KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Semi-Supervised and Explainable Machine Learning with an Application to the Low-Voltage Grid

Jonas Soenen

Supervisor:
Prof. dr. ir. H. Blockeel

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Computer Science

August 2023

Semi-Supervised and Explainable Machine Learning with an Application to the Low-Voltage Grid

Jonas SOENEN

Examination committee:
Prof. dr. ir. D. Vandermeulen, chair
Prof. dr. ir. H. Blockeel, supervisor
Prof. dr. ir. G. Janssens
Prof. dr. ing. P. Karsmakers
Dr. K. Vanthournout
(EnergyVille, VITO)
Prof. dr. S. Džeroski
(Institut Jožef Stefan, Slovenia)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Computer Science

August 2023

© 2023 KU Leuven – Faculty of Engineering Science
Uitgegeven in eigen beheer, Jonas Soenen, Celestijnenlaan 200A box 2402, B-3001 Leuven (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm, electronic or any other means without written permission from the publisher.

Acknowledgments

Although pursuing a PhD may appear to be a solitary journey, I wouldn't have achieved it without the support of numerous individuals.

To begin with, I'd like to express sincere gratitude towards my promotor Hendrik. Thanks to him, I had the privilege of embarking on this PhD journey and received guidance, feedback, and advice that proved instrumental in successfully completing it. Early in my PhD, I realized that a promotor is somewhat like a quirky machine learning algorithm. As the saying goes, "Garbage in = garbage out". To make the most of your promotor, you need to feed them the right data, context, and questions. Simply walking into a meeting and dumping what is on your mind doesn't work. (Believe me, I have tried.) However, once I cracked that code, Hendrik turned into an invaluable resource, always ready to assist whenever I encountered a challenge or hit a roadblock. While Hendrik's unwavering commitment to precise written communication could be a tad demanding, in hindsight, I truly appreciate the emphasis he placed on it: my own writing and communication skills have greatly improved because of it.

I extend my sincere gratitude to the members of my Jury for dedicating their time, sharing their expertise and conducting a thorough evaluation of my thesis. Their valuable insights and feedback have helped to shape the final version of this manuscript. Special appreciation goes to Gerda Janssens and Peter Karsmakers, whose discussions over the last four years contributed to the refinement of my research. Koen Vanthournout's unwavering commitment in managing the collaboration between KU Leuven and EnergyVille played a crucial role in enabling all the research in Chapter 4. Sašo Džeroski not only evaluated the manuscript but also sparked several interesting discussions during the preliminary defense. Finally, I would like to express my thanks to Dirk Vandermeulen for chairing the defenses.

I feel incredibly fortunate to have done my PhD in the company of exceptional colleagues. My heartfelt thanks go to everyone at the DTAI research group.

Your contributions, whether through light-hearted lunch conversations, serious work-related discussion, or shared grievances about “Reviewer 2” have all played a part in creating a positive and supportive atmosphere. I want to extend special thanks to a few remarkable individuals. Sebastijan and Toon, my master thesis advisors, which unknowingly set me on a path that brought me to where I stand today. Elia, Vincent, Lorenzo, Wannes, and Jesse who have been invaluable collaborators in my work around anomaly detection. The two generations of office 03.184 for making that room a fun and productive environment, with Aras in particular for ensuring that "deze jongentjes fruit eten". Laurens for his advice on anything implementation and renovation related. Jessa and Robin for their expert advice and support on the more general aspects of 'promotor-management' and doing a PhD. My appreciation also extends to Stella, Karen, Karin, and the rest of the secretariat. Your support in organizing and ordering equipment has significantly eased my role as a system group representative.

A heartfelt thank you to the entire team behind the low voltage grid use case. Thijs, your subtle (and not-so-subtle) pushes and guidance have accelerated my progress significantly. I'm also grateful to all other researchers involved in the use case for the engaging monthly meetings and the enjoyable days in Genk. In particular, Lola's support helped me stay motivated throughout the final phases of my PhD.

I would also like to the Flemish Government for funding my work under the “Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen” programme and the Flemish DSO Fluvius for supporting the research in Chapter 4.

I wish to thank my friends and family for the support during my PhD. They provided the necessary encouragement but also shared countless dinners, weekends and engaging activities that allowed me to recharge my batteries. First and foremost, I extend my deepest gratitude to my parents, Stefaan and Ine. Your relentless effort and continuous support have been the cornerstone of my personal growth, shaping me into the person I am today. I would like to thank my siblings, Jakob, Ruth and Laure for the joyful moments of goofing around at family gatherings. My sincere appreciation goes out to the mythical pink fluffy unicorns whose companionship has brightened up many not-always-pasta evenings. Additionally, I'd like to thank my high school friends for the epic and enduring RISK (and variants) games (at least until someone's attention span ran out, and that person decided ruin the game). You've all been an essential part of my life and I'm grateful for the memories we've created together.

Finally, Charlotte, your presence in my life has been an incredible source of joy, love, and laughter. Thank you for your love and being by my side through all the ups and downs. You bring out the best in me.

Abstract

Due to the enormous increase in available data and the computational power needed to process it, interest in the use of machine learning (ML) keeps increasing as it is able to extract valuable insights from this data. In an effort to make it easier to use ML in practice, this thesis proposes several novel ML techniques driven by two industry needs: (1) to make machine learning systems understand what the user wants and (2) to make the user understand what machine learning systems say or predict. Moreover, we also demonstrate the value of ML on a concrete contemporary use case: the reinforcement of the low-voltage grid.

In standard unsupervised clustering, it is difficult to tell the algorithm what the user expects. There are often many possible clusterings for the same set of examples. Therefore, the user has to experiment with different algorithms, similarity metrics and hyperparameter configurations to obtain the desired clustering. Active semi-supervised clustering offers an alternative: the algorithm gathers constraints by asking the user questions and uses those constraints to produce a clustering that matches the user's expectation. However, handling noise remains a challenge: if some constraints are incorrect, existing algorithms often fail to produce a satisfactory result.

To address this limitation, our first contribution is a framework that makes most constraint-based clustering algorithm robust to noise. By reasoning about noise probabilistically, the framework is able to detect and correct the noisy constraints such that the clustering algorithm can produce a high-quality clustering even if some of the constraints provided by the user are incorrect.

Next to clustering, we also looked into anomaly detection (AD). Many AD algorithms only identify anomalies, they do not explain *why* they label a particular instance as anomalous. This makes it difficult to take action without further analysis: maybe the identified anomaly exhibits rare but completely normal behavior, e.g., maintenance works. If an anomaly detector can explain its predictions, it is immediately clear why the algorithm thinks a particular

instance is anomalous. These explanations enable users to verify the algorithm's reasoning and make it easier to act upon the results.

As our second contribution, we provide an extensive overview of dependency-based AD, an anomaly detection strategy that is explainable. The main idea is to model the dependencies between attributes and mark instances that violate these dependencies as anomalous. These dependencies and violations can serve directly as explanations: e.g., "Animals with a tail usually have a backbone; however, a scorpion has a tail but no backbone". We compare existing dependency-based detectors and highlight the strengths of every method. Moreover, we identify two weaknesses of existing models and propose two novel ideas to solve them. Although further research is still required, we believe the presented insights are very relevant for future developments in the field of dependency-based AD.

Besides, the AD literature does not agree on *how* to set the hyperparameters of different anomaly detectors when experimentally comparing algorithms. Most papers compare either performance using "default" settings, or maximal performance under optimal settings. These methodologies are either too pessimistic, assuming no tuning, or unrealistically optimistic, assuming optimal tuning; and often result in methodologically unsound and irreproducible comparisons between algorithms. As our third contribution, we therefore propose to use a small validation set to tune an anomaly detector's hyperparameters to achieve a realistic, fair, sound, and reproducible comparison.

Finally, in our last contribution, we illustrate the value of ML by applying it to help plan a cost-efficient reinforcement of the low voltage grid (LVG). To plan such a reinforcement, one needs to calculate the typical currents and voltages throughout the grid. This calculation requires the electricity consumption time series of each connected consumer as an input. However, for many consumers, these time series are unknown and need to be estimated from the available information. Therefore, we developed two methods to predict the electricity consumption time series of a particular household given consumer, weather and calendar information. The first method solves the task but is complex and slow with limited interpretability. Therefore, we also developed a second method that uses predictive clustering. This second model is significantly simpler with similar predictive performance as our first approach. Moreover, it is fast and interpretable, allowing a domain expert to inspect which time series are predicted in which circumstances. We hope our contribution will help to reduce the cost of the reinforcement of the LVG while also giving distribution system operators more insight into their data.

Beknopte samenvatting

Vanwege de enorme toename van beschikbare data en rekenkracht nodig om deze te verwerken, blijft de interesse in machinaal leren (ML) toenemen. Dit omdat ML waardevolle inzichten uit deze data kan halen. Deze thesis introduceert nieuwe ML-technieken die voortkomen uit twee industriële behoeften: (1) om ML-systemen rekening te laten houden met gebruikersbehoeften en (2) om de gebruiker de voorspellingen van het systeem te doen begrijpen. Bovendien tonen we ook de waarde van ML aan met een hedendaagse toepassing: het versterken van het laagspanningsnetwerk.

In standaard ongesuperviseerd clusteren is het moeilijk om rekening te houden met gebruikersverwachtingen. Er zijn vaak meerdere clusteringsmogelijkheden voor dezelfde set instanties waardoor experimentatie met verschillende algoritmes en instellingen vereist is om de gewenste clustering te bekomen. Actief semi-gesuperviseerd clusteren biedt een alternatief door vragen te stellen aan de gebruiker om zo constraints (beperkingen) te verzamelen. Deze constraints worden dan gebruikt om een clustering in lijn met de gebruikersverwachtingen te produceren. Het omgaan met ruis blijft echter een uitdaging: als sommige constraints onjuist zijn, falen bestaande algoritmes om een goed resultaat te produceren.

Als eerste bijdrage introduceren we een raamwerk dat actief semi-gesuperviseerde algoritmes robuust maakt tegen ruis. Door probabilistisch over ruis te redeneren kan het raamwerk incorrecte beperkingen detecteren en corrigeren. Op die manier kan het algoritme een goede clustering produceren zelfs als sommige van de gegeven beperkingen onjuist zijn.

Naast clusteren hebben we tevens anomaliedetectie (AD) onderzocht. Veel AD-algoritmes identificeren anomalieën maar leggen niet uit *waarom* ze een bepaald geval als abnormaal bestempelen. Dit maakt het moeilijk om actie te ondernemen zonder verdere analyse. Als een anomaliedetector zijn voorspellingen uitlegt is meteen duidelijk waarom het algoritme een bepaald geval als abnormaal

beschouwt. Deze verklaringen laten gebruikers toe de redenering van het algoritme te verifiëren en maken het makkelijker om, op basis van de resultaten, actie te ondernemen.

Als tweede bijdrage geven we daarom een uitgebreid overzicht van dependency-based (op-afhankelijkheden-gebaseerde) AD: een strategie voor AD die uitlegbaar is. Deze methoden modelleren de afhankelijkheden tussen attributen en markeren instanties, die deze afhankelijkheden schenden, als abnormaal. De afhankelijkheden en schendingen kunnen rechtstreeks dienen als verklaringen. Bijvoorbeeld, "Dieren met een staart hebben meestal een ruggengraat; echter, een schorpioen heeft een staart maar geen ruggengraat". Bovendien identificeren we twee zwakke punten van bestaande modellen en stellen we twee nieuwe ideeën voor om deze op te lossen. Hoewel verder onderzoek nodig is, geloven we dat de gepresenteerde inzichten zeer relevant zijn voor toekomstige ontwikkelingen op het gebied van dependency-based AD.

Daarnaast is er in de AD literatuur geen overeenstemming over *hoe* de hyperparameters van verschillende anomaliedetectoren moeten worden afgestemd bij het experimenteel vergelijken van algoritmes. De meeste papers vergelijken ofwel prestaties met "standaard" instellingen, ofwel maximale prestaties onder optimale instellingen. Deze methodologieën zijn ofwel te pessimistisch, waarbij wordt uitgegaan van geen enkele afstemming van de hyperparameters, of onrealistisch optimistisch, waarbij wordt uitgegaan van optimale afstemming. Deze methodologieën leiden daarnaast vaak tot methodologisch onjuiste en niet-reproduceerbare vergelijkingen. Daarom stellen we als derde bijdrage voor om een kleine validatieset te gebruiken om de hyperparameters af te stemmen voor een realistische, eerlijke, betrouwbare en reproduceerbare vergelijking.

Als laatste bijdrage laten we de waarde van ML zien door het te gebruiken om een kostenefficiënte versterking van het laagspanningsnet te ondersteunen. Het plannen van zo'n versterking vereist het berekenen van de typische stromen en spanningen in het hele net. Hiervoor zijn tijdsreeksen van het elektriciteitsverbruik van elke aangesloten gebruiker nodig. Aangezien deze tijdsreeksen vaak ontbreken, hebben we twee methoden ontwikkeld om ze te voorspellen op basis van consumentendata, weergegevens en kalanderinformatie. Onze eerste methode lost de taak op, maar is complex en traag met beperkte interpreteerbaarheid. Daarom hebben we een tweede methode ontwikkeld die predictive clustering (voorspellend clusteren) gebruikt. Dit tweede model is eenvoudiger met vergelijkbare voorspellende accuraatheid en is snel en interpreteerbaar. Zo kan een expert bekijken welke tijdsreeksen voorspeld worden in welke omstandigheden. We hopen dat onze bijdrage de kosten van versterking van het laagspanningsnet verlaagt en netbeheerders meer datagedreven inzichten bezorgt.

Contents

Abstract iii

Beknopte samenvatting v

List of Abbreviations vii

List of Symbols vii

Contents vii

List of Figures xi

List of Tables xiii

1 Introduction 1
- 1.1 Contribution 1: Making constraint-based clustering algorithms robust to user mistakes . . . 3
- 1.2 Contribution 2: Clustering and sampling electricity consumption time series . . . 4
- 1.3 Contribution 3: Proper evaluation of anomaly detection methods 6
- 1.4 Contribution 4: Explainable anomaly detection through dependency-based AD . . . 6
- 1.5 Contributions not discussed in this dissertation . . . 8

2 Background 11
- 2.1 Machine learning . . . 11
  - 2.1.1 Reinforcement learning . . . 11
  - 2.1.2 Supervised learning . . . 12
  - 2.1.3 Unsupervised learning . . . 12
  - 2.1.4 Semi-supervised learning . . . 13
  - 2.1.5 Inductive and transductive learning . . . 14

2.2 Clustering . . . 15
2.2.1 Distance measures . . . 16
2.2.2 Clustering approaches . . . 17
2.2.3 Semi-supervised clustering . . . 19
2.2.4 Evaluation of clustering algorithms . . . 20
2.3 Anomaly detection . . . 21
2.3.1 Anomaly detection approaches . . . 22
2.3.2 Evaluation of anomaly detection . . . 25

3 Tackling Noise in Active Semi-Supervised Clustering 27
3.1 Introduction . . . 28
3.2 Related work . . . 29
3.3 Reasoning about noisy constraints . . . 31
3.3.1 Terminology and background . . . 31
3.3.2 Overview . . . 32
3.3.3 Defining the confidence of a constraint set . . . 33
3.3.4 Approximating the confidence of a constraint set . . . 34
3.3.5 Selecting redundant queries . . . 38
3.4 Integration in a state-of-the-art active semi-supervised clustering algorithm . . . 39
3.5 Experiments . . . 41
3.5.1 Algorithms . . . 41
3.5.2 Experimental methodology . . . 42
3.5.3 Datasets . . . 42
3.5.4 Results . . . 43
3.5.5 Evaluation of noise detection . . . 43
3.5.6 Runtime . . . 44
3.6 Conclusion . . . 46

4 Scenario generation of residential electricity consumption through sampling of historical data 47
4.1 Introduction . . . 48
4.2 Related work . . . 50
4.2.1 Electrical load forecasting . . . 51
4.2.2 Residential load modeling . . . 51
4.2.3 Generative models . . . 52
4.2.4 Summary . . . 53
4.3 The expert-based and data-driven approach . . . 53
4.3.1 Problem statement and notation . . . 53
4.3.2 Methodology . . . 54
4.3.3 The expert-based technique . . . 54
4.3.4 The data-driven technique . . . 58
4.4 The predictive clustering approach . . . 63

4.4.1 Problem Statement and notation . . . 63
4.4.2 Generating scenarios with predictive clustering . . . 64
4.5 Experimental methodology . . . 67
4.5.1 Datasets . . . 67
4.5.2 Cross validation . . . 70
4.5.3 Performance metric . . . 72
4.5.4 Scenario Generation Methods . . . 72
4.6 Experimental results . . . 73
4.6.1 Scenario generation performance . . . 73
4.6.2 Influence of dataset size . . . 76
4.6.3 Interpretability . . . 78
4.7 Conclusion . . . 80

5 Evaluating unsupervised anomaly detection algorithms 83
5.1 Introduction . . . 83
5.2 Unsupervised Anomaly detection . . . 85
5.3 Current Methodologies . . . 85
5.4 A strategy based on validation sets . . . 87
5.4.1 Validation set size V . . . 88
5.5 Experiments . . . 90
5.5.1 Q1: Does the methodology for selecting hyperparameters affect the ranking of a benchmarking study? . . . 94
5.5.2 Q2: How does the methodology for selecting hyperparameters affect the performance of an algorithm? . . . 95
5.5.3 Q3: Is a small labeled validation set sufficient for identifying a good set of hyperparameters? . . . 97
5.5.4 Q4: Is an algorithm's performance on the validation set always representative of its performance on the test set? 98
5.6 Conclusion . . . 99

6 Modeling non-functional dependencies in dependency-based anomaly detection 101
6.1 Introduction . . . 102
6.2 Current state of dependency-based anomaly detection . . . 103
6.2.1 Advantages of dependency-based anomaly detection . . . 104
6.2.2 Scoring mechanisms . . . 105
6.2.3 Handling anomalies in the training set: semi- and unsupervised dependency-based anomaly detection . . . 113
6.2.4 Summary . . . 115
6.3 Modeling non-functional dependencies . . . 116
6.3.1 Modeling non-functional dependencies with density trees 116
6.4 Leveraging patterns in the anomalies . . . 117
6.5 Conclusion . . . 120

7 Conclusion **121**
7.1 Summary and contributions 121
7.2 Future work 123

Bibliography **127**

Curriculum Vitae **145**

List of Publications **147**

List of Figures

2.1 Illustration multiple possible clusterings for the same dataset . 13
2.2 Illustration of supervised inductive learning 14
2.3 Illustration of supervised transductive learning 15
2.4 Illustration of KNN and LOF 23
2.5 Illustration of isolation forest 25

3.1 Illustration semi-supervised clustering concepts 28
3.2 Example: enumerating consistent constraint sets 36
3.3 Illustration of the marginalization over the irrelevant constraints in nCOBRAS 39
3.4 Average ARI of different clustering algorithms with varying amounts of noise 43
3.5 Average aligned rank of different clustering algorithms with varying amounts of noise 44
3.6 Runtime comparison of different clustering algorithms 44

4.1 Effect of averaging on load time series 49
4.2 Visual representation of the electricity consumption time series and associated attributes 55
4.3 Illustration of the expert-based technique 56
4.4 Illustration of the data-driven technique 59
4.5 Scenario generation energy score comparison 74
4.6 Influence of training set size on the energy score 76
4.7 Influence of dataset size on execution times 77
4.8 PCT learned on Flanders dataset 79
4.9 Influence of sunHour on consumers with solar panels 80
4.10 Influence of temperature on consumers with high yearly consumption 80

5.1 AD performance according to different hyperparameter selection methodologies 95
5.2 Tuned performance for different validation set sizes 95
5.3 AUC on the validation and test set 98

6.1 CFA scoring for irrelevant target attribute 107
6.2 CFA scoring for non-functional dependency 107
6.3 ALSO scoring for non-functional dependency 109
6.4 Illustration of FRaC scoring on a non-functional dependency . 112
6.5 Illustration of FRaC scoring failing to model a non-functional dependency 113
6.6 Illustration of ALSO's cross validation to avoid modeling anomalies 114
6.7 Illustration of the difference between density and residual-based scoring 118
6.8 Illustration of density-based scoring for a non-functional dependency 118

List of Tables

3.1 Precision and recall of noise correction in nCOBRAS 45

4.1 Dataset characteristics 68
4.2 Available consumer attributes in the Ireland dataset 70
4.3 Available consumer attributes in the London dataset 71
4.4 Scenario generation algorithm runtimes 75

5.1 Hyperparameter grid 91
5.2 Anomaly detection dataset characteristics 92
5.3 Anomaly detection performance of several AD-algorithms based on different hyperparameter selection methodologies 94

Chapter 1

Introduction

The 21st century is characterized by an enormous increase in the amount of available data. Having this data available raises the question: "What can we learn from this data?". Manually inspecting the data becomes very difficult as soon as the dataset becomes too big to be visualized. Fortunately, there has also been an enormous rise in computational power which enables computers to do more complex analysis on these datasets (semi-) automatically.

These trends have spurred the development of machine learning, the discipline of extracting patterns from data. Machine learning systems are able to improve their performance through experience, e.g., by inspecting a set of examples from a certain process. The behavior of such a system is thus not programmed upfront, but rather *learned* from the provided examples.

The Flemish government is aware of the potential of these new technologies and has the ambition to make Flanders a leader in rolling out machine learning and - more broadly - artificial intelligence (AI). To realize this ambition, the Flemish government launched the Flemish AI plan in 2019. The plan has three main goals: (1) stimulating the use of AI in companies, (2) raising awareness and educating the general public, and (3) strengthening the AI research already present in Flanders. To strengthen the AI research in Flanders, the Flemish government decided to fund the Flanders AI research plan which focuses on AI research driven by the needs of companies, organizations, government and citizens; but part of the plan is also dedicated to demonstrating the value of AI on concrete use cases in health, industry and society.

This dissertation was funded by the Flanders AI research plan and therefore, in line with the plan's goals, roughly contains two types of contributions. On

the one hand, we developed novel methods driven by the industry's need for explainable models and algorithms that can leverage domain expert feedback. On the other hand, we demonstrate the value of AI by contributing to the cost-efficient reinforcement of the low-voltage grid using machine learning methods.

The first theme throughout my PhD is the use of constraints to include domain expert feedback and expertise into otherwise completely unsupervised algorithms. By specifying such constraints, the domain expert can guide the algorithm towards a good solution. Alternatively, the algorithm can take the initiative and ask the expert questions with the goal of eliciting constraints that are maximally informative; this is called active learning. This makes these otherwise unsupervised algorithms significantly easier to apply in practice: instead of trying out multiple algorithms and tweaking the configuration of those algorithms, the expert just provides some constraints or answers some questions posed by the algorithm.

The second theme is that of explainability; many machine learning systems just produce a solutions or predictions without explaining how they arrive at this conclusion. This makes it more difficult to verify the systems predictions and build trust in the system. When a system can explain why it is producing a certain result, this explanation can be checked to make sure the system's reasoning is correct. Moreover, these explanations can also help to understand patterns in the data and can therefore generate insight.

In the next sections, we summarize the four main contributions of my PhD.

- In our first contribution, we investigate the influence of incorrect or *noisy* constraints on constraint-based clustering and propose a framework to make existing approaches robust to noise (Section 1.1).
- In our second contribution, we describe how we applied ML techniques to help enable a cost-efficient reinforcement of the low voltage grid (Section 1.2).
- In our third contribution, we critique the current methodologies for comparing the performance of different anomaly detection algorithms and propose an alternative methodology that results in a realistic, fair and reproduceable comparison (Section 1.3).
- In our fourth contribution, we review the existing literature on dependency-based anomaly detection: a strategy to detecting anomalies that is explainable. We highlight two drawbacks of current systems and propose ideas that might help to alleviate these issues in the future (Section 1.4).

Finally, we briefly describe some of the other contributions not further discussed in this dissertation (Section 1.5).

1.1 Contribution 1: Making constraint-based clustering algorithms robust to user mistakes

Many machine learning systems are fully supervised meaning that from a labeled set of examples they learn a mapping from the example itself to its label. If there are no labels available, there are unsupervised algorithms which try to extract useful insights from a dataset of examples without any label information. Clustering is such an unsupervised learning task where the goal is to discover a natural grouping of the examples. More specifically, given a set of examples or *instances*, the goal is to partition these instances into clusters (i.e., groups) such that instances within the same cluster are more similar to one another than instances from different clusters.

However, for most unsupervised learning tasks there is a certain degree of subjectivity involved. For a single dataset, multiple valid solutions might exist, and different users will prefer a different solution for their specific application. In a supervised learning setting, this is not a problem; the labels themselves contain the expected output and therefore guide the algorithm towards the desired solution. When no labels are available, no such mechanism exists and a practitioner will have fiddle with the hyperparameters of the algorithm or try different algorithms until the desired output is obtained. This often results in a trail and error procedure which requires in depth knowledge about the available algorithms and their hyperparameters.

Active semi-supervised clustering offers a solution by integrating the user in the clustering process such that the user can guide the algorithm towards a good clustering. During the execution of the clustering algorithm, the algorithm will ask the user questions of the form: "Should instances x and y be in the same cluster?". Answering such a query results in either a must-link constraint between x and y, i.e., x and y must be in the same cluster, or a cannot-link constraint, i.e., x and y must be in different clusters. By taking these constraints into account, the clustering algorithm is guided towards a good clustering. Because the algorithm can decide which queries are asked, it can ask the most informative queries first to ensure a high quality clustering while limiting the number of queries the domain expert has to answer.

Unfortunately, these active semi-supervised clustering algorithms make the assumption that all constraints given by the user are correct. Therefore, if the

user makes a mistake (i.e., provides an incorrect or *noisy* constraint), this will have a significant negative impact on the clustering quality.

To solve this drawback, we developed a framework that can detect and correct these noisy constraints. We achieve this by asking redundant queries and reasoning probabilistically about mistakes in the set of all constraints. This framework can be used to make most constraint-based clustering algorithms robust to noise.

This contribution was published in a peer-reviewed conference [125] and is discussed in Chapter 3.

J. Soenen, S. Dumančić, T. Van Craenendonck, and H. Blockeel. "Tackling Noise in Active Semi-supervised Clustering". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by F. Hutter, K. Kersting, J. Lijffijt, and I. Valera. Cham: Springer International Publishing, 2021, pp. 121–136. ISBN: 978-3-030-67661-2

1.2 Contribution 2: Clustering and sampling electricity consumption time series

Within the Flanders AI research use cases, we started a collaboration with EnergyVille[1] to illustrate the value of active constraint-based clustering by applying it on a real world industrial use case. The initial goal set by EnergyVille was to cluster electricity consumption time series of consumers connected to the low voltage grid such that consumers with similar impact on the grid occurred in the same cluster. Instead of trying to precisely define what 'impact on the grid' means and embedding that knowledge in a custom hand-crafted similarity metric to use for clustering, we planned to apply active semi-supervised clustering such that the domain experts from EnergyVille could guide the clustering towards their expected clustering.

However, later it became apparent that the end goal was not the clustering itself but to sample time series from these clusters for households where no electricity consumption measurements are available. More specifically, the end goal of EnergyVille is to calculate the currents and voltages in the low-voltage grid to support a cost-efficient reinforcement of the low-voltage grid. To achieve

[1] EnergyVille is a collaboration between Belgian research partners KU Leuven, VITO, imec and UHasselt in the fields of sustainable energy and intelligent energy systems. `https://www.energyville.be/en`

this, they have developed a digital twin of the low-voltage grid that given the layout of the cables and the electricity consumption profiles of all the connected consumers, simulates how electricity flows through the grid. However, for most consumers, the electricity consumption time series are unavailable.

To solve this, we developed two methods to generate electricity consumption time series for these unmeasured households by leveraging the limited attributes that are known for the household. Both methods cluster the available electricity consumption time series and then use the available attributes from an unmeasured consumer to sample time series from the most relevant clusters.

The first method clustered the time series first, and only afterwards, learned a second model to associate the clusters with the available attributes. The clustering was therefore completely independent of the attributes which causes the second model that associates clusters with attributes to become complex and uninterpretable.

We later improved upon this first method by constraining the clustering itself such that each cluster has a description in attribute space. This results in a fast and simple model to associate the attributes with the clusters that is fully interpretable.

The first method is published in a peer reviewed journal [126], while we are preparing a publication on the second method. Both methods are explained in Chapter 4.

J. Soenen, A. Yurtman, T. Becker, R. D'hulst, K. Vanthournout, W. Meert, and H. Blockeel. "Scenario generation of residential electricity consumption through sampling of historical data". In: *Sustainable Energy, Grids and Networks* 34 (2023), p. 100985. ISSN: 2352-4677. DOI: https://doi.org/10.1016/j.segan.2022.100985. URL: https://www.sciencedirect.com/science/article/pii/S2352467722002302

J. Soenen, A. Yurtman, T. Becker, K. Vanthournout, and H. Blockeel. "Generating electricity consumption scenarios with predictive clustering trees". In: *In preparation* (2023)

1.3 Contribution 3: Proper evaluation of anomaly detection methods

After clustering, we started looking into anomaly detection, another task that is commonly solved in an unsupervised way. The goal of anomaly detection is to detect instances that are suspicious because they don't conform to the 'normal' behavior that the other instances in the dataset usually exhibit.

Researchers have proposed a large variety of anomaly detection algorithms and their performance is greatly affected by how a user sets each algorithm's hyperparameters. However, the anomaly detection literature does not agree on how to set these hyperparameters when experimentally comparing different algorithms. Most papers compare either performance using "default" settings, or maximal performance under optimal settings. We argue that both strategies fail to capture what practitioners are actually interested in: how well does the algorithm perform in practice? Current methodologies are either too pessimistic, assuming no tuning, or unrealistically optimistic, assuming optimal tuning. Morover, these methodologies often result in methodologically unsound and irreproducible comparisons between algorithms. We therefore propose to use a small validation set to tune an anomaly detector's hyperparameters on a per dataset basis and argue that this methodology results in a realistic, fair and sound comparison between anomaly detectors.

This contribution has been published in a peer-reviewed workshop [128] and is explained in Chapter 5.

J. Soenen*, E. Van Wolputte*, L. Perini, V. Vercruyssen, W. Meert, J. Davis, and H. Blockeel. "The effect of hyperparameter tuning on the comparative evaluation of unsupervised anomaly detection methods". In: *Proceedings of the KDD'21 Workshop on Outlier Detection and Description*. 2021, pp. 1–9

1.4 Contribution 4: Explainable anomaly detection through dependency-based AD

Many of the current anomaly detectors cannot explain why they mark a certain instance as anomalous. The criteria used to detect anomalous behavior include

*Both authors contributed equally to this work

distance to the k'th nearest neighbor [107], lying in a low density region [18], or being easy to isolate from other instances with random splits [81]. While these are all proven to be effective criteria for detecting anomalies, it is difficult for a domain expert to understand why an instance is marked as anomalous based on these criteria. This makes it difficult to take action based on the detected anomalies.

Together with Elia van Wolputte, we developed AD-Mercs an unsupervised anomaly detector that is able to explain why a certain instance is suspicious. AD-Mercs uses an ensemble of decision trees to model the dependencies that exist between attributes in a dataset. An instance is then marked as anomalous if it violates a dependency or if it adheres to the same dependency as other anomalies. AD-Mercs therefore reasons about two types of evidence: deviating from normal behavior or adhering to anomalous behavior.

In terms of anomaly detection performance, AD-Mercs is competitive with other anomaly detectors on standard dataset, but only consistently better in specific (synthetic) circumstances. These lackluster results prompted us to take a step back and thoroughly reevaluate the ideas behind AD-Mercs with respect to other anomaly detectors that use dependency violation as a criterion for anomalousness.

As such, this dissertation does not discuss AD-Mercs in detail but discusses existing dependency-based anomaly detectors, highlights two drawbacks of these existing methods and introduces the two ideas in AD-Mercs that we developed to alleviate these drawbacks. We believe that the presented ideas and insights will be very relevant for future developments in the field of dependency-based anomaly detection.

This contribution is based on our paper about AD-Mercs which has not yet been published in a peer-reviewed venue but is available on arXiv. We explain this contribution in Chapter 6.

J. Soenen*, E. V. Wolputte*, V. Vercruyssen, W. Meert, and H. Blockeel. *AD-MERCS: Modeling Normality and Abnormality in Unsupervised Anomaly Detection*. 2023. arXiv: `2305.12958 [cs.LG]`

*Both authors contributed equally to this work

1.5 Contributions not discussed in this dissertation

Next to these four contributions, I also collaborated with other researchers on several topics related to my own work. These collaborations resulting in three papers published at different peer-reviewed venues.

Explaining clusters with constraints

In collaboration with Simiao Lin, we made COBRAS, a state-of-the-art active semi-supervised clustering algorithm, more explainable by leveraging the constraints to build explanations about why two instances are in the same or different clusters.

This work is published at a peer-reviewed workshop:

S. Lin, A. Yurtman, J. Soenen, and H. Blockeel. "LinC: Explaining Time Series Clusterings with User-Provided Constraints". In: *Proceedings of the ECML'23 workshop on Explainable AI for Time Series*. 2023

A scalable ensemble approach to electricity consumption forecasting

In collaboration with a team of researchers from DTAI and STADIUS (two research groups within KU Leuven), we designed a novel long-term electricity consumption forecasting method which finished third in the IEEE-CIS competition (and ranks first after leveraging insights from another team[2]). This contribution was published as a journal paper:

L. Botman, J. Soenen, K. Theodorakos, A. Yurtman, J. Bekker, K. Vanthournout, H. Blockeel, B. De Moor, and J. Lago. "A Scalable Ensemble Approach to Forecast the Electricity Consumption of Households". In: *IEEE Transactions on Smart Grid* 14.1 (2023), pp. 757–768. DOI: 10.1109/TSG.2022.3191399

[2]The smart meters used to collect the data were battery operated. So if the final known month of a particular household had zero consumption, the best prediction for the future was also zero (although, the household probably did consume electricity).

Adding support for missing value to dynamic time warping (DTW)

Our work on electricity consumption time series inspired research to handle missing data in time series when calculating the dissimilarity between time series with dynamic time warping. This contribution is published at a peer-reviewed conference:

A. Yurtman, J. Soenen, W. Meert, and H. Blockeel. "Estimating Dynamic Time Warping Distance Between Time Series with Missing Data". In: *Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*. 2023

Chapter 2

Background

In this chapter, we give a short introduction into machine learning, followed by an introduction into clustering and anomaly detection, the two domains in which most of the contributions of this dissertation are situated in.

2.1 Machine learning

Machine learning studies systems and algorithms that learn from experience. Depending on the way a machine learning system gathers or receives experience, the system is classified in a different *learning paradigm*. There are three main learning paradigms: learning through interaction with the environment (reinforcement learning, Section 2.1.1), learning from labeled data (supervised learning, Section 2.1.2), and learning from unlabeled data (unsupervised learning, Section 2.1.3). In this dissertation, we also discuss semi-supervised learning, a paradigm between unsupervised and supervised learning (Section 2.1.4). In semi-supervised learning supervision is available for some (but not all) examples in the dataset. Finally, we discuss the differences between inductive and transductive learning (Section 2.1.5).

2.1.1 Reinforcement learning

In reinforcement learning, an autonomous agent learns how it should act in an environment to achieve positive outcomes. It gathers experience through interaction with this environment: the agent takes actions and observes the

outcome. Sequences of actions that result in positive outcomes are *reinforced*, while sequences of actions with negative results are avoided in the future. The agent has to balance exploration, attempting to find new strategies to reach positive outcomes by selecting suboptimal actions, and exploitation, using known action sequences to get the positive outcomes that the agent already discovered.

2.1.2 Supervised learning

In supervised learning, the system learns from a set of labeled examples. The goal is to learn a mapping from the example itself to its label. As such, given a set of labeled examples, a supervised learning system will learn a model that can predict the label of new unseen examples. The learner will inspect the examples and look for patterns that hold, such that it can exploit those same patterns to make predictions for the unseen examples.

For example, you might want to predict whether a specific person will like a particular movie. As such, you first gather a dataset of labeled examples by asking that person to rate a large amount of movies. Given these labeled examples, you can learn a predictive model that predicts the rating of the movie given some information about the movie. Such a predictive model can then be used to predict the rating for a movie that the person has never seen before. This is exactly what streaming platforms like Netflix do, given the ratings you provided on previously watched movies, the recommendation system will try to predict which other movies you will like.

Note that for the same set of examples, there are different sets of labels that might be of interest. For a dataset of movies, you might be interested in predicting ratings for each movie while an online store might be interested in predicting the genre of the movie to automatically categorize movies in their online interface. By providing a particular set of labels, you tell the learning algorithm what target concept you are interested in.

The main drawback of supervised learning is that it requires a fully labeled dataset. In practice, a dataset might not have the label of interest available. In that case, a team of domain experts has to label a significant amount of examples before you are able to apply any supervised learning technique. This label gathering can be very labor-intensive and is therefore expensive.

2.1.3 Unsupervised learning

In unsupervised learning, there is no supervision, meaning that there is no label associated with each example. The learning algorithm tries to extract patterns



(a) Clustering based on facial expression

(b) Clustering based on person identity

Figure 2.1: For the same set of examples multiple valid clusterings exists. *Source of images: CMU face dataset* (`https://www.cs.cmu.edu/afs/cs/project/vision/vasc/idb/www/html/face/`)

that hold for this data or tries to find a model that appropriately describes the dataset. Because no labels are required, these techniques are often easier to apply. There are many unsupervised learning tasks including finding natural groups in the data (clustering), representing the data in a lower-dimensional space without loss of information (dimensionality reduction), finding reoccuring patterns in time series (motif discovery), detecting examples that do not conform to the usual behavior (anomaly detection), and many more.

One issue with unsupervised learning is that, for the same set of examples, multiple possible solutions might exists and no labels are available to guide the algorithm towards a particular solution. For example, in clustering, the same set of examples can often be clustered in multiple valid ways (Figure 2.1). However, for a specific application some of these clusterings might be better suited than others. In a completely unsupervised setting, the only ways to influence the result is to change the algorithm, the parameters of the algorithm or the dissimilarity metric that is used to quantify how dissimilar two examples are from one another. This often results in a trail and error procedure of changing the algorithm, the parameters or distance metric and inspecting the result to see if the clustering has improved. Not only can this be a very frustrating process, it is often very difficult to predict upfront how a change in algorithm, parameters, or distance metric will influence the clustering.

2.1.4 Semi-supervised learning

To make it easier to apply unsupervised algorithms in practice, there is a need to be able to guide the algorithms towards good solutions without having to

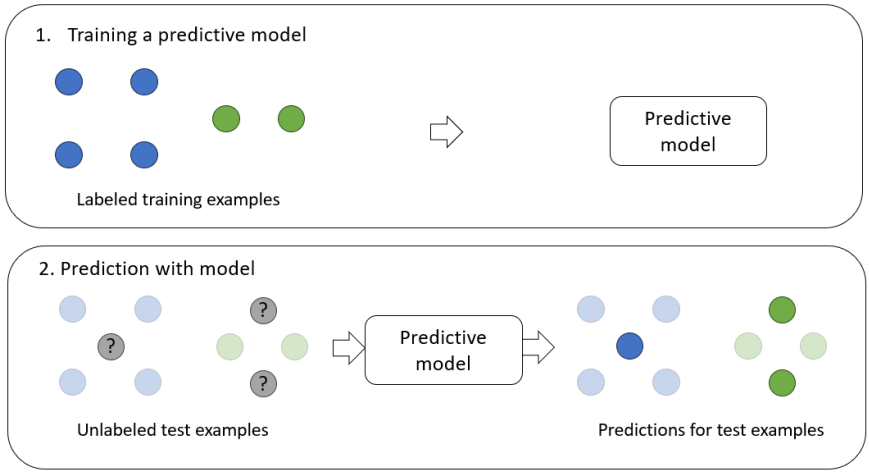


Figure 2.2: Simple illustration of supervised inductive learning: the labeled training examples are used to train a model and this model is then used to make predictions for unseen test examples.

tinker with technical details such as algorithm types, parameters and distance metrics.

This is exactly what semi-supervised learning aims to offer: a learning setting in between unsupervised and supervised learning where the machine learning model is presented with a set of unlabeled examples and some (limited) supervision to point the model towards good solutions. This supervision often takes the form of labels for some of the examples (e.g., semi-supervised anomaly detection [148], PU-learning for binary classification [7]) or pairwise constraints [11] for clustering.

2.1.5 Inductive and transductive learning

Supervised, semi-supervised and unsupervised approaches can be further classified into inductive and transductive learning.

Standard supervised learning is an example of inductive learning: a general model is induced from a set of examples and that model is used to make predictions for new instances (Figure 2.2). As such there is reasoning from specific cases to a more general model (induction). The general model can then

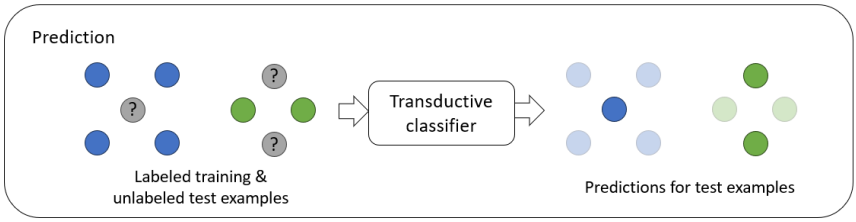


Figure 2.3: Simple illustration of supervised transductive learning: the labeled training examples and unlabeled test examples are both provided as input to the transductive classifier which outputs predictions for the test examples. No model is learned.

be used to reason about specific cases (deduction). Most supervised learning techniques fall under this category.

In transductive learning, no general model is constructed. The system reasons directly from specific training cases to specific test cases (transduction; Figure 2.3). In supervised learning, k-nearest neighbors is a well known example of transductive learning. No model is learned, the prediction of a test example in simply an aggregate over the labels of the k examples that are closest to the test example. In unsupervised learning, clustering is commonly done in a transductive manner: each example is just assigned to a cluster, no model is constructed.

2.2 Clustering

The goal of clustering is to divide a set of examples (or *instances* as they are often called in clustering) into groups or *clusters* such that instances within one cluster are more similar to one another than instances from different clusters. Clustering is used to discover the underlying structure of the data.

Formally, the standard formulation of clustering is as follows, given a dataset of instances $\mathcal{D}$, partition the instances x_i in $\mathcal{D}$ into clusters $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K$ such that clusters are non-overlapping

$$\mathcal{C}_i \cap \mathcal{C}_j = \emptyset \quad \forall i, j \text{ and } i \neq j \tag{2.1}$$

and every instance in $\mathcal{D}$ is part of at least one cluster

$$\bigcup_{i=1}^{K} \mathcal{C}_i = \mathcal{D}. \tag{2.2}$$

The number of clusters K is often a hyper parameter of the clustering algorithm and has to be set up front. This standard formulation is called hard partitional clustering.

Common variations of hard partitional clustering include hierarchical clustering [98], where a hierarchy of clusterings is constructed, and fuzzy clustering [157], where each instance can belong to multiple clusters with a degree of membership.

2.2.1 Distance measures

One important consideration when performing clustering is the notion of similarity that is used. Recall that we defined clustering as dividing instances into clusters such that instances within one cluster are more *similar* to one another than instances from different clusters. Before clustering, it is required to mathematically define what *similar* means. This is done by defining a dissimilarity or distance function d that takes two instance from the dataset as input and outputs a number that reflects how far apart or how dissimilar the two instances are:

$$d : \mathcal{D} \times \mathcal{D} \to \mathcal{R}. \tag{2.3}$$

The higher $d(x_i, x_j)$ the further apart they are considered. A dissimilarity function should be symmetric and positive:

$$d(x_i, x_j) = d(x_j, x_i) \qquad \forall i, j \qquad \text{(Symmetry)}$$

$$d(x_i, x_j) \geq 0 \qquad \forall i, j \qquad \text{(Positivity)}$$

If a dissimilarity function is also reflexive and satisfies the triangle inequality it is called a *distance metric*.

$$d(x_i, x_j) \leq d(x_i, x_k) + d(x_k, x_j) \qquad \forall i, j, k \qquad \text{(Triangle inequality)}$$

$$d(x_i, x_j) = 0 \Leftrightarrow x_i = x_j \qquad \forall i, j \qquad \text{(Reflexivity)}$$

In some cases, a similarity metric is used instead of a distance metric which should return a higher number if the two instances are more similar to one another. A similarity metric should also satisfy some properties similar to the ones defined above for a distance metric [155].

For continuous attribute-value data, the Euclidean distance is often used as a distance metric:

$$d_{\text{euclidean}}(x_i, x_j) = ||x_i - x_j||_2 \tag{2.4}$$

with $|| \cdot ||_2$ the euclidean vector norm. This distance metric gives each attribute equal importance. However, attributes with a bigger range of possible values will have a bigger influence on the distance than attributes with a small range. Therefore, the dataset is often normalized upfront to ensure that all attributes range from 0 to 1 or standardized to ensure that the mean and standard deviation of each attribute is the same.

However, the euclidean distance is by far not the only metric that can be used other possibilities include the cosine distance, mahalanobis distance, correlation, etc. The same is true for other data types such as time series, graphs and attribute-value data with nominal or ordinal attributes. In principle, any function that adheres to the properties above could be used. For a particular application one could even design a custom distance function that captures the application's notion of similarity perfectly.

2.2.2 Clustering approaches

There are many different algorithms and approaches to clustering, each algorithm with its own assumptions and bias [155]. Giving an overview of all different approaches is outside the scope of this dissertation. We therefore limit ourselves to a few of the most relevant approaches for our work.

K-Means

K-Means [85] is probably the simplest but also the most often used clustering procedure. Each cluster $\mathcal{C}_i$ has an associated cluster centroid $\overline{\mathcal{C}_i}$. K-means tries to find a clustering such that the total squared distance of each instance to the centroid of the cluster it belongs to (also called *inertia*) is minimized:

$$\text{minimize} \sum_{i=1}^{K} \sum_{x_j \in \mathcal{C}_i} \left(||x_j, \overline{\mathcal{C}_i}||_2 \right)^2 \tag{inertia}$$

It optimizes this total distance using an iterative local optimization procedure: the cluster centroids are first initialized at random and then an assignment and update step are executed alternately until convergence. In the assignment step, each instance is assigned to the cluster of the closest centroid. In the update step, the cluster centroids are recomputed as the element-wise mean of all the

instances assigned to that cluster. Many variations of this basic algorithm exist including better ways to initialize the cluster centroids [4] and different ways of optimizing the inertia [60].

Hierarchical clustering

In hierarchical clustering, the data is organized into a dendogram or binary tree based on the distances between the instances. Each node in the tree represents a set of instances, starting with the root node which contains all instances and ending with the leaf nodes which each contain only a single instance. Each intermediate node then represents a cluster of similar instances, which is the union of its child nodes and in its turn a subset of its parent. In a dendogram, the height between a parent and its children expresses the distance between the two child clusters (or instances).

This representation allows a domain expert to visually traverse the potential clustering structures of the dataset. At each intermediate node, the expert can choose to go for a more refined clustering and travel further down or stop at this node. This is especially helpful if the data itself also has a hierarchical structure. To get a final clustering, the dendogram is 'cut' at the desired level of granularity.

There are two families of hierarchical clustering algorithms, bottom-up or agglomerative clustering [33, 124], where the algorithm starts at the leafs with all instances in its own cluster and merges the closest pair of clusters until the root, and top-down or divisive clustering [72], where the algorithm starts at the root with all instances in the same cluster and recursively splits this cluster until each instance is in its own cluster.

Spectral clustering

Spectral clustering [35, 150] has it roots in graph theory, where the approach is used to identify communities of nodes in a graph based on the edges connecting them. In spectral clustering the instances are first transformed into a graph where each instance becomes a node and weighted edges between nearby nodes are added. The weight on an edge represents the similarity between the two connected instances, the higher the weight the more similar the instances are considered. Given such a graph, spectral clustering can be seen as an approximate solution to a graph cut problem where the goal is to find a partition of the graph into groups such that edges between a group have a low weight and edges within a group have a high weight [150].

Again many different variations of spectral clustering exist, but the basic procedure is as follows:

- Transform the instances into a graph and construct the weighted adjacency matrix of that graph.
- Compute the graph Laplacian L of that adjacency matrix.
- Compute the first k eigenvectors $u_1, \ldots, u_k$ of the Laplacian L.
- Construct a matrix U containing the eigenvectors $u_1, \ldots, u_k$ as columns.
- Cluster the rows of matrix U with K-Means.

This approach is often suited when the clusters are non-convex. However, a typical implementation of spectral clustering has many hyperparameters: which instances to connect with each other in the graph or which similarity measure is used to weigh the edges.

Predictive clustering trees

In predictive clustering [13], the attributes of a dataset are classified into (possibly overlapping) target attributes and descriptive attributes. The goal of predictive clustering is to cluster the instances based on their target attributes but, at the same time, learn an assignment function that assigns unseen instances to a cluster given only their descriptive attributes.

Predictive clustering trees [12] (PCTs) are a particular method for predictive clustering. A PCT is a decision tree where every leaf represents a cluster of instances. The splits in the tree are made on the descriptive attributes and are chosen such that the leafs form a good clustering. To assign a new instance to a cluster, the tree is traversed by following the branches that are satisfied for the new instance until a leaf is reached. More details on predictive clustering trees are explained in Section 4.4 where we use PCTs to generate electricity consumption scenarios for unmeasured consumers.

2.2.3 Semi-supervised clustering

Given the fact that clustering is unsupervised and the vast number of choices that need to be made (similarity metric, clustering algorithm, and hyperparameters) clustering can become a time intensive trail and error process. One might have to try multiple configurations before a desired clustering is obtained and tuning

the configuration is difficult as it is difficult to judge upfront what the impact of a particular change will be.

One approach to solving this problem is to use clustering algorithms that can leverage a limited amount of additional supervision to guide the clustering algorithm towards the desired clustering. This learning setting is called semi-supervised clustering.

There are several ways to provide some additional supervision [20] but pairwise constraints are by far the most common supervision type. A pairwise constraint is a constraint between two instances that can take two values: a must-link constraint indicates that the two instances should be in the same cluster, a cannot-link constraint indicates that the two instances should be in different clusters. This idea was first introduced by Wagstaff et al. with COPK-Means [152] but by now many unsupervised clustering algorithm have a variant that can leverage pairwise constraints: DBSCAN [78], information-maximization [21], hierarchical clustering [164], predictive clustering trees [132, 133] and many others.

Another approach to integrate constraints, is to learn a distance function based on the constraints and use that distance function in an unsupervised clustering algorithm. Many semi-supervised spectral clustering algorithms are based on this idea, in the construction of the graph used for spectral clustering the constraints are used to adapt the graph to pull instances involved in a must-link constraint closer together and push instances with a cannot-link further apart [75]. Other algorithms integrate elements of both approaches, e.g., MPCK-means [11] is an adaption of K-means that takes the constraints into account during the clustering procedure and at the same time learns a distance metric to use during clustering.

2.2.4 Evaluation of clustering algorithms

To evaluate how well a clustering algorithm clustered a particular dataset, the produced clustering X is compared with the ground truth clustering of the data Y using the adjusted rand index (ARI) [67]. The adjusted rand index is based on the rand index, but adjusts it such that adjusted rand index of a random clustering is equal to 0.

The rand index calculates the ratio of pairs of instances for which the produced clustering and the ground truth clustering agree over the number of pairs of instances. To define the rand index between a clustering X and a clustering Y,

- let a be the number of pairs of instances where both instances are in the same cluster in both clusterings,
- and let b be the number of pairs of instances where both instances are in different clusters in both clusterings

then

$$\text{rand index}(X, Y) = \frac{a+b}{\binom{n}{2}} \tag{2.5}$$

The adjusted rand index is an adjusted version of the rand index such that the expected adjusted rand index of a random clustering is equal to 0:

$$\text{adjusted rand index}(X, Y) = \frac{\text{rand index} - \text{expected rand index}}{\max \text{rand index} - \text{expected rand index}} \tag{2.6}$$

To calculate the adjusted rand index, consider the following contingency table for clustering X and Y:

$X \backslash Y$	Y_1	Y_2	$\cdots$	Y_s	sums
X_1	n_{11}	n_{12}	$\cdots$	n_{1s}	a_1
X_2	n_{21}	n_{22}	$\cdots$	n_{2s}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_r	n_{r1}	n_{r2}	$\cdots$	n_{rs}	a_r
sums	b_1	b_2	$\cdots$	b_s	

(2.7)

where n_{ij} counts the number of elements that are in cluster i of clustering X and in cluster j of clustering Y. Using the values $n_{i,j}$, a_i and b_j from the contingency table, the adjusted rand index can be calculates as

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right] \Big/ \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right] \Big/ \binom{n}{2}} \tag{2.8}$$

The adjusted rand index (ARI) is a number between -0.5 and 1 where 1 indicates a perfect clustering, 0 a random clustering and anything below 0 a clustering that is worse than random.

2.3 Anomaly detection

In anomaly detection, the goal is to automatically identify the instances that do not conform to normal or expected behavior and therefore in some sense

anomalous [28]. These deviating instances are called *anomalies*. In anomaly detection, there are often no (or not many) labels available because anomalous behavior is very rare and reproducing anomalous behavior for the sake of data collection can be very costly (e.g., letting a machine go past its usual mode of operation with the risk of breaking it). Under this assumption, anomaly detection is often considered as an unsupervised problem. In an application where the examples are measurements from a production chain, an anomaly might indicate that one of the machines is starting to operate differently then it usually does which in turn could indicate a broken part or the need for maintenance.

This is closely related to clustering. In clustering the goal is to find the underlying structure of the data by classifying instances into clusters. In anomaly detection the goal is to find the instances that do not conform to the structure of the dataset, i.e., instances that do not belong to any of the clusters.

Most anomaly detection algorithms don't directly classify instances as anomalous or normal but rather assign an *anomaly score* to each instance. The anomaly score represents the degree to which the anomaly detector thinks this instance is anomalous. Usually, the higher the anomaly score of an instance the more anomalous the instance is.

So more formally, an anomaly detector takes a set of instances $\mathcal{D}$ as input and gives each instance an anomaly score s. To convert these anomaly scores to a prediction for each instance, it is often assumed that the contamination factor c, i.e., the ratio of anomalies in the dataset, is know. The instances are ordered in increasing order of anomalousness and the top $c\%$ of the instances is predicted as anomaly.

Most anomaly detectors are transductive as there is no general model involved, reasoning happens directly on the instance level. As we'll discuss in Chapter 6, there are other anomaly detectors that reason inductively and where a general model is learned to assign anomaly scores to new instances. However, unsupervised transductive anomaly detection is by far the most common setting for anomaly detection.

2.3.1 Anomaly detection approaches

Similar to clustering, there are numerous approaches to anomaly detection [28]. All of them make different assumptions as to what constitutes in anomalous behavior. In this section, we discuss some of the most well known algorithms and the assumptions they make about anomalous behavior.

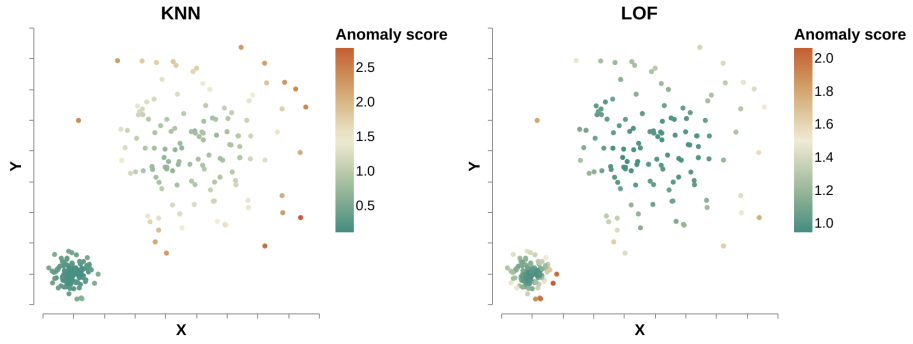


Figure 2.4: **(Left)** KNN fails to detect the anomalies around the dense cluster because of the differences in density between the two clusters. **(Right)** LOF correctly takes into account the density difference between the clusters.

K-nearest neighbor

K-nearest neighbor (KNN) [107] is a distance-based anomaly detector. It assumes that anomalies are far away from other instances in the dataset. Therefore, they propose the distance to the k'th nearest neighbor as an anomaly score. Normal instances are close to many other normal instances; therefore, the distance to the k'th nearest neighbor is low. Anomalies are far away from most instances and will thus have a high distance to the k'th nearest neighbor.

Just as in clustering, the notion of distance is again very important requiring to select an appropriate distance measure. However, in anomaly detection this aspect of the problem seems to get less attention and algorithms that require a distance measure almost always use the standard Euclidean distance.

KNN works well when the density of the instances is more or less equal over the full data space. When this is not the case, KNN will not detect all anomalies (left side of Figure 2.4). In a low density cluster it is expected that instances lie further from each other than in a dense cluster. KNN doesn't take this into account and will just consider any instance in a low density clusters as more anomalous than an instance from a high density cluster.

Local outlier factor

Local outlier factor (LOF) [18] is a density-based anomaly detector and assumes that anomalies lie in regions that have a low density relative to the density of the neigboring instances. As such, LOF uses the distances of an instance to its

neighbors to approximate the density around that instance and compares that density with the density of its neighbors. When the density of an instance is lower than the density of its neighbors that instance is considered anomalous. Using the notion of 'relative' density allows LOF to handle datasets with different local densities (right side of Figure 2.4)

Formally, LOF first approximates the density of each instance x_i as the inverse of the average reachability distance to its k-nearest neighbors.

- The reachability distance between instances x_i and x_j is defined as:

$$\text{reachability-dist}_k(x_i, x_j) = \begin{cases} \text{k-dist}(x_j), & \text{if } x_i \in \text{k-nearest neighbors}(x_j) \\ \text{dist}(x_i, x_j) & \text{otherwise} \end{cases} \tag{2.9}$$

  where k-dist is the distance of x_j to its k-nearest neighbor. Using the reachability distance instead of just the distance has a smoothing effect, "small" distances to instance x_j are set to k-dist(x_j).

- This reachability distance is then used to the local reachability density $lrd(x_i)$ of each instance x_i:

$$lrd(x_i) = \frac{1}{\frac{1}{|\text{KNN}(x_i)|}\sum_{x_j \in \text{KNN}(x_i)} \text{reachability-dist}_k(x_i, x_j)} \tag{2.10}$$

  where KNN(x_i) are the k-nearest neighbors of x_i.

To get the final anomaly score, the local reachability density of an instance is compared to the local reachability density of each of its k-nearest neighbors:

$$\text{LOF}(x_i) = \frac{1}{|KNN(x_i)|} \sum_{x_j \in KNN(x_i)} \frac{lrd_k(x_j)}{lrd_k(x_i)}. \tag{2.11}$$

If the local reachability density of the instance is lower than that of its neighbors, the instance gets a high anomaly score.

Isolation forests

Finally, isolation forests [81] is an isolation-based anomaly detector: it assume that anomalies are easy to isolate from the other instances with axis parallel splits. The main idea behind isolation forest is to learn an ensemble of trees where each tree isolates every instance in its own leaf with completely random splits (random split attribute and random split point). They then use the path

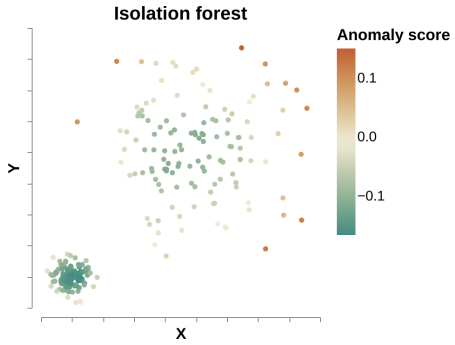


Figure 2.5: Isolation forest anomaly score on a simple 2D dataset

length of the leaf of an instance to the root as an indicator for anomalousness. If this distance is small that means that the instance can be isolated from the other data points with a small amount of splits. This leaf-to-root path length of an instance is averaged over all trees of the ensemble and used as the anomaly score for that instance. For completeness, Figure 2.5 shows the result of isolation forest on a simple 2D dataset.

2.3.2 Evaluation of anomaly detection

There are two main approaches to evaluate anomaly detection: evaluate the full ranking of the instances or evaluate the classification performance when the top instances in the ranking are marked as anomalous [22].

To evaluate the full ranking, the area under ROC curve (AUC) is often used. The AUC is a number between 0 and 1, that intuitively represents the probability that two randomly chosen instances are ranked correctly (i.e., the anomaly is more anomalous according to the ranking than the normal instance). A random ranking therefore results in an AUC of 0.5.

To calculate the area under the ROC curve, first the ROC curve is constructed by plotting the true positive rate (TPR) over the false positive rate (FPR) for all the different classification thresholds.

$$TPR = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{2.12}$$

$$FPR = \frac{\text{false positives}}{\text{false positives} + \text{true negatives}} \tag{2.13}$$

The AUC is then the area under this curve between a FPR of 0 and 1.

To evaluate the performance when the top instances in the ranking are marked as anomalous, scores such as precision @ k, i.e., the precision when the instances with the k most anomalous scores are marked as anomalous, and F1-score, i.e., the F1 score when the top contamination rate instances are marked as anomalous, are often used.

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{2.14}$$

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{2.15}$$

$$\text{F1 score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{2.16}$$

Chapter 3

Tackling Noise in Active Semi-Supervised Clustering

This chapter introduces a novel framework to handle noisy constraints in active constraint-based clustering. The framework asks additional redundant constraints from the user and uses those to reason probabilistically about noise in the set of constraints. This way, the framework can detect which constraints are most likely to be noisy and correct them. By using this novel framework, mistakes that a practitioner makes while answering queries will not have a big negative impact on the quality of the resulting clustering. Our framework can be used to make most active semi-supervised clustering algorithms robust to noise.

The content of this chapter is based on the following publication [125]:

J. Soenen, S. Dumančić, T. Van Craenendonck, and H. Blockeel. "Tackling Noise in Active Semi-supervised Clustering". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by F. Hutter, K. Kersting, J. Lijffijt, and I. Valera. Cham: Springer International Publishing, 2021, pp. 121–136. ISBN: 978-3-030-67661-2

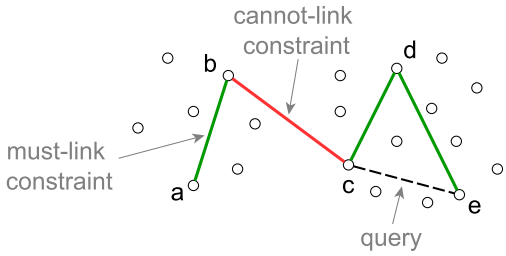


Figure 3.1: In semi-supervised clustering, constraints between pairs of instances guide the clustering algorithm towards the desired solution. A must-link constraint indicates that two instances must be in the same cluster (e.g. a and b), a cannot-link constraint indicates that two instances must be in different clusters (e.g. b and c).

3.1 Introduction

Despite being one of the fundamental data mining tasks, clustering is an inherently subjective problem [24, 84]: different users often expect different clusterings of the same dataset. In contrast to a supervised learning setting, there is no way to select the clustering algorithm, its hyper-parameters and a similarity metric based on data only; the user has to try different settings until an informative clustering is found.

Semi-supervised clustering relies on a limited amount of supervision to guide the clustering process towards an informative clustering. Such supervision comes in the form of *pairwise constraints*: a *must-link constraint* between two instances indicates that the instances must be in the same cluster, while *a cannot-link constraint* indicates that the instances must be in different clusters (Figure 3.1).

These pairwise constraints are often gathered in advance without knowing to which extent they are useful for the clustering process. Moreover, obtaining constraints usually requires human intervention and can be prohibitively expensive. In this case, it is beneficial to let the algorithm actively query the user: the algorithm presents the user with a specific pair of instances (*a query*) and the user answers with a 'must-link' or 'cannot-link' *constraint*. This way, an algorithm can pose the most informative queries first and produce a high quality clustering with a substantially smaller number of constraints. This setting is known as *active semi-supervised clustering*.

The prevalent assumption among existing semi-supervised clustering approaches is that all user-provided constraints are correct. However, this assumption is

often violated as the user might only have a vague idea of the clustering structure of the data or might simply make mistakes while answering queries. It is thus likely that a user provides inaccurate or even contradicting constraints. These noisy constraints can have a detrimental impact on the quality of the produced clustering. This is especially true for active methods which try to minimize the number of queries by asking only the most informative ones – wrongly answering such a query will have a significant impact on the final clustering.

In this chapter, we tackle the problem of handling noise in *active* semi-supervised clustering. The core idea behind our approach is to *intentionally introduce redundancy in the constraint set, by means of querying constraints that would form cycles in the available set.* Whether or not the resulting cycles are consistent will then be used to reason about noise in the constraints. For instance, answering a redundant query between instances c and e in Figure 3.1 would make a cycle $cdec$. If the user answers this query with a cannot-link constraint, the new cycle is inconsistent: the newly provided constraint states that instances c and e should not be in the same cluster, while the available must-links ($c - b$ and $b - e$) state that they should. This is contradictory information implying the existence of a noisy constraint. On the other hand, if the user answers with a must-link constraint, the new constraint forms a consistent cycle with the must-link constraints (c, d) and (d, e) and, therefore, increases our confidence in the correctness of the constraints in this cycle. To reason about noise in a principled way, we use a probabilistic model that entails the reasoning illustrated above.

To show the effectiveness of our method, we integrate it into COBRAS [144], a state-of-the-art active semi-supervised clustering algorithm. COBRAS will select queries that are informative for clustering; our approach will complement these constraints by selecting queries that help to detect and correct the noisy constraints. We focus on COBRAS because it is one of the approaches most sensitive to noise. In our experiments, we show that with our approach COBRAS becomes significantly more robust to noise.

3.2 Related work

While there is a substantial amount of research on constraint-based clustering, including active learning approaches, almost none of it explicitly deals with noisy constraints.

Most existing methods for semi-supervised clustering take pairwise constraints into account by optimising a loss function that includes a penalty for each violated constraint (e.g., PCK-means [6], MPCK-means [11], LCVQE [101] and

COSC [108]). Thus, the constraints are interpreted as soft constraints. This makes these methods robust to noise in the constraints, but it does not really *counter* noise. The approach reflects the viewpoint that it may not be possible to satisfy all constraints, but without distinguishing two different reasons for this: because the constraint is simply wrong (noisy) or because a good solution that satisfies it could not be found. Ideally, when a constraint is deemed likely to be noisy, the penalty for violating it should be lower. Moreover, whether the constraint is likely noisy should not be assessed based on how well the constraint fits the inductive bias of the clustering system, as the purpose of these constraints is exactly to change that bias.

A system that does actively try to reduce the effect of noisy constraints is COP-RF [165]. Before the clustering process starts, COP-RF filters out 50% of the constraints that are the least similar to all other constraints of the same type. Besides this, the approach by Yang et al. [158] formulates a maximum entropy model that can learn from noisy pairwise constraints. This model can be used to learn a kernel matrix from the noisy pairwise constraints, which is then used to cluster the dataset using spectral clustering [123].

None of the above approaches are active learners: they consider the set of constraints as fixed, rather than actively looking for new constraints. In an active learning context, dealing with noise is even more important: to minimize the number of queries (instances pairs to be labeled by the user), active learners try to eliminate as much redundancy as possible in the constraints, when in fact such redundancy is crucial in a noisy context. In the context of active constraint-based clustering, the only noise-handling approach we are aware of is the work by Mazumbar and Saha [88]. They propose a method that theoretically guarantees that the correct ground truth is recovered with high probability. However, this method requires multiple constraints per instance in the dataset, so the number of queries becomes orders of magnitude larger than for typical active constraint-based clustering systems. This renders the method practically infeasible for large datasets.

What we propose is a method that tries to identify noisy constraints in a way that is independent of the clustering system, and thus could in principle be combined with any of the existing constraint-based clustering methods (whether active or not, though non-active methods become active if this procedure is included). The method employs a small number of additional queries to that aim.

3.3 Reasoning about noisy constraints

In active semi-supervised clustering, the algorithm actively queries the user for the constraints between specific pairs of instances. The constraints obtained from the user are leveraged during clustering to produce high quality results. Usually, active semi-supervised clustering algorithms avoid asking redundant constraints. In a noiseless world, these redundant constraints do not give the algorithm any additional information. However, when there is noise, redundant constraints are crucial to reason about noisy constraints in a manner independent from clustering bias.

Our approach expects a set of constraints and will ask additional redundant queries to reason about the noise in these constraints. The end result is a corrected version of the given constraint set that is likely to be free of noisy constraints. This corrected set of constraints can then be used for clustering. Before we present the details of our approach, we first introduce the necessary terminology.

3.3.1 Terminology and background

We will denote a must-link constraint between instances x and y as $ml(x, y)$ and a cannot-link constraint as $cl(x, y)$. Let S be the set of instance pairs for which the user has provided a constraint. Let $\mathcal{U}$ be the set of all constraints that are obtained from the user, such that $\mathcal{U}$ contains $ml(x, y)$ or $cl(x, y)$ for each $(x, y) \in S$. The set $\mathcal{G}$ stands for the set of ground truth constraints about the instance pairs in S. The constraints in $\mathcal{G}$ are unknown; our goal is to recover these constraints from $\mathcal{U}$. If a new query is posed, the query itself is added to S, the user's answer is added to $\mathcal{U}$ and $\mathcal{G}$ will contain the corresponding ground truth constraint. For notational convenience, $\mathcal{C}(x, y)$ will denote the constraint type (ml or cl) of the constraint in the constraint set $\mathcal{C}$ between the instances x and y. A **noisy constraint** is any user-provided constraint that differs from the corresponding ground truth constraint, i.e. $\mathcal{U}(x, y) \neq \mathcal{G}(x, y)$.

For example, suppose the user has provided the constraints in Figure 3.1; then $\mathcal{U} = \{ml(a, b), cl(b, c), ml(c, d), ml(d, e)\}$. However, the constraint between d and e is actually a cannot-link constraint in the ground truth constraint set, thus $\mathcal{G} = \{ml(a, b), cl(b, c), ml(c, d), cl(d, e)\}$. In this case, the user-provided constraint $ml(d, e)$ is noisy (because $\mathcal{U}(d, e) = ml$ and $\mathcal{G}(d, e) = cl$).

Given a constraint set, we can deduce additional constraints that are implied by the constraints in the given set:

$$ml(x, y) \wedge ml(y, z) \Rightarrow ml(x, z) \qquad \text{(must-link transitivity)}$$

$$ml(x, y) \wedge cl(y, z) \Rightarrow cl(x, z) \qquad \text{(cannot-link entailment)}$$

We call a constraint set **consistent** if there is no instance pair (x, y) for which both a must-link and a cannot-link constraint are implied. In other words, a consistent constraint set does not contain any contradicting constraints.

If we represent a constraint set as a graph where nodes are instances and edges indicate the corresponding constraints (as in Figure 3.1), then the following holds:

Proposition 1. *A constraint set is inconsistent if and only if its graph contains a cycle with exactly one cannot-link edge. We call such a cycle an* ***inconsistent cycle****.*

Proof. The *if* part in the above claim is trivial. To explain the *only if* part, note that the propagation rules can only derive both ml and cl for the same instance pair (x, y) if there are at least two paths between x and y, one of which is ml-only (hence the ml derivation) and the other contains exactly one cl (two cls would break the derivation chain). These two paths form a cycle with exactly one cannot-link edge. □

If the user-provided constraint set $\mathcal{U}$ is inconsistent, there is no clustering where all constraints from $\mathcal{U}$ are satisfied and thus there must be at least one constraint in $\mathcal{U}$ that is noisy. Moreover, there is at least one noisy constraint in each inconsistent cycle in $\mathcal{U}$.

3.3.2 Overview

The goal of our work is to find a constraint set $\mathcal{C}$, derived from the user-provided constraints $\mathcal{U}$, such that we are reasonably confident that $\mathcal{C}$ corresponds to ground truth $\mathcal{G}$. To reason about the correctness of a constraint set $\mathcal{C}$, we define a probabilistic model that quantifies $P(\mathcal{G} = \mathcal{C} \mid \mathcal{U})$, i.e. the probability that the ground truth is equal to a set of constraints $\mathcal{C}$ given a set of user constraints $\mathcal{U}$. Our goal is to find a constraint set $\mathcal{C}$ such that $P(\mathcal{G} = \mathcal{C} \mid \mathcal{U}) \geq \alpha$ with α a predefined threshold. In the following, we call $P(\mathcal{G} = \mathcal{C} \mid \mathcal{U})$ the **confidence in** $\mathcal{C}$.

Algorithm 1: Verification procedure

Function `verify(`$\mathcal{U}$`):`
  $\mathcal{C} \leftarrow most_likely(\mathcal{U})$
  while $confidence(\mathcal{C},\mathcal{U}) < \alpha$ **do**
    $new_con \leftarrow$ ask_informative_redundant_query($\mathcal{U}$)
    $\mathcal{U} \leftarrow \mathcal{U} \cup \{new_con\}$
    $\mathcal{C} \leftarrow most_likely(\mathcal{U})$
  return $\mathcal{C}$

When the user-provided constraint set $\mathcal{U}$ is consistent, the most likely constraint set $\mathcal{C}$ is equal to $\mathcal{U}$. When $\mathcal{U}$ is inconsistent, there is noise in $\mathcal{U}$. In this case, the most likely constraint set $\mathcal{C}$ is the consistent constraint set that differs from $\mathcal{U}$ in as few constraints as possible.[1]

However, $\mathcal{C}$ does not automatically satisfy the confidence threshold α. There might not be enough redundancy in the constraint set to support $\mathcal{C}$. Therefore, we select additional redundant queries that complete cycles among the constraints in $\mathcal{U}$ until the confidence of the most likely constraint set $\mathcal{C}$ reaches the threshold. Adding the new constraint to $\mathcal{U}$ changes the most-likely constraint set $\mathcal{C}$. The confidence of the new $\mathcal{C}$ might be higher than that of the previous $\mathcal{C}$, e.g. the new constraint is consistent with the previous $\mathcal{C}$, or lower, e.g. a new inconsistent cycle is detected. By carefully selecting which queries we ask the user, we aim to reach the confidence threshold with as few queries as possible. Pseudocode for this procedure can be found in Algorithm 1.

In the remainder of this section, we present the probabilistic model used to calculate the confidence of a constraint set. However, because evaluating this model entails summing over all possible constraint sets, exact computation of the confidence is computationally infeasible. Therefore, we introduce a simple approximation of the actual confidence by focusing on a subset of all possible constraint sets and provide a procedure to efficiently enumerate this subset. Lastly, the way we select informative redundant queries is explained.

3.3.3 Defining the confidence of a constraint set

We develop a probabilistic approach where the value of all user constraints $\mathcal{U}(x, y)$ and the corresponding ground truth constraints $\mathcal{G}(x, y)$ are considered

[1] there might be multiple most likely constraint sets with equal likelihood

random variables. We use a Bayesian approach that computes $P(\mathcal{G} = \mathcal{C} \mid \mathcal{U})$ using the Bayes rule:

$$P(\mathcal{G} = \mathcal{C} \mid \mathcal{U}) = \frac{P(\mathcal{U} \mid \mathcal{G} = \mathcal{C})P(\mathcal{G} = \mathcal{C})}{\sum_{\mathcal{C}' \in \mathbb{C}} P(\mathcal{U} \mid \mathcal{G} = \mathcal{C}')P(\mathcal{G} = \mathcal{C}')} \tag{3.1}$$

Where $\mathbb{C}$ is the set of all possible constraint sets about the pairs in S

For the prior $P(\mathcal{G} = \mathcal{C})$, we assume a uniform distribution over all *consistent* constraint sets $\mathcal{C}$. This implies that all inconsistent constraint sets have a probability of 0, thus we can sum over all consistent constraint sets instead of all constraint sets. As $P(\mathcal{G} = \mathcal{C})$ is equal for every consistent constraint set, it can be ignored (assuming $\mathcal{C}$ is consistent, otherwise $P(\mathcal{G} = \mathcal{C} \mid \mathcal{U})$ is equal to 0).

We assume i.i.d. noise: there is a fixed probability ν that the user answers a query incorrectly. Consequently, for each pair (x, y) and label c, $P(\mathcal{U}(x, y) \neq c \mid \mathcal{C}(x, y) = c) = \nu$. With this assumption in place, the likelihood $P(\mathcal{U} \mid \mathcal{G} = \mathcal{C})$ can be written in function of n, the number of constraints in $\mathcal{U}$, and d, the number of constraints where $\mathcal{U}$ and $\mathcal{C}$ disagree.

$$P(\mathcal{U} \mid \mathcal{G} = \mathcal{C}) = \nu^d (1 - \nu)^{n-d} \tag{3.2}$$

Note that the assumptions behind the probabilistic model are likely to be violated in practice; for instance, some pairs of instances may be easier to correctly label than others. Despite being violated in practical cases, the model allows us to set formal foundations for tackling noise.

3.3.4 Approximating the confidence of a constraint set

To calculate the confidence of a certain constraint set $\mathcal{C}$, we have to sum over all consistent constraint sets $\mathcal{C}'$ (see denominator of Equation 3.1). Naively enumerating all consistent constraint sets is not practically feasible as the number of consistent constraint sets is, in the worst case, exponential in the number of constraints. Therefore, we introduce an approximation that relies on a subset of the consistent constraint sets that contribute the most to the confidence and provide a procedure that finds these constraint sets efficiently.

High-likelihood constraint sets.

Given our noise model, the contribution of a consistent constraint set $\mathcal{C}'$ to the denominator of the confidence is equal to $\nu^d (1 - \nu)^{n-d}$ with n the total

number of constraints in $\mathcal{U}$ and d the number of pairs for which $\mathcal{U}$ and $\mathcal{C}$ disagree (Equation 3.1, 3.2). For small ν, terms with high d contribute little to the sum. We therefore only sum over consistent constraint sets $\mathcal{C}'$ with $d \leq k$, for some parameter k called the **approximation order**. Higher values for k result in more accurate approximations, but increase the execution time of the algorithm considerably, as the size of the search space is exponential in k.

This approximation is accurate if it captures most of the probability mass, that is, when there is a high probability that the number of noisy constraint in $\mathcal{U}$ is smaller than the approximation order k. From our noise assumption follows that the number of noisy constraints in $\mathcal{U}$ follows a binomial distribution distributed with $|\,\mathcal{U}\,|$ trials and success probability ν.

$$P(\#\text{noisy constraint in } \mathcal{U} \leq k) = \sum_{d=0}^{k} \binom{|\mathcal{U}|}{d} \nu^d (1-\nu)^{|\mathcal{U}|-d} \tag{3.3}$$

If k is fixed, $P(\#\text{noisy constraints in } \mathcal{U} \leq k)$ increases when the size of $\mathcal{U}$ decreases or the noise probability ν decreases. Thus our approximation is most accurate for small constraint sets and low noise probabilities. If the approximation order k is too small relative to $|\mathcal{U}|$ and ν, the approximation overestimates the confidence and our approach will ask less redundant constraints than actually necessary to reach the desired confidence level.

To make our approximation more accurate for bigger constraint sets we employ two tricks: we use a dynamically growing *effective approximation order* and verify the user-provided constraints in *batches*.

When using a *static* approximation order k, the approximation becomes less accurate as the number of identified noisy constraints increases. After identifying k noisy constraints, all consistent constraint sets with more than k noisy constraints are not considered in the approximation of the confidence. This results in a highly overestimated confidence and causes the procedure to never identify more than k noisy constraints.

In order to resolve this, we use a *dynamically growing* **effective approximation order** k' instead of the static approximation order k. The effective approximation order k' is determined as the sum of the number of noisy constraints in the current most likely constraint set and the approximation order k. The effective approximation order thus grows each time a noisy constraint is detected in the user-provided constraints $\mathcal{U}$. This ensures that the approximation can always reason about k additional noisy constraints on top of the noisy constraints that have already been identified.

Additionally, the user-provided constraints can be verified in batches to enhance the accuracy of the approximation; instead of verifying all user-provided

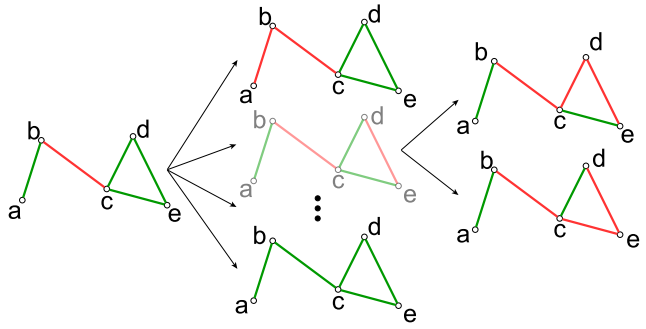


Figure 3.2: An example of an incomplete search tree to find consistent constraint sets. The starting constraint set is consistent, hence every constraint is a candidate for flipping. The second constraint set at depth 1 is inconsistent, only the constraints between instance pairs (c, d) and (c, e) need to be considered as candidates ((d, e) was flipped in a previous recursive call).

constraints $\mathcal{U}$ in one single go, $\mathcal{U}$ is divided in several smaller batches. One after the other, each of these batches is added to a constraint set $\mathcal{U}'$. After adding a batch to $\mathcal{U}'$, the full set is verified. Because the preexisting constraints in $\mathcal{U}'$ have been verified in the previous iteration, it is unlikely that these preexisting constraints contain undetected noisy constraints. The number of noisy constraints in the newly added batch $\mathcal{B}$ follows the same distribution as in Equation 3.3 but proportional to the $|\mathcal{B}|$ instead of $|\mathcal{U}|$, with $|\mathcal{B}| < |\mathcal{U}|$. Therefore, the accuracy of the approximation will mostly depend on the size of the newly added batch.

For several active semi-supervised clustering approaches it is actually natural to verify the constraints in batches. NPU [154] and COBRAS [144] produce a new intermediate clustering every couple of queries. Therefore, it makes sense to verify the constraints right before each intermediate clustering is produced as to make sure that none of these intermediate clusterings is based on a noisy constraint.

Enumerating consistent constraint sets.

Algorithm 2 outlines a simple yet efficient recursive procedure to enumerate all consistent constraint sets up to the effective approximation order k'. These consistent constraints sets are used to approximate the confidence of a constraint set, determine the most likely constraint set and select redundant queries. In each recursive call, one constraint in a constraint set $\mathcal{A}$ is flipped until it reaches

Algorithm 2: Procedure to enumerate consistent assignments

```
Function solve(A, flipped):
    if unsat_user_constraints(A) > k' then
        return ∅
    solutions ← ∅
    if A is consistent then
        solutions ← solutions ∪ {A}
        candidates ← U \ flipped
    else
        cycle ← find_inconsistent_cycle(A)
        candidates ← cycle \ flipped
    for con in candidates do
        flipped ← flipped ∪ {con}
        A' ← A \ con ∪ {con.flip()}
        solutions ← solutions ∪ solve(A', flipped)
    return solutions
```

the desired effective approximation order. If the constraint set $\mathcal{A}$ is consistent, each of the constraints in $\mathcal{A}$ is considered as a candidate to be flipped next. If the constraint set $\mathcal{A}$ contains an inconsistent cycle, the only way this constraint set can become consistent is to flip one of the constraints involved in the inconsistent cycle. Therefore, only the constraints involved in the inconsistent cycle have to be considered as candidates to be flipped next. Every consistent constraint set that is encountered during this procedure is stored. To ensure that every consistent constraint set is only encountered once, a set of constraints *flipped* is maintained that keeps track of all the constraints that have already been flipped and should not be flipped again. Figure 3.2 shows part of the search tree explored by Algorithm 2 for an example scenario.

Finding inconsistent cycles.

To be able to use Algorithm 2, we need to detect when a constraint set is inconsistent and, if so, find an inconsistent cycle in the constraint set.

To check if a constraint set is inconsistent, we use a procedure similar to the feasibility test with must-link and cannot-link constraints of Davidson and Ravi [31]. The procedure maintains a set of must-link components, groups of instances that are interconnected by must-link constraints, and checks whether there are two instances within the same must-link component that are connected by a

cannot-link constraint. If such a pair of instances is found, this cannot-link forms an inconsistent cycle with some must-links from this must-link component and thus the constraint set is inconsistent. To reconstruct the inconsistent cycle, we start from the two instances connected by the cannot-link and search for a path between them that only contains must-link constraints.

3.3.5 Selecting redundant queries

Now that we have all pieces to calculate the confidence of the most likely constraint set $\mathcal{C}$, we still have to select an informative query that will help us to reach the confidence threshold.

We distinguish two different scenarios in which a redundant query needs to be selected.

1. If there is only one most likely constraint set $\mathcal{C}$, we aim to increase the confidence of this constraint set. Therefore, we query the pair (x, y) for which the total likelihood of consistent constraint sets that imply a different constraint value for (x, y) than $\mathcal{C}$ is maximal. As before, we only use the consistent constraint sets up to the current effective approximation order k'.
2. If there are multiple most likely constraint sets, we query a pair that will reduce the amount of most likely constraint sets. As such, we aim to query a pair (x, y) for which the amount of most likely constraint sets that imply must-link is approximately equal to the amount of most likely constraint sets that imply cannot-link. The answer to this query will thus agree with approximately half of these constraint sets and disagree with the other half. Adding the answer of this query to $\mathcal{U}$ causes the confidence of the agreeing constraint sets to increase and the confidence of the disagreeing constraint sets to decrease. This gives rise to a binary-search-like procedure which is repeated until only one most likely constraint set remains.

This greedy query strategy only asks queries that immediately add additional redundancy to the constraint set. If there is a single isolated (i.e., not connected to any other constraints) must-link constraint between instances x and y, this constraint could be verified by asking two queries between x and some new instance z and between y and z. Because the first query on its own is not immediately informative, our query selection strategy will never select it.

However, for almost all active semi-supervised clustering algorithms the (relevant) constraint set form a connected graph where every pair of must-link components is connected with a cannot-link constraint. The constraints in

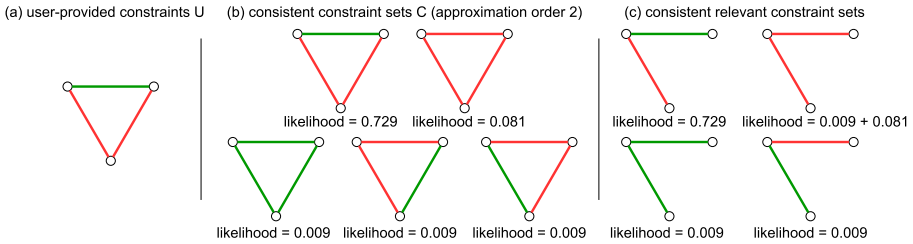


Figure 3.3: Illustration of how the consistent constraints sets are marginalized over the irrelevant constraints to only retain the relevant constraints. (a) shows the given user-provided constraints (b) shows all consistent constraint sets and their likelihood derived from the user-provided constraints with approximation order 2 (c) shows the marginalized constraint sets with their likelihood.

a must-link component can be verified if there are enough instances involved compared to the number of noisy constraints that occur. The relation between two must-link components can be verified if both must-link components have enough instances (compared to the number of noisy constraints). Therefore, the greedy query selection strategy might fail to verify constraint sets with must-link components that only involve a couple of instances. If this scenario occurs and there are no interesting queries are left (e.g., all possible instance pair within a single must-link component have been queried) and the confidence of the most likely constraint set is still lower than α, the most likely constraint set (or one of the most likely constraint sets if there are multiple) is assumed to be correct.

3.4 Integration in a state-of-the-art active semi-supervised clustering algorithm

COBRAS [144] is a state-of-the-art active semi-supervised clustering algorithm. Its strengths are that it is time-efficient, query-efficient, and that it provides intermediate results. Its query-efficiency, however, naturally makes it more sensitive to noise. Moreover, COBRAS has no noise handling mechanism, as the algorithm aims to satisfy all of the user-provided constraints. Therefore, it is a good candidate for testing the effectiveness of our noise correcting approach.

COBRAS is an iterative algorithm where each iteration consists of a splitting phase and a merging phase. In its splitting phase, COBRAS identifies so-called super-instances, which are groups of instances that are assumed to belong to the

same cluster. During splitting, a small amount of queries are used to estimate the number of super-instances that are needed. The new super-instances (groups of instances) are then merged into clusters by querying constraints between different super-instances. Two super-instances are merged into the same cluster if the constraint between them is a must-link; two super-instances are not merged together if the constraint between them is a cannot-link. It is after this phase that our noise-correction approach is deployed.

After each merging phase, our approach tries to recover the correct constraints $\mathcal{C}$ from the user-provided constraints $\mathcal{U}$. If no noisy constraints are detected (i.e. $\mathcal{C} = \mathcal{U}$), COBRAS can safely continue with the next iteration. Otherwise, the current clustering is based on at least one noisy constraint and has to be corrected. Therefore, the merging phase of COBRAS is repeated with the corrected constraint set $\mathcal{C}$ instead of $\mathcal{U}$. We refer to the thus obtained noise-robust version of COBRAS as nCOBRAS.

Reasoning on a subset of relevant constraints

In COBRAS, only a subset of all the constraints provided by the user is used to construct the next intermediate clustering (i.e. only the constraints between the current super-instances are used). Thus, it makes sense to only verify the constraints from this relevant subset and not waste queries on obtaining the correct values of the irrelevant constraints. However, the irrelevant constraints might still complete cycles with relevant constraints and thus might still provide useful information. Therefore, to calculate the confidence of a subset of relevant constraints $\mathcal{C}_{rel}$, we reason on the full set of constraints and marginalize over the irrelevant constraints $\mathcal{C}_{irrel}$.

$$P(\mathcal{G}_{rel} = \mathcal{C}_{rel} \mid \mathcal{U}) = \sum_{\mathcal{C}_{irrel} \in \mathbb{C}_{irrel}} P(\mathcal{G} = \mathcal{C}_{rel} \cup \mathcal{C}_{irrel} \mid \mathcal{U}) \tag{3.4}$$

Where $\mathbb{C}_{irrel}$ is the set of all possible constraint sets involving the same pairs as the irrelevant constraints

To ensure that the redundant query selection procedure only selects queries that are informative to verify the relevant constraints, all consistent constraint sets enumerated by Algorithm 2 are marginalized over the irrelevant constraints. The active query selection procedure is then executed on the marginalized constraint sets that only contain the relevant constraints (Figure 3.3).

3.5 Experiments

We compare the clustering performance of several active semi-supervised clustering approaches in the presence of varying amounts of noise. We are especially interested to investigate **(Q1)** whether nCOBRAS is indeed more noise robust than COBRAS and **(Q2)** how nCOBRAS performs compared to existing active constraint-based clustering approaches.

3.5.1 Algorithms

We compare nCOBRAS[2] to the following active semi-supervised clustering algorithms:

- COBRAS [144], the original COBRAS algorithm with no noise handling mechanism. We use the code provided by the authors.[3]
- NPU [154] is an active query selection scheme that can be used with any semi-supervised clustering algorithm, we use it with:
  - MPCK-means [11], a constrained modification of k-means that uses a modified objective function and allows constraint violation. We use the implementation in the wekaUT package.[4]
  - COSC [108], a constrained extension of spectral clustering that optimizes a modified objective function. We use the fast implementation of the code provided by the authors.[5][6]

NPU with MPCK-means (NPU_MPCK) and NPU with COSC (NPU_Cosc) require the desired number of clusters K before clustering. In our experiments the true value of K (as indicated by the class labels) is given to these algorithms. This gives NPU-MPCK and NPU-Cosc an advantage over COBRAS and nCOBRAS, which do not require the number of clusters.

nCOBRAS needs three hyperparameters to be set: an estimated noise probability ν, the confidence threshold α and the approximation order k. In our experiments, we provide nCOBRAS with the true noise probability for ν, except when there is no noise: there, we set ν to 0.05 instead of 0, so that we can get an idea

[2] `https://github.com/jonassoenen/noise_robust_cobras`
[3] `https://dtai.cs.kuleuven.be/software/cobras/`
[4] `https://www.cs.utexas.edu/users/ml/risc/code/`
[5] `https://www.ml.uni-saarland.de/code/cosc/cosc.htm`
[6] In some cases the code provided by the authors fails to produce a clustering; if this happens, we use the clustering from the previous NPU iteration instead

of the cost of noise-handling when it is in fact unnecessary. The value of the confidence threshold α is set to 0.95 and an approximation order of 3 is used.

3.5.2 Experimental methodology

We perform 3 times 10-fold cross-validation and report the average clustering quality. The algorithms cluster the full dataset, but only query constraints between instances that are both in the training set. The clustering quality is determined by calculating the adjusted rand index (ARI) [67] on the instances in the test set. The ARI measures the similarity between the produced clustering and the ground truth clustering. A random clustering has an expected ARI of 0, while a clustering identical to the ground truth clustering has an ARI of 1.

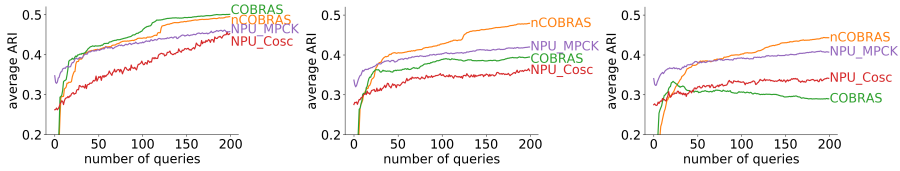
To guarantee that COBRAS and nCOBRAS do not query any test instances during clustering, we ensure that all super-instance representatives are in the training set. For NPU, the selection of the most informative instance x^* is modified to exclude points not in the training set.

We also report the average aligned rank (AAR) [48, 64], which ranks the overall performance of each of the algorithms. The lower the AAR the better the algorithm performs comparatively to the other algorithms. To calculate the average aligned rank, for each dataset d and each algorithm a, compute the difference between the average ARI of algorithm a on dataset d and the average ARI achieved by all algorithms on dataset d. The resulting differences are ranked from high to low. The AAR for an algorithm is the average of the positions of its entries in the sorted list.

The noisy constraints are generated according to the noise assumption presented in Section 3.3: for an experiment with $x\%$ of noise, each of the queries has a probability of $x\%$ that the query is answered incorrectly.

3.5.3 Datasets

For our experiments, we use 21 clustering tasks defined over 17 datasets as in Van Craenendonck et al. [144]. These datasets include 15 UCI classification datasets: iris, wine, dermatology, hepatitis, glass, ionosphere, optdigits389, ecoli, breast-cancer-wisconsin, segmentation, column 2C, parkinsons, spambase, sonar and yeast. For these datasets, the class labels indicate the target clustering. Additionally, we consider 4 clusterings of the CMU faces dataset and cluster 2 different subsets of the 20 newsgroup text dataset. For details about the preprocessing and the clustering tasks, we refer to Van Craenendonck et al.[144].



no noise 5% of noise 10% of noise

Figure 3.4: Average ARI comparison of the different algorithms in the presence of varying amounts of noise (higher is better).

3.5.4 Results

We ran each of the algorithms for 200 queries with varying amounts of noise (0%, 5% and 10%). For each noise value, we calculate the average ARI and the average aligned rank of each algorithm over all clustering tasks. The results are shown in Figure 3.4 and Figure 3.5.

(Q1) The results (Figure 3.4) confirm that nCOBRAS is less sensitive to noise than COBRAS. The average clustering quality of COBRAS significantly decreases when the amount of noise increases: with 10 % of noise, its ARI decreases by half. In contrast, the average clustering quality of nCOBRAS is relatively stable over all amounts of noise. Noise causes nCOBRAS' improvement to slightly slow down: with more noise, nCOBRAS has to ask more redundant queries to verify the constraints. Moreover, with no noise nCOBRAS' performance is very close to that of COBRAS which does not reason about the correctness of the constraints.

(Q2) The results also indicate that nCOBRAS outperforms both competitors, NPU-Cosc and NPU-MPCK, for all levels of noise. NPU-MPCK outperforms nCOBRAS for small amounts of constraints; this is due to the combined unsupervised and semi-supervised objectives of MPCK-means which allows it to get to a better initial clustering than nCOBRAS. Both NPU-MPCK and NPU-Cosc are less sensitive to noise than COBRAS.

3.5.5 Evaluation of noise detection

To show that our approach actually finds and corrects the majority of the noisy constraints, we calculate the average precision and average recall achieved by our noise detection algorithm when correcting a relevant constraint set with 5% of noise (Table 3.1). For the majority of the datasets, the average recall is higher than 93% and the average precision is higher than 95%. This means our

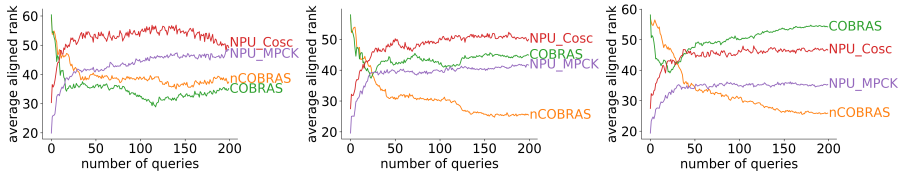


Figure 3.5: Average aligned rank of the different algorithms in the presence of varying amounts of noise (lower is better).

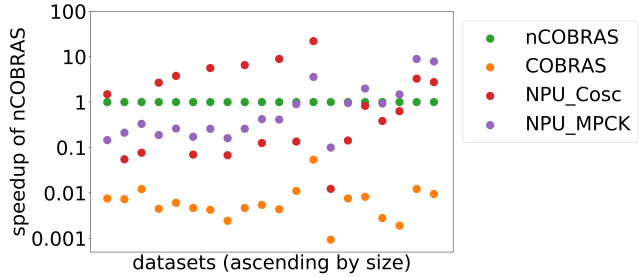


Figure 3.6: Ratio of competitor to nCOBRAS average runtime for each of the 21 clustering tasks (sorted ascending by size) when there is 5% noise and the algorithms are run for 200 queries.

approach identifies almost all noisy constraints in the relevant constraints, and almost all constraints flagged by our approach are indeed noisy. For datasets where the average batch size is high, the precision and recall are typically lower; in this case our approximation fails to produce a reasonable estimate of the confidence.

3.5.6 Runtime

Figure 3.6 shows the ratio of the average run time of each competitor to the average runtime of nCOBRAS for each of the 21 clustering tasks with 5% of noise for 200 queries. COBRAS is by far the fastest algorithm. nCOBRAS' noise identification procedure makes it substantially slower than COBRAS, but it is still about as fast as the other systems.

Table 3.1: The average precision, the average recall, and the average batch size achieved by nCOBRAS for each dataset.

dataset	Precision	Recall	Batch size
breast-cancer-wisconsin	0.9984	0.9426	2.1717
column_2C	0.9989	0.9342	2.5741
dermatology	0.9477	0.8250	2.5893
ecoli	0.9239	0.8748	4.5755
faces_expression_imagenet	0.8865	0.6896	13.5932
faces_eyes_imagenet	0.9914	0.9417	2.2642
faces_identity_imagenet	0.7643	0.3396	66.1111
faces_pose_imagenet	0.9961	0.9470	27.3407
glass	0.9535	0.9066	9.6528
hepatitis	0.9965	0.9856	2.6405
ionosphere	0.9924	0.9675	2.7861
iris	0.9941	0.9512	2.3684
newsgroups_diff3	0.9918	0.9616	1.9859
newsgroups_sim3	0.9817	0.9487	3.2786
optdigits389_full	0.9666	0.9392	2.0097
parkinsons	0.9960	0.9420	2.9945
segmentation	0.8584	0.7915	14.4579
sonar	0.9986	0.9563	3.6352
spambase	0.9988	0.9882	2.8426
wine	0.9918	0.9483	2.5797
yeast	0.9243	0.8513	15.1755

3.6 Conclusion

In this chapter, we have presented a novel approach to handling noise in active semi-supervised clustering. Our method reasons probabilistically about the correctness of constraints and can correct noisy constraints with the help of additional redundant queries. We integrated our method into the state-of-the-art semi-supervised clustering algorithm COBRAS that usually outperforms its competitors but is very sensitive to noise. Under noisy conditions, some of COBRAS' competitors (which are less sensitive to noise) produce better clusterings than COBRAS. Our noise robust version of COBRAS makes COBRAS significantly more robust to noise and makes it outperform its competitors again, even under noisy conditions.

Chapter 4

Scenario generation of residential electricity consumption through sampling of historical data

This chapter discusses the research we did within the low voltage use case of the Flanders AI research program. In collaboration with EnergyVille, we developed several techniques to predict the electricity consumption time series of a specific household under specific circumstances for households where no fine-grained electricity consumption measurements are available. The developed techniques enable EnergyVille to do detailed simulations of the low voltage grid to efficiently plan the reinforcement of the grid. The reinforcement of the grid is one of the challenges we need to tackle to support our future energy needs.

The content of this chapter is based on the following two publications [126, 127] (one of which is published, the other in preparation):

J. Soenen, A. Yurtman, T. Becker, R. D'hulst, K. Vanthournout, W. Meert, and H. Blockeel. "Scenario generation of residential electricity consumption through sampling of historical data". In: *Sustainable Energy, Grids and Networks* 34 (2023), p. 100985. ISSN: 2352-4677. DOI: https://doi.org/10.1016/j.segan.2022.100985. URL: https://www.sciencedirect.com/science/article/pii/S2352467722002302

J. Soenen, A. Yurtman, T. Becker, K. Vanthournout, and H. Blockeel. "Generating electricity consumption scenarios with predictive clustering trees". In: *In preparation* (2023)

4.1 Introduction

The world is transitioning away from fossil fuels towards renewable energy sources to reduce greenhouse gas emissions. The European Commission has made binding declarations to achieve zero net greenhouse gas emissions by 2050 in the European green deal [44]. Because the production and use of energy accounts for more than 75% of the EU's greenhouse gas emissions, decarbonizing the EU's energy system is critical to reach carbon neutrality [43]. The United States recently approved the Inflation Reduction Act [137], a significant investment in the modernization of the American energy system. This legislation, combined with previous initiatives, puts the US on track to reduce their greenhouse gas emissions with an estimated 40% by 2030 (compared to 2005)[143]. Countries all around the world are setting similar targets[1].

This large-scale energy transition will require significant investments on many fronts, one of which is the low-voltage grid (LVG). Widespread adoption of electric vehicles, heat pumps, and photovoltaic (PV) panels will change electricity consumption and production habits significantly [93]. The addition of these devices will increase the peak load on the LVG substantially, pushing it beyond its current limits. To handle future demand, the LVG must be reinforced [130]. Moreover, it is anticipated that the pace of reinforcement will significantly surpass the present rate of renewal for the LVG [39].

To plan such a reinforcement for the LVG effectively, one needs to know its current state, that is, the currents and voltages on individual cables and buses on fine-grained time scales (e.g., 15 min). These quantities are typically not measured for the entire grid due to its massive scale, and hence, need to be estimated with a model. Typical modeling approaches, such as a power flow calculation, require the electricity consumption time series of each connected consumer as input.

However, in Europe, for the vast majority of consumers, these electricity consumption time series are unavailable. Not all consumers have smart meters installed [149], the data might be measured but cannot be used due to privacy reasons, or it is too cumbersome to securely transmit, store and manage all

[1] See `https://climateactiontracker.org/` for a comprehensive overview

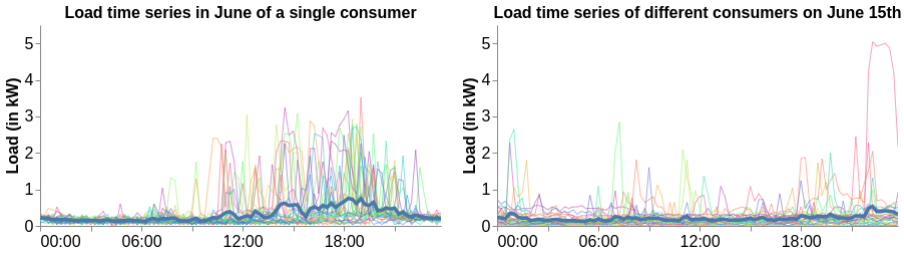


Figure 4.1: Averaging load time series removes load peaks and smooths the individual time series excessively. (left) daylong load time series of a single consumer in June (thin lines) and their average (thick line). (right) daylong load time series of different consumers on the 15th of June and their average (thick line). The sampling period is 15 min.

the data on a central server. Therefore, a distribution system operator (DSO) only has access to the electricity consumption time series of a limited set of so-called *measured consumers* [91]. Additionally, DSOs typically have access to certain general (but often limited) attributes about *every* consumer such as yearly consumption and connection capacity.

To solve this data availability problem, one could predict the electricity consumption time series of the unmeasured consumers using the available data (attributes for all consumers and time series for a limited set of consumers).

However, the literature on predicting electricity consumption time series given some consumer attributes is limited. DSOs often use synthetic load profiles (SLPs) to model the electricity consumption of several consumer types. Such an SLP consists of a single time series for each consumer type which represents the *average* behavior. However, as these SLPs are averages, they vastly underestimate peak loads and don't capture the full variability in load (Figure 4.1). Therefore, SLPs are not suited to analyse the LVG load and congestion risk.

Given the drawbacks of SLPs, many existing LVG studies sample randomly from the known time series [91, 92, 146] or learn a probabilistic model from the known time series and generate samples from that model [9, 10]. These approaches sample multiple load time series or *load scenarios* to capture the full variability of the load. However, they do not leverage the available consumer attributes, so the generated scenarios are not tailored toward individual consumers.

We close this gap in the literature by proposing techniques to generate realistic and accurate daylong load time series scenarios for a specific unmeasured consumer under specific circumstances, given the attributes of that consumer and daily attributes describing the circumstances (e.g., weather and calendar

information). To this end, we leverage a dataset of a limited number of consumers (the so-called *measured consumers*) with historical time series data as well as consumer and daily attributes. To generate time series data for a given unmeasured consumer with certain attributes, we sample from the most relevant part of this historical dataset based on the given attribute values. By sampling from historical data, any scenario that we generate is an actual load time series of a real household and therefore realistic.

In our first publication on the topic [126], we proposed two techniques to solve this scenario generation task. The expert-based technique uses expert knowledge to define how the available attributes are used to sample load scenarios from the historical data. On the other hand, the data-driven technique doesn't require any expert knowledge and employs machine learning techniques to *learn* how to leverage the attributes to sample scenarios. Our experiments show that both techniques outperform the baselines of random sampling and SLPs [126]. The data-driven technique performs better than the expert-based technique and, contrary to the later, can easily be applied to datasets with different attributes without requiring any domain knowledge.

In our second publication on the topic, we further improved upon the data-driven technique. The data-driven technique is a complex model consisting of multiple ensembles of decision trees, resulting in a computationally inefficient model with only limited interpretability. By using a predictive clustering tree (PCT) [12], we were able to simplify the data-driven technique considerably without loss of predictive performance. Because this simplified model consists of only a single decision tree, it is interpretable, helping to build trust in the model, and scaleable, enabling the method to be trained on large datasets. Our experiments show that this approach based on predictive clustering is at least as accurate as the previously proposed data-driven and expert-based techniques while being significantly faster in training and prediction. The interpretability of the model enables DSO's to discover and understand the patterns in the data.

In the remainder of this chapter, we discuss the related work (Section 4.2), the developed methods (Section 4.3), the experimental methodology (Section 4.5), the results (Section 4.6), and, finally, the conclusion (Section 4.7).

4.2 Related work

There are three fields that are related to scenario generation of unmeasured consumers but none of them currently offers a solution: time series forecasting, residential load modeling and generative models. Electrical load forecasting

(Section 4.2.1) generally assumes that there are historical electricity consumption data available for the consumer whose future consumption is to be predicted; however, such historical consumption data are not available in our setting. Residential load modeling techniques (Section 4.2.2) aim to model the process that generates the total electricity consumption of a residence or group of residences. However, existing techniques in this field are not appliceable for one of three reasons: they study averaged load time series, they use detailed information not available to a DSO (or information that the DSO cannot use due to privacy regulations), or they don't tailor the generated scenarios to specific consumers. Finally, fully parametric generative models (Section 4.2.3) have been applied for scenario generation on the distribution and utility level; however, it is still unclear whether or not they can capture the characteristics of individual consumers.

4.2.1 Electrical load forecasting

At first sight, scenario generation might look similar to time series forecasting; however, there is a key difference. In forecasting, the goal is to predict future electricity consumption values given past electricity consumption values of the same consumer [109, 122, 151]. In the scenario generation setting that we consider, there are no historical consumption values available for the profiles that have to be predicted.

4.2.2 Residential load modeling

A more closely related field is that of residential load modeling. The goal of residential load modeling is to model the process that generates the total electricity consumption of a residence or group of residences [56, 104, 135]. There are three categories of residential load modeling techniques: bottom-up, top-down and hybrid techniques.

Bottom-up approaches Bottom-up techniques aggregate and extrapolate from fine-grained electricity consumption information, micro-variables (e.g., appliance-use data, occupant behavior, or individual consumer consumption profiles) to load patterns or profiles on a coarser level (e.g., a single consumer, a group of consumers, or a nation). Examples of this type of load modeling technique include Capasso et al. [23] and Gottwalt et al. [55]. These techniques are generally inapplicable as the fine-grained electricity consumption information required is not available to DSOs; this type of data is more difficult to gather than the electricity consumption profiles themselves.

Top-down approaches Top-down techniques use total electricity consumption time series and try to relate that electricity consumption to aggregate-level macro variables (e.g., household characteristics, weather, and macro-economic indicators) and/or stochastic predictors (arising from the analysis of historical time series data) [104]. Some of these top-down techniques can generate electricity consumption profiles for individual connection points. However, many of these techniques generate averaged or aggregated load profiles to represent the general trend of electricity consumption profiles [89]. Similar to SLPs, these averaged or aggregated load profiles do not capture the full variability in load scenarios and are therefore unsuited as input to a power flow calculation. Other techniques do try to generate realistic load profiles but fail to capture the temporal correlation or can only leverage a single attribute [139] (or in some cases no attributes at all [77]).

Hybrid approaches Hybrid techniques [5, 71] combine elements and methods from both bottom-up and top-down techniques. These techniques are also inapplicable, as there are no micro variables available in our problem setting.

4.2.3 Generative models

Generative neural network architectures such as variational auto-encoders [74], generative adversarial networks [53], normalizing flows [110] and diffusion models [63] could also be used to conditionally generate load scenarios. However, such generative models offer very limited or no interpretability, resulting in a model that is difficult to trust without verifying the predictions. They furthermore introduce additional modeling challenges, as they should reproduce appropriate autocorrelation structure where peaks are correctly timed and scaled. In contrast, our approach of sampling from historic data guarantees that generated scenarios are realistic.

To the best of our knowledge, these generative models have only been applied on easier tasks: generating scenarios for aggregated electricity consumption data (e.g., at the utility or distribution level) [8, 37, 49] or generating load scenarios for large buildings [159]. Such electricity consumption time series (aggregated or from large buildings) contain less variance and are more predictable making them easier to model.

4.2.4 Summary

In summary, none of the existing techniques are able to generate realistic electricity consumption scenarios for individual consumers given consumer and daily attributes. Time series forecasting and bottom-up residential load modeling require data that are unavailable in our setting. Some top-down residential load modeling techniques can generate electricity consumption time series for individual consumers. However, these techniques suffer from at least one of two drawbacks: (1) they generate time series which are unrealistic (e.g., averaged); or (2) they can only leverage a limited amount of attributes to generate scenarios (e.g., only yearly consumption). For fully parametric generative models it is still unclear whether they can capture the characteristics of individual electricity consumption time series.

Therefore, we propose three scenario generation techniques that can leverage both attributes about the consumer as daily attributes about the circumstances to generate realistic electricity consumption scenarios. All techniques use the attributes to sample scenarios from real historical electricity consumption time series thereby guaranteeing that the generated scenarios are realistic.

4.3 The expert-based and data-driven approach

4.3.1 Problem statement and notation

Given: a collection of N_c yearlong measurements of measured consumers:

$$\{(\mathbf{t}_i, \mathbf{y}_i, [\mathbf{d}_{i,1}, \cdots, \mathbf{d}_{i,365}]) \ : \ i = 1, \ldots, N_c\} \tag{4.1}$$

where each yearlong measurement is represented by a tuple of a yearlong electricity consumption time series $\mathbf{t}_i$, consumer attributes $\mathbf{y}_i$, and daily attributes $\mathbf{d}_{i,j}$ for each day j of the year. The yearlong electricity consumption time series, represented by the vector $\mathbf{t}_i$, is a concatenation of 365 daylong time series represented by vectors $\mathbf{t}_{i,j}$ $(j = 1, \ldots, 365)$ of length T (the number of consumption measurements in a day). The vector $\mathbf{y}_i$ of consumer attributes contains all known attributes for the consumer such as yearly consumption and connection capacity. Note that these attributes might differ for different years of a single consumer. The vector $\mathbf{d}_{i,j}$ contains daily attributes for day j of the year in which $\mathbf{t}_i$ was measured and consists of calendar information (such as day of week) and weather information (such as average temperature of the day). A consumer that has multiple years of available data is represented by multiple tuples in the dataset, one for each available year of measurements. A visual

illustration of the dataset is provided in Figure 4.2.

Do: generate N_s load scenarios $\{\hat{\mathbf{t}}^{(s)} : s = 1, \ldots, N_s\}$, i.e., a daylong electricity consumption time series, for a given unmeasured consumer with consumer attributes $\mathbf{y}'$ for a given day with daily attributes $\mathbf{d}'$.

Note that the main goal is not forecasting, but modeling consumer electricity consumption behavior at the present time (i.e., the time at which the historical data used were measured). Our models are able to generate scenarios for the future, possibly with reduced accuracy because (1) consumer and weather attributes might need to be estimated or forecasted, and (2) the time series in the historical dataset becomes less representative over time as consumer behavior changes.

4.3.2 Methodology

To generate a single daylong load scenario for a given unmeasured consumer with consumer attributes $\mathbf{y}'$ for a given day with daily attributes $\mathbf{d}'$, both of the expert-based and data-driven follow a two-level approach: first, the consumer attributes $\mathbf{y}'$ are used to select yearlong load time series. Second, from these yearlong load time series, the daily attributes $\mathbf{d}'$ are used to select daylong load time series. Finally, from these selected daylong load time series, a single time series is sampled as the load scenario.

The expert-based technique (Section 4.3.3) uses domain knowledge to generate load scenarios; it selects yearlong time series of consumers whose attributes are similar to $\mathbf{y}'$ (based on an expert-defined subset $\mathbf{y}_{\text{expert}}$ of all available attributes $\mathbf{y}$) and then selects a daylong time series which adheres to expert-defined conditions based on $\mathbf{d}'$. On the other hand, the data-driven technique (Section 4.3.4) automatically learns which consumer attributes $\mathbf{y}$ and daily attributes $\mathbf{d}$ result in similar consumption patterns and uses this knowledge to generate accurate load scenarios.

4.3.3 The expert-based technique

The expert-based technique uses two types of domain knowledge to generate load scenarios: a set of expert-selected consumer attributes $\mathbf{y}_{\text{expert}}$ (selected from all available attributes $\mathbf{y}$) to measure the similarity between consumers and expert-defined rules to select daylong time series. Concretely, this technique first clusters the set of available yearlong time series based on the expert-selected

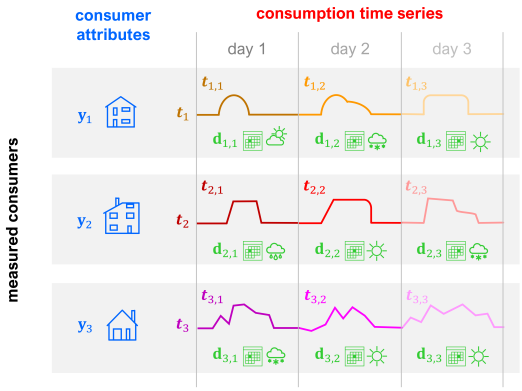


Figure 4.2: A visual representation of the dataset used in this paper (as described in Section 4.4.1). Each measured consumer i has a load consumption time series $\mathbf{t}_i$, consumer attributes $\mathbf{y}_i$ and daily attributes $\mathbf{d}_{i,j}$. The illustration shows 3 measured consumers with 3 days of year; the full dataset contains 1000 measured consumers, each with a full year (365 days) of load measurements.

attributes $\mathbf{y}_{\text{expert}}$ (Section 4.3.3, Figure 4.3a). Then, to generate a load scenario for a given unmeasured consumer with consumer attributes $\mathbf{y}'$ and a given day with daily attributes $\mathbf{d}'$, a single yearlong load time series $\mathbf{t}_{\hat{\imath}}$ is randomly selected from the cluster that is closest to the given consumer attributes $\mathbf{y}'_{\text{expert}}$. From this yearlong load time series $\mathbf{t}_{\hat{\imath}}$, a random daylong load time series $\mathbf{t}_{\hat{\imath},\hat{\jmath}}$ which adheres to the expert-defined conditions based on certain daily attributes in $\mathbf{d}'$ is selected as the scenario (Section 4.3.3, Figure 4.3b).

Clustering

To easily find historical load time series with similar expert-selected consumer attributes $\mathbf{y}_{\text{expert},i}$ to the given expert-selected attributes $\mathbf{y}'_{\text{expert}}$, we cluster the set $\{\mathbf{y}_{\text{expert},i}\}_{i=i,\dots,N_c}$ of consumer attributes (of all measured consumers) into clusters $\{\mathcal{C}_k\}_{k=1,\dots,K_c}$ with corresponding centroids $\{\overline{\mathcal{C}}_k\}_{k=1,\dots,K_c}$. The clustering operation partitions the consumer attributes into groups, called *clusters*, such that attribute vectors in the same cluster are more similar to each other than attribute vectors from distinct clusters. The *centroid* $\overline{\mathcal{C}}_k$ of a cluster $\mathcal{C}_k$ is the element-wise average of all attribute vectors in that cluster. Figure 4.3a depicts a clustering of the expert-selected attributes based on the example dataset shown in Figure 4.2.

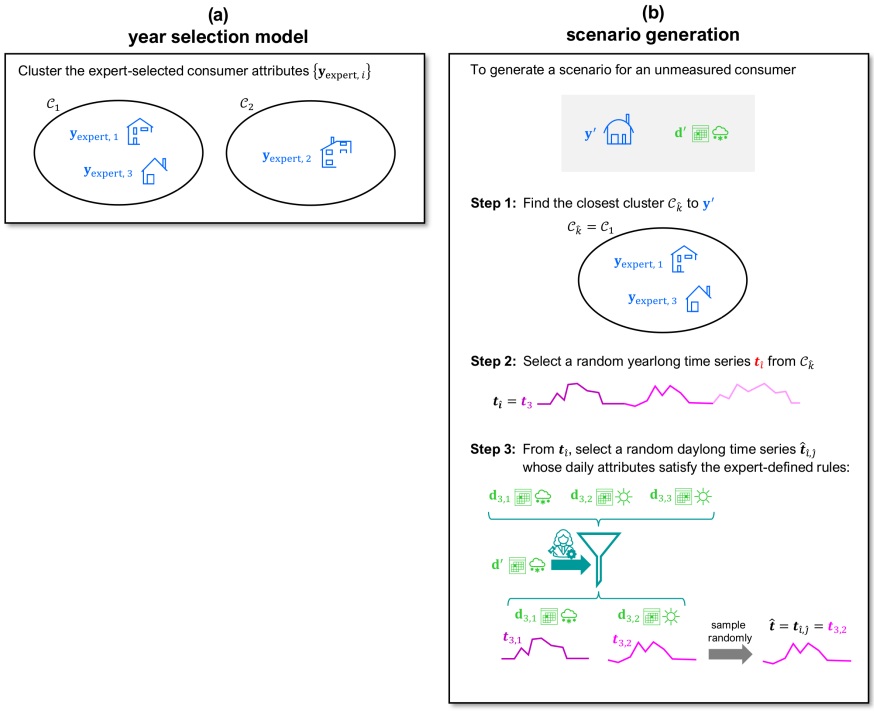


Figure 4.3: Illustration of the expert-based technique using the small dataset shown in Figure 4.2. **(a)** The training phase consists of clustering of the expert-selected consumer attributes (Section 4.3.3). **(b)** The scenario generation process uses the clustering of expert-selected consumer attributes and an expert-defined set of conditions to select a day (Section 4.3.3).

We use the K-*means* algorithm for clustering [85]. As a distance metric, we use the Euclidean distance on attributes normalized into the interval $[0, 1]$, which is common practice [69]. Although the K-means algorithm requires a predetermined hyperparameter K for the number of clusters, we select K automatically based on the *elbow method*. The elbow method executes the K-means algorithm multiple times with increasing K values and selects the smallest K value after which further increasing K does not significantly increase the clustering quality. To select such an 'optimal' K value, we use the Kneedle algorithm [114]. The clustering quality is assessed through inertia, which is the sum of squared distances of attribute vectors to their cluster centroids.

Scenario generation

To generate a single daylong load scenario for a given unmeasured consumer with consumer attributes $\mathbf{y}'$ for a given day with daily attributes $\mathbf{d}'$, the expert-based approach first samples a yearlong time series based on the consumer attributes $\mathbf{y}'$ (using the clustering of consumer attributes $\{\mathcal{C}_k\}_{k=1,\dots,K_c}$) and then, from that selected yearlong time series, it samples a day based on the daily attributes $\mathbf{d}'$ (using the expert-defined conditions). This process is demonstrated in Figure 4.3b.

Sampling a yearlong time series To sample a yearlong load time series based on consumer attributes $\mathbf{y}'$, we find the cluster $\mathcal{C}_{\hat{k}}$ whose centroid $\overline{\mathcal{C}}_{\hat{k}}$ is closest to $\mathbf{y}'_{\text{expert}}$ such that

$$\hat{k} = \arg\min_{k} ||\mathbf{y}'_{\text{expert}} - \overline{\mathcal{C}}_k|| \tag{4.2}$$

where $||\cdot||$ denotes the Euclidean norm. Then, we pick a random yearlong time series $\mathbf{t}_{\hat{\imath}}$ from the closest cluster $\overline{\mathcal{C}}_{\hat{k}}$ such that

$$\hat{\imath} = \textsc{random}(\{i : \mathbf{y}_{\text{expert},i} \in \mathcal{C}_{\hat{k}}\}) \tag{4.3}$$

where the function $\textsc{random}(S)$ uniformly selects a random element from the provided set S.

Sampling a daylong time series To sample a daylong time series from the selected yearlong time series $\mathbf{t}_{\hat{\imath}}$ based on the daily attributes $\mathbf{d}'$, we select a random daylong time series $\mathbf{t}_{\hat{\imath},\hat{\jmath}}$ which is of the same day type (i.e., weekday or weekend) as the given day (as specified in the daily attributes $\mathbf{d}'$) and has a temperature that differs from $\mathbf{d}'$ by at most 2.5°C. The selected daylong time series is used as the scenario: $\hat{\mathbf{t}} = \mathbf{t}_{\hat{\imath},\hat{\jmath}}$. If there are no such days, a random day $\mathbf{t}_{\hat{\imath},\hat{\jmath}}$ is selected that is of the same day type as $\mathbf{d}'$ and occurs in the same month as $\mathbf{d}'$. This procedure can be described as follows:

$$\begin{aligned} J_1 = \big\{ j : \ & \textsc{nonmissing}(\mathbf{t}_{\hat{\imath},j}) \\ & \wedge \ \textsc{isweekend}(\mathbf{d}') = \textsc{isweekend}(\mathbf{d}_{\hat{\imath},j}) \\ & \wedge \ |\textsc{temp}(\mathbf{d}') - \textsc{temp}(\mathbf{d}_{\hat{\imath},j})| \leq 2.5 \ \big\} \end{aligned} \tag{4.4}$$

$$\begin{aligned} J_2 = \big\{ j : \ & \textsc{nonmissing}(\mathbf{t}_{\hat{\imath},j}) \\ & \wedge \textsc{isweekend}(\mathbf{d}') = \textsc{isweekend}(\mathbf{d}_{\hat{\imath},j}) \\ & \wedge \textsc{month}(\mathbf{d}') = \textsc{month}(\mathbf{d}_{\hat{\imath},j}) \big\} \end{aligned} \tag{4.5}$$

$$\hat{\jmath} = \begin{cases} \text{RANDOM}(J_1) & \text{if } J_1 \neq \emptyset \\ \text{RANDOM}(J_2) & \text{otherwise} \end{cases} \tag{4.6}$$

where the function NONMISSING($\mathbf{t}$) returns `True` if and only if all time samples in the time series $\mathbf{t}$ are observed, ISWEEKEND($\mathbf{d}'$) returns `True` only when the day is in the weekend according to its attributes $\mathbf{d}'$, TEMP($\mathbf{d}'$) returns the temperature value specified in $\mathbf{d}'$, and MONTH($\mathbf{d}'$) returns the month specified in $\mathbf{d}'$.

4.3.4 The data-driven technique

Contrary to the expert-based technique, where the domain expert needs to specify how the consumer attributes and daily attributes influence the load of a consumer, the data-driven technique *learns* how the consumer and daily attributes relate to the load of a consumer based on the available historical data. The data-driven technique consists of a two-level data-driven selection process (for yearlong and daylong time series) and a sampling step. Both levels require training: the first level is trained by clustering the yearlong time series and training a machine-learning classifier to select clusters based on given consumer attributes $\mathbf{y}'$ (Section 4.3.4, Figure 4.4a). The second level is trained separately for each first-level cluster, by clustering the daylong time series in that first-level cluster and training a machine-learning classifier to select clusters based on given daily attributes $\mathbf{d}'$ (Section 4.3.4, Figure 4.4b). To generate a load scenario for given consumer attributes $\mathbf{y}'$ and daily attributes $\mathbf{d}'$, the first-level classifier is employed to select a first-level cluster (of yearlong time series). Then, the second-level classifier corresponding to the selected first-level cluster is used to select a second-level cluster (of daylong time series). Finally, from that second-level cluster, a random daylong time series is selected as the scenario (Section 4.3.4, Figure 4.4c).

Training of the first level: consumer selection model

The training phase for the first level consists of *clustering* the yearlong time series $\{\mathbf{t}_i\}_{i=1,\ldots,N_c}$ and *training a probabilistic classifier* that predicts the cluster from given consumer attributes $\mathbf{y}'$. The training of this first level is illustrated in Figure 4.4a.

Clustering Contrary to the expert-based technique, which clusters the attributes $\{\mathbf{y}_i\}_{i=1,\ldots,N_c}$ of the measured consumers, the data-driven technique clusters the yearlong time series $\{\mathbf{t}_i\}_{i=1,\ldots,N_c}$ into clusters $\{\mathcal{C}_k^{\text{year}}\}_{k=1,\ldots,K_{\text{year}}}$.

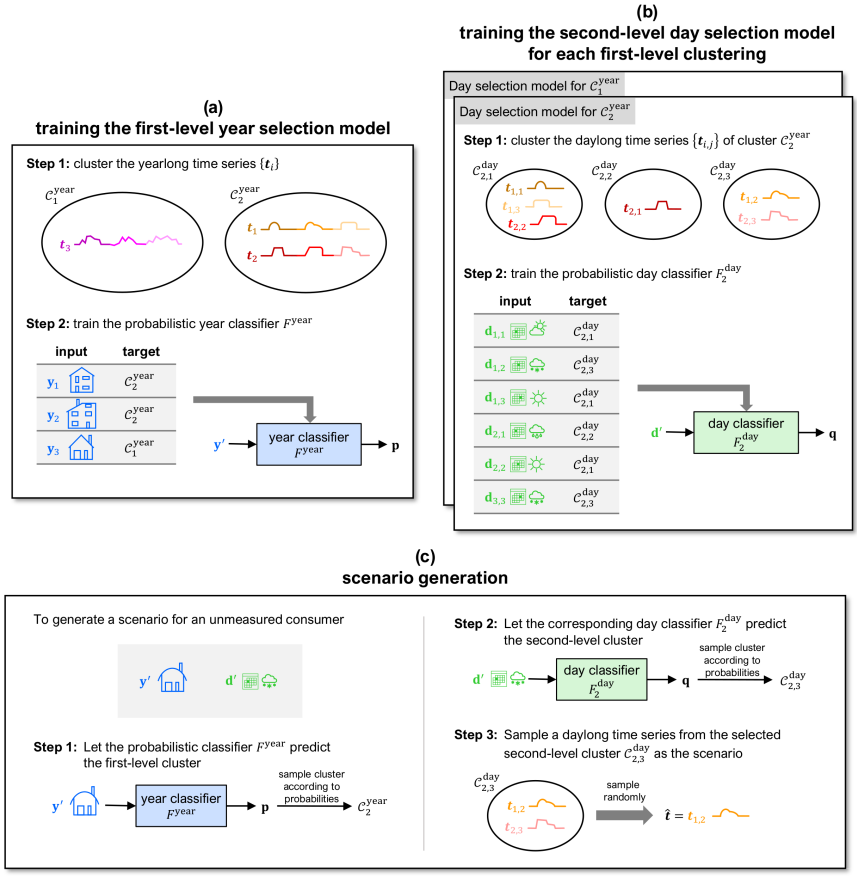


Figure 4.4: Illustration of the proposed data-driven technique using the small dataset shown in Figure 4.2. **(a)** Training the year selection model entails clustering the yearlong time series and learning a probabilistic year classifier (Section 4.3.4). **(b)** For each cluster of yearlong time series, a separate day selection model is trained (Section 4.3.4). **(c)** To generate a scenario both the year and day selection models are used to decide (Section 4.3.4).

This process groups consumers with similar load patterns, which is desired for scenario generation, instead of those with similar attribute values. To perform clustering, we employ the K-medoids algorithm [86, 117][2] with a custom dissimilarity metric that is similar to Euclidean distance but can handle missing data. The custom dissimilarity between two yearlong time series $\mathbf{t}_{i_1}$ and $\mathbf{t}_{i_2}$ with T time samples is defined as

$$\mathrm{d}(\mathbf{t}_{i_1}, \mathbf{t}_{i_2}) = \textsc{mean}\Big(\big\{(\mathbf{t}_{i_1}[t] - \mathbf{t}_{i_2}[t])^2 : \quad \mathbf{t}_{i_1}[t] \text{ and } \mathbf{t}_{i_2}[t] \text{ are not missing,} \\ t = 1, \ldots, T\big\}\Big), \quad (4.7)$$

which excludes time instances at which any of the time series have missing values. We determine the number of clusters K_{year} automatically using the elbow method, as in the clustering step of the expert-based technique (Section 4.3.3).

Training a probabilistic classifier Given an unmeasured consumer with consumer attributes $\mathbf{y}'$, the goal is to select a cluster of yearlong time series with load patterns that are similar to the unmeasured consumer. Previously, in the expert-based technique, the clustering was based on consumer attributes $\mathbf{y}$, making it straightforward to find a cluster related to given attributes $\mathbf{y}'$. Now, the clustering is based on yearlong time series $\mathbf{t}_i$ which do not directly relate to attributes $\mathbf{y}'$. Therefore, the data-driven technique trains a probabilistic classifier F^{year} that predicts, for each cluster $\mathcal{C}_k^{\text{year}}$, the probability p_k that the given consumer with attributes $\mathbf{y}'$ belongs to that cluster:

$$F^{\text{year}} : \mathbf{y}' \mapsto \mathbf{p} = [p_1, \ldots, p_{K_{\text{year}}}] \quad (4.8)$$

where $\sum_{k=1}^{K_{\text{year}}} p_k = 1$. The classifier is trained using the dataset

$$\big\{(\mathbf{y}_i, k) : \quad \mathbf{t}_i \in \mathcal{C}_k^{\text{year}}, \quad i = 1, \ldots, N_c\big\}. \quad (4.9)$$

that contains pairs of attribute vectors $\mathbf{y}_i$ and true class labels k for every measured consumer. This way, the classifier relates the consumer attributes $\mathbf{y}$ with the clusters obtained from yearlong time series data. In this work, we used the random forest algorithm [17] as a probabilistic classifier by using the mean predicted class probabilities of the trees in the forest.

[2] K-medoids has the advantage that its cluster centers are original time series from the dataset. In contrast, K-means' cluster centers are the (element-wise) average of the time series in a cluster which might smooth out relevant consumption patterns and result in nonsensical electricity consumption time series.

Training of the second level: day selection model

For each first-level cluster $\mathcal{C}_k^{\text{year}}$ (of yearlong time series), a second-level day selection model is trained to select daylong time series from that cluster $\mathcal{C}_k^{\text{year}}$. Similar to the consumer selection model, this day selection model is trained by performing a (second-level) *clustering* of the daylong time series and *training a probabilistic classifier* that predicts the (second-level) clusters from given daily attributes $\mathbf{d}'$. The training of the second level is illustrated in Figure 4.4b.

Clustering For every first-level cluster $\mathcal{C}_k^{\text{year}}$ (of yearlong time series), the data-driven technique clusters the daylong time series of the cluster $\mathcal{C}_k^{\text{year}}$ into clusters $\{\mathcal{C}_{k,l}^{\text{day}}\}_{l=1,\ldots,L_k}$. Because it is generally desired that the generated scenarios don't contain any missing values, only the set $\mathcal{D}_k$ of *fully observed* daylong time series in $\mathcal{C}_k^{\text{year}}$ are clustered:

$$\mathcal{D}_k = \Big\{\mathbf{t}_{i,j} : \ \mathbf{t}_i \in \mathcal{C}_k^{\text{year}}, \ \mathbf{t}_{i,j}[t] \text{ is not missing for } t = 1,\ldots,96, \\ j = 1,\ldots,365\Big\}. \quad (4.10)$$

To cluster the fully observed daylong time series $\mathcal{D}_k$, the K-means algorithm [85] with Euclidean distance is used[3]. The number of clusters L_k is selected automatically using the elbow method (separately for each first-level cluster $\mathcal{C}_k^{\text{year}}$), as described in Section 4.3.3.

Training a probabilistic classifier For a given first-level cluster (of yearlong time series), the goal is to select a cluster of daylong time series with similar load patterns as the given day with daily attributes $\mathbf{d}'$. Therefore, a (second-level) probabilistic classifier F_k^{day} is trained for each first-level cluster $\mathcal{C}_k^{\text{year}}$. Each classifier F_k^{day} predicts, for each second-level cluster $\mathcal{C}_{k,l}^{\text{day}}$, the probability q_l that the given day with attributes $\mathbf{d}'$ belongs to that cluster:

$$F_k^{\text{day}} : \mathbf{d}' \mapsto \mathbf{q} = [q_1,\ldots,q_{L_k}] \quad (4.11)$$

where $\sum_{l=1}^{L_k} q_l = 1$. The classifier F_k^{day} is trained using the dataset

$$\big\{(\mathbf{d}_{i,j}, l) : \ \mathbf{t}_{i,j} \in \mathcal{C}_{k,l}^{\text{day}}, \ l = 1,\ldots,L_k\big\} \quad (4.12)$$

[3]Because the number of days to be clustered can be large, K-means (computational complexity linear in the number of instances) is preferred over K-medoids (computational complexity quadratic in the number of instances).

that contains pairs of daily attribute vectors $\mathbf{d}_{i,j}$ and true class labels l for every fully observed daylong time series in $\mathcal{D}_k$. This way, the classifier relates the daily attributes $\mathbf{d}$ with the second-level clusters obtained from daylong time series data. As in the training of the first level, we use the random forest algorithm [17] as a probabilistic classifier by using the mean predicted class probabilities of the trees in the forest.

Scenario generation

To generate a single daylong load scenario for a given unmeasured consumer with consumer attributes $\mathbf{y}'$ for a given day with daily attributes $\mathbf{d}'$, the first-level classifier F^{year} selects a first-level cluster $\mathcal{C}^{\text{year}}_{\hat{k}}$ (of yearlong time series). Then, the corresponding second-level classifier $F^{\text{day}}_{\hat{k}}$ selects a second-level cluster $\mathcal{C}^{\text{day}}_{\hat{k},\hat{l}}$ (of daylong time series). Finally, a random daylong time series $\hat{\mathbf{t}}$ is sampled from the selected second-level cluster $\mathcal{C}^{\text{day}}_{\hat{k},\hat{l}}$. This scenario generation procedure is illustrated in Figure 4.4c.

First level: selecting yearlong candidates Given consumer attributes $\mathbf{y}'$, the probabilistic classifier F^{year} predicts the cluster probabilities $\mathbf{p} = F^{\text{year}}(\mathbf{y}')$. Based on $\mathbf{p}$, a cluster $\mathcal{C}^{\text{year}}_{\hat{k}}$ (of yearlong time series) is selected at random such that

$$\hat{k} = \textsc{random}\left([1, \ldots, K_{\text{year}}],\, \mathbf{p}\right) \tag{4.13}$$

where the function $\textsc{random}(\mathbf{v}, \mathbf{p})$ returns v_n with probability p_n.

Second level: selecting daylong candidates Given daily attributes $\mathbf{d}'$, the probabilistic classifier $F^{\text{day}}_{\hat{k}}$ of the selected first-level cluster $\mathcal{C}^{\text{year}}_{\hat{k}}$ (of yearlong time series) predicts the cluster probabilities $\mathbf{q} = F^{\text{day}}_{\hat{k}}(\mathbf{d}')$. Based on $\mathbf{q}$, a second-level cluster $\mathcal{C}^{\text{day}}_{\hat{k},\hat{l}}$ (of daylong time series) is selected randomly such that

$$\hat{l} = \textsc{random}\left([1, \ldots, L_{\hat{k}}],\, \mathbf{q}\right). \tag{4.14}$$

Random sampling From the selected second-level cluster $\mathcal{C}^{\text{day}}_{\hat{k},\hat{l}}$ (of daylong time series), a daylong time series is randomly selected as the generated scenario

$$\hat{\mathbf{t}} = \textsc{random}\left(\mathcal{C}^{\text{day}}_{\hat{k},\hat{l}}\right). \tag{4.15}$$

4.4 The predictive clustering approach

The predictive clustering approach was developed after the data-driven technique with the aim of simplifying the model to obtain a computationally efficient and interpretable model. The data-driven technique clusters the historical time series and learns a probabilistic classifier that learns to predict which cluster a time series belongs to based on its associated attributes. The drawback of this two-step approach is that clustering and the subsequent classification are performed independently. Therefore, the clustering might not be aligned with the available attributes, making it difficult for the probabilistic classifier to assign attribute vectors to clusters. This simple approach would entail clustering all historical daylong time series, which is computationally expensive. Hence, the data-driven approach applies the clustering and classification procedure twice in a two level-approach. In the first level, it uses the household attributes to select a cluster of yearlong time series and then, in the second level, within the selected cluster of yearlong time series, the daily attributes are used to select a cluster of daylong time series to sample from.

The predictive clustering approach is similar to the data-driven technique as it also samples scenarios from a set of historical time series; however, it drastically simplifies the procedure to determine which time series to sample from. Instead of first clustering the time series and then learning a classifier, our proposed technique uses a predictive clustering tree to cluster the time series and, at the same time, learn a description of each cluster based on the available attributes. This guarantees that the clustering is aligned with the attributes. Moreover, because predictive clustering scales linearly in the number of instances, there is no need for a two-level procedure. We learn a single decision tree that uses both the daily and consumer attributes to select daylong time series directly, resulting in a simple, fast, and interpretable model.

4.4.1 Problem Statement and notation

Because the predictive clustering approach is much simpler than the data-driven method, we also explain it using a simplified notation where both the daily and consumer attributes are denoted as a single vector $\mathbf{a}_i$ and $\mathbf{t}_i$ refers directly to a daylong time series (as there is no need for a yearly level anymore).

Given: A dataset $\mathcal{D}$ that contains N historical daylong electricity consumption time series $\mathbf{t}_i$ each associated with an attribute vector $\mathbf{a}_i$:

$$\mathcal{D} = \{(\mathbf{a}_i, \mathbf{t}_i) \ : \ i = 1, \ldots, N\}. \tag{4.16}$$

The attribute vector $\mathbf{a}_i$ contains all attributes that are available for the scenario generation task such as consumer attributes (e.g., yearly consumption), weather attributes (e.g., temperature), and calendar info (e.g., day of week).

Do: Generate N_S load scenarios $\{\hat{\mathbf{t}}^{(s)} : s = 1, \ldots, N_s\}$, i.e., daylong electricity consumption time series, for a specific attribute vector $\mathbf{a}'$, where N_s is the number of scenarios to be generated which is specified in advance.

4.4.2 Generating scenarios with predictive clustering

Our methodology uses a predictive clustering tree (PCT), a machine-learning model that combines elements from both clustering and prediction [12, 38]. A PCT is a decision tree where splits are based on the attributes in $\mathbf{a}_i$ and each leaf node represents a cluster of time series $\mathbf{t}_i$. A PCT can be learned with standard greedy top-down decision tree induction where splits on $\mathbf{a}_i$ are chosen to minimize the intra-cluster variance of the time series $\mathbf{t}_i$ in each child node (Section 4.4.2). Minimizing the intra-cluster variance leads to a clustering where time series with similar consumption patterns end up in the same leaf (or cluster) and dissimilar time series end up in different clusters. The learned decision tree allows us to predict the cluster of an arbitrary attribute vector $\mathbf{a}'$. Because clustering and building the prediction model happen simultaneously, the clusters are constructed in such a way that they are separable in the space of all attributes and are, therefore, well-suited for prediction.

To generate scenarios for a given attribute vector $\mathbf{a}'$, the attribute vector $\mathbf{a}'$ is assigned to one of the clusters by traversing the tree from root to leaf following the edges that are satisfied for $\mathbf{a}'$; the scenarios are then sampled randomly from the selected cluster (Section 4.4.2).

Building a predictive clustering tree

Building a predictive clustering tree is similar to standard greedy top-down decision tree induction used in a number of algorithms such as C4.5 [105]. Starting with the root node that contains all instances, nodes are recursively split until no suitable split can be found or some stopping criterion is reached (e.g., maximum tree depth). Pseudocode for this procedure can be found in Algorithm 3 where lines 1–9 show the recursive splitting procedure and line 16 the stopping criterion.

To find the best way to split a node D, the tree induction algorithm exhaustively searches for the splitting condition on an attribute in $\mathbf{a}$ to minimize the variance

of the time series within each resulting child node. More specifically, the tree induction algorithm chooses the split that results in the highest variance reduction. The variance reduction h for splitting a node D_P into the partition $\mathcal{P}$ that contains child nodes $\{D_j\}$ is

$$h(D_P, \mathcal{P}) = \text{var}(D_P) - \sum_{D_j \in \mathcal{P}} \frac{|D_j|}{|D_P|} \text{var}(D_j) \tag{4.17}$$

where $\text{var}(\cdot)$ denotes the variance of a node and $|\cdot|$ the number of instances in a node. A negative variance reduction corresponds to a split that increases the variance, and hence, is not considered as a valid split. Pseudocode for the splitting procedure can be found in Algorithm 3, lines 11–18.

Given an arbitrary distance metric $\text{dist}(\cdot)$ between time series, the variance of a node D can be computed in two ways [38]:

- in $O(N^2)$ distance computations, using the sum of squared pairwise distances:

$$\text{var}(D) = \frac{1}{|D|^2} \sum_{(\mathbf{a}_i, \mathbf{t}_i) \in D} \sum_{(\mathbf{a}_j, \mathbf{t}_j) \in D} \text{dist}(\mathbf{t}_i, \mathbf{t}_j)^2 \tag{4.18}$$

- or, in $O(N)$ distance computations, if the centroid of a node can be efficiently computed, as the average squared distance to the centroid $\overline{D}$:

$$\text{var}(D) = \frac{1}{|D|} \sum_{(\mathbf{a}_i, \mathbf{t}_i) \in D} \text{dist}(\mathbf{t}_i, \overline{D})^2$$

$$\text{with } \overline{D} = \arg\min_{\mathbf{t}} \sum_{(\mathbf{a}_i, \mathbf{t}_i) \in D} \text{dist}(\mathbf{t}, \mathbf{t}_i)^2. \tag{4.19}$$

In this paper, we use the Euclidean distance for scalability, as most datasets in this field contain $\approx 10^6$ time series (Section 4.5.1). Under the Euclidean distance, the centroid $\overline{D}$ of a node can be computed in linear time using:

$$\overline{D} = \frac{1}{|D|} \sum_{(\mathbf{a}_i, \mathbf{t}_i) \in D} \mathbf{t}_i. \tag{4.20}$$

Therefore, using Equation 4.19, the variance can be computed in linear time. On the other hand, for many time-series-specific distance metrics, no closed-form expression for the centroid $\overline{D}$ is known and one would have to resort to (iterative) approximate algorithms or to using equation 4.18 to calculate

the variance in quadratic time. For example, for the dynamic time warping distance[113] between time series the centroid needs to be approximated using dynamic time warping barycenter averaging, which is not guaranteed to converge to the global optimum[102]. However, if these scaling issues can be overcome, time-series-specific distance metrics might bring performance improvements. For example, the dynamic time warping distance would be able to ignore small differences in timing of a certain consumption pattern grouping consumers that behave similarly but not exactly at the time in the same cluster.

We only consider numerical attributes, therefore, all splits are binary splits of the form $attribute < threshold$ where the attribute is one of the attributes in $\mathbf{a}$ and $threshold \in \mathbb{R}$.

To avoid overfitting, we first learn a tree using the majority of the available data $\mathcal{D}$, constraining the tree by setting the maximum depth and the minimum samples per leaf. Afterwards, we prune this tree using the remainder of the dataset $\mathcal{D}$ (the so-called pruning set) with reduced error pruning[106], which removes splits from the tree that do not reduce the error, in our case, variance, on the pruning set.

Scenario generation

After learning a PCT, generating scenarios for a certain attribute vector $\mathbf{a}'$ is straightforward: First, assign the attribute vector to one of the leaves in the tree by traversing the tree and, at each split, selecting the child node whose condition is satisfied for $\mathbf{a}'$. Second, as this leaf corresponds to a cluster of time series $\{\mathbf{t}_i\}$, randomly sample N_s scenarios and return these as generated scenarios $\hat{\mathbf{t}}^{(s)}$ for $s = 1, \cdots, N_s$.

An ensemble of predictive clustering trees

Similar to Cevid et al. [25], these predictive clustering trees could also be used in an ensemble. In our experiments for scenario generation of unmeasured consumers, an ensemble of PCTs offers a small performance benefit. However, there is a trade-off, an ensemble of trees offers less interpretability than a single tree. Some examples of techniques to offer (partial) interpretability for ensembles of trees are feature importances [17] or a simplified surrogate model [59]. Because we consider interpretability very important and because the performance improvement was only small, we chose to only learn a single tree.

Algorithm 3: PCT induction algorithm

```
function build_tree(D)
    split, P* ← best_split(D)
    if split ≠ None then
        subtrees ← []
        for D_i ∈ P do
            subtrees[i] ← build_tree(D_i)
        return node(split, subtrees)
    else
        return leaf(D)

function best_split(D)
    h*, s*, P* ← 0, None, None
    for each possible split s do
        P ← partition induced by split s on D
        h ← var(D) − Σ_{D_i∈P} (|D_i|/|D|) var(D_i)
            /* Allowed checks whether the split is allowed according to the tree building
            constraints (e.g., maximum depth) */
        if h > h* ∧ allowed(test, P) then
            h*, s*, P* ← h, s, P
    return s*, P*
```

4.5 Experimental methodology

In this section, we describe the details of our experiments: the three datasets we used (Section 4.5.1), the cross-validation scheme (Section 4.5.2), the implemented scenario generation techniques (Section 4.5.4), and the performance metric (Section 4.5.3).

4.5.1 Datasets

We use three datasets of load time series acquired from three different locations: Flanders (a region in Belgium), Ireland, and London. The datasets from Ireland and London are publicly available (upon request)[29, 138]. Every dataset is enriched with publicly available weather data and calendar information (Section 4.5.1). We consider the Flanders dataset as the most realistic dataset for our problem setting because the available attributes are in fact known by the

Table 4.1: Characteristics of the used datasets (detailed descriptions are provided in Section 4.5.1)

	Flanders	Ireland	London
# yearlong time series	4363	3488	2621
# consumers	2200	3488	2476
Sampling period	15 min	30 min	30 min
Missingness	0.30%	0.05%	0.28%
Measurement years	2010-2017	2010	2012-2013
Availability	private	on request [29]	on request [138]

DSO, whereas the Ireland and London datasets contain more detailed attributes that are difficult to collect on a large scale. However, these publicly available datasets do facilitate comparison with other methods and make our results reproducible.

We learn and evaluate the models directly on the unnormalized data as we are not only interested in the shape of the generated scenarios but also their scale. The data-driven and expert-ased approach proposed by Soenen et al. [126] require full yearlong time series for every consumer and cannot handle partially recorded years. To ensure compatibility and a fair comparison, every dataset is preprocessed to retain only full yearlong time series. To ensure reproducibility, the code of the proposed method, the dataset details and all dataset preprocessing code for the two open datasets are available online[4]. An overview of the dataset characteristics, after preprocessing, is given in Table 4.1. In what follows, we describe each dataset in detail.

Flanders dataset

The dataset from Flanders is acquired from residential consumers and businesses connected to the LVG by Fluvius, the DSO in Flanders. The dataset consists of 4363 yearlong time series sampled every 15 min that belong to 2200 unique consumers between 2010 and 2017. The available attributes are yearly consumption (in kWh); connection capacity (in kVA), i.e., the maximum power that a consumer can draw from the grid; whether or not PV panels are installed; and PV capacity (in kVA), i.e., the maximum power that the installed PV

[4] `https://github.com/jonassoenen/predclus_scengen`

panels can produce. Because of privacy reasons, this dataset cannot be made publicly available.

Ireland dataset

The publicly available Ireland dataset originates from the Smart Metering Electricity Consumer Behaviour Trials executed by the Commission for Energy Regulation (CER)[29]. The dataset contains 30-min load measurements from over 5000 Irish homes and businesses from July 2009 until December 2010 with detailed survey data for every consumer. Because the survey data of homes and businesses are different and cannot be consolidated into a single attribute set, we focused on the 3488 residential consumers in the dataset. The survey data is filtered and preprocessed into 33 attributes relating to the inhabitants, building, home heating type, water heating type, cooking energy source, and yearly electricity consumption (Table 4.2).

London dataset

The publicly available London dataset originates from the Dynamic Time-of-Use Electricity Pricing trial part of the Low Carbon London Project [138]. The dataset contains 30-minute load measurements from November 2011 until February 2014 complemented with survey data of every consumer. Discarding incompletely measured years results in 2621 yearlong time series from 2476 unique consumers where all consumers have data in 2013 and a few consumers have data for both 2012 and 2013. The survey data is filtered and preprocessed into 75 attributes relating to the inhabitants, building, insulation, home heating type, water heating type, appliances, and yearly electricity consumption (Table 4.3).

Data enrichment with weather and calendar attributes

Every dataset is enriched with 8 weather attributes and 7 calendar attributes. The weather attributes are: minimum, maximum and average temperature during the day; average feels-like temperature during the day (all temperatures are in °C); sun hour, i.e., a measure for the amount of solar radiation; the UV index; concentration of water vapor in the air (in %); and wind speed (in km/h). The historical weather attributes were obtained from WorldWeatherOnline[5]. Because the location of the individual consumers is unknown for the Flanders

[5] `https://www.worldweatheronline.com/`

Table 4.2: Available consumer attributes in the Ireland dataset

Attribute category	Attributes, possible values and encoding
Inhabitants	Respondent age, number of adults, and number of children.
Building	Build year, home age, total floor area, number of bedrooms, and home type (apartment, bungalow, detached, semi-detached, terraced, or unknown).
Energy source	Heating type (central electrical, plugin electrical, gas, oil, solid fuel, renewable, or other), water heating type (central electrical, electric immersion, electric instance, gas, oil, solid fuel, renewable, or other), and cooking energy source (electric, gas, oil, or solid fuel).
Consumption	Yearly consumption (in kWh).

and Ireland datasets, we used weather information from Brussels and Dublin, respectively. For the London dataset, we simply used weather information from London.

The calendar attributes are day of week, day of month, day of year, month, season, whether or not it is weekend, and whether or not it is a holiday. These attributes are derived from the date itself except for the holiday information which is obtained with the "holidays" Python package[6].

4.5.2 Cross validation

To measure the performance of each scenario generation technique, we apply 5-fold cross-validation. For each dataset, the unique consumers are randomly partitioned into 5 approximately equal-sized folds[7]. The data from four of these folds are used as a training set while the remaining fold is used as a test set. Models are trained based on the tuples $\{(\mathbf{a}_i, \mathbf{t}_i)\}$ in the training set. For every tuple $(\mathbf{a}', \mathbf{t})$ in the test set, the trained model generated N_s scenarios $\{\hat{\mathbf{t}}^{(s)}\}$ based on $\mathbf{a}'$. These scenarios are then compared with the ground truth $\mathbf{t}$ using the energy score (explained in Section 4.5.3). This process is repeated five times such that each fold is used for testing exactly once.

[6] `https://pypi.org/project/holidays/`

[7] This ensures that consumers whose load is being predicted are not included in the training set (avoiding leakage [73]).

Table 4.3: Available consumer attributes in the London dataset

Attribute category	Attributes, possible values and encoding
Inhabitants	Number of inhabitants, number of males, number of females, mean inhabitant age, min inhabitant age, max inhabitant age, work from home (never, occasionally, about half the time, or most/all weekdays; ordinal encoding), home ownership (owner, rent, free rent, shared ownership, other or missing; one-hot encoding).
Building	Building type (mobile, flat, house-terraced, house-semi-detached, or house-detached; ordinal encoding), number of rooms, number of bedrooms.
Insulation	Double glazing, roof insulation, wall insulation, floor insulation, and warm water tank insulation. For all insulation attributes the possible values are yes, no, don't know, or missing with an ordinal encoding.
Heating	Heating type (gas, electric, solid fuel, oil, other), heating control (manual boiler, manual valves, automatic temperature, automatic timed), and water heating (electric with storage, missing, gas, or gas with storage; ordinal encoding).
Appliances	Information on whether the inhabitants own various appliances such as fridges, freezers, cooking equipment, laundry machines, computers, screens, etc...
Consumption	Yearly consumption (in kWh).

4.5.3 Performance metric

Evaluating the quality of the generated scenarios is not straightforward. If exactly the same circumstances $\mathbf{a}'$ would occur multiple times, we still expect the electricity consumption time series of a specific consumer to vary considerably due to the inherent stochasticity of human behavior and unobserved variables. In our problem setting, it is important that the generated scenarios based on $\mathbf{a}'$ reflect this variance. If a standard performance metric such as mean squared error were used to measure the quality of the generated scenarios, consistently predicting the expected value would be optimal; but this would underestimate the possible variation, which is undesired in our problem setting. Furthermore, since electricity consumption time series typically have many peaks, evaluation using the mean squared error suffers from the double peak penalty effect [58]. Therefore, the mean squared error is not a suitable evaluation metric.

The Energy Score (ES) [50] is a suitable measure to assess the quality of the generated scenarios [90]. The ES [50] is a multivariate generalization of the conditional ranked probability score (CRPS) [87] which assesses the quality of a multivariate probabilistic forecast. The ES is a proper score [19] that assesses both the calibration and sharpness of the predicted multivariate distribution [50]. In the case of scenario generation, where a set $\left\{\hat{\mathbf{t}}^{(s)}\right\}_{s=1,\dots,N_s}$ of equally likely scenarios is predicted for a given test sample $\mathbf{t}$, the ES can be calculated as

$$\mathrm{ES}\left(\left\{\hat{\mathbf{t}}^{(s)}\right\}_{s=1,\dots,N_s}, \mathbf{t}\right) = \frac{1}{N_s}\sum_{s=1}^{N_s}\left\|\hat{\mathbf{t}}^{(s)} - \mathbf{t}\right\| - \frac{1}{2N_s^2}\sum_{s=1}^{N_s}\sum_{r=1}^{N_s}\left\|\hat{\mathbf{t}}^{(s)} - \hat{\mathbf{t}}^{(r)}\right\| \quad (4.21)$$

where $||\cdot||$ denotes the Euclidean norm. We report the average ES over all the daylong time series in the test set, where a lower ES means more accurate scenarios.

4.5.4 Scenario Generation Methods

In our experiments, we evaluate four scenario generation techniques:

- *Random sampling (baseline).* Random sampling samples scenarios randomly from all daylong time series in the training set disregarding the attributes.

- *The expert-based technique* as explained in Section 4.3.3. The consumers are clustered based on the normalized yearly consumption and connection capacity. The number of clusters is automatically selected from {5, 10, ..., 100} by the elbow method[114].
- *The data-driven technique* as explained in Section 4.3.4. The number of clusters is automatically selected from {5, 10, ..., 100} with the elbow method[114] using the Random Forest Classifier[17] from scikit-learn[100] with default hyperparameters.
- *The proposed predictive clustering approach* as explained in Section 4.4. 75% of the training set is used to build the PCT with a maximum depth of 12 and minimum 300 samples per leaf. The remaining 25% is used as a pruning set (to prune the PCT). We learn PCTs using the Clus software package[8].

For each daylong time series in the test set, each technique generates 250 daylong load scenarios ($N_s = 250$). The experiments were run on a linux server with 128 GB of RAM and 2 Intel(R) Xeon(R) Silver 4214 CPUs with 12 cores and 24 threads each. Part of the data-driven and expert-based method are multi-threaded giving them a significant advantage over the proposed approach when measuring runtimes.

4.6 Experimental results

Our experiments consist of three parts: a general comparison between the expert-based technique, the data-driven technique and predictive clustering approach (Section 4.6.1); an experiment to study the influence of the dataset size on the proposed techniques (Section 4.6.2); and a use case study on the Flanders dataset to illustrate the interpretability of the predictive clustering approach (Section 4.6.3). For a more detailed analysis of the data-driven technique (e.g., the influence of the number of clusters and an ablation study), we refer to the original paper [126].

4.6.1 Scenario generation performance

We compare the scenario generation performance of the predictive clustering approach to the expert-based and data-driven approaches[126], and to a baseline

[8] `https://dtai.cs.kuleuven.be/clus/`

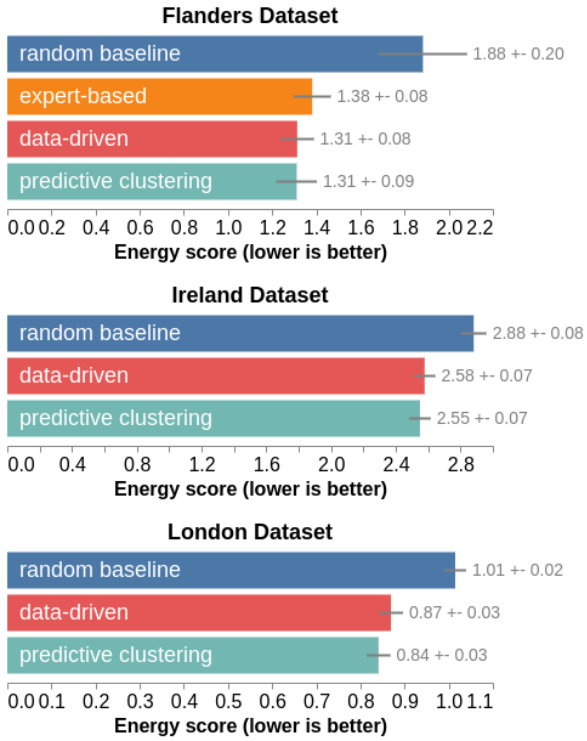


Figure 4.5: The proposed predictive clustering technique performs on par with the existing data-driven approach, and, on the London dataset, slightly outperforms it. The bars show the average ES over all folds, the gray lines shows the standard deviation of ES over the folds.

of random sampling. The comparison is performed based on the Flanders, London, and Irish datasets using the average ES over all cross-validation folds as the performance metric. Because the expert-based approach requires the connection capacity as an attribute, it can only be executed on the Flanders dataset. To compare the execution time of the methods, we also measured the training time (the time it took to train the model on the training set) and the scenario generation time (the total time it took to generate the scenarios for the test set).

Table 4.4: For every dataset, the average training time and prediction (i.e., scenario generation) time over every fold.

		Runtime (in sec)	
Dataset	**Method**	Training	Prediction
Flanders	data-driven	6388	7505
	expert-based	70	4087
	predictive clustering	454	96
Ireland	data-driven	3707	24372
	predictive clustering	340	77
London	data-driven	2966	19718
	predictive clustering	407	58

Results

All proposed techniques generate scenarios that are significantly better than the random sampling baseline (Figure 4.5). The data-driven technique generates scenarios that are better than the expert-based technique without requiring any domain knowledge on the Flanders dataset. However, the predictive clustering approach generates scenarios that are as accurate as the data-driven technique, and, on the London dataset, even slightly more accurate than the data-driven technique.

In terms of training time, the predictive clustering approach is on average 10 times faster than the data-driven method, but significantly slower than the expert-based method (Table 4.4). In terms of scenario generation time, predictive clustering is by far the fastest method. Prediction is very slow for the data-driven method (taking up to 6 hours for the Ireland dataset) as it needs to make multiple predictions with a random forest.

Conclusion

In terms of energy score, our (simple) predictive clustering approach performs on par with, or slightly better than, the complex data-driven approach while being significantly faster in both training and scenario generation time.

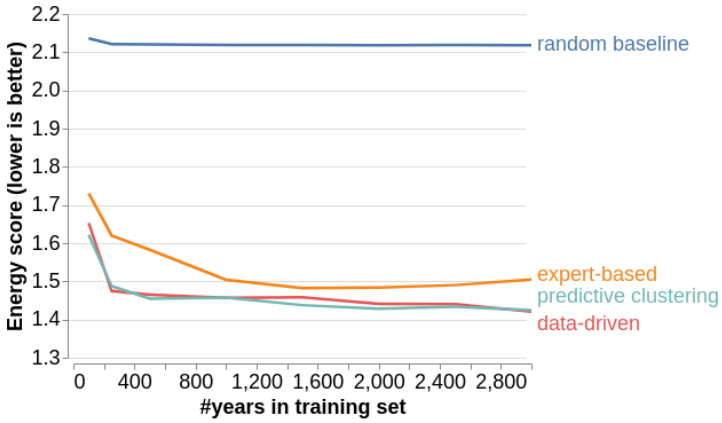


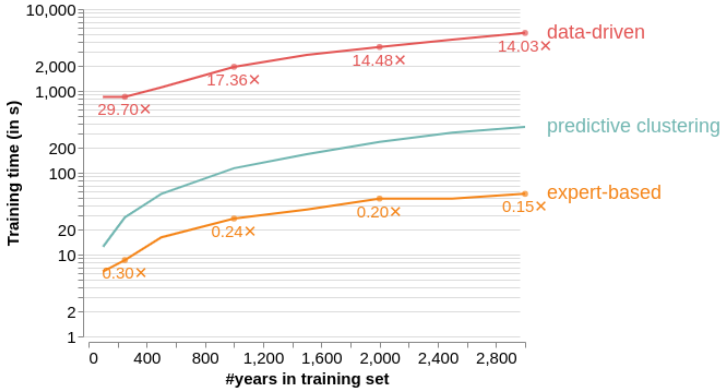
Figure 4.6: Evolution of the energy score as a function of training set size for every scenario generation method.

4.6.2 Influence of dataset size

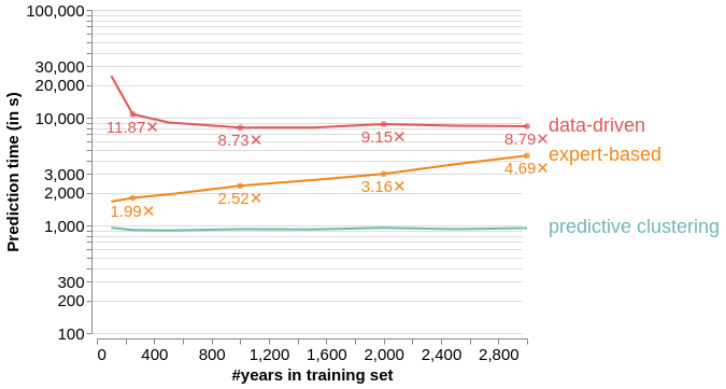
In this experiment, we study the influence of the dataset size on the performance and the execution time of every method. To achieve this, a test set of 500 yearlong time series is randomly sampled from the Flanders dataset. Next, multiple training sets of sizes [100, 250, 500, 1000, 1500, 2000, 2500, 3000] are sampled from the full dataset, ensuring each smaller dataset is a subset of the next (bigger) dataset. Then, every method is trained with every training set and evaluated on the fixed test set with the energy score. As before, we also measure the training and scenario generation time.

Results

In terms of ES, both predictive clustering and data-driven techniques keep producing better and better scenarios as the number of yearly profiles in the training set grows (Figure 4.6). However, most of the ES reduction happens between 100 and 500 years in the training set. The expert-based method keeps improving up to 1500 years but never reaches the same ES as the other two techniques. The random baseline does not benefit from a bigger training set; its ES is almost constant around 2.11. In terms of training time, predictive clustering is more than 10 times faster than the data-driven method but slower than the expert-based method (Figure 4.7a). To generate predictions for 500×365 days, predictive clustering is the fastest method regardless of the training set size (Figure 4.7b).



(a) Time to train each model



(b) Time to generate scenarios for test set

Figure 4.7: Training and prediction times as a function of training set size for every method on the Flanders dataset. For the data-driven and expert-based method, the relative execution time compared to predictive clustering is shown as well (e.g. for 250 years in the training set, data-driven needs 29.7 times more time to train a model than the proposed approach).

Conclusion

Both predictive clustering and the data-driven method benefit from larger training sets, albeit with diminishing returns.

4.6.3 Interpretability

To illustrate the interpretability of the predictive clustering model, we show a PCT trained on the Flanders dataset in Figure 4.8. The tree itself is learned with a maximum depth of 7 to ensure that the full tree fits on a single page. The tree is also compressed such that multiple consecutive binary splits on the same attribute become a single n-ary split. The compressed tree is equivalent to the original tree but is easier to interpret. Moreover, because each node in the tree corresponds to a cluster of time series, it is easy to see how a certain split influences the predictions of the model.

From this tree (Figure 4.8), it is clear that yearly consumption is by far the most used attribute in the tree. This is unsurprising as yearly consumption provides direct information about the absolute consumption of the profile. The other used attributes are the PV capacity, sun hour, temperature, day of the week, and connection power. Not all of these are used in the same subtrees: sun hour is only used to make splits for profiles that have PV panels reflecting that if you have PV panels, the sun hour attribute has a big influence on the consumption time series. On the other hand, connection power is a feature that is only used for consumers with high yearly consumption. Splitting on connection power might help to distinguish residential customers from small and medium enterprises.

Not only can this tree be inspected to see how the predictions are made, but by looking at all the time series in a leaf (or any internal node), one can also inspect what the model predicts in a certain branch of the tree or what influence a certain split has on the predictions. To visualize the time series in a node, we calculate the $0.05, 0.1, \ldots, 0.90, 0.95$ quantiles and show those to give an idea of the distribution of the consumption values. In Figure 4.9, we show the influence of the split on sun hour for consumers with a yearly consumption ≤ 4285 kWh that have a PV panel installation with a power > 4 kVA. In Figure 4.10, we show the influence of the split on feelsLikeC temperature for consumers with a yearly consumption between 9907 and 21798 kWh. From the time series, it seems as if the consumers with a yearly consumption in this range are using accumulation heating with a tariff scheme that favors night consumption. When it is cold, accumulation heating draws power during the night in order to slowly release the created heat during the day. The PCT has learned that if feelsLikeC is lower than 5, significantly more electricity is needed for heating.

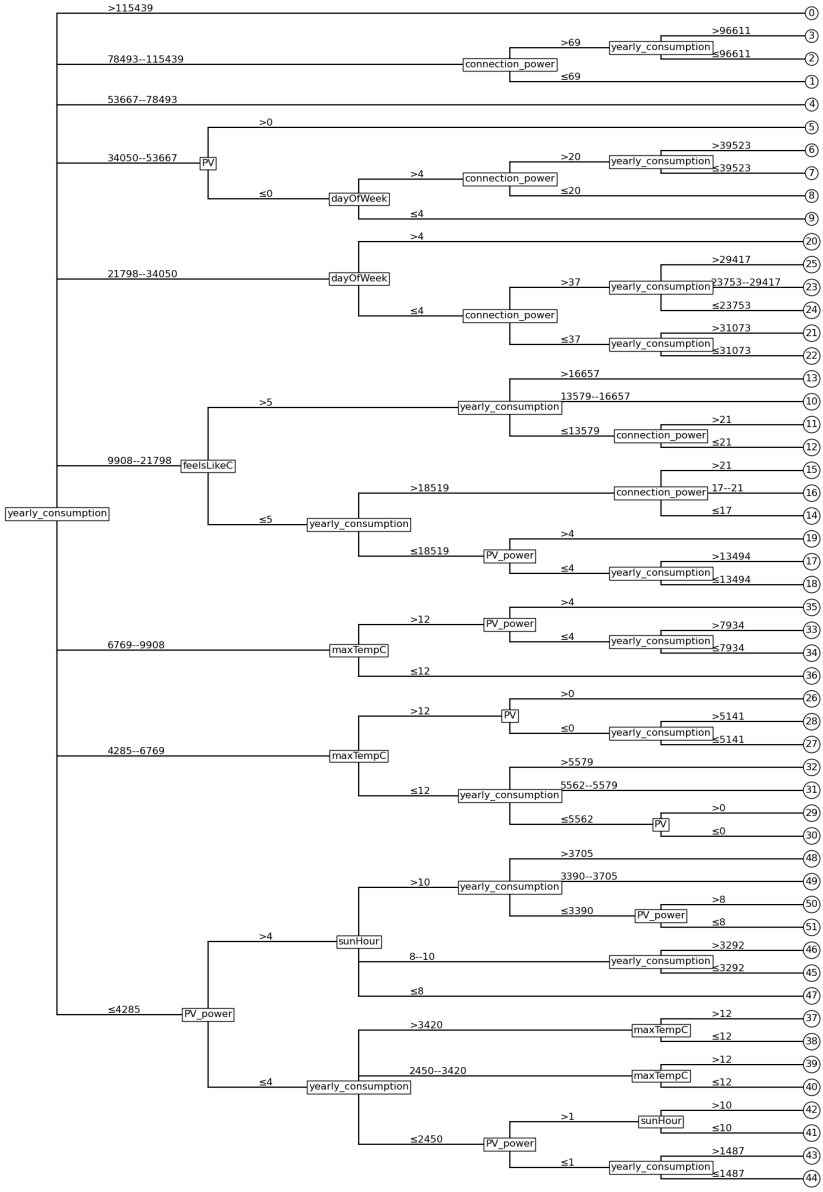


Figure 4.8: The tree learned with predictive clustering on the Flanders dataset

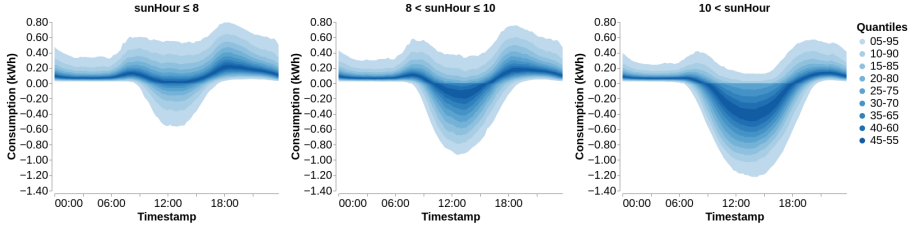


Figure 4.9: Visualization of the influence of the split on sunHour for consumers with a yearly consumption $\leq$ 4285 kWh and PV power $>$ 4 kVA for the Flanders dataset. Quantiles of all time series in the cluster are shown.

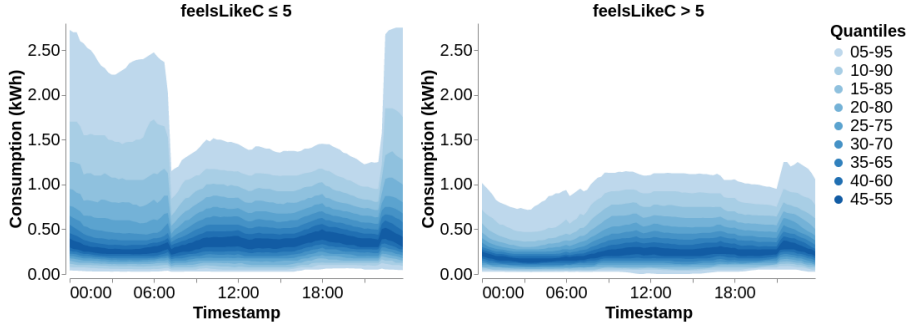


Figure 4.10: Visualization of the influence of the split on feelsLikeC for consumers with a yearly consumption in between 9908 kWh and 21798 kWh for the Flanders dataset. Quantiles of all time series in the cluster are shown.

4.7 Conclusion

In this chapter, we have proposed three techniques to generate realistic and accurate electricity consumption scenarios for unmeasured consumers based on the available consumer attributes and circumstantial attributes (such as weather and calendar information). The first, expert-based, technique relies on domain knowledge to sample scenarios. The second, data-driven, technique employs machine learning to learn how the attributes can be used to accurately sample scenarios. The third technique, the predictive clustering technique, also uses machine learning techniques to learn how to sample but is significantly simpler than the data-driven approach. Surprisingly, in our experiments, predictive clustering trees slightly outperforms the data-driven technique while being significantly faster. Moreover, the predictive clustering tree can be easily visualized such that a domain expert can observe which predictions are made

in which circumstances, enabling the expert to gain new insights and trust the scenarios generated by the model. Our predictive clustering technique might help distribution system operators to model the unmeasured consumers more accurately, resulting in more accurate low-voltage grid studies.

Future work

While we have proven that we can generate more accurate scenarios using the energy score, it remains an open question whether these more accurate scenarios also have a positive effect on the grid simulation methods, e.g., a power flow calculation. Such a study is planned but could unfortunately not take place during my PhD.

Acknowledgments

We would like to thank the Flemish DSO Fluvius for supporting the work described in this chapter.

Chapter 5

Evaluating unsupervised anomaly detection algorithms

In this chapter, we review the two common methodologies to tune the hyperparameters in order to compare the anomaly detection performance of different anomaly detection algorithms over a set of benchmark datasets. We argue that both existing methodologies have significant drawbacks and therefore propose to use a validation set to tune the hyperparameters for a fair, reproducible, and sound comparison.

The content of this chapter is based on the following publication [128]:

J. Soenen*, E. Van Wolputte*, L. Perini, V. Vercruyssen, W. Meert, J. Davis, and H. Blockeel. "The effect of hyperparameter tuning on the comparative evaluation of unsupervised anomaly detection methods". In: *Proceedings of the KDD'21 Workshop on Outlier Detection and Description*. 2021, pp. 1–9

5.1 Introduction

While all unsupervised anomaly detection (AD) algorithms have hyperparameters that can greatly affect performance, deciding how to set them is challenging. The goal is to find hyperparameters that yield good performance. However,

*Both authors contributed equally to this work

measuring performance requires having labeled data, which is assumed not to be available in an unsupervised setting. In fact, the standard motivation for treating anomaly detection as an unsupervised problem is that acquiring labeled data is often difficult, if not infeasible, in practice.

In the literature, researchers tend to cope with this problem in two different ways. The conservative point of view is to perform no tuning and simply use the same hyperparameter configuration on each dataset [2, 27, 30, 34, 40, 42, 51, 54, 57, 68, 76, 81, 94, 97, 99, 103, 111, 115, 121, 131, 134, 140–142, 147, 153, 156]. This can be thought of as adhering to "reasonable defaults." The opposite point of view is to report results for the hyperparameters that maximize the performance of the detector on each dataset [22, 26, 46, 51, 61, 65, 68, 70, 83, 118, 162]. This is akin to assuming that one has access to an oracle that can always provide the optimal hyperparameters for a dataset.

Unfortunately, both approaches have significant drawbacks. First, neither approach is likely appropriate to answer the standard question asked when empirically evaluating a new anomaly detection method: is the new algorithm useful in practice, i.e., are there datasets on which it outperforms the existing anomaly detection algorithms? On the one hand, using defaults likely provides a pessimistic estimate of performance and disadvantages algorithms whose hyperparameters have a large influence on performance. On the other hand, reporting results that maximize performance literally is "tuning on the test set" which is methodologically unsound. Second, both approaches are likely not indicative of what would happen in practice. Practitioners will likely want to tune the parameters in some way instead of falling back on defaults and don't have an oracle that provides them with the optimal hyperparameters.

In this chapter, we argue that anomaly detection should follow standard ML practice: use a labeled validation set to select appropriate hyperparameters. This offers a principled way to choose reasonable hyperparameter values that enables a fair, reproducible and relevant comparison between algorithms (relevant in the sense that in many practical use cases, it is realistic that the same tuning method can be used). We propose a concrete evaluation strategy, argue for its relevance and conduct an empirical evaluation. First, we demonstrate that existing strategies can lead to substantially different conclusions. Second, as our proposal relies on a validation set which is preferably as small as possible, we also investigate the effect of the size of the validation set on our evaluation strategy. These experiments show that our strategy is feasible in practice, as the majority of AD-methods can be tuned on a relatively small validation set.

5.2 Unsupervised Anomaly detection

Viewed from a learning perspective, an unsupervised anomaly detection method is an algorithm that analyses a dataset $D \subset \mathcal{X}$ and returns either a prediction of which instances in D are anomalous (transductive setting), or a function that can predict for any instance in the instance space $\mathcal{X}$ whether it is anomalous (inductive setting). We here make abstraction of whether the method is transductive or inductive; we assume it results in a "model" M, the quality of which is measured by some function q. Often, $q(M)$ is computed by comparing M's predictions with the ground truth on some dataset, which may or may not be equal to D. Most algorithms for anomaly detection have hyperparameters that affect their behavior. We therefore formalize these algorithms as follows: an anomaly detection algorithm A is a function that, given a dataset D and values for the algorithm's hyperparameters θ, returns a model $A(\theta, D)$. Given a quality criterion q, the task of hyperparameter tuning, for a given dataset D, then boils down to:

$$\text{find } \theta^* = \arg\max_{\theta} q\left(A\left(\theta, D\right)\right). \tag{5.1}$$

5.3 Current Methodologies

Research on anomaly detection often involves determining which algorithms perform best on certain datasets or under certain conditions. This is typically determined empirically. We discuss two methodologies frequently used in the literature, plus a variation that acts as a reference point in our experiments, and we discuss in what ways they are flawed. We assume a *benchmarking* setup: multiple algorithms A_i are evaluated on b benchmark datasets $\mathcal{B} = \{D_1, \ldots, D_b\}$ using a performance metric q.

Out-of-the-box performance. The most straightforward and popular methodology [2, 27, 30, 34, 40, 42, 51, 54, 57, 68, 76, 81, 94, 97, 99, 103, 111, 115, 121, 131, 134, 140–142, 147, 153, 156] [1] is sticking to "default" hyperparameters $\bar{\theta}_j$ as recommended in the literature. Such recommendations come in two forms: as fixed values (e.g. in `iForest` [81], set the number of trees as $t = 100$) or as simple rules of thumb, based on some dataset characteristic (e.g. $\sqrt{|D|}$ bins for

[1] In all of these references, we observe the use of default hyperparameters. We do not mean to suggest they *exclusively* rely on this methodology. For more details -including excerpts relevant to our categorization- we refer to our full literature review, available at https://github.com/ML-KULeuven/comparative-evaluation-of-anomaly-detection-methods

`HBOS` [52]). This yields for each algorithm A and dataset D_j the *out-of-the-box performance*

$$q\left(A\left(\bar{\theta}_j, D_j\right)\right).$$

The overall **out-of-the-box performance** of an algorithm A is estimated as the average over all datasets used in the experimental comparison:

$$Q_{outofthebox}(A) = \frac{1}{b}\sum_{j=1}^{b} q\left(A\left(\bar{\theta}_j, D_j\right)\right). \tag{5.2}$$

This method is simple but has multiple drawbacks:

1. It is relevant if we assume that there is no automatic way of choosing better hyperparameters, and that practitioners will make no effort to find good hyperparameters for a task. This may not be realistic.
2. If the above assumption is wrong, it tends to underestimate the potential of algorithms whose performance strongly depends on the hyperparameters.
3. Different papers and different implementations use different "defaults", which still leaves the researcher with having to choose one of them.
4. It is often not known how these defaults were chosen. If the default for an algorithm A was chosen based on the observation that it works well on some collection of datasets that (unbeknownst to the researcher) happens to overlap with one used in a new investigation, then A is at an advantage, which may lead the researcher to wrong conclusions.
5. For a newly proposed method, no default exists yet; the question remains how to choose a default. Because of the previous point, it is likely that the new default is chosen based on a different procedure than the other defaults, which jeopardizes the fairness of the comparison between algorithms.

Peak performance. Another popular methodology [22, 26, 46, 51, 61, 65, 68, 70, 83, 118, 162] [2] is to use ground truth labels and select the optimal hyperparameters θ_j^* for each problem,

$$\theta_j^* = \arg\max_{\theta} q(A(\theta, D_j)).$$

Averaging this over all datasets gives the **peak performance** of an algorithm:

$$Q_{peak}(A) = \frac{1}{b}\sum_{j=1}^{b} q(A(\theta_j^*, D_j)). \tag{5.3}$$

[2] Same comment as footnote 5.3, but for optimal hyperparameters instead of defaults.

This strategy essentially measures the potential of an algorithm. Its drawbacks are:

1. It is relevant if we assume that practitioners can always select the optimal parameters for a task. This is usually unrealistic.
2. It overestimates the expected performance of the algorithm when the above assumption is false.

Using this strategy is essentially a version of "tuning on the test set", which is generally considered unsound.

Best-default performance. Unlike the previous strategies, this one is not common in the literature. Rather, it serves as a reference point in our experiments. Instead of selecting the optimal hyperparameters for each problem individually, one can select the hyperparameters that perform best on the whole collection of datasets, on average:

$$\tilde{\theta} = \arg\max_{\theta} \frac{1}{b} \sum_{j=1}^{b} q(A(\theta, D_j))$$

$\tilde{\theta}$ are the "optimal default" hyperparameters for the collection of datasets under consideration. The **best-default performance** of an algorithm is

$$Q_{bestdefault}(A) = \frac{1}{b} \sum_{j=1}^{b} q\left(A\left(\tilde{\theta}, D_j\right)\right). \tag{5.4}$$

Like out-of-the-box performance, best-default performance assumes no problem-specific tuning is done. The main difference is that it computes a good default for a given collection of datasets, rather than obtaining it from external sources. This makes it unambiguous, and allows treating all algorithms on an equal footing. On the other hand, it is unsound for exactly the same reason as peak performance: it has access to information that would not normally be available to a practitioner. The pessimism of using default hyperparameters and the optimism of "tuning on the test set" partially cancel each other out.

5.4 A strategy based on validation sets

Because it is in their best interest to achieve the best possible performance, our methodology assumes that typical practitioners do an honest effort to find good

hyperparameters. In particular, our interpretation of "honest effort" is as follows: the practitioner has access to (or alternatively, spends some time creating) a labeled subset $D^L \subset D$ and uses that information to tune the hyperparameters. In a research setting, given a collection of datasets D_j, this can be simulated by determining

$$\theta_j^\dagger = \arg\max_\theta q_{D_j^L}\left(A(\theta, D_j)\right)$$

and the **tuned performance** for an algorithm A is then:

$$Q_{tuned}(A) = \frac{1}{b}\sum_{j=1}^{b} q_{D_j \setminus D_j^L}\left(A\left(\theta_j^\dagger, D_j\right)\right). \tag{5.5}$$

Note that the quality criterion q is now replaced by two variants $q_{D_j^L}$ and $q_{D_j \setminus D_j^L}$. This reflects the fact that q still measures how close the model's predictions are to the ground truth, but on different subsets of the original dataset: disjoint subsets are used for determining the best hyperparameter values, and for evaluating the model with those values.

In fact, this methodology is nothing new. The use of a labeled validation set to select appropriate hyperparameters is standard practice in supervised ML and is also supported by recent AD toolboxes [79]. Nevertheless, this methodology appears to be largely overlooked in the context of comparative evaluation of AD methods. Unlike the methodologies discussed in Section 5.3, benchmarking with such tuned performances is accurate, reproducible, and sound. One might ask why such standard practice is not already common in anomaly detection. A possible answer is that it requires a labeled subset of data, which goes against the idea of unsupervised learning. However, in our experience, the assumption that the practitioner is willing to provide a small amount of labeled data, in return for a more accurate anomaly detector, is much more realistic than the assumption that they are not interested in getting better than out-of-the-box performance, or that they can guess the optimal hyperparameter settings necessary to obtain peak performance.

5.4.1 Validation set size V

One important parameter of this procedure is the size of the validation set. Increasing the size of the validation set increases the likelihood that the model's performance on the validation set is indicative for the performance on the full dataset, because the validation set becomes more representative. On the other hand, the bigger the validation set, the more labels that need to be gathered by the practitioner. This introduces a trade-off: the validation set should be

small to limit the labelling effort but not too small as the best performing hyperparameters on a small set might not perform well on the rest of the data.

So, what is the minimum validation set size required to select good hyperparameters? In this paper, we determine this minimum in two ways. First, from a practical point of view, we want to know how many labeled instances are needed in order for this methodology to work. In our experiments, we determine this by investigating the effect of validation set size on our proposed strategy. Second, we also propose a statistical criterion which ensures that, on a given dataset, the validation set size is large enough to distinguish between the performance of a given model and a random classifier. In fact, if the validation set size does not statistically allow inferring that the given model performs differently from random predictions, then comparing different detectors is meaningless. In both cases, we ensure that the validation set contains at least one anomaly and that its contamination (i.e. the fraction of anomalies) is as close as possible to the contamination of the full dataset.

Statistical criterion for the validation set size V

The goal of this criterion is to determine a lower bound on the size of the validation set such that the AUC, i.e., the area under the *receiver operating characteristic* (ROC) curve [16, 45], of a given anomaly detector is statistically different from that of a random predictor. We use AUC because it is a common performance metric in anomaly detection [1, 22]. Similarly to [161], we follow three steps to derive this bound. In the first step, we observe that the AUC on a discrete set of examples takes values in $\left\{\frac{i}{m_0 m_1} : 0 \leq i \leq m_0 m_1\right\}$, where m_0 is the number of normal examples (class 0) in the validation set, and m_1 the number of anomalies (class 1). ROC curves can be seen as 2D lattice paths starting from $(0,0)$ and ending in $(1,1)$. Every time the threshold used to generate the ROC curve moves, the path takes a direction and makes a step of length $\frac{1}{m_0 m_1}$. The area under any path corresponds to the AUC in our setting. In the second step, we investigate the behavior of the AUC under random paths. According to [136], this area has a normal distribution with mean $\frac{1}{2}$ and variance $\frac{m_0+m_1+1}{12 m_0 m_1}$, to the limit when $m_0, m_1 \to +\infty$. Given that the contamination factor is $\gamma = \frac{m_1}{m_0+m_1}$, and that the size of the validation set is $V = m_0 + m_1$, we can rewrite the variance as $\frac{V+1}{12\gamma(1-\gamma)V^2}$. Finally, when $V \to +\infty$, using $\mathcal{U}$ as the AUC of a random classifier we derive that

$$\mathcal{U} \sim \mathbb{N}\left(\frac{1}{2}, \frac{V+1}{12\gamma(1-\gamma)V^2}\right) \implies \mathbb{P}(\mathcal{U} \geq s) \approx 1 - \Phi\left(\frac{\left(s-\frac{1}{2}\right)V}{\sqrt{\frac{V+1}{12\gamma(1-\gamma)}}}\right), \quad (5.6)$$

where Φ is the cumulative distribution function (cdf) of the standard normal distribution. Note that $\mathcal{U}$ follows a normal distribution only in the limit [136]. Thus, our approximation may have (small) approximation errors. As a result, given a level of significance p, we choose the size of the validation set V such that $\mathbb{P}(\mathcal{U} \geq s) \leq p$. This means that the AUC achieved by our detector is statistically different from the random prediction with a confidence level of $1 - p$.

In our experiments, for each dataset, we use the average out-of-the-box performance of all algorithms as the AUC value s, the real contamination (i.e., the fraction of anomalies) of the validation set as γ and a significance level p of 0.05, unless explicitly stated otherwise.

5.5 Experiments

We demonstrate the impact of hyperparameter selection on the outcomes of a benchmarking study. In the same context, we investigate the benefits and limitations of our proposal to tune hyperparameters based on a validation set. This yields four research questions:

Q1 Does the methodology for selecting hyperparameters affect the ranking of a benchmarking study?

Q2 How does the methodology for selecting hyperparameters affect the performance of an algorithm?

Q3 Is a small labeled validation set sufficient for identifying a good set of hyperparameters?

Q4 Is an algorithm's performance on the validation set always representative of its performance on the test set?

Datasets

We use a collection of 16 benchmark datasets (Campos et al. [22]) which are often used in the AD-literature. There are multiple versions of each dataset. We use the normalized version, without duplicates. If the contamination (i.e. the fraction of anomalies) exceeds 20%, we use one of the subsampled versions at random (Table 5.2).

Table 5.1: Hyperparameter grid used in our experiments. Random seeds are fixed to ensure reproducibility.

Algorithm	Hyperparameters
`LOF` and `kNN`	$k = \{3, 5, 7, \ldots, 299\}$
`HBOS`	$n_bins = \{5, 10, 15, \ldots, 100\}$
`iForest`	$n_estimators = \{25, 50, 75, \ldots, 300\}$ $max_samples = \{0.1, 0.2, 0.3, \ldots, 1\}$ $max_features = \{0.1, 0.2, 0.3, \ldots, 1\}$ $random_state = 3423452345$
`CBLOF`	$n_clusters = \{2, 4, 6, \ldots, 48\}$ $\alpha = \{0.1, 0.2, 0.3, \ldots, 0.9\}$ $\beta = \{2, 4, 6, \ldots, 20\}$ $use_weights = \{True, False\}$ $random_state = 123412351$
`OCSVM`	$kernel =$'rbf' $\nu = \{0.02, 0.04, 0.06, \ldots, 1\}$ $\gamma = \{0.001, 0.005, 0.01, 0.05, \ldots, 5000, 10000\}$

Algorithms

We use six well-known unsupervised anomaly detectors from four different families:

1. **Density-based:** *local outlier factor* (`LOF`) [18], *histogram-based outlier detection* (`HBOS`) [52], and *cluster-based local outlier factor* (`CBLOF`) [62].
2. **Proximity-based:** *k-nearest-neighbor-based outlier detection* (`kNN`) [36].
3. **Isolation-based:** *isolation forest* (`iForest`) [81].
4. **Kernel-based:** *one-class support vector machine* (`OCSVM`) [116].

We use the implementations available in the `PyOD` python package [163].

Table 5.2: Dataset characteristics of the datasets used in our experiments. The contamination γ is the fraction of anomalous instances.

Dataset name	#attributes m	#instances $\|D\|$	contamination γ
ALOI	27	49534	3.0%
Annthyroid	21	7129	7.5%
Arrhythmia	259	256	4.7%
Cardiotocography	21	1681	2.0%
InternetAds	1555	1966	18.7%
Ionosphere	32	250	10.0%
KDDCup99	40	48113	0.4%
Lymphography	18	148	4.1%
PageBlocks	10	5393	9.5%
Pima	8	555	9.9%
Shuttle	9	260	5.0%
SpamBase	57	2579	2.0%
Stamps	9	340	9.1%
WBC	9	200	5.0%
WDBC	30	100	10.0%
Waveform	21	3443	2.9%

Default hyperparameters

To measure out-of-the-box performance, we need default hyperparameters. As suggested in the literature, we set: $k = max\,(10, 0.03 \cdot |D|)$ for `kNN` and `LOF` [41, 103]; fixed-width histograms with $\sqrt{|D|}$ bins for `HBOS` [52]; number of trees $t = 100$ and number of samples per tree $\phi = 256$ for `iForest` [81]; $\alpha = 0.90$, $\beta = 5$, and k-means with $k = 10$ as a clustering algorithm for `CBLOF` [62]. Finally for `OCSVM`, we use the implementation defaults: rbf kernel with $\nu = 0.5$ and $\gamma = 1/m$, with m the number of features of dataset D.

Optimal & best-default hyperparameters

Optimal and best-default hyperparameters (Section 5.3) are found via an exhaustive gridsearch (Table 5.1).

Performance metrics

We report the average area under the *receiver operating characteristic* (ROC) curve (AUC) [16, 45], a canonical choice [1, 22] in anomaly detection, as well as the average rank. To calculate the average rank, for each dataset, we rank the algorithms from best to worst performer according to AUC. The average rank of a method is the average position of the method in each of the rankings.

While AUC is still the most prominent performance metric in anomaly detection, it should be noted that ROC curves can present an overly optimistic view of an algorithm's performance if there is a large class imbalance (which is often the case for anomaly detection) [32]. Precision-Recall (PR) curves have been proposed as an alternative to ROC curves for imbalanced scenarios [112]. However, Flach and Kull [47] argue that the area under such a PR curve also has some issues related to incoherent scales and propose to use Precision-Recall-Gain curves (and the area under such a curve) as an alternative.

Experimental setup

For each experiment and dataset, we do the following:

1. Select a test set, which will be the same for each algorithm.
2. From the remaining instances select the validation set, its size specified as either a maximum number of instances, or via our statistical criterion (Eq. 5.6) for a given p-value. The validation set contains at least one anomaly and its contamination is as close as possible to the contamination of the full dataset. Like the test set, the validation set is the same for each algorithm.
3. When doing hyperparameter selection via our proposal, tune the hyperparameters on the validation set.
4. Measure the AUC on the test set, regardless of the methodology used to select hyperparameters. This ensures that all reported performance estimates are comparable.
5. To account for variance in performance estimates due to the random test and validation sets, this procedure is repeated ten times with different test and validation sets. We report the average performance over these ten runs.

Table 5.3: Benchmarking several AD-algorithms, based on different methodologies for hyperparameter selection. Size of validation set for tuned performance determined via Eq. 5.6 with $p = 0.05$. For each algorithm, we report average AUC and average rank (lower rank is better), across the entire benchmark.

Peak (Eq. 5.3)			Best-default (Eq. 5.4)			Out-of-the-box (Eq. 5.2)			Tuned (Eq. 5.5)		
algorithm	*avg AUC*	*rank*	*algorithm*	*avg AUC*	*rank*	*algorithm*	*avg AUC*	*rank*	*algorithm*	*avg AUC*	*rank*
IForest	0.88	2.72	IForest	0.84	2.75	IForest	0.82	2.66	IForest	0.85	2.44
CBLOF	0.87	3.03	CBLOF	0.83	3.25	KNN	0.82	2.91	KNN	0.84	2.5
OCSVM	0.88	3.28	KNN	0.83	3.34	CBLOF	0.8	3.22	HBOS	0.81	3.56
LOF	0.87	3.5	LOF	0.81	3.59	HBOS	0.8	3.81	LOF	0.82	3.81
KNN	0.86	3.88	HBOS	0.8	3.91	LOF	0.77	4.12	CBLOF	0.8	3.94
HBOS	0.83	4.59	OCSVM	0.8	4.16	OCSVM	0.77	4.28	OCSVM	0.75	4.75

5.5.1 Q1: Does the methodology for selecting hyperparameters affect the ranking of a benchmarking study?

To answer **Q1**, we repeatedly conduct a benchmark study: first based on peak, best-default and out-of-the-box performance, as described in Section 5.3; then based on tuned performance (our proposal), as explained in Section 5.4.

Results. Table 5.3 shows that different hyperparameter selection methodologies lead to different rankings of the algorithms. Based on peak performance, `iForest` performs best, closely followed by `CBLOF` and `OCSVM`. According to best-default performance, the top three should be `iForest`, `CBLOF` and `kNN`. Out-of-the-box performance also indicates these three as top performers, but puts `kNN` ahead of `CBLOF`. Finally, tuned performance also indicates `iForest` and `kNN` as the top performers, followed by `HBOS`. But `CBLOF`, which was among the top performers according to all other methodologies, now drops to second to last in the ranking.

Conclusion. When benchmarking AD algorithms, the hyperparameter selection methodology influences the results to such an extent that the final conclusions of the study are affected. This is due the fact that some algorithms benefit more than others from dataset-specific hyperparameters. In our experiment, `CBLOF` illustrates this nicely: it is the second best performer with optimal hyperparameters, but just the penultimate method with tuned hyperparameters. Similarly, `OCSVM` is among the top performers with optimal hyperparameters, but ranks last in all other evaluation settings.

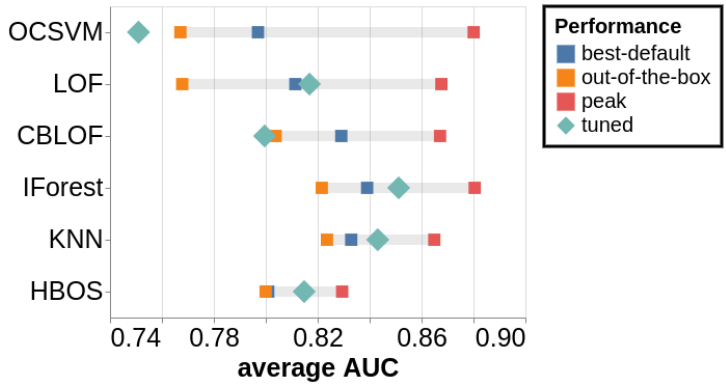
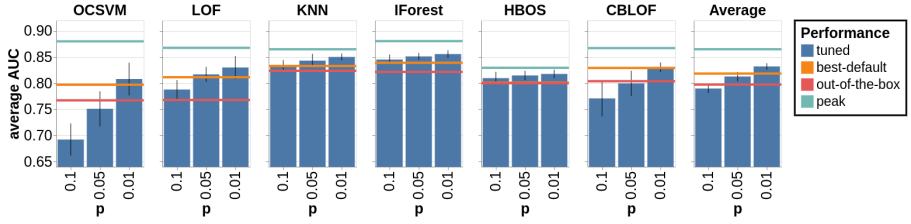
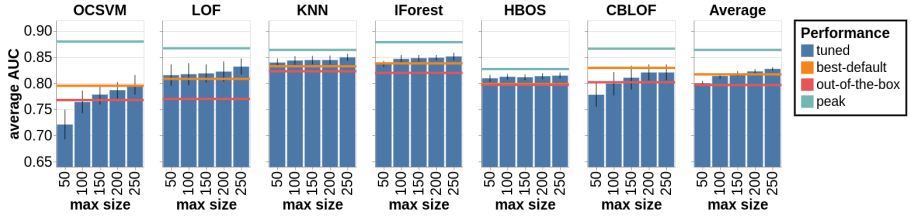


Figure 5.1: Performance (in AUC) according to each hyperparameter selection methodology, for each algorithm in our study, averaged across all datasets.

5.5.2 Q2: How does the methodology for selecting hyperparameters affect the performance of an algorithm?



(a) Validation size via Eq. 5.6 for different p values.



(b) Maximum validation size as an absolute amount of instances.

Figure 5.2: Tuned performance (averaged over all datasets) for different validation set sizes. Vertical black lines indicate the standard deviation of tuned performance across different runs. Out-of-the-box, best-default and peak performance estimates shown as horizontal lines.

We now know that the ranking produced by a benchmarking study can change, depending on how hyperparameter are selected (**Q1**). Digging deeper, we focus on the impact of hyperparameter selection on the performance of individual algorithms (**Q2**).

Results. Figure 5.1 shows that some algorithms are more sensitive than others when it comes to hyperparameter selection.

First, the gap between out-of-the-box and peak performance tells us how much an algorithm could *theoretically* benefit from optimal dataset-specific hyperparameters. In general, all algorithms benefit from optimal hyperparameters: on average, the AUC increases with 8.6% when comparing out-of-the-box to peak performance. `OCSVM` and `LOF` benefit the most with AUC improvements of 14.7% and 13.0%, respectively; these algorithms will appear a lot stronger in a comparative evaluation based on peak performance than they would in one based on out-of-the-box performance. `HBOS` benefits the least from optimal hyperparameters with an improvement of only 3.7%.

Second, the tuned performances tell us how much of those theoretical performance gains can be realized *in practice* with a reasonable amount of tuning. Tuning `LOF`, `iForest`, `kNN`, and `HBOS` yields an improvement over out-of-the-box performance of 6.4%, 3.6%, 2.4%, and 1.9% respectively. For these algorithms, roughly half of the theoretically possible performance gain is realized with a reasonable amount of tuning; on average, the tuned hyperparameters perform similar to or slightly better than the best-default performance. In contrast, for `CBLOF` and `OCSVM` the tuned performance is on-average worse than their out-of-the-box performance.

Finally, regardless of how the default hyperparameters are set (fixed values or via rules of thumb), best-default performance always exceeds out-of-the-box performance.

Conclusion. In theory, each algorithm in our study could benefit from optimal hyperparameters for each dataset, as peak performance consistently exceeds out-of-the-box performance. In practice, the performance achieved with reasonable tuning strongly depends on the algorithm. For most (`HBOS`, `iForest`, `kNN` and `LOF`), tuned performance exceeds best-default performance. For others (`CBLOF` and `OCSVM`), tuning is counterproductive, with tuned performance lower than out-of-the-box performance. Furthermore, whether or not tuning helps for a particular algorithm does not depend on potential performance gains: although `OCSVM` and `LOF` both have a large performance gap between their out-of-the-

box and peak performance, LOF benefits from tuning, but OCSVM does not. In Section 5.5.4, we investigate what makes CBLOF and OCSVM difficult to tune.

5.5.3 Q3: Is a small labeled validation set sufficient for identifying a good set of hyperparameters?

In Section 5.5.2, we show that tuning your hyperparameters on a validation set enables some algorithms to do better than their out-of-the-box performance. Ideally, this validation set is as small as possible, because a smaller validation set requires less labeling effort from the practitioner. So to answer **Q3**, we measure the effect of the size of the validation set on the tuned performance. To do so, we extend the results of Table 5.3 by computing tuned performance for several other validation set sizes. We determine these validation set sizes in two different manners: First, by using our statistical criterion (Eq. 5.6, Section 5.4.1). Second, by using a (maximum) absolute validation set size n, whilst ensuring that at most 25% of each dataset is used for tuning: i.e., validation set size $V = \min\left(n, \frac{|D|}{4}\right)$. Because the contamination of validation set is kept similar to the contamination of the full dataset (Table 5.2), the validation set often contains only a few anomalies.

Results. Fig. 5.2 shows various tuned performances for different validation set sizes. Overall (rightmost panels, Fig. 5.2), a validation set size of 100 instances, or alternatively, a validation set size determined via our criterion (Eq.5.6) using $p = 0.05$, is sufficient for tuned performance estimates to consistently exceed the out-of-the-box performances. However, we observe substantial differences between individual algorithms. For HBOS, iForest, LOF and kNN, a small validation set of 50 instances suffices to get a tuned performance similar to their best-default performance. OCSVM and CBLOF need a larger validation set (250 instances) before they reach best-default performance.

Conclusion. On average, in our experiments, hyperparameters acquired by tuning on a validation set of 100 instances (or using our statistical criterion with $p = 0.05$) yield performances that exceed the out-of-the-box performance. Overall, larger validation sets lead to higher tuned performances. As for individual algorithms, some algorithms seem more difficult to tune than others as they need more labelled data before exceeding default-performance. OCSVM and CBLOF are prime examples: they require fairly large validation sets ($p = 0.01$

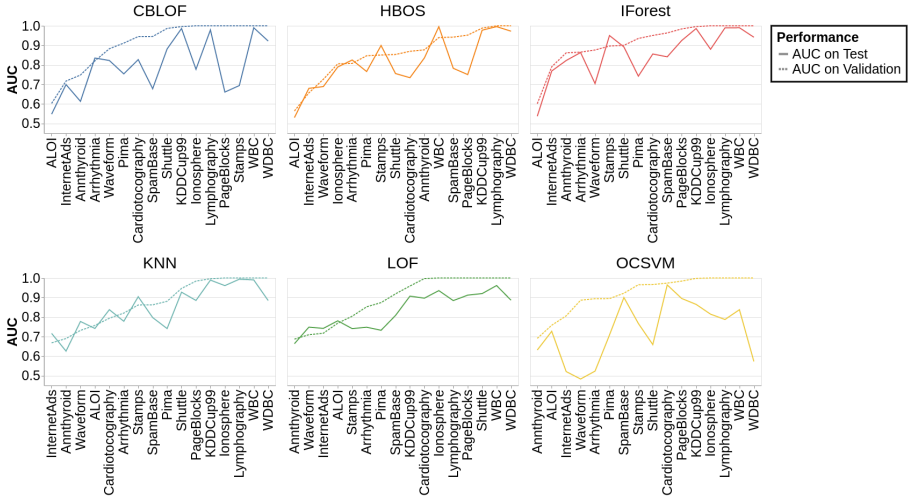


Figure 5.3: AUC on validation set (dashed line) and on test set (continuous line) for each algorithm and for each dataset. Validation set size via Eq. 5.6 with $p = 0.05$.

or 250 instances) to find hyperparameters that perform similar to best-default performance[3].

5.5.4 Q4: Is an algorithm's performance on the validation set always representative of its performance on the test set?

Using a small validation set to select good hyperparameters assumes that the performance of a detector on the validation set is indicative of its performance on the test set. However, as `CBLOF` and `OCSVM` are clearly more difficult to tune than others (**Q2**, **Q3**), it seems that this assumption is not always satisfied. This is what we aim to verify with **Q4**: given a dataset D, is an individuals algorithm's performance on the validation set always representative of its performance on the test set?

Results. Fig. 5.3 depicts the performance on validation[4] and on test set for the best performing hyperparameters of each algorithm and each dataset. It shows that performance on the validation set is not always indicative of the performance

[3]Emmott et al. [40] also observed that `OCSVM`'s hyperparameters are hard to optimize.
[4]Validation set size determined via our statistical criterion with $p = 0.05$.

on the test set, and that the size of this effect differs from algorithm to algorithm. For `HBOS`, `kNN`, `iForest` and `LOF`, the mean absolute difference between the average validation and test set performance over 10 random validation sets is 0.033, 0.041, 0.061 and 0.079 respectively. For these algorithms, the validation set performance matches (reasonably) well with that of the test set. For `CBLOF` and `OCSVM`, we observe large discrepancies on some datasets; the mean absolute difference between the average validation and test set performance is 0.109 and 0.164, respectively. This explains why we observed (Sections 5.5.2, 5.5.3) that these methods are difficult to tune: hyperparameters that work well on your validation set can completely fail on the test set.

Conclusions. Central to our methodology is the assumption that an algorithm's performance on the validation set provides a reliable estimate of its performance on the test set. For `CBLOF` and `OCSVM`, this is clearly not the case, which makes them difficult to tune. Indeed, under realistic conditions (i.e. given a limited amount of labeled data) these methods struggle to realize their full potential (Sections 5.5.2, 5.5.3). This shows that the usefulness of a small, labeled validation set is tightly intertwined with the kind of model you are using; e.g. a validation set that is "sufficiently representative" for `iForest`, can still be inadequate for `OCSVM`.

5.6 Conclusion

We summarize our main observations in five takeaway messages:

1. When benchmarking anomaly detection algorithms, the final ranking depends on the methodology used to select your hyperparameters (Table 5.3).
2. For most algorithms, tuning their hyperparameters on a small validation set yields better performance than using the default hyperparameters (Fig. 5.1).
3. None of the algorithms in our study, however, are able to realize their peak performance with a reasonable amount of tuning (Fig. 5.1).
4. The potential benefit of tuning does not depend on whether the algorithm has a large gap between out-of-the-box and peak performance, but on whether the algorithm is difficult to tune or not (Figs. 5.2, 5.3).
5. A small validation set containing only a few anomalies suffices to achieve those benefits. Concretely, we advise a validation set size of 100 instances,

or alternatively, a validation set size determined via our criterion (Eq.5.6) using $p = 0.05$ (Fig. 5.2).

Ultimately, we can draw three conclusions. One, it is indeed the case that out-of-the-box performance is an overly pessimistic estimate, whereas peak performance is overly optimistic (points 2-3). Two, our proposed methodology is the only one that yields *realistic* performance estimates, because it takes into account the difficulty of tuning a particular detector on a particular dataset (point 4). Three, our methodology is practically feasible, as a relatively small validation set with a few anomalies is sufficient to tune the hyperparameters (point 5). Moreover, we also derived a theoretical lower bound (Eq. 5.6) on the validation set size below which it is difficult to distinguish the AUC of a given anomaly detector with that of a random detector.

Future work

One interesting direction for future work is to investigate the use of other quality criteria that not only take the model performance into consideration but also the model complexity. In this way, simpler models will be preferred over more complex models if their performance on the validation set is similar. Possible candidate metrics[5] include the Akaike Information Criterion (AIC) [3] and Bayesian Information Criterion (BIC) [119].

[5]Both AIC and BIC measure the goodness of fit of the model on the training data instead of the performance on a hold-out validation set. Some adaptation of these metrics might be required.

Chapter 6

Modeling non-functional dependencies in dependency-based anomaly detection

In this chapter, we give an overview of dependency-based anomaly detection, a strategy to detect anomalies by modeling dependencies between attributes in a dataset. We present two novel insights that might help to increase the anomaly detection performance of this type of models in the future.

This chapter contains ideas from the following arXiv paper [129] supplemented with additional analysis and insights:

J. Soenen*, E. V. Wolputte*, V. Vercruyssen, W. Meert, and H. Blockeel. *AD-MERCS: Modeling Normality and Abnormality in Unsupervised Anomaly Detection.* 2023. arXiv: `2305.12958 [cs.LG]`

*Both authors contributed equally to this work

6.1 Introduction

This line of research started with the development of MERCS (Multi-directional Ensemble of Regression and Classification treeS) [145] within our research group. MERCS is a versatile model, meaning that, once trained on a dataset D, it can predict any set of attributes in D from any other set of attributes in D. This stands in contrast with standard machine learning where the prediction task has to be defined up front and a custom model is trained to predict that particular output from a particular input.

The authors hypothesized that these kinds of versatile models could be used for anomaly detection [145]. In the author's short description of their idea, anomalies are instances where some of the attribute values are different from what is expected after inspecting earlier data. To learn what is expected, they propose to learn a MERCS model that captures the different relations, or *dependencies*, between the attributes. To check whether a certain instance is anomalous, the MERCS model predicts the value for each attribute of that instance and compares that prediction with the observed value. Instances where one or more attribute values deviate from the prediction are marked as anomalous.

Together with Elia Van Wolputte, we developed AD-MERCS: an anomaly detector based on a MERCS model but adapted towards anomaly detection. Our long term goal was to develop a semi-supervised anomaly detector which could explain its predictions but also leverage feedback on those predictions from a domain expert. The first step towards that goal was to develop an unsupervised anomaly detector that was able to explain why it marked a certain instance as anomaly. In a second step, we would extend that model to include expert-feedback. Unfortunately, we never got to the second step.

AD-MERCS is not the only anomaly detection model that uses dependencies between attributes to detect anomalies. There are several *dependency-based anomaly detectors*, anomaly detectors that look for dependencies between attributes and mark instances that violate these dependencies as anomalies.

AD-MERCS introduces two novel ideas for dependency-based anomaly detection: (1) modeling non-functional dependencies and (2) leveraging patterns that hold for anomalies but not for normal instances. The resulting model is competitive with other anomaly detectors on standard datasets but only consistently better in specific (synthetic) circumstances [129]. These lackluster results prompted us to take a step back and thoroughly reevaluate the two main ideas behind AD-MERCS with respect to the already existing dependency-based anomaly detectors.

This chapter does not discuss AD-MERCS in detail, but instead focuses on the

Algorithm 4: Main idea behind dependency-based anomaly detection

function `train(`*dataset* $\mathcal{D}$`):`
  foreach *attribute* $A_k \in \mathcal{D}$ **do**
    // learn model M_k that predicts A_k from all other attributes
    $M_k \leftarrow$ `learn_model(`$\mathcal{D}$, *input* $= \mathcal{D} \setminus A_k$, *target* $= A_k$`)`
function `score(`*instance* x`):`
  foreach *attribute value* x_k *of* x **do**
    // predict value of x_k using model M_k
    $x'_k \leftarrow$ `predict(`M_k, *input* $= x \setminus x_k$`)`
    // compare observed value x_k with prediction x'_k
    $d_k \leftarrow$ `compare(`x_k, x'_k`)`
  // aggregate attribute-wise deviations d_k into anomaly score
  return `aggregate(`$d_1, d_2, \ldots, d_m$`)`

insights obtained by reevaluating our approach. We believe these insights will be very relevant for future developments in the field of dependency-based anomaly detection. In Section 6.2, we explain the general scheme of dependency-based anomaly detection and discuss the current state of the art. We then describe the two main novel ideas behind AD-MERCS: modeling non-functional dependencies (Section 6.3), and leveraging anomalous patterns for better anomaly detection (Section 6.4). Finally, we summarize the main take-away messages in the conclusion (Section 6.5).

6.2 Current state of dependency-based anomaly detection

The main idea behind dependency-based anomaly detection is to, for each attribute, learn a predictive model that predicts this attribute given all other attributes. In this way, the predictive models aim to capture the dependencies that hold between attributes. To test whether an instance is anomalous, each predictive model generates a prediction for its target attribute. These predictions are compared with the corresponding observed values. An instance where the predicted values are very different from the observed values is marked as anomalous (Algorithm 4). Dependency-based anomaly detection is an inductive learner. From the instances, a general model is learned and this model can then be used to score unseen instances.

This type of anomaly detection scheme has a number of advantages: the ability to handle mixed-type datasets, the ability to explain anomalies, and the ability to work with high dimensional datasets. We discuss all of these advantages in Section 6.2.1. Afterwards, we discuss two specific aspects of current dependency-based anomaly detectors : "Which dependencies are the anomaly detectors able to model?" (Section 6.2.2) and "How do dependency-based anomaly detectors avoid modeling the anomalies in the training set?" (Section 6.2.3).

6.2.1 Advantages of dependency-based anomaly detection

Dependency-based AD has a number of advantages. Interestingly, some of these advantages are only explicitly mentioned for or implemented in specific dependency-based anomaly detection algorithms. However, the advantages could also be implemented for other dependency-based anomaly detectors.

Ability to handle mixed-type datasets By mixing classification and regression models that can handle both discrete and continuous input attributes, a dependency-based anomaly detector can detect anomalies in datasets with both continuous and discrete features![96]. FRaC is the only dependency-based anomaly detector that does this explicitly, but the same principle is applicable to other dependency-based anomaly detectors as well.

Ability to explain anomalies If the underlying prediction model can explain its predictions, these explanations can be used to construct explanations as to why an instance is anomalous [99]: for each dependency that is violated, the target value can be compared to the expected value (i.e., the prediction from the predictive model) together with the explanation why this value is expected. E.g., for attribute A, the anomaly detector expects `<prediction>` because `<explanation>`; however, the actual value was `<observed value>`. ALSO is the only algorithm that mentions this, but again, the same principle is applicable to other dependency-based anomaly detectors as well.

Ability to handle high dimensional datasets Under two conditions – which most dependency-based anomaly detectors satisfy – the resulting dependency-based anomaly detector can also work in high-dimensional datasets.

- *Condition 1: robustness to 'irrelevant' input attributes.* For a certain target attribute, it is highly likely that only a few of the input attributes

contain any information about the target attribute (especially in high-dimensional datasets). Therefore, the predictive model used to model the dependencies should be robust to these *irrelevant* input attributes. All existing dependency-based anomaly detectors do this by using a model with a feature selection mechanism (such as decision trees) [96, 99] or by introducing an explicit feature selection step before learning a model [82].

- *Condition 2: robustness to 'irrelevant' target attributes.* There might be attributes for which there is no dependency with any of the other attributes; i.e., they are independent of the other attributes. The predictive model will not be able to predict these *irrelevant* attributes. This may cause large deviations between predicted and observed values while the instance itself is not anomalous. Therefore, most anomaly detectors (such as ALSO and FRaC) introduce an additional mechanism that aims to make them robust against these irrelevant target attributes.

6.2.2 Scoring mechanisms

Which anomalies a dependency-based anomaly detector can detect mostly depends on which types of dependencies the anomaly detector can capture. Most dependency-based anomaly detectors use a residual-based scoring scheme: a predictive model makes a single prediction and the difference between this prediction and the observed value is used as an indicator for anomalousness. However, with this basic residual-based scoring scheme, only dependencies where the values of the input attributes uniquely determine the value of the output attribute, so-called *functional dependencies*, can be modeled. Residual-based scoring on its own cannot capture *non-functional dependencies*, i.e., dependencies where the values of the input attributes do not determine a single value for the output attribute. These non-functional dependencies could occur due to inherent randomness or due to unobserved variables. In what follows, we discuss the scoring scheme of three influential dependency-based anomaly detectors and discuss which dependencies they can and cannot model.

Cross feature analysis: average match count and average probability

The dependency-based anomaly detection scheme was first used by Huang et al. to detect routing intrusions in ad-hoc mobile computer networks [66]. They propose a dependency-based anomaly detector called cross feature analysis (CFA). The method focuses on discrete attributes and therefore only uses classification models to model the dependencies between attributes. They propose two scoring schemes:

- *Average match count* where they use standard classification models and the anomaly score of instance x is the number of times the prediction of the classification model x'_k matches the observed value x_k divided by the number of attributes in the dataset m:

$$\text{average match count} = \frac{1}{m}\sum_{k=1}^{m} \mathbb{I}[x'_k = x_k], \tag{6.1}$$

  where $\mathbb{I}$ is the indicator function which returns 1 if its condition is satisfied. Instances with a low average match count are considered anomalous.

- *Average probability* where probabilistic classification models are used that predict a discrete probability distribution $P(X'_k = x)$ over all classes instead of a single prediction x'_k. Instead of looking at the most likely class, average probability looks at the probability of the observed value x_k according to the predicted probability distribution $P(X'_k = x)$:

$$\text{average probability} = \frac{1}{m}\sum_{i=k}^{m} P(X'_k = x_k) \tag{6.2}$$

  Intuitively, average probability simply looks at how likely the observed value is according to the probabilistic classifier and aggregates these probabilities for every attribute. Instances with a low average probability are unlikely and therefore anomalous.

Average probability scoring can express many more dependencies than average match count scoring by taking the uncertainty of the classifier into account. Average match count scoring can only capture functional dependencies; if non-functional dependencies occur, average match count fails to model them resulting in bad anomaly detection performance. On the other hand, average probability is able to capture these non-functional dependencies by using probabilistic classifiers.

We illustrate this based on two example scenarios:

- *Scenario 1: irrelevant target attribute.* If the target attribute is completely independent of the other attributes, the optimal non-probabilistic classifier will just predict the *majority class*, i.e., the most common value of the target attribute, independent of the input attributes. In this scenario, average match count will consider the majority class perfectly normal and will mark any other value as equally anomalous (left side of Figure 6.1). This negatively impacts the quality of the anomaly scores. With average probability scoring, the best probabilistic classifier will predict the

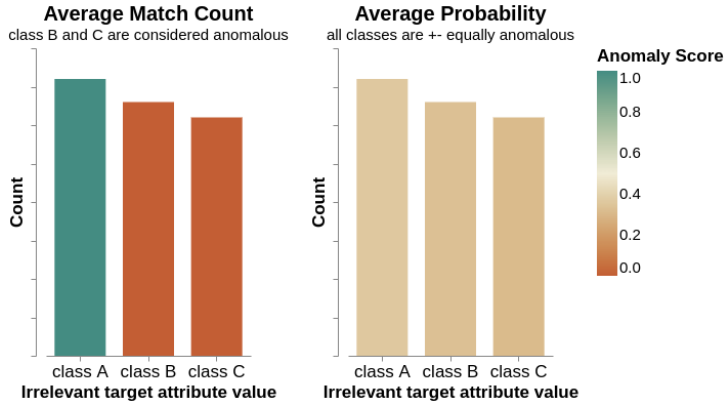


Figure 6.1: **(Left)** Average match count considers the most common value for the irrelevant target attribute completely normal and the rest anomalous. **(Right)** Average probability scores uses the empirical distribution function of the irrelevant target value to score instances; only uncommon values are seen as anomalous.

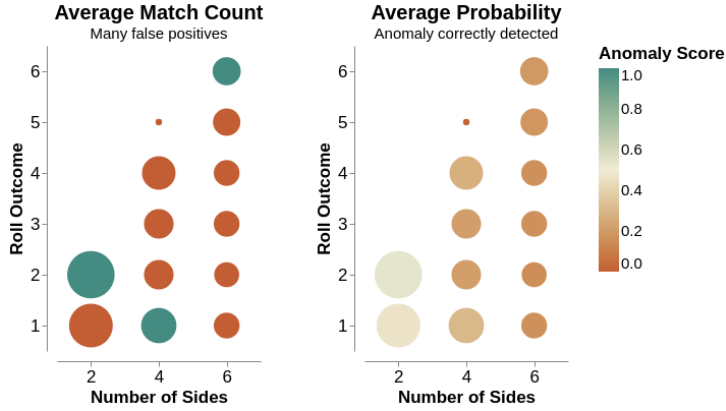


Figure 6.2: **(Left)** Average match count scores considers the most common outcome for each die as normal and the others, incorrectly, as anomalous (the larger the circle the more times this outcome was observed). **(Right)** Average probability correctly marks the roll of 5 with a 4-sided die as most anomalous. However, it considers rolls from a dice with 6 sides more anomalous than a dice with 2 sides.

empirical distribution of the target attribute and only uncommon values according to this target attribute will be marked anomalous (right side of Figure 6.1).

- *Scenario 2: non-functional dependency.* As an example non-functional dependency, consider the dependency between the number of sides on a die and the result of rolling that die. If a die has 4 sides the expected result is an integer between 1 and 4. For an 6-sided die, the expected result is an integer between 1 and 6. Again, if the die has 4 sides, with average match count, the classifier will predict the number which by chance was thrown the most and will mark any other results as anomalous (left side of Figure 6.2). With average probability scoring, the probabilistic classifier will correctly reflect that all numbers between 1 and 4 are more or less equally likely and will not mark them as anomalous. However, if there is a result 5 for a dice with 4 sides, this is correctly identified as anomalous (right side of Figure 6.2).

Although, average probability scoring can model non-functional dependencies there is one undesired consequence of using the probabilities directly. When rolling a 2-sided die, the probability of all normal outcomes is $\frac{1}{2}$. However, when rolling a 6-sided die, the probability of each normal outcome is only $\frac{1}{6}$. As such, average probability sees rolls from dices with more sides as more anomalous than dices with less sides (right side of Figure 6.2). In some cases, this might cause the roll of a 6-sided die to be considered more anomalous than the roll of a 2-sided die with some small deviations in other attributes.

Summary Average probability scoring can model non-functional dependencies and as a result is also robust against irrelevant target attributes. Average match count does not have these properties and will produce bad anomaly scores when any of the dependencies in the dataset is non-functional. However, both of these scoring mechanisms are limited to discrete attributes.

ALSO: weighted euclidean distance

ALSO [99] is an unsupervised anomaly detector and focuses on continuous attributes. It only uses deterministic regression models, limiting the representational power of the models themselves to functional dependencies. Similar to average match count scoring of CFA, this causes issues with irrelevant target attributes and other non-functional dependencies.

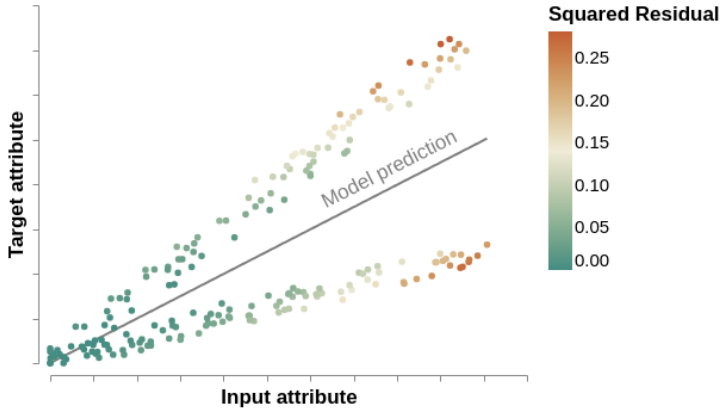


Figure 6.3: Even though ALSO is not able to give sensible scores to non-functional dependencies, the predictive model still performs better than consistently predicting the mean and therefore gets a non-zero influence on the final anomaly scores (for this example $w_k \approx 0.25$).

To avoid these issues, ALSO adds a weight w_k to each attribute k when aggregating the attribute-wise deviations:

$$\text{weighted euclidean distance} = \sqrt{\frac{1}{\sum_{k=1}^{m} w_k} \sum_{k=1}^{m} w_k \cdot (x'_k - x_k)^2} \tag{6.3}$$

The attribute weight w_k is determined based on how well the predictive model is able to predict the target attribute based on the root relative squared error (RRSE):

$$w_k = 1 - min(1, RRSE_k) \tag{6.4}$$

$$RRSE_k = \sqrt{\frac{\sum_{x \in \mathcal{D}} (x_k - x'_k)^2}{\sum_{x \in \mathcal{D}} (x_k - \overline{A_k})^2}} \tag{6.5}$$

where $\overline{A_k}$ is the average value of attribute k. Therefore, when the predictive model is unable to predict the target attribute, this predictive model (and its learned dependencies) will get a low weight in the calculation of the anomaly score of an instance. More specifically, if the average squared error of the predictor is larger than the average error of predicting the mean, w_k is equal to 0 and this predictive model does not influence the anomaly scores at all.

This has the following consequences:

- *Irrelevant target attributes are ignored.* In the case of an irrelevant target attribute, the optimal predictor will consistently predict the mean resulting in a root mean squared error of 1 and a weight w_k of 0.
- *Non-functional dependencies still negatively impact the anomaly scores* If the predictive model is able to at least partially model the non-functional dependency and perform better than consistently predicting the mean, the weight w_k will be greater than zero. This gives the predictive model still some influence on the final anomaly scores even though the model could be generating non-sensical anomaly scores (Figure 6.3).

Summary ALSO cannot model non-functional dependencies. By weighting the predictive models, it aims to reduce the influence of bad performing predictive models. This completely ignores irrelevant target attributes but non-functional dependencies can still have a negative influence on the anomaly scores.

FRaC: normalized surprisal

FRaC [95, 96] is the only existing model that can capture some (but not all) non-functional dependencies in continuous attributes. Moreover, as mentioned in Section 6.2.1, FRaC is the only dependency-based anomaly detector that handles both discrete and continuous attributes. FRaC uses standard non-probabilistic classification and regression algorithms to capture dependencies. However, instead of using the deviations themselves as a basis for the anomaly score, they look at the distribution of said deviations and check how surprising a certain deviation is. Therefore, it is not necessarily true that a large deviation is anomalous; a large deviation is only considered anomalous if it is uncommon according to the distribution of the deviation.

For discrete target attributes, FRaC computes a set of (x_k, x'_k) pairs (one for each instance in the training set) and uses these to model the probability to observe x_k given that x'_k was predicted $P(X_k = x_k \mid X'_k = x'_k)$.

For continuous target attributes, FRaC computes the differences of the observed and predicted values $\{r_{i,k} = x'_{i,k} - x_{i,k} \mid x_i \in \mathcal{D}\}$ and models the distribution of this residual $P(R_k = r_{i,k})$ with a smoothed histogram.

To score a certain attribute value x_k, the negative log likelihood of $P(X_k = x_k \mid X'_k = x'_k)$ or $P(R_k = r_{i,k})$ is used if the k'th attribute is discrete or

continuous, respectively:

$$\text{surprisal}(x_k, x'_k) = \begin{cases} -\log(P(X_k = x_k \mid X'_k = x'_k)) & \text{if } A_k \text{ is discrete} \\ -\log(P(R_k = x'_k - x_k)) & \text{if } A_k \text{ is continuous} \end{cases} \tag{6.6}$$

Figure 6.4 illustrates this scoring mechanism on a non-functional dependency that FRaC can capture.

However, with this scoring scheme, the surprisal tends to be higher for attributes with high-entropy value distributions (e.g., features with many possible values) even if the predictive model is relatively accurate. To normalize these surprisal scores over the attributes, the expected surprisal, the entropy of the attribute distribution[1] $entropy(A_k)$, is substracted from the surprisal score:

$$\text{normalized surprisal}(x_k, x'_k) = \text{surprisal}(x_k, x'_k) - entropy(A_k) \tag{6.7}$$

Finally, for each instance, the attribute-wise normalized surprisal scores are summed to get a final anomaly score:

$$\text{normalized surprisal}(x) = \sum_{k=1}^{m} \text{normalized surprisal}(x_k, x'_k) \tag{6.8}$$

This scoring mechanism enables FRaC to handle irrelevant target attributes. If the target is irrelevant, a reasonable predictive model will always predict the same value (the majority class for classification, the mean for regression). In this case, both $P(X_k \mid X'_k)$ for classification, and, $P(R_k)$ for regression, will resemble the empirical distribution of the target attribute. Therefore, it will mark uncommon values as more anomalous than common ones.

Moreover, FRaC can also correctly capture some specific types of non-functional dependencies but fails to model the full variety of non-functional dependencies. To correctly model all non-functional dependencies, you need to model the conditional probability distribution $P(target \mid inputs)$.

For discrete target attributes, FRaC models $P(target \mid prediction)$, which works if the distribution of the target is indeed only dependent on the prediction. Otherwise, this mechanism results in non-sensical anomaly scores. For example, in the dice example, assume a predictive model that predicts the outcome of a role from the number of sides of the die. If by chance the 6 and 4-sided die both have 4 as most observed outcome, then $P(X_k = x_k | X'_k = 4)$ will consider outcomes 5 and 6 as anomalous (because outcomes of 1 to 4 are more common

[1] For both discrete and continuous target attributes this entropy is calculated using the (smoothed) histogram used to model $P(X_k \mid X'_k)$ and $P(R_k)$, respectively.

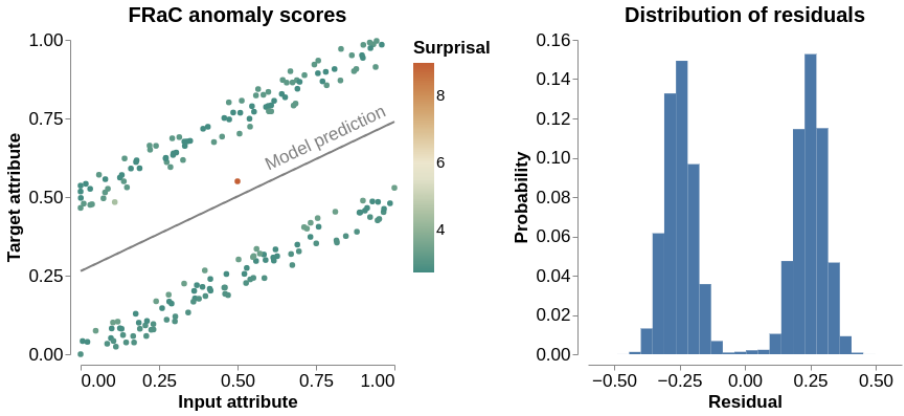


Figure 6.4: An example non-functional dependency that FRaC is able to model. Residuals around -0.25 and 0.25 are common according to the distribution of the residuals (right). Therefore, instances with this residual are not seen as anomalous. A residual of around 0 is very uncommon, therefore instances with a 0 residual are correctly identified as anomalous.

when rolling a 4 or 6-sided die). This example illustrates that using FRaCs idea to model $P(target \mid prediction)$ for scoring discrete attributes can have some unexpected results. Note that for this example CFA's mean probability scoring would work better. The probabilistic classifier could just directly capture the uncertainty in the non-functional dependency.

For continuous attributes, FRaC models $P(residual)$, which works if the distribution of the residual is independent of the instance itself. However, when some instances have more uncertainty than others, this fails to produce satisfactory anomaly scores (Figure 6.5).

Summary FRaC is able to model some (but not all) non-functional dependencies for both discrete and continuous attributes. However, because it still uses standard (non-probabilistic) predictive models its capabilities are limited.

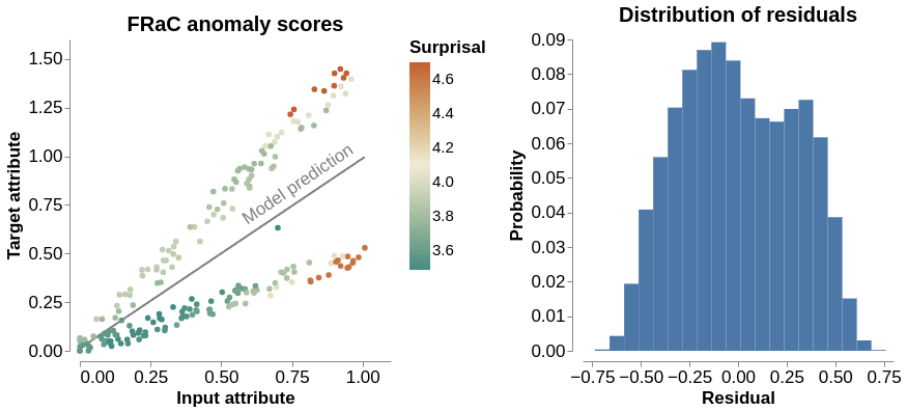


Figure 6.5: An example non-functional dependency that FRaC is not able to model. In this case, the distribution of the residuals is different for different instances. However, because FRaC only models the overal residual (not conditioned on anything), a residual 0 is considered perfectly normal. While if the input attribute is 0.75, a residual of 0 is clearly anomalous.

6.2.3 Handling anomalies in the training set: semi- and unsupervised dependency-based anomaly detection

Different dependency-based anomaly detectors assume different learning settings and have different ways of coping with anomalies in the training set. Naturally, if you train predictive models on a dataset that contains both normal and anomalous instances, those predictive models might learn to capture some of that anomalous behavior. This could negatively impact the anomaly detection performance.

Semi-supervised dependency-based AD

The first dependency-based anomaly detection models operated in a semi-supervised inductive fashion [66, 96]: they trained a model on a dataset that only contained normal instances and that model then assigns anomaly scores to unseen instances. So because the training set only contains normal instances, the predictive models only observe normal behavior and should never predict anomalous behavior.

This is the ideal setting for dependency-based AD. Unfortunately, this type of clean dataset might not be available in practice. For example, if a company just

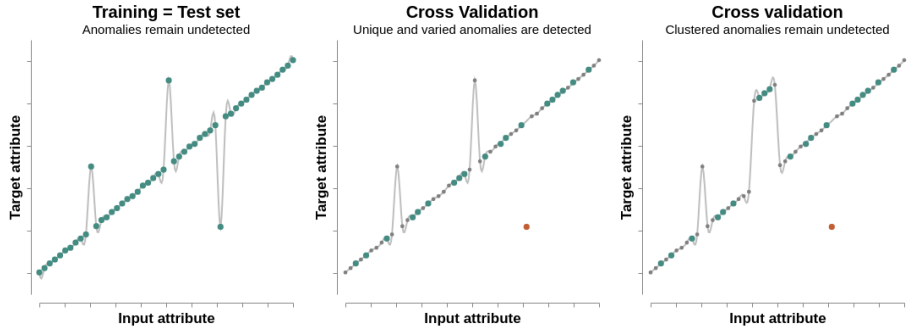


Figure 6.6: Illustration of how cross validation can help avoid modeling anomalies in the training set. **(Left)** When no cross validation is used, a model that overfits is able to predict the anomalies causing them to go undetected. **(Middle)** With cross validation (training set in gray, scores of test set instances in color), the model scores instances that it has never seen before; the anomaly in the test set is not easily spotted (the other anomalies will be detected in the other folds). **(Right)** Cross validation only defends against unique and varied anomalies, a group of clustered anomalies might still go undetected.

has raw collected data over time that data is likely to contain some anomalies.

Unsupervised dependency-based AD

ALSO [99] works in the standard unsupervised anomaly detection setting. In this setting, given an unlabeled dataset, the goal is to assign anomaly scores to the instances in that same dataset. In the anomaly detection literature this is the most common setting as it is often assumed that the dataset is unlabeled. However, this complicates things for dependency-based anomaly detection as the dataset now also contains anomalous instances. The predictive models might also learn to replicate anomalous behavior of the anomalies (especially if the model used is prone to overfitting). This then causes the predictive models to predict the anomalous behavior, causing the anomalies to remain completely undetected (left side of Figure 6.6).

To limit the impact of overfitting on the anomalies, ALSO introduces a cross validation scheme: to generate anomaly scores for the data in a specific fold, the predictive models are only trained on the data in all other folds. This makes it such that when a model is generating a prediction for a certain instance, that instance was not used to train the model. (middle of Figure 6.6)

However, this cross validation scheme only guards against anomalies that deviate from the normal behavior in unique and varied ways. If a group of anomalies deviates from the norm in the same (predictable) way, this group might get split over the training and test set. Because the anomalous behavior is in the train set, the predictive models might learn to predict the anomalous behavior. Second, these models are then used to score the anomalies of this group that are in the test set causing these anomalies to remain undetected (right side of Figure 6.6).

FRaC was first proposed in a semi-supervised inductive setting, Noto et al. [95] later observed that the same model also worked in an unsupervised inductive setting. Instead of training the model on a dataset with only normal instances, the model is trained on an unlabeled training set (a mix of anomalous and normal data) and is then used to assign anomaly scores to unseen instances. So just as in the previous case, the training set now contains both normal and anomalous instances.

However, there is a subtle difference with ALSO: FRaC doesn't generate a score for the instances in the training set, only for new unseen instances from a test set. So similar to the middle panel of Figure 6.6, a model that overfits will not have a big negative impact; as long as the anomalous behavior is very diverse, anomalies in the test set should be detected. However, as with ALSO, if a group of anomalies deviates from the norm in the same (predictable) way, anomalies in the test set that exhibit this same anomalous behavior will probably remain undetected. Note that FRaC can easily also generate scores for the training instances by introducing a cross validation scheme similar to ALSO.

6.2.4 Summary

Given all of this background, the most feature-rich dependency-anomaly detector would actually be a conjunction of ideas from different existing detectors:

- Using FRaC's scoring scheme to support both discrete and continuous attributes while being robust to irrelevant attributes and correctly modeling some non-functional dependencies.
- CFA's mean probability scoring for discrete attributes might be a better alternative to FRaC's scoring for discrete attributes. However, more experimentation is necessary to confirm this hypothesis.
- Generating explanations with ALSO's explanation scheme.
- Enabling the detection of anomalies in the training set with ALSO's cross validation scheme.

These different ideas seem compatible with each other without requiring much changes. Therefore, integrating these different ideas into a single model would mostly be an implementation effort. Still, this would be interesting to consolidate all existing literature and to illustrate the true potential of dependency-based anomaly detection.

On the other hand, there are two things that current anomaly detection methods don't consider. First, the current methods cannot model non-functional dependencies. Therefore, if one of the dependencies between attributes is non-functional, anomalies violating this dependency will go undetected and, in some cases, normal instances are wrongly seen as anomalous. Second, if the anomalies also follow predictable patterns, ALSO's cross validation scheme is not sufficient to defend against these anomalies. The underlying model might still learn to model these anomalies, again resulting in undetected anomalies. In the next two sections, we discuss two ideas that might help solve these issues.

6.3 Modeling non-functional dependencies

As discussed Section 6.2.2, different dependency-based anomaly detectors have different ways to model dependencies and to score instances but none of them are able to model all non-functional dependencies. CFA can model non-functional dependencies by using probabilistic classification models, but is limited to discrete attributes and it sensitive to the entropy of the probability distributions. Both ALSO and FRaC use a regressor which by definition can only model functional dependencies and use the deviation between prediction and observed value as an indicator for anomalousness. They then extend their scoring mechanism to handle irrelevant target attributes and, in the case of FRaC, a limited number of non-functional dependencies.

When developing AD-MERCS, we introduced a novel density-based scoring mechanism to model non-functional dependencies appropriately which we explain in the next section.

6.3.1 Modeling non-functional dependencies with density trees

AD-MERCS uses standard decision trees to model the dependencies between attributes but modifies them to allow for a density-based scoring mechanism. During training, the target values of the training instances in each leaf are not aggregated into a single value that is used for prediction. Instead, AD-MERCS models the probability distribution κ of the target value in each

leaf with a histogram (for categorical variables) or a continuous probability distribution[2](for numerical attributes). To score an instance, AD-MERCS first classifies the instance to one of its leafs and check how likely the observed value of the target value is according to the density in that leaf.

Inside a leaf, lower values of κ indicate less likely target attribute values. However, there are two reasons we cannot do anomaly detection using κ values directly. First, distinguishing high values of κ is pointless: all those indicate "normality" anyway. Second, κ values are incomparable between leaves: densities are normalized in terms of their area, not height, so their values depend on the local range of the target variable. As a solution, the κ values are translated into likelihood values ω using the following equation:

$$\omega_j(x) = \begin{cases} \frac{\kappa_j(x)}{\tau_j(\rho)} & \kappa_j(x) < \tau_j(\rho) \\ 1 & \kappa_j(x) \geq \tau_j(\rho) \end{cases} \tag{6.9}$$

where threshold $\tau_j(\rho)$ is exceeded by $\rho\%$ of the instances in leaf j. In other words, the $(100-\rho)\%$ lowest κ values are linearly mapped to the interval $[0, 1]$, the rest is mapped to 1. The closer to 0 an instance's ω value is, the more likely it is anomalous. Figure 6.7 illustrates this density-based scoring on a bimodal distribution of target values. Figure 6.8 shows that this mechanism is able to capture the non-functional dependency that FRaC was unable to model.

6.4 Leveraging patterns in the anomalies

In Section 6.2.3, we distinguished two different kinds of anomalies that contaminate the training set.

1. Unique and varied anomalies: anomalies that deviate from normal behavior in varied and different ways. This type of anomaly in the training set can be defended against by using cross validation.
2. Grouped anomalies: a group of anomalies that deviate from normal behavior in the same predictable way. For these anomalies, cross validation does not offer a solution.

Existing transductive dependency-based anomaly detection methods use cross validation to defend against unique and varied anomalies. However, they have no

[2]Any density estimation method could in principle be used, AD-MERCS used Gaussian kernel density estimation [120], selecting the kernel bandwidth automatically using the approach of Botev et al. [14].

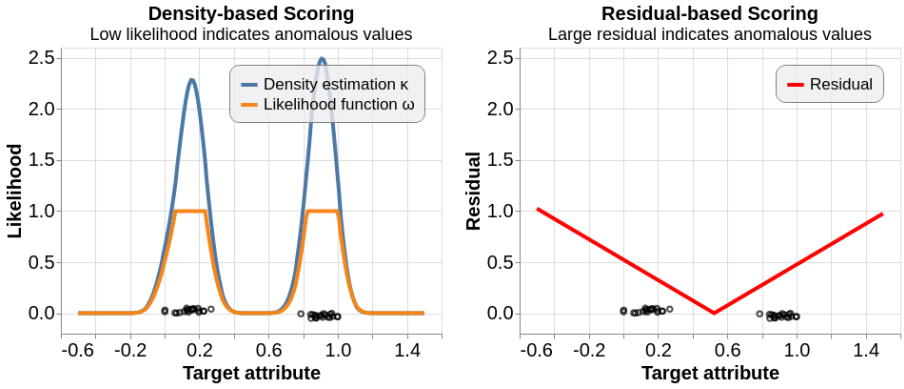


Figure 6.7: Density-based and residual-based scoring for a leaf where target values (black circles) follow a bimodal distribution. **(Left)** Density-based scoring correctly considers 0.5 as anomalous (likelihood of 0). The density estimation κ is shown in blue and the likelihood function ω used for scoring is in orange. **(Right)** Because the tree predicts 0.5 for any instance classified in this leaf, residual-based scoring considers instances with a target value of 0.5 perfectly normal in this leaf (the residual is 0), while this value can reasonably be assumed anomalous as it has never been observed before.

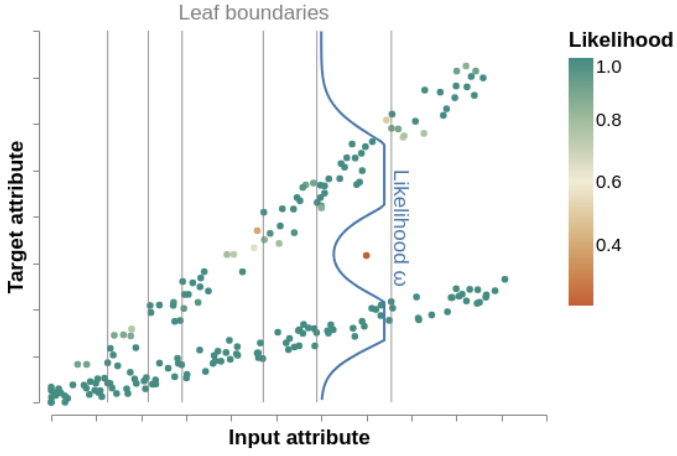


Figure 6.8: The density-based scoring mechanism of AD-MERCS can model non-functional dependencies. The leaf boundaries are shown in gray. The likelihood function for the leaf that contains the anomaly is shown in blue.

defense against grouped anomalies. Predictable behavior is always assumed to be normal, even though some of the anomalies might also exhibit characteristics that are predictable.

AD-MERCS doesn't make this same assumption; in AD-MERCS an instance may be anomalous because it deviates from a pattern that holds for normal instances, or - and this is novel - precisely because it adheres to a pattern that only holds for anomalous instances. To detect these anomalous patterns, AD-MERCS reasons over the different predictive models in the ensemble. Initially, an instance can be considered likely to be anomalous when it violates many different patterns that are modeled by the different trees in the ensemble. Now, when a single leaf in one of the decision trees contains many instances that are deemed anomalous by other trees in the ensemble, it is reasonable to assume that the leaf defines a set of conditions that are only fulfilled by anomalies instances; we call this an *anomalous context*. Instances in this leaf that were not yet (convincingly) identified as anomalies can now also be assumed to be anomalous. As such, AD-MERCS has a mechanism to protect itself against these grouped anomalies.

However, surprisingly, AD-MERCS does not use cross validation to guard against unique and varied anomalies in the training set. While detecting anomalous contexts might help to detect some of these unique and varied anomalies, cross validation is a simpler and likely more effective defense against this type of anomalies.

This raises the first open question: "Can cross-validation help AD-MERCS to deal with anomalies in the training set?" Intuitively, cross validation could help AD-MERCS do already deal with unique and varied anomalies in the training set, leaving only the grouped anomalies to be taken care of through anomalous contexts. Preliminary experiments indicate that cross-validation does help AD-MERCS perform better on benchmark datasets. However, the combination of cross-validation with anomalous context has not been investigated yet.

The second open question is "Do these grouped anomalies occur in real-world datasets?". In the paper we wrote on AD-MERCS, the benefit of these anomalous contexts was shown based on a synthetic dataset. It is therefore still unclear whether these 'grouped' anomalies occur in real-world datasets and whether this mechanism of anomalous contexts helps to detect them.

As such, the idea of using anomalous contexts is interesting but more investigation is required to really prove the value of anomalous contexts over the simpler cross validation.

6.5 Conclusion

In this chapter, we have given a broad overview of the dependency-based anomaly detection literature: we discussed the advantages of dependency-based anomaly detection in general, and the advantages of specific dependency-based anomaly detection over others. Through our description of the different models, we have shown that the best ideas from different dependency-based anomaly detectors are probably compatible with each other. This means that a dependency-based anomaly detector might exists that combines the best of all other models.

Finally, we also highlighted two weaknesses of the current dependency-based anomaly detectors and proposed two ideas that might help to alleviate these drawbacks.

- *Modeling non-functional dependencies* None of the existing dependency-based anomaly detectors can model all non-functional dependencies. We have presented a novel density-based scoring mechanism that can model these non-functional dependencies resulting in sensible anomaly scores even if non-functional dependencies are present in the data.
- *Clustered anomalies* The current way to guard against anomalies in the training set is to use cross validation. However, this mechanism does not defend against clustered anomalies. We have presented anomalous contexts as a way to detect these clustered anomalies although they might be modeled by (some of the) predictive models.

We hope the presented insights inspires future work in the field of dependency-based anomaly detection.

Chapter 7

Conclusion

In this final chapter, we summarize the main conclusions of the dissertation and give several directions for future work.

7.1 Summary and contributions

During my PhD, we have developed novel machine learning methods driven by the industry's need for explainable and (active) semi-supervised models. This with the broader goal of making user understand the machine learning models better and making machine learning models understand the users better. Most of the presented contributions situated themselves around clustering and anomaly detection in the field of unsupervised and semi-supervised learning. In the following paragraphs, we summarize each contribution in turn.

A framework to make active constraint-based clustering algorithms robust to user mistakes

Context In active semi-supervised clustering the algorithm queries the domain expert for constraints that guide the algorithm towards the desired clustering. This way, the domain expert obtains a clustering suitable for his/her application without experimenting with different unsupervised algorithms, hyperparameter settings and similarity metrics. However, existing active semi-supervised clustering algorithms are sensitive to noisy constraints. For example, COBRAS assumes that every user-provided constraint is correct and will enforce these

constraint in the clustering causing a significant decrease in clustering quality whenever a noisy constraint occurs.

Contribution A novel framework to make any active semi-supervised clustering algorithms robust to noisy constraints. The framework reasons about noise in the constraint set to detect and correct the noisy constraints. We believe our framework solves one of the major shortcomings of existing active semi-supervised clustering algorithms and makes these methods more widely applicable in practice.

A method to generate electricity consumption time series for unmeasured households to help plan the reinforcement of the low voltage grid

Context To help plan the reinforcement of the low voltage grid, EnergyVille calculates the currents and voltages in the low-voltage grid. As input for this simulation, EnergyVille needs fine-grained electricity consumption time series for all connected households. Unfortunately, these time series are unavailable for the vast majority of households.

Contribution We developed two approaches to generate daylong electricity consumption time series from the available household, weather and calendar attributes. Both approaches are entirely data-driven, the expert does not have to handcraft any rules or select relevant features. The first approach clusters the time series first and, only afterwards, learns a classifier to associate the clusters with the known attributes. The second approach uses predictive clustering to simultaneously learn a classifier and cluster the time series. Integrating clustering and classification results in a simple and fast model without losing any predictive performance (compared to the first approach). Moreover, this simple model is fully interpreteable allowing experts to inspect which time series are predicted in which circumstances. We hope these models help EnergyVille to set the next step towards a cost-efficient reinforcement of the low-voltage grid.

A new experimental methodology to ensure sound, fair and realistic comparison between anomaly detectors

Context To compare different anomaly detection algorithms on a set of benchmark datasets, the hyperparameters of each anomaly detector need to be set. In the literature, there are two common strategies: (1) set the

hyperparameters to their 'default' values resulting in a pessimistic performance estimation or (2) set the hyperparameters to their optimal values by tuning on the test set which is unsound and results in an overly optimistic performance estimation.

Contribution We propose to use a small labeled validation set to tune the hyperparameters of anomaly detection algorithms. We argue that this results in a sound, fair, realistic and reproduceable comparison between algorithms which is necessary to ensure that the field moves in the right direction. We hope this contribution can serve as a step towards unifying the different anomaly detection evaluation methodologies.

Novel insights into dependency-based anomaly detection

Context Dependency-based anomaly detection models the dependencies between attributes in a dataset and uses these to detect anomalies. The advantages of models are that these algorithms can handle high dimensional mixed-attribute datasets and that they can explain why an instance is marked as anomalous.

Contribution First, we discussed the main dependency-based anomaly detection models and discussed them in detail offering an easy entry point into the field. Second, we identified two drawbacks of the current dependency-based anomaly detection models: (1) they cannot capture non-functional dependencies and (2) anomalies that exhibit predictable behavior might remain undetected. For each of these drawbacks, we presented an idea that might help alleviate this drawback in the future. Although we can show the benefit of these ideas on simple examples and synthetic datasets, it remains an open question whether these ideas or variations of them can also help to detect anomalies in real-world datasets. With this contribution, we hope to inspire and accelerate future work in the field of dependency-based anomaly detection which we believe is a promising anomaly detection approach.

7.2 Future work

There are several interesting directions for future work.

Evaluation of scenario generation in the low voltage decision support pipeline

In Chapter 4, we presented two novel models that generate more accurate electricity consumption scenarios than previously used models as measured by the energy score between the predictions and a hold-out test set. However, it is still unclear how these more accurate electricity consumption scenarios impact the low voltage grid simulation results. An important future work is therefore to completely integrate the novel scenario generation methods into the simulation pipeline to measure if these improved scenarios also result in a more accurate simulation.

Benchmark datasets in anomaly detection

Proper evaluation of anomaly detection is required to make sure the field moves in the right direction. In Chapter 5, we discussed the need for fair and realistic hyperparameter tuning. Another aspect of the evaluation of anomaly detectors that doesn't receive enough attention is the datasets that are used to measure the performance of different detectors.

First, there is the question if the current datasets used are even suitable for anomaly detection. Many anomaly detection datasets are not real-world anomaly detection datasets but derived from classification datasets: one class is chosen as the anomalous class and is subsampled to make the anomalies rare [22]. Moreover, little is known about the anomalies in these datasets, making it difficult to analyze the capabilities of a particular detector.

We believe there is a need for real-world anomaly detection datasets with known anomaly characteristics. These datasets would open up a path to more insightful evaluation where the capabilities of a particular algorithm to detect a particular type of anomalies can be measured. In this way, we can start evaluating which algorithm's biases are suited for which types of problems. This can help practitioners to select a suitable algorithm for their application. Of course, this presumes that we have a way of characterizing different types of anomalies and datasets; an area which also requires more research.

Synthetic datasets offer a partial solution to this problem [40, 41]. These datasets allow complete control over the characteristics of a dataset and the anomalies therein. However, with synthetic datasets the question remains whether real-world datasets with similar characteristics exists. If no real-world counterpart exists, you might be solving a non-existing problem.

Future direction for dependency-based anomaly detection

Non-functional dependencies and anomalies that exhibit predictable behavior As discussed in Chapter 6, the current dependency-based anomaly detectors seem to struggle with non-functional dependencies and anomalies that exhibit predictable behavior. Further investigation into these areas might be able to improve the anomaly detection quality of dependency-based anomaly detectors. We believe that an investigation into the characteristics of anomalies in the currently used benchmark datasets (as discussed in the previous point) will help to generate more insight around these two specific drawbacks.

Semi-supervised dependency-based anomaly detection The long term goal of our research on dependency-based anomaly detection was to leverage the explanations generated by these dependency-based anomaly detectors to develop a semi-supervised dependency-based anomaly detector. However, due to the difficulties we encountered, we never got to that point. We still believe that these explanations offer a unique (and interpreteable) way for the expert to check the algorithms reasoning and to provide additional feedback to increase the anomaly detection performance. Instead of asking: "is this instance anomalous or not?" (as most semi-supervised anomaly detectors do). A dependency-based anomaly detector could ask things like: "I believe this instance is anomalous because this and this reason, is this correct?". In first instance, feedback could just consist of a checking whether the explanation is correct. Later, it might be possible to enable the expert to correct the algorithm's explanation and to then use this as feedback to improve the anomaly detection performance.

Tuning the hyperparameters of the predictive models One direction for future research that we didn't explore is the automatic tuning of the predictive models used to capture the dependencies between attributes. Some of these dependencies might be very simple requiring only simple models while other might require more complex models to be modeled appropriately. All the current methods use the same set of hyperparameters for all the predictive models. Having a mechanism that can adapt the hyperparameters to the prediction problem at hand might result in better anomaly detection and reduce the chance of overfitting on anomalies in the training set.

Bibliography

[1] C. C. Aggarwal. *Outlier Analysis*. Springer Intl. Publishing AG, 2017.

[2] F. Ahmed and A. Courville. "Detecting Semantic Anomalies". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.04 (2020), pp. 3154–3162. DOI: 10.1609/aaai.v34i04.5712. URL: https://ojs.aaai.org/index.php/AAAI/article/view/5712.

[3] H. Akaike. "A new look at the statistical model identification". In: *IEEE Transactions on Automatic Control* 19.6 (1974), pp. 716–723. DOI: 10.1109/TAC.1974.1100705.

[4] D. Arthur and S. Vassilvitskii. "K-Means++: The Advantages of Careful Seeding". In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '07. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. ISBN: 9780898716245.

[5] R. Bartels, D. G. Fiebig, M. Garben, and R. Lumsdaine. "An end-use electricity load simulation model: Delmod". In: *Utilities Policy* 2.1 (1992), pp. 71–82. ISSN: 0957-1787. DOI: https://doi.org/10.1016/0957-1787(92)90055-N. URL: https://www.sciencedirect.com/science/article/pii/095717879290055N.

[6] S. Basu, A. Banerjee, and R. J. Mooney. "Active semi-supervision for pairwise constrained clustering". In: *Proceedings of the 2004 SIAM international conference on data mining*. 2004, pp. 333–344.

[7] J. Bekker and J. Davis. "Learning from positive and unlabeled data: A survey". In: *Machine Learning* 109 (2020), pp. 719–760.

[8] N. M. M. Bendaoud, N. Farah, and S. Ben Ahmed. "Comparing Generative Adversarial Networks architectures for electricity demand forecasting". In: *Energy and Buildings* 247 (2021), p. 111152. ISSN: 0378-7788. DOI: https://doi.org/10.1016/j.enbuild.2021.111152. URL:

https://www.sciencedirect.com/science/article/pii/S0378778821004369.

[9] R. Bernards, J. Morren, and H. Slootweg. "Statistical modelling of load profiles incorporating correlations using Copula". In: *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*. Sept. 2017, pp. 1–6. DOI: 10.1109/ISGTEurope.2017.8260181.

[10] R. Bernards, W. van Westering, J. Morren, and H. Slootweg. "Analysis of Energy Transition Impact on the Low-Voltage Network Using Stochastic Load and Generation Models". In: *Energies* 13.22 (2020). ISSN: 1996-1073. DOI: 10.3390/en13226097. URL: https://www.mdpi.com/1996-1073/13/22/6097.

[11] M. Bilenko, S. Basu, and R. J. Mooney. "Integrating constraints and metric learning in semi-supervised clustering". In: *Twenty-first international conference on Machine learning - ICML '04*. 2004.

[12] H. Blockeel, L. D. Raedt, and J. Ramon. "Top-Down Induction of Clustering Trees". In: *Proceedings of the Fifteenth International Conference on Machine Learning*. ICML '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 55–63. ISBN: 1558605568.

[13] H. Blockeel. *Top-down induction of first order logical decision trees*. eng. 1998.

[14] Z. I. Botev, J. F. Grotowski, D. P. Kroese, et al. "Kernel density estimation via diffusion". In: *The annals of Statistics* 38.5 (2010), pp. 2916–2957.

[15] L. Botman, J. Soenen, K. Theodorakos, A. Yurtman, J. Bekker, K. Vanthournout, H. Blockeel, B. De Moor, and J. Lago. "A Scalable Ensemble Approach to Forecast the Electricity Consumption of Households". In: *IEEE Transactions on Smart Grid* 14.1 (2023), pp. 757–768. DOI: 10.1109/TSG.2022.3191399.

[16] A. P. Bradley. "The use of the area under the ROC curve in the evaluation of machine learning algorithms". In: *Pattern recognition* 30.7 (1997), pp. 1145–1159.

[17] L. Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[18] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. "LOF: Identifying Density-based Local Outliers". In: *Proceedings of the 2000 ACM SIGMOD Intl. Conference on Management of Data*. 2000, pp. 93–104.

[19] J. Bröcker and L. A. Smith. "Scoring Probabilistic Forecasts: The Importance of Being Proper". In: *Weather and Forecasting* 22.2 (2007), pp. 382–388. DOI: 10.1175/WAF966.1. URL: https://journals.ametsoc.org/view/journals/wefo/22/2/waf966_1.xml.

[20] J. Cai, J. Hao, H. Yang, X. Zhao, and Y. Yang. "A review on semi-supervised clustering". In: *Information Sciences* 632 (2023), pp. 164–200. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2023.02.088. URL: https://www.sciencedirect.com/science/article/pii/S0020025523002840.

[21] D. Calandriello, G. Niu, and M. Sugiyama. "Semi-supervised information-maximization clustering". In: *Neural networks* 57 (2014), pp. 103–111.

[22] G. O. Campos, A. Zimek, J. Sander, R. J. G. B. Campello, B. Micenková, I. A. E. Schubert, and M. E. Houle. "On the Evaluation of Unsupervised Outlier Detection: Measures, Datasets, and an Empirical Study". In: *Data Mining and Knowledge Discovery* 30.4 (2016), pp. 891–927.

[23] A. Capasso, W. Grattieri, R. Lamedica, and A. Prudenzi. "A bottom-up approach to residential load modeling". In: *IEEE Transactions on Power Systems* 9.2 (May 1994), pp. 957–964. ISSN: 1558-0679. DOI: 10.1109/59.317650.

[24] R. Caruana, M. Elhawary, N. Nguyen, and C. Smith. "Meta Clustering". In: *Sixth International Conference on Data Mining (ICDM'06)*. 2006, pp. 107–118.

[25] D. Ćevid, L. Michel, J. Näf, P. Bühlmann, and N. Meinshausen. "Distributional random forests: Heterogeneity adjustment and multivariate distributional regression". In: *The Journal of Machine Learning Research* 23.1 (2022), pp. 14987–15065.

[26] R. Chalapathy, A. K. Menon, and S. Chawla. "Anomaly detection using one-class neural networks". In: *arXiv preprint arXiv:1802.06360* (2018).

[27] R. Chalapathy, A. K. Menon, and S. Chawla. "Robust, deep and inductive anomaly detection". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2017, pp. 36–51.

[28] V. Chandola, A. Banerjee, and V. Kumar. "Anomaly detection: A survey". In: *ACM Comput. Surv.* 41.3 (2009), pp. 1–58.

[29] Commission for Energy Regulation (CER). *CER Smart Metering Project - Electricity Consumer Behaviour Trial, 2009-2010*. [dataset] 1st Edition, SN: 0012-00. https://www.ucd.ie/issda/data/commissionforenergyregulationcer/. Irish Social Science Data Archive, 2012.

[30] D. Datta, M. R. Islam, N. Self, A. Meadows, J. Simeone, W. Outhwaite, C. Hin Keong, A. Smith, L. Walker, and N. Ramakrishnan. "Detecting Suspicious Timber Trades". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.08 (2020), pp. 13248–13254. DOI: 10.1609/aaai.v34i08.7032. URL: https://ojs.aaai.org/index.php/AAAI/article/view/7032.

[31] I. Davidson and S. S. Ravi. "Agglomerative Hierarchical Clustering with Constraints: Theoretical and Empirical Results". In: *Knowledge Discovery in Databases: PKDD 2005*. Ed. by A. M. Jorge, L. Torgo, P. Brazdil, R. Camacho, and J. Gama. 2005, pp. 59–70.

[32] J. Davis and M. Goadrich. "The Relationship between Precision-Recall and ROC Curves". In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML '06. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006, pp. 233–240. ISBN: 1595933832. DOI: 10.1145/1143844.1143874. URL: https://doi.org/10.1145/1143844.1143874.

[33] D. Defays. "An efficient algorithm for a complete link method". In: *The computer journal* 20.4 (1977), pp. 364–366.

[34] R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui. "A comparative evaluation of outlier detection algorithms: Experiments and analyses". In: *Pattern Recognition* 74 (2018), pp. 406–421.

[35] W. E. Donath and A. J. Hoffman. "Lower Bounds for the Partitioning of Graphs". In: *IBM Journal of Research and Development* 17.5 (1973), pp. 420–425. DOI: 10.1147/rd.175.0420.

[36] S. A. Dudani. "The distance-weighted k-nearest-neighbor rule". In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-6.4 (1976), pp. 325–327.

[37] J. Dumas, A. Wehenkel, D. Lanaspeze, B. Cornélusse, and A. Sutera. "A deep generative model for probabilistic energy forecasting in power systems: normalizing flows". In: *Applied Energy* 305 (2022), p. 117871. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2021.117871. URL: https://www.sciencedirect.com/science/article/pii/S0306261921011909.

[38] S. Džeroski, V. Gjorgjioski, I. Slavkov, and J. Struyf. "Analysis of Time Series Data with Predictive Clustering Trees". In: *Knowledge Discovery in Inductive Databases*. Ed. by S. Džeroski and J. Struyf. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 63–80. ISBN: 978-3-540-75549-4.

[39] S. Elmallah, A. M. Brockway, and D. Callaway. "Can distribution grid infrastructure accommodate residential electrification and electric vehicle adoption in Northern California?" In: *Environmental Research: Infrastructure and Sustainability* 2.4 (Nov. 2022), p. 045005. DOI: 10.1088/2634-4505/ac949c. URL: https://dx.doi.org/10.1088/2634-4505/ac949c.

[40] A. Emmott, S. Das, T. Dietterich, A. Fern, and W.-K. Wong. "A Meta-Analysis of the Anomaly Detection Problem". In: (Mar. 3, 2015). arXiv: 1503.01158v2 [cs.AI].

[41] A. F. Emmott, S. Das, T. Dietterich, A. Fern, and W.-K. Wong. "Systematic construction of anomaly detection benchmarks from real data". In: *Proceedings of the ACM SIGKDD workshop on outlier detection and description*. 2013, pp. 16–21.

[42] E. Eskin. "Anomaly Detection over Noisy Data using Learned Probability Distributions". In: *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*. Ed. by P. Langley. Morgan Kaufmann, 2000, pp. 255–262.

[43] European Commission. *Energy and the Green Deal*. https://commission.europa.eu/strategy-and-policy/priorities-2019-2024/european-green-deal/energy-and-green-deal_en. [Online; accessed 28-Mar-2023]. 2023.

[44] European Parliament. *EU responses to climate change*. https://www.europarl.europa.eu/news/en/headlines/society/20180703ST007129/eu-responses-to-climate-change. [Online; accessed 14-Mar-2023]. 2022.

[45] T. Fawcett. "An introduction to ROC analysis". In: *Pattern recognition letters* 27.8 (2006), pp. 861–874.

[46] L. Feremans, V. Vercruyssen, B. Cule, W. Meert, and B. Goethals. "Pattern-Based Anomaly Detection in Mixed-Type Time Series". In: *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2019, Würzburg, Germany, September 16-20, 2019, Proceedings, Part I*. Ed. by U. Brefeld, É. Fromont, A. Hotho, A. J. Knobbe, M. H. Maathuis, and C. Robardet. Vol. 11906. Lecture Notes in Computer Science. Springer, 2019, pp. 240–256. DOI: 10.1007/978-3-030-46150-8_15. URL: https://doi.org/10.1007/978-3-030-46150-8%5C_15.

[47] P. Flach and M. Kull. "Precision-Recall-Gain Curves: PR Analysis Done Right". In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett. Vol. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/33e8075e9970de0cfea955afd4644bb2-Paper.pdf.

[48] S. García, A. Fernández, J. Luengo, and F. Herrera. "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power". In: *Information Sciences* 180.10 (2010), pp. 2044–2064. DOI: 10.1016/j.ins.2009.12.010.

[49] L. Ge, W. Liao, S. Wang, B. Bak-Jensen, and J. R. Pillai. "Modeling Daily Load Profiles of Distribution Network for Scenario Generation Using Flow-Based Generative Network". In: *IEEE Access* 8 (2020), pp. 77587–77597. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2989350.

[50] T. Gneiting, L. Stanberry, E. Grimit, L. Held, and N. Johnson. "Assessing probabilistic forecasts of multivariate quantities, with an application to ensemble predictions of surface winds". In: *TEST* 17.2 (July 22, 2008), pp. 211–235. DOI: 10.1007/s11749-008-0114-x.

[51] I. Golan and R. El-Yaniv. "Deep Anomaly Detection Using Geometric Transformations". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/5e62d03aec0d17facfc5355dd90d441c-Paper.pdf.

[52] M. Goldstein and A. Dengel. "Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm". In: *KI-2012: Poster&Demo Track* (2012), pp. 59–63.

[53] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[54] P. Gopalan, V. Sharan, and U. Wieder. "PIDForest: Anomaly Detection via Partial Identification". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/eb6dc8aba23375061b6f07b137617096-Paper.pdf.

[55] S. Gottwalt, W. Ketter, C. Block, J. Collins, and C. Weinhardt. "Demand side management—A simulation of household behavior under variable prices". In: *Energy Policy* 39.12 (2011). Clean Cooking Fuels and Technologies in Developing Economies, pp. 8163–8174. ISSN: 0301-4215. DOI: https://doi.org/10.1016/j.enpol.2011.10.016. URL: https://www.sciencedirect.com/science/article/pii/S0301421511008007.

[56] A. Grandjean, J. Adnot, and G. Binet. “A review and an analysis of the residential electric load curve models”. In: *Renewable and Sustainable Energy Reviews* 16.9 (2012), pp. 6539–6565. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2012.08.013. URL: https://www.sciencedirect.com/science/article/pii/S1364032112004820.

[57] E. Gutflaish, A. Kontorovich, S. Sabato, O. Biller, and O. Sofer. “Temporal Anomaly Detection: Calibrating the Surprise”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (2019), pp. 3755–3762. DOI: 10.1609/aaai.v33i01.33013755. URL: https://ojs.aaai.org/index.php/AAAI/article/view/4261.

[58] S. Haben, J. Ward, D. Vukadinovic Greetham, C. Singleton, and P. Grindrod. “A new error measure for forecasts of household-level, high resolution electrical energy consumption”. In: *International Journal of Forecasting* 30.2 (2014), pp. 246–256. ISSN: 01692070. DOI: 10.1016/j.ijforecast.2013.08.002. URL: http://dx.doi.org/10.1016/j.ijforecast.2013.08.002.

[59] S. Hara and K. Hayashi. “Making Tree Ensembles Interpretable: A Bayesian Model Selection Approach”. In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. Ed. by A. Storkey and F. Perez-Cruz. Vol. 84. Proceedings of Machine Learning Research. PMLR, 2018, pp. 77–85. URL: https://proceedings.mlr.press/v84/hara18a.html.

[60] J. A. Hartigan and M. A. Wong. “Algorithm AS 136: A K-Means Clustering Algorithm”. In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28.1 (1979), pp. 100–108. ISSN: 00359254, 14679876. URL: http://www.jstor.org/stable/2346830 (visited on 08/07/2023).

[61] V. Hautamäki, I. Kärkkäinen, and P. Fränti. “Outlier Detection Using k-Nearest Neighbour Graph”. In: *17th International Conference on Pattern Recognition, ICPR 2004, Cambridge, UK, August 23-26, 2004*. IEEE Computer Society, 2004, pp. 430–433. DOI: 10.1109/ICPR.2004.1334558. URL: https://doi.org/10.1109/ICPR.2004.1334558.

[62] Z. He, X. Xu, and S. Deng. “Discovering cluster-based local outliers”. In: *Pattern Recognit. Lett.* 24.9-10 (2003), pp. 1641–1650. DOI: 10.1016/S0167-8655(03)00003-5.

[63] J. Ho, A. Jain, and P. Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG].

[64] J. Hodges, E. L. Lehmann, et al. “Rank methods for combination of independent experiments in analysis of variance”. In: *The Annals of Mathematical Statistics* 33.2 (1962), pp. 482–497.

[65] K.-j. Hsiao, K. Xu, J. Calder, and A. Hero. "Multi-criteria Anomaly Detection using Pareto Depth Analysis". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/1543843a4723ed2ab08e18053ae6dc5b-Paper.pdf.

[66] Y.-a. Huang, W. Fan, W. Lee, and P. Yu. "Cross-feature analysis for detecting ad-hoc routing anomalies". In: *23rd International Conference on Distributed Computing Systems, 2003. Proceedings.* May 2003, pp. 478–487. DOI: 10.1109/ICDCS.2003.1203498.

[67] L. Hubert and P. Arabie. "Comparing partitions". In: *Journal of Classification* 2 (1985), pp. 193–218.

[68] T. Iwata and M. Yamada. "Multi-view Anomaly Detection via Robust Probabilistic Latent Variable Models". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper/2016/file/0f96613235062963ccde717b18f97592-Paper.pdf.

[69] A. K. Jain. "Data clustering: 50 years beyond K-means". In: *Pattern Recognition Letters* 31.8 (2010). Award winning papers from the 19th International Conference on Pattern Recognition (ICPR), pp. 651–666. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2009.09.011. URL: https://www.sciencedirect.com/science/article/pii/S0167865509002323.

[70] J. Janssens. "Outlier selection and one-class classification". English. Series: TiCC Ph.D. Series Volume: 27. PhD thesis. 2013. ISBN: 9789082027310.

[71] B. J. Johnson, M. R. Starke, O. A. Abdelaziz, R. K. Jackson, and L. M. Tolbert. "A MATLAB based occupant driven dynamic model for predicting residential power demand". In: *2014 IEEE PES T&D Conference and Exposition*. 2014, pp. 1–5. DOI: 10.1109/TDC.2014.6863381.

[72] L. Kaufman and P. Rousseeuw. *Divisive Analysis (Program DIANA)". Finding Groups in Data: An Introduction to Cluster Analysis*. 6. Wiley, 1990, pp. 253–279. ISBN: 978-0-470-31748-8.

[73] S. Kaufman, S. Rosset, and C. Perlich. "Leakage in Data Mining: Formulation, Detection, and Avoidance". In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '11. San Diego, California, USA: Association for Computing Machinery, 2011, pp. 556–563. ISBN: 9781450308137. DOI:

10.1145/2020408.2020496. URL: https://doi.org/10.1145/2020408.2020496.

[74] D. P. Kingma and M. Welling. *Auto-Encoding Variational Bayes*. 2022. arXiv: 1312.6114 [stat.ML].

[75] D. Klein, S. D. Kamvar, and C. D. Manning. "From Instance-level Constraints to Space-Level Constraints: Making the Most of Prior Knowledge in Data Clustering". In: *International Conference on Machine Learning*. 2002. URL: https://api.semanticscholar.org/CorpusID:8968350.

[76] H. Kriegel, M. Schubert, and A. Zimek. "Angle-based outlier detection in high-dimensional data". In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*. Ed. by Y. Li, B. Liu, and S. Sarawagi. ACM, 2008, pp. 444–452. DOI: 10.1145/1401890.1401946. URL: https://doi.org/10.1145/1401890.1401946.

[77] W. Labeeuw and G. Deconinck. "Residential Electrical Load Model Based on Mixture Model Clustering and Markov Models". In: *IEEE Transactions on Industrial Informatics* 9.3 (Aug. 2013), pp. 1561–1569. ISSN: 1941-0050. DOI: 10.1109/TII.2013.2240309.

[78] L. Lelis and J. Sander. "Semi-supervised density-based clustering". In: *2009 Ninth IEEE International Conference on Data Mining*. IEEE. 2009, pp. 842–847.

[79] Y. Li, D. Zha, P. Venugopal, N. Zou, and X. Hu. "Pyodds: An end-to-end outlier detection system with automated machine learning". In: *Companion Proceedings of the Web Conference 2020*. 2020, pp. 153–157.

[80] S. Lin, A. Yurtman, J. Soenen, and H. Blockeel. "LinC: Explaining Time Series Clusterings with User-Provided Constraints". In: *Proceedings of the ECML'23 workshop on Explainable AI for Time Series*. 2023.

[81] F. T. Liu, K. M. Ting, and Z.-H. Zhou. "Isolation forest". In: *2008 Eighth IEEE Intl. Conference on Data Mining*. IEEE. 2008, pp. 413–422.

[82] S. Lu, L. Liu, J. Li, D. Le, and J. Liu. "LoPAD: A Local Prediction Approach to Anomaly Detection". In: *PAKDD 2020*. Vol. 12085. LNAI. Springer, 2020, pp. 660–673. DOI: 10.1007/978-3-030-47436-2_50.

[83] Y.-C. Lu, F. Chen, Y. Chen, and C.-T. Lu. "A Generalized Student-t Based Approach to Mixed-Type Anomaly Detection". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 27.1 (2013). URL: https://ojs.aaai.org/index.php/AAAI/article/view/8581.

[84] U. von Luxburg, R. C. Williamson, and I. Guyon. "Clustering: Science or Art?" In: *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. Vol. 27. Proceedings of Machine Learning Research. 2012, pp. 65–79.

[85] J. MacQueen. "Classification and analysis of multivariate observations". In: *5th Berkeley Symp. Math. Statist. Probability*. 1967, pp. 281–297.

[86] F. Maranzana. "On the location of supply points to minimize transportation costs". In: *IBM Systems Journal* 2.2 (1963), pp. 129–135. DOI: 10.1147/sj.22.0129.

[87] J. Matheson and R. Winkler. "Scoring Rules for Continuous Probability Distributions". In: *Management Science* 22.10 (June 1976), pp. 1087–1096. DOI: 10.1287/mnsc.22.10.1087.

[88] A. Mazumdar and B. Saha. "Clustering with noisy queries". In: *Advances in Neural Information Processing Systems*. 2017, pp. 5788–5799.

[89] F. McLoughlin, A. Duffy, and M. Conlon. "A clustering approach to domestic electricity load profile characterisation using smart metering data". In: *Applied Energy* 141 (2015), pp. 190–199. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2014.12.039. URL: https://www.sciencedirect.com/science/article/pii/S0306261914012963.

[90] C. Möhrlen, J. W. Zack, and G. Giebel. "Chapter Fourteen - Assessment of forecast performance". In: *IEA Wind Recommended Practice for the Implementation of Renewable Energy Forecasting Solutions*. Ed. by C. Möhrlen, J. W. Zack, and G. Giebel. Wind Energy Engineering. Academic Press, 2023, pp. 125–145. ISBN: 978-0-443-18681-3. DOI: https://doi.org/10.1016/B978-0-44-318681-3.00026-X. URL: https://www.sciencedirect.com/science/article/pii/B9780443186813000026X.

[91] A. Navarro-Espinosa and P. Mancarella. "Probabilistic modeling and assessment of the impact of electric heat pumps on low voltage distribution networks". In: *Applied Energy* 127 (2014), pp. 249–266. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2014.04.026. URL: https://www.sciencedirect.com/science/article/pii/S0306261914003786X.

[92] A. Navarro-Espinosa and L. F. Ochoa. "Probabilistic Impact Assessment of Low Carbon Technologies in LV Distribution Systems". In: *IEEE Transactions on Power Systems* 31.3 (May 2016), pp. 2192–2203. ISSN: 1558-0679. DOI: 10.1109/TPWRS.2015.2448663.

[93] M. Neaimeh, M. Deakin, R. Jenkinson, and O. Giles. "Democratizing electricity distribution network analysis". In: *Data-Centric Engineering* 4 (2023), e3. DOI: 10.1017/dce.2022.41.

[94] H. V. Nguyen, E. Müller, J. Vreeken, F. Keller, and K. Böhm. "CMI: An information-theoretic contrast measure for enhancing subspace cluster and outlier detection". In: *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM. 2013, pp. 198–206.

[95] K. Noto, C. Brodley, and D. Slonim. "FRaC: a feature-modeling approach for semi-supervised and unsupervised anomaly detection". In: *Data mining and knowledge discovery* 25.1 (2012), pp. 109–133.

[96] K. Noto, C. Brodley, and D. Slonim. "Anomaly Detection Using an Ensemble of Feature Models". In: *2010 IEEE International Conference on Data Mining*. Dec. 2010, pp. 953–958. DOI: 10.1109/ICDM.2010.140.

[97] G. Pang, L. Cao, L. Chen, D. Lian, and H. Liu. "Sparse Modeling-Based Sequential Ensemble Learning for Effective Outlier Detection in High-Dimensional Numeric Data". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (2018). URL: https://ojs.aaai.org/index.php/AAAI/article/view/11692.

[98] S. Patel, S. Sihmar, and A. Jatain. "A study of hierarchical clustering algorithms". In: *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*. 2015, pp. 537–541.

[99] H. Paulheim and R. Meusel. "A decomposition of the outlier detection problem into a set of supervised learning problems". In: *Machine Learning* 100.2-3 (2015), pp. 509–531.

[100] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[101] D. Pelleg and D. Baras. "K-Means with Large and Noisy Constraint Sets". In: *Machine Learning: ECML 2007*. 2007, pp. 674–682.

[102] F. Petitjean, A. Ketterlin, and P. Gançarski. "A global averaging method for dynamic time warping, with applications to clustering". In: *Pattern Recognition* 44.3 (2011), pp. 678–693. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2010.09.013. URL: https://www.sciencedirect.com/science/article/pii/S003132031000453X.

[103] T. Pevnỳ. "Loda: Lightweight on-line detector of anomalies". In: *Machine Learning* 102.2 (2016), pp. 275–304.

[104] E. Proedrou. "A Comprehensive Review of Residential Electricity Load Profile Models". In: *IEEE Access* 9 (2021), pp. 12114–12133. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3050074.

[105] J. R. Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[106] J. R. Quinlan. "Simplifying decision trees". In: *Int. J. Man Mach. Stud.* 27 (1987), pp. 221–234.

[107] S. Ramaswamy, R. Rastogi, and K. Shim. "Efficient Algorithms for Mining Outliers from Large Data Sets". In: *SIGMOD Rec.* 29.2 (May 2000), pp. 427–438. ISSN: 0163-5808. DOI: 10.1145/335191.335437. URL: https://doi.org/10.1145/335191.335437.

[108] S. S. Rangapuram and M. Hein. "Constrained 1-Spectral Clustering." In: *AISTATS*. Vol. 30. 2012, p. 90.

[109] M. Reis, A. Garcia, and R. J. Bessa. "A scalable load forecasting system for low voltage grids". In: *2017 IEEE Manchester PowerTech*. June 2017, pp. 1–6. DOI: 10.1109/PTC.2017.7980936.

[110] D. J. Rezende and S. Mohamed. *Variational Inference with Normalizing Flows*. 2016. arXiv: 1505.05770 [stat.ML].

[111] S. Roberts and L. Tarassenko. "A probabilistic resource allocating network for novelty detection". In: *Neural Computation* 6.2 (1994), pp. 270–284.

[112] T. Saito and M. Rehmsmeier. "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets". In: *PLOS ONE* 10.3 (Mar. 2015), pp. 1–21. DOI: 10.1371/journal.pone.0118432. URL: https://doi.org/10.1371/journal.pone.0118432.

[113] H. Sakoe and S. Chiba. "Dynamic programming algorithm optimization for spoken word recognition". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26.1 (1978), pp. 43–49. DOI: 10.1109/TASSP.1978.1163055.

[114] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan. "Finding a "Kneedle" in a Haystack: Detecting Knee Points in System Behavior". In: *2011 31st International Conference on Distributed Computing Systems Workshops*. June 2011, pp. 166–171. DOI: 10.1109/ICDCSW.2011.20.

[115] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. "Estimating the support of a high-dimensional distribution". In: *Neural computation* 13.7 (2001), pp. 1443–1471.

[116] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, J. C. Platt, et al. "Support vector method for novelty detection." In: *NIPS*. Vol. 12. Citeseer. 1999, pp. 582–588.

[117] E. Schubert and P. J. Rousseeuw. "Fast and eager k-medoids clustering: O(k) runtime improvement of the PAM, CLARA, and CLARANS algorithms". In: *Information Systems* 101 (2021), p. 101804. ISSN: 0306-4379. DOI: https://doi.org/10.1016/j.is.2021.101804. URL: https://www.sciencedirect.com/science/article/pii/S0306437921000557.

[118] E. Schubert, A. Zimek, and H. Kriegel. "Generalized Outlier Detection with Flexible Kernel Density Estimates". In: *Proceedings of the 2014 SIAM International Conference on Data Mining, Philadelphia, Pennsylvania, USA, April 24-26, 2014*. Ed. by M. J. Zaki, Z. Obradovic, P. Tan, A. Banerjee, C. Kamath, and S. Parthasarathy. SIAM, 2014, pp. 542–550. DOI: 10.1137/1.9781611973440.63. URL: https://doi.org/10.1137/1.9781611973440.63.

[119] G. Schwarz. "Estimating the Dimension of a Model". In: *The Annals of Statistics* 6.2 (1978), pp. 461–464. DOI: 10.1214/aos/1176344136. URL: https://doi.org/10.1214/aos/1176344136.

[120] D. W. Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[121] X.-R. Sheng, D.-C. Zhan, S. Lu, and Y. Jiang. "Multi-View Anomaly Detection: Neighborhood in Locality Matters". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (2019), pp. 4894–4901. DOI: 10.1609/aaai.v33i01.33014894. URL: https://ojs.aaai.org/index.php/AAAI/article/view/4418.

[122] H. Shi, M. Xu, and R. Li. "Deep Learning for Household Load Forecasting—A Novel Pooling Deep RNN". In: *IEEE Transactions on Smart Grid* 9.5 (Sept. 2018), pp. 5271–5280. ISSN: 1949-3061. DOI: 10.1109/TSG.2017.2686012.

[123] J. Shi and J. Malik. "Normalized cuts and image segmentation". In: *Departmental Papers (CIS)* (2000), p. 107.

[124] R. Sibson. "SLINK: an optimally efficient algorithm for the single-link cluster method". In: *The computer journal* 16.1 (1973), pp. 30–34.

[125] J. Soenen, S. Dumančić, T. Van Craenendonck, and H. Blockeel. "Tackling Noise in Active Semi-supervised Clustering". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by F. Hutter, K. Kersting, J. Lijffijt, and I. Valera. Cham: Springer International Publishing, 2021, pp. 121–136. ISBN: 978-3-030-67661-2.

[126] J. Soenen, A. Yurtman, T. Becker, R. D'hulst, K. Vanthournout, W. Meert, and H. Blockeel. "Scenario generation of residential electricity consumption through sampling of historical data". In: *Sustainable Energy, Grids and Networks* 34 (2023), p. 100985. ISSN: 2352-4677. DOI: https://doi.org/10.1016/j.segan.2022.100985. URL: https://www.sciencedirect.com/science/article/pii/S2352467722002302.

[127] J. Soenen, A. Yurtman, T. Becker, K. Vanthournout, and H. Blockeel. "Generating electricity consumption scenarios with predictive clustering trees". In: *In preparation* (2023).

[128] J. Soenen*, E. Van Wolputte*, L. Perini, V. Vercruyssen, W. Meert, J. Davis, and H. Blockeel. "The effect of hyperparameter tuning on the comparative evaluation of unsupervised anomaly detection methods". In: *Proceedings of the KDD'21 Workshop on Outlier Detection and Description*. 2021, pp. 1–9.

[129] J. Soenen*, E. V. Wolputte*, V. Vercruyssen, W. Meert, and H. Blockeel. *AD-MERCS: Modeling Normality and Abnormality in Unsupervised Anomaly Detection*. 2023. arXiv: 2305.12958 [cs.LG].

[130] A. Speakman, O. Harris, C. Birkinshaw-Doyle, and D. Mills. *Low voltage network capacity study*. Tech. rep. Accessed: 2023-04-19. Element Energy and EA Technology, GOV.UK, 2022. URL: https://www.gov.uk/government/publications/low-voltage-network-capacity-study.

[131] G. Steinbuss and K. Böhm. "Benchmarking Unsupervised Outlier Detection with Realistic Synthetic Data". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 15.4 (2021), pp. 1–20.

[132] J. Struyf and S. Dzeroski. "Clustering Trees with Instance Level Constraints". In: *Machine Learning: ECML 2007, 18th European Conference on Machine Learning, Warsaw, Poland, September 17-21, 2007, Proceedings*. Ed. by J. N. Kok, J. Koronacki, R. L. de Mántaras, S. Matwin, D. Mladenic, and A. Skowron. Vol. 4701. Lecture Notes in Computer Science. Springer, 2007, pp. 359–370. DOI: 10.1007/978-3-540-74958-5_34. URL: https://doi.org/10.1007/978-3-540-74958-5%5C_34.

[133] J. Struyf and S. Džeroski. "Constrained Predictive Clustering". In: *Inductive Databases and Constraint-Based Data Mining*. Ed. by S. Džeroski, B. Goethals, and P. Panov. New York, NY: Springer New York, 2010, pp. 155–175. ISBN: 978-1-4419-7738-0. DOI: 10.1007/978-1-4419-7738-0_7. URL: https://doi.org/10.1007/978-1-4419-7738-0_7.

[134] M. Sugiyama and K. Borgwardt. "Rapid Distance-Based Outlier Detection via Sampling". In: *Advances in Neural Information Processing Systems*. Ed. by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper/2013/file/d296c101daa88a51f6ca8cfc1ac79b50-Paper.pdf.

[135] L. G. Swan and V. I. Ugursal. "Modeling of end-use energy consumption in the residential sector: A review of modeling techniques". In: *Renewable and Sustainable Energy Reviews* 13.8 (2009), pp. 1819–1835. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2008.09.033. URL: https://www.sciencedirect.com/science/article/pii/S1364032108001949.

[136] L. Takács. "Some asymptotic formulas for lattice paths". In: *Journal of statistical planning and inference* 14.1 (1986), pp. 123–142.

[137] The White House. *Inflation Reduction Act Guidebook*. https://www.whitehouse.gov/cleanenergy/inflation-reduction-act-guidebook/. [Online; accessed 14-Mar-2023]. 2022.

[138] Tindemans, S., Strbac, G., Schofield, J. R, Woolf, M., Carmichael, R., Bilton, M. *Low Carbon London Project: Data from the Dynamic Time-of-Use Electricity Pricing Trial, 2013*. [data collection] SN: 7857. http://doi.org/10.5255/UKDA-SN-7857-2. UK Data Service, 2016.

[139] J.-F. Toubeau, M. Hupez, V. Klonari, Z. De Grève, and F. Vallée. "Statistical load and generation modelling for long term studies of low voltage networks in presence of sparse smart metering data". In: *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*. 2016, pp. 3900–3905. DOI: 10.1109/IECON.2016.7793228.

[140] H. Trittenbach and K. Böhm. "One-Class Active Learning for Outlier Detection with Multiple Subspaces". In: *Proceedings of the 28th ACM Intl. Conference on Information and Knowledge Management*. ACM. 2019, pp. 811–820.

[141] H. Trittenbach, K. Böhm, and I. Assent. "Active Learning of SVDD Hyperparameter Values". In: *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE. 2020, pp. 109–117.

[142] H. Trittenbach, A. Englhardt, and K. Böhm. "An overview and a benchmark of active learning for outlier detection with one-class classifiers". In: *Expert Systems with Applications* (2020), p. 114372.

[143] U.S. Department of Energy Office of Policy. *The Inflation Reduction Act Drives Significant Emissions Reductions and Positions America to Reach Our Climate Goals*. https://www.energy.gov/sites/default/files/2022-08/8.18%20InflationReductionAct_Factsheet_Final.pdf. [Online; accessed 14-Mar-2023]. 2022.

[144] T. Van Craenendonck, S. Dumančić, E. V. Wolputte, and H. Blockeel. "COBRAS: Interactive Clustering with Pairwise Queries". In: *Proceedings of the 17th International Symposium on Intelligent Data Analysis*. Springer, 2018.

[145] E. Van Wolputte, E. Korneva, and H. Blockeel. "MERCS: multi-directional ensembles of regression and classification trees". In: *32nd AAAI Conference on Artificial Intelligence*. 2018.

[146] E. Veldman and R. A. Verzijlbergh. "Distribution Grid Impacts of Smart Electric Vehicle Charging From Different Perspectives". In: *IEEE Transactions on Smart Grid* 6.1 (Jan. 2015), pp. 333–342. ISSN: 1949-3061. DOI: 10.1109/TSG.2014.2355494.

[147] V. Vercruyssen, W. Meert, and J. Davis. "Transfer Learning for Anomaly Detection through Localized and Unsupervised Instance Selection". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.04 (2020), pp. 6054–6061. DOI: 10.1609/aaai.v34i04.6068. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6068.

[148] V. Vercruyssen, W. Meert, G. Verbruggen, K. Maes, R. Bäumer, and J. Davis. "Semi-Supervised Anomaly Detection with an Application to Water Analytics". In: *2018 IEEE International Conference on Data Mining (ICDM)*. Nov. 2018, pp. 527–536. DOI: 10.1109/ICDM.2018.00068.

[149] S. Vitiello, N. Andreadou, M. Ardelean, and G. Fulli. "Smart Metering Roll-Out in Europe: Where Do We Stand? Cost Benefit Analyses in the Clean Energy Package and Research Trends in the Green Deal". In: *Energies* 15.7 (2022). ISSN: 1996-1073. DOI: 10.3390/en15072340. URL: https://www.mdpi.com/1996-1073/15/7/2340.

[150] U. Von Luxburg. "A tutorial on spectral clustering". In: *Statistics and computing* 17 (2007), pp. 395–416.

[151] J. Vossen, B. Feron, and A. Monti. "Probabilistic Forecasting of Household Electrical Load Using Artificial Neural Networks". In: *2018 IEEE International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*. June 2018, pp. 1–6. DOI: 10.1109/PMAPS.2018.8440559.

[152] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl, et al. "Constrained k-means clustering with background knowledge". In: *Icml*. Vol. 1. 2001, pp. 577–584.

[153] L. Xiong, B. Póczos, and J. Schneider. "Group Anomaly Detection using Flexible Genre Models". In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger. Vol. 24. Curran Associates, Inc., 2011. URL: https://proceedings.neurips.cc/paper/2011/file/eaae339c4d89fc102edd9dbdb6a28915-Paper.pdf.

[154] S. Xiong, J. Azimi, and X. Z. Fern. "Active learning of constraints for semi-supervised clustering". In: *IEEE Transactions on Knowledge and Data Engineering* 26.1 (2013), pp. 43–54.

[155] R. Xu and D. Wunsch. "Survey of clustering algorithms". In: *IEEE Transactions on Neural Networks* 16.3 (2005), pp. 645–678. DOI: 10.1109/TNN.2005.845141.

[156] X. Xu, H. Liu, and M. Yao. "Recent progress of anomaly detection". In: *Complexity* 2019 (2019).

[157] M.-S. Yang. "A survey of fuzzy clustering". In: *Mathematical and Computer Modelling* 18.11 (1993), pp. 1–16. ISSN: 0895-7177. DOI: https://doi.org/10.1016/0895-7177(93)90202-A. URL: https://www.sciencedirect.com/science/article/pii/089571779390202A.

[158] T. Yang, R. Jin, and A. K. Jain. "Learning from Noisy Side Information by Generalized Maximum Entropy Model". In: *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*. 2010, pp. 1199–1206.

[159] Y. Ye, M. Strong, Y. Lou, C. A. Faulkner, W. Zuo, and S. Upadhyaya. "Evaluating performance of different generative adversarial networks for large-scale building power demand prediction". In: *Energy and Buildings* 269 (2022), p. 112247. ISSN: 0378-7788. DOI: https://doi.org/10.1016/j.enbuild.2022.112247. URL: https://www.sciencedirect.com/science/article/pii/S0378778822004182.

[160] A. Yurtman, J. Soenen, W. Meert, and H. Blockeel. "Estimating Dynamic Time Warping Distance Between Time Series with Missing Data". In: *Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*. 2023.

[161] C. Zeinstra, R. Veldhuis, and L. Spreeuwers. "How Random is a Classifier given its Area under Curve?" In: *2017 International Conference of the Biometrics Special Interest Group (BIOSIG)*. IEEE. 2017, pp. 1–4.

[162] M. Zhao and V. Saligrama. "Anomaly detection with score functions based on nearest neighbor graphs". In: *arXiv preprint arXiv:0910.5461* (2009).

[163] Y. Zhao, Z. Nasrullah, and Z. Li. "PyOD: A Python Toolbox for Scalable Outlier Detection". In: *Journal of Machine Learning Research* 20.96 (2019), pp. 1–7. URL: http://jmlr.org/papers/v20/19-011.html.

[164] L. Zheng and T. Li. "Semi-supervised hierarchical clustering". In: *2011 IEEE 11th international conference on data mining*. IEEE. 2011, pp. 982–991.

[165] X. Zhu, C. C. Loy, and S. Gong. "Constrained Clustering With Imperfect Oracles". In: *IEEE Transactions on Neural Networks and Learning Systems* 27.6 (2016), pp. 1345–1357.

Curriculum Vitae

Jonas Soenen was born in Leuven on August 19 1996. He obtained a bachelors in Computer Science (2017) and masters engineering in Computer Science (2019) from the KU Leuven. In September 2019, Jonas started his PhD under the supervision of Prof. dr. ir. Hendrik Blockeel in the Machine learning research group at the Computer Science department of KU Leuven. His PhD focused on semi-supervised clustering and explainable anomaly detection. During his PhD he also collaborated with EnergyVille to apply machine learning techniques to plan the reinforcement of the Low Voltage Grid.

List of Publications

Peer-reviewed journal publications in chronological order

- **(in preparation)** J. Soenen, A. Yurtman, T. Becker, K. Vanthournout, and H. Blockeel. "Generating electricity consumption scenarios with predictive clustering trees". In: *In preparation* (2023)
- J. Soenen, A. Yurtman, T. Becker, R. D'hulst, K. Vanthournout, W. Meert, and H. Blockeel. "Scenario generation of residential electricity consumption through sampling of historical data". In: *Sustainable Energy, Grids and Networks* 34 (2023), p. 100985. ISSN: 2352-4677. DOI: https://doi.org/10.1016/j.segan.2022.100985. URL: https://www.sciencedirect.com/science/article/pii/S2352467722002302
- L. Botman, J. Soenen, K. Theodorakos, A. Yurtman, J. Bekker, K. Vanthournout, H. Blockeel, B. De Moor, and J. Lago. "A Scalable Ensemble Approach to Forecast the Electricity Consumption of Households". In: *IEEE Transactions on Smart Grid* 14.1 (2023), pp. 757–768. DOI: 10.1109/TSG.2022.3191399

Peer-reviewed conference publications in chronological order

- A. Yurtman, J. Soenen, W. Meert, and H. Blockeel. "Estimating Dynamic Time Warping Distance Between Time Series with Missing Data". In: *Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*. 2023
- J. Soenen, S. Dumančić, T. Van Craenendonck, and H. Blockeel. "Tackling Noise in Active Semi-supervised Clustering". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by F. Hutter, K. Kersting, J. Lijffijt, and I. Valera. Cham: Springer International Publishing, 2021, pp. 121–136. ISBN: 978-3-030-67661-2

Peer-reviewed workshop publications in chronological order

- S. Lin, A. Yurtman, J. Soenen, and H. Blockeel. "LinC: Explaining Time Series Clusterings with User-Provided Constraints". In: *Proceedings of the ECML'23 workshop on Explainable AI for Time Series*. 2023
- J. Soenen*, E. Van Wolputte*, L. Perini, V. Vercruyssen, W. Meert, J. Davis, and H. Blockeel. "The effect of hyperparameter tuning on the comparative evaluation of unsupervised anomaly detection methods". In: *Proceedings of the KDD'21 Workshop on Outlier Detection and Description*. 2021, pp. 1–9



FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
DECLARATIVE LANGUAGES AND ARTIFICIAL INTELLIGENCE
Celestijnenlaan 200A box 2402
B-3001 Leuven
jonas.soenen@cs.kuleuven.be
http://www.dtai.cs.kuleuven.be