. For some datasets, REDS and REDS+ were able to discover subgroups of certain quality from more than four times fewer examples than conventional algorithms.

In contrast to existing subgroup discovery algorithms, REDS is applicable to a semi-supervised setting and is compatible with the existing active learning techniques.

List of Figures

List of Tables

Bibliography

[AB21] Vadim Arzamasov and Klemens Böhm. REDS: rule extraction for discovering scenarios. In *SIGMOD Conference*, pages 115–128. ACM, 2021.

[ABJ18] Vadim Arzamasov, Klemens Böhm, and Patrick Jochem. Towards concise models of grid stability. In *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids, SmartGridComm 2018, Aalborg, Denmark, October 29-31, 2018*, pages 1–6, 2018.

[AEP08] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Mach. Learn.*, 73(3):243–272, 2008.

[AKV19] Stamatios-Aggelos N. Alexandropoulos, Sotiris B. Kotsiantis, and Michael N. Vrahatis. Data preprocessing in predictive data mining. *Knowledge Eng. Review*, 34:e1, 2019.

[ALM+01] Ralph G Andrzejak, Klaus Lehnertz, Florian Mormann, Christoph Rieke, Peter David, and Christian E Elger. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Physical Review E*, 64(6):061907, 2001.

[AO01] Jian An and Art B. Owen. Quasi-regression. *J. Complex.*, 17(4):588–607, 2001.

[AP06] Martin Atzmüller and Frank Puppe. SD-Map — A fast algorithm for exhaustive subgroup discovery. In *Knowledge Discovery in Databases: PKDD 2006, 10th European Conference on Principles and Practice of Knowledge Discovery in Databases, Berlin, Germany, September 18-22, 2006, Proceedings*, pages 6–17, 2006.

[AR13] Turaj Amraee and Soheil Ranjbar. Transient instability prediction using decision tree technique. *IEEE Transactions on power systems*, 28(3):3028–3037, 2013.

[Atz15] Martin Atzmueller. Subgroup discovery. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, 5(1):35–49, 2015.

[B+01] Leo Breiman et al. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science*, 16(3):199–231, 2001.

[BAS07] Einat Neumann Ben-Ari and David M Steinberg. Modeling data from computer experiments: An empirical comparison of kriging with mars and projection pursuit regression. *Quality Engineering*, 19(4):327–338, 2007.

[BCN06] Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, pages 535–541, 2006.

[BCV13] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, 2013.

[BFOS84] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[BH99] Michael R. Berthold and Klaus-Peter Huber. Constructing fuzzy graphs from examples. *Intell. Data Anal.*, 3(1):37–53, 1999.

[BL10] Benjamin P Bryant and Robert J Lempert. Thinking inside the box: a participatory, computer-assisted approach to scenario discovery. *Technological Forecasting and Social Change*, 77(1):34–49, 2010.

[BN03] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[Bre96] Leo Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, 1996.

[Bre01] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001.

[BRH19] Marcus D. Bloice, Peter M. Roth, and Andreas Holzinger. Biomedical image augmentation using Augmentor. *Bioinform.*, 35(21):4522–4524, 2019.

[BS19] Murad Badarna and Ilan Shimshoni. Selective sampling for trees and forests. *Neurocomputing*, 358:93–108, 2019.

[BSH+10] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. How to explain individual classification decisions. *J. Mach. Learn. Res.*, 11:1803–1831, 2010.

[BSW14] Pierre Baldi, Peter Sadowski, and Daniel Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5:4308, 2014.

[Bur98] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, 1998.

[CF16] Luis M Candanedo and Véronique Feldheim. Accurate occupancy detection of an office room from light, temperature, humidity and CO_2 measurements using statistical learning models. *Energy and Buildings*, 112:28–39, 2016.

[CGdJ+11] Cristóbal J. Carmona, Pedro González, María José del Jesus, M. Navío-Acosta, and L. Jiménez-Trevino. Evolutionary fuzzy rule extraction for subgroup discovery in a psychiatric emergency department. *Soft Comput.*, 15(12):2435–2448, 2011.

[CGdJH14] Cristóbal J. Carmona, Pedro González, María José del Jesús, and Francisco Herrera. Overview on evolutionary subgroup discovery: Analysis of the suitability and potential of the search performed by evolutionary algorithms. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, 4(2):87–103, 2014.

[CGK15] Xiaodong Cui, Vaibhava Goel, and Brian Kingsbury. Data augmentation for deep convolutional neural network acoustic modeling. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015*, pages 4545–4549, 2015.

[CKG+11] Weiwei Cheng, Gjergji Kasneci, Thore Graepel, David H. Stern, and Ralf Herbrich. Automated feature generation from structured knowledge. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011*, pages 1395–1404, 2011.

[CLG+15] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 1721–1730, 2015.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition.* The MIT Press and McGraw-Hill Book Company, 2001.

[CNL11] Adam Coates, Andrew Y. Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pages 215–223, 2011.

[CS95] Mark W. Craven and Jude W. Shavlik. Extracting tree-structured representations of trained networks. In *Advances in Neural Information Processing Systems 8, NIPS, Denver, CO, USA, November 27-30, 1995*, pages 24–30, 1995.

[CSZ06] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors. *Semi-Supervised Learning*. The MIT Press, 2006.

[dFM15] Enric Junqué de Fortuny and David Martens. Active learning-based pedagogical rule extraction. *IEEE Trans. Neural Netw. Learning Syst.*, 26(11):2664–2677, 2015.

[DG17] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[DHL+13] Siddhartha R. Dalal, Bing Han, Robert J. Lempert, Amber Jaycocks, and Andrew Hackbarth. Improving scenario discovery using orthogonal rotations. *Environ. Model. Softw.*, 48:49–64, 2013.

[DL18] Guozhu Dong and Huan Liu. *Feature engineering for machine learning and data analytics*. CRC Press, 2018.

[DLR77] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[Dom97] Pedro Domingos. Knowledge acquisition form examples vis multiple models. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 98–106. Morgan Kaufmann Publishers Inc., 1997.

[Dom12] Pedro M. Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55(10):78–87, 2012.

[DR12] Ofer Dor and Yoram Reich. Strengthening learning algorithms by feature discovery. *Inf. Sci.*, 189:176–190, 2012.

[DVK17] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *CoRR*, abs/1702.08608, 2017.

[DZOV18] Jiayi Duan, Ziheng Zeng, Alina Oprea, and Shobha Vasudevan. Automated generation and selection of interpretable features for enterprise security. In *IEEE International Conference on Big Data, Big Data 2018, Seattle, WA, USA, December 10-13, 2018*, pages 1258–1265, 2018.

[FF99] Jerome H. Friedman and Nicholas I. Fisher. Bump hunting in high-dimensional data. *Statistics and Computing*, 9(2):123–143, 1999.

[FGL12] Johannes Fürnkranz, Dragan Gamberger, and Nada Lavrac. *Foundations of Rule Learning*. Cognitive Technologies. Springer, 2012.

[FNP08] Giovanni Filatrella, Arne Hejde Nielsen, and Niels Falsig Pedersen. Analysis of a power grid using a Kuramoto-like model. *The European Physical Journal B*, 61(4):485–491, 2008.

[Fre13] Alex Alves Freitas. Comprehensible classification models: a position paper. *SIGKDD Explorations*, 15(1):1–10, 2013.

[FSK08] Alexander I. J. Forrester, Andras Sobester, and Andy J. Keane. *Engineering Design via Surrogate Modelling — A Practical Guide*. Wiley, 2008.

[FZP+10] Wei Fan, Erheng Zhong, Jing Peng, Olivier Verscheure, Kun Zhang, Jiangtao Ren, Rong Yan, and Qiang Yang. Generalized and heuristic-free feature construction for improved accuracy. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2010, April 29 - May 1, 2010, Columbus, Ohio, USA*, pages 629–640, 2010.

[GCD+10] Dirk Gorissen, Ivo Couckuyt, Piet Demeester, Tom Dhaene, and Karel Crombecq. A surrogate modeling and adaptive sampling toolbox for computer based design. *J. Mach. Learn. Res.*, 11:2051–2055, 2010.

[GG15] Jian Guo and Stephen Gould. Deep CNN ensemble with data augmentation for object detection. *CoRR*, abs/1506.07224, 2015.

[GL02] Dragan Gamberger and Nada Lavrac. Expert-guided subgroup discovery: Methodology and application. *J. Artif. Intell. Res.*, 17:501–527, 2002.

[GL07] David G Groves and Robert J Lempert. A new analytic method for finding policy-relevant scenarios. *Global Environmental Change*, 17(1):73–85, 2007.

[GLS+13] Salvador García, Julián Luengo, José Antonio Sáez, Victoria López, and Francisco Herrera. A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *IEEE Trans. Knowl. Data Eng.*, 25(4):734–750, 2013.

[GMMP20] Riccardo Guidotti, Anna Monreale, Stan Matwin, and Dino Pedreschi. Black box explanation by learning image exemplars in the latent feature space. *CoRR*, abs/2002.03746, 2020.

[GMP14] David García, Antonio González Muñoz, and Raúl Pérez. A feature construction approach for genetic iterative rule learning algorithm. *J. Comput. Syst. Sci.*, 80(1):101–117, 2014.

[GMR+18] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. Local rule-based explanations of black box decision systems. *CoRR*, abs/1805.10820, 2018.

[GMR+19] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5):93:1–93:42, 2019.

[GR09] Henrik Grosskreutz and Stefan Rüping. On subgroup discovery in numerical domains. *Data Min. Knowl. Discov.*, 19(2):210–226, 2009.

[GRS16] Céline Guivarch, Julie Rozenberg, and Vanessa Schweizer. The diversity of socio-economic pathways and CO_2 emissions scenarios: Insights from the investigation of a scenarios database. *Environ. Model. Softw.*, 80:336–353, 2016.

[GRW08] Henrik Grosskreutz, Stefan Rüping, and Stefan Wrobel. Tight optimistic estimates for fast subgroup discovery. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML/PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part I*, pages 440–456, 2008.

[Gun16] D Gunning. Explainable artificial intelligence (XAI) DARPA-BAA-16-53. *Defense Advanced Research Projects Agency*, 2016.

[HBV06] Johan Huysmans, Bart Baesens, and Jan Vanthienen. Using rule extraction to improve the comprehensibility of predictive models. *Available at SSRN 961358*, 2006.

[HCGdJ11] Francisco Herrera, Cristóbal J. Carmona, Pedro González, and María José del Jesús. An overview on subgroup discovery: foundations and applications. *Knowl. Inf. Syst.*, 29(3):495–525, 2011.

[HCL+03] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification, 2003.

[HDM+11] Johan Huysmans, Karel Dejaeger, Christophe Mues, Jan Vanthienen, and Bart Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decis. Support Syst.*, 51(1):141–154, 2011.

[Hel16] Sumyea Helal. Subgroup discovery algorithms: A survey and empirical evaluation. *J. Comput. Sci. Technol.*, 31(3):561–576, 2016.

[HHRK15] David Hadka, Jonathan D. Herman, Patrick M. Reed, and Klaus Keller. An open source framework for many-objective robust decision making. *Environ. Model. Softw.*, 74:114–129, 2015.

[HRZC15] Jonathan D Herman, Patrick M Reed, Harrison B Zeff, and Gregory W Characklis. How should robustness be defined for water systems planning under change? *Journal of Water Resources Planning and Management*, 141(10):04015012, 2015.

[HS64] J. H. Halton and G. B. Smith. Algorithm 247: Radical-inverse quasi-random point sequence. *Commun. ACM*, 7(12):701–702, 1964.

[HSBV08] Johan Huysmans, Rudy Setiono, Bart Baesens, and Jan Vanthienen. Minerva: Sequential covering for rule extraction. *IEEE Trans. Systems, Man, and Cybernetics, Part B*, 38(2):299–309, 2008.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition.* Springer Series in Statistics. Springer, 2009.

[HVD15] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

[IH90] Tsutomu Ishigami and Toshimitsu Homma. An importance quantification technique in uncertainty analysis for computer models. In *[1990] Proceedings. First International Symposium on Uncertainty Modeling and Analysis*, pages 398–403. IEEE, 1990.

[III65] H. J. Scudder III. Probability of error of some adaptive pattern-recognition machines. *IEEE Trans. Inf. Theory*, 11(3):363–371, 1965.

[IP16] Tushith Islam and Erik Pruyt. Scenario generation using adaptive sampling: The case of resource scarcity. *Environ. Model. Softw.*, 79:285–299, 2016.

[JA06] Bathiya Jayasekara and Udaya D Annakkage. Derivation of an accurate polynomial representation of the transient stability boundary. *IEEE transactions on Power Systems*, 21(4):1856–1863, 2006.

[JMK12] Rachel T Johnson, Douglas C Montgomery, and Kathryn S Kennedy. Hybrid space-filling designs for computer experiments. In *Frontiers in Statistical Quality Control 10*, pages 287–301. Springer, 2012.

[Kam14] Oliver Kamps. Characterizing the fluctuations of wind power production by multi-time statistics. In *Wind Energy-Impact of Turbulence*, pages 67–72. Springer, 2014.

[KBL94] Prabha Kundur, Neal J Balu, and Mark G Lauby. *Power system stability and control*, volume 7. McGraw-hill New York, 1994.

[KC16] JH Kwakkel and SC Cunningham. Improving scenario discovery by bagging random boxes. *Technological Forecasting and Social Change*, 111:124–134, 2016.

[KJ16] Jan H. Kwakkel and Marc Jaxa-Rozen. Improving scenario discovery for handling heterogeneous uncertainties and multinomial classified outcomes. *Environ. Model. Softw.*, 79:311–321, 2016.

[Kle15] Jack PC Kleijnen. *Design and Analysis of Simulation Experiments*, volume 230. Springer, 2015.

[KLJ03] Branko Kavsek, Nada Lavrac, and Viktor Jovanoski. APRIORI-SD: Adapting association rule learning to subgroup discovery. In *Advances in Intelligent Data Analysis V, 5th International Symposium on Intelligent Data Analysis, IDA 2003, Berlin, Germany, August 28-30, 2003, Proceedings*, pages 230–241, 2003.

[KLZG05] Petra Kralj, Nada Lavrac, Blaz Zupan, and Dragan Gamberger. Experimental comparison of three subgroup discovery algorithms: Analysing brain ischemia data. *Information Society*, pages 220–223, 2005.

[KMP17] Ambika Kaul, Saket Maheshwary, and Vikram Pudi. AutoLearn — automated feature generation and selection. In *2017 IEEE International Conference on Data Mining, ICDM 2017, New Orleans, LA, USA, November 18-21, 2017*, pages 217–226, 2017.

[KNS+16] Udayan Khurana, Fatemeh Nargesian, Horst Samulowitz, Elias Khalil, and Deepak Turaga. Automating feature engineering. *Transformation*, 10(10):10, 2016.

[KP13] Jan H Kwakkel and Erik Pruyt. Exploratory modeling and analysis, an approach for model-based foresight under deep uncertainty. *Technological Forecasting and Social Change*, 80(3):419–431, 2013.

[KS13] Miroslav Kobetski and Josephine Sullivan. Apprenticeship learning: Transfer of knowledge via dataset augmentation. In *Image Analysis, 18th Scandinavian Conference, SCIA 2013, Espoo, Finland, June 17-20, 2013. Proceedings*, pages 432–443, 2013.

[KSS16] Gilad Katz, Eui Chul Richard Shin, and Dawn Song. ExploreKit: Automatic feature generation and selection. In *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*, pages 979–984, 2016.

[KST18] Udayan Khurana, Horst Samulowitz, and Deepak S. Turaga. Feature engineering for predictive modeling using reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3407–3414, 2018.

[KTSP16] Udayan Khurana, Deepak S. Turaga, Horst Samulowitz, and Srinivasan Parthasrathy. Cognito: Automated feature engineering for supervised learning. In *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain*, pages 1304–1307, 2016.

[KV15] James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2015, Campus des Cordeliers, Paris, France, October 19-21, 2015*, pages 1–10, 2015.

[LBB08] Robert J Lempert, Benjamin P Bryant, and Steven C Bankes. Comparing algorithms for scenario discovery. *RAND, Santa Monica, CA*, 2008.

[LBH+06] Crystal Linkletter, Derek Bingham, Nicolas W. Hengartner, David Higdon, and Kenny Q. Ye. Variable selection for gaussian process models in computer experiments. *Technometrics*, 48(4):478–490, 2006.

[Lee13] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 2, 2013.

[Ley02] Michael Ley. The DBLP computer science bibliography: Evolution, research issues, perspectives. In *String Processing and Information Retrieval, 9th International Symposium, SPIRE 2002, Lisbon, Portugal, September 11-13, 2002, Proceedings*, pages 1–10, 2002.

[LG04] Nada Lavrac and Dragan Gamberger. Relevancy in constraint-based subgroup discovery. In *Constraint-Based Mining and Inductive Databases, European Workshop on Inductive Databases and Constraint Based Mining, Hinterzarten, Germany, March 11-13, 2004, Revised Selected Papers*, pages 243–266, 2004.

[LGPB06] Robert J. Lempert, David G. Groves, Steven W. Popper, and Steve Bankes. A general, analytic method for generating robust strategies and narrative scenarios. *Management Science*, 52(4):514–528, 2006.

[Lij07] Mark G Lijesen. The real-time price elasticity of electricity. *Energy economics*, 29(2):249–258, 2007.

[Lip18] Zachary C. Lipton. The mythos of model interpretability. *Commun. ACM*, 61(10):36–43, 2018.

[LKFT04] Nada Lavrac, Branko Kavsek, Peter A. Flach, and Ljupco Todorovski. Subgroup discovery with CN2-SD. *J. Mach. Learn. Res.*, 5:153–188, 2004.

[LMS+18] Hoang Thanh Lam, Tran Ngoc Minh, Mathieu Sinn, Beat Buesser, and Martin Wistuba. Learning features for relational data. *CoRR*, abs/1801.05372, 2018.

[LRRV13] José María Luna, José Raúl Romero, Cristóbal Romero, and Sebastián Ventura. Discovering subgroups by means of genetic programming. In *Genetic Programming - 16th European Conference, EuroGP 2013, Vienna, Austria, April 3-5, 2013. Proceedings*, pages 121–132, 2013.

[LRRV14] José María Luna, José Raúl Romero, Cristóbal Romero, and Sebastián Ventura. On the use of genetic programming for mining comprehensible rules in subgroup discovery. *IEEE Trans. Cybernetics*, 44(12):2329–2341, 2014.

[LSC+16] Robert J Lyon, BW Stappers, Sally Cooper, JM Brooke, and JD Knowles. Fifty years of pulsar candidate selection: From simple filters to a new principled real-time classification approach. *Monthly Notices of the Royal Astronomical Society*, 459(1):1104–1123, 2016.

[LTS+17] Hoang Thanh Lam, Johann-Michael Thiebaut, Mathieu Sinn, Bei Chen, Tiep Mai, and Oznur Alkan. One button machine for automating feature engineering in relational databases. *CoRR*, abs/1706.00327, 2017.

[LWM13] Jason L. Loeppky, Brian J. Williams, and Leslie M. Moore. Global sensitivity analysis for mixture experiments. *Technometrics*, 55(1):68–78, 2013.

[LXL+12] Jing Liu, Yang Xiao, Shuhui Li, Wei Liang, and C. L. Philip Chen. Cyber security and privacy issues in smart grids. *IEEE Communications Surveys and Tutorials*, 14(4):981–997, 2012.

[MDSESM04] LS Moulin, AP Alves Da Silva, MA El-Sharkawi, and Robert J Marks. Support vector machines for transient stability analysis of large-scale power systems. *IEEE Transactions on Power Systems*, 19(2):818–825, 2004.

[MHMK13] Peter J Menck, Jobst Heitzig, Norbert Marwan, and Jürgen Kurths. How basin stability complements the linear-stability paradigm. *Nature physics*, 9(2):89–92, 2013.

[Mil19] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.*, 267:1–38, 2019.

[MMM06] Max D Morris, Leslie M Moore, and Michael D McKay. Sampling plans based on balanced incomplete block designs for evaluating the importance of computer model inputs. *Journal of Statistical Planning and Inference*, 136(9):3203–3220, 2006.

[MNFK12] Michael Mampaey, Siegfried Nijssen, Ad Feelders, and Arno J. Knobbe. Efficient algorithms for finding richer subgroup descriptions in numeric and nominal data. In Mohammed Javeed Zaki, Arno Siebes, Jeffrey Xu Yu, Bart Goethals, Geoffrey I. Webb, and Xindong Wu, editors, *12th IEEE International Conference on Data Mining, ICDM 2012, Brussels, Belgium, December 10-13, 2012*, pages 499–508. IEEE Computer Society, 2012.

[Moo10] Hyejung Moon. *Design and analysis of computer experiments for screening input variables*. Ohio State University, 2010.

[MR02] Shaul Markovitch and Dan Rosenstein. Feature generation using general constructor functions. *Mach. Learn.*, 49(1):59–98, 2002.

[MR14] Geoffrey J. McLachlan and Suren I. Rathnayake. On the number of components in a Gaussian mixture model. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, 4(5):341–355, 2014.

[MWS+14] Debsankha Manik, Dirk Witthaut, Benjamin Schäfer, Moritz Matthiae, Andreas Sorge, Martin Rohden, Eleni Katifori, and Marc Timme. Supply networks: Instabilities without overload. *The European Physical Journal Special Topics*, 223(12):2527–2547, 2014.

[MWZ+97] JD McCalley, Shimo Wang, Q-L Zhao, G-Z Zhou, RT Treinen, and AD Papalexopoulos. Security boundary visualization for systems operation. *IEEE Transactions on Power Systems*, 12(2):940–947, 1997.

[NC05] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*, pages 625–632, 2005.

[NLW09] Petra Kralj Novak, Nada Lavrac, and Geoffrey I. Webb. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *J. Mach. Learn. Res.*, 10:377–403, 2009.

[NM15] Takashi Nishikawa and Adilson E Motter. Comparative analysis of existing models for power-grid synchronization. *New Journal of Physics*, 17(1):015012, 2015.

[NSK+17] Fatemeh Nargesian, Horst Samulowitz, Udayan Khurana, Elias B. Khalil, and Deepak S. Turaga. Learning feature engineering for classification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 2529–2535, 2017.

[NZA12] Kourosh Neshatian, Mengjie Zhang, and Peter Andreae. A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming. *IEEE Trans. Evolutionary Computation*, 16(5):645–661, 2012.

[NZJ07] Kourosh Neshatian, Mengjie Zhang, and Mark Johnston. Feature construction and dimension reduction using genetic programming. In *AI 2007: Advances in Artificial Intelligence, 20th Australian Joint Conference on Artificial Intelligence, Gold Coast, Australia, December 2-6, 2007, Proceedings*, pages 160–170, 2007.

[OCA+18] Runhai Ouyang, Stefano Curtarolo, Emre Ahmetcik, Matthias Scheffler, and Luca M Ghiringhelli. SISSO: A compressed-sensing method for identifying the best low-dimensional descriptor in an immensity of offered candidates. *Physical Review Materials*, 2(8):083802, 2018.

[OCO+17] Randal S. Olson, William G. La Cava, Patryk Orzechowski, Ryan J. Urbanowicz, and Jason H. Moore. PMLB: a large benchmark suite for machine learning evaluation and comparison. *BioData Mining*, 10(1):36:1–36:13, 2017.

[OM16] Randal S. Olson and Jason H. Moore. TPOT: A tree-based pipeline optimization tool for automating machine learning. In *Proceedings of the 2016 Workshop on Automatic Machine Learning, AutoML 2016, co-located with 33rd International Conference on Machine Learning (ICML 2016), New York City, NY, USA, June 24, 2016*, pages 66–74, 2016.

[OO04] Jeremy E Oakley and Anthony O'Hagan. Probabilistic sensitivity analysis of complex models: A bayesian approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66(3):751–769, 2004.

[OOR+18] Avital Oliver, Augustus Odena, Colin Raffel, Ekin Dogus Cubuk, and Ian J. Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 3239–3250, 2018.

[OSFN03] Fernando E. B. Otero, Monique M. S. Silva, Alex Alves Freitas, and Júlio C. Nievola. Genetic programming for attribute construction in data mining. In *Genetic Programming, 6th European Conference, EuroGP 2003, Essex, UK, April 14-16, 2003. Proceedings*, pages 384–393, 2003.

[P+99] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[P+13] Victor Picheny et al. A benchmark of kriging-based infill criteria for noisy optimization. *Structural and Multidisciplinary Optimization*, 48(3), 2013.

[PCZ+19] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. SpecAugment: A simple data augmentation method for automatic speech recognition. In *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 2613–2617, 2019.

[PH90] Giulia Pagallo and David Haussler. Boolean feature discovery in empirical learning. *Mach. Learn.*, 5:71–99, 1990.

[Pie92] Henri Pierreval. Rule-based simulation metamodels. *European Journal of Operational Research*, 61(1-2):6–17, 1992.

[PMS97] M Pazzani, Subramani Mani, and WR Shankle. Comprehensible knowledge discovery in databases. In *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society*, pages 596–601, 1997.

[PS09] Selwyn Piramuthu and Riyaz T. Sikora. Iterative feature construction for improving inductive learning algorithms. *Expert Syst. Appl.*, 36(2):3401–3406, 2009.

[PSLB15] Andrew M Parker, Sinduja V Srinivasan, Robert J Lempert, and Sandra H Berry. Evaluating simulation-derived scenarios for effective decision support. *Technological Forecasting and Social Change*, 91:64–77, 2015.

[PTR95] WW Price, CW Taylor, and GJ Rogers. Standard load models for power flow and dynamic performance simulation. *IEEE Transactions on power systems*, 10(CONF-940702-), 1995.

[PVG+11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron

Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, 2011.

[RRM99] Greg Ridgeway, Thomas Richardson, and David Madigan. Discussion on the paper by Friedman and Fisher. *Statistics and Computing*, 9(2):150–152, 1999.

[RRRA12] Daniel Rodríguez, Roberto Ruiz, José C. Riquelme, and Jesús S. Aguilar-Ruiz. Searching for rules to detect defective modules: A subgroup discovery approach. *Inf. Sci.*, 191:14–30, 2012.

[RS00] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[RSG16] Marco Túlio Ribeiro, Sameer Singh, and Carlos 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, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

[RSTW12] Martin Rohden, Andreas Sorge, Marc Timme, and Dirk Witthaut. Self-organized synchronization in decentralized power grids. *Physical review letters*, 109(6):064101, 2012.

[Rud19] Cynthia 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.

[RW06] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006.

[S+78] Gideon Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[S+00] Andrea Saltelli et al. *Sensitivity Analysis*. 2000.

[SB05] Matthew Goble Smith and Larry Bull. Genetic programming with a genetic algorithm for feature construction and selection. *Genetic Programming and Evolvable Machines*, 6(3):265–281, 2005.

[SB13] S. Surjanovic and D. Bingham. Virtual library of simulation experiments: Test functions and datasets, 2013.

[See00] Matthias Seeger. Learning with labeled and unlabeled data (technical report). *Edinburgh University*, 2000.

[Set09] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[SGA+16] Benjamin Schäfer, Carsten Grabow, Sabine Auer, Jürgen Kurths, Dirk Witthaut, and Marc Timme. Taming instabilities in power grid networks by decentralized control. *The European Physical Journal Special Topics*, 225(3):569–582, 2016.

[Sil86] Bernard W. Silverman. *Density Estimation for Statistics and Data Analysis*. Springer, 1986.

[SKFK16] Hao Song, Meelis Kull, Peter A. Flach, and Georgios Kalogridis. Subgroup discovery with proper scoring rules. In Paolo Frasconi, Niels Landwehr, Giuseppe Manco, and Jilles Vreeken, editors, *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part II*, volume 9852 of *Lecture Notes in Computer Science*, pages 492–510. Springer, 2016.

[SL97] Rudy Setiono and Huan Liu. NeuroLinear: From neural networks to oblique decision rules. *Neurocomputing*, 17(1):1–24, 1997.

[SL99] IM Sobol and Yu L Levitan. On the use of variance reducing multipliers in monte carlo computations of a global sensitivity index. *Computer Physics Communications*, 117(1):52–61, 1999.

[SL09] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

[SMFdF18] Claudio De Stefano, Marilena Maniaci, Francesco Fontanella, and Alessandra Scotto di Freca. Reliable writer identification in medieval manuscripts through page layout features: The "avila" bible case. *Eng. Appl. Artif. Intell.*, 72:99–110, 2018.

[SMTW15] Benjamin Schäfer, Moritz Matthiae, Marc Timme, and Dirk Witthaut. Decentral smart grid control. *New journal of physics*, 17(1):015002, 2015.

[Son09] Parikshit Sondhi. Feature construction methods: A survey. Technical report, Univeristy of Illinois at Urbana Champaign, 2009.

[SPFK13] Katrin Schmietendorf, Joachim Peinke, Rudolf Friedrich, and Oliver Kamps. Self-organized synchronization and voltage stability in networks of synchronous machines. *CoRR*, abs/1307.2748, 2013.

[SPKA01] Timothy W. Simpson, J. D. Poplinski, P. N. Koch, and J. K. Allen. Metamodels for computer-based engineering design: Survey and recommendations. *Eng. Comput. (Lond.)*, 17(2):129–150, 2001.

[SRK15] Dmitry Storcheus, Afshin Rostamizadeh, and Sanjiv Kumar. A survey of modern questions and challenges in feature extraction. In *Proceedings of the 1st Workshop on Feature Extraction: Modern Questions and Challenges, FE 2015, co-located with the 29th Annual Conference on Neural Information Processing Systems (NIPS 2015), Montreal, Canada, December 11-12, 2015*, pages 1–18, 2015.

[SS83] GO Schneller and GP Sphicas. Decision making under uncertainty: Starr's domain criterion. *Theory and Decision*, 15(4):321–336, 1983.

[SSM05] J. Sayyad Shirabad and T.J. Menzies. The PROMISE Repository of Software Engineering Databases. School of Information Technology and Engineering, University of Ottawa, Canada, 2005.

[SSP03] Patrice Y. Simard, David Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *7th International Conference on Document Analysis and Recognition (ICDAR 2003), 2-Volume Set, 3-6 August 2003, Edinburgh, Scotland, UK*, pages 958–962, 2003.

[Sta63] Martin Kenneth Starr. *Product design and decision theory*. Prentice-Hall, 1963.

[Ste87] Michael Stein. Large sample properties of simulations using latin hypercube sampling. *Technometrics*, 29(2):143–151, 1987.

[SWN03] Thomas J. Santner, Brian J. Williams, and William I. Notz. *The Design and Analysis of Computer Experiments*. Springer series in statistics. Springer, 2003.

[TGH15] Isaac Triguero, Salvador García, and Francisco Herrera. Self-labeled techniques for semi-supervised learning: Taxonomy, software and empirical study. *Knowl. Inf. Syst.*, 42(2):245–284, 2015.

[TGLS16] Evelina Trutnevyte, Céline Guivarch, Robert Lempert, and Neil Strachan. Reinvigorating the scenario technique to expand uncertainty consideration. *Climatic change*, 135(3-4):373–379, 2016.

[TK01] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66, 2001.

[TLZM16] Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11–15, 2016*, pages 287–297, 2016.

[Tuk77] John W. Tukey. *Exploratory data analysis*. Addison-Wesley series in behavioral science : quantitative methods. Addison-Wesley, 1977.

[TZX16] Binh Tran, Mengjie Zhang, and Bing Xue. Multiple feature construction in classification on high-dimensional data using GP. In *2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016, Athens, Greece, December 6-9, 2016*, pages 1–8, 2016.

[UBA14] Andreas Ulbig, Theodor S Borsche, and Göran Andersson. Impact of low rotational inertia on power system stability and operation. *IFAC Proceedings Volumes*, 47(3):7290–7297, 2014.

[ULHM15] Laura Uusitalo, Annukka Lehikoinen, Inari Helle, and Kai Myrberg. An overview of methods to evaluate uncertainty of deterministic models in decision support. *Environ. Model. Softw.*, 63:24–31, 2015.

[UR16] Berk Ustun and Cynthia Rudin. Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, 102(3):349–391, 2016.

[Vap99] Vladimir Vapnik. An overview of statistical learning theory. *IEEE Trans. Neural Networks*, 10(5):988–999, 1999.

[VGBS19] Michael Vollmer, Lukasz Golab, Klemens Böhm, and Divesh Srivastava. Informative summarization of numeric data. In *Proceedings of the 31st International Conference on Scientific and Statistical Database Management, SSDBM 2019, Santa Cruz, CA, USA, July 23-25, 2019*, pages 97–108, 2019.

[VI19] Vladimir Vapnik and Rauf Izmailov. Rethinking statistical learning theory: learning using statistical invariants. *Machine Learning*, 108(3):381–423, 2019.

[vLK12] Matthijs van Leeuwen and Arno J. Knobbe. Diverse subgroup set discovery. *Data Min. Knowl. Discov.*, 25(2):208–242, 2012.

[VVA+12] Alexander Vergara, Shankar Vembu, Tuba Ayhan, Margaret A Ryan, Margie L Homer, and Ramón Huerta. Chemical gas sensor drift compensation using classifier ensembles. *Sensors and Actuators B: Chemical*, 166:320–329, 2012.

[VvRBT13] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luís Torgo. OpenML: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013.

[WAF16] Michael Wainberg, Babak Alipanahi, and Brendan J. Frey. Are random forests truly the best classifiers? *J. Mach. Learn. Res.*, 17:110:1–110:5, 2016.

[WBS+92] William J Welch, Robert J Buck, Jerome Sacks, Henry P Wynn, Toby J Mitchell, and Max D Morris. Screening, predicting, and computer experiments. *Technometrics*, 34(1):15–25, 1992.

[Web01] Geoffrey I. Webb. Discovering associations with numeric variables. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, San Francisco, CA, USA, August 26-29, 2001*, pages 383–388, 2001.

[WHG+06] Brian Williams, Dave Higdon, Jim Gattiker, Leslie Moore, Michael McKay, Sallie Keller-McNulty, et al. Combining experimental data and computer simulations, with an application to flyer plate experiments. *Bayesian Analysis*, 1(4):765–792, 2006.

[WLK13] Warren E. Walker, Robert J. Lempert, and Jan H. Kwakkel. *Deep Uncertainty*, pages 395–402. Springer US, Boston, MA, 2013.

[Wro97] Stefan Wrobel. An algorithm for multi-relational discovery of subgroups. In *Principles of Data Mining and Knowledge Discovery, First European Symposium, PKDD '97, Trondheim, Norway, June 24-27, 1997, Proceedings*, pages 78–87, 1997.

[WS06] G Gary Wang and Songqing Shan. Review of metamodeling techniques in support of engineering design optimization. In *ASME 2006 international design engineering technical conferences and computers and information in engineering conference*, pages 415–426. American Society of Mechanical Engineers Digital Collection, 2006.

[Wu14] Ying Nian Wu. *Data Augmentation*, pages 165–166. Springer US, Boston, MA, 2014.

[WY15] William Yang Wang and Diyi Yang. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2557–2563, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[XHLL19] Qizhe Xie, Eduard H. Hovy, Minh-Thang Luong, and Quoc V. Le. Self-training with noisy student improves ImageNet classification. *CoRR*, abs/1911.04252, 2019.

[YL09] I-Cheng Yeh and Che-hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Syst. Appl.*, 36(2):2473–2480, 2009.

[YN89] Taketoshi Yoshida and Shinichi Nakasuka. A dynamic scheduling for flexible manufacturing systems: hierarchical control and dispatching by heuristics. In *Proceedings of the 28th IEEE Conference on Decision and Control,*, pages 846–852. IEEE, 1989.

[Zhu05] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

[ZJC03] Zhi-Hua Zhou, Yuan Jiang, and Shifu Chen. Extracting symbolic rules from trained neural network ensembles. *AI Commun.*, 16(1):3–15, 2003.

[ZL05] Zhi-Hua Zhou and Ming Li. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Trans. Knowl. Data Eng.*, 17(11):1529–1541, 2005.

[ZTT16] Maciej Zieba, Sebastian K. Tomczak, and Jakub M. Tomczak. Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction. *Expert Syst. Appl.*, 58:93–101, 2016.

[ZWD16] Guoqiang Zhong, Lina Wang, and Junyu Dong. An overview on data representation learning: From traditional feature learning to recent deep learning. *CoRR*, abs/1611.08331, 2016.