



Operational Decision-Making with Machine Learning and Causal Inference

Dissertation presented to obtain the joint degree of
Doctor in Business Economics at KU Leuven and
Doctor in Mathematics at the University of Antwerp

by

Toon Vanderschueren

Since the theses in the series published by the Faculty of Economics and Business are the personal work of their authors, only the latter bear full responsibility.

DOCTORAL COMMITTEE

Promotors

Prof. dr. Wouter Verbeke

KU Leuven

Prof. dr. Tim Verdonck

University of Antwerp

Co-Promotor

Prof. dr. Bart Baesens

KU Leuven

Doctoral committee

Prof. dr. Tias Guns

KU Leuven

Prof. dr. David Martens

University of Antwerp

Prof. dr. Mihaela van der Schaar

University of Cambridge

Chair

Prof. dr. Martina Vandebroek

KU Leuven

ACKNOWLEDGEMENTS

The past four years have been quite a journey. I began my PhD during a lockdown in Leuven, spent some time living in Cambridge and Amsterdam, and traveled to places like Hawaii, Copenhagen, and Turin. Throughout these experiences, I've had the privilege of meeting and spending time with some truly amazing people. I would like to take a moment to thank them here.

First and foremost, thank you Wouter, for your unwavering support, invaluable advice, and patient guidance over the past few years. Your positivity and enthusiasm have been a constant source of motivation, and I've learned so much from you—both as a researcher and a person. You've always encouraged me to pursue my own ideas and provided me the freedom and opportunities to do so, while also offering direction when necessary. It has been a genuine pleasure to work with you and get to know you. I already look forward to drinking many more Chimay Bleues with you in the future!

One of the great benefits of my joint PhD was that I had not one, but two fantastic promotors. Thank you, Tim, for your kindness, good humor, and for always finding time for me despite your busy schedule. I've always looked forward to our meetings, whether online or over lunch during my visits to Antwerp. Your unwavering support, thoughtful feedback, and constant willingness to brainstorm new ideas have been invaluable throughout my PhD.

I also had the pleasure of having a wonderful co-promotor that I've enjoyed collaborating with tremendously: thank you, Bart, for your insightful advice and great sense of humor. I fondly remember the first year of my PhD, during the lockdown, when it was just you, Wouter, and me in the office. Your experience and pragmatism have been incredibly helpful. Thank you also for regularly checking in, both about research and life in general.

I would like to extend my sincere gratitude to the members of my stellar

committee. Special thanks to Mihaela for welcoming me to Cambridge. Your energy and vision have been truly inspiring, and I'm grateful that we've been able to continue our collaboration afterward. Thank you, Tias and David, for being part of my committee. Your helpful feedback, constructive comments, and valuable suggestions have been greatly appreciated. I would also like to thank Prof. Martina Vandebroek for kindly agreeing to chair my committee.

Over the past years, I've had opportunities to collaborate with some truly amazing people. Thank you to my co-authors: Simon, Robert, Chris, Jeroen, and Alicia. I'm deeply grateful to have had the privilege of working with you. Not only are you all incredibly smart and driven, but you're also a genuine pleasure to be around. I would also like to thank all my wonderful colleagues at Booking.com. I already miss my days in Amsterdam—not just the amazing coffees, lunches and gyms, but especially the people I shared them with! Special thanks to Hugo and Felipe, for being such fantastic colleagues and inspiring mentors, guiding me in terms of research, corporate life, and personal growth. Finally, I'm incredibly grateful for the support at BNP Paribas Fortis: thank you Siert, Ina, Manuel, Bart, and everyone else.

My research group, LIRIS, has been a constant source of support and friendship over the years, from coffee breaks to conferences and happy hours. Special thanks to everyone whom I had the honor of sharing an office with: Alexander, Boje, Bruno, Daan, and Jakob. Thank you also to Philipp, Yannis, Charlotte, Alexandre, Jari, Carlos, Hans, Veda, Ziboud, Raf, Björn, Jente, Elena, Manon, Margot, Felix, Brecht, Yanyi, Marco, and Yongbo.

Finally, I am forever indebted to my amazing friends and family. Thank you to my extraordinary friends for the countless memories we've shared over the years. We have gone from spending time with each other every day at the Bayo, Pizza Mansion, Krek, Bar Tam and Popol, to being spread across different cities and even countries. Even so, it always feels like we just saw each other yesterday, even after a few months apart. I would also like to express my heartfelt gratitude to my family, especially my parents, for their persistent encouragement and unwavering support. Last but not least, thank you, Cloë, for everything—your patience, kindness, and love. This was only possible with you, and I wouldn't want to do it any other way.

Toon Vanderschueren

SUMMARY

Optimizing operational decisions, routine actions within some business or operational process, is a key challenge across a variety of domains and application areas. The increasing availability of data, computational power, and advanced machine learning (ML) algorithms offers exciting opportunities for *data-driven* decision support. To advance the potential of ML for optimizing operational decision-making, we explore two research directions, aiming to develop ML models that are *decision-focused* and *causal*. This dissertation presents several developments in machine learning in these two areas.

ML is effective at making predictions from historical data: for example, estimating a transaction's fraud probability by comparing it to past cases. However, decision-makers not only need to consider these predictions, but also the operational context. For example, the decision-maker uses predicted fraud probabilities to determine which transactions to investigate, while aiming to minimize monetary losses due to fraud and considering the available capacity of the fraud investigations team. Predictions can help reduce uncertainty (e.g., by predicting the fraud probability), but standard ML models are prediction-focused, instead of decision-focused. This distinction involves two challenges for data-driven decision-making. First, prediction-focused models prioritize predictive accuracy instead of the resulting decision quality (e.g., fraud losses recovered by the bank). Second, these models fail to account for operational constraints, such as the available investigation capacity. *Decision-focused learning* aims to improve data-driven decision-making by addressing these issues and incorporating the operational context into the optimization of ML models. In this dissertation, we analyze cost-sensitive learning within this prediction-optimization framework and evaluate general strategies for making cost-optimal decisions with ML. Additionally, we propose a novel ML method for optimal decision-making under capacity constraints based on learning to rank.

To make effective decisions, a decision-maker has to estimate the causal effect of possible interventions in order to choose actions that achieve the desired outcome. Unfortunately, standard ML models identify correlations in the data instead of causal relationships. Because of this, these models cannot guarantee the effectiveness of decisions made based on their predictions. *Causal inference* provides a formal framework for reasoning about causality and identifying causal effects from data. This dissertation explores the intersection of causality and ML. First, we illustrate the potential of causal ML for optimizing preventive maintenance. Next, we propose novel causal ML methods for predicting causal effect distributions and for addressing informative sampling when predicting treatment outcomes over time. We also argue for a practical, end-to-end perspective for building ML pipelines for causal inference and propose an automated framework doing so. Finally, we combine decision-focused learning with causal inference by introducing ranking metalearners to optimize treatment decisions under capacity constraints.

CONTENTS

Doctoral Committee	v
Acknowledgements	vii
Summary	xi
I Prologue	1
1 Introduction	3
1.1 Motivating Examples and Applications	4
1.2 Decision-Focused Learning	6
1.3 Causal Inference	7
1.4 Contributions and Outline	8
II Decision-Focused Learning	13
2 Predict-then-Optimize or Predict-and-Optimize? An Empirical Evaluation of Cost-sensitive Learning Strategies	15
2.1 Introduction	16
2.2 Related work	17
2.2.1 Types of costs	19
2.2.2 Cost-sensitive classification in the predict-and-optimize framework	20
2.3 Methodology	24
2.3.1 Experimental design	25
2.3.2 Experimental procedure and evaluation metrics	27

2.4	Empirical results	29
2.4.1	Data	29
2.4.2	Results	30
2.5	Discussion	35
2.6	Conclusion	39
3	Optimally Allocating Limited Resources to Uncertain Tasks	41
3.1	Introduction	42
3.2	Related work	43
3.2.1	Uncertainty in assignment problems	43
3.2.2	Predict-and-optimize	44
3.2.3	Classification	44
3.2.4	Learning to rank	45
3.3	Problem formulation	45
3.4	Methodology	48
3.4.1	Two-stage predict-then-optimize	48
3.4.2	Integrated predict-and-optimize using learning to rank	51
3.5	Empirical results	52
3.5.1	Data	53
3.5.2	Results	55
3.6	Conclusion	57
III	Causal Inference	61
4	Optimizing the Preventive Maintenance Frequency with Causal Machine Learning	63
4.1	Introduction	64
4.2	Related work	66
4.2.1	Time-based maintenance	66
4.2.2	Imperfect maintenance	66
4.2.3	Condition-based maintenance	67
4.2.4	Prescriptive analytics and causal inference	68
4.3	Problem overview	69
4.4	Methodology	72
4.4.1	Assumptions	72
4.4.2	Predictive model to estimate the effect of the PM frequency on overhaul and failure rates	74
4.4.3	Optimization of the maintenance cost	74
4.5	Results	75
4.5.1	Data	75
4.5.2	Semi-synthetic data generating procedure	75
4.5.3	Performance evaluation	79

4.5.4	Empirical results	81
4.6	Conclusion	85
4.6.1	Limitations	86
4.6.2	Managerial implications	87
4.6.3	Future work	87
5	NOFLITE: Learning to Predict Individual Treatment Effect Distributions	89
5.1	Introduction	90
5.2	Related work	91
5.2.1	Treatment effect estimation	91
5.2.2	Normalizing flows	93
5.3	Problem Formulation	94
5.4	NOFLITE: Estimating ITE distributions using normalizing flows	96
5.4.1	Architecture	96
5.4.2	Optimization	98
5.4.3	Inference	99
5.5	Results	99
5.5.1	Data and benchmarks	100
5.5.2	Performance metrics	101
5.5.3	Empirical results	102
5.6	Conclusion	103
6	Accounting for Informative Sampling When Learning to Forecast Treatment Outcomes Over Time	107
6.1	Introduction	108
6.2	Problem Formalization	111
6.2.1	Problem structure: Complete versus observed data . .	111
6.2.2	Distinguishing between different sampling patterns . .	112
6.3	Goals, Assumptions and Inherent Challenges	113
6.3.1	Goal: Forecasting treatment outcomes	113
6.3.2	Identifying assumptions	114
6.3.3	What makes learning CAPOs from observational data challenging?	116
6.4	Learning to Forecast Under Informative Sampling	117
6.4.1	Learning To Forecast Using Inverse Intensity Weights	117
6.4.2	TESAR-CDE: Forecasting with Intensity-weighted Neural CDEs	118
6.5	Results	122
6.5.1	Simulation: lung cancer treatment	122
6.5.2	Empirical results	123
6.6	Conclusion	125

7	AutoCATE: Towards End-to-End, Automated Treatment Effect Estimation	127
7.1	Introduction	128
7.2	Related Work	129
7.2.1	Automated Machine Learning (AutoML)	129
7.2.2	Treatment Effect Estimation and Model Validation	130
7.3	Problem Formulation	131
7.4	AutoCATE: End-To-End, Automated CATE Estimation	132
7.4.1	Stage 1: Evaluation—Designing a Proxy Risk and Evaluation Protocol	133
7.4.2	Stage 2: Estimation—Building a CATE Estimation Pipeline	134
7.4.3	Stage 3: Ensembling—Selecting and Ensembling Estimation Pipelines	135
7.4.4	ML Pipeline Building Blocks: Preprocessing and ML Baselearners	135
7.4.5	Low-Code CATE Estimation Through AutoCATE’s API	136
7.5	Empirical Evaluation: Comparing Automated Strategies	136
7.5.1	Experimental Setup: Data and Evaluation Metrics	136
7.5.2	Analyzing AutoCATE—Stage 1: Evaluation Protocol	137
7.5.3	Analyzing AutoCATE—Stage 2: Estimation Protocol	139
7.5.4	Analyzing AutoCATE—Stage 3: Ensembling Protocol	140
7.5.5	Benchmarking AutoCATE Against Common Alternatives	141
7.6	Conclusion	142
IV	Decision-Focused Causal Inference	145
8	Metalearners for Ranking Treatment Effects	147
8.1	Introduction	148
8.2	Related Work	150
8.2.1	Prediction-Focused Learning: Effect Estimation	150
8.2.2	Decision-Focused Learning: Effect Ranking	151
8.3	Problem Formulation	153
8.3.1	Notation and Optimization Problem	153
8.3.2	Assumptions	154
8.3.3	Evaluating a Treatment Policy	154
8.4	Methodology	154
8.4.1	Optimizing a Ranking Objective	155
8.4.2	Ranking Metalearners	158
8.5	Empirical Results	161
8.5.1	Data and Benchmarks	162

8.5.2	Comparing Performance for the Different Objectives and Metalearners (RQ1)	163
8.5.3	Analyzing Alternative Metrics (RQ2) and Design Choices (RQ3)	164
8.6	Conclusion	165
V	Epilogue	169
9	Conclusion	171
9.1	Contributions and Managerial Implications	172
9.2	Limitations and Future Work	173
	References	177
VI	Appendices	215
	Appendices	217
A	Predict-then-Optimize or Predict-and-Optimize? An Empirical Evaluation of Cost-sensitive Learning Strategies	217
A.1	Data	218
A.2	Instance- or class-dependent cost training: additional results	218
B	Optimizing the Preventive Maintenance Frequency with Causal Machine Learning	221
B.1	Hyperparameter optimization	222
C	NOFLITE: Learning to Predict Individual Treatment Effect Distributions	223
C.1	Data sets and associated data generating processes	224
C.2	Hyperparameter optimization	224
D	Accounting for Informative Sampling When Learning to Forecast Treatment Outcomes Over Time	227
D.1	Extended Related Work	228
D.1.1	Forecasting treatment effects over time	228
D.1.2	Informative sampling in ML	229
D.1.3	Neural ODEs	229
D.1.4	Missing data	230
D.2	List of Mathematical Symbols	231
D.3	TESAR-CDE: Multitask Training Procedure	232
D.4	TESAR-CDE: Implementation	234

D.4.1	Weight truncation	234
D.4.2	Hyperparameter optimization	234
D.4.3	A note on adding counts	235
D.5	Tumor Growth Simulation	236
D.6	Additional Results	239
E	AutoCATE: Towards End-to-End, Automated Treatment Effect Estimation	241
E.1	Background on CATE Estimation	242
E.1.1	Challenges: The Fundamental Problem and Confounding	242
E.1.2	Assumptions For Identifiability	243
E.1.3	CATE Estimation: Meta- and Baselearners	243
E.2	AutoCATE: Additional Information	244
E.2.1	Metalearners	244
E.2.2	Evaluation Risk Measures	245
E.2.3	Preprocessor and Baselearner Search Spaces	246
E.2.4	Example ML Pipeline	248
E.2.5	AutoCATE’s API: Additional Information	248
E.3	Data: Additional Information	249
E.4	Additional Results	250
E.4.1	Stage 1: Evaluation	250
E.4.2	Stage 2: Estimation	251
E.4.3	Stage 3: Ensembling	251
E.4.4	Benchmarking AutoCATE	252
E.4.5	Analyzing AutoCATE’s Results	252
E.5	Comparing Software Packages for CATE Estimation	253
F	Metalearners for Ranking Treatment Effects	257
F.1	Problem Formulation: Identifiability Assumptions	258
F.2	Empirical Results: Additional Experiments	258
F.2.1	Comparing Performance for the Different Objectives and Metalearners (RQ1): Additional Results	259
F.2.2	Analyzing Alternative Metrics (RQ2) and Design Choices (RQ3): Additional Results	259
F.2.3	Sensitivity Analysis	260
	List of Tables	267
	List of Figures	275
	Publication list	285

Part I

PROLOGUE



1

INTRODUCTION

There have recently been remarkable advances in machine learning (ML) and artificial intelligence. In 2012, AlexNet revolutionized computer vision, by training a deep learning architecture on the large ImageNet data set, leveraging graphics processing units (GPUs) for efficient training [1]. The transformer architecture was proposed in 2017 [2] and enabled efficient training of large language models such as BERT [3] or generative pre-trained transformers (GPT) [4]. In 2018, AlphaFold achieved significant progress in predicting protein structures by training ML components on a large data set of 3D protein structures [5]. Despite spanning different data modalities, these advances share common elements: the combination of increasing *data* availability, enhanced *computing power*, and sophisticated *ML algorithms* designed to extract patterns from data.

The advancement of these technologies and the increasing abundance of data suggest exciting prospects for using ML in all aspects of our society. This dissertation explores the use of ML to support and improve operational decision-making. Operational decisions are those made frequently and on a large scale, e.g., as part of a business process or governmental policy. The large volumes of data available from past operational decisions suggest opportunities for ML to extract insights from historical data and improve future decisions. These opportunities are found in a wide variety of applications across different fields. In the *financial* domain, ML could help to analyze a

financial transaction’s fraud risk or score a customer’s creditworthiness. In *marketing*, ML could help predict customer churn and optimize marketing efforts in order to limit this churn. In *maintenance*, ML could predict asset failures and help optimize maintenance operations to minimize failures. Finally, data-driven methodologies could help *healthcare* practitioners analyze different treatment plans and by predicting possible treatment outcomes for a patient. This dissertation demonstrates the potential of machine learning for a variety of decision-making problems across these applications.

Although these applications are incredibly diverse, ranging from marketing to healthcare, and from machines to patients, decision-making in these applications can be analyzed in a *similar operational framework*. For each case, there is abundant historical data on the decision that was made, the information that was used to make that decision, and the observed outcome. For example, in fraud detection, historical transactions are available, including information about that transaction (e.g., the transaction time), the decision that was made (blocked or not), and its outcome (fraudulent or not). The key research question we aim to tackle is:

Given available data on past operational decisions, how can we analyze, improve and optimize future decisions?

Given the data coming from decision-making processes and the available computing power, applying ML seems both natural and straightforward. However, as this dissertation will illustrate, general-purpose ML algorithms are not inherently designed for decision-making problems. Because of this, applying ML algorithms out-of-the-box does not fully exploit their potential, which could lead to suboptimal decisions. This dissertation explores two key dimensions for optimizing the development and application of ML algorithms to maximize their effectiveness in operational decision-making:

1. *Decision-focused learning*: Understanding the operational context is essential for effectively applying ML and integrating it in the decision-making process.
2. *Causal inference*: Estimating the causal effects of possible decisions ensures that decisions have the envisioned impact.

1.1 Motivating Examples and Applications

This section describes some of the applications and problems addressed in this dissertation. These cases demonstrate how the requirements of decision-focused learning and causal inference appear in real-world scenarios.

Marketing. A crucial aspect of marketing is building and retaining long-term customer relationships. Significant progress has been made in using customer data to predict churn, but knowing if a customer will churn does not translate directly into an actionable marketing strategy. For instance, should pricing be adjusted for customers likely to churn? Should we offer a gift to build brand loyalty? As customers may respond differently to these interventions, we need to know their responses to optimize our marketing strategy. This is exactly the goal of uplift modeling: predict the causal effect of a marketing intervention to help *prescribe* the optimal action. Additionally, retention campaigns often face budget constraints or geographic restrictions. The goal of retention is not just to retain as many customers as possible, but also to focus on retaining the most profitable ones. As such, the predicted customer responses need to be integrated in a wider operational decision-making framework, formalized as a constrained *optimization* problem. This example shows how prescriptive analytics can help optimize operational marketing decisions by combining causal inference to estimate the causal effect of a marketing intervention, and a decision-focused approach to account for the true objective and possible constraints.

Maintenance. Maintenance is an important challenge for companies: although it incurs significant costs, ensuring assets are up and running is essential for smooth operations. Advances in sensor technology have led to an abundance of asset data. Initially, this data was used for *predictive* maintenance to anticipate asset failures. However, merely knowing that an asset might fail is not the most useful information for maintenance technicians, as it might be too late to prevent the failure. Instead, *prescriptive* maintenance estimates the causal effect of maintenance interventions on reducing the failure probability, enabling data-driven maintenance plans. Additionally, maintenance *optimization* involves planning maintenance activities across assets to optimize business metrics, such as order fulfillment or downtime costs, while considering operational constraints like technician availability. Again, this example illustrates that causal predictions need to be embedded in an operational decision-making problem to fully exploit the potential of machine learning.

Healthcare. The proliferation of electronic health records has paved the way for individualized healthcare. An initial application of machine learning in this domain was diagnostic and *predicted* diseases based on patient information. Additionally, machine learning can enable a prognostic approach and *prescribe* optimal treatments, by predicting patient outcomes for a potential treatment plans. In the future, these prescriptive models could be integrated into the *optimization* of healthcare operations, mini-

mizing healthcare costs and improving scheduling efficiency. For instance, this framework could help decide which patients we schedule for surgery to maximize life expectancies, given the limited availability of personnel and operating rooms.

Other applications. A wide diversity of applications can be analyzed within a similar predict-prescribe-optimize framework. In *fraud detection*, a pervasive problem, predictive models are typically used. Nevertheless, there is potential for preventive approaches that prescribe anti-fraud interventions and integrate these recommendations into anti-fraud operations, considering fraud investigator availability and transaction costs. In *credit scoring*, predictive models assess the risk of giving a loan. As this risk is affected by the interest rate of the loan, there is potential for prescriptive pricing to manage credit risk [6]. Additionally, a lender needs to manage its portfolio and optimize its global risk across loans. In another application, *human resources*, predictive models could not only forecast employee turnover, but also prescribe retention strategies and help optimize the composition of the workforce. Similar analyses can be applied to *economics* (economic forecasting, policy simulation, economy optimization), *energy management* (load forecasting, demand response, energy grid optimization), *supply chain management* (demand forecasting, inventory prescription, supply chain optimization), and *public health* (epidemiological forecasting, intervention analysis, public health policy).

In each of these examples, machine learning could simply be applied as a predictive tool. However, additional value can be unlocked by using machine learning to prescribe optimal decisions. To make the most of the available data for decision-making, the effects of possible interventions can be predicted and integrated into a broader decision-making framework. As such, these examples illustrate the importance of decision-focused learning and causal inference for optimizing decision-making.

1.2 Decision-Focused Learning

Operational decisions need to take a variety of factors into account and deal with uncertainty inherent to the decision-making context. Machine learning can help to reduce this uncertainty by predicting outcomes from data. This approach is typically implemented within a prediction-optimization framework. First, machine learning is used to predict outcomes relevant to the decision-making problem. For example, predicting the likelihood that investigating a financial transaction will uncover fraud based on data from past

investigations. Second, these predictions are incorporated in a (constrained) optimization problem, as studied in operations research, where an objective function is optimized subject to constraints. For example, which transactions should we investigate, given that we have four investigators who can each look at ten transactions?

Although ML can provide exceptional predictive power, models typically do not consider the operational context in which these predictions will be applied. This might result in suboptimal decisions in the second optimization stage for two reasons. First, the objective of the ML model is not aligned with the operational objective. ML models are typically designed to achieve maximal predictive accuracy. In contrast, decisions are evaluated based on criteria such as revenue, profit, production downtime, or quality of patient care. Because of this misalignment, the most accurate ML model may not result in the best decisions. Second, decisions need to consider operational constraints, such as budget limitations, production capacity, or legal constraints. ML models aim to make accurate predictions for all instances, but some predictions might be irrelevant given these constraints.

Decision-focused learning provides an alternative to traditional prediction-focused approaches by training an ML model to directly optimize the quality of the final decisions within an integrated prediction-optimization framework. This approach takes the relevant objective and constraints into account and aligns the ML model with the operational problem, as traditionally studied in operations research (OR). Therefore, decision-focused learning can be seen as a bridge between ML and OR, offering a framework for using ML for optimal decision-making.

1.3 Causal Inference

Decisions aim to effect change by intervening in the world. The impact or effect resulting from a decision is a causal quantity—the decision “causes” the change. This differs fundamentally from the correlational patterns typically learned in ML. For example, an ML model may correctly predict that patients that receive a new state-of-the-art medical treatment are 10% more likely to recover from a disease. However, the model does not consider that receiving this treatment is also correlated with better healthcare, more wealth, or better nutrition—all factors contributing to recovery. In this example, the positive correlation between treatment and recovery may be due to correlation of the treatment and these other factors, rather than the treatment itself. Clearly, correlational inferences alone are insufficient for sound decision-making.

Causal inference provides a framework for reasoning about causal effects and leveraging ML to estimate these effects from data. The key is to consider the causal structure behind the data, in which we can distinguish at least three types of variables. First, the treatment is the decision we are considering (e.g., what therapy to give a cancer patient). Second, the outcome is what we aim to optimize (e.g., the patient’s quality adjusted life years). Third, confounders are factors that influenced both the treatment decision and outcome (e.g., the patient’s age). The goal of treatment effect estimation is to determine the causal effect of the treatment on the outcome, while accounting for the influence of confounders on the treatment and outcome.

From a predictive viewpoint, estimating causal effects presents unique challenges. First, the fundamental problem of causal inference is that the causal effect itself is never observed [7]. The causal effect of an action is defined as the difference in outcomes between taking that action and not taking it. However, for each instance, we either take the action or do not. Therefore, we only observe one factual outcome and the other, counterfactual outcome is never observed. Because of this, we never know the treatment effect itself, which means that there is no ground truth train or validate ML models. Second, the outcome we observe is typically not random: for example, whether a patient receives a certain treatment depends on confounders such as comorbidities, age, or the physician’s beliefs, etc. These confounding factors introduce a covariate shift between the training distribution and the unknown, counterfactual test distribution, further complicating causal effect estimation.

1.4 Contributions and Outline

The contributions presented in this dissertation lie at the intersection of machine learning, operational research, and causal inference. Each chapter contributes either to decision-focused learning, causal inference, or both. An overview is provided in Figure 1.1. We introduce each contribution below.

Chapter 2. As discussed in the previous sections, one of the key challenges for decision-focused learning is aligning the predictive and decision objectives. In cost-sensitive machine learning, the goal is to incorporate costs related to different predictive outcomes. For example, in fraud detection, the cost of a false negative (missing a fraudulent transaction) is much higher than the cost of a false positive (flagging a legitimate transaction). Additionally, these costs are instance-dependent and vary across transactions. Chapter 2 analyzes cost-sensitive learning and presents a taxonomy

Chapter	Title	Decision-Focused	Causal Inference
2	Predict-then-optimize or predict-and-optimize? An empirical evaluation of cost-sensitive learning strategies	✓	✗
3	A new perspective on classification: optimally allocating limited resources to uncertain tasks	✓	✗
4	Optimizing the preventive maintenance frequency with causal machine learning	✗	✓
5	NOFLITE: Learning to predict individual treatment effect distributions	✗	✓
6	Accounting for informative sampling when learning to forecast treatment outcomes over time	✗	✓
7	AutoCATE: Towards end-to-end, automated treatment effect estimation	✗	✓
8	Metalearners for ranking treatment effects	✓	✓

Figure 1.1: **Dissertation overview.** This dissertation discusses seven chapters. For each chapter, we indicate whether it contributes to decision-focused learning, causal inference, or both.

of different possible approaches. We distinguish between methodologies that integrate costs during the predictive phase and those that consider costs during the decision-making phase. Furthermore, we differentiate between class-dependent and instance-dependent costs. We compare the different approaches empirically using nine data sets from a variety of application areas, providing a comprehensive comparison of methods within this framework. This part has been published as [8]:

- Vanderschueren, T., Verdonck, T., Baesens, B., & Verbeke, W. (2022). Predict-then-optimize or predict-and-optimize? An empirical evaluation of cost-sensitive learning strategies. *Information Sciences*, 594, 400-415.

Chapter 3. In practice, the decision-making *objective* is only part of the problem. Operational *constraints*, such as budget limitations or resource availability, are also key considerations. Chapter 3 explores decision-making problems in which the capacity to act on decisions is limited. For example, in fraud detection, investigators can only review a fraction of all transactions. Traditional ML models would focus equally on correctly predicting the fraud probability for both suspicious and non-suspicious transactions. This approach does not consider the capacity limitations and is not aligned with the optimization phase. Conversely, Chapter 3 introduces a novel methodology for identifying the transactions that should be prioritized given the

available capacity, ignoring less relevant transactions. The resulting models are inspired by learning to rank and are directly optimized for expected profit given the available capacity. This part has been published as [9]:

- Vanderschueren, T., Baesens, B., Verdonck, T., & Verbeke, W. (2024). A new perspective on classification: optimally allocating limited resources to uncertain tasks. *Decision Support Systems*, 179, 114151.

Chapter 4. Other contributions of this dissertation lie in the area of causal inference. Chapter 4 demonstrates how causal inference and machine learning can support decision-making with a case study on preventive maintenance. Traditional approaches rely on expert-driven mathematical formalizations of the problem and optimize maintenance decisions within this framework. In contrast, we propose a data-driven approach that learns the causal effect of maintenance from historical data and prescribes optimal maintenance decisions accordingly. Causal machine learning enables the estimation of possible outcomes resulting from different preventive maintenance frequencies, which allows for prescribing an asset-specific maintenance policy that minimizes the total maintenance cost. We additionally compare our causal, prescriptive approach with a purely predictive approach. This part has been published as [10]:

- Vanderschueren, T., Boute, R., Verdonck, T., Baesens, B., & Verbeke, W. (2023). Optimizing the preventive maintenance frequency with causal machine learning. *International Journal of Production Economics*, 258, 108798.

Chapter 5. Decision-making often not only depends on the expected effect of a decision, but also its probability distribution. Knowing the effect's distribution enables analyzing its expected utility or quantifying the effect's uncertainty. For example, a doctor might want to know the probability that a treatment will have a strictly positive effect. Most existing methods for predicting treatment effects only estimate the expected effect. Conversely, Chapter 5 presents NOFLITE, a methodology for predicting an individual's treatment effect distribution. NOFLITE is based on normalizing flows, enabling us to directly optimize the model's quality of fit, i.e., its likelihood. This part has been published as [11]:

- Vanderschueren, T., Berrevoets, J., & Verbeke, W. (2023). NOFLITE: Learning to predict individual treatment effect distributions. *Transactions on Machine Learning Research*.

Chapter 6. In many settings, decisions go beyond a single action and instead involve complex plans with different actions carried out over time. For

example, in healthcare, a doctor could prescribe a cancer patient a treatment plan consisting of different cycles of chemotherapy with varying dosages. This temporal perspective to decision-making is beneficial for a variety of applications, but comes with unique challenges. Most existing work aimed to tackle one of these challenges: time-dependent confounding, where treatment decisions were based on past treatments, outcomes, and covariates. In contrast, Chapter 6 looks at another challenge, informative sampling, where the act of observing itself is dependent on past treatments, outcomes, and covariates. We provide an overview of different sampling mechanisms, analyze how informative sampling can lead to bias, and present a methodology for learning treatment effects in the presence of informative sampling. This part has been published as [12]:

- Vanderschueren, T.*, Curth, A.*, Verbeke, W., & van der Schaar, M. (2023, July). Accounting for informative sampling when learning to forecast treatment outcomes over time. In *International Conference on Machine Learning* (pp. 34855-34874). PMLR.

Chapter 7. Challenges in causal inference, such as confounding or informative sampling, not only complicate the training of ML models, but also their validation. These challenges therefore limit the adoption of causal ML models, despite significant methodological advances. To encourage more widespread adoption of these methods, Chapter 7 argues for a more holistic view on the development of these ML pipelines. We describe the search for an ML pipeline for causal effect estimation as a search problem, which we call the counterfactual combined algorithm selection and hyperparameter optimization problem. We differentiate between design choices in three steps: evaluation, estimation, and ensembling. The resulting framework, **AutoCATE**, is the first automated ML solution tailored for treatment effect estimation that tackles each step with automated, data-driven protocols. **AutoCATE** facilitates the empirical comparison and analysis of different design choices, offering valuable guidelines for both practitioners and researchers. This part has not yet been published outside this dissertation.

Chapter 8. In causal inference, metalearners are general frameworks that use supervised machine learning algorithms to estimate treatment effects. A common application for estimating treatment effects is to determine which instances should be treated given limited treatment capacity or budgets. Standard metalearners do not consider these treatment limitations and, as such, may result in suboptimal treatment allocation. Therefore, Chapter 8 introduces a decision-focused methodology for causal inference, ranking metalearners, that maximizes the total effect resulting from a treatment policy given limited treatment capacity. Similar to the approach discussed in Chap-

ter 3, these metalearners achieve this by objective functions, inspired by the literature on learning to rank. We compare these ranking metalearners with their traditional counterparts empirically using data from applications where instances need to be prioritized for treatment. This part has been made available online as [13]:

- Vanderschueren, T., Verbeke, W., Moraes, F., & Proença, H. M. (2024). Metalearners for ranking treatment effects. *arXiv preprint arXiv:2405.02183*.

Chapter 9. We end this dissertation with a general conclusion in Chapter 9, where we discuss our contributions and their managerial implications, as well as limitations and potential directions for future work.

Part II

DECISION-FOCUSED LEARNING



2

PREDICT-THEN-OPTIMIZE OR PREDICT-AND-OPTIMIZE? AN EMPIRICAL EVALUATION OF COST-SENSITIVE LEARNING STRATEGIES

Predictive models are increasingly being used to optimize decision-making and minimize costs. A conventional approach is *predict-then-optimize*: first, a predictive model is built; then, this model is used to optimize decision-making. A drawback of this approach, however, is that it only incorporates costs in the second stage. Conversely, the *predict-and-optimize* approach proposes learning a predictive model by directly minimizing the cost of the downstream decision-making task. This is achieved by using a task-specific loss function incorporating the costs of different outcomes in the first stage, with the eventual aim of obtaining more cost-effective decisions in the second stage. This work compares both approaches in the context of cost-sensitive classification. Conceptually, we use the two-stage framework to categorize existing cost-sensitive learning methodologies by differ-

entiating between methodologies for cost-sensitive model training and decision-making. Empirically, we compare and evaluate both approaches using different cost-sensitive training and decision-making methodologies, as well as both class-dependent and instance-dependent cost-sensitive methods. This is achieved using real-world data from a range of application areas and a combination of cost-sensitive and cost-insensitive performance measures. The key finding is that the decision-making strategy is generally found to be more effective than training with a task-specific loss or their combination.

2.1 Introduction

Predictive models are increasingly being used to optimize decision-making. In many applications, the goal is to minimize the cost incurred through decisions. A conventional approach is to *predict-then-optimize*: in the first stage, a predictive model is built to maximize its predictive power; then, in the second stage, decisions are made based on the model's predictions and the costs associated with decisions. However, a drawback of this approach is that it only considers costs in the second decision-making stage. Conversely, several recent works proposed an alternative, integrated *predict-and-optimize* approach [14], [15]. This approach works by integrating costs within the learning objective of the predictive model in the first stage. Thus, model learning is decision-focused: the quality of the predictions on downstream decision-making is directly considered [15]. The goal of this approach is to make more cost-effective decisions. Therefore, the model's predictions need only be accurate insofar as this contributes to optimal decision-making in the second stage.

We use the predict-and-optimize approach to analyze an earlier line of work on cost-sensitive machine learning [16], [17]. Although predict-and-optimize has typically been applied to problems such as stochastic programming and combinatorial optimization [14], [15], the goal of cost-sensitive methodologies is similar to the one in predict-and-optimize in the sense that both aim to obtain better decisions by aligning the predictive model with the decision-making context. Even though a variety of cost-sensitive learning methodologies have been proposed to more effectively deal with classification tasks where different decisions have different costs associated with them, it is not clear which of these approaches work best and how they relate to each other. The lack of understanding of these methods is due to a combination of reasons. First, novel approaches are often only compared to their cost-insensitive counterparts. Second, a variety of different metrics are used

to judge these methodologies. Third, a limited number of datasets are typically used. These are often also proprietary, making it impossible to replicate findings.

Using the two-stage framework, we categorize existing techniques as either learning cost-sensitive models in the first stage or making cost-sensitive decisions in the second stage. Thus, we can empirically compare the predict-then-optimize and predict-and-optimize approaches for cost-sensitive classification. Our main contributions are as follows:

- Conceptually, we review the literature on cost-sensitive learning and differentiate between two general approaches using the two-stage framework: cost-sensitive training of models and cost-sensitive decision-making.
- Empirically, we conduct an extensive evaluation to compare predict-then-optimize and predict-and-optimize using nine real-world datasets from different application areas. Moreover, we analyze different methods of incorporating costs during training and during decision-making, as well as their combinations. We also look at the effect of incorporating costs at an instance level as opposed to a class level.
- To facilitate replication of the presented results and encourage further research on instance-dependent cost-sensitive learning, the full experimental code is made publicly available at <https://github.com/toonvds/CostSensitiveLearning>.

2.2 Related work

Before applying the two-stage framework to cost-sensitive classification, we summarize existing work based on two criteria (see Table 2.1): 1) whether the costs are class- or instance-dependent (see section 2.2.1) and 2) whether costs are integrated before, during or after the training of a classification model (see section 2.2.2). Before training, instances can be preprocessed, i.e., they can be sampled, weighted, or relabeled (e.g., MetaCost [18]). During training, costs can be incorporated in the learning algorithm, e.g., with custom decision tree splitting criteria or through a cost-sensitive objective function. After training, the decision threshold can be made cost-sensitive.

There are other cost-sensitive strategies that are not covered by these criteria and outside the scope of this work. Several methodologies look at cost-sensitive feature [47] or model selection [48]. A recent, dedicated framework and overview of cost-sensitive ensemble methods is presented in [49]. Moreover, whereas this work focuses on cost-sensitive learning in the context of supervised learning, other work has focused on cost-sensitive semi-supervised

Table 2.1: **Cost-sensitive learning overview.** We present an overview of various cost-sensitive learning methods in terms of the type of costs, place with respect to model training and classifier(s) used when applicable. *Costs*—CD: class-dependent, ID: instance-dependent; *Classifiers*—boosting, DR: decision rule, DT: decision tree, LR: logistic regression, NB: Naive Bayes, NN: neural network, SVM: support vector machine, -: classifier-agnostic.

Ref.	Costs		Place w.r.t. training			Classifier(s)
	CD	ID	Before	During	After	
[19]	✓	×	×	✓	✓	DR
[20]	✓	×	×	✓	×	DT
[21]	✓	×	×	×	✓	DR, NN
[22]	✓	×	✓	✓	✓	NN
[23]	✓	×	×	✓	×	BO
[18]	✓	×	✓	×	×	-
[24]	✓	×	×	✓	×	SVM
[25]	✓	×	×	✓	×	DT
[26]	✓	×	×	✓	×	NB
[27]	✓	×	✓	×	×	DT
[28]	✓	×	✓	×	✓	NN
[29]	✓	×	×	×	✓	-
[30]	✓	×	×	✓	×	BO
[31]	✓	×	✓	×	×	-
[32]	✓	×	✓	×	×	-
[33]	✓	×	×	✓	×	SVM
[34]	✓	×	×	✓	✓	BO
[35]	✓	×	×	✓	×	BO
[36]	✓	×	×	✓	✓	LR
[37]	✓	×	×	✓	✓	DT
[38]	×	✓	×	✓	×	BO
[39]	×	✓	×	×	✓	-
[40]	×	✓	✓	×	×	BO
[41]	×	✓	×	✓	×	SVM
[42]	×	✓	×	✓	×	DT
[43]	×	✓	×	✓	✓	LR
[44]	×	✓	×	×	✓	-
[45]	×	✓	×	✓	×	DT
[46]	×	✓	×	✓	✓	BO, LR

[50] and positive-unlabeled learning [51]. Finally, a related line of work in regression considers asymmetric objectives to more closely align a regression model’s learning objective with the decision-making task [16].

2.2.1 Types of costs

In classification, costs can be formalized with a cost matrix [17]. Similar to how the confusion matrix in Table 2.2a differentiates between outcomes depending on the actual and predicted class, a cost matrix associates a cost to these different outcomes. In Table 2.2b, a cost matrix is shown for the setting with class-dependent costs. When costs are instance-dependent, each instance will have a different cost matrix, denoted by the index i in Table 2.2c. Note that this framework also allows the inclusion of benefits or profits in the form of negative costs.

Table 2.2: **Cost matrix.** Extending the confusion matrix (2.2a) to a class- (2.2b) and instance-dependent cost matrix (2.2c).

(a) Confusion matrix	(b) Class-dependent costs	(c) Instance-dependent costs																																				
<table style="border-collapse: collapse; width: 100%;"> <tr> <td style="border-right: 1px solid black; padding: 5px;"></td> <td style="padding: 5px; text-align: center;">Actual</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;"></td> <td style="padding: 5px; text-align: center;">0 1</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px; text-align: center;">Predicted</td> <td style="padding: 5px;"> <table style="border-collapse: collapse; width: 100%;"> <tr> <td style="padding: 5px; text-align: center;">0</td> <td style="padding: 5px; text-align: center;">TN</td> <td style="padding: 5px; text-align: center;">FN</td> </tr> <tr> <td style="padding: 5px; text-align: center;">1</td> <td style="padding: 5px; text-align: center;">FP</td> <td style="padding: 5px; text-align: center;">TP</td> </tr> </table> </td> </tr> </table>		Actual		0 1	Predicted	<table style="border-collapse: collapse; width: 100%;"> <tr> <td style="padding: 5px; text-align: center;">0</td> <td style="padding: 5px; text-align: center;">TN</td> <td style="padding: 5px; text-align: center;">FN</td> </tr> <tr> <td style="padding: 5px; text-align: center;">1</td> <td style="padding: 5px; text-align: center;">FP</td> <td style="padding: 5px; text-align: center;">TP</td> </tr> </table>	0	TN	FN	1	FP	TP	<table style="border-collapse: collapse; width: 100%;"> <tr> <td style="border-right: 1px solid black; padding: 5px;"></td> <td style="padding: 5px; text-align: center;">Actual</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;"></td> <td style="padding: 5px; text-align: center;">0 1</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px; text-align: center;">Predicted</td> <td style="padding: 5px;"> <table style="border-collapse: collapse; width: 100%;"> <tr> <td style="padding: 5px; text-align: center;">0</td> <td style="padding: 5px; text-align: center;">c^{TN}</td> <td style="padding: 5px; text-align: center;">c^{FN}</td> </tr> <tr> <td style="padding: 5px; text-align: center;">1</td> <td style="padding: 5px; text-align: center;">c^{FP}</td> <td style="padding: 5px; text-align: center;">c^{TP}</td> </tr> </table> </td> </tr> </table>		Actual		0 1	Predicted	<table style="border-collapse: collapse; width: 100%;"> <tr> <td style="padding: 5px; text-align: center;">0</td> <td style="padding: 5px; text-align: center;">c^{TN}</td> <td style="padding: 5px; text-align: center;">c^{FN}</td> </tr> <tr> <td style="padding: 5px; text-align: center;">1</td> <td style="padding: 5px; text-align: center;">c^{FP}</td> <td style="padding: 5px; text-align: center;">c^{TP}</td> </tr> </table>	0	c^{TN}	c^{FN}	1	c^{FP}	c^{TP}	<table style="border-collapse: collapse; width: 100%;"> <tr> <td style="border-right: 1px solid black; padding: 5px;"></td> <td style="padding: 5px; text-align: center;">Actual</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;"></td> <td style="padding: 5px; text-align: center;">0 1</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px; text-align: center;">Predicted</td> <td style="padding: 5px;"> <table style="border-collapse: collapse; width: 100%;"> <tr> <td style="padding: 5px; text-align: center;">0</td> <td style="padding: 5px; text-align: center;">c_i^{TN}</td> <td style="padding: 5px; text-align: center;">c_i^{FN}</td> </tr> <tr> <td style="padding: 5px; text-align: center;">1</td> <td style="padding: 5px; text-align: center;">c_i^{FP}</td> <td style="padding: 5px; text-align: center;">c_i^{TP}</td> </tr> </table> </td> </tr> </table>		Actual		0 1	Predicted	<table style="border-collapse: collapse; width: 100%;"> <tr> <td style="padding: 5px; text-align: center;">0</td> <td style="padding: 5px; text-align: center;">c_i^{TN}</td> <td style="padding: 5px; text-align: center;">c_i^{FN}</td> </tr> <tr> <td style="padding: 5px; text-align: center;">1</td> <td style="padding: 5px; text-align: center;">c_i^{FP}</td> <td style="padding: 5px; text-align: center;">c_i^{TP}</td> </tr> </table>	0	c_i^{TN}	c_i^{FN}	1	c_i^{FP}	c_i^{TP}
	Actual																																					
	0 1																																					
Predicted	<table style="border-collapse: collapse; width: 100%;"> <tr> <td style="padding: 5px; text-align: center;">0</td> <td style="padding: 5px; text-align: center;">TN</td> <td style="padding: 5px; text-align: center;">FN</td> </tr> <tr> <td style="padding: 5px; text-align: center;">1</td> <td style="padding: 5px; text-align: center;">FP</td> <td style="padding: 5px; text-align: center;">TP</td> </tr> </table>	0	TN	FN	1	FP	TP																															
0	TN	FN																																				
1	FP	TP																																				
	Actual																																					
	0 1																																					
Predicted	<table style="border-collapse: collapse; width: 100%;"> <tr> <td style="padding: 5px; text-align: center;">0</td> <td style="padding: 5px; text-align: center;">c^{TN}</td> <td style="padding: 5px; text-align: center;">c^{FN}</td> </tr> <tr> <td style="padding: 5px; text-align: center;">1</td> <td style="padding: 5px; text-align: center;">c^{FP}</td> <td style="padding: 5px; text-align: center;">c^{TP}</td> </tr> </table>	0	c^{TN}	c^{FN}	1	c^{FP}	c^{TP}																															
0	c^{TN}	c^{FN}																																				
1	c^{FP}	c^{TP}																																				
	Actual																																					
	0 1																																					
Predicted	<table style="border-collapse: collapse; width: 100%;"> <tr> <td style="padding: 5px; text-align: center;">0</td> <td style="padding: 5px; text-align: center;">c_i^{TN}</td> <td style="padding: 5px; text-align: center;">c_i^{FN}</td> </tr> <tr> <td style="padding: 5px; text-align: center;">1</td> <td style="padding: 5px; text-align: center;">c_i^{FP}</td> <td style="padding: 5px; text-align: center;">c_i^{TP}</td> </tr> </table>	0	c_i^{TN}	c_i^{FN}	1	c_i^{FP}	c_i^{TP}																															
0	c_i^{TN}	c_i^{FN}																																				
1	c_i^{FP}	c_i^{TP}																																				

Class-dependent costs

Various cost-sensitive machine learning techniques have been proposed for dealing with class-dependent costs. In this setting, one class is more important in terms of costs, and because of that, a cost-sensitive model should focus more on correctly classifying this class compared to a cost-insensitive model. In the simple case of a linear decision boundary, class-dependent costs result in a parallel shift away from the more costly class (see Figure 2.1).

Even though no general benchmarking studies exist, two works analyze class-dependent cost-sensitive boosting specifically and find cost-sensitive decision-making to be the most effective strategy [34], [52]. Finally, note that the literature on class-dependent cost-sensitive learning is intertwined with the literature on learning with class imbalance, and by using the appropriate costs, similar techniques can be used. For a recent survey on class imbalance, we refer the reader to [53].

Instance-dependent costs

Conceptually, many of the techniques for dealing with class-dependent costs can and have been transferred to the instance-dependent setting. However, instance-dependent costs create an additional degree of complexity, as they depend not only on the class but also on characteristics of the instance (e.g., the transaction’s amount in fraud detection). For a simple linear classifier, class-dependent costs result in a parallel shift of the cost-insensitive optimal decision boundary, whereas instance-dependent costs can additionally result in a rotation of this boundary (see Figure 2.1). This illustrates that when costs are instance-dependent, the learner needs to consider both the class distribution (explicitly) and cost distribution (implicitly). In theory, including instance-dependent costs in decision-making can lead to lower overall costs [41]. However, despite the conceptual differences, the benefits and drawbacks of using instance- rather than class-dependent costs on the performance of the learning algorithms have not yet been examined empirically.

2.2.2 Cost-sensitive classification in the predict-and-optimize framework

Machine learning models are increasingly being used to support and optimize decision-making. The conventional two-stage *predict-then-optimize* approach builds a predictive model with the aim of maximizing its accuracy in the first stage and then uses this model to optimize decision-making in the second stage. Conversely, *predict-and-optimize* is a recent paradigm that directly optimizes a predictive model by using a task-specific loss function in the first stage to optimize decision-making in the second stage [15]. The benefit of an integrated approach is that it directly learns a model to minimize the cost of the eventual decisions. The model in the predict-then-optimize approach might produce more accurate predictions overall, but the model in the predict-and-optimize is decision-focused instead of prediction-focused: it learns to accurately predict only insofar as it impacts the decision-making in the second stage, and as such, the resulting decisions are of higher quality [14].

We can apply this two-stage framework to cost-sensitive classification: in the first stage, a predictive model (i.e., a classifier) is built; in the second stage, this model is used to assign class labels to instances in order to minimize the resulting cost. Thus, we can classify existing cost-sensitive learning methodologies as either learning a predictive model in the first stage or optimizing decisions in the second stage. This distinction is based on whether costs are integrated before, during or after the training of a model (see Table 2.1). The first category, *cost-sensitive training of models*, consists of

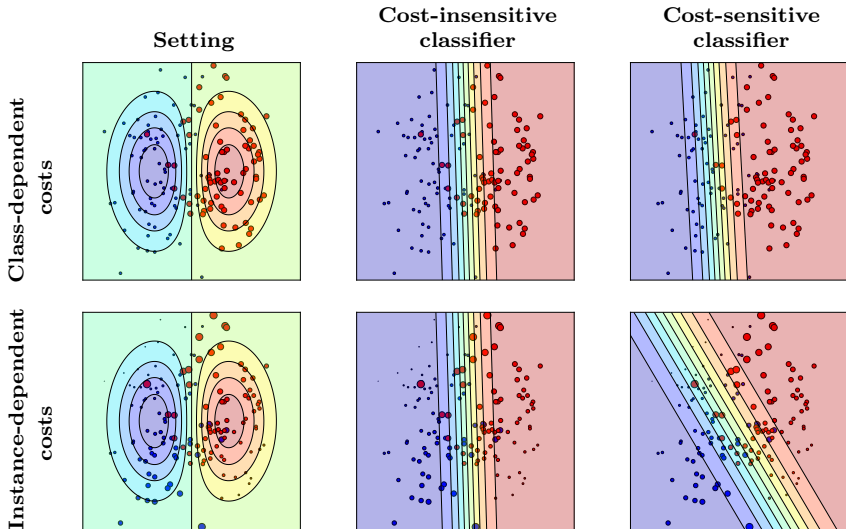


Figure 2.1: **Toy example with class-dependent (top) and instance-dependent costs (bottom).** (Left) Two classes and the probability distribution are shown, with the instance size proportional to its misclassification cost. (Middle) The resulting decision boundary for a *cost-insensitive* classifier mimics the underlying probability distribution. (Right) For a *cost-sensitive* classifier, the decision boundary lies further from the more costly class when costs are class-dependent. With instance-dependent costs, the decision boundary is not only related to the probability distribution, but also the cost distribution.

techniques that are applied before or during training to build a classifier, whereas the second, *cost-sensitive decision-making*, consists of thresholding techniques that are applied after training to make decisions. Note that several approaches are possible in each stage – several of these are described in the following.

Cost-sensitive training of classification models

In the first stage, a predictive model is learned. A traditional approach learns a model by maximizing its likelihood – independent of how predictions are used in the downstream task. Alternatively, learning can be done with a task-specific loss function to align the model with the objective of the downstream task and obtain a cost-sensitive model. Thus, the quality of the predictions on the resulting solution of the downstream task is directly considered [14].

Training with a traditional classification objective also leads to tradeoffs in the resulting model’s accuracy for different regions of the input-output space. However, in contrast to the decision-focused approach, this tradeoff might not be optimal for the downstream task [14]. An illustration of the different tradeoffs for a cost-insensitive and a cost-sensitive linear model can be seen in Figure 2.1.

In general, machine learning algorithms can be understood in terms of risk minimization [54]. In this framework, the goal of a learning algorithm is to find the classifier that minimizes the risk. Formally, for a distribution $p(\mathbf{x}, y)$ and a classifier $f_\theta : \mathbf{X} \rightarrow [0, 1] : \mathbf{x} \mapsto f_\theta(\mathbf{x})$ defined by parameters $\theta \in \Theta$, the risk to be minimized is:

$$R(\theta) = \int \int \mathcal{L}(y, \mathbf{x}, \theta) p(\mathbf{x}, y) d\mathbf{x} dy,$$

where $\mathcal{L}(y, \mathbf{x}, \theta)$ represents the loss or objective function for a classifier $f_\theta(\mathbf{x})$ and data (\mathbf{x}, y) [32]. In reality, the true joint probability distribution $p(\mathbf{x}, y)$ is unknown. Consequently, the learner relies on the empirical density to minimize the risk given the available training data. This is the principle of empirical risk minimization (ERM) [54]. For a dataset $(\mathbf{x}_i, y_i) \in \mathcal{D}$ with $i \in \{1, \dots, N\}$, the empirical risk is defined as:

$$R_{emp}(\theta) = \mathcal{E}_{x, y \sim \mathcal{D}} \left[\mathcal{L}(y_i, \mathbf{x}_i, \theta) \right] = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(y_i, \mathbf{x}_i, \theta).$$

Clearly, it is essential to choose an appropriate loss function \mathcal{L} . A first and straightforward candidate is the zero-one loss comparing the actual y and predicted label \hat{y} : $\mathcal{L}^{0/1}(y, \hat{y}) = I(y \neq \hat{y})$, although it is common to use a convex surrogate. A popular choice is the **cross-entropy** loss, which is equivalent to the maximum likelihood (ML) method [54]. In binary classification, we have $\mathcal{L}^{CE}(y_i, \mathbf{x}_i, \theta)$:

$$y_i \log f_\theta(\mathbf{x}_i) + (1 - y_i) \log(1 - f_\theta(\mathbf{x}_i)). \quad (2.1)$$

However, as argued above, a disadvantage of the maximum likelihood approach is that it does not take into account the costs of different decisions. Consequently, using this loss function, the empirical risk fails to reflect the true risk of the downstream task. To solve this issue, the ERM framework can be extended to include costs: given a dataset $(\mathbf{x}_i, y_i, \mathbf{c}_i) \in \mathcal{D}$ for $i \in \{1, \dots, N\}$ with an instance’s cost matrix \mathbf{c}_i , a cost-sensitive loss function $\mathcal{L}(y, \mathbf{x}, \mathbf{c}, \theta)$ can be defined [32]. In this way, the empirical risk can be made cost-sensitive, and a task-specific loss can be used.

A first approach for a task-specific loss is to weight the training examples by their misclassification cost [17], [40]. This can be formulated in terms of a **weighted cross-entropy** loss function $\mathcal{L}^{wCE}(y_i, \mathbf{x}_i, \mathbf{c}_i, \theta)$ [32]:

$$c_i^{FN} y_i \log f_\theta(\mathbf{x}_i) + c_i^{FP} (1 - y_i) \log(1 - f_\theta(\mathbf{x}_i)). \quad (2.2)$$

Note that this approach is equivalent to oversampling proportional to misclassification costs [32].

A second task-specific approach builds on the idea that the optimal cost-sensitive prediction minimizes the expected cost [17]. Using this, an alternative loss function can be defined that equals the expected cost [43], [46]. The corresponding empirical risk is the **average expected cost** $\mathcal{L}^{AEC}(y_i, \mathbf{x}_i, \mathbf{c}_i, \theta)$:

$$y_i \left(f_\theta(\mathbf{x}_i) c_i^{TP} + (1 - f_\theta(\mathbf{x}_i)) c_i^{FN} \right) + (1 - y_i) \left(f_\theta(\mathbf{x}_i) c_i^{FP} + (1 - f_\theta(\mathbf{x}_i)) c_i^{TN} \right). \quad (2.3)$$

Cost-sensitive decision-making

The predictive model learned in the first stage is used to make decisions in the second stage. In the case of cost-sensitive classification, the predicted posterior probabilities are used to classify instances with the aim of minimizing the resulting cost. This is achieved by applying an appropriate decision threshold. There are several policies that can be used to optimize decision-making in the second stage.

The first and most natural candidate is the instance-dependent cost-sensitive threshold, which predicts the class with the minimal expected risk. Because this risk depends not only on the posterior probabilities but also on the associated costs, an instance's optimal classification should consider both its posterior probability and its cost related to the different outcomes [17]. Formally, for an instance i , a prediction \hat{y}_i has a certain risk $R(\hat{y}_i | \mathbf{x}_i, y)$ associated with it depending on its posterior probability and cost matrix:

$$R(\hat{y}_i | \mathbf{x}_i, y_i) = \begin{cases} p(y_i = 0 | \mathbf{x}_i) c_i^{TN} + p(y_i = 1 | \mathbf{x}_i) c_i^{FN} & \text{if } \hat{y}_i = 0 \\ p(y_i = 0 | \mathbf{x}_i) c_i^{FP} + p(y_i = 1 | \mathbf{x}_i) c_i^{TP} & \text{if } \hat{y}_i = 1 \end{cases}$$

The optimal decision \hat{y}_i^* minimizes this risk, i.e., $\hat{y}_i^* = 1$ if $R(\hat{y}_i = 1 | \mathbf{x}_i) < R(\hat{y}_i = 0 | \mathbf{x}_i)$. Using this, the optimal decision threshold t_i^* for an instance can be found: $\hat{y}_i^* = 1$ if $p(y_i = 1 | \mathbf{x}_i) > t_i^*$, with

$$t_i^* = \frac{c_i^{FP} - c_i^{TN}}{c_i^{FP} - c_i^{TN} + c_i^{FN} - c_i^{TP}}. \quad (2.4)$$

For a given classifier θ , the score $f_\theta(\mathbf{x}_i)$ can be used as an estimate of the posterior probability $p(y_i = 1|\mathbf{x}_i)$. However, it is important to note that this requires the model to produce calibrated probabilities or that some calibration method is first applied to the model’s output.

In addition to the instance-dependent cost-sensitive threshold, several alternative decision-making strategies are possible. For example, by using the average cost matrix, a single class-dependent cost-sensitive threshold can be used for all instances. Furthermore, instead of the theoretically motivated optimal thresholds, several alternatives are possible. Empirical thresholding searches for the threshold that gives the lowest cost on a validation set [29]. Moreover, a common heuristic is to use the class imbalance threshold, which uses the prior probability of the minority class as a threshold $t^{CI} = P(Y = 1)$. The idea is that this will compensate for the lack of focus on this class, which is often more important in terms of costs.

2.3 Methodology

The goal of this work is to empirically analyze different instance-dependent cost-sensitive learning approaches on the resulting classification performance in terms of both costs and errors. Therefore, following the presented analysis of the literature, we formulate three key research questions to study the effect of cost-sensitive training using task-specific loss (RQ1), cost-sensitive decision-making (RQ2) and their combination (RQ3). Moreover, we look at the effect of considering costs at an instance level (RQ4). For each question, several hypotheses are proposed.

RQ1. Does instance-dependent cost-sensitive training result in improved performance compared to training without costs?

- $\mathcal{H}1.1$: In terms of *costs*, cost-sensitive training results in better performance compared to training without costs.
- $\mathcal{H}1.2$: In terms of *errors*, cost-insensitive training results in better performance compared to training with costs.

RQ2. Does instance-dependent cost-sensitive thresholding result in improved performance compared to class-dependent thresholding?

- $\mathcal{H}2.1$: In terms of *costs*, instance-dependent cost-sensitive thresholding results in improved performance compared to class-dependent thresholding.

- $\mathcal{H}2.2$: In terms of *costs*, calibrating probabilities results in more effective thresholding.
- $\mathcal{H}2.3$: In terms of *errors*, instance-dependent cost-sensitive thresholding results in improved performance compared to class-dependent thresholding.
- $\mathcal{H}2.4$: In terms of *errors*, calibrating probabilities results in more effective thresholding.

RQ3. Does combining cost-sensitive training and cost-sensitive thresholding result in improved performance compared to either method separately or completely cost-insensitive classification?

- $\mathcal{H}3.1$: In terms of *costs*, combining cost-sensitive training and cost-sensitive thresholding results in improved performance compared to either method separately or completely cost-insensitive classification.
- $\mathcal{H}3.2$: In terms of *errors*, combining cost-sensitive training and cost-sensitive thresholding results in improved cost performance compared to either method separately or completely cost-insensitive classification.

RQ4. Is it beneficial to train with instance-dependent costs instead of class-dependent costs?

- $\mathcal{H}4.1$: In terms of *costs*, using instance-dependent costs results in better performance compared to class-dependent costs.
- $\mathcal{H}4.2$: In terms of *errors*, using instance-dependent costs results in better performance compared to class-dependent costs.

2.3.1 Experimental design

In this section, we describe the experimental design that is used to answer the proposed research questions empirically. We analyze the effect of different factors in the decision-focused learning framework (see Figure 2.2 for an overview): cost-sensitive training of models in the first stage, cost-sensitive decision-making in the second stage and the combination of both. Finally, to look at the effect of training with instance-dependent costs, we also compare these models with models trained with class-dependent costs in terms of both the scores and decisions.

Cost-sensitive training

To compare different approaches to learn a predictive model in the first stage, we will compare a traditional, cost-insensitive approach (cross-entropy

\mathcal{L}^{CE}) with two cost-sensitive task-specific objective functions: an indirect, weighted approach (weighted cross-entropy \mathcal{L}^{wCE}) and a direct approach (average expected cost \mathcal{L}^{AEC}) (see equations 2.1, 2.2 and 2.3). These are implemented using three different types of classifiers: logistic regression, neural network and gradient boosting. For the neural network, we use a multilayer perceptron with one hidden layer and hyperbolic tangent as activation function. This results in a total of 9 models (see Table 2.3). For neural networks and gradient boosting, hyperparameter selection is based on the best value of the objective function on a validation set.

These classifiers are frequently adopted in both science and industry, and can be considered as representative of prominent and diverse types of machine learning techniques: they span both linear and nonlinear models, both tree-based and neural-based models, as well as both ensembles and single classifiers. This selection is further motivated by strong performance reported across various benchmarking studies [e.g., 55]. Finally, as all three methodologies optimize an objective function, they allow for a direct and fair comparison of general cost-sensitive learning strategies.

Cost-sensitive decision-making

To consider the effect of cost-sensitive decision-making in the second stage, we compare a range of nine different thresholding strategies: the theoretically optimal instance-dependent cost-sensitive threshold (IDCS) (see equation 2.4) or the equivalent with calibrated probabilities (IDCS*), as well as their class-dependent variants (CDCS and CDCS*). Calibration is performed with the nonparametric isotonic regression, which has been shown to achieve good results when enough data are available [56]. Furthermore, we include different types of empirical thresholding techniques by finding

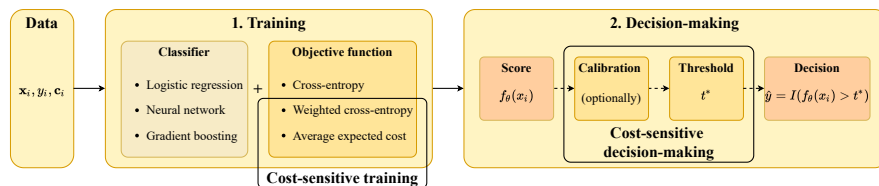


Figure 2.2: **Overview of the experimental design using the two-staged framework.** In the first stage, a predictive model is built by training a type of classifier with an objective function, which can be both cost-sensitive or cost-insensitive. In the second stage, the predictions of this model are used to make decisions. Both the scores and decisions are evaluated.

Table 2.3: **Overview of the different models.** These are obtained by combining the different objective functions with the different types of classifiers.

	Logistic regression	Neural network	Gradient boosting
\mathcal{L}^{CE}	logit	net	boost
\mathcal{L}^{wCE}	wlogit	wnet	wboost
\mathcal{L}^{AEC}	cslogit	csnet	csboost

the best threshold in terms of instance-dependent costs, class-dependent costs and F1 score on a validation set. Finally, we also include using the class-imbalance (CI) ratio $P(Y = 1)$ and the default threshold for binary classification ($t = 0.5$).

To summarize, instance-dependent cost-sensitive learning is analyzed by comparing different objective functions for different classifiers in the first stage and decision-making strategies in the second stage. This allows us to compare the predict-then-optimize and predict-and-optimize approaches, as well as to analyze different cost-sensitive techniques in each stage. Moreover, we also look at the effect of using instance-dependent costs as opposed to using class-dependent costs.

2.3.2 Experimental procedure and evaluation metrics

For the empirical evaluation, a 2×5 -fold stratified cross-validation procedure is used (see Algorithm 1). This is repeated for each dataset. Using this framework, we conduct two experiments for each model: one with instance-dependent costs and one with class-dependent costs. The full experimental procedure is available in the code.

We use a variety of metrics to evaluate the models. These can be categorized based on two criteria: whether these incorporate costs (cost sensitivity) and whether they look at probabilities or decisions (threshold dependency). To assess the importance of costs during training independently from the thresholding strategy, we rely on threshold-independent metrics. To compare the different thresholding strategies, we use threshold-dependent metrics.

Several cost-insensitive metrics are used to assess the models' ability to accurately classify instances. First, two threshold-independent metrics are the area under the ROC curve (AUROC) and average precision (AP), which summarize the ROC and precision-recall curves, respectively. The latter may be more informative given the high degree of class imbalance that is

Algorithm 1: Experimental procedure per dataset

Result: Evaluation metrics
Load data;
Initialize cost matrix;
Split data into 5 stratified folds;
for *each fold* $i \in 1 : 5$ **do**
 for *each repetition* $j \in 1 : 2$ **do**
 Test data = fold i ;
 Training data = 75% of remaining data;
 Validation data = 25% remaining data;

 # Preprocess data:
 Convert categorical features (using WoE encoding);
 Standardize data: $z = \frac{x-\mu}{\sigma}$;
 if *training with class-dependent costs* **then**
 Average cost matrix for training set;
 Average cost matrix for validation set;
 end

 # Train and evaluate models:
 Train models;
 Set decision thresholds;
 Evaluate model outputs and predictions for different
 thresholds;
 end
end
Summarize evaluation metrics over all folds;

typically encountered in cost-sensitive applications [57]. Moreover, the Brier score is used to assess whether the model’s outputs are calibrated probabilities. Finally, to evaluate the impact of the decision-making threshold, we use the F1-score.

Moreover, performance is also judged in terms of costs. Again, several threshold-independent metrics are applicable. First, the average expected cost (AEC, see equation 2.3) is used. Second, Spearman’s rank correlation coefficient ρ is used to look at the correlation between probabilities and costs for positive instances. This metric analyzes whether cost-sensitive models prioritize correctly classifying costlier instances. Finally, one cost-sensitive, threshold-dependent metric is also used: cost savings. These compare the total costs incurred by a model to classify by predicting all instances as the cheapest default class (either 0 or 1) [43]:

$$\text{Savings} = \frac{\text{Cost}(f_{\theta}(\mathbf{x})) - \min\{\text{Cost}(f_0(\mathbf{x})), \text{Cost}(f_1(\mathbf{x}))\}}{\text{Cost}(f_{\theta}(\mathbf{x}))} \quad (2.5)$$

The domain of this ratio is $[-\infty, 1]$, where 1 is the perfect model, but when the model does better than predicting the default class, we obtain savings in $]0, 1]$.

To test the statistical significance of the results, we use two types of tests depending on whether we are performing multiple or pairwise comparisons [58]. In the case of multiple comparisons, Friedman tests with Nemenyi post hoc correction are used. These are visualized using critical difference diagrams that show the average rankings (where a lower rank is better). Models that are not connected in this diagram have significantly different mean ranks. For pairwise comparison, Wilcoxon signed-rank tests are used. A significance level of 5% is used primarily, except where both 5% and 10% are used when indicated.

2.4 Empirical results

In this section, the empirical results are presented. First, the data and corresponding cost matrices are described. Second, the results are presented, and these findings are used to answer the proposed research questions.

2.4.1 Data

The data are from a diverse set of classification tasks where costs are instance-dependent: fraud detection, direct marketing, customer churn and credit scoring (see Table 2.4). All datasets are publicly available (see appendix

A.1). In each dataset, there is some degree of class imbalance with the positive class being the minority, though some cases are more extreme than others. The cost matrices depend on the application area and are adopted from earlier work (for an overview, see Table 2.5). The idea behind these is provided below.

Table 2.4: **Overview of the datasets.** Size (N), dimensionality (D) and degree of class imbalance (% Pos) are shown.

Application	Dataset	Abbr.	N	D	% Pos
Fraud detection	Kaggle Credit Card Fraud	<i>KCCF</i>	282,982	29	0.16
	Kaggle IEEE Fraud Detection	<i>KIFD</i>	590,540	431	3.50
Direct marketing	KDD Cup 1998	<i>KDD</i>	191,779	22	5.07
	UCI Bank Marketing	<i>UBM</i>	45,211	15	11.70
Churn prediction	Kaggle Telco Customer Churn	<i>KTCC</i>	7,032	19	26.58
	TV Subscription Churn	<i>TSC</i>	9,379	46	4.79
Credit scoring	Kaggle Give Me Some Credit	<i>GMSC</i>	112,915	10	6.74
	UCI Default of Credit Card Clients	<i>DCCC</i>	30,000	23	22.12
	VUB Credit Scoring	<i>VCS</i>	18,917	16	16.95

Fraud detection In fraud detection, a positive prediction triggers an investigation that has a fixed cost c_f , while a missed fraudulent transaction incurs a cost equal to its amount A_i (see Table 2.5a). For both datasets, c_f is set to 10 following [46].

Direct marketing A similar reasoning applies here: any direct marketing action results in a fixed cost c_f , and missing a potential success incurs an instance-dependent cost (see Table 2.5b). Whereas *KDD* uses the amount A_i and $c_f = 0.68$ following both [39] and [49], *UBM* instead uses the expected interest given A_i and $c_f = 1$, following [45].

Customer churn For customer churn prediction, c_i^{FP} and c_i^{FN} are set at 2 and 12 times the monthly amount A_i for *KTCC*, respectively, following [49] (see Table 2.5c). For *TSC*, the cost matrix provided with the dataset is used (not shown here, see [59]).

Credit scoring Finally, for credit scoring, the costs of a *FP* and *FN* are calculated following [43] with both a function of the loan amount A_i .

2.4.2 Results

In this section, we report the results of the experiments and discuss the implications for the research questions of this study. First, we compare training

Table 2.5: **Cost matrices for the different applications.** For each application, we present the different costs associated with different outcomes. A_i , Int_i , c_i^{FN} and c_i^{FP} represent instance-dependent costs, and c_f is a fixed cost.

(a) Fraud detection			(b) Direct marketing			(c) Customer churn			(d) Credit scoring		
		y			y			y			y
		0	1			0	1			0	1
\hat{y}	0	0	A_i	\hat{y}	0	0	A_i/Int_i	\hat{y}	0	0	$12A_i$
\hat{y}	1	c_f	c_f	\hat{y}	1	c_f	c_f	\hat{y}	1	$2A_i$	0
		0	c_i^{FN}			0	c_i^{FP}			0	c_i^{FN}
		1	c_f			1	c_f			1	0

with the three different objective functions with threshold-independent metrics. Second, we use threshold-dependent metrics to analyze the different thresholding strategies for the different models. Third, we compare the results of this analysis by training with class-dependent costs. Complete results on the different experiments can be found in the digital appendix.

Cost-sensitive training

We start by looking at two traditional evaluation metrics: the area under the ROC curve (AUROC) and the average precision (AP) (see Figure 2.3). The cost-insensitive methodologies (net, boost, logit) have the best scores for both of these metrics, although only the difference with the worst classifier, cslogit, is significant at a 5% level.

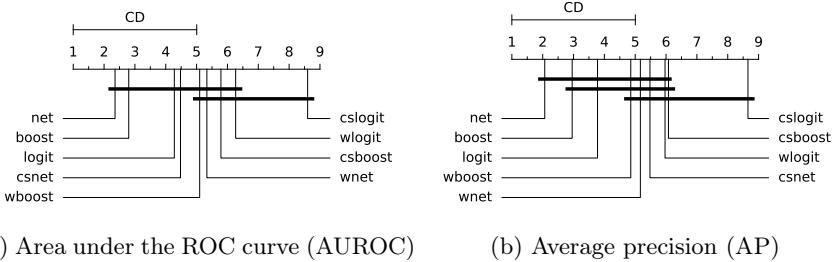


Figure 2.3: **Cost-insensitive metrics:** critical difference diagrams for the AUROC and AP

The AUROC and AP do not consider costs. Therefore, the next metric is the average expected cost (AEC) (see Figure 2.4a). Unsurprisingly, the best performing classifiers are those directly optimizing this expected cost. In almost all cases, the differences with the cost-insensitive models are statistically significant at the 5% level. Models trained with a cost-weighted

objective function perform worse but still better than the cost-insensitive classifiers.

Similarly, Spearman’s rank correlation coefficient ρ is used to compare the correlation between the predicted probabilities and costs for the positive instances (see Figure 2.4b). The cost-sensitive classifiers perform better on average. For this metric, there does not seem to be a substantial difference between training with weighted cross-entropy and the average expected cost.

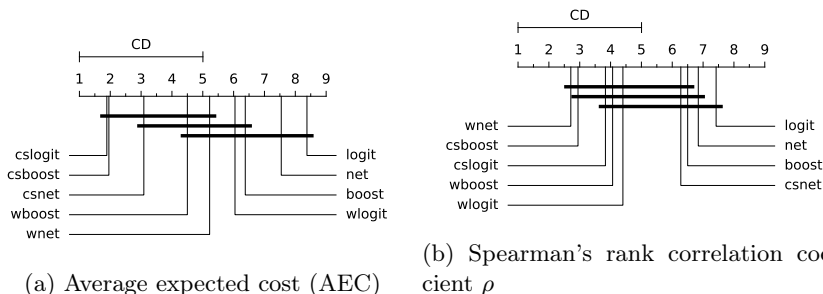


Figure 2.4: **Cost-sensitive metrics:** critical difference diagrams for the AEC and Spearman ρ .

The tradeoff between minimizing costs or errors seems to more strongly affect the least flexible classifier, logistic regression. Cslogit has the worst performance for the cost-insensitive metrics but the best performance in terms of AEC. In contrast, logit performs well for AUROC and AP but is the worst in terms of the cost-sensitive metrics. This indicates that there is a larger tradeoff between minimizing costs and errors for a more inflexible, linear model compared to the neural networks and gradient boosting.

In conclusion, cost-sensitive models perform worse on average for traditional, cost-insensitive evaluation metrics but better in terms of cost-sensitive metrics. This indicates that minimizing errors or minimizing costs are two fundamentally different objectives. Moreover, the type of objective function seems to be more important than the type of classifiers, as neither logistic regression, neural networks nor gradient boosting consistently outperform another category.

Cost-sensitive decision-making

To analyze the different approaches to cost-sensitive decision-making, we first compare the savings (see Table 2.6) and then the F1 scores (see Table 2.8) for each model and thresholding strategy averaged across all datasets. This

also allows us to analyze the effect of using a cost-sensitive objective function in the first stage on the quality of the decisions in the second stage.

Table 2.6: **Savings: comparison of the different thresholding strategies (averaged across all datasets)**. Best and second-best result for each model are denoted in **bold** and *italic*.

	IDCS	IDCS*	CDCS	CDCS*	Empirical			CI	0.5
					ID	CD	F1		
logit	0.36	<i>0.35</i>	0.29	0.30	0.30	0.30	0.24	0.09	0.06
wlogit	0.14	<i>0.36</i>	0.06	0.37	0.37	0.37	0.34	-1.33	0.37
cslogit	0.38	<i>0.37</i>	0.38	0.38	0.38	0.38	<i>0.37</i>	0.38	0.38
net	0.41	<i>0.40</i>	0.35	0.35	0.35	0.35	0.29	0.20	0.12
wnet	0.13	0.36	0.13	<i>0.39</i>	<i>0.39</i>	<i>0.39</i>	0.35	-0.66	0.40
csnet	<i>0.36</i>	0.39	0.34	0.35	0.35	0.34	0.29	0.34	0.34
boost	0.41	<i>0.40</i>	0.35	0.36	0.36	0.36	0.29	0.32	0.13
wboost	0.30	0.37	0.25	<i>0.36</i>	<i>0.36</i>	<i>0.36</i>	0.30	0.23	0.32
csboost	0.39	0.36	0.39	0.39	0.39	0.39	0.34	0.39	<i>0.38</i>

In terms of savings (see Table 2.6), the importance of the decision-making strategy is strongly related to the objective function that is used to train a classifier. For the cost-insensitive models (trained with cross-entropy), it is absolutely crucial to not use the default threshold 0.5 and instead use the instance-dependent cost-sensitive threshold. When a cost-weighted objective function is used, good results can be obtained either when $t = 0.5$, when probabilities are calibrated and a cost-sensitive threshold is used, or when the threshold is tuned empirically. Conversely, the models trained with AEC achieve relatively stable savings across thresholding strategies. In other words, using a task-specific loss function is related to the performance of different decision-making strategies, with the direct approach giving the most consistent results across strategies.

Moreover, the type of decision-making strategy that is used in the first stage, i.e., the threshold, is more important than the type of objective function used to train the predictive model in the first stage. In fact, given that an appropriate threshold is used, it is only beneficial in terms of savings to use a cost-sensitive objective function for the simplest model: logistic regression. For neural networks and gradient boosting, the cost-insensitive models also achieve good results given that the optimal threshold is used. The best savings overall are obtained when a cost-insensitive model is combined with instance-dependent cost-sensitive thresholding.

Calibrating probabilities achieve better results only for either the weighted cross-entropy or for class-dependent cost-sensitive thresholds. In fact, the two best savings are obtained without calibration. For models trained with a

normal cross-entropy loss, calibration does not result in a higher Brier score, suggesting that these probabilities were already calibrated (see Table 2.7). Although the largest improvement of calibration is observed for the models trained with AEC, this only leads to an improvement in terms of savings for csnet. Only the models trained with weighted cross-entropy have a much better performance after calibration.

In terms of savings, it is clearly beneficial to consider costs during decision-making: empirical thresholding with the F1-score, the class imbalance heuristic or $t = 0.5$ can obtain bad results (depending on the objective function). In general, thresholding on an instance level also seems to be favorable to class-dependent thresholding. Finally, both theoretical and empirical thresholding can achieve good results.

Table 2.7: **Brier score before and after calibration for the different models (averaged across all datasets)**. The Brier score of models trained with a cost-sensitive objective function improves considerably, whereas it is stable for the models trained with a cross-entropy loss.

<i>Calibration</i>	logit	wlogit	cslogit	net	wnet	csnet	boost	wboost	csboost
<i>Before</i>	0.07	0.16	0.24	0.07	0.16	0.23	0.07	0.13	0.19
<i>After</i>	0.07	0.07	0.08	0.07	0.07	0.07	0.07	0.08	0.08
<i>Difference</i>	0.00	-0.09	-0.17	0.00	-0.09	-0.16	0.00	-0.05	-0.11

The best thresholding strategies in terms of F1 scores do not necessarily achieve the lowest costs (see Table 2.8). This emphasizes that there is also a clear difference between minimizing errors and costs in the decision-making stage. The best results in terms of the F1 score are obtained when the threshold is tuned empirically to maximize this metric. Again, calibrating probabilities is only beneficial for the models trained with weighted cross-entropy. For these models, however, empirical thresholding is more effective than theoretical thresholding. Finally, note how using $t = 0.5$ achieves relatively good results in terms of the F1 score, even though it does not result in large cost savings.

Is it beneficial to train with instance-dependent costs instead of class-dependent costs?

First, we look at the effect of using instance-dependent costs during training as opposed to training with class-dependent costs in terms of cost-insensitive metrics (see Tables 2.9 and B1). Although the results are fairly similar for the two settings, training with class-dependent costs achieves better results for these metrics for almost all cases. Based on this observation, it can be

Table 2.8: **F1 Score: comparison of the different thresholding strategies (averaged across all datasets)**. Best and second-best result for each model are denoted in **bold** and *italic*.

	IDCS	IDCS*	CDCS	CDCS*	Empirical ID	Empirical CD	F1	CI	0.5
logit	0.33	0.33	<i>0.39</i>	<i>0.39</i>	<i>0.39</i>	<i>0.39</i>	0.42	0.30	0.31
wlogit	0.23	0.30	0.26	<i>0.39</i>	0.38	0.38	0.41	0.21	<i>0.39</i>
cslogit	0.36	0.31	0.39	0.39	<i>0.38</i>	<i>0.38</i>	0.39	0.39	0.39
net	0.36	0.36	<i>0.42</i>	<i>0.42</i>	<i>0.42</i>	<i>0.42</i>	0.47	0.34	0.37
wnet	0.23	0.29	0.26	<i>0.39</i>	0.38	<i>0.39</i>	0.43	0.22	<i>0.39</i>
csnet	0.39	0.35	0.41	0.42	0.41	0.42	0.45	0.41	<i>0.43</i>
boost	0.39	0.36	<i>0.45</i>	0.44	0.43	0.44	0.48	0.40	0.40
wboost	0.31	0.32	0.35	0.41	0.41	0.41	0.45	0.33	<i>0.43</i>
csboost	0.35	0.28	0.36	0.36	0.36	0.36	0.40	0.36	<i>0.38</i>

concluded that training with instance-dependent costs may be disadvantageous in terms of errors.

Next, we consider cost-sensitive metrics (see Tables 2.10 and B2). Here, training with instance-dependent costs achieves comparatively better results. Using instance-dependent costs consistently leads to lower average expected costs (though the difference is not always significant). Additionally, in terms of Spearman’s ρ , it is better for all models, and this difference is significant except for csnet. In terms of savings, instance-dependent costs are better on average, although not consistently.

2.5 Discussion

In this section, we draw upon the results of the empirical evaluation to answer the four key research questions that were previously proposed. An overview of findings per research question can be found in Table 2.11.

Does cost-sensitive training result in improved performance compared to training without costs? Cost-sensitive training achieves better performance in terms of cost-sensitive, but performs worse in terms of cost-insensitive metrics. Cost-sensitive objectives result in a lower expected cost and learn to prioritize costly instances based on the Spearman correlation between model outputs and costs for positive instances. This is observed for both cost-sensitive objective functions: the indirect, weighted approach (weighted cross-entropy) and the direct approach (average expected cost). These findings illustrate that there is a tradeoff between minimizing costs or minimizing errors during training, indicating that these are two fundamen-

Table 2.9: **Instance-dependent or class-dependent costs: cost-insensitive metrics per model.** Significantly better results are denoted in **bold** (5%) and *italic* (10%).

Metric	Costs	wlogit	cslogit	wnet	csnet	wboost	csboost
AUROC	ID	0.76	0.72	0.76	0.77	0.77	0.76
	CD	0.77	<i>0.73</i>	0.78	0.77	0.78	0.79
AP	ID	0.38	0.27	0.40	<i>0.38</i>	0.42	0.38
	CD	0.42	0.27	0.45	0.36	0.45	0.44
F1 IDCS	ID	0.23	0.36	0.23	0.39	0.31	0.35
	CD	0.24	0.37	0.25	0.39	0.32	0.38
F1 IDCS*	ID	0.30	0.31	0.29	0.35	0.32	0.28
	CD	0.34	0.32	0.35	0.35	0.35	0.35
F1 CDCS	ID	0.26	0.39	0.26	0.41	0.35	0.36
	CD	0.27	0.39	0.27	0.42	0.34	<i>0.41</i>
F1 CDCS*	ID	0.39	0.39	0.39	0.42	0.41	0.36
	CD	0.41	0.39	0.42	0.42	0.43	0.43
F1 Emp ID	ID	0.38	0.38	0.38	0.41	0.41	0.36
	CD	0.41	0.39	0.42	0.42	0.43	0.43
F1 Emp CD	ID	0.38	0.38	0.39	0.42	0.41	0.36
	CD	0.41	0.39	0.42	<i>0.42</i>	0.43	0.43
F1 Emp F1	ID	0.41	0.39	0.43	0.45	0.45	0.40
	CD	0.44	0.39	0.46	0.45	0.46	0.47
F1 CI	ID	0.21	0.39	<i>0.22</i>	0.41	0.33	0.36
	CD	0.21	0.39	0.21	<i>0.42</i>	0.33	<i>0.40</i>
F1 0.5	ID	0.39	0.39	0.39	0.43	0.43	0.38
	CD	0.41	0.39	0.42	0.43	0.45	0.44

tally different objectives.

Does instance-dependent cost-sensitive thresholding result in improved performance compared to class-dependent thresholding?

In terms of costs, cost-sensitive thresholding at an instance level was observed to be the most successful decision-making strategy, outperforming all other decision-making thresholds. Calibrating probabilities was only beneficial when the weighted cross-entropy or a class-dependent threshold was used. The differences in best-performing thresholds when optimizing for savings or F1 score illustrate that minimizing errors and costs are also two different objectives in the decision-making stage.

Does combining cost-sensitive training and cost-sensitive thresholding result in improved performance compared to either method separately or completely cost-insensitive classification?

Combining

Table 2.10: **Instance-dependent or class-dependent costs: cost-sensitive metrics per model.** Significantly better results are denoted in **bold** (5%) and *italic* (10%). AEC is normalized between 0 and 1 per dataset (lower is better).

Metric	Costs	wlogit	cslogit	wnet	csnet	wboost	csboost
AEC	ID	<i>0.56</i>	0.07	<i>0.47</i>	0.18	<i>0.41</i>	0.06
	CD	0.68	0.25	0.59	0.18	0.48	0.21
Spearman’s ρ	ID	0.09	0.11	0.16	-0.06	0.13	0.23
	CD	-0.10	-0.05	-0.10	-0.07	-0.07	-0.10
Savings IDCS	ID	0.14	0.38	0.13	0.36	0.30	0.39
	CD	0.14	0.32	0.17	0.36	0.30	0.38
Savings IDCS*	ID	0.36	<i>0.37</i>	0.36	0.39	0.37	0.36
	CD	<i>0.38</i>	0.36	0.40	0.39	0.38	<i>0.39</i>
Savings CDCS	ID	0.06	0.38	0.13	0.34	0.25	0.39
	CD	0.03	0.31	0.08	0.34	0.22	0.35
Savings CDCS*	ID	0.37	0.38	0.39	0.35	0.36	0.39
	CD	0.32	0.31	0.34	0.35	0.34	0.35
Savings Emp ID	ID	0.37	0.38	0.39	0.35	0.36	0.39
	CD	0.32	0.31	0.34	0.34	0.34	0.35
Savings Emp CD	ID	0.37	0.38	0.39	0.34	0.36	0.39
	CD	0.32	0.31	0.34	0.34	0.34	0.35
Savings Emp F1	ID	0.34	0.37	0.35	0.29	0.30	0.34
	CD	0.27	0.31	0.27	0.29	0.26	0.27
Savings CI	ID	-1.33	0.38	-0.66	0.34	0.23	0.39
	CD	-1.59	0.31	-0.81	0.34	0.19	0.34
Savings 0.5	ID	0.37	0.38	0.40	0.34	0.32	<i>0.38</i>
	CD	0.33	0.31	0.35	0.34	0.29	0.34

cost-sensitive training and decision-making did not necessarily achieve better results. In fact, the best savings were obtained by training with a cost-insensitive objective function and using the instance-dependent cost-sensitive threshold. This illustrates that the type of thresholding is more important than the type of objective function in terms of costs.

Is it beneficial to train with instance-dependent costs instead of class-dependent costs? In terms of both training and thresholding, using instance-dependent instead of class-dependent costs was observed to achieve better results for cost-sensitive metrics, but worse results for traditional cost-insensitive metrics. Specifically, not using costs at all is preferential for minimizing errors, using instance-dependent costs is optimal for minimizing costs, and using class-dependent costs lies somewhere between these two.

Table 2.11: **Summary of the key findings.** We present a summary of the results per research question and hypothesis. Performance is judged in terms of costs and errors. Each question is answered with yes (✓), no (✗) or inconclusive (?).

Research question	Costs	Errors
1. Does instance-dependent cost-sensitive training result in improved performance compared to training without costs?		
Instance-dependent cost-sensitive training results in better performance compared to training without costs.	✓	✗
2. Does instance-dependent cost-sensitive thresholding result in improved performance compared to class-dependent thresholding?		
Instance-dependent cost-sensitive thresholding results in improved performance compared to class-dependent thresholding.	✓	✗
Calibrating probabilities results in more effective thresholding.	?	?
3. Does combining cost-sensitive training and cost-sensitive thresholding result in improved performance compared to either method separately or completely cost-insensitive classification?		
Combining cost-sensitive training and cost-sensitive thresholding results in improved performance compared to either method separately or completely cost-insensitive classification.	✗	✗
4. Is it beneficial to train with instance-dependent costs instead of class-dependent costs?		
Using instance-dependent costs results in better performance compared to class-dependent costs.	✓	✗

2.6 Conclusion

In this paper, we presented a focused review and empirical analysis of instance-dependent cost-sensitive classification. Conceptually, we reviewed cost-sensitive classification through the lens of predict-and-optimize and differentiated between different methods for both cost-sensitive training and decision-making. Several key methodologies were implemented for different classifiers, and the resulting models were compared empirically on nine datasets from different application areas. Based on the experimental results obtained from this large-scale benchmarking experiment, we answered four research questions (see Table 2.11 for an overview).

These findings stress the importance of considering the right objective for an application. Optimizing for accuracy can be detrimental to a classifier’s performance when the actual objective is to minimize costs, which is the case in a large variety of business applications. For this, it is especially important to consider the right type of thresholding strategy. Overall, a conceptually simple yet well-performing strategy is to first train a cost-insensitive model and only introduce costs in a second stage through instance-dependent thresholding. In other words, using a task-specific loss in the first stage does not result in better decisions in the second stage, given that the optimal decision-making policy is used.

These results correspond with empirical research in the class-dependent setting: two works compared cost-sensitive boosting algorithms with cost-sensitive thresholding and found the latter to be the more effective strategy [34], [52]. Nevertheless, theoretical results in the class-dependent setting suggest that cost-sensitive training can be optimal under certain conditions. For example, under model misspecification, a cost-sensitive objective function [32] can be preferential to theoretical thresholding. Consequently, a direction for future research is to extend the theoretical analysis from the class-dependent setting toward instance-dependent costs. Additionally, it will be interesting to investigate the influence of the characteristics of the cost distribution and cost matrix on the performance of instance-dependent cost-sensitive training and decision-making methods. By sharing our code, we hope to encourage and facilitate further research on instance-dependent cost-sensitive learning.

3

A NEW PERSPECTIVE ON CLASSIFICATION: OPTIMALLY ALLOCATING LIMITED RESOURCES TO UNCERTAIN TASKS

A central problem in business concerns the optimal allocation of limited resources to a set of available tasks, where the payoff of these tasks is inherently uncertain. Typically, such problems are solved using a classification framework, where task outcomes are predicted given a set of characteristics. Then, resources are allocated to the tasks predicted to be the most likely to succeed. We argue, however, that using classification to address task uncertainty is inherently suboptimal as it does not take into account the available capacity. We present a novel solution that directly optimizes the assignment's expected profit given limited, stochastic capacity. This is achieved by optimizing a specific instance of the net discounted cumulative gain, a commonly used class of metrics in learning to rank. We demonstrate that our new method achieves higher expected profit and expected precision compared to a classification approach for a wide variety of application areas.

3.1 Introduction

Optimally allocating limited resources is a central problem in economics [60] and operations research [61]–[63]. It is often complicated further by uncertainty inherent to the considered problem. On the one hand, future resource capacity may be limited and not known exactly in advance. On the other hand, the tasks that require resources might have uncertain payoff. This situation is commonly encountered in various real-world applications. As a running example, consider the case of credit card fraud detection. Fraud analysts can only investigate a limited number of transactions each day. A priori, it is not known whether investigating a transaction will uncover a fraudulent case. The general challenge is how to optimally allocate limited resources to maximize business pay-off, e.g., how to optimally allocate fraud investigators to suspicious transactions to minimize losses due to fraud. By learning from historical data, machine learning models can assist decision-makers by predicting the most relevant tasks (e.g., transactions) based on their characteristics.

Prior work addresses the problem of uncertain task outcomes via classification [e.g., 55], [64]–[69]. The most promising tasks can be identified by estimating the probability of success for each task. The problem of allocating stochastic, limited capacity could then be addressed separately in a second stage, when assignment decisions are made by prioritizing tasks based on the estimated probabilities to result in a successful outcome. In our running example, this strategy would correspond to first predicting which instances are most likely to be fraudulent, and then investigating the most suspicious transactions. This strategy is commonly used as a decision support tool for fraud detection, but also other domains where similar problems arise, such as direct marketing, churn prediction, or credit scoring. In this article, however, we argue and demonstrate that this approach based on classification models is suboptimal when resources are limited, because a classification model does not take capacity limitations into account. Hence, although only the most promising tasks can be executed, the model focuses equally on accurately predicting probabilities for tasks that are highly unlikely to be successful and, consequently, to be executed.

To tackle this challenge, we propose a novel approach based on learning to rank that simultaneously accounts for both resource and task uncertainty. When resources are limited, we demonstrate that this approach is superior to allocation based on classification. First, we show theoretically how learning to rank can directly optimize the assignment’s expected profit given limited, stochastic capacity. By considering the available capacity during optimization, the model focuses on correctly ranking the most promising tasks,

proportional to their likelihood of being processed under limited capacity. Second, while instances are processed individually in classification, learning to rank explicitly considers a task’s relevance in comparison to the other available tasks. The benefit of this approach is that we only care about relative positions in the ranking, corresponding to the need to prioritize tasks relative to each other.

Our contributions are threefold. First, we formalize the problem of allocating limited, stochastic resources to uncertain tasks by framing it as an assignment problem. Second, we propose a novel, integrated predict-and-optimize approach to solve this problem based on learning to rank. We contrast our approach with a two-stage predict-then-optimize framework that first uses a classification model to predict task outcomes and then solves the assignment problem using the predicted task probabilities. Third, we compare both methods empirically using various real life data sets from different application areas.

3.2 Related work

The proposed solution in this paper relates to prior work on uncertainty in assignment problems, predict-and-optimize, classification, and learning to rank. In this section, we briefly introduce each line of work, describe its relationship to our contribution, and clarify the remaining research gap that our work aims to address.

3.2.1 Uncertainty in assignment problems

Optimal allocation of resources and decision-making under uncertainty are key problems in operations research [61], [62]. In this work, we consider an assignment problem. This is a general problem formulation in which the goal is to find an optimal matching of workers and tasks subject to certain constraints. This type of problem has been analyzed extensively [70] and applied to a diverse range of tasks [e.g., 71], [72]. Moreover, various extensions consider different sources of uncertainty: uncertain worker capacity, uncertain task presence (i.e., outcomes), or uncertain task-worker profits [73]–[75]. This work focuses on a specific type of linear assignment problem, in which we simultaneously address two sources of uncertainty: uncertain capacity and uncertain task success. However, instead of assuming that task success follows a probability distribution, we use a predictive model to estimate it. Although our aim is similar to Johari, Kamble, and Kanoria [76], they consider an online setting, where workers arrive and depart over time with uncertainty, with a focus on trading-off exploration and exploitation.

In contrast, we assume that the worker capacity follows a known, static probability distribution. Moreover, they consider fixed job types with certain outcomes, while we learn these outcomes using a predictive model. In general, our work is different from most work in this category as we aim to simultaneously tackle the prediction of task success as well as the optimization of the assignment problem, while most work in this category is limited to the optimization.

3.2.2 Predict-and-optimize

The intersection of operations research and machine learning has increasingly drawn the attention of researchers from both fields [77], [78]. In particular, recent work on predict-and-optimize is relevant [15], [79], [80]. The central aim in predict-and-optimize is to align a predictive model more closely with the downstream decision-making context [81]. This is achieved by fusing the prediction and optimization phases and training the model in an end-to-end manner, with the aim of obtaining higher quality decisions [82]. Ranking specifically has been studied in this context. Demirović, Stuckey, Bailey, *et al.* [83] use ranking to solve a ranking problem with uncertainty in the objective function—similar to task uncertainty in our work. However, in contrast to this work, they do not account for uncertainty in the constraint. Moreover, their method is limited to pairwise ranking, whereas we optimize a listwise objective, allowing us to consider the stochastic capacity in the optimization of the model. Demirović, J Stuckey, Bailey, *et al.* [84] are limited to linear predictive models. In contrast, our method is compatible with a variety of linear and non-linear machine learning algorithms. Their analysis considers more general optimization problems with uncertainty in the objective function. Conversely, our proposed solution is tailor-made to this specific problem setting, allowing us to use the problem structure in our solution. In general, most work in predict-and-optimize does not account for uncertainty in the constraints or optimization problem [85].

3.2.3 Classification

Classification is a task in machine learning where the goal is to predict the class of an instance given its characteristics. For instance, classifying a task as either successful or not is a binary classification problem. Existing work typically considers the applications in this paper as classification problems, e.g., fraud detection [68], [69], credit scoring [55], [65], direct marketing [64] and customer churn prediction [66], [67]. Moreover, to align the models more closely with the decision-making context, cost-sensitive classification has been used [37], [43], [86], [87]. Cost-sensitive methodologies incorporate the costs of different decisions into the optimization or use of

predictive models [8], [17], [88]. Cost-sensitive variants have been proposed for different classification models, such as logistic regression and gradient boosting [43], [87]. Nevertheless, these consider a different setting: classify instances. Conversely, our work aims to prioritize instances, to process given limited worker capacity. The output of a classification model is often used to rank instances, reflected by widely used evaluation metrics that analyze this ranking, such as the receiver operating characteristics curve and precision–recall curve [57]. However, in contrast to our work, these approaches do not consider the available capacity during optimization of the models. Although limited capacity has been acknowledged in the literature (e.g., in fraud detection [89], direct marketing [90] or churn prediction [91]), no existing solution explicitly addresses this issue. Shifman, Cohen, Huang, *et al.* [92] consider a cost-sensitive classification problem with resource constraints. However, in contrast to our work, they consider misclassification costs to be unknown and do not consider uncertainty in the capacity constraint.

3.2.4 Learning to rank

In learning to rank, the goal is to predict the order of instances relative to each other, based on their characteristics. Although learning to rank originated in the field of information retrieval, it is a general framework that has been applied to a variety of problems that have traditionally been solved with classification models, such as software defect prediction [93], credit scoring [94] and uplift modeling [95]. Moreover, similar to cost-sensitive classification, the learning to rank framework has been extended to incorporate costs of instances to align the optimization of the model more closely with the resulting decisions [96]. However, our approach is the first to explicitly consider the available capacity during the optimization of the ranking model.

3.3 Problem formulation

This work addresses the problem of optimally assigning limited and stochastic resources to tasks with uncertain outcomes to maximize the expected profit. In our running example of fraud detection, the goal would be to uncover fraudulent transactions by having fraud investigators look at them, with the aim of minimizing that day’s losses due to fraud. On the one hand, there is task uncertainty. Before investigating a transaction, the outcome of the investigation is uncertain—though this could be estimated based on historical data. On the other hand, there is an uncertain resource constraint. The availability of fraud investigators is uncertain, as well as their productivity on that day. Using historical data, we assume that a worker capacity

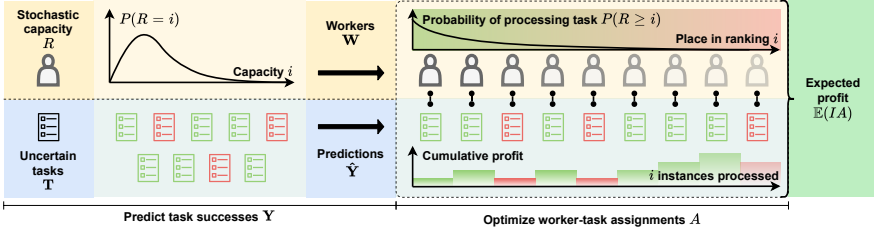


Figure 3.1: **Problem overview.** Our setting concerns a type of linear assignment problem with two sources of uncertainty: stochastic worker capacity and uncertain task outcomes. To account for stochastic capacity in the assignment problem, the capacity distribution is converted to workers with decreasing processing probabilities. Task outcomes are also uncertain and need to be predicted. The key objective is to assign workers to tasks to maximize the resulting expected profit.

distribution can be estimated. In the following, we formalize this problem as a general optimization problem.

In this section, we formalize this setting as a linear assignment problem, in which the goal is to optimally assign workers to tasks, where both workers and tasks are sources of uncertainty. The exact number of workers is uncertain at the time when resources need to be allocated, but we assume it is governed by a known probability distribution. In practice, this distribution can be estimated from historical data on the available resources or based on domain knowledge. Alternatively, a deterministic capacity can be considered. Second, task outcomes are also uncertain and need to be predicted using historical data on similar tasks. A graphical overview of the problem is shown in Figure 3.1. In the following, we introduce and formally define each element of the assignment problem.

Stochastic capacity

The available resources or number of workers W is a discrete random variable described by a known probability distribution: $W \sim Dist$. In this work, we consider a common situation where the expected capacity is smaller than the number of available tasks: $\mathbb{E}(W) \ll N$. In expectation, the stochastic capacity can be converted to a sequence of N workers with monotonically decreasing expected success rates. Each rate w_j equals the worker's probability of being available given $W \sim Dist$ and is described by the complementary cumulative probability distribution function: $w_j = P(W \geq j) = 1 - F_W(j)$. This yields a monotonically decreasing sequence of N worker success rates $\mathbf{W} = (w_1 \dots w_N) = \{1 - F_W(j)\}_{j=1}^N$ with $w_1 \geq \dots \geq w_N$. Given

Table 3.1: **Notation table.** We give an overview of the notation used in this work. For each symbol, we give both the general name and its role in our running example of fraud detection.

Symbol	Definition (with an example for fraud detection)
W	Stochastic worker capacity (number of tasks processed by fraud specialists)
\mathbf{W}	Vector of worker probabilities w_j with $w_j = P(W \geq j)$
T	Number of tasks (transactions considered)
R	Task rewards r_i (transaction payoff, i.e., fraud amount intercepted - processing cost)
Y	Task outcome y_i (fraudulent or legitimate)
A	Assignment matrix a_{ij} (which transactions fraud specialists should investigate)
\mathbf{v}	Payoff when executing a task
\mathbf{c}	Cost matrix
x	Task characteristics (time and place where transaction was made)
f_θ	Predictive model
π	Permutation of instances, i.e., a ranking

$\mathbb{E}(W) \ll N$, we expect that most tasks will not be executed and most w_j will be (close to) zero. This formulation will allow us to optimize the expected objective in section 3.4.

Uncertain tasks

There is also uncertainty regarding task outcomes. To address this uncertainty, we predict it using historical data on similar tasks. Let $\mathcal{T} = (\mathcal{X}, \mathcal{Y}, \mathcal{V})$ be the domain of all possible tasks $t_i = (\mathbf{x}_i, y_i, \mathbf{v}_i)$, where $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d$ is a set of characteristics and $y_i \in \mathcal{Y} = \{0, 1\}$ is a binary label equal to 1 if the task is successful and 0 otherwise. Moreover, $\mathbf{v}_i = \{v_i^+, v_i^-\} \in \mathcal{V} \subset \mathbb{R}^2$ denotes the payoff if the task is executed, with v_i^+ if task i was successful ($y_i = 1$) and v_i^- otherwise. A task's reward is defined as $r_i = y_i v_i^+ + (1 - y_i) v_i^-$. We have N available tasks to be allocated $\mathbf{T} = \{(\mathbf{x}_i, y_i, \mathbf{v}_i) : i = 1, \dots, N\}$, although y_i is unknown when resources need to be allocated. Given historical data, a (deterministic) predictive model can estimate task outcomes y_i resulting in N predictions.

Matching workers and tasks

Workers and tasks can then be combined in an expected profit matrix $P = (p_{ij})$, where $p_{ij} = r_i w_j$ is the profit of assigning task i to worker j for $i, j = 1, \dots, N$. Given P , the goal is to find the optimal assignment matrix $A = (a_{ij})$, where $a_{ij} = 1$ if worker i is assigned to task j and 0 otherwise, for $i, j = 1, \dots, N$. This results in the following linear assignment problem:

$$\text{maximize } \sum_{i=1}^N \sum_{j=1}^W a_{ij} r_i \quad (3.1)$$

$$\text{subject to } \sum_{i=1}^N a_{ij} \leq 1 \quad i = 1, \dots, N; \quad (3.2)$$

$$\sum_{j=1}^W a_{ij} = 1 \quad j = 1, \dots, W; \quad (3.3)$$

$$a_{ij} \in \{0, 1\} \quad i = 1, \dots, N; \quad j = 1, \dots, W; \quad (3.4)$$

$$W \sim \text{Dist} \quad (3.5)$$

where conditions 3.2 and 3.3 specify that each task is assigned to exactly one worker and vice versa; condition 3.4 imposes absolute assignments by restricting a_{ij} to 0 or 1. Condition 3.5 specifies that the resource capacity or number of workers is described by a known probability distribution Dist .

3.4 Methodology

We present two approaches for the problem presented in Section 3.3. On the one hand, a two-stage predict-then-optimize framework can be used. In the first stage, we predict the task successes $\hat{\mathbf{Y}}$. Here, we show how different types of classification objectives can be used to predict task success. In the second stage, we optimize the assignment of tasks to workers to obtain an assignment matrix A . For this, we provide an analytical solution and prove its optimality. On the other hand, we present an integrated predict-and-optimize framework for prediction and optimization by leveraging learning to rank techniques.

3.4.1 Two-stage predict-then-optimize

This section presents a conventional two-stage approach for solving the problem. In the first stage, a classification model predicts each task's probability of success. Existing approaches in classification can be used to optimize this model for either accuracy or profit [87]. In the second stage,

tasks are assigned to workers based on these predicted probabilities. We present a straightforward procedure for this assignment and prove its optimality.

Predicting task outcomes using classification

To handle the uncertainty regarding task outcomes, we train a classification model to predict whether a task will be successful. Given historical data $\mathcal{D}_{\text{Train}}$, the goal is to predict y_i using a classifier $f_\theta : \mathcal{X} \rightarrow [0, 1] : \mathbf{x} \mapsto f_\theta(\mathbf{x})$ defined by parameters $\theta \in \Theta$ that predicts the probability of a task being successful. Classifier training can be accomplished with different objective functions. We present two alternatives: one that focuses optimization on accuracy and one that optimizes the classification cost.

The conventional approach is to train the classifier with the aim of maximizing accuracy. This can be achieved using the maximum likelihood approach or, equivalently, by minimizing the cross-entropy loss function:

$$\mathcal{L}^{\text{CE}} = y_i \log f_\theta(\mathbf{x}_i) + (1 - y_i) \log(1 - f_\theta(\mathbf{x}_i)). \quad (3.6)$$

A drawback of this approach is that the solution ignores some of the problem specifications. Some tasks are more important to classify correctly than others, depending on their cost (or profit) when executed. Therefore, in cost-sensitive learning, these costs are incorporated into the training of a model. In classification, the cost of a decision depends on whether it was classified correctly and on the task itself. These costs are formalized with the concept of a cost matrix \mathbf{c}_i [17]:

$$\begin{array}{rcc} & & \text{Actual class } y_i \\ & & \begin{array}{cc} 0 & 1 \end{array} \\ \text{Predicted class } \hat{y}_i & \begin{array}{cc} 0 & 1 \end{array} & \begin{pmatrix} c_i^{\text{TN}} & c_i^{\text{FN}} \\ c_i^{\text{FP}} & c_i^{\text{TP}} \end{pmatrix} \end{array} \quad (3.7)$$

This way, we can directly minimize the average expected cost of predictions, as an alternative to the cross-entropy loss [43], [87]:

$$\begin{aligned} \mathcal{L}^{\text{AEC}} = & y_i \left(f_\theta(\mathbf{x}_i) c_i^{\text{TP}} + (1 - f_\theta(\mathbf{x}_i)) c_i^{\text{FN}} \right) \\ & + (1 - y_i) \left(f_\theta(\mathbf{x}_i) c_i^{\text{FP}} + (1 - f_\theta(\mathbf{x}_i)) c_i^{\text{TN}} \right). \end{aligned} \quad (3.8)$$

\mathcal{L}^{AEC} is a semidirect predict-and-optimize method: it incorporates some information of the downstream decision-making task, but learning is still separated from optimization [83], [84].

Optimizing worker–task assignments

Given task predictions $\hat{\mathbf{Y}}$, we can optimize the task–worker assignments. Although various general algorithms have been proposed to solve assignment problems, our formulation can be solved analytically. Here, we present this solution and prove its optimality. The key insight is that, in expectation, the worker capacity can be seen as a sequence of workers with decreasing success rates, with each success rate the probability of that working existing given $W \sim \text{Dist}$. In other words, this probability is given by the complementary cumulative probability distribution function: $w_j = P(W \geq j) = 1 - F_W(j)$. Based on this, we can sort the tasks in terms of expected reward and the workers in terms of expected probability. Matching these two sortings then optimizes the assignment problem, where the most promising tasks are assigned to the most likely workers.

Theorem 1. $\mathbf{W} = \{w_i\}_{i=1}^N$ is a sequence of monotonically decreasing worker success rates such that $w_1 \geq \dots \geq w_N$ with $w_i \in [0, 1]$ for $i = 1, \dots, N$. $\hat{\mathbf{R}} = (\hat{r}_1 \dots \hat{r}_N)$ are the predicted task rewards arranged in decreasing order such that $\hat{r}_1 \geq \dots \geq \hat{r}_N$. For the resulting expected profit matrix $P = (p_{ij})$ with $p_{ij} = w_i \hat{r}_j$, the optimal assignment is $A^* = I_N$.

Proof. Proof of Theorem 1.

$A^* = I_N$ is a feasible solution: it is straightforward to verify that the identity matrix satisfies constraints 3.2, 3.3 and 3.4 of the assignment problem. Moreover, the solution is the result of a greedy strategy: at each step m , we assign worker w with probability w_m to the highest remaining task m with payoff \hat{r}_m . To prove the optimality of this strategy, we show that it does not deviate from the optimal solution at each step up until the final solution is obtained.

First, the best single worker–task assignment is selected: the highest profit p_{ij} is $p_{11} = w_1 \hat{r}_1$; no other higher profit exists as no higher w_i or \hat{r}_j exist. Next, we continue this strategy of selecting the best remaining worker–task assignment until there are no tasks left. We can show that, at each step, no other assignment matrix leads to a larger profit than this one. At step m , the profit obtained given assignment matrix A^* equals $p_{11} + p_{22} + \dots + p_{mm} = w_1 \hat{r}_1 + w_2 \hat{r}_2 + \dots + w_m \hat{r}_m$.

Deviating from A^* at a certain step means that at least one worker must be assigned to another task. We prove that no alternative assignment leads to a higher profit. Consider switching the assignments of tasks i and j with $i < j$. In the case that task j has already been assigned to a worker, we have:

$$p_{ii} + p_{jj} \geq p_{ij} + p_{ji}$$

$$\begin{aligned}
&\Leftrightarrow w_i \hat{r}_i + w_j \hat{r}_j \geq w_i \hat{r}_j + w_j \hat{r}_i \\
&\Leftrightarrow w_i(\hat{r}_i - \hat{r}_j) \geq w_j(\hat{r}_i - \hat{r}_j) \\
&\Leftrightarrow w_i \geq w_j \text{ and } \hat{r}_i - \hat{r}_j \geq 0.
\end{aligned}$$

In the case that task j has not yet been assigned, we have:

$$\begin{aligned}
&p_{ii} \geq p_{ij} \\
&\Leftrightarrow w_i \hat{r}_i \geq w_i \hat{r}_j \\
&\Leftrightarrow w_i \geq 0 \text{ and } \hat{r}_i \geq \hat{r}_j
\end{aligned}$$

In both cases, the final statements follow from \mathbf{W} and $\hat{\mathbf{R}}$ being monotonically decreasing and $i < j$, or from $w_i \in [0, 1]$. \square

3.4.2 Integrated predict-and-optimize using learning to rank

In this section, we present a novel integrated approach for solving the assignment problem in Section 3.3. Previously, we showed how the optimal assignment is $A^* = I_N$ if \mathbf{W} and $\hat{\mathbf{R}}$ are arranged in decreasing order. Given that \mathbf{W} is defined as a decreasing sequence, the challenge of optimizing the assignment can also be seen as correctly predicting the order of expected task rewards $\hat{\mathbf{R}}$. This formulation is equivalent to an alternative interpretation of the assignment problem as finding the optimal assignments by permuting the rows and columns of the profit matrix P such that the resulting sum of the elements on the diagonal is maximized, or formally [74]:

$$\max_{\pi \in \Pi_n} \sum_{i=1}^N p_{i, \pi(i)} \tag{3.9}$$

for $\pi \in \Pi_N$ with Π_N the set of all permutations of the indices $\{1, \dots, N\}$, i.e., $\pi : \{1, \dots, N\} \mapsto \{1, \dots, N\}$. In our case, we need to find the optimal permutation of available tasks $\pi(\mathbf{T})$.

In this formulation, the assignment problem can be seen as predicting the optimal permutation $\pi(\mathbf{T})$ based on characteristics of the available tasks. Formally, let $g_\theta : \mathcal{X} \rightarrow \mathbb{R} : \mathbf{x} \mapsto g_\theta(\mathbf{x})$ be a ranking model. The goal is to find parameters $\theta \in \Theta$ such that the ordering of the mapping of tasks $g_\theta(x_1) \geq \dots \geq g_\theta(x_n)$ corresponds to the ordering of their rewards $r_1 \geq \dots \geq r_n$. A ranking based on g_θ can be seen as a permutation π of the indices $\{1, \dots, n\}$.

The expected profit of a permutation $\pi(\mathbf{T})$ given a capacity W can be optimized directly using learning to rank. The key insight is that for a given

permutation π of tasks \mathbf{T} , the expected profit $\sum_{i=1}^N w_i \hat{r}_{\pi(i)}$ of a ranking is equivalent to its discounted cumulative gain (DCG), which is a commonly used class of metrics in learning to rank [97]. Typically, the DCG is defined with discount $\frac{1}{\log_2(i+1)}$ and gain $2^{t_i} - 1$ for $i \in \{1, \dots, n\}$. However, to match the expected profit, our formulation uses discount $\{w_i\}_{i=1}^N$ corresponding to the capacity distribution, gain equal to 1 for all i , and relevance \hat{r}_i . By dividing the DCG by its ideal value (IDCG), the normalized DCG (NDCG) is obtained: $\text{NDCG} = \frac{\text{DCG}}{\text{IDCG}}$ with $\text{NDCG} \in [0, 1]$.

Optimizing the NDCG (or equivalently, the expected profit) directly is challenging as it depends on the predicted relative positions of instances instead of the model’s outputs $g_\theta(\mathbf{x}_i)$. Nevertheless, various algorithms have been proposed for this task in the literature on learning to rank. In this work, we use the widely used LambdaMART [98], which uses a combination of the LambdaRank loss and gradient boosting of decision trees to construct the ranking model. In this way, we can train a ranking model g_θ to optimize the NDCG or expected profit of the assignments directly.

Finally, we need to specify each task’s relevance, which serves as a label according to which the ranking would ideally be constructed. Because the ranking corresponds to the priority that should be given to tasks, it should respect the ordering in terms of both outcome y_i and task payoffs \mathbf{v}_i . In other words, successful tasks should be more relevant than unsuccessful tasks, and a more profitable task should be more relevant. Therefore, we use a task’s reward r_i as a cost-sensitive relevance, as it uses an instance’s class label y_i and its cost matrix \mathbf{c}_i (see Equation (3.7)). By means of this approach, a positive task’s relevance is the profit (or equivalently, the negative cost) obtained by classifying it positively minus the profit obtained by classifying it negatively; vice versa for negative tasks. Thus, we obtain the relevance or reward r_i as follows:

$$r_i = y_i v_i^+ + (1 - y_i) v_i^- = y_i (c_i^{\text{FN}} - c_i^{\text{TP}}) + (1 - y_i) (c_i^{\text{TN}} - c_i^{\text{FP}}).$$

Alternatively, if the goal is to optimize for accuracy rather than cost, we can use class label y_i as the relevance of instance i .

3.5 Empirical results

In this section, we empirically evaluate and compare the two-stage and the integrated approach for a variety of tasks. We use publicly available data from a variety of application areas. For each application, the goal is to optimally allocate resources to optimize the expected cost given stochastic

capacity. All code for the experimental analysis will be made available online upon publication of this paper.

To compare the different approaches, we use gradient boosting to train the predictive models. Four different objectives are compared, depending on the task (classification or learning to rank) and on whether they aim to maximize precision or profit. First, `xgboost` and `csboost` are conventional approaches based on classification. More specifically, `xgboost` denotes a conventional classification model using the cross-entropy loss \mathcal{L}^{CE} (see Equation (3.6)), while `csboost` uses a cost-sensitive objective function \mathcal{L}^{AEC} (see Equation (3.8)). Second, `LambdaMART` and `csLambdaMART` are integrated predict-and-optimize approaches based on learning to rank. `LambdaMART` uses the binary class label y_i , whereas `csLambdaMART` uses task payoffs r_i as relevance. All models are implemented in Python using the `xgboost` package [99]. Gradient boosting is a popular methodology for both classification and ranking that has great predictive performance, as illustrated by recent benchmarking studies [55], [100].

3.5.1 Data

The data sets are enlisted in Table 3.2 and stem from different application areas: customer churn prediction, credit scoring and direct marketing. They all concern binary classification where tasks are either successful or unsuccessful. Resources are limited and stochastic: we assume a lognormal capacity distribution $W \sim \mathcal{LN}(\mu = \log(100), \sigma = 1)$.

The cost matrices are taken from earlier work on cost-sensitive classification (see Table 3.3). In churn prediction, we have c_i^{FP} and c_i^{FN} as, respectively, 2 and 12 times the monthly amount A_i for KTCC following Petrides and Verbeke [88]; whereas we follow the cost matrix given with the data set for TSC [101]. For credit scoring, we calculate the instance-dependent costs c_i^{FP} and c_i^{FN} as a function of the loan amount A_i following Bahnsen, Aouada, and Ottersten [43]. In direct marketing, a positive classification incurs a fixed cost $c_f = 1$, while missing a potential success incurs an instance-dependent cost equal to the expected interest given A_i , following Bahnsen, Aouada, and Ottersten [45]. Similarly, in fraud detection, a positive prediction leads to an investigation that entails a fixed cost c_f , and missing a fraudulent transaction leads to a cost equal to its amount A_i . We use $c_f = 10$, following Höppner, Baensens, Verbeke, *et al.* [87].

Table 3.2: **Data sets overview.** For each data set, we present the application area, abbreviation, number of instances (N), class imbalance in terms of proportion of positive instances (% Pos), and corresponding reference.

Application	Abbr.	N	% Pos	Ref.
Churn prediction	KTCC	7,032	26.58	[102]
	TSC	9,379	4.79	[101]
Credit scoring	HMEQ	1,986	19.95	[103]
	BN1	3,123	33.33	[55]
	BN2	7,190	30.00	[55]
	VCS	18,917	16.95	[104]
	UK	30,000	4.00	[55]
	DCCC	30,000	22.12	[105]
	GMSC	112,915	6.74	/
Direct marketing	UBM	45,211	11.70	[106]
	KDD	191,779	5.07	/
Fraud detection	KCCF	282,982	0.16	[107]
	KIFD	590,540	3.50	/
	ACCF	3,639,323	0.65	[108]

Table 3.3: **Cost matrices for the different application areas.** For each application, we present the costs for all outcomes in terms of predicted (\hat{y}) and actual (y) labels. A_i , c_i^{FN} , c_i^{FP} and Int_i represent instance-dependent costs and c_f is a fixed cost.

		y_i					y_i					y_i					y_i		
		0	1	1			0	1	1			0	1	1			0	1	1
\hat{y}_i	0	0	$12A_i$	0	\hat{y}_i	0	0	c_i^{FN}	0	\hat{y}_i	0	0	A_i/Int_i	0	\hat{y}_i	0	0	A_i	0
1	$2A_i$	0	0	0	1	c_i^{FP}	0	0	0	1	c_f	c_f	c_f	c_f	1	c_f	c_f	c_f	c_f

(a) Churn prediction (b) Credit scoring (c) Direct marketing (d) Fraud detection

3.5.2 Results

We present the results using various performance metrics to compare the different models. The main metric of interest is either the expected precision or the expected profit given the stochastic capacity distribution W , depending on whether accuracy or profit is the objective. Furthermore, we present several additional classification and ranking metrics to gain more insight into the differences between the methodologies. For each metric, we present the average over all data sets and test whether the best performance is significantly different from the others using a Friedman test on the rankings with Bonferroni–Dunn post hoc correction [58], [109], [110] (see Table 3.4).

Expected precision and expected profit

In terms of expected precision, LambdaMART is the best performing model. Two models optimize for accuracy: LambdaMART and xgboost. The ranking model, LambdaMART, outperforms the classification model, xgboost. In terms of expected profit, the cost-sensitive ranking model, csLambdaMart, performs best. Of the two models optimizing for accuracy, xgboost and LambdaMART, the ranking model again achieves better results, although this difference is not statistically significant. This increase in performance of the rankings models compared to classification models illustrates the potential benefit of our integrated ranking approach when capacity is constrained. We further compare the trade-off between profit and precision in Figure 3.2 by plotting the rankings for each data set. To get an idea of the densities for the different models, we estimate it using a Gaussian kernel and show it for probabilities greater than 0.5. Although the densities overlap, the ranking models outperform their classifying counterparts in their respective category. Again, this demonstrates the benefit of integrating the capacity constraint in the optimization.

Table 3.4: **Evaluation metrics overview.** We present an overview of the evaluation metrics. The average and standard deviation over all data sets are shown, with the best result denoted in **bold**. Results that are not significantly different from the best result are underlined ($\alpha = 0.05$). This is based on a Friedman test on the rankings with Bonferroni–Dunn post hoc correction. For both expected precision and profit, the ranking models perform best in their respective category. For the classification metric, average precision, the cost-insensitive classifier, xgboost, performs best. Conversely, for the ranking metrics, namely, Spearman correlation and the area under the cumulative profit curve, the ranking models outperform their classifying counterparts.

	Expected precision	Expected profit	Average precision	Spearman correlation	AUCPC
xgboost	0.496 \pm 0.08	0.212 \pm 0.05	0.942 \pm 0.01	-0.038 \pm 0.03	0.555 \pm 0.07
csboost	0.587 \pm 0.06	<u>0.294</u> \pm 0.05	0.908 \pm 0.02	0.226 \pm 0.07	0.566 \pm 0.07
LambdaMART	0.656 \pm 0.07	0.247 \pm 0.05	<u>0.937</u> \pm 0.01	-0.030 \pm 0.04	0.536 \pm 0.06
csLambdaMART	0.609 \pm 0.07	0.359 \pm 0.05	0.934 \pm 0.01	0.383 \pm 0.08	0.600 \pm 0.06

Average precision, Spearman’s ρ and AUCPC

These metrics weight all instances in the ranking equally, as opposed to the previous metrics that weighted instances depending on their probability of being processed given the capacity distribution [57]. On the one hand, we consider a classification metric: given the high degree of class imbalance for some data sets, we use the average precision. On the other hand, we consider two ranking metrics: the area under the cumulative profit curve and Spearman’s rank correlation coefficient ρ .

First, we assess the quality of the model’s predictions with a standard classification metric: average precision (AP). This metric summarizes the precision-recall curve and looks at the trade-off between precision and recall at different thresholds. As expected, the cost-insensitive classification model, xgboost, performs best. This result is no surprise, given that xgboost is a classification model that optimizes for accuracy. However, this conventional classification metric has only weak correlation with the expected precision, suggesting that it is not a good indicator of performance. Therefore, this results gives rise to an important insight: when there is limited capacity to act on predictions, traditional classification metrics are not a good indicator of performance.

We also adopt two ranking metrics. First, we use Spearman’s rank correlation coefficient to quantify the correlation between the ranking of the predictions and the ranking of the task payoffs. csLambdaMart is the best performing model, outperforming csboost. Moreover, both cost-insensitive

models have a correlation of approximately 0. This is as expected, as these models do not take payoff into account in their optimization. Second, the cumulative profit curve plots the profit that is realized as a function of the number k of first ranked instances, with $k \in [1, N]$. We compare the area under this curve with the area of a random ranking and one of the optimal ranking to obtain a value between 0 and 1. csLambdaMART performs best, though neither the difference with xgboost nor csboost is statistically significant. Compared to the classification metric, these results are more aligned with the expected precision and profit.

These findings indicate that metrics for evaluating the ranking quality, such as Spearman's ρ or the AUCPC, are more suitable than classification metrics, such as the average precision, for evaluating a model's performance under limited capacity. Moreover, our results suggest that ranking as a solution more closely aligns with the problem of allocating limited resources to uncertain tasks than classification, which is also confirmed by the superior performance of ranking models compared to classification models in terms of expected precision and expected profit. This represents an important insight, given the abundance of existing work using classification models for these application areas where capacity constraints are commonly encountered.

Top k metrics

Finally, we also consider metrics focusing solely on the top of the ranking. Given limited capacity, these are the instances that will be prioritized. We can evaluate this critical part of the ranking by looking at the precision and profit of the ranking for the first k instances for different values of k (see Figure 3.3). The ranking model optimizing for accuracy, LambdaMART, performs best in terms of precision@ k , while the ranking model optimizing for profit, csLambdaMART, has the best performance in terms of profit@ k . Again, these findings suggest that ranking models perform better given limited worker capacity, due to their ability to better prioritize the most important tasks at the top of the ranking. Indeed, given limited capacity, these are the tasks that will be executed.

3.6 Conclusion

In this work, we formally introduced and defined a commonly encountered problem: how to optimally allocate limited, stochastic resource capacity to tasks with uncertain payoff to maximize the expected profit. Moreover, we contribute by proposing a novel integrated solution using learning to rank and empirically comparing it with a more conventional predict-then-optimize approach using a classification model.

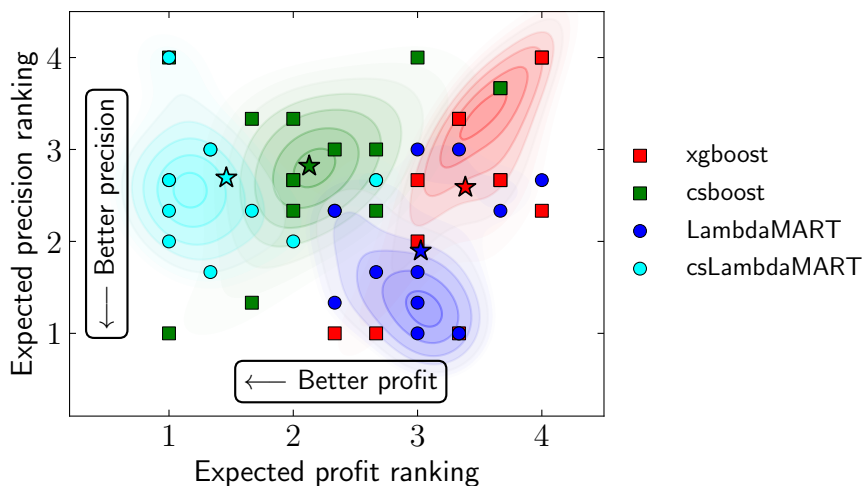


Figure 3.2: **Comparing the methodologies in terms of expected precision and profit.** We plot each methodologies' ranking in terms of expected profit and expected precision on each data set. For each method, the average ranking is shown with a star (☆). Moreover, the ranking density is fitted with a Gaussian kernel; for visual clarity, only probabilities greater than 0.5 are shown. On average, csLambdMART performs best in terms of expected profit, while LambdaMART performs best in terms of expected precision.

Our findings illustrate the benefit of approaching this problem as a ranking problem, which allows us to consider the availability of limited and stochastic resources. Theoretically, we show how the expected profit for a given capacity distribution can be optimized directly using learning to rank with a specific formulation of the net discounted cumulative gain as the objective. Empirical results for a variety of applications show that ranking models achieve better performance in terms of expected profit or expected precision, depending on the objective. Moreover, good results in terms of ranking metrics are more indicative of good performance in terms of expected profit compared to conventional classification metrics. This illustrates how ranking is more closely aligned with the problem at hand compared to classifying. In summary, in the common scenario where decision-makers are constrained by limited resources, deciding upon resource allocation using classification models is inferior to using learning to rank. These findings have important implications for practitioners in a variety of application areas.

Managerial implications Our findings have significant implications for

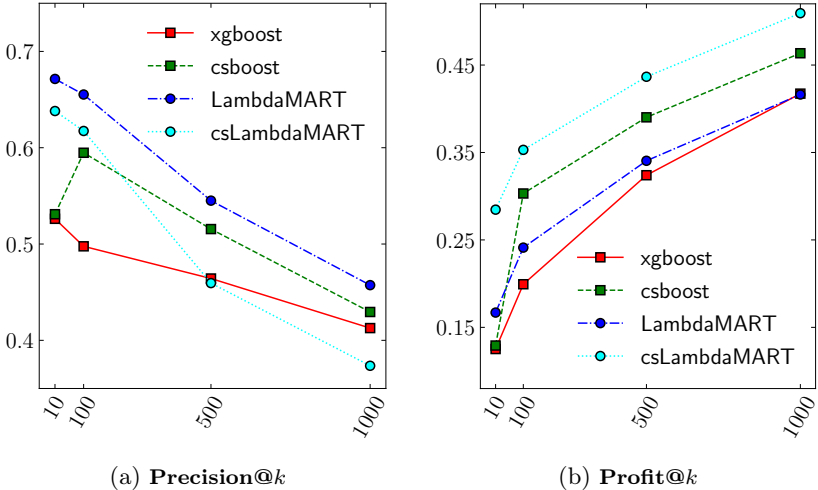


Figure 3.3: **Evaluating the top k ranked instances.** Precision (a) and profit (b) for obtained by the top k instances in the ranking for the different models averaged over all data sets. The ranking models outperform the classifiers in the metric they optimize for: LambdaMART is the best in terms of precision; csLambdaMART has the best profit.

practitioners that use predictive models for decision support in applications where resource capacity to act upon predictions is limited. This situation is commonly encountered in applications such as fraud detection, credit scoring, churn prediction, and direct marketing. Our work shows that, when decision-makers are faced with the challenge of optimally allocating limited, stochastic resource capacity to tasks with uncertain payoffs, they should consider adopting a ranking-based approach. We demonstrated that optimizing the expected precision or profit with a ranking model leads to improved decision-making compared to a commonly used approach using classification models. Similarly, we showed that ranking metrics provide a more accurate assessment of performance than classification metrics in settings where resources are constrained. Our results underscore the importance of embracing learning to rank over traditional classification methods in resource allocation decisions, which has important implications for practitioners seeking to maximize profitability and efficiency in applications with resource constraints.

Our work opens several promising directions for future research. For example, it would be interesting to consider a temporal variant of the assignment problem with tasks arriving sequentially in time. Although this problem has been studied extensively for stochastic or random arrival rates [111]–[113],

future work could consider the addition of a predictive ranking model to address uncertainty regarding task outcomes. Another possible extension would be to consider tasks that require varying degrees of resources. For example, in credit scoring, loans with a large principal require more resources.

Part III

CAUSAL INFERENCE



4

OPTIMIZING THE PREVENTIVE MAINTENANCE FREQUENCY WITH CAUSAL MACHINE LEARNING

Maintenance is a challenging operational problem where the goal is to plan sufficient preventive maintenance (PM) to avoid asset overhauls and failures. Existing work typically relies on strong assumptions (1) to model the asset's overhaul and failure rate, assuming a stochastic process with known hazard rate, (2) to model the effect of PM on this hazard rate, assuming the effect is deterministic or governed by a known probability distribution, and (3) by not taking asset-specific characteristics into account, but assuming homogeneous hazard rates and PM effects. Instead of relying on these assumptions to model the problem, this work uses causal inference to *learn* the effect of the PM frequency on the overhaul and failure rate, conditional on the asset's characteristics, from observational data. Based on these learned outcomes, we can optimize each asset's PM frequency to minimize the combined cost of failures, overhauls, and preventive maintenance. We validate our approach on real-life data of more than 4,000 maintenance contracts from an industrial partner. Empirical results on semi-synthetic data show that our methodology

based on causal machine learning results in individualized maintenance schedules that are more accurate and cost-effective than a non-causal approach that does not deal with selection bias and a non-individualized approach that prescribes the same PM frequency to all machines.

4.1 Introduction

Maintenance constitutes an intricate operational problem. The challenge is to avoid failures and costly overhauls, while simultaneously minimizing the cost of preventive maintenance (PM). We consider the problem of deciding on the frequency of PM interventions, where the optimal frequency minimizes the combined cost of both PM and detrimental outcomes resulting from deterioration, such as failures or overhauls. To optimize the PM frequency, existing work typically makes strong assumptions regarding the asset's *hazard rate*, i.e., the frequency with which failures and overhauls occur, and the *effect of PM* on this hazard rate. Moreover, existing maintenance policies assume asset homogeneity in the hazard rate and/or PM effect by not taking asset characteristics into account. We argue that all of these assumptions can be violated in practice.

First, most work assumes the asset's *overhaul and failure rates* follow a stochastic process that is known to the decision-maker, which is typically not the case in practice [114]. Moreover, estimating the parameters of the stochastic process from data is challenging due to censoring [115], [116] and still requires assuming a certain type of statistical distribution that might not coincide with the actual overhaul or failure rate. Finally, most existing work assumes asset homogeneity and does not incorporate the effects of the asset's characteristics on the overhaul and failure rates. In reality, however, an older asset might be more prone to failure and require more PM interventions than a more recent one.

Existing work also requires assumptions on *the effect of PM* on the overhaul and failure rates. A broad spectrum of maintenance effects have been studied in the literature, ranging from perfect maintenance, which restores the system to a state as good as new, to worst maintenance, where maintenance causes the asset to fail [117]. Existing approaches in imperfect maintenance assume that the effect is either deterministic or stochastic following a specified probability distribution. These assumed effects, however, might not always correspond to the actual effect. Moreover, the effect of PM is typically assumed to be identical for all assets. In reality, the effect of the same type of PM intervention could be very different for different assets. For example, changing a gear would likely have a different impact on a brand-

new asset compared to the exact same maintenance intervention on an old, worn-down asset.

In this work, we relax these assumptions regarding the hazard rate and the PM effect. Instead, we propose a data-driven maintenance policy that learns the effect of the PM frequency on the resulting overhaul and failure rates, conditional on the asset’s characteristics. This approach allows to flexibly learn the outcomes for different PM frequencies from historical, observational data using machine learning, rather than assuming a prespecified (or known) hazard rate and PM effect based on expertise, and to design an asset-specific PM schedule based on the learned outcomes.

These benefits are achieved by framing maintenance as a problem of causal inference. We argue that the challenge in maintenance is that, for each specific asset, we only observe one overhaul and failure rate corresponding to the PM frequency that was administered in practice. We never observe the counterfactual outcomes, i.e., what would have happened if that asset had received more or less maintenance. Because of this, we never know whether the optimal PM frequency was prescribed. Causal inference offers a solution to this problem by predicting each individual asset’s hypothetical overhauls and failures at different PM frequencies. By learning a model that predicts the overhaul and failure rate given the PM frequency, we can optimize the PM schedule to minimize the total estimated cost. Essentially, we propose using observational data to learn an asset-specific digital twin for maintenance that predicts the overhaul and failure rate should an asset be prescribed a certain PM frequency.

This work contributes to the extant literature on preventive maintenance by proposing a novel prescriptive framework for maintenance that prescribes each asset’s desired preventive maintenance frequency based on the estimated effect of PM on its overhaul and failure rates. To this aim, we frame maintenance as a problem of causal inference and leverage state-of-the-art machine learning methods for causal inference. These models learn to estimate an asset’s potential outcomes for different PM frequencies from observational data. Moreover, we formulate a prescriptive policy that uses the potential outcomes to decide on the optimal PM frequency that minimizes the total cost of failures, overhauls and PM interventions. Empirically, we contribute by demonstrating the use of our causal inference framework on a dataset consisting of more than 4,000 maintenance contracts of industrial equipment provided by an industrial partner. Finally, as our proposed approach itself comes with assumptions, we discuss their viability in the context of maintenance.

4.2 Related work

Maintenance has been studied extensively in operations research, with a wide variety of proposed maintenance policies [118]–[120]. Our work touches upon the literature on time-based maintenance, imperfect maintenance, condition-based maintenance, as well as prescriptive analytics and causal inference.

4.2.1 Time-based maintenance

We consider the problem of finding an optimal PM frequency, equivalent to finding the optimal period between PM interventions, known as time-based maintenance [121]. This approach has been widely studied and, because of its simplicity, it is still frequently used in practice [122], [123]. The key idea is to perform PM with a constant frequency throughout the asset’s lifetime. Typically, this optimal PM frequency is found by modelling the stochastic overhaul and failure rates using a statistical distribution and then finding the PM frequency that minimizes the estimated total cost [122].

The drawback of most existing time-based maintenance policies is that they model failures and overhauls using an assumed stochastic process. Estimating the parameters of this stochastic process can be difficult due to censoring. This is because, in reality, assets are often maintained before failure occurs. Even if the parameters of the stochastic process can be estimated from data, the process itself can be misspecified. Moreover, existing work on time-based maintenance typically does not consider asset heterogeneity. Our proposed approach does not rely on a parametric model of the asset’s overhauls and failures, but estimates each asset’s overhaul and failure rates given the PM frequency using a flexible machine learning model, conditional on that asset’s characteristics.

4.2.2 Imperfect maintenance

Most existing work assumes that preventive maintenance restores the system to a state that is as good as new. However, maintenance is typically imperfect in reality. Different maintenance effects have been studied in the literature, ranging from maintenance that restores the system to a perfect state to maintenance that makes the system’s state worse [117]. Consequently, developing maintenance policies that incorporate imperfect maintenance is an important research problem.

Existing work models the effect of imperfect maintenance as either stochastic (based on a known probability distribution) or deterministic [117], [124]. Stochastic effects include the (p, q) rule, where maintenance is as good as new with probability p and as good as old with probability $q = 1 - p$ [125]–

[127], and its age-dependent variant $(p(t), q(t))$ [128]. Other work assumes a deterministic effect. Improvement factor models assume that maintenance decreases the system's failure rate by a deterministic improvement factor [129]. Similarly, in virtual age models, imperfect maintenance decreases the system's age or failure rate with a deterministic factor q where $0 < q < 1$ [130], [131].

The literature has proposed methods for estimating the parameters of these imperfect maintenance models from data and corresponding goodness-of-fit tests [132]–[134]. However, these approaches still start from a (deterministic or stochastic) model of the PM effect that can be misspecified in practice. Moreover, the goodness-of-fit tests only verify whether the model corresponds to the asset pool globally. Conversely, our approach estimates an effect that is, first, model-free as it does not assume a certain type of effect and, second, machine-dependent, as it is based on individual characteristics. This is achieved by learning the overhaul and failure rates for different PM frequencies from observational data. Finally, a key difference with our approach is that we do not consider the effect of a single PM intervention, but rather focus on the outcomes over a period of time caused by different PM frequencies.

4.2.3 Condition-based maintenance

Data-driven, condition-based maintenance policies have recently gained importance in the maintenance literature [135]. Condition-based maintenance is a policy in which maintenance is optimized based on the machine's state or its characteristics [136], [137]. Especially relevant to our work are recent, predictive maintenance approaches that learn a predictive model from data to decide on the appropriate maintenance interventions [138], [139]. Various authors propose using neural networks due to their flexibility and ability to extract features from data [140]–[143].

A typical approach is to predict the machine's health from its characteristics and apply maintenance when a degradation threshold is reached. This is achieved by monitoring the machine's health using a data-driven model to predict whether a failure is imminent. When the perceived risk is too high, e.g., exceeding a degradation threshold, an intervention can be scheduled to avoid failure [e.g., as in 144]–[149]. Therefore, various works have proposed predicting failures using machine learning models [150]–[152] with several recent approaches based on neural networks specifically [153]–[162]. By incorporating asset characteristics in the estimation, predictive policies can account for asset heterogeneity.

The downside of condition-based approaches is that they only predict the

asset's deterioration and do not consider the impact of PM on this deterioration. The time at which the deterioration threshold is reached and PM is planned, might not correspond to the optimal timing to most effectively perform maintenance and remedy the deterioration. Ideally, maintenance should not be performed just before the asset fails, but at the time when it is most effective at lowering the asset's failure probability. To this end, it is important to estimate the asset's hazard rate *resulting from* a certain PM frequency, which is exactly what our approach aims to achieve.

Similar to the general literature on imperfect maintenance, existing condition-based approaches that consider imperfect maintenance also assume either a deterministic or stochastic maintenance effect. There exist three broad categories of condition-based approaches that account for imperfect maintenance [137]. A first category considers minimal maintenance with a *deterministic* effect, in which a system has several deterioration stages and imperfect maintenance returns the system to the previous stage. A second category considers *stochastic* effects, where the maintenance effect is governed by an assumed probability distribution. Finally, in *improvement factor* models, imperfect maintenance decreases the system's hazard rate with a (deterministic) factor between zero and one. To the best of our knowledge, no existing condition-based approaches aim to *learn* the effect of maintenance from data.

4.2.4 Prescriptive analytics and causal inference

Instead of assuming a hazard rate or PM effect, this work uses machine learning models to learn the effect of maintenance using techniques from causal inference. Causal inference aims to estimate the effect of a certain cause from data, in our case the failure rate resulting from a given PM frequency. Ideally, estimating maintenance effects would be done by conducting a randomized controlled trial: assigning different PM frequencies to a collection of (similar) machines and comparing the outcomes [163]. However, in practice, this approach can be prohibitively expensive, unfeasible, or even unethical (e.g., when considering life support equipment in hospitals). In maintenance, it would generally be challenging to randomly assign varying levels of PM to different machines, because an excessively low PM frequency might risk not ensuring minimal service levels. When randomized controlled trials are impossible, we need to rely on historical, observational data of machines and their maintenance to learn the outcomes caused by different PM frequencies.

The challenge of working with observational data is that this data is biased due to existing maintenance policies that were in use [163]. For example, as a result of an existing policy, machines more prone to failure might have been more likely to receive more maintenance in the past. This phenomenon, called selection bias or confounding bias, can result in biased estimates of

the counterfactual outcomes if ignored. Under certain assumptions, specialized tools from the causal inference literature can be used to tackle exactly this problem and learn causal effects from observational data, i.e., in the presence of selection bias [164]. Specifically, our work is related to learning potential outcomes for continuous-valued interventions [165]–[169], e.g., the PM frequency¹. Learning the outcomes for different levels of a continuous treatment is also referred to as learning a dose-response curve.

Causal inference has been applied to a variety of applications, such as personalized medicine [170], economic policy design [171], marketing [172], [173], and education [174]. Moreover, it is related to prescriptive analytics [175], [176], which has recently gained importance in operations research [177], [178]. This work uses causal inference to predict a machine’s failure rate and overhaul rate for different PM frequencies and, consequently, to decide upon a personalized PM schedule. To the best of our knowledge, this is the first application of causal inference for maintenance optimization.

4.3 Problem overview

This work aims to solve the problem of prescribing an asset’s PM frequency to minimize the costs resulting from overhauls, failures, and PM. In particular, we are motivated by the challenge faced by a provider of full-service maintenance contracts. The service provider is responsible for maintaining the client’s asset at a predetermined price [179]. To maximize its profit margin, the service provider needs to decide on the PM frequency that minimizes the costs of failures, overhauls, and PM. The PM frequency is usage-based and defined over a running period, which corresponds to a standardized number of running hours. For each contract, the service provider has access to contract characteristics, such as the type of machine it concerns and the machine’s age at contract start. We consider each machine as a single-unit system.

We assume the service provider conducts a single type of planned PM intervention and needs to decide on the frequency of these interventions. Planned PM aims to prevent two types of events: overhauls and failures. The first, *overhauls*, are unplanned, comprehensive maintenance interventions during which large parts of the machinery need to be replaced. From the viewpoint of the full-service maintenance provider, these are the most costly type of event. The second, machine *failures*, are also unplanned and result in an urgent need for maintenance as the machine stops running until corrective

¹The number of PM interventions is discrete, but the number of PM interventions per running period (i.e., PM frequency) is continuous-valued.

maintenance occurs. A failure also incurs a cost to the service provider that is smaller than the cost of an overhaul, but larger than the cost of PM.

The overall goal is to find each contract’s optimal PM frequency that minimizes the combined cost of planned PM, overhauls and failures, from the perspective of the service provider. Although planning more PM interventions is likely to result in less overhauls and failures, it comes at an increased maintenance cost. Therefore, the optimal PM frequency is a trade-off between costs resulting from planned PM on the one hand and costs resulting from unplanned overhauls and failures on the other hand. Due to heterogeneity in the contracts and associated machines, maintenance might need to be planned more frequently for some contracts. Therefore, it is important to consider the contract’s characteristics when deciding on the PM frequency. To this aim, the service provider has access to information on past contracts, including the administered PM frequency, and the overhaul and failure rates observed for that PM frequency.

Let each contract be defined as a tuple $(\mathbf{X}, T, O(T), F(T))$. $\mathbf{X} \in \mathcal{X} \subset \mathbb{R}^d$ denotes a vector of (static) characteristics of the contract and the associated machine. The treatment, in our case the PM frequency, corresponds to the number of PM interventions that were applied per running period, and is denoted as $T \in \mathcal{T} \subset \mathbb{R}^+$. Finally, $O \in \mathcal{O} \subset \mathbb{R}^+$ and $F \in \mathcal{F} \subset \mathbb{R}^+$ are the observed number of overhauls and failures per running period, i.e., the overhaul and failure rates. Following the causal inference literature, we adopt the Rubin–Neyman potential outcomes framework [180], [181] and denote the overhaul intensity O and failure rate F for a given maintenance frequency t as $O(t)$ and $F(t)$.

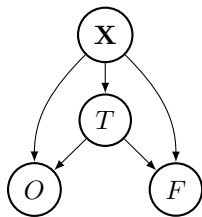


Figure 4.1: **Diagram illustrating the assumed causal relationships between the different variables.** X : Asset characteristics, T : PM frequency, O : Overhaul rate, and F : Failure rate.

The objective is to decide on each asset’s optimal PM frequency t_i^* that minimizes the total cost per running period. We assume a cost model per running period similar to [123]. Each asset i ’s cost per running period consists of the combined costs of PM, overhauls and failures, which all depend

on the decision-variable, i.e., the PM frequency t_i :

$$c_i(t_i) = \underbrace{c_t t_i}_{\text{PM}} + \underbrace{c_o o_i(t_i)}_{\text{Overhauls}} + \underbrace{c_f f_i(t_i)}_{\text{Failures}}, \quad (4.1)$$

for $i \in \{1, \dots, n\}$. We assume that the costs of PM, overhauls, and failures are deterministic and known ($c_t, c_o, c_f \in \mathbb{R}^+$).

To assist the service-provider’s decision-making, data is available on m past contracts $\mathcal{D} = \{(\mathbf{x}_i, t_i, o_i, f_i)\}_{i=1}^n$. For each of these past contracts, only one potential outcome was observed for O and F given that contract’s PM frequency T : $o_i(t_i)$ and $f_i(t_i)$. The other, counterfactual outcomes are never observed—this is known as the fundamental problem of causal inference [182]. The challenge in causal inference is to predict, for a new contract, the potential outcomes for all possible values of T using historical, observational data.

For each observed contract i , decisions regarding the administered PM frequency t_i were based on its characteristics \mathbf{x}_i according to a (possibly unknown) existing policy, resulting in selection bias or confounding bias in the data. In observational data, we can expect a relationship between an asset’s characteristics and the PM frequency it received. For example, the service provider might know from experience that older machines are more likely to fail when not receiving frequent PM and, because of this, typically prescribed more maintenance to those machines in the past. Factors that influence both the administered PM frequency and the outcome, the failure and overhaul rate, are called confounders. In this example, age is a confounder affecting both the received PM frequency and the resulting failure rate. We show the assumed causal structure of the problem in Figure 4.1.

The presence of confounders and selection bias is typically the case when working with observational data. This is because past PM frequencies were not assigned at random, but based on information on the contract and machine. Because of the associations between confounders and the PM frequency, assets that received relatively infrequent PM are different from assets that received relatively more frequent PM. This phenomenon, called selection bias, complicates learning the relationships between overhaul and failure rates, the PM frequency, and asset characteristics. Therefore, when learning a predictive model for estimating the overhaul and failure rate resulting from a given PM frequency from observational data, we are required to adjust for selection bias to obtain unbiased estimates.

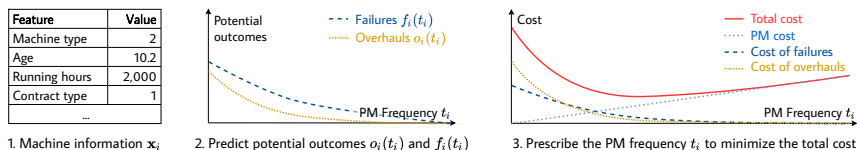


Figure 4.2: **Methodology overview.** We present a high-level overview of our methodology. First, machine characteristics \mathbf{x}_i are used to predict the potential outcomes in terms of overhauls $o_i(t)$ and failures $f_i(t)$. Based on these estimates, the total cost for different PM frequencies $t \in T$ can then be estimated. Finally, the PM frequency \hat{t}_i^* is chosen to minimize the total expected cost.

4.4 Methodology

Our methodology consists of a predict-then-optimize framework, see Figure 4.2 for a high-level overview. First, we predict each new contract’s potential outcomes, i.e., its overhaul $o_i(t)$ and failure rate $f_i(t)$ for PM frequencies $t \in T$, based on its characteristics \mathbf{x}_i . Therefore, the first step is to train a machine learning model to estimate these potential outcomes from observational data on past contracts \mathcal{D} . In a second phase, we use these predictions to estimate each contract’s total cost per running period for different PM frequencies $t \in T$. The PM frequency is chosen to minimize the resulting total cost of overhauls, failures, and PM.

In what follows, we first introduce standard assumptions that are required to estimate potential outcomes from observational data in Section 4.4.1. Second, we describe how we estimate the potential outcomes by learning a causal machine learning model from observational data. We use a state-of-the-art methodology called SCIGAN [169]. This is described in Section 4.4.2. Third, in Section 4.4.3, we describe how these predictions are used to determine each machine’s optimal PM frequency that minimizes the total estimated cost.

4.4.1 Assumptions

The challenge in estimating potential outcomes from observational data is dealing with selection bias. Learning unbiased estimates of the potential outcomes from observational data requires making three standard assumptions: consistency, overlap, and unconfoundedness [165], [169]. The first, *consistency*, means that each contract’s observed outcomes for O and F given PM frequency $T = t$ are its potential outcomes $O(t)$ and $F(t)$:

Assumption 1. Consistency $Y = Y(t)$ for all $t \in T$.

This assumption implies that there is only one version of the treatment and that the mechanism used to assign the treatment does not matter. It is violated if, for example, the prescribed PM is not performed for some assets, e.g., when some clients do not adhere to the PM frequency prescribed by the service provider. Consistency may seem straightforward, but ensures that the PM schedule prescribed by the service provider will be observed in practice.

The second assumption, *overlap or positivity*, ensures that each possible contract \mathbf{x}_i has a non-zero probability of receiving each frequency of PM interventions t_i :

Assumption 2. *Overlap* For all $\mathbf{x} \in \mathcal{X}$ with $p(\mathbf{x} > 0)$ and $t \in \mathcal{T} : 0 < p(t|\mathbf{x}) < 1$.

This implies that, for each observed machine, it was a priori possible to observe each PM frequency, although not necessarily with the same probability. This assumption would be violated when, for example, machines older than five years always receive at least ten PM interventions per running period. In that case, the probability of receiving a PM frequency lower than ten is zero for those machines, implying a violation of the overlap assumption. In that case, we would not be able to account for selection bias, as no observations would exist to infer what would happen to old machines at low PM frequencies.

The third and final assumption, *unconfoundedness or no hidden confounders*, ensures that there are no unobserved variables influencing both the treatment assignment T and a potential outcome $O(t)$ or $F(t)$:

Assumption 3. *Unconfoundedness* Conditional on machine characteristics \mathbf{X} , potential outcomes $O(t)$ and $F(t)$ are independent of the PM frequency T :

$$\{O(t), F(t) | t \in \mathcal{T}\} \perp\!\!\!\perp T | \mathbf{X}.$$

This assumption implies that all information that informed decisions regarding past PM frequencies are included in the data. This assumption would be violated if, for example, machines in some locations were maintained more frequently in the past, but no record of the machine's location was kept. If hidden confounders are present, it is impossible to adjust for the hidden confounder and, consequently, for selection bias based on the observed data. Given that Assumptions 1 to 3 are met, controlling for confounding resulting from machine characteristics \mathbf{x}_i allows accounting for selection bias in observational data and obtain unbiased estimates.

4.4.2 Predictive model to estimate the effect of the PM frequency on overhaul and failure rates

First, we need to predict each contract’s potential outcomes, i.e., its overhaul $o_i(t)$ and failure rate $f_i(t)$ for each PM frequency $t \in T$, based on its characteristics \mathbf{x}_i . To this aim, we learn two machine learning models $g_o : \mathcal{X} \times \mathcal{T} \rightarrow \mathcal{O}$ and $g_f : \mathcal{X} \times \mathcal{T} \rightarrow \mathcal{F}$ defined by parameters $\theta_o, \theta_g \in \Theta$. The goal is to obtain unbiased estimators of the potential outcomes:

$$g_o(t, \mathbf{x}) = \mathbb{E}[O(t)|\mathbf{X} = \mathbf{x}], \quad (4.2)$$

$$g_f(t, \mathbf{x}) = \mathbb{E}[F(t)|\mathbf{X} = \mathbf{x}]. \quad (4.3)$$

In this work, we learn g_o and g_f using SCIGAN, a recently proposed methodology for predicting potential outcomes of continuously-valued treatments that achieved state-of-the-art performance across a variety of settings [169]. Each model is learned in two steps. First, a generative adversarial network (GAN) [183] is trained to model the distribution of the potential outcomes, conditional on the contract’s characteristics. This is achieved by training two neural networks, where the generator network learns to generate counterfactual contracts that cannot be distinguished from factual, observed contracts by the discriminator network. In a second phase, the GAN is used to augment the observed training data with generated counterfactual samples. This way, the augmented data set contains all potential outcomes, including the factual outcome and the generated, counterfactual outcomes. This way, the fundamental problem of causal inference is alleviated as we “observed” all potential outcomes for each contract and, because of this, the augmented data set does not suffer from selection bias. Using the augmented data set, a predictive model can be trained to predict the potential outcomes in a supervised manner. For this, we again use a neural network. Each network is implemented as a multilayer perceptron (MLP). Appendix B.1 provides more information on the training and hyperparameter optimization of the models.

4.4.3 Optimization of the maintenance cost

The predicted potential outcomes allow estimating the costs incurred at different PM frequencies. It can be seen that, using the predicted potential outcomes, all terms in Equation (4.1) depend on the PM frequency t_i :

$$c_i(t_i) = c_t t_i + c_o o_i(t_i) + c_f f_i(t_i). \quad (4.4)$$

Each machine’s optimal PM frequency t_i^* is found by minimizing the expected cost: $t_i^* = \operatorname{argmin}_{t_i} c_i(t_i)$ for all $i \in \{1, \dots, n\}$. To account for het-

erogeneity in the contracts, the PM frequency is optimized for each specific machine.

4.5 Results

We validate our methodology empirically using data provided by an original equipment manufacturer that offers full-service maintenance contracts to their customer base. By optimizing the PM frequency, they can minimize the total cost of such a contract, resulting from PM, overhauls, and failures. In Section 4.5.1, we first present the data used in our experimental analysis. Section 4.5.2 describes the semi-synthetic data generating procedure that we used to evaluate the predicted potential outcomes and prescribed PM frequencies. In Section 4.5.3, we present the evaluation metrics and benchmarks used. Finally, Section 4.5.4 presents the empirical results of our experimental analysis.

4.5.1 Data

Our data set contains more than 4,000 full-service maintenance contracts. For each contract i , we have information \mathbf{x}_i relating to the characteristics of the machine, the contract, and maintenance-related events. An overview of the information available in the data is presented in Table 4.1 and an excerpt is shown in Table 4.2. Maintenance-related events (PM interventions, overhauls, and failures) are presented per running period, which is a set number of running hours. For reasons of confidentiality, the exact number of running hours per period is not revealed in this article. Costs are averaged over all events and re-scaled for reasons of confidentiality.

The data is preprocessed as follows. Categorical variables are encoded with dummies and \mathbf{x}_i is standardized. The PM interventions, overhauls, and failures that occurred throughout the contract are converted to the number of events per running period to calculate each contract's PM frequency, overhaul rate, and failure rate. For future contracts, the exact number of running hours might not be known when the contract starts, but an estimate would typically be available.

4.5.2 Semi-synthetic data generating procedure

In order to obtain a good predictive model, we need to be able to accurately predict the overhaul and failure rates at different PM frequencies. However, if we test this predictor's accuracy using only observational data, we can verify the model's ability to accurately predict the observed outcome, the

Table 4.1: **Data overview.** Overview of the available contract information on machine and contract characteristics, preventive maintenance interventions, overhauls, and failures. For confidentiality, we present PM interventions, overhauls, and failures per running period, which is an undisclosed number of running hours. Similarly, the costs are averaged and re-scaled.

Variable	Domain
Machine information	
Type	$\{1, \dots, 7\}$
Age at contract start (in years)	$[0, 39]$
Running hours at contract start	$[2500, 110000]$
Contract information	
Type	$\{1, 2\}$
Duration (in days)	$[180, 5850]$
Running hours during contract	$[0, 186000]$
Average running hours per year	$[300, 8500]$
Preventive maintenance per running period	
PM frequency	$[0, 20]$
Outcomes per running period	
Number of overhauls	$[0, 128]$
Number of failures	$[0, 185]$
Average costs (in €)	
Preventive maintenance	73
Overhaul	207
Failure	104

Table 4.2: **Data excerpt.** We present an excerpt of the data set, showing examples of covariates related to the machine and contract \mathbf{x}_i , and maintenance related events per running period: the observed PM frequency t_i , the overhaul rate $o_i(t_i)$, and the failure rate $f_i(t_i)$.

Machine information			Contract information				Outcome freq.		
Type	Age [years]	Running hours contract start	Type	Duration [days]	Running hours over contract	Running hours avg per year	PM t_i	Overhaul $o_i(t_i)$	Failure $f_i(t_i)$
4	0	528.88	1	1,826	12,391.63	2,434.67	1.42	0.19	0.91
5	12	77,301.37	1	1,764	29,131.68	4,907.42	2.24	2.57	7.24
6	15	39,312.72	1	2,555	8,906.65	2,694.49	2.89	5.92	8.47
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
2	16	61,948.75	0	1,764	21,303.56	3,912.07	4.02	0.25	2.52

overhaul and failure rates only at the observed PM frequency t_i (the observed outcome), but not the overhaul and failure rates if the machine had received more or less maintenance (the unobserved outcomes). This makes the evaluation of causal models challenging, as only one potential outcome is observed for each contract in our dataset. Therefore, we rely on semi-synthetic data to evaluate our model. This approach is commonly used in both maintenance [see e.g., 179] and causal inference [e.g., 170].

The key idea of the semi-synthetic setup is to create a test set containing each contract’s potential outcomes at all possible PM frequencies, instead of only the observed outcome at one administered PM frequency. This is achieved by generating the outcomes at all possible PM frequencies $o_i(t)$ and $f_i(t)$ for all possible PM frequencies $t \in T$, based on the contract’s real characteristics \mathbf{x}_i . This allows us to create (1) a training set with only one observed PM frequency for each contract, equivalent to observational data, and (2) a test set containing potential outcomes for all possible PM frequencies for each contract, which are never observed in reality but needed for evaluation.

Potential outcomes $o_i(t)$ and $f_i(t)$ are generated based on the observed characteristics \mathbf{x}_i and PM frequencies $t \in T$. For the failure rates, we have:

$$f_i(t) = 9\sigma\left(\underbrace{\mathbf{v}_f^\top \mathbf{x}_i}_{\text{Base rate}} - \underbrace{\frac{1}{10}\sigma(\mathbf{w}_f^\top \mathbf{x}_i)t}_{\text{PM effect}} + \underbrace{\epsilon_f}_{\text{Noise}}\right), \quad (4.5)$$

with $\mathbf{v}_f, \mathbf{w}_f \sim \mathcal{U}((0, 1)^{d \times 1})$ and $\epsilon_f \sim \mathcal{N}(0, 1)$. σ denotes the logistic function. This way, each machine has a base failure rate that is diminished by administering more frequent PM, where both the base rate and PM effect depend on the contract’s characteristics \mathbf{x}_i . The factor 9 rescales the average failure rate to roughly the same number in the original, observed data. For

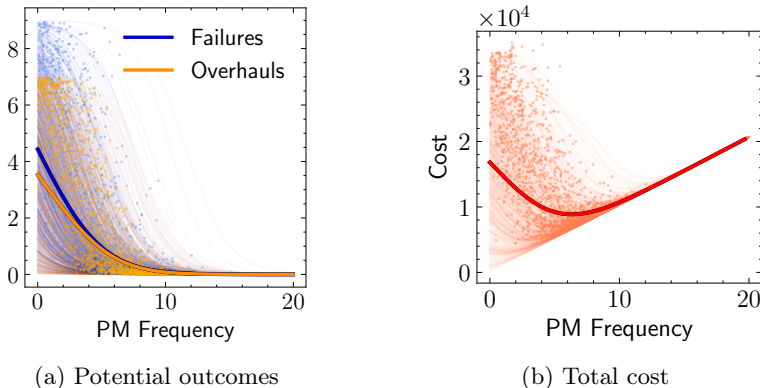


Figure 4.3: **Semi-synthetic data.** We represent the observed outcomes for contracts in the training and validation set by dots and the potential outcomes for contracts in the test set by a line. The bold lines illustrate the overhaul rate, failure rate, and total cost averaged across all contracts.

the overhaul rates, we similarly have:

$$o_i(t) = 7\sigma\left(\underbrace{\mathbf{v}_o^\top \mathbf{x}_i}_{\text{Base rate}} - \underbrace{\frac{1}{10}\sigma(\mathbf{w}_o^\top \mathbf{x}_i)}_{\text{PM effect}}t + \underbrace{\epsilon_o}_{\text{Noise}}\right), \quad (4.6)$$

where $\mathbf{v}_o, \mathbf{w}_o \sim \mathcal{U}((0, 1)^{d \times 1})$ and $\epsilon_o \sim \mathcal{N}(0, 1)$.

Using the semi-synthetic setup, the contracts in the test set have known potential outcomes for all possible values of $t_i \in \mathcal{T}$ based on Equations (4.5) and (4.6). Conversely, the training and validation sets include only one observed outcome for one PM frequency t_i . An illustration of a generated data set is shown in Figure 4.3. The training, validation, and test sets respectively consist of 50%, 25% and 25% of the data. Experiments are repeated five times.

In a first analysis, we use the PM frequency t_i that was observed in practice for the training and validation set. In other words, we only simulate the overhaul and failure rates. In a subsequent analysis, we evaluate our approach for different levels of selection bias by also controlling the observed PM frequencies in the training and validation set. For this, we manipulate the level of selection bias by making the observed PM frequencies t_i more or less dependent on the contract characteristics \mathbf{x}_i , using an approach similar to [169]. More specifically, we control the selection bias by assigning PM

frequencies based on sampling from a beta distribution:

$$t_i \sim 20 \text{Beta}\left(1 + \frac{\lambda\delta_i}{10}, 1 + \lambda\delta_i\right), \quad (4.7)$$

where $\delta_i = \sigma(\mathbf{w}_b \mathbf{x}_i)$ with $\mathbf{w}_b \sim \mathcal{U}((0, 1)^{d \times 1})$. δ_i ensures that assignment of the PM frequency is based on observed features \mathbf{x}_i . This way, we control the level of selection bias by setting λ . A value of $\lambda = 0$ results in $\text{Beta}(1, 1)$ or a uniform distribution. This implies that we randomly assign each machine's PM frequency with equal probability for each PM frequency in T . Therefore, $\lambda = 0$ results in a situation equivalent to a randomized controlled trial. Higher values of λ imply more selection bias, with $\lambda = 30$ resulting in an overall distribution of the PM frequencies over the entire training set that is similar to the observed distribution. In other words, $\lambda = 30$ corresponds to a realistic level of selection bias. Figure 4.4a shows each contract's distribution from which the PM frequency is sampled, for different values of λ . A higher value of λ increases the diversity of the different contracts' PM frequency distributions, resulting in more selection bias in the training data. Figure 4.4b compares the observed distribution of PM frequencies over all contracts in the original data and the overall distributions of PM frequencies resulting from different values of λ .

4.5.3 Performance evaluation

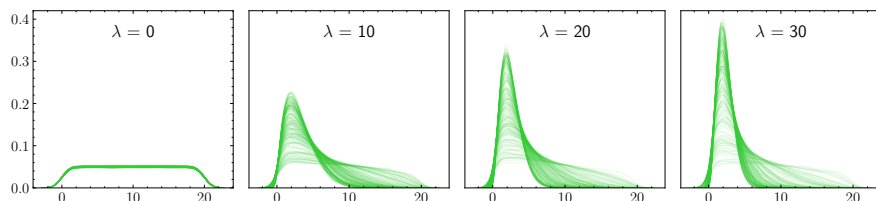
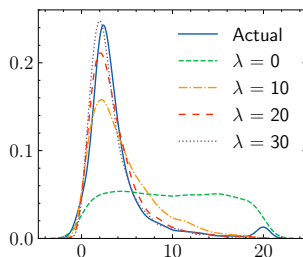
We evaluate our predict-then-optimize approach using three different metrics. First, we evaluate the ability of the machine learning model to accurately predict a contract's overhaul $o_i(t)$ and failure rate $f_i(t)$ over different levels of PM frequencies $t \in T$. This is measured using the mean integrated square error (MISE) [168], [184]:

$$\text{MISE} = \frac{1}{n} \sum_{i=1}^n \int_0^m (y_i(t) - \hat{y}_i(t))^2 dt, \quad (4.8)$$

for $y_i(t) \in \{o_i(t), f_i(t)\}$. Because we simulate the outcomes (see Figure 4.3), we know the ground truth $y_i(t)$ for each $t \in T$. Second, to evaluate the accuracy of the prescribed maintenance frequencies \hat{t}_i^* , we consider a variant of the policy error (PE) [168] that compares the prescribed PM frequency with the optimal PM frequency:

$$\text{PE} = \frac{1}{n} \sum_{i=1}^n (t_i^* - \hat{t}_i^*)^2. \quad (4.9)$$

The optimal PM frequency t_i^* can be found numerically by searching over the total cost incurred at each possible PM frequency $t \in T$. Third, we

(a) Effect of λ on the individual machines' PM frequency distributions

(b) PM frequencies

Figure 4.4: **Simulating selection bias.** (4.4a) We simulate a training data set where each contract's (observed) PM frequency is drawn from its probability distribution, shown in green, using Equation (4.7) for different values of λ . When $\lambda = 0$, each contract has equal probabilities of receiving each PM frequency between 0 and 20, corresponding to randomly assigned PM frequencies. Increasing λ makes the distributions more dependent on contract characteristics and therefore more diverse. This way, certain contracts will more likely receive less frequent PM, resulting in selection bias. Higher values of λ imply more diversity in the distributions and, consequently, more selection bias. (4.4b) We show how the PM frequency is distributed among the different contracts, both in reality and as a result of different values of λ . Larger values of λ result in more selection bias, with a value of 30 resulting in an overall PM frequency distribution close to the original.

Table 4.3: **Methodologies overview.** Our proposed, individual policy, SCIGAN-ITE, prescribes the PM frequency based on the individual treatment effect (ITE) estimated using SCIGAN. This proposed approach is analyzed using an ablation study and compared with two variants. The first, MLP-ITE, does not account for selection bias. The second, SCIGAN-ATE, is a general policy based on the average treatment effect (ATE) and is not individualized towards each individual machine.

Methodology	Selection bias?	Individualized?
SCIGAN-ITE	✓	✓
MLP-ITE	✗	✓
SCIGAN-ATE	✓	✗

evaluate the prescribed maintenance frequency in terms of costs using the policy cost ratio (PCR) that compares the costs of the estimated optimal PM frequency $c_i(\hat{t}_i^*)$ with the cost of the optimal PM frequency $c_i(t_i^*)$:

$$\text{PCR} = \frac{1}{n} \sum_{i=1}^n \frac{c_i(\hat{t}_i^*)}{c_i(t_i^*)}. \quad (4.10)$$

For all metrics, a lower value indicates better performance with 0 being the optimal value for MISE and PE and 1 for PCR.

Our proposed maintenance policy uses SCIGAN to learn the individual treatment effects (ITE), i.e., each contract’s overhaul and failure rate for different PM frequencies and will be referred to as SCIGAN-ITE. We benchmark against two other policies (see Table 4.3). First, a policy based on a neural network (MLP) that learns o_i and f_i given \mathbf{x}_i and t_i in a completely supervised manner without adjusting for selection bias (MLP-ITE). This allows us to assess whether there is a benefit of using the GAN to adjust for selection bias. Second, the average policy (SCIGAN-ATE) sets a single optimal \hat{t}^* for all contracts based on the average (instead of the individual) PM effect. This allows to validate the benefit of an individualized policy tailored towards each specific machine.

4.5.4 Empirical results

In this section, we present the results of the semi-synthetic experiments based on more than 4,000 maintenance contracts. Section 4.5.4 addresses (1) whether there is improved performance resulting from adjusting for selection bias and (2) whether an individualized policy per contract outperforms a general policy that does not take contract characteristics into account. In

Section 4.5.4, we show the importance of accounting for selection bias by evaluating the different policies' performance for varying levels of selection bias (Section 4.5.4).

Benefits of a causal, individualized PM policy

Table 4.4 reports the empirical results for the predictions and PM frequencies obtained using each methodology. The left part of Table 4.4 compares the ability of SCIGAN and MLP of accurately predicting the overhaul and failure rate at different PM frequencies. SCIGAN achieves the lowest error measured by the MISE. It predicts the overhaul and failure rate more accurately than the supervised MLP that does not account for selection bias. In the right part of Table 4.4, we assess the quality of the PM frequencies prescribed by the different approaches. These results show that the relatively more accurate predictions of the individualized, prescriptive approach (SCIGAN-ITE) also result in better PM frequencies. On the one hand, SCIGAN-ITE prescribes PM frequencies that are closer to the optimal PM frequency compared to the supervised (MLP-ITE) and non-individualized approach (SCIGAN-ATE), measured using the PE. On the other hand, SCIGAN-ITE also results in the lowest total cost as indicated by the PCR, achieving a cost that is 7% higher than the optimal policy, compared to 11% for MLP-ITE and 24% SCIGAN-ATE. The gap of 7% between SCIGAN-ITE and the optimal policy can be explained by the model being trained on limited data and the presence of noise in the data.

Figure 4.5 takes a closer look at these results, by showing how each model's performance of each contract individually, rather than looking only at the average performance over all contracts. The left panel in Figure 4.5 assesses how close each contract's PM frequency is to the optimal PM frequency, by showing each model's error distribution, i.e., the differences between the prescribed and optimal PM frequencies for all contracts. For SCIGAN-ITE most of the errors are close to zero, indicating that the prescribed PM frequency is typically reasonably close to the optimal PM frequency. By comparison, MLP-ITE and SCIGAN-ATE more frequently prescribe a PM frequency that deviate from the optimal PM frequency, illustrated by the heavier tails in their distributions. The right panel in Figure 4.5 looks at the costs resulting from each contract's prescribed PM frequencies, by showing the distribution of each contract's PCR, i.e., the cost resulting from the prescribed PM frequency relative to the cost incurred by the optimal PM frequency. SCIGAN-ITE typically frequently obtains a PCR close to one, indicating that it incurs costs that are close to the optimal policy. By comparison, MLP-ITE and especially SCIGAN-ATE more frequently incur costs that are much higher than the costs resulting from the optimal

Table 4.4: **Empirical evaluation.** We compare performance for the different policies over five simulation runs. We evaluate each model’s ability to accurately predict the potential outcomes $o_i(t)$ and $f_i(t)$ using the MISE, as well as each model’s ability to accurately prescribe PM frequencies (PE) and to minimize costs (PCR). For all metrics, a lower value is better.

	MISE	
	Overhauls	Failures
SCIGAN	7.71 ± 0.60	14.16 ± 1.68
MLP	10.25 ± 1.33	18.27 ± 3.65
	PE	PCR
SCIGAN-ITE	2.40 ± 0.46	1.07 ± 0.01
MLP-ITE	4.36 ± 1.25	1.11 ± 0.02
SCIGAN-ATE	8.77 ± 1.07	1.24 ± 0.04

PM frequency. These findings correspond to the findings averaged over all contracts in Table 4.4.

The improved performance of SCIGAN compared to a standard MLP suggests that learning PM effects from observational data requires accounting for selection bias. Moreover, the relatively worse performance of the non-individualized approach, SCIGAN-ATE, compared to the individualized approach, SCIGAN-ITE, shows the benefit of an individualized, machine-dependent policy for imperfect maintenance that takes into account machine characteristics and accounts for machine heterogeneity.

Importance of accounting for selection bias

The results in the previous section were obtained for the level of selection bias that was observed in reality, by using the PM frequencies in the training set as originally observed. In this section, we obtain more insight into the influence of selection bias by comparing the performance of SCIGAN-ITE and MLP-ITE for varying levels of selection bias. This is achieved by controlling the level of selection bias using λ (see Equation (4.7)). At $\lambda = 0$, there is no selection bias. In this case, each contract’s PM frequency is randomly drawn from the domain of all possible PM frequencies, with each contract having equal probabilities of receiving each PM frequency. In other words, setting $\lambda = 0$ results in data similar to a randomized controlled trial, which would be ideal for learning causal effects. Even though randomly assigning PM frequencies is not reasonable in the context of maintenance, this simulation

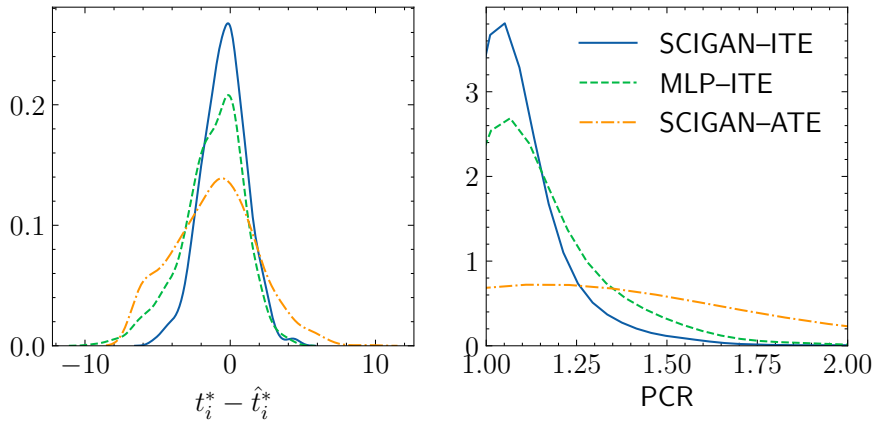


Figure 4.5: **Evaluating the policies' decisions.** We compare the accuracies and costs of the prescribed PM frequencies by looking at each model's performance over all contracts. (Left) We show how the differences between the prescribed and optimal PM frequency are distributed per model. (Right) We show the distribution of all contracts' policy cost ratios resulting from each model. Results are shown for one representative iteration.

allows us to study the influence of selection bias on performance. Increasing λ makes a machine's observed PM frequency less random and more dependent on its characteristics and, therefore, results in more selection bias.

Figure 4.6 compares SCIGAN's and MLP's abilities of predicting overhauls and failures, as well as their ability of prescribing good PM frequencies, for varying levels of selection bias. SCIGAN achieves good predictive performance in terms of MISE for the entire range of operating conditions, ranging from no selection bias and randomly assigned PM frequencies ($\lambda = 0$) to realistic levels of selection bias ($\lambda = 30$). Conversely, the MLP, a supervised approach that does not adjust for selection bias, accurately predicts the potential outcomes when the PM frequencies in the training set are randomized ($\lambda = 0$), but results in notably worse predictions compared to SCIGAN when selection bias is present at higher levels of λ . This result implies that it is important to adjust for dependencies between a contract's characteristics and its observed PM frequency when estimating PM effects from observational data. Similarly, SCIGAN is robust towards higher levels of λ and selection bias in terms of decision-making, illustrated by stable values for PE and PCR across different levels of selection bias, whereas MLP results in less accurate and more costly decisions as bias increases.

Observational data on maintenance operations is likely to contain selection

bias, because a machine’s PM frequency that was observed in the past will not have been assigned randomly, but based on machine characteristics—be it following a technician’s expertise or an existing maintenance policy. Our empirical results demonstrate the importance of dealing with selection bias when working with observational data. Moreover, our results indicate that the generative model in SCIGAN is able to accurately generate counterfactual outcomes to overcome selection bias, resulting in both better predictions and decisions compared to the MLP that does not use this generative model.

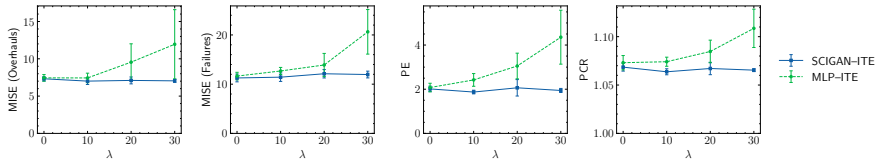


Figure 4.6: **Results for varying levels of selection bias.** We show results for different levels of selection bias in terms of λ (see Equation (4.7)). Although SCIGAN-ITE performs similar to MLP-ITE for lower values of λ , it has better performance for stronger levels of bias in terms of MISE, PE, and PCR.

4.6 Conclusion

This work proposes a novel way to optimize the preventive maintenance frequency. Our causal inference approach predicts how the failure and overhaul rate would be impacted by a certain PM frequency, taking the asset’s characteristics into account. This is achieved by relying on state-of-the-art machine learning methodologies for causal inference that learn an asset’s outcomes for different PM frequencies from observational data on assets that were maintained in the past. The benefit of our approach is that, unlike existing approaches, our methodology does not need strong assumptions regarding the failure or overhaul rate or PM effect. These are usually assumed to be known and are difficult to verify from data due to censoring, as assets are usually maintained before failure occurs. Moreover, existing approaches typically do not account for asset heterogeneity. Conversely, our approach is to learn an asset’s overhaul and failure rate resulting from a given PM frequency from observational data using flexible machine learning models. This allows to estimate what will happen for a contract given a certain PM frequency, in terms of overhauls and failures, and makes it possible to prescribe the PM frequency that minimizes the combined costs resulting from overhauls, failures, and PM.

Theoretically, we contribute by framing time-based maintenance as a prob-

lem of causal inference and by proposing a predict-then-optimize framework to solve this problem. Empirically, we validate our approach with semi-synthetic experiments using real-life data on more than 4,000 full-service maintenance contracts. We find that our proposed approach outperforms both an approach that does not account for selection bias and a non-individualized approach in terms of both accuracy and cost of the prescribed PM schedules. Moreover, we highlight the importance of dealing with selection bias when learning from observational data. Past maintenance decisions were likely not made at random, but based on the asset's characteristics. Because of this, machine learning models need to account for dependencies between asset characteristics and the observed PM frequency in order to obtain a good estimate of the overhaul and failure rate for a given PM frequency. These findings show that our proposed approach offers a powerful and flexible policy for individualized maintenance.

4.6.1 Limitations

Our data-driven approach requires observational data to train the machine learning models. When limited data is available, more simple machine learning methodologies based on, for example, linear regression can be preferred to the presented approach based on neural networks. Choosing and validating causal inference models is an active area of research [185], [186].

Causal inference not only requires data, but also requires that certain assumptions regarding the data are met. The first, overlap, implies that each asset could in principle receive each possible PM frequency, albeit not with the same probability. This requires a degree of flexibility or variability in how PM frequencies were assigned in the past. Alternatively, it might require some experimentation to provide insight into deviations from the existing policy. Overlap can be tested [187] and characterized [188] from data. Moreover, recent work has looked at characterizing uncertainty in regions where overlap is violated [189], [190].

The second assumption, unconfoundedness, is untestable in practice [165]. However, it can be assessed by people with domain-knowledge that were in charge of making maintenance decisions. The relevant question is whether all relevant information regarding the assignment of past PM frequencies is included in the data. If there are unobserved confounders, adequately adjusting for selection bias might not be possible, which would result in biased estimates of the overhaul and failure rate. Recently, sensitivity analyses have been suggested to assess the influence of hidden confounders [191], [192]. Similarly, methods have been proposed for quantifying ignorance regarding the potential outcomes due to possible violations of these assumptions [193].

4.6.2 Managerial implications

Optimizing maintenance using causal inference and machine learning offers a potentially flexible and powerful maintenance policy. Our approach prescribes the optimal PM frequency to each individual asset by comparing different counterfactual outcomes that would result from different maintenance frequencies, by learning a causal machine learning model from data on assets that were maintained in the past. Under the right conditions, causal inference represents a viable and performant paradigm for maintenance optimization. However, our approach also requires a different way of thinking about maintenance optimization. A completely data-driven policy for preventive maintenance is based on assumptions regarding the data and the models learned from this data. Therefore, maintenance practitioners should check whether their setting allows for causal inference, i.e., whether the requirements presented in Section 4.4.1 are met. If not, practitioners might consider altering their maintenance operations to satisfy these conditions, e.g., by running small-scale experiments to observe the effect of deviating from their existing policies.

4.6.3 Future work

In terms of future work, it would be valuable to not only optimize the frequency of one type of PM intervention, but also consider different possible interventions in terms of their depth and costs. This way, it would be possible to alternate cheap, quick visits and more expensive and thorough visits throughout the asset's lifetime. Moreover, it would be interesting to incorporate more flexible timing of maintenance interventions and consider sequences of different maintenance interventions, potentially prescribed based on real-time dynamic data obtained through sensors. Sequences of treatments have also received attention in the literature on causal inference [e.g., 194]–[196]. Finally, it would be interesting to look at ways of more closely integrating the predictive model in the decision-making step, e.g., by using approaches for integrated predict-and-optimize [197] or cost-sensitive learning [8].

5

NOFLITE: LEARNING TO PREDICT INDIVIDUAL TREATMENT EFFECT DISTRIBUTIONS

Estimating the effect of a treatment on an individual's outcome of interest is an important challenge in various fields, such as healthcare, economics, marketing, and education. Previous work in machine learning has focused on estimating the expected value of the treatment effect. However, effective personalized decision-making requires more than just the treatment expected effect; it requires knowing the entire treatment effect distribution. Knowing this distribution allows analyzing the treatment's expected utility or quantifying the uncertainty regarding a treatment's effect. This information is essential for prescribing optimal treatments. The ability of a model to predict accurate individual treatment effect distributions is captured by its likelihood. In light of this, we propose a novel neural architecture, **NOFLITE**, that uses normalizing flows to directly optimize this likelihood, while simultaneously learning flexible estimates of the individual treatment effect distribution. Experiments on various semi-synthetic data sets show that **NOFLITE** outperforms existing methods in terms of loglikelihood. Moreover, we illustrate how the

predicted distributions can enable an in-depth analysis of the treatment effect and more accurate decision-making.

5.1 Introduction

Knowing how a certain treatment or action will affect an instance’s outcome of interest is of great importance in various domains, such as healthcare [170], marketing [198], education [199], and economics [10]. A wide variety of existing work has looked at using machine learning (ML) for estimating the individual treatment effect, to help decision-makers optimize treatment assignment at an individual level. Existing work on treatment effect estimation in ML has proposed novel approaches based on a variety of different ML algorithms, including neural networks [200]–[203], Gaussian processes [204], and decision trees [205]–[207], as well as general meta-learners [208], [209].

In spite of this growing body of literature, existing work has almost exclusively focused on accurately estimating the *expected* value of the treatment effect. We argue, however, that a more comprehensive approach is needed: *to effectively support decision-making, we require accurately modeling the effect’s entire distribution*. Such an approach is essential for adopting ML for treatment decision-making in practice. First, it unlocks a wide range of descriptive statistics, enabling **a more detailed analysis** of the treatment effect. For example, it allows for reasoning about uncertainty of events resulting from a treatment: e.g., to get uncertainty intervals or estimate the probability of the treatment’s effect being positive. This way, the treatment effect distribution subsumes other common estimands which focus on a single attribute of this distribution, such as the mean, median, or other quantiles. Second, the treatment effect distribution is essential for **deciding upon the optimal treatment**. By pairing the treatment effect distribution with a utility function, we can obtain the expected utility (see Figure 5.1 for a graphical illustration). Utility functions have been influential in a variety of fields, including economics [210], [211], game theory [212], insurance [213], design [214], and healthcare [215].

Contributions. The goal in this work is to estimate individual treatment effect distributions. To obtain good estimates of this distribution, we require learning a model with a high *likelihood* from observational data. This work addresses treatment effect estimation through this lens and, in doing so, makes three contributions. **1.** To learn models with high likelihood, we propose a novel neural architecture, **NOFLITE**, which employs normalizing flows to learn flexible estimates of the treatment effect distributions. **2.** We propose an end-to-end training strategy to directly maximize the metric of

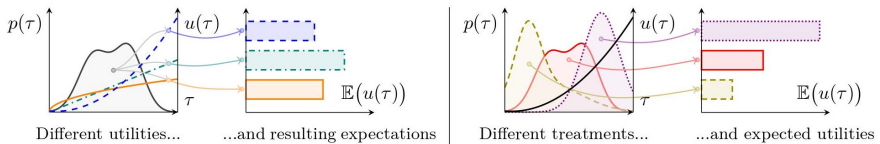


Figure 5.1: **Optimizing treatment decisions requires knowing the treatment effect distribution.** Predicting the treatment effect distribution $p(\tau)$ allows for assessing a treatment’s utility by pairing it with a utility function $u(\tau)$ and obtaining its expected utility $\mathbb{E}(u(\tau))$. Consequently, the individual treatment effect distribution is instrumental to analyzing treatment utilities (on the left) and comparing treatment preferences (on the right). *Left:* For a given treatment and the corresponding treatment effect distribution $p(\tau)$, we can compare different utility functions $u(\tau)$ and their expected utilities. *Right:* Similarly, for a given utility function $u(\tau)$, different treatment effect distributions $p(\tau)$ incur different expected utilities. These types of analyses are not possible using only the individual’s expected treatment effect $\mathbb{E}(\tau)$, illustrating the importance of estimating the entire treatment effect distribution $p(\tau)$ for personalized decision-making.

interest—the model’s likelihood—while regularizing to account for treatment assignment bias. This way, we aim to learn an unbiased model with high likelihood given the (unknown) test distribution from observational training data. **3.** We evaluate our method empirically and compare performance to other state-of-the-art approaches using three semi-synthetic data sets. Contrary to existing work, this evaluation is centered around the model’s likelihood.

5.2 Related work

Our work builds upon existing work in (individual) treatment effect estimation and normalizing flows. In this section, we discuss the most closely related literature for each category in turn.

5.2.1 Treatment effect estimation

A wide variety of methodologies have been proposed for estimating the *expected* individualized treatment effect (ITE). For a complete overview, we refer to [164]. Even though estimating *the distribution* of the ITE has received far less attention, several methods that were originally proposed for estimating the *expected value* are nevertheless capable of learning distributional estimands. These methods include Bayesian Additive Regression Trees

(BART) [205] and Causal Multi-task Gaussian Processes (CMGP) [204], as well as a variety of metalearners – general combinations of supervised ML methods [208]. Particularly relevant to our work are recent approaches based on generative neural networks. On the one hand, generative adversarial networks have been used to learn the counterfactual distribution and deal with confounding bias, for binary treatments [GANITE; 216] and multiple treatments with continuous dosages [SCIGAN; 169]. On the other hand, variational autoencoders have been used to adjust for a hidden confounder using a (noisy) proxy [217].

To the best of our knowledge, the only work explicitly looking at learning individual treatment effect distributions is the recently proposed Collaborating Causal Networks (CCN) [218]. As opposed to NOFLITE, CCN does not learn by maximizing the model’s likelihood, but rather uses collaborating networks [219], where one network is trained to estimate the cumulative distribution function and another model estimates its inverse. An additional, more subtle distinction is that they focus on estimating and evaluating the distribution of the potential outcomes, instead of the treatment effect distribution. A structured comparison of existing methodologies and the proposed NOFLITE is provided in Table 5.1.

In this work, we rely on a generative neural network. More specifically, we estimate the treatment effect distribution using normalizing flows, a flexible type of generative model that transforms a distribution to a simple prior through a series of learned transformations. Similar in spirit to our work is the recent Interventional Normalizing Flows [220]. They propose to use normalizing flows to estimate the density of the *average* treatment effect resulting from an intervention, building upon the theoretical results of [221]. In contrast, we aim to predict distributions at an *individual* level.

Although our work focuses on the static setting, other work has looked at forecasting treatment outcomes over time [12], [222]. In this setting, CF-ODE has recently been proposed for learning uncertainty estimates for treatment outcomes over time [223].

The fact that the expectation of the treatment effect is insufficient for decision-making has motivated previous work in related areas. Most closely related to ours are methods for predicting the treatment effect distribution at the population level, instead of the individual level [220], [221], [224]–[226]. Other work has been done on different, but related problems, such as estimating quantile treatment effects [227], [228], finding the treatment regime that optimizes the quantile treatment effect [229], conformal inference of treatment effects [230], [231], or bounding the potential outcomes to support decision-making [232]. More generally, other related work in machine learn-

Table 5.1: **Literature table.** We compare NOFLITE with existing methodologies for estimating individual treatment effect distributions. Three dimensions are considered: **(1)** whether a neural network is used, **(2)** whether the model’s likelihood \mathcal{L}_θ is optimized directly, and **(3)** whether it adjust for confounding.

Method	Neural?	Likelihood?	Confounding?	Ref.
BART	✗	✓	✗	[205]
CMGP	✗	✓	✓	[204]
CEVAE	✓	✗	✗	[217]
GANITE	✓	✗	✓	[216]
CCN	✓	✗	✓	[218]
NOFLITE	✓	✓	✓	(Ours)

ing aims to estimate confidence intervals or distribution properties for more comprehensive off-policy evaluation [e.g. 233]–[238] or learns to optimize the distribution of returns of a reinforcement learning agent [239], [240].

5.2.2 Normalizing flows

Our approach builds upon a type of deep generative model called normalizing flows. Normalizing flows offer distinct advantages over other types of generative models like generative adversarial networks [241] or variational autoencoders [242]. The key benefit of using normalizing flows is that they allow for an exact evaluation of the density. Consequently, the model can be directly optimized for the metric we are interested in: the model’s likelihood. We provide a brief introduction to normalizing flows below. For a more detailed overview, we refer to [243].

A normalizing flow is an invertible mapping $\mathbf{g} : \mathcal{Y} \mapsto \mathcal{Z}$ from the empirical/original data space \mathcal{Y} to a latent space \mathcal{Z} [244], [245]. During *training*, the flow learns to map the empirical distribution $p(y)$ to a known (simple) prior distribution $p(z)$, typically a Gaussian distribution. The mapping \mathbf{g} consists of a series of invertible transformations $\mathbf{g}(y) = g_1 \circ \dots \circ g_k(y)$ with parameters θ learned by a neural network. This way, the density can be obtained using the change of variables formula:

$$p(y) = p_Z(\mathbf{g}(y)) \left| \det \left(\frac{\partial \mathbf{g}(y)}{\partial y} \right) \right|. \quad (5.1)$$

Using this formulation, we can evaluate the model’s density exactly. Consequently, we can optimize the mapping \mathbf{g} to directly maximize the model’s

likelihood. After training, *inference* can be done by sampling from the simple prior $p(z)$ and transforming the sample based on the inverse flow $\mathbf{g}^{-1}(z)$.

Normalizing flows have been successfully applied in a wide range of tasks, such as generating images [245]–[247], audio [248]–[251], and graphs [252], as well as reinforcement learning [253]–[255] and energy forecasting [256], [257].

Different families of transformations g have been proposed in the literature on normalizing flows. In this work, we use deep sigmoidal flows [258]. Our choice is motivated by sigmoidal flows offering a flexible transformation – the resulting flow is a universal approximator – and their excellent empirical performance. Deep sigmoidal flows use a transformation g that is a strictly monotonic neural network. The parameters of this network are given by a conditioner network. To comply with the monotonicity requirement, the learned transformer parameters are restricted to strictly positive weights and strictly monotonic activation functions—more specifically, a sigmoid activation. Although the inverse transformation g^{-1} is not known analytically, it can be approximated numerically. There is a wide variety of other flow transformations that could potentially also be used with NOFLITE, including gaussianizations flows [259], [260], residual flows [261], or neural spline flows [262].

5.3 Problem Formulation

Notation. We describe our problem setting using the Neyman-Rubin potential outcomes framework [263]. Let each instance be defined by covariates $X \in \mathcal{X} \subset \mathbb{R}^d$, a binary treatment indicator $T \in \mathcal{T} = \{0, 1\}$, and an outcome $Y \in \mathcal{Y} \subset \mathbb{R}$. Let the potential outcomes $Y^{(0)}, Y^{(1)} \in \mathcal{Y} \subset \mathbb{R}$ be defined as the outcomes that would be observed given treatment $T = 0$ and $T = 1$.

Goal. Given an individual’s covariates x , we are interested in predicting its individual treatment effect¹ $p(\tau)$, i.e., the difference between both potential outcomes $\tau = Y^{(1)} - Y^{(0)}$. Most existing work aims to learn an individual’s *expected* treatment effect:

$$\mathbb{E}(\tau) = \mathbb{E}(Y^{(1)} - Y^{(0)} \mid X = x). \quad (5.2)$$

¹When using the term Individual Treatment Effect (ITE), we refer to the instance’s measured covariates included in X . Note, however, that these covariates need not completely describe this individual and, because of this, may refer to multiple individuals. The only requirement is that X satisfies the ignorability assumptions (1-3). Because of this distinction, earlier work has argued that it is more precise to denote τ as the Conditional Average Treatment Effect (CATE), see [264].

We argue, however, that this expectation itself is insufficient for decision-making in many applications. Instead, we need to learn the *distribution* of the individual treatment effect:

$$p(\tau) = P(Y^{(1)} - Y^{(0)} \mid X = x) = P(Y^{(1)} \mid X = x) - P(Y^{(0)} \mid X = x). \quad (5.3)$$

This distribution can be used to optimize an individual’s treatment. For example, a decision-maker can use it to assess probabilistic statements, such as the probability of the treatment effect being strictly positive $p(\tau > 0)$. Alternatively, the ITE distribution can be used to evaluate treatment decisions by pairing it with a (personalized) utility function to obtain the expected utility: $\mathbb{E}(u(\tau)) = \int u(\tau)p(\tau) d\tau$, see Figure 5.1.

Our goal is to obtain good estimates of the treatment effect distribution, i.e., to obtain a model $\theta \in \Theta$ with a *high likelihood* $p(\tau|\theta) = \prod_{i=1}^n p(\tau_i|\theta) = \prod_{i=1}^n p(y_i^{(1)} - y_i^{(0)}|\theta)$ given a (hypothetical) test set $\mathcal{D}_{\text{test}} = \{(x_i, y_i^{(0)}, y_i^{(1)})\}_{i=1}^n$ containing both counterfactuals. Compared to point estimates such as the mean squared error, the likelihood can incorporate uncertainty and capture the entire data distribution. This way, it provides a more comprehensive measure of model performance and facilitates more robust decision-making.

Data and assumptions. We assume access to an observational dataset $\mathcal{D}_{\text{train}} = \{(x_i, t_i, y_i^{(t_i)})\}_{i=1}^n$ sampled from the joint distribution $p(X, T, Y)$. Learning a model for estimating the individual treatment effect distribution from this data is challenging for several reasons. We only observe one factual outcome $Y^{(t)}$ in practice, while the other, counterfactual outcome $Y^{(1-t)}$ is never observed. Consequently, the treatment effect itself is never observed, which is known as the fundamental problem of causal inference [182]. Additionally, because the data are observational, treatments were assigned by a (potentially unknown) policy based on instance covariates. Therefore, it is necessary to adjust for confounding in order to obtain unbiased estimates of $p(y^{(t)})$ and, consequently, $p(\tau)$. A final challenge is that we need to learn each individual’s entire distribution based on only one sample for each individual by leveraging data from similar individuals.

To identify the individualized treatment effect from observational data, we require the following standard assumptions [263], [265]:

Assumption 4 (Consistency). *An instance’s observed outcome given a treatment is equal to its potential outcome: $Y \mid X, T = Y^{(t)} \mid X$.*

Assumption 5 (Overlap). *Each instance has a strictly positive probability of receiving each treatment: $0 < \mathbb{P}(T = t \mid X = x) < 1, \forall t \in T, \forall x \in \mathcal{X}$.*

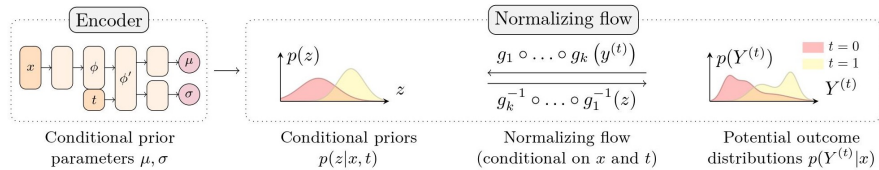


Figure 5.2: **NOFLITE architecture overview.** We visualize the S-learner configuration of our method. *Optimization:* NOFLITE learns (1) a conditional prior $p(z|x, t)$ for each instance, and (2) an invertible mapping $\mathbf{g}(y)$ from that conditional prior to the empirical distribution $p(y^{(t)}|x)$, possibly conditioned on x and/or t . Both the encoder and normalizing flow use neural networks and are trained with gradient descent. *Inference:* Based on input (x, t) , the conditional prior $p(z|x, t)$ is estimated. Samples z are drawn from this prior and transformed using the inverse flow $\mathbf{g}^{-1}(z)$ to estimate the distribution of each potential outcome $p(y^{(t)}|x)$.

Assumption 6 (Unconfoundedness). *Potential outcomes are independent of the treatment given the covariates:* $(Y^{(0)}, Y^{(1)}) \perp\!\!\!\perp T \mid X$.

5.4 NOFLITE: Estimating ITE distributions using normalizing flows

To tackle the problem formulated above, we propose NOFLITE²: a neural architecture using normalizing flows for estimating individual treatment effects distributions. A high-level overview of the architecture is shown in Figure 5.2. NOFLITE consists of two parts. The first part learns to predict a simple, conditional prior $p(z|x, t)$, in this case a Gaussian distribution parametrized by (μ, σ) . The second part learns to transform this prior to a more complex posterior distribution of the potential outcome $p(y|x, t)$. The entire model is trained end-to-end by directly maximizing its likelihood, while regularizing to deal with confounding. Both the model architecture and training procedure are explained in more detail below.

5.4.1 Architecture

NOFLITE’s architecture consists of two parts (see Figure 5.2): (1) an encoder f , i.e., a neural network that estimates a simple, parametrized prior

²All code is available at <https://github.com/toonvds/NOFLITE>.

distribution, and (2) a normalizing flow \mathbf{g}^{-1} to transform this prior to a more complex posterior distribution. In the following, we provide a detailed description of both modules.

Encoder: learning conditional priors

The first part of the model is a neural network that encodes the input (x, t) as a simple prior distribution $p(z|x, t)$. This prior can be seen as a first approximation of the empirical distribution. We use a normal distribution defined by parameters μ and σ : $f : (X, T) \mapsto (M, \Sigma)$. Depending on the application, other distributions could be used, such as a uniform or log-normal distribution. The only requirements are that the distribution is defined by a finite set of parameters and that its likelihood can be computed analytically.

To deal with confounding, x is first transformed to a balanced representation ϕ . This is achieved by learning a representation in which the distributional distance between the treated and non-treated representations is minimized [200], [201]. The training procedure is discussed in detail below.

Normalizing flow: learning complex posteriors

Although the encoder could be trained on its own to maximize the likelihood of the conditional prior, the model's hypothesis space would in that case be limited to simple parametric distributions, such as the normal distribution. Therefore, to augment the representational capacity of our model, the second part of the network uses a normalizing flow that gradually transforms the simple prior distribution $p(z|t, x)$ into a more complex posterior distribution $p(y^{(t)}|x, t)$ that can match more complex potential outcome distributions. The number of flow transformations k in \mathbf{g} is a hyperparameter that can be tuned. This number can be used to tune the complexity of the transformation and, therefore, the complexity of the posterior distribution.

Normalizing flows are an active field of research and, as such, new flow transforms continue to be proposed. In principle, any type of normalizing flow is compatible with NOFLITE, as long as some requirements are fulfilled. First, they have to be compatible with a univariate distribution (e.g. coupling layers [245] are not suitable). Second, we have to be able to compute both \mathbf{g} and \mathbf{g}^{-1} . Nevertheless, these requirements leave a variety of flow transformations [e.g., 261], [262], with several potential candidates that have been shown to be universal approximators [e.g., 258], [260].

In this work, we instantiate \mathbf{g} with a sigmoidal flow [258]. This flow type offers great flexibility and is a universal approximator. The transformation is defined by a monotonic transformer network, whose parameters are

estimated by a second conditioner network. By constraining the outputs of the conditioner network, the monotonicity of the transformer network is ensured. Specifically, the transformer network has strictly positive weights and strictly monotonic activation functions. The conditioner network can take different factors into account, such as covariates x and/or treatment t . Although the inverse mapping is not known analytically, we can use an interpolation search to find it numerically.

Metalearner configuration

We implement our model in two metalearner configurations. For the *S-learner* configuration, the encoder concatenates the balanced representation ϕ with the treatment t and uses this combination as inputs to predict the parameters of the conditional prior $p(z|x, t)$. For the *T-learner* configuration, both treatments have individual output heads after the shared balanced representation ϕ . Similarly, we define two corresponding metalearners for the normalizing flow. The *S-learner* shares the flow across treatments, potentially conditioning the flow transformation on the treatment—depending on the flow type. The *T-learner* learns a separate flow per treatment. Either one might be more suitable, depending on the data generating process; we see this is a hyperparameter that can be tuned.

5.4.2 Optimization

The model is trained end-to-end to simultaneously estimate accurate treatment effect distributions by maximizing the likelihood, while regularizing to deal with confounding:

$$\mathcal{L}_{\text{NOFLITE}} = -\mathcal{L}_{\text{LL}} + \lambda \mathcal{L}_{\text{MMD}} \quad (5.4)$$

with hyperparameter $\lambda \in \mathbb{R}^+$ trading-off likelihood maximization and balancing.

To calculate the *model likelihood*, the encoder and normalizing flow cooperate and meet in the middle, i.e., in the latent space Z . On the one hand, the encoder maps the inputs (x, t) to a prior distribution $\mathcal{N}(\mu, \sigma)$. On the other hand, the normalizing flow g maps the outcome y to a latent z . The goal of both components is to cooperate and maximize the likelihood of z given $\mathcal{N}(\mu, \sigma)$. This results in the following training objective:

$$\mathcal{L}_{\text{LL}} = \log p_Z(\mathbf{g}(y)) + \log \left| \det \left(\frac{\partial \mathbf{g}(y)}{\partial x} \right) \right|. \quad (5.5)$$

The first term, the likelihood of $\mathbf{g}(y)$, can be computed analytically given the Gaussian prior from the encoder. The second term, the log determinant of the Jacobian, can be calculated from the flow transformation. This

way, using a normalizing flow allows computing the likelihood exactly and, therefore, optimizing it directly. In practice, we minimize the negative log-likelihood $-\mathcal{L}_{LL}$, which is equivalent to maximizing this log-likelihood.

We learn a *balanced representation* to deal with confounding [200], [201]. This is achieved by minimizing the distributional distance between the representations of different treatments. Specifically, we use the linear Maximum Mean Discrepancy (MMD) [266]:

$$\mathcal{L}_{\text{MMD}} = 2 \left\| \frac{1}{n_0} \sum_{i \in \mathcal{D}_0} \phi(x_i) + \frac{1}{n_1} \sum_{j \in \mathcal{D}_1} \phi(x_j) \right\|^2, \quad (5.6)$$

where \mathcal{D}_0 and \mathcal{D}_1 denote the control and treatment group with n_0 and n_1 elements (per batch), respectively.

5.4.3 Inference

Inference is done in two steps. For each instance, samples are drawn from its prior distribution $z \sim N(\mu(x, t), \sigma(x, t))$. These samples are then transformed to samples from the posterior using the inverse mapping: $\hat{y} = \mathbf{g}^{-1}(z)$. After this process is completed for both treatments, a sample for the ITE is obtained by taking the difference between each sampled potential outcome: $\hat{\tau} = \hat{y}^{(1)} - \hat{y}^{(0)}$.

5.5 Results

In this section, we evaluate our proposed approach and compare it with several benchmarks. Our main goal is to assess whether NOFLITE learns to predict accurate individual treatment effect distributions from different types of observational data sets. More specifically, our experiments aim to answer three questions. (1) *Does NOFLITE predict accurate individual treatment effect distributions?* This is our main question of interest. We evaluate this with the loglikelihood. (2) *Does the predicted distribution allow for a more detailed analysis of individual the treatment effect, based on statistics derived from this distribution?* This is evaluated by looking at the predicted distribution’s expected value (using the precision in estimation of heterogeneous treatment effects), as well as the derived confidence intervals (using the intersection-over-union and empirical coverage). (3) *Does the predicted distribution enable accurate decision-making when paired with a utility function?* We evaluate this by looking at the accuracy of the recommended treatments.

The remainder of this section starts by describing our experimental setup, including data sets and benchmark methodologies in Section 5.5.1 and evaluation metrics in Section 5.5.2. The empirical results are presented in Section 5.5.3. More information on hyperparameter tuning for NOFLITE is provided in Appendix C.2.

5.5.1 Data and benchmarks

Evaluating individualized estimates of the treatment effect is challenging, because only one outcome is observed and, because of that, we do not have access to the ground truth. To overcome this challenge, we follow existing work and evaluate based on semi-synthetic data. More specifically, in this work, we evaluate NOFLITE using commonly used benchmark data sets: IHDP, EDU, and News. These are introduced briefly in the following; Appendix C.1 provides more detailed information on the data generating processes.

IHDP. The Infant Health and Development Program [IHDP; 205] is a semi-synthetic data set that is commonly used to evaluate machine learning models for causal inference. Covariates are based on a real-world randomized experiment in which some infants ($n = 747$) were targeted with child care and home visits. The resulting outcome is simulated based on these covariates ($x \in \mathbb{R}^{25}$) and the treatment, with the resulting treatment effect following a normal distribution. Although this dataset has recently been criticized [267], we include it due to it being a widely-used benchmark.

EDU. The Education data set [EDU; 218] measures the effect of providing a mother with adult education benefits on their children’s learning. The data is simulated based on covariates $x \in \mathbb{R}^{32}$ using two (non-linear) neural networks, with added Gaussian ($t = 0$) or exponential noise ($t = 1$). For both potential outcomes, the noise level depends on one of the covariates; more specifically, a single binary variable. Confounding is introduced through covariate-based propensities and removing well-balanced subjects.

News. The News data set [200] shows the effect of reading an article on either mobile or desktop (t) on the reader’s experience (y), based on the article’s content in word counts (x). The data is simulated using a topic model $z(x)$, which is used to define two centroids in the topic space: z_0^c (desktop) and z_1^c (mobile). Device assignment and reader experience are both simulated based on the similarity of the article’s topic $z(x)$ to the centroids z_0^c and z_1^c . The resulting data is very high-dimensional ($d = 3,477$).

Benchmark methodologies. We compare NOFLITE with several other machine learning models capable of learning distributions. First, we compare against Causal Multi-task Gaussian Processes³ [CMGP; 204]. Second, we compare against other methods relying on generative neural networks. The Causal Effect Variational Autoencoder [CEVAE; 217] uses a variational autoencoder to model latent confounders given noisy proxies. GANITE [216] uses generative adversarial networks to deal with selection bias and to obtain a probabilistic estimate of the treatment effect. Finally, we compare against Causal Collaborating Networks [CCN; 218].

5.5.2 Performance metrics

The main goal in this work is to predict accurate individual treatment effect distributions. Therefore, the main metric of interest is the model’s *loglikelihood*, which allows for a comprehensive assessment of each instance’s predicted distribution, by quantifying how likely the test data is given the predicted distributions. We estimate the loglikelihood as follows. For each instance i , we sample 500 samples from the model θ based on the covariates x_i . Then, for each instance, we fit a Gaussian kernel density estimator $\mathbf{kde}_i(x_i, \theta)$ using these samples and estimate the loglikelihood of the true treatment effect according to this kernel density estimator $\log p(\tau_i | \mathbf{kde}_i(x_i, \theta))$. The loglikelihood is averaged over all instances:

$$\text{LL} = \frac{1}{n} \sum_{i=1}^n \log p(\tau_i | \mathbf{kde}_i(x_i, \theta)). \quad (5.7)$$

The loglikelihood evaluates the predicted distributions globally. As argued in the motivation, statistics derived from the distribution can be used to facilitate decision-making. Therefore, to facilitate a more holistic assessment of performance, we present additional metrics that analyze specific properties of the predicted distribution. First, we use the square root of the *precision in estimation of heterogeneous effects* [$\sqrt{\text{PEHE}}$, see 205] to evaluate the accuracy of the expected value of the individual treatment effect:

$$\text{PEHE} = \frac{1}{n} \sum_{i=1}^n \left(\left(y_i^{(1)} - y_i^{(0)} \right) - \mathbb{E} \left(\hat{y}_i^{(1)} - \hat{y}_i^{(0)} \right) \right)^2 = \frac{1}{n} \sum_{i=1}^n (\tau_i - \hat{\tau}_i)^2. \quad (5.8)$$

Moreover, we evaluate the *empirical coverage* (Cov) of the estimated 90% confidence interval \hat{CI} , i.e., the probability that an observed sample of the

³For the high dimensional News data set, we take the first 100 principal components to avoid excessive training times. This number was tuned based on a validation set.

ITE lies within the estimated interval:

$$\text{Cov} = \frac{1}{n} \sum_{i=1}^n \mathbb{1}(\tau_i \in \hat{CI}_i). \quad (5.9)$$

For the 90% CI, the empirical coverage should ideally be 0.90 with small variance between iterations. For some data sets, we know the distribution that was used to generate the data. In these cases, we can additionally compare the predicted and true confidence intervals. We propose to evaluate their overlap using the *intersection-over-union* (IoU) comparing the predicted and ground truth confidence intervals:

$$\text{IoU} = \frac{CI \cap \widehat{CI}}{CI \cup \widehat{CI}}. \quad (5.10)$$

This metric is bounded between 0 and 1. The worst value, 0, indicates an empty intersection. The best value, 1, indicates an intersection equal to the union and, consequently, an estimated confidence interval that is equal to the ground truth.

Finally, we evaluate the quality of decisions made based on the predicted distributions, for a given utility distribution. The decision to treat or not is based on whether the expected utility is positive. For IHDP, we use $u(\tau) = (\tau - 4)^3$, given that the average treatment effect will be 4 on average [205]. For EDU, we use $u(\tau) = (\tau - 1)^3$, as the treatment effect was empirically observed to be approximately 1 on average. The predicted optimal treatment decision is compared with the theoretical optimal based on the ground truth distribution using the *accuracy* (Acc). For the News data, we cannot do this analysis, as the true distribution is not known. For all metrics and models, we use 500 samples (per instance) for evaluation.

5.5.3 Empirical results

We compare the different models for the IHDP, EDU, and News data sets in Table 5.2. In terms of loglikelihood, NOFLITE obtains the best performance out of all methods under consideration for each data set. These findings demonstrate NOFLITE’s ability to learn accurate individual treatment effect distributions from a variety of observational data sets and associated data generating processes. For the IHDP data set, NOFLITE slightly outperforms the next-best model, CMGP. Both methods use a Gaussian prior, which matches IHDP’s data generating process. On the EDU data set, the data generating process is more complex and requires more flexibility. Indeed, the Gaussian prior of CMGP results in relatively worse performance and the more complex models, such as NOFLITE and CCN, perform better. This

illustrates how the normalizing flows can allow for a more flexible model of posterior when required. Finally, the good performance on the News data set illustrates NOFLITE’s ability of learning from high dimensional data. As opposed to CMGP, which needs PCA preprocessing, it can also handle this high dimensional data out-of-the-box.

When we look at the metrics evaluating statistics derived from the treatment effect distribution, i.e. the PEHE and IoU, there is no clear winner overall: either CMGP, CCN or NOFLITE result in the best performance. Nevertheless, for all metrics, NOFLITE’s performance is competitive with the best performing methodology for each data set. This illustrates how optimizing the likelihood and learning distributions results in good performance for the metrics evaluating properties of this distribution. Nevertheless, if there is one particular metric of interest (e.g., PEHE), other objectives than NOFLITE’s loglikelihood might be preferable (e.g., the mean squared error). Moreover, we observe that NOFLITE obtains relatively high accuracy for both the IHDP and EDU data sets. This illustrates that pairing the NOFLITE’s predicted distributions with a utility function enables qualitative decision-making.

Finally, we illustrate how NOFLITE can be used for practical applications. We do this by training our model on an iteration of the News data set and showing the model’s output for a few selected test instances, see Figure 5.3. Figures 5.3a to 5.3c show the ITE distribution and related statistics based on samples from the learned model. This not only allows for visualizing the estimated distribution $p(\tau_i)$ and its associated expected value $\mathbb{E}(\tau)$, but also for assessing the uncertainty using the 90% confidence interval or a boxplot. Moreover, due to the News data set being semi-synthetic, we can compare the ground truth τ_{observed} with the estimated distribution. Finally, the model can be used to consider the treatment decision for an instance by, e.g., looking at the probability of its treatment effect being strictly positive. Figure 5.3d shows the heterogeneity in distributions of different instances, both in terms of expected value and shape.

5.6 Conclusion

Estimating an instance’s individual treatment effect distribution is an essential requirement for personalized decision-making. To this aim, we presented NOFLITE, a flexible neural method for estimating treatment effect distributions that directly optimizes the metric of interest for this task: the model’s likelihood. By leveraging normalizing flows, the model is not constrained to any particular parametric distribution, but can instead trade off a simple normal distribution with a more complex posterior, depending on the data. Experiments on a variety of data sets demonstrated NOFLITE’s good perfor-

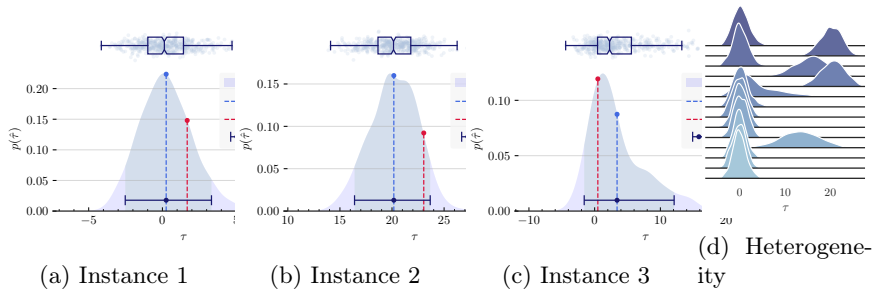


Figure 5.3: **NOFLITE illustration.** We visualize NOFLITE’s output for selected test instances for one iteration of the News data set. *Individual treatment effect distributions:* Figures (a-c) visualize one particular instance’s treatment effect distribution and related statistics based on samples from the model. *Distribution heterogeneity:* Figure (d) compares the predicted individual treatment effect distributions of several instances.

mance in practice and underlined its excellent representational capacity, as illustrated by its ability to obtain high likelihoods for a variety of data set sizes, dimensionalities, and data generating processes.

Future work in normalizing flows could benefit our method, as novel flow transformations will be compatible with our method and help address potential limitations of our work. First, depending on the flow transformation that is used, training or inference can be slow in normalizing flows. Second, NOFLITE introduces a variety of hyperparameters. Therefore, novel flow transformation with less hyperparameters might benefit adoption of our method in practice. Related to these points, it would be interesting to incorporate advances in other types of generative models, such as diffusion models [see e.g. 268], in future work.

Although estimating individual treatment effect distributions can be very valuable, it constitutes a challenging problem – especially when working with high-dimensional, observational data. For many applications, the available data may be limited. Although NOFLITE can be used without flow transformations, its flexibility comes from the normalizing flows, which require data to be trained successfully. Although NOFLITE’s hyperparameters can be tuned by looking at the fit on a validation set, validating causal inference models itself is a challenging problem [269]. Therefore, we consider it a promising area for future work to analyze NOFLITE from a theoretical perspective and come up with performance guarantees based on statistical learning theory.

Additionally, our method relies on the standard ignorability assumptions in causal inference. Judging the feasibility of these assumptions is impossible based on data alone and requires the judgment of domain experts. There is a growing body of work looking at learning treatment effects under violations of these assumptions (e.g., under hidden confounding, see [270], [271]). Another interesting direction for future work is to analyze performance of our method in settings where the ignorability assumptions are violated, and to extend our methodology to account for these violations.

Table 5.2: **Empirical results.** We compare NOFLITE against a variety of existing methods for three data sets: (a) IHDP, (b) EDU, and (c) News. For each metric, arrows indicate whether a lower (\downarrow) or higher (\uparrow) value is better; the ideal coverage is 90%. We show the best result in **bold**. We highlight the loglikelihood (LL) in gray to emphasize that this is our main metric of interest, as it evaluates the quality of the predicted distributions. For all data sets, GANITE achieves a loglikelihood of less than -10 , indicated by a dash ($-$).

	LL (\uparrow)	\sqrt{PEHE} (\downarrow)	IoU (\uparrow)	Cov (90%)	Acc (\uparrow)
CMGP	$-1.95_{\pm 0.07}$	$0.73_{\pm 0.09}$	$0.77_{\pm 0.01}$	$0.88_{\pm 0.00}$	$0.90_{\pm 0.01}$
BART	$-2.16_{\pm 0.05}$	$2.22_{\pm 0.36}$	$0.63_{\pm 0.02}$	$0.83_{\pm 0.01}$	$0.84_{\pm 0.01}$
GANITE	—	$6.32_{\pm 0.89}$	$0.01_{\pm 0.00}$	$0.01_{\pm 0.00}$	$0.55_{\pm 0.02}$
CEVAE	$-2.97_{\pm 0.09}$	$5.71_{\pm 0.89}$	$0.25_{\pm 0.01}$	$0.98_{\pm 0.00}$	$0.57_{\pm 0.01}$
CCN	$-2.16_{\pm 0.08}$	$1.46_{\pm 0.17}$	$0.66_{\pm 0.02}$	$0.84_{\pm 0.00}$	$0.86_{\pm 0.01}$
NOFLITE	$-1.90_{\pm 0.01}$	$1.09_{\pm 0.20}$	$0.75_{\pm 0.00}$	$0.88_{\pm 0.00}$	$0.90_{\pm 0.01}$

(a) IHDP ($n = 747$; $d = 25$)

	LL (\uparrow)	\sqrt{PEHE} (\downarrow)	IoU (\uparrow)	Cov (90%)	Acc (\uparrow)
CMGP	$-1.74_{\pm 0.01}$	$0.22_{\pm 0.01}$	$0.56_{\pm 0.00}$	$0.91_{\pm 0.00}$	$0.70_{\pm 0.00}$
BART	$-1.71_{\pm 0.01}$	$0.53_{\pm 0.01}$	$0.56_{\pm 0.00}$	$0.89_{\pm 0.00}$	$0.69_{\pm 0.00}$
GANITE	—	$1.26_{\pm 0.08}$	$0.37_{\pm 0.03}$	$0.46_{\pm 0.03}$	$0.72_{\pm 0.01}$
CEVAE	$-2.67_{\pm 0.03}$	$2.20_{\pm 0.25}$	$0.25_{\pm 0.00}$	$1.00_{\pm 0.00}$	$0.49_{\pm 0.01}$
CCN	$-1.65_{\pm 0.01}$	$0.31_{\pm 0.01}$	$0.64_{\pm 0.01}$	$0.87_{\pm 0.00}$	$0.76_{\pm 0.01}$
NOFLITE	$-1.62_{\pm 0.01}$	$0.26_{\pm 0.01}$	$0.64_{\pm 0.01}$	$0.89_{\pm 0.00}$	$0.76_{\pm 0.01}$

(b) EDU ($n = 8,627$; $d = 32$)

	LL (\uparrow)	\sqrt{PEHE} (\downarrow)	Cov (90%)
CMGP	$-2.29_{\pm 0.03}$	$2.21_{\pm 0.05}$	$0.95_{\pm 0.00}$
BART	$-2.43_{\pm 0.04}$	$2.71_{\pm 0.12}$	$0.97_{\pm 0.00}$
GANITE	—	$18.91_{\pm 11.29}$	$0.01_{\pm 0.00}$
CEVAE	$-2.83_{\pm 0.04}$	$3.74_{\pm 0.18}$	$0.97_{\pm 0.00}$
CCN	$-2.25_{\pm 0.03}$	$2.23_{\pm 0.04}$	$0.84_{\pm 0.00}$
NOFLITE	$-2.15_{\pm 0.02}$	$2.18_{\pm 0.05}$	$0.93_{\pm 0.00}$

(c) News ($n = 5,000$; $d = 3,477$)

6

ACCOUNTING FOR INFORMATIVE SAMPLING WHEN LEARNING TO FORECAST TREATMENT OUTCOMES OVER TIME

Machine learning (ML) holds great potential for accurately forecasting treatment outcomes over time, which could ultimately enable the adoption of more individualized treatment strategies in many practical applications. However, a significant challenge that has been largely overlooked by the ML literature on this topic is the presence of *informative sampling* in observational data. When instances are observed irregularly over time, sampling times are typically not random, but rather informative—depending on the instance’s characteristics, past outcomes, and administered treatments. In this work, we formalize informative sampling as a covariate shift problem and show that it can prohibit accurate estimation of treatment outcomes if not properly accounted for. To overcome this challenge, we present a general framework for learning treatment outcomes in the presence of informative sampling using inverse intensity-weighting, and propose a novel method, TESAR-CDE, that instantiates this frame-

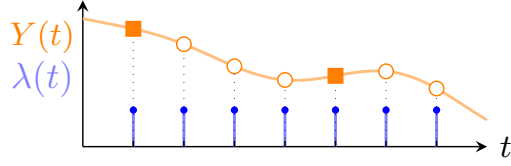
work using Neural CDEs. Using a simulation environment based on a clinical use case, we demonstrate the effectiveness of our approach in learning under informative sampling.

6.1 Introduction

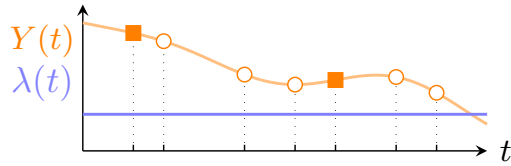
Due to its importance in applications ranging from economics to health-care and marketing, the problem of estimating personalized causal effects of actions – e.g., treatments, interventions, or policies – has received wide attention in the recent machine learning (ML) literature [209]. Effectively using real data for estimating such effects requires dealing with unique challenges arising from its observational nature. Therefore, the recent ML literature on treatment effect estimation has paid great attention to solving methodological issues arising due to treatment assignment biases in static [201] and longitudinal settings [272]. This paper focuses on another challenge that has been largely overlooked by the ML literature on treatment effect estimation, despite its relevance and prevalence in practice: the problem of *informative sampling*, sometimes also called *informed presence bias* [273]. That is, in observational data, the timing at which an observation was made is often not random, but rather indicative of some underlying information relevant to the estimation problem of interest.

In electronic health records, for example, patients are typically not recorded *randomly* over time, but *informatively* [274]: observations are only recorded at irregular visits to a health care provider, with visit times typically depending on the patient’s past and present characteristics, evolving health state, and administered treatments. The resulting sampling mechanism is inherently intertwined with the patient’s observed outcomes and treatments, with more check-ups being scheduled for patients in critical condition or to follow up after a treatment. Throughout this work, we will refer to examples from health care due to their societal relevance and intuitive appeal, but the problem of informative sampling appears in a wide variety of other domains, such as policy design [274], epidemiology [275], economics [276], or maintenance [10].

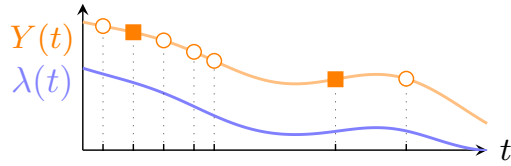
Informative sampling poses an important challenge, as it can bias estimates of causal effects when not accounted for [274], [277], [278]. Intuitively, informative sampling leads to relatively more measurements of abnormal values and fewer measurements of normal values and, therefore, selection bias in the data [279], [280]. Standard statistical methods can estimate causal effects in the presence of informative sampling, given a well-specified model of the sampling mechanism [281]. However, existing approaches for modeling the sampling mechanism from the (bio)statistics literature assume a



(a) Regular sampling



(b) Sampling completely at random (SCAR)



(c) Sampling at random (SAR)

Figure 6.1: **Problem illustration: sampling mechanisms.** We show an instance’s latent trajectory ($Y(t)$) and sampling intensity ($\lambda(t)$) over time t , along with administered treatments (■) and observations (○) resulting from different sampling mechanisms. **(a) Regular.** Samples are obtained at regular intervals over time. **(b) SCAR.** Samples are irregular, drawn at completely random intervals over time. **(c) SAR.** Sampling times are *irregular, but not completely random*: e.g., there might be more samples when the outcome is large. We refer to the dependence of the sampling intensity on an instance’s covariates, treatments, and/or outcomes as *informative sampling*. Whereas existing work in the ML literature assumes regular sampling or SCAR, this work is, to the best of our knowledge, the first to consider learning to forecast treatment outcomes given SAR.

certain parametric form or latent variable(s), which might not match the actual data-generating process. Moreover, Farzanfar, Abumuamar, Kim, *et al.* [282]’s survey on longitudinal healthcare research finds that these methods are rarely used in practice, leaving potential bias largely unaddressed. Therefore, this work examines the use of flexible ML methods for this task and investigates the unique methodological challenges arising therein.

Related work.¹ Since the initial seminal ML work on heterogeneous treatment effect estimation considering binary treatments and static data [200], [201], this literature has grown rapidly both by making methodological refinements in the original setting [203], [283] and by considering new settings, such as continuous treatments [272] or survival outcomes [284]. Recent extensions have specifically explored using ML methods for estimating treatment effects *over time*, such as RNNs [272], [285]–[287], transformers [288], and Neural ODEs [289], [290] or Neural CDEs [291].

Most existing ML work on causal inference in a temporal setting has, to the best of our knowledge, implicitly relied on strict assumptions regarding the sampling mechanism. The majority assumes regular and uninformative sampling times (Figure 6.1a). Only very recent work relying on neural differential equations to model the effects of treatment in continuous time [289]–[291] *allows* for observations to be irregular (Figure 6.1b), but *does not consider or account for potential bias* resulting from sampling times being informative rather than completely random, which is the focus of this work (Figure 6.1c). This stands in stark contrast to the close attention paid in the treatment effect estimation literature to other sources of *covariate shift* arising in observational data, e.g., due to static treatment assignment [200], treatment assignment over time [272], censoring [284] or competing events [292]. In this spirit, we find it important to study when and how the informativeness of sampling acts as an additional source of covariate shift in this setting.

Contributions. Despite the rapid recent expansion of the ML literature on estimating treatment effects, we believe that there is still a fundamental lack of understanding, or even formalization, of the challenges arising due to one of the most fundamental features of observational data: *sampling, the act of observation itself, can be inherently informative*. Therefore, we focus on understanding and analyzing the challenges that arise from informative sampling and propose strategies to alleviate bias arising in this context. In doing so, we make three contributions: **(1)** We formalize the problem of forecasting treatment outcomes under informative sampling as a machine learning problem and characterize the key challenges arising therein as a consequence

¹We discuss the related work more extensively in Appendix D.1.

of covariate shift induced by informative sampling. **(2)** We present a general strategy for tackling this challenge and propose a novel method for learning under informative sampling, TESAR-CDE, that instantiates this framework using Neural CDEs. **(3)** We design a simulation environment based on a clinical use case to study the effect and different drivers of informative sampling and use it to empirically demonstrate that our proposed method is able to correct for the resulting bias that existing methods can suffer from.

6.2 Problem Formalization: Data Structure and Informative Sampling Mechanism

This section describes and formalizes the problem of forecasting treatment outcomes in the presence of informative sampling. We investigate the assumptions required for and challenges inherent to tackling this problem in Section 6.3. We build on the exposition in Lin, Scharfstein, and Rosenheck [274], who study longitudinal *outcome prediction* in the presence of informative sampling but do not explicitly consider estimating treatment effects, and build on ideas from Seedat, Imrie, Bellot, *et al.* [291] and Lok [293] who study forecasting treatment outcomes in continuous time but do not consider informative sampling.

6.2.1 Problem structure: Complete versus observed data

Underlying complete data structure. We consider data collected over a period $[0, T]$ in which instances are characterized by a d -dimensional covariate path $X : [0, T] \rightarrow \mathbb{R}^d$ and a treatment path $A : [0, T] \rightarrow \{0, 1\}$ – which jumps to 1 only at time steps t when treatment is administered² – both of which possibly modulate an outcome process of interest $Y : [0, T] \rightarrow \mathbb{R}$. While we only observe the outcome Y associated with the treatment path A that was actually administered (sometimes also referred to as the *factual outcome*), we assume that any instance is characterised by a possibly infinite number of *potential outcomes* $Y_a : [0, T] \rightarrow \mathbb{R}$ associated with other feasible treatment paths a .

Observed data structure. Paths X , A and Y are only sampled (observed) at possibly irregular and discrete time-points, such as scheduled check-ups

²In this exposition, we assume that $A(t) = 1$ only at single time-steps where treatment is administered; for treatments that are administered over a time-period $[t_1, t_2]$ one could instead define a counting process that jumps whenever treatment status is *changed* as in Seedat, Imrie, Bellot, *et al.* [291] and Lok [293]. Further, as noted in Seedat, Imrie, Bellot, *et al.* [291], this definition can be generalized to multiple treatments by assuming A to be a multivariate process.

or unscheduled appointments. Therefore, we additionally define a counting process $N : [0, T] \rightarrow \mathbb{N}_0$ recording the number of observations made by time t . This process jumps whenever a new observation is sampled³, so that $dN(t) = 1$ if an instance is sampled at time t and $dN(t) = 0$ otherwise, where $dN(t) = N(t) - \lim_{s \uparrow t} N(s)$. As in Lin, Scharfstein, and Rosenheck [274], for a variable $V(t)$, let $V^o(t) = V(\max s : 0 \leq s \leq t, dN(s) = 1)$ denote its most recent observation by time t , $\bar{V}^o(t) = \{V^o(s) : 0 \leq s \leq t\}$ its observed history by time t and $\bar{V}(t) = \{V(s) : 0 \leq s \leq t\}$ its full (yet possibly not observed) history by time t . Further, let $\bar{V}^o(t^-)$ and $\bar{V}(t^-)$ denote the same histories where the upper limit does *not* include t . Then, for a study following n instances until time T , we observe a dataset $\mathcal{D} = \{\mathcal{O}_i\}_{i=1}^n$ consisting of n i.i.d. copies of $\mathcal{O} = \mathcal{F}^o(T)$ where $\mathcal{F}^o(T) = (\bar{X}^o(T), \bar{N}(T), \bar{A}^o(T), \bar{Y}^o(T))$.

6.2.2 Distinguishing between different sampling patterns

In order to define what distinguishes the *informativeness* of different sampling patterns, we first need to introduce a conditional *intensity* $\lambda(t)$ which governs the observation process $N(t)$. In its most general form, adopting the notation of Lin, Scharfstein, and Rosenheck [274], this can be defined through

$$\mathbb{P}(dN(t) = 1 | \bar{X}(T), \bar{N}(T), \bar{A}(T), \bar{Y}(T)) = \lambda(t)dt, \quad (6.1)$$

For notational convenience, we omit conditioning in $\lambda(t)$.

Using this definition, we can differentiate between different sampling mechanisms [294], giving rise to a classification similar to missingness mechanisms in static data [295]. This categorization is based on the causal role of the instance history in relation to the observation intensity (Figure 6.1 shows a graphical overview):

- **Regular sampling** (Figure 6.1a). Instances are observed at K regular (pre-determined) timesteps $\mathcal{T} = \{t_1, \dots, t_K\}$, so that $\lambda(t)dt = 1$ if $t \in \mathcal{T}$ else $\lambda(t)dt = 0$. Most existing related work [e.g. 272], [285], [288] implicitly relies on this assumption.
- **Sampling completely at random (SCAR; Figure 6.1b)**. Instances are observed at completely random time-steps, with the intensity independent of all variables: $\lambda(t)dt = \mathbb{P}(dN(t) = 1 | \bar{X}(T), \bar{N}(T), \bar{A}(T), \bar{Y}(T)) = \mathbb{P}(dN(t) = 1)$. Recent work on treatment effect estimation from irregularly sampled data using neural differential equations [289]–[291] is explicitly only equipped to handle this scenario.

³For ease of exposition, we assume that whenever an instance is observed at t , we record all of $X(t)$, $A(t)$ and $Y(t)$. Nevertheless, it would be possible to relax this by instead introducing separate counting processes for each variable or component thereof.

- **Sampling at random (SAR;** Figure 6.1c). Being observed at time t is independent of the (up until then unknown) outcome at time t given the observed history up to time t :

$$\begin{aligned}\lambda(t)dt &= \mathbb{P}(dN(t)=1|\bar{X}(T), \bar{N}(T), \bar{A}(T), \bar{Y}(T)) \\ &= \mathbb{P}(dN(t)=1|\bar{X}^o(t), \bar{N}(t^-), \bar{A}^o(t^-), \bar{Y}^o(t^-))\end{aligned}$$

This work investigates the challenges of learning given SAR, which is considerably weaker than SCAR: it allows, for example, for patients to have more frequent visits due to past outcomes, administered treatments, or worsening symptoms (provided that these are recorded in X).

We also consider a stricter variant, which we will refer to as the *strong SAR assumption*: here we assume that $\lambda(t)dt = \mathbb{P}(dN(t) = 1|\bar{X}(T), \bar{N}(T), \bar{A}(T), \bar{Y}(T)) = \mathbb{P}(dN(t) = 1|\bar{X}^o(t^-), \bar{N}(t^-), \bar{A}^o(t^-), \bar{Y}^o(t^-))$. This differs from the more general (weaker) SAR assumption above in that observing a patient at time t , i.e. $dN(t)$, cannot depend on the covariates $X^o(t)$ to be observed *at* time t . As we discuss in the next sections, the weaker SAR assumption already allows identification of treatment effects in our setting, while the strong SAR assumption can greatly simplify estimation of intensities.

- **Sampling not at random (SNAR).** The most general scenario is one where observing is *not* independent of future outcomes conditional on the observed history – i.e. $\lambda(t)dt \neq \mathbb{P}(dN(t)=1|\bar{X}^o(t), \bar{N}(t^-), \bar{A}^o(t^-), \bar{Y}^o(t^-))$. This would be the case, e.g., if patients chose to visit due to worsening symptoms that are *not recorded* in X and hence act as a latent cause of the intensity *and* outcome. In this scenario, outcomes cannot be consistently forecast unless further assumptions regarding the sampling or outcome-generating mechanism are made. Therefore, we rely on sampling at random in this work.

6.3 Forecasting Treatment Outcomes Under Informative Sampling: Goals, Assumptions and Inherent Challenges

6.3.1 Goal: Forecasting treatment outcomes

We aim to estimate **conditional average potential outcomes (CAPOs)** $\mu_{a,t}(\tau)$ at a future time $t + \tau$, $\tau \in (0, \tau_{\max}]$ (with $\tau_{\max} \leq T - t$):

$$\mu_{a,t}(\tau) = \mathbb{E}[Y_a(t+\tau)|\bar{X}^o(t), \bar{N}(t), \bar{A}^o(t), \bar{Y}^o(t)] \quad (6.2)$$

i.e., the instance's expected outcome under treatment plan a conditional on its full observed history $\mathcal{H}^o(t) = \{\bar{X}^o(t), \bar{N}(t), \bar{A}^o(t), \bar{Y}^o(t)\}$ up to time t . We only consider viable treatment plans a subject to $a(t^*) = A(t^*)$ for $t^* \leq t$ – i.e., those that do not modify the past, factual treatment history prior to the current time t . Such an estimate could be used in practice to decide between competing treatment plans based on expected outcome under either choice. In line with Gische, West, and Voelkle [296], we purposefully use the term *forecasting* instead of *predicting* throughout to signify that we wish to give *causal interpretation* to the modeled effects of treatments. This is because, analogously to the standard static setting, unless we make further identifying assumptions, we can in general not assume that predictions based on expectations of the form $\mathbb{E}[Y^o(t + \tau) | A = a, \mathcal{H}^o(t)]$ are equal to forecasts based on expectations of the form $\mathbb{E}[Y_a(t + \tau) | \mathcal{H}^o(t)]$.

6.3.2 Identifying assumptions

To ensure *identification* of causal claims from observational data, we need to introduce additional assumptions. First, we make assumptions that correspond to adaptations of the standard *ignorability* assumptions [181] from the standard static setting to our setting. To do so, we define *treatment propensities* for single time-steps $\pi(a(t)) = \mathbb{P}(A(t) = a(t) | \bar{X}(T), \bar{Y}(T), \bar{A}(T), \bar{N}(T))$ and entire trajectories $\pi_t(a) = \mathbb{P}(A = a | \mathcal{H}^o(t))$ given history until time t .

Assumption 7. Consistency. *Given an observed treatment path A , we observe the outcome corresponding to the associated potential outcome: $Y^o(t) = Y_A(\max s : 0 \leq s \leq t, dN(t) = 1)$.*

Assumption 8. Unconfoundedness. *The treatment propensity $\pi(a, t)$ does not depend on future outcomes or unobserved information:*

$$\begin{aligned} \pi(a(t)) &= \mathbb{P}(A(t) = a(t) | \bar{X}(T), \bar{Y}(T), \bar{A}(T), \bar{N}(T)) \\ &= \mathbb{P}(A(t) = a(t) | \bar{X}^o(t), \bar{N}(t), \bar{A}^o(t^-), \bar{Y}^o(t^-)) \end{aligned}$$

Assumption 9. Overlap (Positivity for treatment). $0 < \mathbb{P}(A = a | \mathcal{H}^o(t)) < 1$, for all admissible treatment paths a and histories $\mathcal{H}^o(t)$ of interest for forecasting.

These assumptions are required regardless of the sampling mechanism. For regular sampling, these reduce to the sequential ignorability assumptions made in earlier work [e.g., 272], [285], [288].

On top of these ignorability assumptions, estimating causal effects under informative sampling requires making additional assumptions regarding the sampling mechanism [277]. In contrast to existing work which *implicitly*

assumed regular observations [e.g., 272], [285] or sampling completely at random [e.g., 291], we *explicitly* state our assumed observation process. Specifically, we rely on the weaker, previously introduced *sampling at random* (SAR) assumption:

Assumption 10. *Sampling at random (SAR).* *The sampling intensity process does not depend on unobserved or future information, i.e., $\lambda(t) = \mathbb{P}(dN(t) = 1 | \bar{X}^o(t), \bar{N}(t^-), \bar{A}^o(t^-), \bar{Y}^o(t^-))$.*

Analogous to assumptions on treatment overlap, we assume the probability of observing at any point in time is bounded away from zero, for any history of interest for forecasting:

Assumption 11. *Positivity of observation.* $\mathbb{P}(dN(t+\tau) = 1 | \mathcal{H}^o(t)) > 0$ for any $\tau \in (0, T - t]$ and history $\mathcal{H}^o(t)$ of interest.

Finally, we assume that all treatment events are observed. In most applications, this is a natural assumption (e.g., if treatments are administered at a hospital). It is generally not possible to estimate treatment effects from observed outcomes without knowing which treatments were administered, unless further assumptions are made [297].

Assumption 12. *Observed treatments.* *All treatments are observed, i.e. $\mathbb{P}(A(t) = 1 | dN(t) = 0) = 0$.*

The identifying assumptions discussed above can equivalently be expressed as a generative model, determining the temporal ordering of realizations of the different observed variables⁴. In particular, at each time t , the visit decision $dN(t)$ is realized first, which can depend on observed histories $\mathcal{H}^o(t^-)$ and covariates to be observed $X^o(t)$ (SAR) or on $\mathcal{H}^o(t^-)$ only (strong SAR); the former implies a setting where e.g. patients present themselves for an appointment due to worsening symptoms while the latter allows only scheduling of future appointments due to symptoms already observed earlier. If $dN(t) = 1$, then covariates $X^o(t)$ are first observed, treatment $A(t)$ is then determined based only on observed information ($\mathcal{H}^o(t^-)$, $N(t)$ & $X^o(t)$) and, finally, the outcome is realized and observed as $Y^o(t)$.

⁴In principle, other generative models could be assumed as long as sufficient exclusion restrictions between observation-/treatment-generating processes and outcome-generating processes are made. For example, the visit process can depend on future treatments if such treatments are pre-scheduled.

6.3.3 What makes learning CAPOs from observational data challenging?

If we had access to the complete data structure with all potential outcomes $Y_a(t)$, learning an estimate $\hat{\mu}_{a,t}(\tau; \mathcal{H}^o(t))$ for the CAPOs with fixed a would be a standard ML problem: we would search a hypothesis function in some hypothesis class \mathcal{F} that minimizes the expected (oracle) risk R^* , i.e.,

$$\hat{\mu}_{a,t}(\tau; \mathcal{H}^o(t)) \in \arg \min_{f_{a,\tau} \in \mathcal{F}} R^*(f_{a,\tau}) \quad (6.3)$$

where, for some loss function ℓ , and using the shorthands $t' = t + \tau$ and $h_t = \mathcal{H}^o(t)$

$$R^*(f_{a,\tau}) = \mathbb{E} \left[\int_0^T \int_t^{\tau_{\max}} \ell(Y_a(t'), f_{a,\tau}(h_t)) d\tau dt \right] = \int_0^T \int_0^{\tau_{\max}} \int \ell(y_a(t'), f_{a,\tau}(h_t)) dP(y_a(t')|h_t) dP(h_t) d\tau dt$$

However, as previously discussed, in reality we only have access to observational data in which patients are (i) incompletely, irregularly, and informatively observed and (ii) characterized by only a single *factual* outcome corresponding to the treatment actually received. If we were to learn a standard ML predictor from this observed data, we would instead be optimizing the observational risk $R^{obs}(h_{a,\tau}) =$

$$\mathbb{E} \left[\int_0^T \int_0^{\tau_{\max}} \mathbb{1}\{A=a\} dN(t') \ell(Y_a(t'), f_{a,\tau}(h_t)) d\tau dt \right] = \int_0^T \int_t^{\tau_{\max}} \int \ell(y_a(t'), f_{a,\tau}(h_t)) \pi_t(a) \lambda_t(t') dP(y_a(t')|h_t) dP(h_t) d\tau dt$$

Thus, unless the τ -step ahead intensity $\lambda_t(t')$, defined through $\lambda_t(t')d\tau = \mathbb{P}(dN(t+\tau)=1|\mathcal{H}^o(t) \cup \bar{A}(t+\tau^-))$, and treatment propensity $\pi_t(a)$ are constant across patient histories, the minimizers of R^* and R^{obs} will in general be different. Intuitively, this is because the distribution of patient characteristics in the observed data can differ from the distribution in the underlying unobserved complete distribution, both due to informative sampling and treatment selection. Thus, the challenge we are facing here is one of *covariate shift* between the training data and hypothetical test data.

Covariate shift and its potential remedies have been studied in much depth in the recent ML literature (see e.g. Redko, Morvant, Habrard, *et al.* [298])

and Farahani, Voghoei, Rasheed, *et al.* [299] for recent overviews). Here, we explore the use of one of the oldest and most well-established solutions: *importance weighting* [300]. That is, as further discussed in the next section, we propose to minimize a weighted observational risk

$$R^w(f_{a,\tau}) = \mathbb{E} \left[\int_0^T \int_0^{\tau_{\max}} w_{a,\tau} \ell(f_{a,\tau}) d\tau dt \right] \quad (6.4)$$

with $\ell(f_{a,\tau}) = \mathbb{1}\{A = a\} dN(t') \ell(Y_a(t'), f_{a,\tau}(h_t))$. For oracle importance weights $w_{a,\tau}^* = \frac{1}{\pi(a)\lambda_t(t')}$ it is easy to see that $R^{obs,w^*}(f_{a,\tau}) = R^*(f_{a,\tau})$.

6.4 Learning to Forecast Treatment Outcomes Under Informative Sampling

This section presents a methodology for learning to forecast treatment outcomes under informative sampling. Section 6.4.1 presents a general framework that is compatible with any ML algorithm capable of predicting outcomes over time. Section 6.4.2 instantiates this framework using Neural CDEs.

6.4.1 Learning To Forecast Using Inverse Intensity Weights

The analysis presented in Section 6.3.3 allows straightforward construction of a framework for learning to forecast treatment outcomes from informatively sampled (SAR) data. Given an ML algorithm \mathcal{A} that can issue continuous-time predictions using irregularly sampled data, one simply needs to fit \mathcal{A} on the observed data while providing appropriate importance weights w . As the true weights will generally be unknown in practice, one might have to use \mathcal{A} to also learn (i) observation intensities and (ii) treatment propensities to gain access to estimates of the true importance weights. As we discuss for a specific example in Section 6.4.2, one could learn such weights either in a pre-processing step or in an end-to-end fashion.

When learning intensity weights, it becomes important whether one makes the general SAR or the strong SAR assumption: under the strong SAR assumption, learning $\lambda_t(t')$ comes down to the easier task of estimating $\mathbb{P}(dN(t') = 1 | \mathcal{H}^o(t))$ directly, where $t' = t + \tau$. Under the more general SAR assumption, one needs to model $dN(t')$ and $X^o(t')$ jointly as a *marked* point process to learn the distributions $P(dN(t'), X^o(t') | \mathcal{H}^o(t)) = P(dN(t') | X^o(t'), \mathcal{H}^o(t)) P(X^o(t') | \mathcal{H}^o(t))$, where $P(X^o(t') | \mathcal{H}^o(t))$ could be a high-dimensional continuous density. In the remainder, we therefore restrict ourselves to the strong SAR setting – allowing us to highlight the

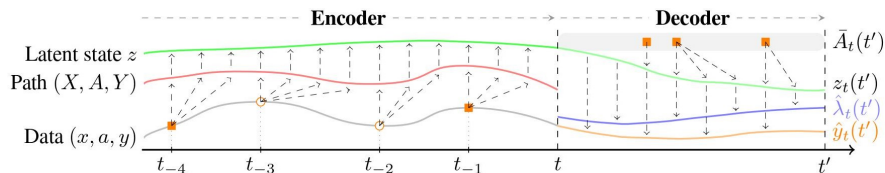


Figure 6.2: **TESAR-CDE: Adapting TE-CDE for learning given SAR.** The history of observations (○) and treatments (■) up to time t is first encoded as a continuous latent path $z(t)$. Based on a future treatment plan $\bar{A}_t(t')$, the decoder then forecasts a future latent path $z_t(t')$, with $t' = t + \tau$. In contrast to TE-CDE, TESAR-CDE (1) uses the latent path $z_t(t')$ to forecast both the outcome $\hat{y}_t(t')$ and intensity $\hat{\lambda}_t(t')$, and (2) uses the intensity to weight the outcome loss using $\mathcal{L}^{\text{WMSE}}$.

challenges arising when learning under some form of informative sampling. It would be an interesting next step to incorporate some of the recent work on Neural Temporal Point Processes [see e.g. 301] to enable learning under more complex dependencies.

In the following, we discuss two possible implementations of this framework using Neural CDEs by extending the methodology for learning continuous time treatment effects of Seedat, Imrie, Bellot, *et al.* [291] (which originally did not correct for informative sampling). Nevertheless, the approach discussed above is more general and could be applied to any ML model that can predict $Y(t)$.

6.4.2 TESAR-CDE: Forecasting with Intensity-weighted Neural CDEs

This section presents TESAR-CDE, a specific implementation of the framework discussed above by extending TE-CDE [291] to account for informative sampling (SAR), see Figure 6.2 for a graphical overview. In the remainder of this work, we focus on the special case where complete treatment plans A are *randomly* assigned and *fixed* at time $t = 0$; such a situation commonly arises in practice, e.g., in a clinical trial with a dynamic observation plan [274], [302]. This allows to *single out* the challenges arising solely due to the presence of informative observations. Moreover, this allows us to highlight that the forecasting problem remains challenging even in the absence of all treatment selection bias (the main challenge addressed in related work). Nevertheless, if required, any existing method equipped to deal with outcome-treatment confounding – e.g., using importance weighting [285] or

adversarial training [291] – could simply be combined with the inverse intensity weighting approaches we discuss and test below.

Background: TE-CDE

Treatment Effect Neural Controlled Differential Equation [TE-CDE; 291] is a recently proposed model for forecasting treatment effects from irregularly sampled data. TE-CDE views observations as samples from an underlying continuous-time process and uses Neural CDEs [303] to learn this latent trajectory. First, an encoder learns a latent path $z(t)$ as the solution of a CDE:

$$z(t_0) = g(X(t_0), A(t_0), Y(t_0)),$$

$$z(t) = z(t_0) + \int_{t_0}^t f_\theta(z(s)) \frac{d(X(s), A(s), Y(s))}{ds} ds$$

with g and f_θ neural networks. This is achieved by solving the above initial value problem (IVP), $\forall s \in [t_0, t]$:

$$z(t) = \text{ODESolve}(f_\theta, z(t_0), \bar{X}(t), \bar{A}(t), \bar{Y}(t))$$

using a numerical ODE solver [303]. The decoder forecasts the future latent path $z_t(t')$ by solving a second IVP given the future treatment plan $\bar{A}_t(t')$:

$$z_t(t') = \text{ODESolve}(f_\phi, z(t), \bar{A}_t(t')),$$

with decoder network f_ϕ and $t' = t + \tau$. A final network f_ψ maps the latent path $z_t(t')$ to the outcome $\hat{y}_t(t') = f_\psi(z_t(t'))$. Figure 6.3a shows the complete architecture.

The entire model (g , f_θ , f_ϕ and f_ψ) is trained by optimizing the mean squared error (MSE) of the predicted outcome:

$$\mathcal{L}_i^{\text{MSE}} = \int_0^T \int_0^{\tau_{\max}} dN_i(t') (y_i(t') - \hat{y}_{i,t}(t'))^2 dt d\tau.$$

This way, the mean squared error is calculated using the observed outcomes in the considered forecasting horizon $(0, \tau_{\max}]$, for each timestep $t \in [0, T]$. To account for bias resulting from time-dependent confounding, TE-CDE also uses domain adversarial training to learn a treatment-invariant representation. However, as discussed above, we focus on unconfounded settings in the remainder of this work and therefore do not include this, though it is straightforward to add it in settings where needed.

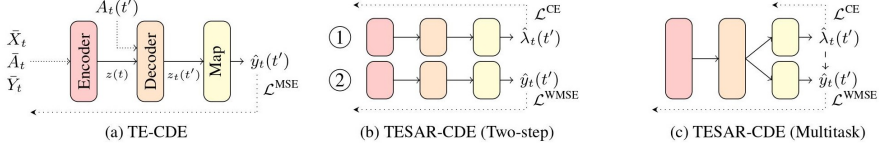


Figure 6.3: **Comparing TESAR-CDE to TE-CDE.** We show TE-CDE and our proposed alternative, TESAR-CDE, in its two-step and multitask configuration. Arrows indicate the input (\longrightarrow), forward pass (\longrightarrow) and back-propagation (\longleftarrow). The multitask model uses the intensity loss only to train the intensity map, but not the shared encoder or decoder. The dashed arrow (\dashrightarrow) indicates that the intensities are used as weights λ_t^{-1} in $\mathcal{L}^{\text{WMSE}}$, but not backpropagated as part of this loss.

TESAR-CDE: Learning to forecast with an inverse-intensity weighted loss

In this section, we instantiate our previously proposed framework using Neural CDEs, resulting in TESAR-CDE, Treatment Effects given Sampling At Random using Neural CDEs. Essentially, we extend TE-CDE for learning under informative sampling. Given (estimated) intensities $\hat{\lambda}_{i,t}(t')$, adapting TE-CDE’s outcome loss to adjust for informative observations is straightforward: the inverse of these estimated intensities can be used as importance weights to train TE-CDE for outcome prediction using a *weighted* MSE:

$$\mathcal{L}_i^{\text{WMSE}} = \int_0^T \int_0^{\tau_{\max}} dN_i(t') \frac{(y_i(t') - \hat{y}_{i,t}(t'))^2}{\hat{\lambda}_{i,t}(t')} dt d\tau.$$

We propose a multi-stage or end-to-end version of TESAR-CDE (Figure 6.3 compares the proposed architectures with TE-CDE). Both predict the intensity from z_t as $\hat{\lambda}_{i,t}(t') = f_{\psi}^{\lambda}(z_t(t'))$. We assume there is a minimal sampling interval dt ; e.g., doctors might not measure covariates more than once per hour. Let $dN_{i,t}(t') = 1$ if instance i was observed in interval $(t, t']$, 0 otherwise. The intensity $\lambda_{i,t}(t')$ can then be approximated by minimizing the cross-entropy

$$\begin{aligned} \mathcal{L}_i^{\text{CE}} = & - \sum_{t=0}^T \sum_{\tau=0}^{\tau_{\max}} \left[dN_{i,t}(t') \log(\hat{\lambda}_{i,t}(t')) \right. \\ & \left. + (1 - dN_{i,t}(t')) \log(1 - \hat{\lambda}_{i,t}(t')) \right], \end{aligned}$$

where $t' = t + \tau$. For applications where no minimal time step dt exists,

neural point processes can be used to learn the intensity in continuous time [see e.g. 301].

The **two-step** procedure consists of two TE-CDE style models that are trained sequentially. A first model predicts the intensity $\lambda_{i,t}(t')$; a second model uses the inverse of these intensities $\lambda_{i,t}(t')^{-1}$ as weights in its weighted MSE loss. Alternatively, we can combine both tasks in a **multitask** framework to predict both intensities and outcomes:

$$\mathcal{L}^{\text{MT}} = (1 - \alpha)\mathcal{L}_i^{\text{WMSE}} + \alpha\mathcal{L}_i^{\text{CE}}, \quad (6.5)$$

with hyperparameter α balancing the importance of the two terms. The intensity loss only optimizes the intensity map f_ψ^λ ; the weighted MSE is used to optimize the rest of the network ($g, f_\theta, f_\phi, f_\psi^y$). Moreover, similar to Hassanpour and Greiner [304]’s architecture for learning importance weights to correct for treatment-outcome confounding, we do not backpropagate with respect to the intensity weights in the weighted MSE for outcome prediction, as this could bias the network to predict small weights (i.e. large intensities) in order to minimize the weighted MSE.

The potential benefits of the multitask framework are threefold. First, learning a shared representation z_t to predict both outcome and intensity results in fewer parameters. Second, it requires only training one network and one call to the ODE solver per iteration, resulting in computational speedups. Third, to reduce the variability due to importance weighting, we only optimize the shared representation of the multitask model for outcome prediction. For bias correction using importance weighting, the shared representation does *not* need to be a sufficient statistic for predicting the intensity. This is because we only need to care about the non-uniformity in observation intensity insofar as it is related to the outcome. The reason for this is conceptually identical to why one should not include predictors of treatment only (a.k.a. instruments) in a propensity score [305] and why sufficient dimensionality reduction before importance weighting is recommended in general applications with covariate shift [306]: importance weighting generally only needs to adjust for shifts in variables that are themselves predictors of the outcome. Our multitask learner implicitly enforces this by optimizing the shared representation based on the outcome loss only. The two-step and multitask architectures are illustrated in Figure 6.3b and Figure 6.3c. Appendices D.3 and D.4 provide more details on the training procedure and implementation.

6.5 Results

To assess the impact of informative sampling, we propose a novel simulation environment that allows us to control the level of informativeness and assess its effect on the resulting model’s performance. Our simulation is inspired by real-world randomized controlled trials that compared treatment regimes in the context of lung cancer [307]–[309]. Given the patient’s history, our goal is to forecast the patient’s tumor size for a potential future treatment plan. Our code is available at <https://github.com/toonvds/TESAR-CDE>.

6.5.1 Simulation: lung cancer treatment

Following existing work [e.g., 288], [291], we simulate data based on the tumor growth model of Geng, Paganetti, and Grassberger [310]. To analyze the effect of informative sampling, we combine this tumor growth model with a sampling mechanism in which the degree of informativeness can be controlled. We refer to Appendix D.5 for more detailed information and visualizations.

Tumor growth simulation. We simulate tumor growth based on a pharmacokinetic-pharmacodynamic model of Geng, Paganetti, and Grassberger [310]. This model simulates the outcome, tumor volume $Y(t)$, based on the historical tumor volume, tumor growth, chemotherapy, and radiotherapy:

$$\frac{dY(t)}{dt} = \left[\overbrace{1 + \rho \log\left(\frac{K}{Y(t)}\right)}^{\text{Tumor growth}} - \overbrace{\beta_c C(t)}^{\text{Chemotherapy}} - \underbrace{(\alpha_r d(t) + \beta_r d(t)^2)}_{\text{Radiotherapy}} + \underbrace{\epsilon(t)}_{\text{Noise}} \right] Y(t), \quad (6.6)$$

with $K, \rho, \beta_c, \alpha_r, \beta_r, \epsilon_t$ sampled following Geng, Paganetti, and Grassberger [310]; $C(t)$ and $d(t)$ are set following existing work [272], [285], [291].

Treatment plans. We differentiate between a sequential and concurrent treatment regime [309]. In the sequential treatment arm, patients receive weekly chemotherapy for five weeks, followed by weekly radiotherapy for five weeks. In the concurrent treatment arm, patients biweekly receive both chemotherapy and radiotherapy for ten weeks. Patients are randomly divided among the two treatment arms based on a Bernoulli distribution with probability $p = 0.5$. This way, there is no confounding: treatment assignments are random and do not change during the trial.

Observation process. We observe patients based on a patient-specific, history-dependent intensity process. This is achieved by simulating each

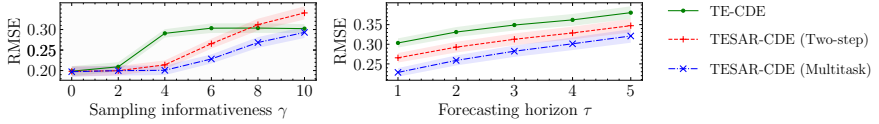


Figure 6.4: **Results for varying informativeness γ and different forecasting horizons τ .** We show the RMSE \pm SE over ten runs. **(Left)** RMSE for increasing levels of informativeness γ , keeping the forecasting horizon fixed at $\tau = 1$. **(Right)** RMSE for an increasing forecasting horizon τ up to five days, keeping informativeness fixed at $\gamma = 6$.

patient’s observation process with intensity $\lambda_i(t)$ in which γ controls the informativeness:

$$\lambda_i(t) = \sigma \left[\gamma \left(\frac{\bar{D}_i(t-)}{D_{\max}} - \frac{1}{2} \right) \right], \quad (6.7)$$

where σ denotes the sigmoid function. $D_{\max} = 13\text{cm}$ denotes the maximal tumor diameter and $\bar{D}(t-)$ is the average tumor diameter over the past 15 days. By simulating the observation process in this way, we can control the degree of informativeness: $\gamma = 0$ implies SCAR as $\lambda_i(t) = 0.5$, while $\gamma > 0$ implies SAR with a larger γ implying more informativeness or dependence between the tumor size and intensity. As γ increases, patients with larger tumors are more likely to be observed, those with smaller tumors less.

Experimental setup. We assume treatments are always observed, as these are planned in advance and administered in the hospital. Nevertheless, at treatment time, we do not necessarily observe the patient’s tumor size, e.g., because observing tumor size requires an invasive procedure separate from the treatment. For the test set, we observe all information at daily intervals. This idealized setup allows us to assess whether the model is able to learn the underlying distribution, as opposed to fitting the observed samples. Similarly, the test data also contains the potential outcomes for both treatment arms. For each experiment, we generate a training set with 200 patients, validation set with 50 patients, and test set with 200 patients, all over a period of 120 days.

6.5.2 Empirical results

This section presents the empirical results using the experimental setup described above. More specifically, we aim to answer three questions: (1) What is the impact of informative sampling?; (2) What is the impact of observation scarcity?; and (3) When does informativeness matter? In Appendix D.6, we present additional experiments to evaluate TESAR-CDE’s intensity prediction and analyze the sensitivity of the multitask configuration to hyperparameter α .

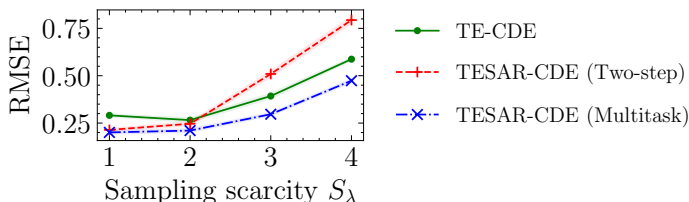


Figure 6.5: **Observation scarcity.** We show the RMSE \pm SE over ten runs at increasing observation scarcity S_λ for fixed informativeness ($\gamma = 4$) and forecasting horizon ($\tau = 1$).

What is the impact of informative sampling? We compare the different models at varying levels of informativeness in Figure 6.4. Increasing γ makes the observation process more informative by having patients with a large tumor visit more and patients with small tumors less (Equation (6.7)). The left side of Figure 6.4 shows the RMSE of the different models at increasing informativeness. As sampling becomes more informative, TE-CDE’s performance deteriorates and is outperformed by the proposed TESAR-CDE. The multitask variant in particular is robust to high levels of informativeness, achieving the lowest RMSE overall. The two-step variant generally outperforms TE-CDE, but performs worse for very high informativeness. As a high γ results in very low intensities for some patients, the importance weights of these observations induce large variance and worse generalization properties in the weighted loss. This phenomenon is a well-known issue in importance weighting more generally [311]. The right of Figure 6.4 shows the performance at different forecasting horizons τ ranging from one to five days at a fixed informativeness of $\gamma = 6$. Both TESAR-CDE variants outperform the standard TE-CDE over all horizons, with the multitask variant again performing the best overall.

What is the impact of observation scarcity? We analyze the influence of less frequent sampling across all patients. We simulate lower overall sampling by scaling all intensities as $\lambda'(t) = \frac{\lambda(t)}{S_\lambda}$ with $S_\lambda \in \{1, 2, 3, 4\}$. Figure 6.5 shows the impact of increasing scarcity on the resulting RMSE. As expected, all models perform worse with less frequent sampling. The two-step TESAR-CDE performs worse than the baseline TE-CDE as scarcity increases, while the multitask TESAR-CDE is again the best performing model overall. This result indicates that the parameter efficiency of the multitask model can be helpful when sampling is scarce.

When does informativeness matter? The previous experiments analyzed informative observation processes where the observation intensity $\lambda(t)$ was directly related to the outcome $Y(t)$. Next, we analyze a special case

of SAR where the intensity depends on information that is *completely unrelated* to the outcome or treatments. This extreme scenario mimics a situation in which there are patients that visit often for reasons unrelated to underlying symptoms or outcome, e.g. when they suffer from hypochondria. We examine this using an intensity that depends on covariates x^λ : $\lambda_i(t) = \sigma\left(\gamma \sum_j^d (c_j x_j)\right)$, where we include $d = 10$ static variables $x^{(i)}$ for each patient, with each $x_j \sim \mathcal{N}(0, 1)$ influencing the intensity through a coefficient $c_j \sim \mathcal{U}(-1, 1)$, but not affecting the patient in any other way. Figure 6.6 shows the prediction error of the different models for an increasing γ , averaged over $\tau \in \{1, \dots, 5\}$. In this scenario, the sampling mechanism does *not* significantly affect performance, with all models having similar performance. If anything, TESAR-CDE (Multitask) performs slightly worse, possibly due to importance weighting being unnecessary and adding variance in this scenario. This result indicates that informative sampling may only matter when it depends on factors influencing *both* observation intensity and outcome.

6.6 Conclusion

This work analyzed and formalized an essential challenge in learning to forecast treatment outcomes over time from observational data: the presence of informative sampling. We differentiated between different sampling mechanisms depending on the causal role of the observation intensity. This categorization allowed us to identify an overlooked, yet common setting in which observations are sampled irregularly over time with the observation intensity depending on the history of the instance’s covariates, outcome, and/or treatments. We formalized learning in this context and analyzed it as a problem of covariate shift. Based on this, we proposed a general framework for learning under informative sampling and a novel method, TESAR-CDE, that instantiates this framework using Neural CDEs. Empirical results demonstrate the improved performance of TESAR-CDE over the current state-of-the-art when sampling is informative.

Accounting for informative sampling when learning to forecast treatment outcomes relies on strong identification assumptions regarding both the treatment assignment and sampling mechanism. As these assumptions are untestable, we need to rely on domain expertise to judge their plausibility in practical applications. For example, the sampling at random assumption would be violated if the observation intensity is affected by an unobserved cause of the outcome. While beyond the scope of the current work, we believe that exploring learning under violations of these assumptions is an important and fruitful area for future research – similar to the rich lines of work exploring estimation of treatment effects with hidden confounders or missing treatment

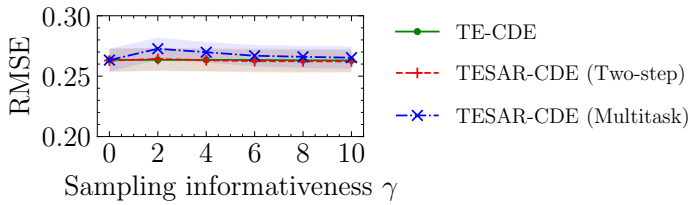


Figure 6.6: **Outcome-unrelated sampling.** We show the RMSE \pm SE over ten runs for a sampling mechanism unrelated to the outcome $Y(t)$ in function of informativeness γ .

information.

7

AUTOCATE: TOWARDS END-TO-END, AUTOMATED TREATMENT EFFECT ESTIMATION

Accurate estimation of heterogeneous treatment effects is critical in domains such as healthcare, economics, and education. While machine learning (ML) has led to significant advances in estimating conditional average treatment effects (CATE), real-world adoption of these methods remains limited due to the complexity of implementing, tuning, and validating them. To this end, we advocate for a more holistic view on the development of ML pipelines for CATE estimation through automated, end-to-end protocols. We formalize the search for an optimal pipeline as a counterfactual Combined Algorithm Selection and Hyperparameter optimization (CASH) problem. We introduce **AUTOCATE**, the first automated solution tailored for CATE estimation that addresses this problem based on protocols for evaluation, estimation, and ensembling. Our experiments show how **AUTOCATE** allows for comparing different protocols, with the final configuration outperforming common strategies. We provide **AUTOCATE** as an open-source software package to help practitioners and researchers develop ML pipelines for CATE estimation.

7.1 Introduction

Accurately estimating causal effects is crucial for high-stakes decisions in domains such as healthcare, education, and economics. Despite advances in machine learning (ML) for estimating the conditional average treatment effect (CATE), real-world adoption remains limited due to the *complexity of developing ML pipelines for CATE estimation*. Methods often involve numerous hyperparameters, and their performance varies significantly across data sets and applications. Moreover, validating counterfactual predictions and tuning pipelines is highly challenging, and the performance of different evaluation criteria varies with the data generating process [269]. For practitioners unfamiliar with ML, such as clinicians or marketers, these challenges often outweigh potential benefits, hindering the practical use of these techniques. To overcome this, we advocate for *automated, end-to-end solutions* for learning ML pipelines for CATE estimation.

The challenge of automated CATE estimation. Despite automated ML (AutoML) making significant progress [see 312], existing solutions do not address the unique challenges of CATE estimation. A key problem is the *lack of ground truth* CATE: the treatment effect is the difference between the outcomes with and without treatment, but only one of these outcomes is observed for each instance. Additionally, which outcome is observed depends on *confounding* variables (e.g., older patients may be more likely to receive treatment), leading to covariate shift [201]. Finally, CATE estimation pipelines are *more complex* than those in supervised learning. Metalearners combine multiple baselearners, possibly including both classification and regression models. Risk measures themselves also require predictions and, therefore, tuning of ML pipelines. These unique challenges complicate both the training and validation of ML pipelines and highlight the need for automated, end-to-end approaches tailored to CATE estimation, which is the focus of this work.

Contributions. To tackle these challenges, we propose a practical and comprehensive solution as the *automated, end-to-end construction and validation of ML pipelines* for CATE estimation:

- **COUNTERFACTUAL CASH**—We formalize the optimization of CATE estimation pipelines as a counterfactual Combined Algorithm Selection and Hyperparameter optimization (CASH) problem. Our solution, **AutoCATE**, automates the search for optimal configurations across preprocessors, metalearners, evaluators, baselearners, and their hyperparameters. The process is organized into three stages—*evaluation*, *estimation*, and *ensembling*—each including several design choices.
- **END-TO-END PROTOCOLS**—We develop end-to-end protocols that ensure

robust performance across diverse data sets and applications. Our approach addresses key aspects often overlooked in CATE estimation, such as preprocessing, feature selection, or ensembling. This perspective uncovers novel *insights* (see Figure 7.1), *questions* (e.g., the intricate trade-off between using data for training or validation) and *solutions* (e.g., multi-objective optimization with different evaluation criteria).

- **SOFTWARE PACKAGE**—We provide AutoCATE as an open-source software package, enabling automated CATE estimation in a few lines of code. This way, we democratize access to advanced ML techniques for CATE estimation and make them accessible for practitioners unfamiliar with ML. Additionally, AutoCATE provides a platform for future research, encouraging research on all aspects of the ML pipeline for CATE estimation that supports practical, real-world applications.

7.2 Related Work

Our work is most related to two areas in ML: (1) AutoML, and (2) CATE estimation and validation.

7.2.1 Automated Machine Learning (AutoML)

AutoML focuses on the automatic and efficient construction of high-performing ML pipelines. This entails making a series of design choices regarding preprocessing, feature transformation and selection, ML algorithms, and hyperparameter tuning [313]. As the optimal choices depend on the data and task, AutoML is essentially a *search problem*. While combinations could be tried randomly, more efficient search methods have been developed, e.g., based on Bayesian optimization [314]–[316]. Similarly, meta-learning has been applied to integrate information across other data sets in the search [317]. AutoML has made significant progress across data modalities, such as structured data [318], text [319] or images [320]. A critical aspect of AutoML is its accessibility, often provided through low-code solutions for practitioners unfamiliar with ML [318], [321]–[323].

Automated solutions exist for a wide range of tasks, including semantic segmentation [324], machine translation [325], reinforcement learning [326], or time series forecasting [322]. For more comprehensive overviews, see [327] and [312]. However, to the best of our knowledge, *AutoML has not yet been applied to CATE estimation*. As discussed, estimating treatment effects presents *unique challenges*, such as the absence of a ground truth, covariate shift due to confounding, and the need for intermediary models in metalearners and risk measures. These complexities render standard Au-

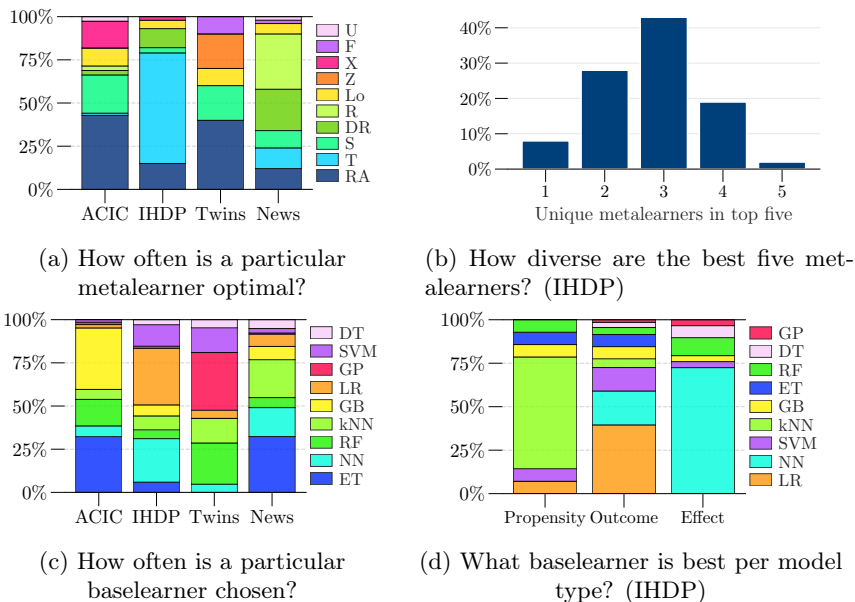


Figure 7.1: **AutoCATE enables insights into CATE estimation.** We analyze hundreds of pipelines optimized by AutoCATE (see Section 7.5). *Metalearners*—(a) Different metalearners can be optimal for a data set, highlighting the need for searching across them. (b) The top five pipelines often feature a mix of different metalearners (e.g. $\{T, T, RA, RA, DR\}$: 3 unique types), showing that different metalearners can perform well and suggesting potential for combining them. *Baselearners*—(c) The chosen baselearners are also diverse, and (d) different model types favor different ones. Using a single baselearner is thus likely suboptimal, supporting our choice to tune submodels independently.

toML approaches ill-suited for CATE estimation and illustrate the need for approaches specialized to CATE estimation.

7.2.2 Treatment Effect Estimation and Model Validation

Estimation. Various ML methodologies have been proposed for estimating treatment effects. Metalearners are general strategies for using standard supervised learning algorithms for CATE estimation [208]. Additionally, various ML algorithms have been adapted for CATE estimation, such as Gaussian processes [204], neural networks [201], [216], decision trees [206],

or random forests [207], [328]. Notably, other parts of the ML pipeline are also more complicated when estimating treatment effects, such as missing value imputation [329], feature selection [330], and ensemble selection [331].

Building an ML pipeline for CATE estimation presents significant challenges, related to the absence of ground truth CATE and the number of design choices involved. Due to the *no free lunch theorem*, no ML algorithm be optimal in all possible settings. Additionally, there is no globally optimal metalearner, as performance similarly depends on the (unknown) data generating process and sample size [209]. Finally, *tuning is more involved*: for example, a *DR-Learner* combines four models (to estimate the propensity, the outcome per treatment group, and the final treatment effect)—each of which can be a different baselearner with separate hyperparameters.

Model validation. As the CATE is unobserved, various evaluation criteria have been proposed for validating CATE estimators. A common approach is the error in predicting the observed potential outcome μ , i.e., the μ -risk. However, this criterion has several limitations [269], [332]: it does not account for confounding, may not accurately predict CATE error¹, and is not applicable to estimators that directly predict the CATE. To mitigate the first issue, an inverse propensity weighted variant μ_{IPW} -risk, can be considered. Other evaluation criteria address all issues by constructing labels based on plug-in estimates (e.g., *S*- or *T*-risk) or metalearner pseudo-outcomes (e.g., *R*- and *DR*-risk), see Appendix E.2.2 for a detailed overview.

There is *no consensus on the optimal validation criterion*. While [333] and [332] advocate for the *R*-risk, [331] favor the *T*- and *DR*-risk. Conversely, [269] show that the effectiveness of different risk measures varies with various factors, such as the metalearner and data generating process, with no single criterion being universally optimal. Additionally, [332] stress the flexibility of the estimators used to construct the pseudo-labels, with [331] recommending the use of AutoML. These complexities and design choices highlight the need for automated procedures.

7.3 Problem Formulation

Notation and assumptions. We represent an instance by a tuple (x, t, y) , with covariates $X \in \mathcal{X} \subset \mathbb{R}^d$, a treatment $T \in \mathcal{T} = \{0, 1\}$, and an outcome $Y \in \mathcal{Y} \subset \mathbb{R}$. The potential outcome Y associated with a treatment t is denoted as $Y(t)$. We aim to estimate the conditional average treatment effect

¹For example, consider the case where both potential outcomes are overestimated by the same amount. Even though μ -risk would indicate a poor model quality, the resulting CATE estimates would still be accurate.

(CATE): $\tau = \mathbb{E}[Y(1) - Y(0)|X]$. Estimating the CATE from observational data requires standard assumptions (see Appendix E.1.2). More background on CATE estimation is provided in Appendix E.1.

Goals and challenges. We aim to develop a *general procedure* for learning a pipeline for CATE estimation from an observational data set. Formally, this is a *counterfactual Combined Algorithm Selection and Hyperparameter optimization* (CASH) problem. It involves searching over ML pipelines a_h with algorithms $a \in \mathcal{A}$ and hyperparameters $h \in \mathcal{H}_a$ to minimize the error on test data $\mathcal{D}_{\text{test}}$:

$$\arg \min_{a,h} \mathcal{L}(a_h | \mathcal{D}_{\text{test}}). \quad (7.1)$$

An algorithm a can be an ML method tailored for CATE estimation or a metalearner combining one or more baselearners. Solving the counterfactual CASH problem involves several *unique challenges*. An algorithm’s quality of fit on the train data $\mathcal{L}(a_h | \mathcal{D}_{\text{train}})$ is unobserved, as there is no ground truth CATE. Additionally, there is covariate shift between the observational training data and test data due to confounding. Both points present challenges for both *building* and *validating* an ML pipeline.

7.4 AutoCATE: End-To-End, Automated CATE Estimation

AutoCATE finds an optimal ML pipeline in three stages: *evaluation*, *estimation*, and *ensembling*:

- (1) **EVALUATION:** In the first stage, we construct a proxy risk for \mathcal{L} based on a risk measure (e.g., R -risk) and evaluation metric (e.g., MSE). To accurately estimate this risk on the validation data, we perform an automated search over preprocessors, ML algorithms, and their hyperparameters.
- (2) **ESTIMATION:** The second stage automatically searches over combinations of preprocessors, metalearners, baselearners, and their hyperparameters to obtain ML pipelines for CATE estimation.
- (3) **ENSEMBLING:** The final stage uses the proxy risk from the first stage to select and combine estimation pipelines from the second stage. The result can be a single ML pipeline or an ensemble.

A high-level overview of AutoCATE’s functionalities and building blocks is shown in Figure 7.2.

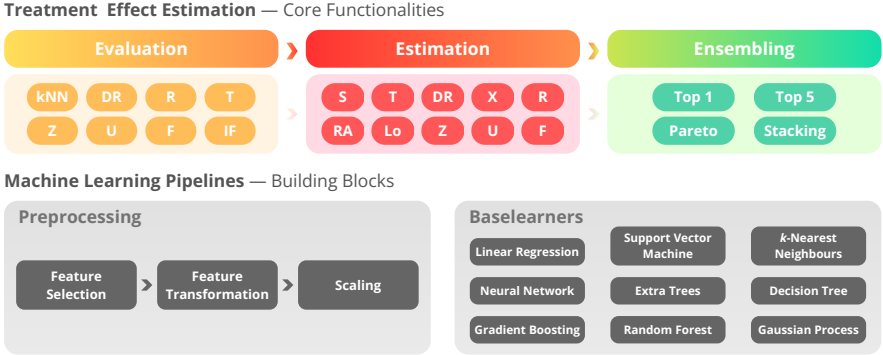


Figure 7.2: **AutoCATE overview.** We estimate treatment effects in three stages: (1) *Evaluation*—learning the appropriate risk measure(s), (2) *Estimation*—tuning a CATE estimation pipeline, and (3) *Ensembling*—selecting a final model or constructing an ensemble. We build ML pipelines for evaluation and estimation based on a collection of *preprocessing algorithms* and ML *baselearners*.

7.4.1 Stage 1: Evaluation—Designing a Proxy Risk and Evaluation Protocol

The counterfactual CASH problem requires minimizing $\mathcal{L}(a_h | \mathcal{D}_{\text{test}})$, which involves two challenges: the lack of ground truth τ and the presence of covariate shift due to confounding. To tackle these, the evaluation stage measures risk by learning *pseudo-labels*—i.e., proxies for τ —from validation data.

Risk measures. AutoCATE includes *different possible risk measures*, described in Appendix E.2.2. We include pseudo-labels used in metalearners (*DR*-, *R*-, *Z*-, *U*-, and *F*), plug-in risks (*T* and *1NN*), and a risk approximation using influence functions (*IF*). We exclude the μ - and μ_{IPW} -risks as they do not apply to all metalearners, and the *S*-risk due to poor results in prior work [e.g., 331]. As constructing these risk measures requires accurately estimating nuisance parameters, we search over preprocessing and ML algorithms to find good-performing ML pipelines.

There is no ground truth, and different measures may be preferable depending on the (unknown) data generating process. To make our evaluation more robust, we allow for *combining different measures*. Similarly, since pseudo-outcomes are learned from data, there is no “true” version, enabling us to construct multiple version of a single risk (e.g., two *R*-risks). Using multiple risk measures results in a multi-objective search problem. To account for

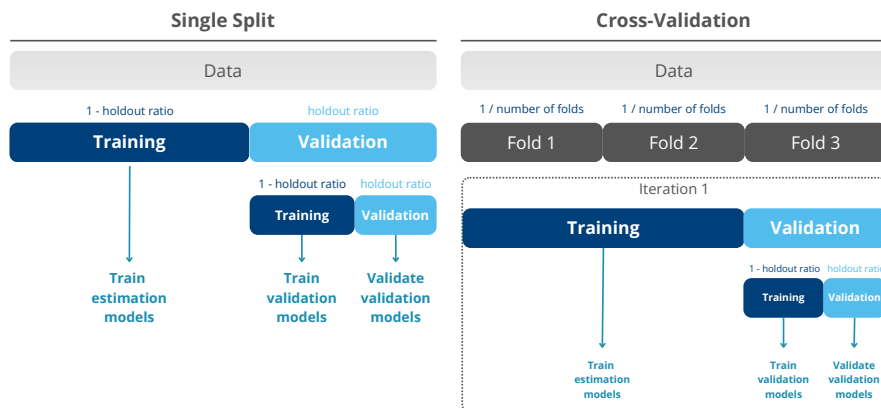


Figure 7.3: **Evaluation framework.** We show two possible frameworks for validating pipelines based on a single split or a cross-validation procedure. For each, the data is split in three groups to (1) train the estimation pipelines, (2) train the validation pipelines, and (3) validate the validation pipelines.

the varying scales of different risks, we normalize them by comparing each model’s performance to an average treatment effect (ATE) baseline.

Metrics and implementation. Given a risk measure, different metrics can compare the pseudo-outcomes and CATE predictions to evaluate the quality of the ML pipeline. We include *general metrics* of predictive accuracy, like the mean squared error (MSE) or mean absolute percentage error (MAPE), and metrics related to a *downstream application*, such as the Area Under the Qini Curve (AUQC) when ranking effects [13]. The R -risk requires a metric that accommodates weights. Finally, we allow for a stratified training-validation split or a stratified k -fold cross-validation procedure. Figure 7.3 shows more information on these evaluation frameworks.

7.4.2 Stage 2: Estimation—Building a CATE Estimation Pipeline

Different *metalearners* can be used to estimate the CATE. Metalearners are general frameworks for using ML algorithms to estimate treatment effects. As such, they are versatile, accommodate various ML algorithms, and can be efficiently trained using existing ML packages. Common examples include the S -Learner (single model with the treatment as a feature), Lo -Learner (single model with treatment interaction terms), and T -Learner (separate models for each treatment group). Other metalearners use pseudo-outcomes that converge to the treatment effect, such as the DR -, X -, R -, RA -, Z -,

U -, and F -Learners. Appendix E.2.1 provides more detailed information on each metalearner. Our package uses the `CausalML` implementations where available [334].

7.4.3 Stage 3: Ensembling—Selecting and Ensembling Estimation Pipelines

The pipelines from the *estimation* stage are evaluated with risk measures from the *evaluation* stage. The final *ensembling* stage selects the best pipeline(s) for prediction. No established methods exist for ensembling CATE estimators and, due to the lack of ground truth, most standard ensembling methods are not applicable. AutoCATE includes can select the best-performing pipeline or the top five for improved robustness and accuracy. We also include a novel stacking procedure that assigns weights (between zero and one) to each pipeline and optimizes these to minimize the squared error with respect to the pseudo-outcomes. The weights are regularized, with tuning on a holdout set.

With *multiple risk measures* in a multi-objective search, there may not be a single optimal pipeline, but rather a Pareto frontier. One strategy is to select all Pareto optimal points, though pipelines that perform very well on only a single measure may not work well generally. To select pipelines with good general performance, we can select the pipeline (or the top five) with the lowest average risk across objectives. Similarly, we can select based on each pipeline’s Euclidean distance to the origin, or its average rank across objectives. Finally, we can apply the abovementioned stacking procedure for each risk measure separately and averaging the weights in a final stacked pipeline.

7.4.4 ML Pipeline Building Blocks: Preprocessing and ML Baselearners

We construct ML pipelines in both the *evaluation* and *estimation* stage. The building blocks for these include preprocessors and ML algorithms, all built on top of `scikit-learn` [335]. For *preprocessing*, we provide different feature selection and scaling algorithms. As *baselearners*, we include different ML algorithms with both classification and regression counterparts, ranging from linear regression to random forests. We provide more information in Appendix E.2.3.

The final search space includes a variety of preprocessors, metalearners, baselearners, and their hyperparameters. Efficient *optimization schemes* such as Bayesian optimization could be used, but we use random search through-

out this work to focus on other design choices in `AutoCATE`. Nevertheless, we implement our search using `optuna` [336], allowing easy integration of sophisticated optimizers like a Tree-structured Parzen Estimator [314].

7.4.5 Low-Code CATE Estimation Through `AutoCATE`'s API

`AutoCATE` is implemented in Python, following `scikit-learn`'s design principles [335]. The low-code API enables automated CATE estimation with just four lines of code:

```
1 from src.AutoCATE import AutoCATE
2 autocate = AutoCATE()
3 autocate.fit(X_train, t_train, yf_train)
4 cate_pred = autocate.predict(X_test)
```

Initialization arguments can be specified (e.g., the number of estimation trials; see Appendix E.2.5).

7.5 Empirical Evaluation: Comparing Automated Strategies

This section empirically compares design choices for solving the counterfactual CASH problem for all three stages: *evaluation* (7.5.2), *estimation* (7.5.3), and *ensembling* (7.5.4). We identify best practices and benchmark the resulting configuration against common approaches for CATE estimation (7.5.5).

7.5.1 Experimental Setup: Data and Evaluation Metrics

Our experiments compare various automated, end-to-end strategies for learning a CATE estimation pipeline. Using `AutoCATE`, we evaluate design choices in each stage: evaluation, estimation, and ensembling. To obtain general insights, we leverage a collection of standard benchmarks for CATE estimation: IHDP [205], ACIC [337], News [200], and Twins [217]; see Appendix E.3 for details. These semi-synthetic benchmarks include 247 distinct data sets that vary in outcome (regression and classification), dimensionality, size, and application area, allowing for a comprehensive analysis `AutoCATE`. Unless noted otherwise, results are reported in precision in estimating heterogeneous treatment effects (PEHE): $\sqrt{\text{PEHE}} = \sqrt{(\tau - \hat{\tau})^2}$.

Table 7.1: **Performance for validation based on different risk measures.** Results in $\sqrt{\text{PEHE}}_{\pm\text{SE}}$ (lower is better). **Bold** highlights the best results, with underlined values falling within 1 standard error. Results for 50 evaluation trials and 50 estimation trials with a T -Learner and gradient boosting.

	DR	F	IF	kNN	R	T	U	Z
<i>IHDP</i>	2.12 $\pm_{.34}$	3.33 $\pm_{.55}$	3.13 $\pm_{.45}$	2.22 $\pm_{.36}$	3.37 $\pm_{.71}$	2.15 $\pm_{.35}$	3.58 $\pm_{.72}$	5.40 $\pm_{.86}$
<i>ACIC</i>	<u>1.56</u> $\pm_{.09}$	1.74 $\pm_{.10}$	2.52 $\pm_{.16}$	1.74 $\pm_{.10}$	1.63 $\pm_{.10}$	1.52 $\pm_{.09}$	1.72 $\pm_{.09}$	2.40 $\pm_{.15}$
<i>Twins</i>	.333 $\pm_{.00}$.340 $\pm_{.00}$.340 $\pm_{.01}$	<u>.323</u> $\pm_{.00}$.335 $\pm_{.00}$.323 $\pm_{.00}$.359 $\pm_{.01}$.350 $\pm_{.01}$
<i>News</i>	2.42 $\pm_{.07}$	<u>2.48</u> $\pm_{.07}$	2.73 $\pm_{.09}$	<u>2.43</u> $\pm_{.07}$	2.51 $\pm_{.08}$	<u>2.42</u> $\pm_{.07}$	2.60 $\pm_{.09}$	3.02 $\pm_{.11}$

(a) Comparing downstream performance for different risk measures

	Combining risks			T -risk—Multiple versions				Best single
	All	DR, T	DR, T, kNN	Top 1	Top 2	Top 3	Top 5	
<i>IHDP</i>	2.48 $\pm_{.36}$	2.19 $\pm_{.35}$	<u>2.13</u> $\pm_{.35}$	2.15 $\pm_{.35}$	2.15 $\pm_{.35}$	2.17 $\pm_{.35}$	2.11 $\pm_{.36}$	2.12 $\pm_{.34}$
<i>ACIC</i>	1.94 $\pm_{.13}$	<u>1.58</u> $\pm_{.09}$	1.60 $\pm_{.09}$	<u>1.52</u> $\pm_{.09}$	1.54 $\pm_{.08}$	<u>1.55</u> $\pm_{.09}$	1.52 $\pm_{.08}$	<u>1.52</u> $\pm_{.09}$
<i>Twins</i>	.331 $\pm_{.01}$.323 $\pm_{.00}$.324 $\pm_{.00}$.323 $\pm_{.00}$	<u>0.323</u> $\pm_{.00}$.323 $\pm_{.00}$.324 $\pm_{.00}$.323 $\pm_{.00}$
<i>News</i>	<u>2.52</u> $\pm_{.07}$	<u>2.41</u> $\pm_{.06}$	2.41 $\pm_{.07}$	<u>2.42</u> $\pm_{.07}$	<u>2.41</u> $\pm_{.07}$	<u>2.43</u> $\pm_{.07}$	<u>2.43</u> $\pm_{.07}$	<u>2.42</u> $\pm_{.07}$

(b) Comparing downstream performance for different combinations of risk measures

7.5.2 Analyzing AutoCATE—Stage 1: Evaluation Protocol

We analyze the evaluation protocol by comparing risk measures, metrics, and evaluation procedures.

How to measure risk regarding CATE predictions?

What risk measure works best? We compare predictive error resulting from model selection with different risk measures in Table 7.1a. Three options consistently show low error: the DR -, kNN -, and T -risk. These results largely correspond with existing work. Curth and Schaar [269] and Mahajan, Mitliagkas, Neal, *et al.* [331] similarly found the DR -risk to work well, though the kNN -risk works comparatively better in our experiments. Although [269] reported worse results for the T -risk, both our findings and those in [331] show that it *can* give good results with proper tuning of the underlying models. We further analyze the impact of tuning in Figure 7.4: increased tuning for the evaluation models generally results in better downstream performance.

Is it beneficial to use multiple risk measures? We explore the impact of

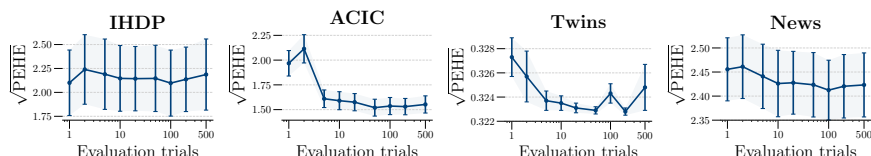


Figure 7.4: **How many iterations should we tune evaluation models?** We compare downstream results, based on different number of trials, used to tune the models underlying the evaluation metrics. Results for a T -risk and 50 estimation trials with a T -Learner and gradient boosting.

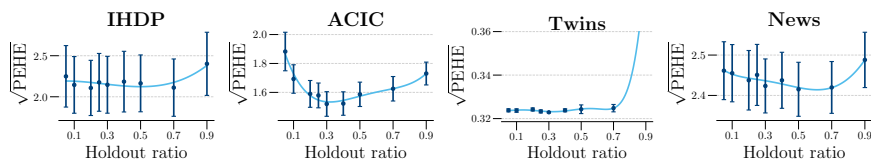


Figure 7.5: **How much data to use for evaluation?** We show results for different holdout ratios and fit a polynomial function for each data set to gain insight into the optimal ratio. Results for 50 evaluation trials with a T -risk and 50 estimation trials with a T -Learner and gradient boosting.

combining different risk measures in a multi-objective search, hypothesizing that this could lead to more robust pipeline selection as each measure is a different proxy to the same ground truth. Table 7.1b shows both results for risk measure combinations, and for multiple versions of a single measure based on different estimates. We observe that combining different types or different versions of risk measures can indeed improve performance, though no strategy substantially improves upon the best single measure.

What evaluation procedure to use?

How to set the holdout ratio? Risk measures require estimates learned from validation data, creating a *trade-off* between using data for evaluation or estimation. Figure 7.5 presents results for different holdout ratios, illustrating this trade-off and showing that a holdout ratio of 30-50% generally works well. We use 30% for holdout in the rest of this work. Although more folds in cross-validation often improve model performance in supervised settings, we do not observe this effect for AutoCATE (see Table B3), likely due to the complex interplay between the number of folds and the holdout ratio.

What evaluation metric to use? All previous experiments used the mean squared error (MSE) to compare the predicted CATE and pseudo-outcome(s), corresponding with the goal of minimizing PEHE. However,

Table 7.2: **Comparing different evaluation metrics.** We compare model selection with different evaluation metrics. For the Twins data set, MAPE cannot be calculated, as the true CATE can be zero. **Bold** highlights the best results, with underlined values falling within 1 standard error. Colored cells show the hypothesis that matching metrics will yield the best performance. Results for 50 evaluation trials with a T -risk and 50 estimation trials with a T -Learner and gradient boosting.

	MSE	MAPE	AUQC		MSE	MAPE	AUQC
$\sqrt{\text{PEHE}}$	2.15 \pm 0.35	2.28 \pm .36	2.26 \pm .41	$\sqrt{\text{PEHE}}$	1.52 \pm .09	1.67 \pm .09	1.50 \pm .08
MAPE	1.76 \pm 1.30	1.40 \pm .94	0.50 \pm .15	MAPE	1.10 \pm .21	1.03 \pm .14	1.11 \pm .24
AUQC	0.92 \pm 0.01	0.88 \pm .02	0.96 \pm .01	AUQC	0.91 \pm .01	0.90 \pm .01	0.91 \pm .01
(a) IHDP				(b) ACIC			
	MSE	MAPE	AUQC		MSE	MAPE	AUQC
$\sqrt{\text{PEHE}}$.323 \pm .00	.323 \pm .00	.344 \pm .00	$\sqrt{\text{PEHE}}$	2.42 \pm .07	2.52 \pm .07	2.46 \pm .07
MAPE	—	—	—	MAPE	5.75 \pm .74	5.83 \pm .69	5.86 \pm .85
AUQC	0.00 \pm .00	0.00 \pm .01	0.03 \pm .01	AUQC	0.66 \pm .01	0.64 \pm .01	0.65 \pm .01
(c) Twins				(d) News			

depending on the downstream application, *alternative metrics* might be more important. Using these in AutoCATE is straightforward. Table 7.2 shows results for two such metrics: the mean absolute percentage error (MAPE) and area under the Qini curve (AUQC). As hypothesized, selecting models based on a particular metric generally improves performance for that metric.

7.5.3 Analyzing AutoCATE—Stage 2: Estimation Protocol

Given an *evaluation* protocol, we can compare strategies for the *estimation* stage. This section examines how including different metalearners and baselearners affects AutoCATE’s performance.

Metalearners. Figure 7.6 compares different versions of AutoCATE with either all meta- and baselearners, or only the best per category. The complete “AllMeta-AllBase” occasionally yields poor performance. Although results generally improve with more trials, this behavior persists after 100 trials for the News data. Further inspection reveals that bad iterations are due to instability of the R - and U -Learners: even when performing well on the validation set, they can perform exceptionally poor after retraining on all data. Other metalearners are *almost never* optimal. Consequently, “Best-Meta” excludes the R -, F -, Z -, and U -Learners, leading to improved stability

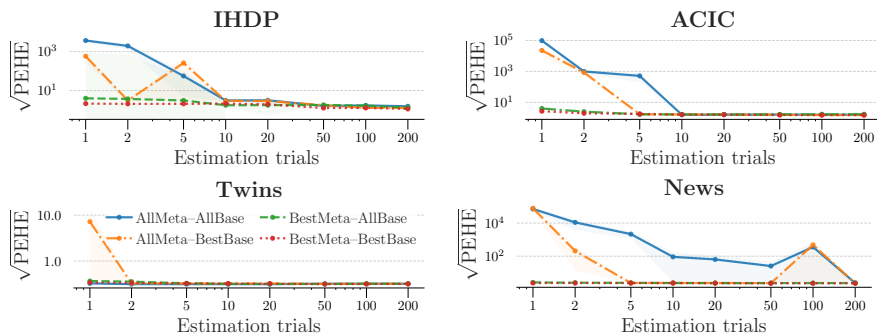


Figure 7.6: **What meta- and baselearners to include?** We compare different search spaces for AutoCATE, either including all metalearners (AllMeta) or only the best (BestMeta), as well as all baselearners (AllBase) or only the best (BestBase). Results for 50 evaluation trials with a T -risk.

and performance across all data sets except Twins. In Appendix E.4.2, we compare metalearners in terms of precision and time efficiency, and show how often each metalearner gets picked in the BestMeta configuration.

Baselearners. The “BestBase” versions in Figure 7.6 only use base learners that typically perform well with tabular data: random forests, extremely randomized trees, gradient boosting, and multilayer perceptrons. This constraint is applied to both evaluation and estimation pipelines. While selecting these baselearners improves performance, it is less significant than filtering metalearners.

7.5.4 Analyzing AutoCATE—Stage 3: Ensembling Protocol

The *ensemble* stage compares pipelines built in the estimation stage using the objective(s) learned in the evaluation stage. Selected pipelines are re-trained on the entire data and saved for inference.

Single objective. With a single objective, we can select the best pipeline (Top 1), the best five (Top 5), or use stacking to build a final estimator that combines all pipelines. Table 7.3a compares these strategies, showing that *combining pipelines improves performance* for all data sets except Twins. Appendix E.4.3 illustrates how an ensemble’s predictions can help assess an estimate’s uncertainty.

Multiple objectives. Model selection is more complex with multiple objectives. We can select the best pipelines based on the average normalized score, Euclidean distance to the origin, or average rank, to then select the

Table 7.3: **Ensemble strategies.** We compare ensembling strategies for a single or multiple objectives in terms of $\sqrt{\text{PEHE}}$. **Bold** highlights the best results, underlined values lie within 1 standard error. Results for 50 evaluation trials and 50 estimation trials with a T -Learner and gradient boosting.

	Top 1	Top 5	Stacking
<i>IHDP</i>	<u>2.15</u> \pm .35	1.90 \pm .34	<u>1.96</u> \pm .34
<i>ACIC</i>	1.52 \pm .09	1.34 \pm .08	<u>1.42</u> \pm .09
<i>Twins</i>	.323 \pm .00	.325 \pm .00	.344 \pm .00
<i>News</i>	2.42 \pm .07	2.33 \pm .06	<u>2.33</u> \pm .06

(a) Comparing ensemble strategies for a single T -risk

	Average		Distance		Ranking		Stacking Pareto	
	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5		
<i>IHDP</i>	2.19 \pm .35	1.84 \pm .31	2.27 \pm .37	2.99 \pm .54	3.58 \pm .66	2.99 \pm .54	<u>1.94</u> \pm .32	2.19 \pm .36
<i>ACIC</i>	1.58 \pm .09	1.35 \pm .08	1.55 \pm .08	<u>1.41</u> \pm .08	1.69 \pm .08	<u>1.41</u> \pm .08	1.43 \pm .09	1.50 \pm .08
<i>Twins</i>	.323 \pm .00	.325 \pm .00	.323 \pm .00	.341 \pm .00	.367 \pm .01	.341 \pm .00	.349 \pm .00	.326 \pm .00
<i>News</i>	2.41 \pm .06	2.32 \pm .06	2.42 \pm .07	<u>2.38</u> \pm .07	2.58 \pm .08	<u>2.38</u> \pm .07	<u>2.34</u> \pm .06	2.39 \pm .07

(b) Comparing ensemble strategies when combining DR - and T -risks

top one or top five pipelines. Alternatively, we can create stacking estimators for each objective and average the weights (“Stacking”), or select all Pareto optimal models (“Pareto”). Table 7.3b compares these strategies. Single pipelines typically underperform compared to ensembles built from the top five pipelines, all Pareto optimal pipelines, or stacking. Selecting based on average performance yields the best performance. No single strategy is consistently optimal.

7.5.5 Benchmarking AutoCATE Against Common Alternatives

This section compares the optimized configuration of AutoCATE with some common alternative approaches for tuning CATE estimation pipelines. These benchmarks select the best model using the error in predicting observed outcomes (μ -risk). We include both S - and T -Learners. For T -Learners, we tune models separately for the control and treatment groups. First, we compare a T -Learner with gradient boosting tuned based on the μ -risk against AutoCATE using only a T -Learner and gradient boosting optimized for T -risk. While these strategies are similar, AutoCATE evaluates the entire pipeline jointly and (potentially) adds preprocessing. Conversely, the tradi-

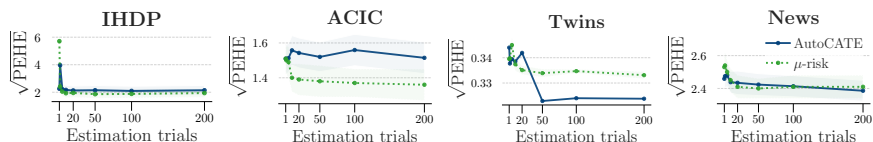


Figure 7.7: **Comparing AutoCATE with tuning based on μ -risk.** We compare tuning a T -Learner with gradient boosting using either AutoCATE (based on a T -risk) or tuning based on the MSE on the observed outcome. AutoCATE uses a T -risk with 50 evaluation trials and top 1 model selection.

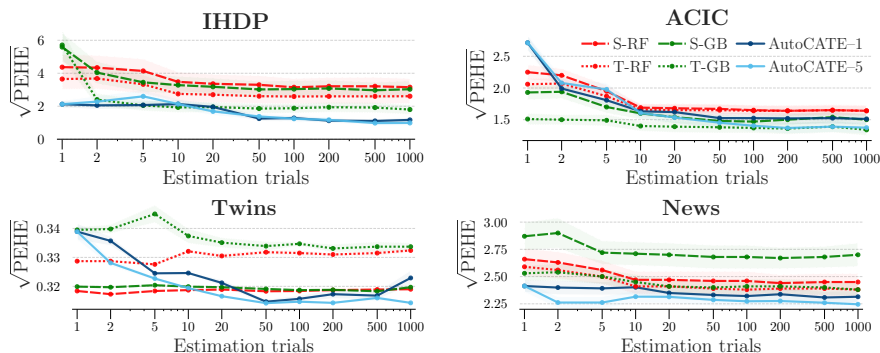


Figure 7.8: **Benchmarking AutoCATE.** We compare AutoCATE with common benchmarks using S - and T -Learners with random forests and gradient boosting. AutoCATE uses a T -risk with 50 evaluation trials and BestMeta-BestBase search spaces, with either Top 1 or Top 5 model selection.

tional T -Learner’s search is more efficient as it tunes models separately per group. Figure 7.7 compares the two approaches: the μ -risk strategy performs worse for Twins, but better for ACIC. Finally, Figure 7.8 compares AutoCATE with S - and T -Learners using random forests and gradient boosting. These approaches are conceptually simple, but represent common and strong baselines. We observe that, for each data set, AutoCATE can obtain at least competitive performance to the best approach. We include additional results on ranking treatment effects with data for uplift modeling in Appendix E.4.4.

7.6 Conclusion

Despite the availability of ML methods for CATE estimation, their *adoption remains limited*, due to the complexity of implementing, tuning, and validating them. We framed the problem of finding an ML pipeline for CATE

estimation as a *counterfactual CASH problem* and proposed **AutoCATE**: the first *end-to-end, automated solution* tailored for treatment effect estimation. Based on this solution, we analyzed design choices for evaluation, estimation, and ensembling, and identified best practices. The resulting approach was validated, outperforming widely used strategies for CATE estimation.

To maximize **AutoCATE**'s practical impact, several *limitations* need to be addressed. Although **AutoCATE** relies on standard *assumptions* for causal inference, it is crucial to assess its robustness against violations of these assumptions and potentially protocols for such scenarios. Additionally, most of the data used in this work is *semi-synthetic* (IHDP, ACIC, and News), which may not fully capture the complexities of real-world data. Although validating CATE estimates remains inherently challenging, approaches from related fields could offer inspiration [see e.g. 95].

AutoCATE enables a *comprehensive analysis* of existing methods (see Figure 7.1 and Appendix E.4.5), facilitating a better understanding of CATE estimation and guiding the development of new approaches. We envision opportunities for *future research* in all stages. For *evaluation*, advanced multi-objective strategies could improve performance and robustness. Novel methods for *estimation* could be automatically discovered using Neural Architecture Search. Generally, efficiency can be improved with better search algorithms or strategies (e.g., by re-using nuisance models across metalearners). Related to this, the optimal time allocation between the stages remains an open question, where meta-learning could help by incorporating data set characteristics [317]. Finally, more advanced *ensembling* could be developed (e.g., combining different metalearners).

Part IV

DECISION-FOCUSED CAUSAL INFERENCE

8

METALEARNERS FOR RANKING TREATMENT EFFECTS

Knowing how an instance will respond to an action is crucial for making informed decisions. In marketing, for example, knowing how a customer will respond to a promotion can help to personalize offerings, target the right customers, and increase conversions. Whereas existing work mostly focuses on accurately estimating treatment effects, operational decision-making often exclusively relies on the ranking of the effects, as a means of prioritizing instances for treatment (e.g., to target the customers with the largest treatment effects). However, existing methods for *treatment effect estimation* do not consider how the estimated effects result in an effect ranking and, consequently, the quality of resulting treatment policies. This mismatch between effect estimation and the operational needs may lead to suboptimal treatment allocation. Conversely, our work explores an alternative approach, *treatment effect ranking*, which directly learns a treatment allocation policy that prioritizes instances based on their treatment effect. We introduce ranking counterparts for a wide range of causal metalearners, building upon objective functions from learning to rank. To scale our methodology to large-scale

Table 8.1: *Treatment Allocation Examples*. We highlight several applications where treatment allocation is required, characterized by (1) an estimated treatment effect, (2) an optimization objective, and (3) operational constraints. In *marketing*, the goal is to target customer segments and drive conversions, while adhering to budget constraints. In *healthcare and epidemiology*, optimal vaccine allocation during pandemics aims to minimize population mortality, subject to vaccine supplies. In *maintenance*, technicians are assigned to maintain assets, prevent failures, and maximizing operational uptime. Finally, in *economics and policy design*, subsidy usage needs to be optimized by efficiently allocating public funds and maximizing health care impact.

<i>Application</i>	<i>Allocation problem</i>	<i>Treatment outcome</i>	<i>Objective</i>	<i>Constraints</i>
Marketing	Targeted advertising	Conversion	Incremental sales	Marketing budget
Healthcare	Pandemic response	Infection	Mortality reduction	Vaccine supply
Maintenance	Preventive maintenance	Failure rate	Asset uptime	Available technicians
Policy design	Targeted subsidies	Bed net purchase	Malaria prevention	Public/policy budget

data sets, we propose an efficient sampling procedure for optimizing these objectives. Theoretically, we show how ranking metalearners directly maximize the area under a policy’s Qini curve. Empirically, we validate our approach and illustrate its practical efficacy through experiments on both synthetic and real-world data.

8.1 Introduction











Decision-makers need to deal with uncertainty regarding the consequences of their decisions. An increasingly popular paradigm to address this challenge is the prediction-optimization framework. In a first *prediction* stage, data is used to estimate the effect of possible actions. In a second *optimization* stage, these predictions are integrated in an optimization problem with the aim of assigning personalized treatment recommendations, i.e., allocating treatments to instances to optimize an objective function, while satisfying operational constraints. These problems are common in various domains: e.g., marketing [338], healthcare [339], maintenance [10], or policy design [340] (see Table 8.1 for some examples). We focus on a specific class of treatment recommendation problems where instances need to be prioritized for treatment (e.g., recommending whom to treat). Our goal is to learn a treatment policy that prioritizes the optimal instances for treatment. A key part of this problem is estimating each instance’s response to a treatment—i.e., its treatment effect—from data using causal inference.

Prediction-Focused Learning: Effect Estimation A common approach to tackle treatment allocation problems is to first *predict* the effect of an action for each instance. To this aim, causal inference can support many decision-making problems: by analyzing the causal effect of past decisions, future decisions can be optimized. A common approach is to first *estimate the causal effect* of possible decisions using methods for treatment effect estimation. For example, in marketing, to estimate how different customers would respond to a marketing incentive. The effect estimates can be integrated in an *optimization* problem to make the final decisions regarding treatment allocation (e.g., to target a specific customer segment). This approach has been adopted to aid decision-making by a variety of technology and e-commerce companies [341]–[343].

Decision-Focused Learning: Effect Ranking Recent work, referred to as *decision-focused learning*, aims to integrate the learning and optimization steps. This approach recognizes that the predictive task (i.e., estimating treatment effects) is only part of a larger optimization problem (i.e., allocating treatments). By integrating the predictive model in the larger optimization pipeline, decision-focused learning aims to learn a predictive model that results in better performance for the downstream task [85]. The key idea is to align the construction of the predictive model with the optimization task.

We analyze a common type of treatment allocation problem, where treatments are allocated to the instances with the largest treatment effect. In these settings, we argue that directly learning an *effect ranking* might be more useful than *effect estimation*. As operational constraints might prevent decision-makers from treating every instance, we require knowing how to prioritize instances based on their treatment effect. Compared to independently estimating each instance’s effect, we argue that directly learning the ranking across instances may yield better results. Because *effect estimation* prioritizes accurate and well-calibrated effect estimates, it overlooks the estimates’ ranking and resulting decision quality (i.e., estimation and optimization are not aligned). While successful when predictions are perfect, this misalignment can result in suboptimal decision-making in reality. Conversely, our work demonstrates that *effect ranking* can directly optimize the quality of the treatment assignment (i.e., the ranking objective is perfectly aligned with the decision-making task). We contrast both approaches in Table 8.2. Additionally, empirical risk minimization only guarantees model generalization for the specific objective that was optimized for [344], further motivating us to find objectives that are aligned to the final optimization problem.

Table 8.2: *Comparing Pointwise Estimation and Effect Ranking for Treatment Allocation*. This illustrative example shows the true treatment effect for several instances, ordered from largest to smallest. The first model estimates each instance’s treatment effect fairly accurately in terms of MSE. However, the ranking of instances based on these estimates differs significantly from their true order, which will result in suboptimal treatment allocation when only some instances can be treated. For the second model, we observe the opposite scenario: estimates are poor in terms of MSE, but their ranking respects the true order.

	Instances (e.g., customers)						Performance	
							MSE	Ranking
True treatment effect	0.20	0.17	0.16	0.15	0.11	0.10	—	—
Effect estimation model	0.22	0.15	0.16	0.17	0.10	0.11		
Effect ranking model	0.25	0.22	0.20	0.15	0.10	0.05		

Contributions This work proposes decision-focused learning framework for treatment recommendation problems. We formalize the class of problems that can be tackled using effect ranking and discuss the underlying assumptions. We describe how learning to rank can be used for this task and propose different *causal metalearners for ranking effects*. Our contributions are as follows. (1) Conceptually, we formalize treatment allocation problems that can be solved using ranking and analyze the underlying assumptions (see Section 8.3). (2) Methodologically, we propose different metalearners for ranking treatment effects, based on pairwise and listwise ranking objectives that scale efficiently to large-scale data sets. We show how our proposed listwise objective directly optimizes the policy’s area under the Qini curve (see Section 8.4). (3) Empirically, we compare our proposed effect ranking with effect estimation using synthetic and real-world data sets (see Section 8.5).

8.2 Related Work

In the following, we discuss two areas of related work. First, causal inference and, more specifically, estimating treatment effects. Second, learning policies for treatment recommendation.

8.2.1 Prediction-Focused Learning: Effect Estimation

Understanding the impact of an action on an instance or individual is crucial in a variety of domains where personalized decision-making is valuable, such as marketing, healthcare, or education. Central to this is causal inference:

using data to draw conclusions regarding causal relationships. Especially relevant to our work is treatment (or causal) effect estimation, where the aim is to learn how some treatment, action, or intervention will affect an instance’s outcome of interest. For a comprehensive review, we refer to Zhang et al. [345]. In the context of marketing, using machine learning (ML) for treatment effect estimation is generally referred to as uplift modeling, as discussed in several surveys [173], [345], [346].

Specialized methods have been proposed for estimating treatment effects. First, general strategies exist for learning these effects. Causal metalearners are modeling frameworks for effect estimation, compatible with various ML algorithms [208], [209], [347]. Relatedly, response transformation approaches transform an instance’s outcome so that it can be modeled using a standard classifier [173], [348]. Second, ML algorithms have been adapted for effect estimation, such as decision trees [206] or random forests [349], [350].

While uplift modeling has traditionally focused on optimizing conversion, practitioners often seek to optimize other metrics related to their business and operational context. Recent work explores cost-sensitive or profit-driven uplift modeling, where the aim is to estimate and maximize profit and cost resulting from targeting policies [175], [198], [351], [352]. For example, the Incremental Profit per Conversion (IPC) has been proposed as a response transformation approach for incremental profit [353].

All work in this category aims to *estimate the effect* of a treatment (e.g., a customer’s incremental conversion probability as a result of receiving a marketing incentive). As discussed in the introduction, these estimates can be used to design a treatment allocation policy (e.g., by targeting customers with a large estimated effect). In practice, operational constraints (such as budget limitations) may call for more complex optimization procedures [338], [354]–[356]. The resulting treatment allocation problem requires solving a constrained optimization problem using the effect estimates as input.

8.2.2 Decision-Focused Learning: Effect Ranking

As argued before, the prediction-focused approach may suffer from a misalignment between prediction and optimization, leading to suboptimal treatment decisions. Additionally, empirical risk minimization only guarantees model generalization for the specific objective that was optimized [344]. Decision-focused learning aims to align the two phases by using an integrated approach and directly optimizing a predictive model for the final optimization problem. Related to this insight, a handful of methods have recently been proposed for treatment recommendation that aim to learn a ranking of instances in terms of their treatment effect. Although these methods were

Table 8.3: *Literature Table*. We categorize related work on effect ranking by differentiating between different (1) ranking approaches (point-, pair-, and listwise) and (2) metalearners.

Ref.	Objective			Metalearners					
	Point	Pair	List	Z	S	T	X	DR	R
[362]	✓	×	×	×	✓	×	×	×	×
[363]	✓	×	×	×	✓	×	×	×	✓
[95]	✓	×	✓	✓	×	×	×	×	×
[344]	✓	✓	×	×	✓	×	×	×	×
[352]	✓	×	×	×	✓	✓	×	×	×
[364]	✓	×	×	×	✓	×	×	×	×
[365]	✓	×	×	×	✓	×	×	×	×
Ours	✓	✓	✓	✓	✓	✓	✓	✓	✓

originally proposed in the context of uplift modeling—framing targeted marketing as a ranking problem—they are more generally applicable. Finally, it has been noted that any score—even non-causal estimands—can be used to prioritize instances for treatment, as long as it is a good proxy for the treatment effect’s magnitude [357], [358].

Table 8.3 highlights related methods for ranking effects, describing the metalearners and objectives that were used, following the literature on learning to rank [359], [360]. The first approach, *pointwise* ranking, relies on an estimate of the treatment effect and corresponds to prediction-focused learning. The second approach, *pairwise* ranking, aims to predict the relative ranking between instances over all pairs of instances. The final approach, *listwise* ranking, optimizes the ranking across all instances in the ranking simultaneously. In the literature on learning to rank, pairwise and listwise ranking approaches have surpassed pointwise approaches, with listwise methods typically performing best [360], [361].

Most existing approaches for ranking effects rely on alternative objective functions that integrate prediction and optimization using Lagrangian duality [362]–[365] or gradient estimation techniques [366]. Alternatively, the causal profit ranker [352] ranks instances in a post-processing stage using pointwise estimates of the expected conversion. More advanced pairwise [344] and listwise [95] learning to rank have also been explored in this context. Conversely, our work explores pointwise, pairwise, and listwise objectives, as well as a wide variety of metalearners. Finally, [367] similarly propose alternative objective functions inspired by learning to rank, but focus on representation learning instead of metalearners, and combine pointwise and listwise losses in a multi-objective framework.

General methodologies have been proposed for learning a treatment policy, directly mapping an instance’s characteristics to a recommended treatment [171], [368]–[372]. In the context of our work, these can be seen as pointwise approaches, as they do not consider the ranking structure of the optimization task.

8.3 Problem Formulation

In this work, we aim to learn a treatment policy that prioritizes instances for treatment to maximize the aggregate effect. As opposed to the existing work, which assumes a problem formulation implicitly, we explicitly formalize our problem setting. In doing so, we reveal the underlying assumptions required for our approach.

8.3.1 Notation and Optimization Problem

Let an instance (e.g., a customer) be described by a tuple (x_i, t_i, y_i) , representing covariates $X \subset \mathbb{R}^d$, an administered treatment $t \in \{0, 1\}$, and the outcome to be optimized $Y \subset \mathbb{R}$. We denote the potential outcome Y associated with a treatment t as $Y(t)$ and an instance i ’s treatment effect as $\tau_i = Y_i(1) - Y_i(0)$ (e.g., a customer’s incremental conversion probability resulting from receiving a discount). We aim to learn a policy π that assigns treatments to instances and maximizes the overall treatment effect, while respecting possible operational constraints. At test time, we assume n instances can be treated, subject to a treatment budget B with $B \leq n$. This yields the following optimization problem:

$$\begin{aligned} \max_{\mathbf{t}_i} \quad & \sum_{i=1}^n \tau_i(t_i) \\ \text{s.t.} \quad & \sum_{i=1}^n t_i \leq B \\ & t_i \in \{0, 1\} \quad \forall i \in \{1, \dots, n\} \end{aligned}$$

At test time, treatment effects τ are unknown and need to be estimated. To this end, we assume access to a historical data set $\mathcal{D} = \{x_i, t_i, y_i(t_i)\}_{i=1}^n$ describing past treatment decisions and the resulting outcomes. These data can be used to estimate the conditional average treatment effect (CATE): $\hat{\tau} = \mathbb{E}(Y(1) - Y(0) \mid X = x)$.

8.3.2 Assumptions

We make several assumptions regarding the causal structure of the data and the operational constraint. To estimate the causal effect from historical data, we require the standard assumptions for identifiability in causal inference [163] (see Appendix F.1 for a more extensive discussion). Additionally, we more formally define the operational constraint. Although we assume that not all instances can be treated and, thus, instance prioritization is required, we assume that the exact budget is not known to the decision-maker a priori. More formally, we state there is no information regarding the budget B a priori:

Assumption 13 (Operational constraint). *We assume the exact budget is unknown, but the expectation is uniformly distributed among $\{1, \dots, n\}$: $B \sim \mathcal{U}(1, n)$.*

We discuss how alternative assumptions regarding the budget would affect our proposed solution below.

8.3.3 Evaluating a Treatment Policy

We assess the quality of a proposed policy using the Qini curve, illustrated in Figure 8.1. This curve shows the cumulative total effect of a policy for a number of treated instances [373], [374]. Given that we assume no information regarding the budget, we measure a policy’s overall quality using the area under the Qini curve (AUQC), quantifying the total cumulative effect over the entire ranking. Formally, we define the (hypothetical) AUQC as

$$\text{AUQC} = \sum_{k=1}^n \sum_{i=1}^k \tau_i \tag{8.1}$$

with τ_i the effect of the instance at position i in the ranking. The normalized AUQC is obtained by comparing it with the expected AUQC of a random ranking and AUQC of a perfect ranking. Typically, the normalized AUQC ranges between zero and one $\in [0, 1]$, though a worse than random policy with $\text{AUQC} < 0$ is also possible. Because effects τ are not observed in reality, Qini curves need to be estimated from data on past treatment allocations [95], [374].

8.4 Methodology

Given the problem setup described above, we now present our proposed methodology, which essentially learns a ranking (or sorting) of each instance’s treatment effect. The optimality of this solution can be seen as

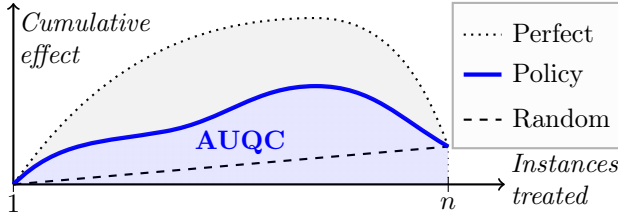


Figure 8.1: *Evaluating a Treatment Allocation Policy.* We compare targeting policies using a Qini curve, depicting the cumulative total effect of a policy for a number of treated instances, summarized by the area under the Qini curve (AUQC).

follows. If only one instance can be treated (i.e., $B = 1$), the optimal solution is to assign the treatment to the instance with the largest treatment effect τ_i . Given an unknown budget and uniform expectation regarding this budget, the optimal solution is then to rank all instances by their treatment effect τ_i and assign treatments to the top instances until the budget runs out. Therefore, our goal is to predict an optimal ordering or assignment policy $\pi \in \Pi_n$ that permutes the test instances $\{1, \dots, n\}$ to the optimal ordering based on descending treatment effects τ .

As previously discussed, most existing approaches first estimate the effects $\hat{\tau}$ and then rank these estimates. However, as discussed above and in Table 8.2, this approach has two drawbacks. First, the estimator’s objective is not aligned with the optimization task, possibly resulting in suboptimal decisions [85]. Second, the resulting model is only guaranteed to generalize for the predictive objective that was used [344]. These issues motivate us to directly learn a ranking policy π based on instance characteristics X , which requires addressing two challenges. First, to find an objective that optimizes a ranking of instances instead of a pointwise estimate (see Section 8.4.1). Second, the ranking needs to be based on the treatment effect τ , which is not observed. Therefore, we extend metalearners for effect estimation to ranking (see Section 8.4.2).

8.4.1 Optimizing a Ranking Objective

In this section, we explore approaches for optimizing a ranking. We discuss pointwise, pairwise, and listwise approaches. We propose a listwise objective that optimizes the policy’s AUQC directly. Additionally, we propose a sampling strategy to improve the efficiency of our proposed ranking objectives.

Ranking objectives

We describe three objectives for learning a ranking policy π that can be used by ML algorithms.

Pointwise The first approach, used by most existing work, is to learn a pointwise estimate of the effect. As the treatment effect itself is never observed, this corresponds to either learning the observed outcome $y(t)$ or a transformed outcome, depending on the metalearner used (see Section 8.4.2). In this work, we use the mean squared error as a pointwise objective to learn the estimand:

$$\mathcal{L}_{\text{Point}}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2. \quad (8.2)$$

While other objectives are possible [e.g. 364], pointwise approaches by definition ignore the instance ranking resulting from the point estimates. This motivates our exploration of alternative objectives.

Pairwise The first ranking approach is the pairwise approach. The idea is to predict, for each pair of instances, how both instances are ranked relative to each other. If all pairs are ranked correctly, the overall ranking will also be correct. We build upon the approach proposed for RankNet [375]. Before we define the pairwise objective, we define the pairwise outcome $y_{i,j}$ that specifies whether instance i or j should be ranked higher, for each pair of instances i and j :

$$y_{i,j} = \begin{cases} 1 & \text{if } y_i \geq y_j \\ 0 & \text{if } y_i < y_j. \end{cases}$$

Similarly, we define a smooth pairwise prediction $\hat{y}_{i,j}$, combining two instances' predictions \hat{y}_i and \hat{y}_j as follows:

$$\hat{y}_{i,j} = \frac{1}{1 + \exp(-\sigma(\hat{y}_i - \hat{y}_j))}, \quad (8.3)$$

where the sigmoid parameter σ controls the smoothness of the comparison. In the extreme $\sigma = \infty$, this becomes a step function. This way, we define pairwise ranking as a binary classification task with the pairwise cross-entropy loss defined as:

$$\mathcal{L}_{\text{Pair}}(y, \hat{y}) = \sum_{i=1}^n \sum_{j=1}^n -y_{i,j} \log(\hat{y}_{i,j}) - (1 - y_{i,j}) \log(1 - \hat{y}_{i,j}). \quad (8.4)$$

Other pairwise objectives exist [e.g. 344] which can be applied to our approach, though we consider this outside the scope of this work.

Listwise A drawback of the pairwise approach is that it overlooks the relative importance of correctly classifying one pair on the listwise ranking quality. To address this issue, the LambdaMART objective [376] adds a weight $\text{NDCG}_{i,j}$ to the pairwise objective, reflecting the increase in normalized discounted cumulative gain (NDCG, see below) achieved by swapping that pair:

$$\mathcal{L}_{\text{List}}(y, \hat{y}) = \sum_{i=1}^n \sum_{j=1}^n \left(-y_{i,j} \log(\hat{y}_{i,j}) - (1 - y_{i,j}) \log(1 - (\hat{y}_{i,j})) \right) \Delta \text{NDCG}_{i,j}. \quad (8.5)$$

The AUQC as a specific instance of the NDCG

The normalized discounted cumulative gain (NDCG) is a class of metrics measuring the quality of a ranking [97]. Formally, we define a ranking π , with π_i representing the i 'th instance in the ranking. An instance's gain g_i represents its value independent of its position in the ranking (e.g., the treatment effect τ). The NDCG decreases the gain for lower ranks, reflecting their decreasing importance, by applying a discount function $d(i)$ to the instance's gain g_i . The discounted cumulative gain (DCG) is the sum of all discounted gains over the ranking $\text{DCG} = \sum_{i=1}^n d(i)g_{\pi_i}$. Finally, the normalized discounted cumulative gain (NDCG) is obtained by comparing the DCG with the perfect ranking's DCG to get a value between zero and one.

We propose a specific instantiation of the NDCG^1 that matches the AUQC. More specifically, we define an instance's gain g_i as its treatment effect τ . The discount function is a linearly decreasing function: for rank i , the discount equals $(n - i + 1)^2$. In this specification, we can show that the listwise objective that optimizes the NDCG allows us to learn an optimal ranking policy π that optimizes the metric of interest, the AUQC (generalizing [95, Section 3.2.2] from the Z-Learner to any model that estimates τ):

Proof. The area under the Qini curve (AUQC) is an instantiation of the discounted cumulative gain (NDCG):

$$\text{AUQC} = \sum_{k=1}^n \sum_{i=1}^k \tau_{\pi_i} = \sum_{i=1}^n \sum_{k=i}^n \tau_{\pi_i}$$

¹The standard formulation of the NDCG is $\sum_{i=1}^n$

²Technically, we do not discount lower ranked instances ($d(i) \leq 1$), but rather promote higher ranked instances ($d(i) \geq 1$) [see 95, eq. 25].

$$= \sum_{i=1}^n \tau_{\pi_i} \sum_{k=i}^n 1 = \sum_{i=1}^n \tau_{\pi_i} (n - i + 1) = \text{DCG}$$

for the DCG with gain $g_{\pi_i} = \tau_{\pi_i}$ and $d(i) = n - i + 1$. Both the AUQC and DCG can similarly be normalized by the ideal ranking, showing that the (normalized) AUQC equals the NDCG. \square

Given that the listwise objective described above can be shown to optimize the NDCG [376], [377], this result proves that our proposed objective can be used to directly optimize the metric of interest: the AUQC. There is one remaining challenge: we do not know the instance’s treatment effect τ and require a valid estimator $\hat{\tau} \rightarrow \tau$. We will tackle this part in Section 8.4.2.

Efficiently scaling to large-scale data sets

Moving from pointwise to pairwise or listwise optimization requires addressing a challenge regarding computational efficiency. Optimizing over pairs of instances results in an increase of the algorithm’s time complexity from $O(n)$ to $O(n^2)$. This complexity is not compatible with the large data sets commonly encountered in applications such as marketing or e-commerce [356].

To address this challenge, we propose an efficient sampling procedure that finds a stochastic estimate of the gradient. Intuitively, instead of calculating the gradient based on all possible pairs, we sample k pairs per instance:

$$\mathcal{L}_{\text{Pair}}(y, \hat{y}) = \sum_{i=1}^n \sum_{j \in \mathcal{J}} -y_{i,j} \log(\hat{y}_{i,j}) - (1 - y_{i,j}) \log(1 - (\hat{y}_{i,j}))$$

with $\mathcal{J} \sim (\mathcal{U}_{[1, \dots, n]})^k$, (8.6)

and equivalently for $\mathcal{L}_{\text{List}}$. This again makes the procedure scale linearly in the number of instances with complexity $O(kn)$. We observe good results for $k = 1$, effectively obtaining the same computational complexity as the pointwise objective. We present a sensitivity analysis for the number of samples k below. We opt for this sampling procedure for its simplicity, although more advanced sampling schemes are possible [see e.g. 361].

8.4.2 Ranking Metalearners

One challenge when learning a treatment assignment policy π is that decisions need to be made based on the treatment effect τ . Indeed, the optimization of the AUQC presented above requires a model to predict the treatment effect $\hat{\tau}$. Predicting a treatment effect τ is a challenge. We never observe the treatment effect itself, but only one potential outcome $y(t)$ for

each instance, i.e., the outcome when targeted or not targeted—also called the fundamental problem of causal inference [182]. In other words, we never actually observe what we are trying to estimate and optimize over.

To tackle this challenge, we implement the objective functions introduced above for different causal metalearners—general strategies for using any ML method for treatment effect estimation. Whereas metalearners have originally been proposed for effect estimation, we propose adaptations for effect ranking below. In practice, this adaptation consists of integrating ranking (i.e., pairwise or listwise) objectives in each training procedure—instead of the traditional pointwise (regression or classification) objectives. In this section, we describe each metalearner and introduce its ranking equivalent. We focus on several established metalearners, but the extension to other metalearners could be done using a similar approach. While our optimization of the AUQC requires metalearners that directly predict the treatment effect τ , we also discuss adaptations of other metalearners—specifically, the S- and T-Learner.

Z-Learner The first metalearner estimates a transformation z of the outcome y —also called the class transformation approach [353], [378], [379]—adjusting the outcome based on the instance’s propensity score³ $\hat{e}(x) = P(T = 1|X = x)$ and the observed treatment:

$$z_i = \begin{cases} y_i/\hat{e}_i & \text{if } t_i = 1 \\ -y_i/(1 - \hat{e}_i) & \text{if } t_i = 0 \end{cases}$$

The Z-Learner estimates the treatment effect using this outcome:

$$f_Z(x) = \mathbb{E}(Z|X) \quad \text{with} \quad \hat{\tau} = f_Z(x)$$

Instead of training this final model $f_Z(x)$ with a pointwise objective, we propose to optimize it using a pairwise or listwise objective.

S-Learner The S-Learner estimates a single model, which takes the treatment as a (regular) feature:

$$f_S(x, t) = \mathbb{E}(Y|X = x, T = t) \quad \text{with} \quad \hat{\tau} = f_S(x, 1) - f_S(x, 0)$$

Again, the model $f_S(x, t)$ can be trained with either a pointwise, pairwise, or listwise objective. For the ranking objectives, the estimated treatment effect τ corresponds to the increase in the ranking score resulting from receiving

³Given that we assume data comes from a randomized trial, we can estimate the propensity score by the proportion of treated instances without fitting a model.

the treatment. Importantly, however, because this metalearner does not directly optimize for τ , the theoretical results of the previous section do not apply. Nevertheless, although there is no guarantee that the listwise objective optimizes the AUQC for the S-Learner, the difference in ranking scores could provide a good heuristic for the treatment effect.

T-Learner The T-Learner trains two models: one model for each treatment group— f_{T_1} for the treatment ($T = 1$) and f_{T_0} for the control group ($T = 0$)—trained as follows:

$$f_{T_1}(x) = \mathbb{E}(Y|X = x, T = 1), \quad f_{T_0}(x) = \mathbb{E}(Y|X = x, T = 0).$$

When combined, these can estimate the treatment effect as follows:

$$\hat{\tau} = f_{T_1}(x) - f_{T_0}(x).$$

We propose to train both models $f_{T_1}(x)$ and $f_{T_0}(x)$ with a pairwise or listwise objective, instead of the traditional pointwise objective. This corresponds to a separate optimization of the AUQC of the treatment ($f_{T_1}(x)$) and control ($f_{T_0}(x)$) groups (based on the outcome y instead of the effect τ). An instance's difference in ranking scores between both groups is used as a proxy for its treatment effect. Similarly to the S-Learner, the theoretical results from the previous section do not apply. Nevertheless, this separate optimization could be a good heuristic for the AUQC [see also 173, eq. 26].

X-Learner The X-Learner first estimates an initial treatment effect by imputing the counterfactual potential outcome using a T-Learner model as follows [208]:

$$D_i^1 = y_i - f_{T_0}(x_i) \text{ if } t_i = 1, \quad D_i^0 = f_{T_1}(x_i) - y_i \text{ if } t_i = 0.$$

The final two models are then trained on the imputed effects:

$$\hat{\tau}^0 = f_X^0(x) = \mathbb{E}(D_i^0|X = x), \quad \hat{\tau}^1 = f_X^1(x) = \mathbb{E}(D_i^1|X = x).$$

To obtain the final predicted effect $\hat{\tau}_i$, we combine these two models:

$$\text{with } \hat{\tau} = g(x)f_X^0(x) + (1 - g(x))f_X^1(x).$$

For the weighting function $g(x)$, we use the estimated propensity score $\hat{e}(x) = P(T|X = x)$ following [208]. Compared to the pointwise variant, we propose to train the final models $f_X^0(x)$ and $f_X^1(x)$ with pairwise or listwise ranking objectives, with the initial models still trained using a pointwise objective. The final ranking score $\hat{\tau}$ is a linear combination of two ranking scores [see 376, note 7.1].

DR-Learner The Doubly Robust or DR-Learner [380], [381] also relies on a final model estimating a pseudo-outcome. In this case, the first stage is based on pointwise estimates from a T-Learner and a propensity model $\hat{e}(x) = P(T|X = x)$. These estimates are combined to create a pseudo-outcome ϕ as follows:

$$\phi_i = \frac{t_i - \hat{e}(x_i)}{\hat{e}(x_i)(1 - \hat{e}(x_i))} (y_i - f_{\tau_{t_i}}(x_i)) + f_{\tau_1}(x_i) - f_{\tau_0}(x_i).$$

This pseudo-outcome is then used to learn a final model $\hat{\tau} = f_{\text{DR}}(\phi|x)$, which can be learned using a point-, pair-, or listwise objective.

R-Learner For the R-Learner [382], we first fit an outcome model $\hat{m}(x) = \mathbb{E}(Y|X = x)$ and propensity model $\hat{e}(x) = P(T|X = x)$. These can then be used to minimize the R-Loss, based on Robinson’s decomposition [383], which can be seen as a weighted MSE:

$$\begin{aligned} \mathcal{L}_{\text{MSE}}^R(y, \hat{y}) &= \frac{1}{n} \sum_{i=1}^n ((y_i - \hat{m}(x_i)) - (t_i - \hat{e}(x_i)) \tau(x_i))^2 \\ &= \frac{1}{n} \sum_{i=1}^n \frac{1}{(t_i - \hat{e}(x_i))^2} \left(\left(\frac{y_i - \hat{m}(x_i)}{t_i - \hat{e}(x_i)} \right) - \tau(x_i) \right)^2. \end{aligned}$$

Instead of this pointwise objective, we propose to use a weighted pair- or listwise objective based on the same weights and labels.

For simplicity, we do not use out-of-fold estimates for any of the intermediary models for any of the metalearners, but rather train all models on the same train set.

8.5 Empirical Results

This section presents the empirical results, comparing our proposed (pairwise and listwise) *effect ranking* metalearners with traditional (pointwise) *effect estimation* metalearners. Our experiments aim to answer three research questions. (*RQ1*) What is the treatment recommendation quality resulting from the different methods, as measured in AUQC? (*RQ2*) What are the performance trade-offs of the pointwise, pairwise, and listwise objectives, in terms of MSE (i.e., pointwise accuracy), Kendall τ (i.e., pairwise rank correlation), and AUQC (i.e., listwise ranking quality)? (*RQ3*) How sensitive are our proposed methods to key hyperparameters? This section presents the setup of our experimental evaluation and the empirical results. Upon publication, all code will be made publicly available.

8.5.1 Data and Benchmarks

To evaluate the performance of the proposed approaches, we use a total of four data sets based on randomized trials: 1) a synthetic dataset; 2) Criteo [341]; 3) Hillstrom (Male and Female) [384]; 4) a proprietary data set from a promotion campaign at a global online travel agency.

We first simulate a **Synthetic** data set. Simulated data allows for a more comprehensive evaluation than real data, as we know the treatment effect for test instances. The data generating process is inspired by an e-commerce setting and similar to existing work [353]. First, we generate customer characteristics as follows: $X \sim \mathcal{N}(0, 1)^d$. Then, we generate a sale (or conversion) probability S based on these characteristics and random coefficients $U_s \sim \mathcal{U}(-1, 1)^d$ as $S = \frac{1}{1 + \exp(-\sum_d U_s X + \epsilon_s)}$, where $\epsilon_s \sim \mathcal{N}(0, 0.1)$. Similarly, we generate a potential revenue R using random coefficients $U_r \sim \mathcal{U}(-1, 1)^d$ as $R = 1 + |\sum_d U_r X| + \epsilon_r$, where $\epsilon_r \sim \mathcal{N}(0, 0.1)$. The cost of the treatment C (the marketing incentive or discount) is defined as 10% of the revenue: $C = 0.1R$. The observed outcome Y , the net revenue generated for that customer, is the revenue for that customer minus the treatment cost—simulated as follows:

$$y_i = \begin{cases} r_i - c_i & \text{if } t_i = 1 \text{ and } s_i = 1 \\ -c_i & \text{if } t_i = 1 \text{ and } s_i = 0 \\ r_i & \text{if } t_i = 0 \text{ and } s_i = 1 \\ 0 & \text{if } t_i = 0 \text{ and } s_i = 0. \end{cases}$$

We generate 10,000 instances with $d = 10$ characteristics.

Next, we also compare with three real-world data sets. The **Criteo** data set [341] is the result of a randomized trial testing whether showing an advertisement increases a customer’s visit or conversion probability. For the outcome, we follow [364] and take the net revenue y as conversion minus visit. To reduce training times, we randomly sample 500,000 instances from this data set.

The **Hillstrom** data set was collected to test whether an e-mail campaign resulted in additional sales. Two treatments were recorded: a **Men’s** and **Women’s** e-mail. Therefore, we split the data in two data sets for both treatments, and use the same control group for both. We calculate each customer’s net revenue as revenue (conversion times spend) minus visit.

Finally, data from a **Promotion** campaign at a large online travel agency was used as a randomized dataset to evaluate the offline performance of different approaches in a real-world setting.

We compare the three objectives and metalearners for gradient boosting

based on LightGBM [385]. We implement our proposed pairwise and listwise objectives as a custom objectives and metrics in LightGBM. For each data set, we run a five-fold cross-validation procedure. We run hyperparameter tuning using 10 random sampling iterations over the following hyperparameters: “num_leaves” $\in [10, 50]$, “learning_rate” $\in [0.01, 0.20]$, “max_depth” $\in [3, 10]$, and “min_data_in_leaf” $\in [10, 30]$. We use 64% of all instances for training, 16% for validation, and 20% for testing.

8.5.2 Comparing Performance for the Different Objectives and Metalearners (RQ1)

For each metalearner, we compare a model trained with only a pointwise objective to our proposed ranking alternatives, based on either pairwise and listwise objectives. We evaluate the quality of each treatment allocation policy by looking at the cumulative treatment effect over the instance ranking, measured using the AUQC presented above. We evaluate the performance for each metalearner and objective over the different data sets (see Figure 8.2 and Appendix F.2.1). Across data sets and metalearners, we observe that listwise metalearners generally result in better treatment prioritization. Over all tested data sets and metalearners, a pointwise objective gives the highest AUQC in only a minority of cases (7/30), while the listwise objective obtains best in class performance in a majority of time (16/30) and the pairwise objective performs similar to the pointwise (7/30). A listwise approach outperforms a pointwise one in a majority of cases (20/30). Only for the `Promotion` data, the pointwise objective performs relatively well.

Interestingly, we observe differences across metalearners in terms of which objective gives the best results. The listwise objective seems favorable for some metalearners (Z-, X-, and R-Learner), while the pairwise objective seems preferable for the S-Learner and the pointwise objective seems best for the DR-Learner. Generally, we also observe that the choice of metalearner is at least equally important as the choice of objective. This finding stresses the importance of testing different metalearners—illustrating the value of our contribution.

When learning a model using ranking objectives, the ranking scores are not properly calibrated. This represents a possible challenge for metalearners that do not directly learn the treatment effect, i.e., the S- and T-Learner. For these metalearners, an instance’s predicted ranking score does not necessarily correspond to its potential outcome. Rather, if one instance’s outcome is larger than another instance’s ($y_i > y_j$), its predicted score will be larger ($\hat{y}_i > \hat{y}_j$). Although the ranking might hold for both potential outcomes, there are no guarantees for the rankings of the estimated treatment effects

derived from these ranking scores (τ_i and τ_j). Because ranking scores are not calibrated, arithmetic operations of the scores (as used by these two metalearners) may not be meaningful. Nevertheless, the ranking versions of the S- and T-Learner perform relatively well in practice, illustrating that they may provide a meaningful heuristic. While calibration methods for ranking models exists, we leave this extension for future work (see conclusion).

8.5.3 Analyzing Alternative Metrics (RQ2) and Design Choices (RQ3)

This section aims to provide a deeper understanding of our proposed approach. To this end, we highlight additional results using the **Synthetic** data, which allows for a more comprehensive analysis as we know the ground truth treatment effects for the test instances.

To answer (RQ2) regarding performance trade-offs of the different objectives and metalearners, we can present additional metrics to allow for a more holistic evaluation (see Figure 8.3 and Appendix F.2.2). Our main metric of interest remains the AUQC: a listwise metric of ranking quality. Additionally, we present two other metrics: a pointwise error metric (MSE) and a pairwise rank correlation coefficient (Kendall τ). First, we observe that ranking objectives give far worse performance in terms of MSE (Appendix F.2.2). In other words, the ranking score does not accurately reflect the size of the treatment effect. In relation to this finding, Figure 8.3 shows that the MSE is not a good predictor of performance in terms of AUQC. Conversely, Kendall τ is a good predictor for AUQC for all models ($\rho > 0$), particularly for the ranking models ($\rho > 0.9$). This finding underscores the importance of ranking metrics for evaluating decision quality and highlights the irrelevance of optimizing pointwise error for treatment prioritization.

Finally, to answer (RQ3), we analyze several design choices of our proposed ranking metalearners in Figure 8.4 and Appendix F.2.3. We observe that the default setting used in the experiments above (sampling iterations $k = 1$, sigmoid $\sigma = 1$, normalizing ranking scores) generally performs well across ranking objectives and metalearners. Interestingly, we observe little performance benefits when training with more sampling iterations. This finding shows that our sampled objective can estimate the ranking objective accurately. Only for the S- and T-Learner without normalization, we observe better performance for a higher k . Additionally, the stochasticity of our sampled objectives may even provide a form of regularization [386].

8.6 Conclusion

This work addressed the problem of optimally prioritizing instances for treatment, an important problem for many applications where not all instances can receive a treatment. Existing approaches typically tackle this problem in a prediction-focused approach by first obtaining a pointwise *effect estimate* for each instance’s treatment effect, and then ranking instances based on these estimates. Conversely, we explore an alternative, decision-focused approach: using objectives that learn to *rank the treatment effects*, we aim to optimize the quality of the resulting treatment policy directly. Building on the literature on learning to rank, we propose pairwise and listwise ranking objectives and show that our proposed listwise objective directly optimizes the policy’s AUQC. Moreover, we propose different ranking metalearners by integrating these ranking objectives in the construction of each metalearner. Empirical results show that our proposed *effect ranking* approach can outperform a pointwise, *effect estimation* approach. In conclusion, our proposed ranking metalearners offer a valuable tool for applications where instances need to be prioritized for treatment.

Our work opens up several exciting directions for future work. First, by building upon advances in learning to rank, such as more advanced listwise objectives [e.g., 361] or calibration of ranking scores [e.g., 387]–[389]. Alternatively, we envision extensions of our approach to more complicated settings. For example, we could consider more advanced operational constraints, such as top k targeting or uncertain treatment capacities [see e.g., 390]; multiple, continuous, or even multidimensional treatments; or more complex objectives (e.g., incremental return on investment). Additionally, it would be insightful to analyze how our proposed ranking metalearners perform when learning from confounded observational data with non-random treatment assignments. Finally, while theoretical results regarding convergence or error bounds are out of the scope of this work, we believe that extending the results obtained for the effect estimation models [e.g., 381], [382] to effect ranking models is a fruitful area of future work.

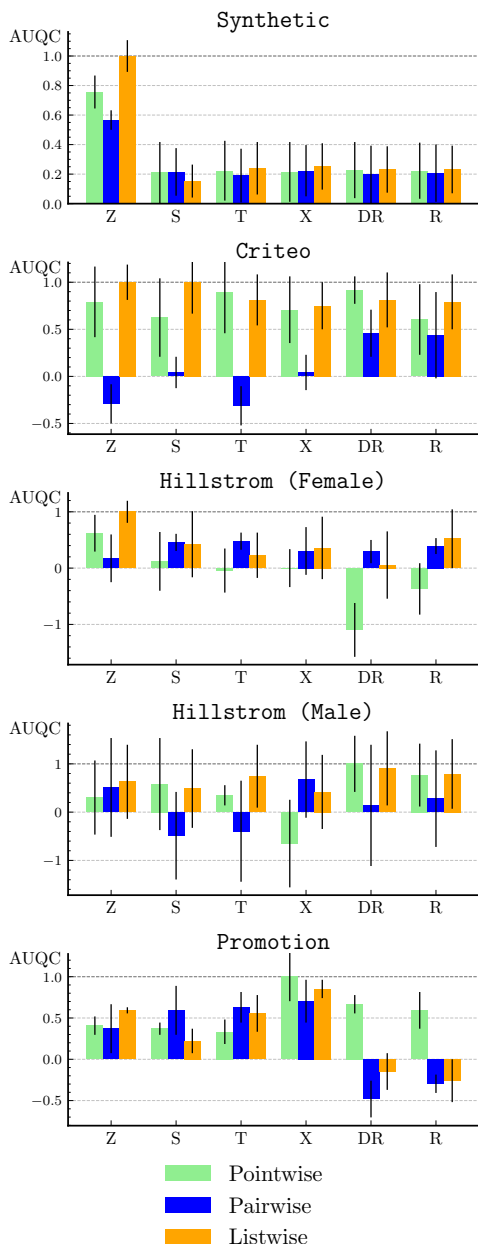


Figure 8.2: *Ranking Quality for Different Objectives and Metalearners.* For each metalearner, we compare a point-, pair-, and listwise version. We show the AUQC \pm one standard error, scaled to have the best result = 1, for five different data sets.

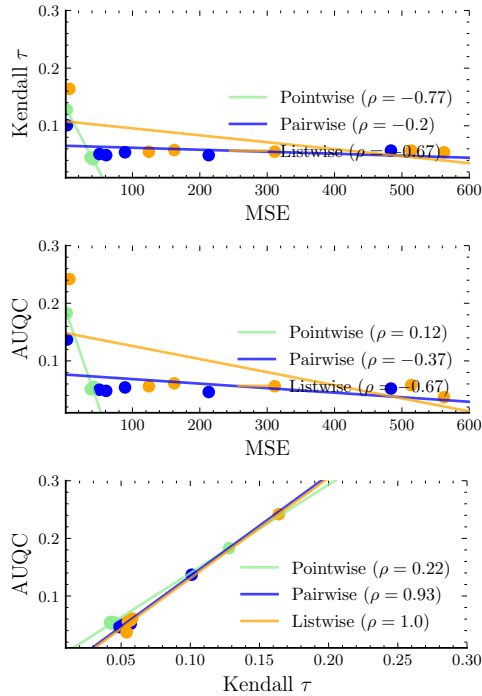


Figure 8.3: *Analyzing Performance Trade-offs on Synthetic Data.* We compare the three different objectives (point-, pair-, and listwise) across meta-learners. Using the `Synthetic` data set, we compare performance in terms of MSE (i.e., pointwise accuracy), Kendall τ (i.e., pairwise rank correlation), and AUQC (i.e., listwise decision quality). For each, we show the correlation ρ .

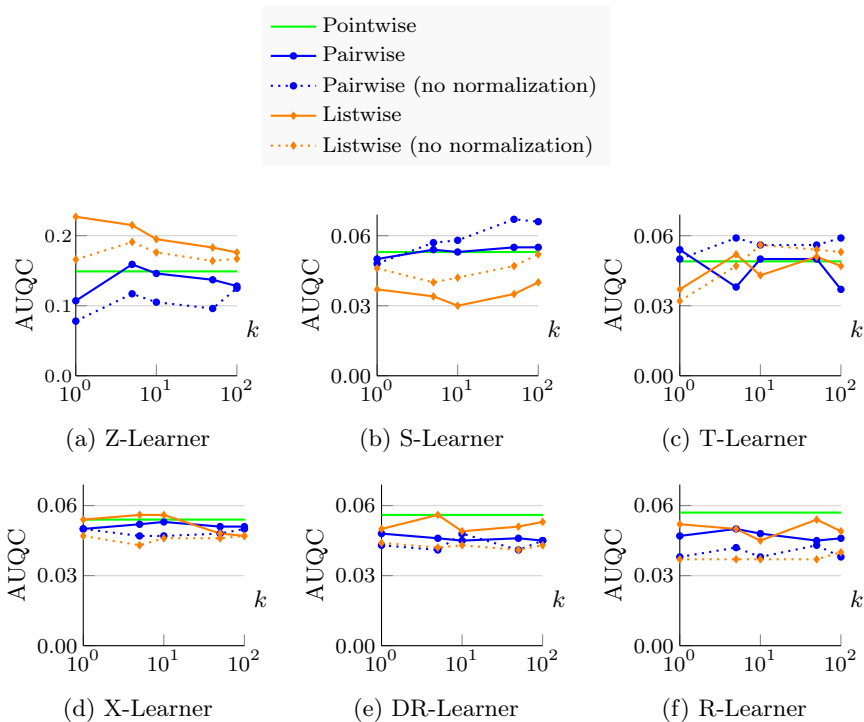


Figure 8.4: *What Is the Effect of the Number of Sampling Iterations k ?* We show performance in terms of AUQC for the different metalearners on the Synthetic data set. We fix the sigmoid parameter $\sigma = 1$ and train with default hyperparameters.

Part V

EPILOGUE



9

CONCLUSION

Recent advances in machine learning (ML) present exciting opportunities for extracting information from data. This dissertation looks at the potential of using ML not just as a predictive tool, but a prescriptive one: by using data on past operational decisions to optimize future decision-making. To maximize the impact of ML in this context, this dissertation has argued that we can advance ML in two ways. First, ML models need to be *decision-focused*—they have to be aligned with the operational context, which can be formulated as a constrained optimization problem. Second, to isolate the *causal impact* of potential decisions and ensure that interventions achieve the desired outcome, ML has to incorporate causal reasoning. By fulfilling these requirements, ML can leverage insights from operations research and causal inference and achieve its fullest potential as a tool for optimizing decision-making. This dissertation presented several contributions that bridging the gaps between these fields.

This section provides a wider context for the presented contributions, in terms of their significance and relevance towards practical applications. Additionally, it will give an overview of the limitations of the current work and outline opportunities for future research.

9.1 Contributions and Managerial Implications

Methodological contributions. This dissertation presented several methodological advances in ML. In Chapter 2, the prediction-optimization framework was used to categorize and analyze cost-sensitive learning strategies. Next, Chapter 3 presented a novel decision-focused method for prioritizing tasks under capacity constraints based on learning to rank. Additionally, this dissertation presented methodological advances in causal ML. TESAR-CDE, presented in Chapter 6, is the first ML method for causal learning in settings with informative sampling. In Chapter 5, NOFLITE was introduced as a novel method for predicting individual treatment effect distributions. AutoCATE, presented in Chapter 7, represents the first AutoML solution tailored to treatment effect estimation, based on automated protocols for evaluation, estimation, and ensembling of causal ML pipelines. Finally, Chapter 8 presented ranking metalearners: a casual, decision-focused ML solution for prioritizing treatments under capacity constraints, bridging the gap between decision-focused learning and causal inference. Collectively, these contributions push the boundaries of how machine learning can be applied to optimize decision-making in complex, real-world scenarios.

Practical contributions. The contributions presented in this dissertation represent significant progress towards solving various real-world problems across a variety of application areas. All contributions were inspired by real-world problems and often resulted from collaborations with industry partners. Chapters 2 and 3 were developed to support fraud detection at BNP Paribas Fortis. Chapter 4 on preventive maintenance was motivated by a real-world case at an anonymous industrial partner, who provided the data to support this work. AutoCATE, presented Chapter 7, explicitly aimed to facilitate more straightforward adoption and implementation of causal ML in real-world settings. The work in Chapter 8 emerged from a collaboration with Booking.com for personalizing their discounts offerings. Additionally, the methods developed in this work are directly applicable to a variety of adjacent applications areas, such as credit scoring or economics. These practical contributions underscore the real-world impact and applicability of the methodologies developed in this dissertation.

Managerial implications. The findings of this dissertation highlight the exciting potential of ML for optimizing decision-making. To fully realize this potential, managers, and data science practitioners have to be mindful of practical considerations and requirements that are crucial for data-driven decision support. First and foremost, suitable data on past decisions needs to be available. Not only are large *amounts* of data typically required for

training modern ML models, causal inference additionally requires the *appropriate information* to isolate the causal effect of the decision from other confounding factors. For decision-focused learning, the operational context needs to be formalized in terms of objectives and constraints in order to align ML models with the decision-making context. To use ML for operational decision-making in practice, data practitioners need to collaborate with domain experts to support their analyses. These experts provide the contextual knowledge required for appropriate data collection and interpretation, while data practitioners offer advanced ML techniques to best learn from this data. As such, supporting decision-making with data requires a coordinated and continued collaboration with different stakeholders in each part of the ML lifecycle, from data collection to model deployment.

9.2 Limitations and Future Work

Despite the exciting recent progress, several challenges and limitations remain to be addressed. This section discusses some key limitations of the presented work and describes potential avenues for future research.

Extending problem formulations and relaxing assumptions. Many of the methods proposed in this work can improve upon conventional methods in certain *settings* and when certain *assumptions* are met. Each chapter’s problem formulation defines the scope of problems that are addressed. An exciting avenue for future work is to extend this scope: first by testing our methods more generally, but also by generalizing them to address related problems. Similarly, we have explicitly laid out the assumptions required for each method where necessary. It would be interesting to assess the sensitivity of our methods to these assumptions and to improve the robustness to violations of these assumptions where possible. In causal inference, there has recently been much progress in this area, such as sensitivity analyses for hidden confounding [391] or methods that can deal with interference [392].

Extended validation of methods. A crucial direction for future work is further practical validation of the proposed methods. In particular, most work on causal inference is benchmarked and validated using semi-synthetic data. Although this practice is standard in this field due to the unavailability of a ground truth in real data, these types of experiments may not capture certain characteristics encountered in the real-world. These complexities can include low signal-to-noise ratios, categorical variables that need to be encoded, or an abundance of irrelevant covariates. Practical validation of causal ML models ideally involves an experimental approach, where differ-

ent policies are implemented and compared in a trial. Although this approach is not straightforward to implement, it is the true test for assessing a method’s practical utility. Finally, the experiments presented in this work often focused on predictive accuracy or profit. Nevertheless, other performance criteria such as interpretability or fairness remain equally essential [393]. The intersections of these fields are fruitful areas for future research.

Causal ML lifecycle. As argued in Chapter 7, more research is needed in overlooked parts of the causal ML lifecycle. AutoCATE aims to contribute here, for example, by including preprocessing steps in the causal ML pipeline. Nevertheless, several questions remain. For example, how should we adapt standard methods for feature selection and preprocessing for causal effect estimation? Additionally, data-centric approaches have only been developed for supervised learning [394], but they could be equally beneficial for causal methods. After deployment, methods need to be developed for dealing with concept drift and for detecting changes in the underlying causal structure. Moreover, better explainability techniques are necessary for auditing and understanding causal ML models [395], [396]. By focusing on all phases of the ML lifecycle, future research can help build more practical, performant and robust causal models that can be implemented in practice.

Extending model capabilities. General advances in ML can enable more capable models in the context of operational decision-making. Currently, we envision at least three dimensions along which models could be improved. First, by extending methods presented for static settings to dynamic, time series settings. Second, by allowing more complex, high-dimensional treatments instead of binary choices or continuous doses. Third, by allowing for more diverse data modalities, such as text (e.g., doctor’s notes), images (e.g., medical scans), or audio (e.g., patient interviews). Given innovations along these three dimensions, the end goal for an operational decision-model should be a model that can integrate new data as it becomes available (e.g., a written report from a doctor’s visit or CT scan) and continuously update its recommended high-dimensional, complex treatment plan.

Towards integrated decision support systems. A final opportunity for ML models is embedding them in more advanced *decision support systems*. In these systems, models could reject making a recommendation [397] or could defer recommendations to an expert when uncertain [398]. Integrating causal and decision-focused ML models in such frameworks could make their implementation more robust and, ultimately, more trustworthy. Alternatively, models could play a more active role in data collection, by probing for information in a chatbot-like setting or actively requesting tar-

geted experiments to reduce uncertainty. Finally, it remains an open question how to best incorporate the available domain knowledge within decision support systems. Not only could the resulting decision support systems provide decision recommendations, they could continuously learn, interact and collaborate with domain experts and the operational environment.

REFERENCES

- [1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.
- [2] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [4] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, *et al.*, “Improving language understanding by generative pre-training,” 2018.
- [5] J. Jumper, R. Evans, A. Pritzel, *et al.*, “Highly accurate protein structure prediction with alphafold,” *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.
- [6] C. Bockel-Rickermann, S. Verboven, T. Verdonck, and W. Verbeke, “A causal perspective on loan pricing: Investigating the impacts of selection bias on identifying bid-response functions,” *arXiv preprint arXiv:2309.03730*, 2023.
- [7] P. W. Holland, “Causal inference, path analysis and recursive structural equations models,” *ETS Research Report Series*, vol. 1988, no. 1, pp. i–50, 1988.
- [8] T. Vanderschueren, T. Verdonck, B. Baesens, and W. Verbeke, “Predict-then-optimize or predict-and-optimize? an empirical evaluation of cost-sensitive learning strategies,” *Information Sciences*, vol. 594, pp. 400–415, 2022.

- [9] T. Vanderschueren, B. Baesens, T. Verdonck, and W. Verbeke, “A new perspective on classification: Optimally allocating limited resources to uncertain tasks,” *Decision Support Systems*, vol. 179, p. 114 151, 2024.
- [10] T. Vanderschueren, R. Boute, T. Verdonck, B. Baesens, and W. Verbeke, “Optimizing the preventive maintenance frequency with causal machine learning,” *International Journal of Production Economics*, vol. 258, p. 108 798, 2023.
- [11] T. Vanderschueren, J. Berrevoets, and W. Verbeke, “Noflite: Learning to predict individual treatment effect distributions,” *Transactions on Machine Learning Research*, 2023.
- [12] T. Vanderschueren, A. Curth, W. Verbeke, and M. Van Der Schaar, “Accounting for informative sampling when learning to forecast treatment outcomes over time,” in *International Conference on Machine Learning*, PMLR, 2023, pp. 34 855–34 874.
- [13] T. Vanderschueren, W. Verbeke, F. Moraes, and H. M. Proença, “Metalearners for ranking treatment effects,” *arXiv preprint arXiv:2405.02183*, 2024.
- [14] P. Donti, B. Amos, and J. Z. Kolter, “Task-based end-to-end model learning in stochastic optimization,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.
- [15] B. Wilder, B. Dilkina, and M. Tambe, “Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 1658–1665.
- [16] C. W. Granger, “Prediction with a generalized cost of error function,” *Journal of the Operational Research Society*, vol. 20, no. 2, pp. 199–207, 1969.
- [17] C. Elkan, “The foundations of cost-sensitive learning,” in *International joint conference on artificial intelligence*, Lawrence Erlbaum Associates Ltd, vol. 17, 2001, pp. 973–978.
- [18] P. Domingos, “Metacost: A general method for making classifiers cost-sensitive,” in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, 1999, pp. 155–164.
- [19] M. Pazzani, C. Merz, P. Murphy, K. Ali, T. Hume, and C. Brunk, “Reducing misclassification costs,” in *Machine Learning Proceedings 1994*, Elsevier, 1994, pp. 217–225.

-
- [20] P. D. Turney, “Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm,” *Journal of artificial intelligence research*, vol. 2, pp. 369–409, 1994.
- [21] T. Fawcett and F. Provost, “Adaptive fraud detection,” *Data mining and knowledge discovery*, vol. 1, no. 3, pp. 291–316, 1997.
- [22] M. Kukar, I. Kononenko, *et al.*, “Cost-sensitive learning with neural networks.,” in *ECAI*, Citeseer, vol. 15, 1998, pp. 88–94.
- [23] G. K. J. Shawe-Taylor and G. Karakoulas, “Optimizing classifiers for imbalanced training sets,” *Advances in neural information processing systems*, vol. 11, no. 11, p. 253, 1999.
- [24] K. Veropoulos, C. Campbell, N. Cristianini, *et al.*, “Controlling the sensitivity of support vector machines,” in *Proceedings of the international joint conference on AI*, Stockholm, vol. 55, 1999, p. 60.
- [25] C. Drummond and R. C. Holte, “Exploiting the cost (in) sensitivity of decision tree splitting criteria,” in *ICML*, vol. 1, 2000.
- [26] X. Chai, L. Deng, Q. Yang, and C. X. Ling, “Test-cost sensitive naive bayes classification,” in *Fourth IEEE International Conference on Data Mining (ICDM’04)*, IEEE, 2004, pp. 51–58.
- [27] K. M. Ting, “An instance-weighting method to induce cost-sensitive trees,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 3, pp. 659–665, 2002.
- [28] Z.-H. Zhou and X.-Y. Liu, “Training cost-sensitive neural networks with methods addressing the class imbalance problem,” *IEEE Transactions on knowledge and data engineering*, vol. 18, no. 1, pp. 63–77, 2005.
- [29] V. S. Sheng and C. X. Ling, “Thresholding for making classifiers cost-sensitive,” in *AAAI*, vol. 6, 2006, pp. 476–481.
- [30] Y. Sun, M. S. Kamel, A. K. Wong, and Y. Wang, “Cost-sensitive boosting for classification of imbalanced data,” *Pattern Recognition*, vol. 40, no. 12, pp. 3358–3378, 2007.
- [31] N. V. Chawla, D. A. Cieslak, L. O. Hall, and A. Joshi, “Automatically countering imbalance and its empirical relationship to cost,” *Data Mining and Knowledge Discovery*, vol. 17, no. 2, pp. 225–252, 2008.
- [32] J. P. Dmochowski, P. Sajda, and L. C. Parra, “Maximum likelihood in cost-sensitive learning: Model specification, approximations, and upper bounds,” *Journal of Machine Learning Research*, vol. 11, no. 12, 2010.

- [33] Y.-F. Li, J. Kwok, and Z.-H. Zhou, “Cost-sensitive semi-supervised support vector machine,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 24, 2010.
- [34] J. Zheng, “Cost-sensitive boosting neural networks for software defect prediction,” *Expert Systems with Applications*, vol. 37, no. 6, pp. 4537–4543, 2010.
- [35] B. Krawczyk, M. Woźniak, and G. Schaefer, “Cost-sensitive decision tree ensembles for effective imbalanced classification,” *Applied Soft Computing*, vol. 14, pp. 554–562, 2014.
- [36] E. Stripling, S. vanden Broucke, K. Antonio, B. Baesens, and M. Snoeck, “Profit maximizing logistic model for customer churn prediction using genetic algorithms,” *Swarm and Evolutionary Computation*, vol. 40, pp. 116–130, 2018.
- [37] S. Höppner, E. Stripling, B. Baesens, S. vanden Broucke, and T. Verdonck, “Profit driven decision trees for churn prediction,” *European journal of operational research*, vol. 284, no. 3, pp. 920–933, 2020.
- [38] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan, “Adacost: Misclassification cost-sensitive boosting,” in *Icml*, Citeseer, vol. 99, 1999, pp. 97–105.
- [39] B. Zadrozny and C. Elkan, “Learning and making decisions when costs and probabilities are both unknown,” in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001, pp. 204–213.
- [40] B. Zadrozny, J. Langford, and N. Abe, “Cost-sensitive learning by cost-proportionate example weighting,” in *Third IEEE international conference on data mining*, IEEE, 2003, pp. 435–442.
- [41] U. Brefeld, P. Geibel, and F. Wyszotzki, “Support vector machines with example dependent costs,” in *European Conference on Machine Learning*, Springer, 2003, pp. 23–34.
- [42] Y. Sahin, S. Bulkan, and E. Duman, “A cost-sensitive decision tree approach for fraud detection,” *Expert Systems with Applications*, vol. 40, no. 15, pp. 5916–5923, 2013.
- [43] A. C. Bahnsen, D. Aouada, and B. Ottersten, “Example-dependent cost-sensitive logistic regression for credit scoring,” in *2014 13th International Conference on Machine Learning and Applications*, IEEE, 2014, pp. 263–269.
- [44] A. C. Bahnsen, A. Stojanovic, D. Aouada, and B. Ottersten, “Improving credit card fraud detection with calibrated probabilities,” in *Proceedings of the 2014 SIAM International Conference on Data Mining*, SIAM, 2014, pp. 677–685.

-
- [45] A. C. Bahnsen, D. Aouada, and B. Ottersten, “Example-dependent cost-sensitive decision trees,” *Expert Systems with Applications*, vol. 42, no. 19, pp. 6609–6619, 2015.
- [46] S. Höppner, B. Baesens, W. Verbeke, and T. Verdonck, “Instance-dependent cost-sensitive learning for detecting transfer fraud,” *European Journal of Operational Research*, vol. 297, no. 1, pp. 291–300, 2022.
- [47] S. Maldonado, Á. Flores, T. Verbraken, B. Baesens, and R. Weber, “Profit-based feature selection using support vector machines—general framework and an application for customer retention,” *Applied Soft Computing*, vol. 35, pp. 740–748, 2015.
- [48] S. Lessmann, J. Haupt, K. Coussement, and K. W. De Bock, “Targeting customers for profit: An ensemble learning framework to support marketing decision-making,” *Information Sciences*, 2019.
- [49] G. Petrides and W. Verbeke, “Cost-sensitive ensemble learning: A unifying framework,” *Data Mining and Knowledge Discovery*, pp. 1–28, 2021.
- [50] T. Wang, Z. Qin, S. Zhang, and C. Zhang, “Cost-sensitive classification with inadequate labeled data,” *Information Systems*, vol. 37, no. 5, pp. 508–516, 2012.
- [51] X. Chen, C. Gong, and J. Yang, “Cost-sensitive positive and unlabeled learning,” *Information Sciences*, vol. 558, pp. 229–245, 2021.
- [52] N. Nikolaou, N. Edakunni, M. Kull, P. Flach, and G. Brown, “Cost-sensitive boosting algorithms: Do we really need them?” *Machine Learning*, vol. 104, no. 2, pp. 359–384, 2016.
- [53] J. L. Leevy, T. M. Khoshgoftaar, R. A. Bauder, and N. Seliya, “A survey on addressing high-class imbalance in big data,” *Journal of Big Data*, vol. 5, no. 1, pp. 1–30, 2018.
- [54] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 1999.
- [55] S. Lessmann, B. Baesens, H.-V. Seow, and L. C. Thomas, “Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research,” *European Journal of Operational Research*, vol. 247, no. 1, pp. 124–136, 2015.
- [56] A. Niculescu-Mizil and R. Caruana, “Predicting good probabilities with supervised learning,” in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 625–632.

- [57] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 233–240.
- [58] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.
- [59] A. C. Bahnsen, D. Aouada, and B. Ottersten, "A novel cost-sensitive framework for customer churn predictive modeling," *Decision Analytics*, vol. 2, no. 1, pp. 1–15, 2015.
- [60] P. A. Samuelson and W. D. Nordhaus, *Economics*, 19th ed. McGraw-Hill/Irwin, 2010.
- [61] L. Ward Jr, "On the optimal allocation of limited resources," *Operations Research*, vol. 5, no. 6, pp. 815–819, 1957.
- [62] H. Everett III, "Generalized lagrange multiplier method for solving problems of optimum allocation of resources," *Operations research*, vol. 11, no. 3, pp. 399–417, 1963.
- [63] A. Calma, W. Ho, L. Shao, and H. Li, "Operations research: Topics, impact, and trends from 1952–2019," *Operations Research*, vol. 69, no. 5, pp. 1487–1508, 2021.
- [64] B. Baesens, S. Viaene, D. Van den Poel, J. Vanthienen, and G. Dedene, "Bayesian neural network learning for repeat purchase modelling in direct marketing," *European Journal of Operational Research*, vol. 138, no. 1, pp. 191–211, 2002.
- [65] B. Baesens, T. Van Gestel, S. Viaene, M. Stepanova, J. Suykens, and J. Vanthienen, "Benchmarking state-of-the-art classification algorithms for credit scoring," *Journal of the operational research society*, vol. 54, no. 6, pp. 627–635, 2003.
- [66] W. Verbeke, D. Martens, C. Mues, and B. Baesens, "Building comprehensible customer churn prediction models with advanced rule induction techniques," *Expert systems with applications*, vol. 38, no. 3, pp. 2354–2364, 2011.
- [67] W. Verbeke, K. Dejaeger, D. Martens, J. Hur, and B. Baesens, "New insights into churn prediction in the telecommunication sector: A profit driven data mining approach," *European journal of operational research*, vol. 218, no. 1, pp. 211–229, 2012.
- [68] V. Van Vlasselaer, T. Eliassi-Rad, L. Akoglu, M. Snoeck, and B. Baesens, "Gotcha! network-based fraud detection for social security fraud," *Management Science*, vol. 63, no. 9, pp. 3090–3110, 2017.

-
- [69] A. Cerioli, L. Barabesi, A. Cerasa, M. Menegatti, and D. Perrotta, “Newcomb–benford law and the detection of frauds in international trade,” *Proceedings of the National Academy of Sciences*, vol. 116, no. 1, pp. 106–115, 2019.
- [70] R. Burkard, M. Dell’Amico, and S. Martello, *Assignment problems: revised reprint*. SIAM, 2012.
- [71] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus, “On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment,” *Proceedings of the National Academy of Sciences*, vol. 114, no. 3, pp. 462–467, 2017.
- [72] D. Bertsimas, A. Delarue, and S. Martin, “Optimizing schools’ start time and bus routes,” *Proceedings of the National Academy of Sciences*, vol. 116, no. 13, pp. 5943–5948, 2019.
- [73] B. Toktas, J. W. Yen, and Z. B. Zabinsky, “Addressing capacity uncertainty in resource-constrained assignment problems,” *Computers & operations research*, vol. 33, no. 3, pp. 724–745, 2006.
- [74] P. A. Krokhmal and P. M. Pardalos, “Random assignment problems,” *European Journal of Operational Research*, vol. 194, no. 1, pp. 1–17, 2009.
- [75] J. Li, B. Xin, P. M. Pardalos, and J. Chen, “Solving bi-objective uncertain stochastic resource allocation problems by the cvar-based risk measure and decomposition-based multi-objective evolutionary algorithms,” *Annals of Operations Research*, vol. 296, no. 1-2, pp. 639–666, 2021.
- [76] R. Johari, V. Kamble, and Y. Kanoria, “Matching while learning,” *Operations Research*, vol. 69, no. 2, pp. 655–681, 2021.
- [77] A. Lodi and G. Zarpellon, “On learning and branching: A survey,” *Top*, vol. 25, no. 2, pp. 207–236, 2017.
- [78] Y. Bengio, A. Lodi, and A. Prouvost, “Machine learning for combinatorial optimization: A methodological tour d’horizon,” *European Journal of Operational Research*, vol. 290, no. 2, pp. 405–421, 2021.
- [79] P. Donti, B. Amos, and J. Z. Kolter, “Task-based end-to-end model learning in stochastic optimization,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, et al., Eds., vol. 30, Curran Associates, Inc., 2017. [Online]. Available: <https://proceedings.neurips.cc/paper/2017/file/3fc2c60b5782f641f76bcefc39fb2392-Paper.pdf>.
- [80] A. N. Elmachtoub and P. Grigas, “Smart “predict, then optimize”,” *Management Science*, 2021.

- [81] J. Mandi, P. J. Stuckey, T. Guns, *et al.*, “Smart predict-and-optimize for hard combinatorial optimization problems,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 1603–1610.
- [82] J. Kotary, F. Fioretto, P. Van Hentenryck, and B. Wilder, “End-to-end constrained optimization learning: A survey,” *arXiv preprint arXiv:2103.16378*, 2021.
- [83] E. Demirović, P. J. Stuckey, J. Bailey, *et al.*, “An investigation into prediction+optimisation for the knapsack problem,” in *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, Springer, 2019, pp. 241–257.
- [84] E. Demirović, P. J. Stuckey, J. Bailey, *et al.*, “Predict+optimise with ranking objectives: Exhaustively learning linear functions,” in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, International Joint Conferences on Artificial Intelligence, 2019, pp. 1078–1085.
- [85] J. Mandi, J. Kotary, S. Berden, *et al.*, “Decision-focused learning: Foundations, state of the art, benchmark and future opportunities,” *arXiv preprint arXiv:2307.13565*, 2023.
- [86] G. Petrides, D. Moldovan, L. Coenen, T. Guns, and W. Verbeke, “Cost-sensitive learning for profit-driven credit scoring,” *Journal of the Operational Research Society*, pp. 1–13, 2020.
- [87] S. Höppner, B. Baesens, W. Verbeke, and T. Verdonck, “Instance-dependent cost-sensitive learning for detecting transfer fraud,” *European Journal of Operational Research*, vol. 297, no. 1, pp. 291–300, 2022.
- [88] G. Petrides and W. Verbeke, “Cost-sensitive ensemble learning: A unifying framework,” *Data Mining and Knowledge Discovery*, pp. 1–28, 2021.
- [89] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, “Credit card fraud detection: A realistic modeling and a novel learning strategy,” *IEEE transactions on neural networks and learning systems*, vol. 29, no. 8, pp. 3784–3797, 2017.
- [90] I. Bose and X. Chen, “Quantitative models for direct marketing: A review from systems perspective,” *European Journal of Operational Research*, vol. 195, no. 1, pp. 1–16, 2009.
- [91] J. Hadden, A. Tiwari, R. Roy, and D. Ruta, “Computer assisted customer churn management: State-of-the-art and future trends,” *Computers & Operations Research*, vol. 34, no. 10, pp. 2902–2917, 2007.

-
- [92] D. A. Shifman, I. Cohen, K. Huang, X. Xian, and G. Singer, “An adaptive machine learning algorithm for the resource-constrained classification problem,” *Engineering Applications of Artificial Intelligence*, vol. 119, p. 105741, 2023.
- [93] X. Yang, K. Tang, and X. Yao, “A learning-to-rank approach to software defect prediction,” *IEEE Transactions on Reliability*, vol. 64, no. 1, pp. 234–246, 2014.
- [94] L. Coenen, W. Verbeke, and T. Guns, “Machine learning methods for short-term probability of default: A comparison of classification, regression and ranking methods,” *Journal of the Operational Research Society*, pp. 1–16, 2020.
- [95] F. Devriendt, J. Van Belle, T. Guns, and W. Verbeke, “Learning to rank for uplift modeling,” *IEEE Transactions on Knowledge and Data Engineering*, 2020.
- [96] R. McBride, K. Wang, Z. Ren, and W. Li, “Cost-sensitive learning to rank,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4570–4577.
- [97] Y. Wang, L. Wang, Y. Li, D. He, W. Chen, and T.-Y. Liu, “A theoretical analysis of ndcg ranking measures,” in *Proceedings of the 26th annual conference on learning theory (COLT 2013)*, Citeseer, vol. 8, 2013, p. 6.
- [98] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao, “Ranking, boosting, and model adaptation,” Technical report, Microsoft Research, Tech. Rep., 2008.
- [99] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, *et al.*, “Xgboost: Extreme gradient boosting,” *R package version 0.4-2*, vol. 1, no. 4, pp. 1–4, 2015.
- [100] B. R. Gunnarsson, S. Vanden Broucke, B. Baesens, M. Óskarsdóttir, and W. Lemahieu, “Deep learning for credit scoring: Do or don’t?” *European Journal of Operational Research*, vol. 295, no. 1, pp. 292–305, 2021.
- [101] A. C. Bahnsen, D. Aouada, and B. Ottersten, “A novel cost-sensitive framework for customer churn predictive modeling,” *Decision Analytics*, vol. 2, no. 1, pp. 1–15, 2015.
- [102] IBM Sample Data Sets, *Telco customer churn, version 1*, Retrieved October 10, 2021 from <https://www.kaggle.com/blastchar/telco-customer-churn/version/1>, 2017.
- [103] B. Baesens, D. Roesch, and H. Scheule, *Credit risk analytics: Measurement techniques, applications, and examples in SAS*. John Wiley & Sons, 2016.

References

- [104] G. Petrides, D. Moldovan, L. Coenen, T. Guns, and W. Verbeke, “Cost-sensitive learning for profit-driven credit scoring,” *Journal of the Operational Research Society*, pp. 1–13, 2020.
- [105] I.-C. Yeh and C.-h. Lien, “The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients,” *Expert Systems with Applications*, vol. 36, no. 2, pp. 2473–2480, 2009.
- [106] S. Moro, P. Cortez, and P. Rita, “A data-driven approach to predict the success of bank telemarketing,” *Decision Support Systems*, vol. 62, pp. 22–31, 2014.
- [107] A. Dal Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi, “Calibrating probability with undersampling for unbalanced classification,” in *2015 IEEE Symposium Series on Computational Intelligence*, IEEE, 2015, pp. 159–166.
- [108] V. Van Vlasselaer, C. Bravo, O. Caelen, *et al.*, “Apate: A novel approach for automated credit card transaction fraud detection using network-based extensions,” *Decision Support Systems*, vol. 75, pp. 38–48, 2015.
- [109] S. Garcia and F. Herrera, “An extension on “statistical comparisons of classifiers over multiple data sets” for all pairwise comparisons.,” *Journal of machine learning research*, vol. 9, no. 12, 2008.
- [110] S. García, A. Fernández, J. Luengo, and F. Herrera, “Advanced non-parametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power,” *Information sciences*, vol. 180, no. 10, pp. 2044–2064, 2010.
- [111] C. Derman, G. J. Lieberman, and S. M. Ross, “A sequential stochastic assignment problem,” *Management Science*, vol. 18, no. 7, pp. 349–355, 1972.
- [112] C. Albright and C. Derman, “Asymptotic optimal policies for the stochastic sequential assignment problem,” *Management Science*, vol. 19, no. 1, pp. 46–51, 1972.
- [113] S. C. Albright, “Optimal sequential assignments with random arrival times,” *Management Science*, vol. 21, no. 1, pp. 60–67, 1974.
- [114] B. de Jonge, W. Klingenberg, R. Teunter, and T. Tinga, “Optimum maintenance strategy under uncertainty in the lifetime distribution,” *Reliability engineering & system safety*, vol. 133, pp. 59–67, 2015.

-
- [115] D. M. Louit, R. Pascual, and A. K. Jardine, “A practical procedure for the selection of time-to-failure models based on the assessment of trends in maintenance data,” *Reliability Engineering & System Safety*, vol. 94, no. 10, pp. 1618–1628, 2009.
- [116] M. Fouladirad, C. Paroissin, and A. Grall, “Sensitivity of optimal replacement policies to lifetime parameter estimates,” *European Journal of Operational Research*, vol. 266, no. 3, pp. 963–975, 2018.
- [117] H. Pham and H. Wang, “Imperfect maintenance,” *European Journal of Operational Research*, vol. 94, no. 3, pp. 425–438, 1996.
- [118] H. Wang, “A survey of maintenance policies of deteriorating systems,” *European Journal of Operational Research*, vol. 139, no. 3, pp. 469–489, 2002.
- [119] S.-H. Ding and S. Kamaruddin, “Maintenance policy optimization—literature review and directions,” *The International Journal of Advanced Manufacturing Technology*, vol. 76, no. 5, pp. 1263–1283, 2015.
- [120] B. de Jonge and P. A. Scarf, “A review on maintenance optimization,” *European Journal of Operational Research*, vol. 285, no. 3, pp. 805–824, 2020.
- [121] R. Barlow and L. Hunter, “Optimum preventive maintenance policies,” *Operations research*, vol. 8, no. 1, pp. 90–100, 1960.
- [122] R. Ahmad and S. Kamaruddin, “An overview of time-based and condition-based maintenance in industrial application,” *Computers & industrial engineering*, vol. 63, no. 1, pp. 135–149, 2012.
- [123] M. Faccio, A. Persona, F. Sgarbossa, and G. Zanin, “Industrial maintenance policy development: A quantitative framework,” *International Journal of Production Economics*, vol. 147, pp. 85–93, 2014.
- [124] S. Chukova, R. Arnold, and D. Q. Wang, “Warranty analysis: An approach to modeling imperfect repairs,” *International journal of production economics*, vol. 89, no. 1, pp. 57–68, 2004.
- [125] T. Nakagawa, “Imperfect preventive-maintenance,” *IEEE Transactions on Reliability*, vol. 28, no. 5, pp. 402–402, 1979.
- [126] T. Nakagawa, “Optimum policies when preventive maintenance is imperfect,” *IEEE Transactions on Reliability*, vol. 28, no. 4, pp. 331–332, 1979.
- [127] M. Brown and F. Proschan, “Imperfect repair,” *Journal of Applied Probability*, vol. 20, no. 4, pp. 851–859, 1983.
- [128] H. W. Block, W. S. Borges, and T. H. Savits, “Age-dependent minimal repair,” *Journal of Applied Probability*, vol. 22, no. 2, pp. 370–385, 1985.

References

- [129] M. A. K. Malik, "Reliable preventive maintenance scheduling," *AIIE Transactions*, vol. 11, no. 3, pp. 221–228, 1979.
- [130] M. Kijima, "Some results for repairable systems with general repair," *Journal of Applied Probability*, vol. 26, no. 1, pp. 89–102, 1989.
- [131] M. Tanwar, R. N. Rai, and N. Bolia, "Imperfect repair modeling using kijima type generalized renewal process," *Reliability Engineering & System Safety*, vol. 124, pp. 24–31, 2014.
- [132] Y. Liu, H.-Z. Huang, and X. Zhang, "A data-driven approach to selecting imperfect maintenance models," *IEEE Transactions on Reliability*, vol. 61, no. 1, pp. 101–112, 2011.
- [133] M. L. G. de Toledo, M. A. Freitas, E. A. Colosimo, and G. L. Gilarioni, "Ara and ari imperfect repair models: Estimation, goodness-of-fit and reliability prediction," *Reliability Engineering & System Safety*, vol. 140, pp. 107–115, 2015.
- [134] M. Zhang and M. Xie, "An ameliorated improvement factor model for imperfect maintenance and its goodness of fit," *Technometrics*, vol. 59, no. 2, pp. 237–246, 2017.
- [135] A. Bousdekis, K. Lepenioti, D. Apostolou, and G. Mentzas, "A review of data-driven decision-making methods for industry 4.0 maintenance applications," *Electronics*, vol. 10, no. 7, p. 828, 2021.
- [136] C. Gits, "Design of maintenance concepts," *International journal of production economics*, vol. 24, no. 3, pp. 217–226, 1992.
- [137] S. Alaswad and Y. Xiang, "A review on condition-based maintenance optimization models for stochastically deteriorating system," *Reliability engineering & system safety*, vol. 157, pp. 54–63, 2017.
- [138] L. Swanson, "Linking maintenance strategies to performance," *International journal of production economics*, vol. 70, no. 3, pp. 237–244, 2001.
- [139] T. P. Carvalho, F. A. Soares, R. Vita, R. d. P. Francisco, J. P. Basto, and S. G. Alcalá, "A systematic literature review of machine learning methods applied to predictive maintenance," *Computers & Industrial Engineering*, vol. 137, p. 106 024, 2019.
- [140] M. Fast, M. Assadi, and S. De, "Condition based maintenance of gas turbines using simulation data and artificial neural network: A demonstration of feasibility," in *Turbo Expo: Power for Land, Sea, and Air*, vol. 43123, 2008, pp. 153–161.
- [141] Z. Tian, "An artificial neural network method for remaining useful life prediction of equipment subject to condition monitoring," *Journal of intelligent Manufacturing*, vol. 23, no. 2, pp. 227–237, 2012.

-
- [142] B. Wu, Z. Tian, and M. Chen, "Condition-based maintenance optimization using neural network-based health condition prediction," *Quality and Reliability Engineering International*, vol. 29, no. 8, pp. 1151–1163, 2013.
- [143] Y. Lu, L. Sun, X. Zhang, F. Feng, J. Kang, and G. Fu, "Condition based maintenance optimization for offshore wind turbine considering opportunities based on neural network approach," *Applied Ocean Research*, vol. 74, pp. 69–79, 2018.
- [144] A. Bey-Temsamani, M. Engels, A. Motten, S. Vandenplas, and A. P. Ompusunggu, "A practical approach to combine data mining and prognostics for improved predictive maintenance," *Data Min. Case Stud*, vol. 36, 2009.
- [145] P. Do, A. Voisin, E. Levrat, and B. Iung, "A proactive condition-based maintenance strategy with both perfect and imperfect maintenance actions," *Reliability Engineering & System Safety*, vol. 133, pp. 22–32, 2015.
- [146] K. Matyas, T. Nemeth, K. Kovacs, and R. Glawar, "A procedural approach for realizing prescriptive maintenance planning in manufacturing industries," *CIRP Annals*, vol. 66, no. 1, pp. 461–464, 2017.
- [147] J. Poppe, R. N. Boute, and M. R. Lambrecht, "A hybrid condition-based maintenance policy for continuously monitored components with two degradation thresholds," *European Journal of Operational Research*, vol. 268, no. 2, pp. 515–532, 2018.
- [148] T. Nemeth, F. Ansari, W. Sihn, B. Haslhofer, and A. Schindler, "Prima-x: A reference model for realizing prescriptive maintenance and assessing its maturity enhanced by machine learning," *Procedia CIRP*, vol. 72, pp. 1039–1044, 2018.
- [149] F. Ansari, R. Glawar, and T. Nemeth, "Prima: A prescriptive maintenance model for cyber-physical production systems," *International Journal of Computer Integrated Manufacturing*, vol. 32, no. 4-5, pp. 482–503, 2019.
- [150] A. Kusiak and A. Verma, "A data-mining approach to monitoring wind turbines," *IEEE Transactions on Sustainable Energy*, vol. 3, no. 1, pp. 150–157, 2011.
- [151] Y.-L. Lee, D.-C. Juan, X.-A. Tseng, Y.-T. Chen, and S.-C. Chang, "Dc-prophet: Predicting catastrophic machine failures in datacenters," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2017, pp. 64–76.

- [152] J. Leukel, J. González, and M. Riekert, “Adoption of machine learning technology for failure prediction in industrial maintenance: A systematic review,” *Journal of Manufacturing Systems*, vol. 61, pp. 87–96, 2021.
- [153] F. Jansen, M. Holenderski, T. Ozcelebi, P. Dam, and B. Tijsma, “Predicting machine failures from industrial time series data,” in *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*, IEEE, 2018, pp. 1091–1096.
- [154] K. Chen, S. Pashami, Y. Fan, and S. Nowaczyk, “Predicting air compressor failures using long short term memory networks,” in *EPIA Conference on Artificial Intelligence*, Springer, 2019, pp. 596–609.
- [155] Z. Chen, K. Gryllias, and W. Li, “Mechanical fault diagnosis using convolutional neural networks and extreme learning machine,” *Mechanical systems and signal processing*, vol. 133, p. 106 272, 2019.
- [156] Z. Chen, A. Mauricio, W. Li, and K. Gryllias, “A deep learning method for bearing fault diagnosis based on cyclic spectral coherence and convolutional neural networks,” *Mechanical Systems and Signal Processing*, vol. 140, p. 106 683, 2020.
- [157] R. Savitha, A. Ambikapathi, and K. Rajaraman, “Online rbm: Growing restricted boltzmann machine on the fly for unsupervised representation,” *Applied Soft Computing*, vol. 92, p. 106 278, 2020.
- [158] P. F. Orrù, A. Zoccheddu, L. Sassu, C. Mattia, R. Cozza, and S. Arena, “Machine learning approach using mlp and svm algorithms for the fault prediction of a centrifugal pump in the oil and gas industry,” *Sustainability*, vol. 12, no. 11, p. 4776, 2020.
- [159] F. Alves, H. Badikyran, H. A. Moreira, *et al.*, “Deployment of a smart and predictive maintenance system in an industrial case study,” in *2020 IEEE 29th International Symposium on Industrial Electronics (ISIE)*, IEEE, 2020, pp. 493–498.
- [160] J. Zhao and W. Huang, “Transfer learning method for rolling bearing fault diagnosis under different working conditions based on cyclegan,” *Measurement Science and Technology*, vol. 33, no. 2, p. 025 003, 2021.
- [161] Z. Ye and J. Yu, “Aksnet: A novel convolutional neural network with adaptive kernel width and sparse regularization for machinery fault diagnosis,” *Journal of Manufacturing Systems*, vol. 59, pp. 467–480, 2021.

-
- [162] J. Figueroa Barraza, L. Guarda Bräuning, R. Benites Perez, C. B. Morais, M. R. Martins, and E. L. Droguett, “Deep learning health state prognostics of physical assets in the oil and gas industry,” *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 236, no. 4, pp. 598–616, 2022.
- [163] D. B. Rubin, “Estimating causal effects of treatments in randomized and nonrandomized studies.,” *Journal of educational Psychology*, vol. 66, no. 5, p. 688, 1974.
- [164] L. Yao, Z. Chu, S. Li, Y. Li, J. Gao, and A. Zhang, “A survey on causal inference,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 15, no. 5, pp. 1–46, 2021.
- [165] G. W. Imbens, “The role of the propensity score in estimating dose-response functions,” *Biometrika*, vol. 87, no. 3, pp. 706–710, 2000.
- [166] K. Hirano and G. W. Imbens, “The propensity score with continuous treatments,” *Applied Bayesian modeling and causal inference from incomplete-data perspectives*, vol. 226164, pp. 73–84, 2004.
- [167] K. Imai and D. A. Van Dyk, “Causal inference with general treatment regimes: Generalizing the propensity score,” *Journal of the American Statistical Association*, vol. 99, no. 467, pp. 854–866, 2004.
- [168] P. Schwab, L. Linhardt, S. Bauer, J. M. Buhmann, and W. Karlen, “Learning counterfactual representations for estimating individual dose-response curves,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 5612–5619.
- [169] I. Bica, J. Jordon, and M. van der Schaar, “Estimating the effects of continuous-valued interventions using generative adversarial networks,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 434–16 445, 2020.
- [170] J. Berrevoets, J. Jordon, I. Bica, M. van der Schaar, *et al.*, “Organite: Optimal transplant donor organ offering using an individual treatment effect,” *Advances in Neural Information Processing Systems*, vol. 33, 2020.
- [171] S. Athey and S. Wager, “Policy learning with observational data,” *Econometrica*, vol. 89, no. 1, pp. 133–161, 2021.
- [172] H. R. Varian, “Causal inference in economics and marketing,” *Proceedings of the National Academy of Sciences*, vol. 113, no. 27, pp. 7310–7315, 2016.
- [173] F. Devriendt, D. Moldovan, and W. Verbeke, “A literature survey and experimental evaluation of the state-of-the-art in uplift modeling: A stepping stone toward the development of prescriptive analytics,” *Big Data*, vol. 6, no. 1, pp. 13–41, 2018.

References

- [174] D. Webbink, “Causal effects in education,” *Journal of Economic Surveys*, vol. 19, no. 4, pp. 535–560, 2005.
- [175] W. Verbeke, D. Olaya, J. Berrevoets, S. Verboven, and S. Maldonado, “The foundations of cost-sensitive causal classification,” *arXiv preprint arXiv:2007.12582*, 2020.
- [176] W. Verbeke, D. Olaya, M.-A. Guerry, and J. Van Belle, “To do or not to do? cost-sensitive causal classification with individual treatment effect estimates,” *European Journal of Operational Research*, 2022.
- [177] D. Bertsimas, J. Dunn, and N. Mundru, “Optimal prescriptive trees,” *INFORMS Journal on Optimization*, vol. 1, no. 2, pp. 164–183, 2019.
- [178] D. Bertsimas and N. Kallus, “From predictive to prescriptive analytics,” *Management Science*, vol. 66, no. 3, pp. 1025–1044, 2020.
- [179] L. Deprez, K. Antonio, and R. Boute, “Pricing service maintenance contracts using predictive analytics,” *European Journal of Operational Research*, vol. 290, no. 2, pp. 530–545, 2021.
- [180] D. B. Rubin, “Direct and indirect causal effects via potential outcomes,” *Scandinavian Journal of Statistics*, vol. 31, no. 2, pp. 161–170, 2004.
- [181] D. B. Rubin, “Causal inference using potential outcomes: Design, modeling, decisions,” *Journal of the American Statistical Association*, vol. 100, no. 469, pp. 322–331, 2005.
- [182] P. W. Holland, “Statistics and causal inference,” *Journal of the American Statistical Association*, vol. 81, no. 396, pp. 945–960, 1986.
- [183] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, “Generative adversarial networks,” *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.
- [184] R. Silva, “Observational-interventional priors for dose-response learning,” *Advances in Neural Information Processing Systems*, vol. 29, 2016.
- [185] A. Alaa and M. Van Der Schaar, “Validating causal inference models via influence functions,” in *International Conference on Machine Learning*, PMLR, 2019, pp. 191–201.
- [186] H. Parikh, C. Varjao, L. Xu, and E. T. Tchetgen, “Validating causal inference methods,” in *International Conference on Machine Learning*, PMLR, 2022, pp. 17 346–17 358.
- [187] L. Lei, A. D’Amour, P. Ding, A. Feller, and J. Sekhon, “Distribution-free assessment of population overlap in observational studies,” 2021.

-
- [188] M. Oberst, F. Johansson, D. Wei, *et al.*, “Characterization of overlap in observational studies,” in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2020, pp. 788–798.
- [189] R. C. Nethery, F. Mealli, and F. Dominici, “Estimating population average causal effects in the presence of non-overlap: The effect of natural gas compressor station exposure on cancer mortality,” *The annals of applied statistics*, vol. 13, no. 2, p. 1242, 2019.
- [190] A. Jesson, S. Mindermann, U. Shalit, and Y. Gal, “Identifying causal-effect inference failure with uncertainty-aware models,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 11 637–11 649, 2020.
- [191] A. D’Amour, “On multi-cause approaches to causal inference with unobserved confounding: Two cautionary failure cases and a promising alternative,” in *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, K. Chaudhuri and M. Sugiyama, Eds., ser. Proceedings of Machine Learning Research, vol. 89, PMLR, Apr. 2019, pp. 3478–3486. [Online]. Available: <https://proceedings.mlr.press/v89/d-amour19a.html>.
- [192] A. Franks, A. D’Amour, and A. Feller, “Flexible sensitivity analysis for observational studies without observable implications,” *Journal of the American Statistical Association*, 2019.
- [193] A. Jesson, S. Mindermann, Y. Gal, and U. Shalit, “Quantifying ignorance in individual-level causal-effect estimates under hidden confounding,” in *International Conference on Machine Learning*, PMLR, 2021, pp. 4829–4838.
- [194] J. M. Robins, “Association, causation, and marginal structural models,” *Synthese*, pp. 151–179, 1999.
- [195] M. A. Hernán, B. Brumback, and J. M. Robins, “Marginal structural models to estimate the joint causal effect of nonrandomized treatments,” *Journal of the American Statistical Association*, vol. 96, no. 454, pp. 440–448, 2001.
- [196] I. Bica, A. M. Alaa, J. Jordon, and M. van der Schaar, “Estimating counterfactual treatment outcomes over time through adversarially balanced representations,” in *International Conference on Learning Representations*, 2019.
- [197] A. N. Elmachtoub and P. Grigas, “Smart “predict, then optimize”,” *Management Science*, vol. 68, no. 1, pp. 9–26, 2022.
- [198] F. Devriendt, J. Berrevoets, and W. Verbeke, “Why you should stop predicting customer churn and start using uplift models,” *Information Sciences*, vol. 548, pp. 497–515, 2021.

References

- [199] D. Olaya, J. Vásquez, S. Maldonado, J. Miranda, and W. Verbeke, “Uplift modeling for preventing student dropout in higher education,” *Decision Support Systems*, vol. 134, p. 113 320, 2020.
- [200] F. Johansson, U. Shalit, and D. Sontag, “Learning representations for counterfactual inference,” in *International conference on machine learning*, PMLR, 2016, pp. 3020–3029.
- [201] U. Shalit, F. D. Johansson, and D. Sontag, “Estimating individual treatment effect: Generalization bounds and algorithms,” in *International Conference on Machine Learning*, PMLR, 2017, pp. 3076–3085.
- [202] Y. Zhang, A. Bellot, and M. Schaar, “Learning overlapping representations for the estimation of individualized treatment effects,” in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2020, pp. 1005–1014.
- [203] A. Curth and M. van der Schaar, “On inductive biases for heterogeneous treatment effect estimation,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 15 883–15 894, 2021.
- [204] A. M. Alaa and M. van der Schaar, “Bayesian inference of individualized treatment effects using multi-task gaussian processes,” *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [205] J. L. Hill, “Bayesian nonparametric modeling for causal inference,” *Journal of Computational and Graphical Statistics*, vol. 20, no. 1, pp. 217–240, 2011.
- [206] P. Rzepakowski and S. Jaroszewicz, “Decision trees for uplift modeling with single and multiple treatments,” *Knowledge and Information Systems*, vol. 32, pp. 303–327, 2012.
- [207] S. Wager and S. Athey, “Estimation and inference of heterogeneous treatment effects using random forests,” *Journal of the American Statistical Association*, vol. 113, no. 523, pp. 1228–1242, 2018.
- [208] S. R. Künzel, J. S. Sekhon, P. J. Bickel, and B. Yu, “Metalearners for estimating heterogeneous treatment effects using machine learning,” *Proceedings of the National Academy of Sciences*, vol. 116, no. 10, pp. 4156–4165, 2019.
- [209] A. Curth and M. van der Schaar, “Nonparametric estimation of heterogeneous treatment effects: From theory to learning algorithms,” in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2021, pp. 1810–1818.
- [210] D. Hall, “Principles of economics,” 1920.

-
- [211] D. Kahneman and A. Tversky, “Prospect theory: An analysis of decision under risk,” *Econometrica*, vol. 47, no. 2, pp. 263–292, 1979.
- [212] J. F. Nash Jr, “The bargaining problem,” *Econometrica: Journal of the econometric society*, pp. 155–162, 1950.
- [213] M. Spence and R. Zeckhauser, “Insurance, information, and individual action,” in *Uncertainty in Economics*, Elsevier, 1978, pp. 333–343.
- [214] D. L. Thurston, “A formal method for subjective design evaluation with multiple attributes,” *Research in engineering Design*, vol. 3, no. 2, pp. 105–122, 1991.
- [215] J. S. Pliskin, D. S. Shepard, and M. C. Weinstein, “Utility functions for life years and health status,” *Operations research*, vol. 28, no. 1, pp. 206–224, 1980.
- [216] J. Yoon, J. Jordon, and M. Van Der Schaar, “Ganite: Estimation of individualized treatment effects using generative adversarial nets,” in *International Conference on Learning Representations*, 2018.
- [217] C. Louizos, U. Shalit, J. M. Mooij, D. Sontag, R. Zemel, and M. Welling, “Causal effect inference with deep latent-variable models,” *Advances in neural information processing systems*, vol. 30, 2017.
- [218] T. Zhou, W. E. Carson IV, and D. Carlson, “Estimating potential outcome distributions with collaborating causal networks,” *Transactions on Machine Learning Research*, 2022.
- [219] T. Zhou, Y. Li, Y. Wu, and D. Carlson, “Estimating uncertainty intervals from collaborating networks,” *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 11 645–11 691, 2021.
- [220] V. Melnychuk, D. Frauen, and S. Feuerriegel, “Normalizing flows for interventional density estimation,” *arXiv preprint arXiv:2209.06203*, 2022.
- [221] E. Kennedy, S. Balakrishnan, and L. Wasserman, “Semiparametric counterfactual density estimation,” *Biometrika*, asad017, 2023.
- [222] I. Bica, A. M. Alaa, J. Jordon, and M. van der Schaar, “Estimating counterfactual treatment outcomes over time through adversarially balanced representations,” in *International Conference on Learning Representations*, 2019.
- [223] E. De Brouwer, J. Gonzalez, and S. Hyland, “Predicting the impact of treatments over time with uncertainty aware neural differential equations,” in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2022, pp. 4705–4722.

- [224] J. DiNardo, N. Fortin, and T. Lemieux, *Labor market institutions and the distribution of wages, 1973-1992: A semiparametric approach*, 1995.
- [225] V. Chernozhukov, I. Fernández-Val, and B. Melly, “Inference on counterfactual distributions,” *Econometrica*, vol. 81, no. 6, pp. 2205–2268, 2013.
- [226] S. Firpo and C. Pinto, “Identification and estimation of distributional impacts of interventions using changes in inequality measures,” *Journal of Applied Econometrics*, vol. 31, no. 3, pp. 457–486, 2016.
- [227] V. Chernozhukov and C. Hansen, “An iv model of quantile treatment effects,” *Econometrica*, vol. 73, no. 1, pp. 245–261, 2005.
- [228] S. Firpo, “Efficient semiparametric estimation of quantile treatment effects,” *Econometrica*, vol. 75, no. 1, pp. 259–276, 2007.
- [229] L. Wang, Y. Zhou, R. Song, and B. Sherwood, “Quantile-optimal treatment regimes,” *Journal of the American Statistical Association*, vol. 113, no. 523, pp. 1243–1254, 2018.
- [230] L. Lei and E. J. Candès, “Conformal inference of counterfactuals and individual treatment effects,” *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 83, no. 5, pp. 911–938, 2021.
- [231] A. Alaa, Z. Ahmad, and M. van der Laan, “Conformal meta-learners for predictive inference of individual treatment effects,” *arXiv preprint arXiv:2308.14895*, 2023.
- [232] M. Makar, F. Johansson, J. Gutttag, and D. Sontag, “Estimation of bounds on potential outcomes for decision making,” in *International Conference on Machine Learning*, PMLR, 2020, pp. 6661–6671.
- [233] N. Jiang and J. Huang, “Minimax value interval for off-policy evaluation and policy optimization,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 2747–2758, 2020.
- [234] Y. Chandak, S. Shankar, and P. S. Thomas, “High-confidence off-policy (or counterfactual) variance estimation,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 6939–6947.
- [235] Y. Chandak, S. Niekum, B. da Silva, E. Learned-Miller, E. Brunskill, and P. S. Thomas, “Universal off-policy evaluation,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 27 475–27 490, 2021.
- [236] T. Xu, Z. Yang, Z. Wang, and Y. Liang, “A unifying framework of off-policy general value function evaluation,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 13 570–13 583, 2022.

-
- [237] Y. Zhang, C. Shi, and S. Luo, “Conformal off-policy prediction,” in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2023, pp. 2751–2768.
- [238] R. Wu, M. Uehara, and W. Sun, “Distributional offline policy evaluation with predictive error guarantees,” *arXiv preprint arXiv:2302.09456*, 2023.
- [239] M. G. Bellemare, W. Dabney, and R. Munos, “A distributional perspective on reinforcement learning,” in *International conference on machine learning*, PMLR, 2017, pp. 449–458.
- [240] M. G. Bellemare, W. Dabney, and M. Rowland, *Distributional reinforcement learning*. MIT Press, 2023.
- [241] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, “Generative adversarial nets,” *Advances in neural information processing systems*, vol. 27, 2014.
- [242] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in *International Conference on Learning Representations*, 2014.
- [243] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, “Normalizing flows for probabilistic modeling and inference,” *Journal of Machine Learning Research*, vol. 22, no. 57, pp. 1–64, 2021.
- [244] D. Rezende and S. Mohamed, “Variational inference with normalizing flows,” in *International conference on machine learning*, PMLR, 2015, pp. 1530–1538.
- [245] L. Dinh, J. Sohl-Dickstein, and S. Bengio, “Density estimation using real nvp,” 2017.
- [246] L. Dinh, D. Krueger, and Y. Bengio, “Nice: Non-linear independent components estimation,” *arXiv preprint arXiv:1410.8516*, 2014.
- [247] D. P. Kingma and P. Dhariwal, “Glow: Generative flow with invertible 1x1 convolutions,” *Advances in neural information processing systems*, vol. 31, 2018.
- [248] A. Oord, Y. Li, I. Babuschkin, *et al.*, “Parallel wavenet: Fast high-fidelity speech synthesis,” in *International conference on machine learning*, PMLR, 2018, pp. 3918–3926.
- [249] S. Kim, S.-g. Lee, J. Song, J. Kim, and S. Yoon, “Flowwavenet: A generative flow for raw audio,” *arXiv preprint arXiv:1811.02155*, 2018.
- [250] D. Tran, K. Vafa, K. Agrawal, L. Dinh, and B. Poole, “Discrete flows: Invertible generative models of discrete data,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.

- [251] R. Prenger, R. Valle, and B. Catanzaro, “Waveglow: A flow-based generative network for speech synthesis,” in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 3617–3621.
- [252] K. Madhawa, K. Ishiguro, K. Nakago, and M. Abe, “Graphnvp: An invertible flow model for generating molecular graphs,” *arXiv preprint arXiv:1905.11600*, 2019.
- [253] T. Haarnoja, K. Hartikainen, P. Abbeel, and S. Levine, “Latent space policies for hierarchical reinforcement learning,” in *International Conference on Machine Learning*, PMLR, 2018, pp. 1851–1860.
- [254] P. N. Ward, A. Smofsky, and A. J. Bose, “Improving exploration in soft-actor-critic with normalizing flows policies,” *arXiv preprint arXiv:1906.02771*, 2019.
- [255] Y. Schroecker, M. Vecerik, and J. Scholz, “Generative predecessor models for sample-efficient imitation learning,” in *International Conference on Learning Representations*, 2019.
- [256] J. Dumas, A. Wehenkel, D. Lanaspèze, B. Cornélusse, and A. Sutera, “A deep generative model for probabilistic energy forecasting in power systems: Normalizing flows,” *Applied Energy*, vol. 305, p. 117871, 2022.
- [257] 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,” *IEEE Access*, vol. 8, pp. 77 587–77 597, 2020.
- [258] C.-W. Huang, D. Krueger, A. Lacoste, and A. Courville, “Neural autoregressive flows,” in *International Conference on Machine Learning*, PMLR, 2018, pp. 2078–2087.
- [259] S. Chen and R. Gopinath, “Gaussianization,” *Advances in neural information processing systems*, vol. 13, 2000.
- [260] C. Meng, Y. Song, J. Song, and S. Ermon, “Gaussianization flows,” in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2020, pp. 4336–4345.
- [261] R. T. Chen, J. Behrmann, D. K. Duvenaud, and J.-H. Jacobsen, “Residual flows for invertible generative modeling,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [262] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, “Neural spline flows,” *Advances in neural information processing systems*, vol. 32, 2019.

-
- [263] D. Rubin, "Causal inference using potential outcomes: Design, modeling, decisions," *Journal of the American Statistical Association*, vol. 100, pp. 322–331, Feb. 2005. DOI: 10.2307/27590541.
- [264] B. G. Vegetabile, "On the distinction between " conditional average treatment effects"(cate) and " individual treatment effects"(ite) under ignorability assumptions," *arXiv preprint arXiv:2108.04939*, 2021.
- [265] P. R. Rosenbaum and D. B. Rubin, "The central role of the propensity score in observational studies for causal effects," *Biometrika*, vol. 70, no. 1, pp. 41–55, 1983.
- [266] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 723–773, 2012.
- [267] A. Curth, D. Svensson, J. Weatherall, and M. van der Schaar, "Really doing great at estimating cate? a critical look at ml benchmarking practices in treatment effect estimation," in *Thirty-fifth conference on neural information processing systems datasets and benchmarks track (round 2)*, 2021.
- [268] L. Yang, Z. Zhang, Y. Song, *et al.*, "Diffusion models: A comprehensive survey of methods and applications," *arXiv preprint arXiv:2209.00796*, 2022.
- [269] A. Curth and M. van der Schaar, "In search of insights, not magic bullets: Towards demystification of the model selection dilemma in heterogeneous treatment effect estimation," 2023.
- [270] N. Kallus, X. Mao, and A. Zhou, "Interval estimation of individual-level causal effects under unobserved confounding," in *The 22nd international conference on artificial intelligence and statistics*, PMLR, 2019, pp. 2281–2290.
- [271] M. Oprescu, J. Dorn, M. Ghoummaid, A. Jesson, N. Kallus, and U. Shalit, "B-learner: Quasi-oracle bounds on heterogeneous causal effects under hidden confounding," 2023.
- [272] I. Bica, A. M. Alaa, J. Jordon, and M. van der Schaar, "Estimating counterfactual treatment outcomes over time through adversarially balanced representations," in *International Conference on Learning Representations*, 2019.
- [273] B. A. Goldstein, N. A. Bhavsar, M. Phelan, and M. J. Pencina, "Controlling for informed presence bias due to the number of health encounters in an electronic health record," *American journal of epidemiology*, vol. 184, no. 11, pp. 847–855, 2016.

- [274] H. Lin, D. O. Scharfstein, and R. A. Rosenheck, “Analysis of longitudinal data with irregular, outcome-dependent follow-up,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 66, no. 3, pp. 791–813, 2004.
- [275] P. Del Moral and L. M. Murray, “Sequential monte carlo with highly informative observations,” *SIAM/ASA Journal on Uncertainty Quantification*, vol. 3, no. 1, pp. 969–997, 2015.
- [276] J. A. Clithero, “Response times in economics: Looking through the lens of sequential sampling models,” *Journal of Economic Psychology*, vol. 69, pp. 61–86, 2018.
- [277] J. M. Robins, A. Rotnitzky, and L. P. Zhao, “Analysis of semiparametric regression models for repeated outcomes in the presence of missing data,” *Journal of the american statistical association*, vol. 90, no. 429, pp. 106–121, 1995.
- [278] C. E. McCulloch, J. M. Neuhaus, and R. L. Olin, “Biased and unbiased estimation in longitudinal studies with informative visit processes,” *Biometrics*, vol. 72, no. 4, pp. 1315–1324, 2016.
- [279] L. Liu, X. Huang, and J. O’Quigley, “Analysis of longitudinal data in the presence of informative observational times and a dependent terminal event, with application to medical cost data,” *Biometrics*, vol. 64, no. 3, pp. 950–958, 2008.
- [280] A. Gasparini, K. R. Abrams, J. K. Barrett, *et al.*, “Mixed-effects models for health care longitudinal data with an informative visiting process: A monte carlo simulation study,” *Statistica Neerlandica*, vol. 74, no. 1, pp. 5–23, 2020.
- [281] M. A. Hernán, M. McAdams, N. McGrath, E. Lanoy, and D. Costagliola, “Observation plans in longitudinal studies with time-varying treatments,” *Statistical methods in medical research*, vol. 18, no. 1, pp. 27–52, 2009.
- [282] D. Farzanfar, A. Abumuamar, J. Kim, E. Sirolich, Y. Wang, and E. Pullenayegum, “Longitudinal studies that use data collected as part of usual care risk reporting biased results: A systematic review,” *BMC Medical Research Methodology*, vol. 17, no. 1, pp. 1–12, 2017.
- [283] N. Hassanpour and R. Greiner, “Learning disentangled representations for counterfactual regression,” in *International Conference on Learning Representations*, 2020.
- [284] A. Curth, C. Lee, and M. van der Schaar, “Survite: Learning heterogeneous treatment effects from time-to-event data,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 26 740–26 753, 2021.

-
- [285] B. Lim, A. Alaa, and M. van der Schaar, “Forecasting treatment responses over time using recurrent marginal structural networks,” in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc., 2018. [Online]. Available: <https://proceedings.neurips.cc/paper/2018/file/56e6a93212e4482d99c84a639d254b67-Paper.pdf>.
- [286] R. Li, Z. Shahn, J. Li, *et al.*, “G-net: A deep learning approach to g-computation for counterfactual outcome prediction under dynamic treatment regimes,” in *Machine Learning for Health*, 2021.
- [287] J. Berrevoets, A. Curth, I. Bica, E. McKinney, and M. van der Schaar, “Disentangled counterfactual recurrent networks for treatment effect inference over time,” *arXiv preprint arXiv:2112.03811*, 2021.
- [288] V. Melnychuk, D. Frauen, and S. Feuerriegel, “Causal transformer for estimating counterfactual outcomes,” in *Proceedings of the 39th International Conference on Machine Learning*, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., ser. Proceedings of Machine Learning Research, vol. 162, PMLR, Jul. 2022, pp. 15 293–15 329. [Online]. Available: <https://proceedings.mlr.press/v162/melnynchuk22a.html>.
- [289] D. Gwak, G. Sim, M. Poli, S. Massaroli, J. Choo, and E. Choi, “Neural ordinary differential equations for intervention modeling,” *arXiv preprint arXiv:2010.08304*, 2020.
- [290] E. De Brouwer, J. Gonzalez, and S. Hyland, “Predicting the impact of treatments over time with uncertainty aware neural differential equations,” in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2022, pp. 4705–4722.
- [291] N. Seedat, F. Imrie, A. Bellot, Z. Qian, and M. van der Schaar, “Continuous-time modeling of counterfactual outcomes using neural controlled differential equations,” in *International Conference on Machine Learning*, PMLR, 2022, pp. 19 497–19 521.
- [292] A. Curth and M. van der Schaar, “Understanding the impact of competing events on heterogeneous treatment effect estimation from time-to-event data,” in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2023, pp. 7961–7980.
- [293] J. J. Lok, “Statistical modeling of causal effects in continuous time,” *The Annals of Statistics*, vol. 36, no. 3, pp. 1464–1507, 2008.

- [294] E. M. Pullenayegum and L. S. Lim, “Longitudinal data subject to irregular observation: A review of methods with a focus on visit processes, assumptions, and study design,” *Statistical methods in medical research*, vol. 25, no. 6, pp. 2992–3014, 2016.
- [295] D. B. Rubin, “Inference and missing data,” *Biometrika*, vol. 63, no. 3, pp. 581–592, 1976.
- [296] C. Gische, S. G. West, and M. C. Voelkle, “Forecasting causal effects of interventions versus predicting future outcomes,” *Structural Equation Modeling: A Multidisciplinary Journal*, vol. 28, no. 3, pp. 475–492, 2021.
- [297] E. H. Kennedy, “Efficient nonparametric causal inference with missing exposure information,” *The international journal of biostatistics*, vol. 16, no. 1, 2020.
- [298] I. Redko, E. Morvant, A. Habrard, M. Sebban, and Y. Bennani, “A survey on domain adaptation theory: Learning bounds and theoretical guarantees,” *arXiv preprint arXiv:2004.11829*, 2020.
- [299] A. Farahani, S. Voghoei, K. Rasheed, and H. R. Arabnia, “A brief review of domain adaptation,” *Advances in data science and information engineering*, pp. 877–894, 2021.
- [300] H. Shimodaira, “Improving predictive inference under covariate shift by weighting the log-likelihood function,” *Journal of statistical planning and inference*, vol. 90, no. 2, pp. 227–244, 2000.
- [301] O. Shchur, A. C. Türkmen, T. Januschowski, and S. Günnemann, “Neural temporal point processes: A review,” *arXiv preprint arXiv:2104.03528*, 2021.
- [302] P. Bužková and T. Lumley, “Semiparametric modeling of repeated measurements under outcome-dependent follow-up,” *Statistics in Medicine*, vol. 28, no. 6, pp. 987–1003, 2009.
- [303] P. Kidger, J. Morrill, J. Foster, and T. Lyons, “Neural controlled differential equations for irregular time series,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 6696–6707, 2020.
- [304] N. Hassanpour and R. Greiner, “Counterfactual regression with importance sampling weights,” in *IJCAI*, 2019, pp. 5880–5887.
- [305] T. J. VanderWeele, “Principles of confounder selection,” *European journal of epidemiology*, vol. 34, pp. 211–219, 2019.
- [306] F. Maia Polo and R. Vicente, “Effective sample size, dimensionality, and generalization in covariate shift adaptation,” *Neural Computing and Applications*, pp. 1–13, 2022.

-
- [307] K. Furuse, M. Fukuoka, M. Kawahara, *et al.*, “Phase iii study of concurrent versus sequential thoracic radiotherapy in combination with mitomycin, vindesine, and cisplatin in unresectable stage iii non-small-cell lung cancer,” *Journal of Clinical Oncology*, vol. 17, no. 9, pp. 2692–2692, 1999.
- [308] A. Aupérin, C. Le Péchoux, E. Rolland, *et al.*, “Meta-analysis of concomitant versus sequential radiochemotherapy in locally advanced non-small-cell lung cancer,” *Database of Abstracts of Reviews of Effects (DARE): Quality-assessed Reviews [Internet]*, 2010.
- [309] W. J. Curran Jr, R. Paulus, C. J. Langer, *et al.*, “Sequential vs concurrent chemoradiation for stage iii non-small cell lung cancer: Randomized phase iii trial rtog 9410,” *Journal of the National Cancer Institute*, vol. 103, no. 19, pp. 1452–1460, 2011.
- [310] C. Geng, H. Paganetti, and C. Grassberger, “Prediction of treatment response for combined chemo-and radiation therapy for non-small cell lung cancer patients using a bio-mathematical model,” *Scientific reports*, vol. 7, no. 1, pp. 1–12, 2017.
- [311] C. Cortes, Y. Mansour, and M. Mohri, “Learning bounds for importance weighting,” *Advances in neural information processing systems*, vol. 23, 2010.
- [312] X. He, K. Zhao, and X. Chu, “Automl: A survey of the state-of-the-art,” *Knowledge-based systems*, vol. 212, p. 106622, 2021.
- [313] S. K. Karmaker, M. M. Hassan, M. J. Smith, L. Xu, C. Zhai, and K. Veeramachaneni, “Automl to date and beyond: Challenges and opportunities,” *ACM Computing Surveys (CSUR)*, vol. 54, no. 8, pp. 1–36, 2021.
- [314] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for hyper-parameter optimization,” *Advances in neural information processing systems*, vol. 24, 2011.
- [315] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms,” *Advances in neural information processing systems*, vol. 25, 2012.
- [316] A. Alaa and M. van der Schaar, “Autoprognosis: Automated clinical prognostic modeling via bayesian optimization with structured kernel learning,” in *International conference on machine learning*, PMLR, 2018, pp. 139–148.
- [317] M. Feurer, A. Klein, K. Eggenberger, J. Springenberg, M. Blum, and F. Hutter, “Efficient and robust automated machine learning,” *Advances in neural information processing systems*, vol. 28, 2015.

- [318] N. Erickson, J. Mueller, A. Shirkov, *et al.*, “Autogluon-tabular: Robust and accurate automl for structured data,” *arXiv preprint arXiv:2003.06505*, 2020.
- [319] X. Shi, J. Mueller, N. Erickson, M. Li, and A. J. Smola, “Benchmarking multimodal automl for tabular data with text fields,” *arXiv preprint arXiv:2111.02705*, 2021.
- [320] E. Bisong and E. Bisong, “Google automl: Cloud vision,” *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, pp. 581–598, 2019.
- [321] E. LeDell and S. Poirier, “H2o automl: Scalable automatic machine learning,” in *Proceedings of the AutoML Workshop at ICML*, ICML San Diego, CA, USA, vol. 2020, 2020.
- [322] D. Jarrett, J. Yoon, I. Bica, Z. Qian, A. Ercole, and M. van der Schaar, “Clairvoyance: A pipeline toolkit for medical time series,” in *International Conference on Learning Representations*, 2021.
- [323] C. Wang, Q. Wu, M. Weimer, and E. Zhu, “Flaml: A fast and lightweight automl library,” *Proceedings of Machine Learning and Systems*, vol. 3, pp. 434–447, 2021.
- [324] L.-C. Chen, M. Collins, Y. Zhu, *et al.*, “Searching for efficient multi-scale architectures for dense image prediction,” *Advances in neural information processing systems*, vol. 31, 2018.
- [325] D. So, Q. Le, and C. Liang, “The evolved transformer,” in *International conference on machine learning*, PMLR, 2019, pp. 5877–5886.
- [326] F. Runge, D. Stoll, S. Falkner, and F. Hutter, “Learning to design rna,” in *International Conference on Learning Representations*, 2019.
- [327] T. Elsken, J. H. Metzen, and F. Hutter, “Neural architecture search: A survey,” *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019.
- [328] M. Oprescu, V. Syrgkanis, and Z. S. Wu, “Orthogonal random forest for causal inference,” in *International Conference on Machine Learning*, PMLR, 2019, pp. 4932–4941.
- [329] J. Berrevoets, F. Imrie, T. Kyono, J. Jordon, and M. van der Schaar, “To impute or not to impute? missing data in treatment effect estimation,” in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2023, pp. 3568–3590.
- [330] Z. Zhao, Y. Zhang, T. Harinen, and M. Yung, “Feature selection methods for uplift modeling and heterogeneous treatment effect,” in *IFIP International Conference on Artificial Intelligence Applications and Innovations*, Springer, 2022, pp. 217–230.

-
- [331] D. Mahajan, I. Mitliagkas, B. Neal, and V. Syrgkanis, “Empirical analysis of model selection for heterogeneous causal effect estimation,” in *The Twelfth International Conference on Learning Representations*, 2023.
- [332] M. Doutréline and G. Varoquaux, “How to select predictive models for causal inference?” *arXiv preprint arXiv:2302.00370*, 2023.
- [333] A. Schuler, M. Baiocchi, R. Tibshirani, and N. Shah, “A comparison of methods for model selection when estimating individual treatment effects,” *arXiv preprint arXiv:1804.05146*, 2018.
- [334] H. Chen, T. Harinen, J.-Y. Lee, M. Yung, and Z. Zhao, “Causalml: Python package for causal machine learning,” *arXiv preprint arXiv:2002.11631*, 2020.
- [335] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, “Scikit-learn: Machine learning in python,” *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.
- [336] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.
- [337] V. Dorie, J. Hill, U. Shalit, M. Scott, and D. Cervone, “Automated versus do-it-yourself methods for causal inference: Lessons learned from a data analysis competition,” *Statistical science*, vol. 34, no. 1, pp. 43–68, 2019.
- [338] J. Albert and D. Goldenberg, “E-commerce promotions personalization via online multiple-choice knapsack with uplift modeling,” in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 2863–2872.
- [339] A. Savachkin and A. Uribe, “Dynamic redistribution of mitigation resources during influenza pandemics,” *Socio-Economic Planning Sciences*, vol. 46, no. 1, pp. 33–45, 2012.
- [340] D. Bhattacharya and P. Dupas, “Inferring welfare maximizing treatment assignment under budget constraints,” *Journal of Econometrics*, vol. 167, no. 1, pp. 168–196, 2012.
- [341] E. Diemert, A. Betlei, C. Renaudin, and M.-R. Amini, “A large scale benchmark for uplift modeling,” in *Proceedings of the AdKDD and TargetAd Workshop, London, UK*, 2018.
- [342] D. Goldenberg, J. Albert, L. Bernardi, and P. Estevez, “Free lunch! retrospective uplift modeling for dynamic promotions recommendation within roi constraints,” in *Proceedings of the 14th ACM Conference on Recommender Systems*, 2020, pp. 486–491.

- [343] V. Syrgkanis, G. Lewis, M. Oprescu, *et al.*, “Causal inference and machine learning in practice with econml and causalml: Industrial use cases at microsoft, tripadvisor, uber,” in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 4072–4073.
- [344] A. Betlei, E. Diemert, and M.-R. Amini, “Uplift modeling with generalization guarantees,” in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 55–65.
- [345] W. Zhang, J. Li, and L. Liu, “A unified survey of treatment effect heterogeneity modelling and uplift modelling,” *ACM Computing Surveys (CSUR)*, vol. 54, no. 8, pp. 1–36, 2021.
- [346] D. Olaya, K. Coussement, and W. Verbeke, “A survey and benchmarking study of multitreatment uplift modeling,” *Data Mining and Knowledge Discovery*, vol. 34, no. 2, pp. 273–308, 2020.
- [347] A. Betlei, E. Diemert, and M.-R. Amini, “Uplift prediction with dependent feature representation in imbalanced treatment and control conditions,” in *Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13–16, 2018, Proceedings, Part V 25*, Springer, 2018, pp. 47–57.
- [348] K. Kane, V. S. Lo, and J. Zheng, “Mining for the truly responsive customers and prospects using true-lift modeling: Comparison of new and existing methods,” *Journal of Marketing Analytics*, vol. 2, no. 4, pp. 218–238, 2014.
- [349] M. Soltys, S. Jaroszewicz, and P. Rzepakowski, “Ensemble methods for uplift modeling,” *Data Mining and Knowledge Discovery*, vol. 29, pp. 1531–1559, 2015.
- [350] L. Guelman, M. Guillén, and A. M. Pérez-Marín, “Uplift random forests,” *Cybernetics and Systems*, vol. 46, no. 3-4, pp. 230–248, 2015.
- [351] R. M. Gubela and S. Lessmann, “Uplift modeling with value-driven evaluation metrics,” *Decision Support Systems*, vol. 150, p. 113648, 2021.
- [352] W. Verbeke, D. Olaya, M.-A. Guerry, and J. Van Belle, “To do or not to do? cost-sensitive causal classification with individual treatment effect estimates,” *European Journal of Operational Research*, vol. 305, no. 2, pp. 838–852, 2023.
- [353] H. M. Proença and F. Moraes, “Incremental profit per conversion: A response transformation for uplift modeling in e-commerce promotions,” *arXiv preprint arXiv:2306.13759*, 2023.

-
- [354] K. Zhao, J. Hua, L. Yan, Q. Zhang, H. Xu, and C. Yang, “A unified framework for marketing budget allocation,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1820–1830.
- [355] Y. Tu, K. Basu, C. DiCiccio, *et al.*, “Personalized treatment selection using causal heterogeneity,” in *Proceedings of the Web Conference 2021*, 2021, pp. 1574–1585.
- [356] M. Ai, B. Li, H. Gong, *et al.*, “Lbcf: A large-scale budget-constrained causal forest algorithm,” in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 2310–2319.
- [357] C. Fernández-Loría and J. Loría, “Learning the ranking of causal effects with confounded data,” *arXiv preprint arXiv:2206.12532*, 2022.
- [358] S. Athey, N. Keleher, and J. Spiess, “Machine learning who to nudge: Causal vs predictive targeting in a field experiment on student financial aid renewal,” *arXiv preprint arXiv:2310.08672*, 2023.
- [359] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, “Learning to rank: From pairwise approach to listwise approach,” in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 129–136.
- [360] T.-Y. Liu *et al.*, “Learning to rank for information retrieval,” *Foundations and Trends® in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.
- [361] I. Lyzhin, A. Ustimenko, A. Gulin, and L. Prokhorenkova, “Which tricks are important for learning to rank?” In *International Conference on Machine Learning*, PMLR, 2023, pp. 23 264–23 278.
- [362] S. Du, J. Lee, and F. Ghaffarizadeh, “Improve user retention with causal learning,” in *The 2019 ACM SIGKDD Workshop on Causal Discovery*, PMLR, 2019, pp. 34–49.
- [363] W. Y. Zou, S. Du, J. Lee, and J. Pedersen, “Heterogeneous causal learning for effectiveness optimization in user marketing,” *arXiv preprint arXiv:2004.09702*, 2020.
- [364] H. Zhou, S. Li, G. Jiang, J. Zheng, and D. Wang, “Direct heterogeneous causal learning for resource allocation problems in marketing,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023, pp. 5446–5454.
- [365] H. Zhou, R. Huang, S. Li, *et al.*, “Decision focused causal learning for direct counterfactual marketing optimization,” in *Proceedings of the 30th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2024.

- [366] Z. Yan, S. Wang, G. Zhou, J. Lin, and P. Jiang, “An end-to-end framework for marketing effectiveness optimization under budget constraint,” *arXiv preprint arXiv:2302.04477*, 2023.
- [367] B. He, Y. Weng, X. Tang, *et al.*, “Rankability-enhanced revenue uplift modeling framework for online marketing,” in *Proceedings of the 30th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2024.
- [368] M. Dudík, J. Langford, and L. Li, “Doubly robust policy evaluation and learning,” in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, 2011, pp. 1097–1104.
- [369] B. Zhang, A. A. Tsiatis, M. Davidian, M. Zhang, and E. Laber, “Estimating optimal treatment regimes from a classification perspective,” *Stat*, vol. 1, no. 1, pp. 103–114, 2012.
- [370] E. B. Laber and Y.-Q. Zhao, “Tree-based methods for individualized treatment regimes,” *Biometrika*, vol. 102, no. 3, pp. 501–514, 2015.
- [371] A. R. Luedtke and M. J. van der Laan, “Super-learning of an optimal dynamic treatment rule,” *The International Journal of Biostatistics*, vol. 12, no. 1, pp. 305–332, 2016.
- [372] T. Kitagawa and A. Tetenov, “Who should be treated? empirical welfare maximization methods for treatment choice,” *Econometrica*, vol. 86, no. 2, pp. 591–616, 2018.
- [373] P. Gutierrez and J.-Y. Gérardy, “Causal inference and uplift modelling: A review of the literature,” in *International Conference on Predictive Applications and APIs*, PMLR, 2017, pp. 1–13.
- [374] E. Sverdrup, H. Wu, S. Athey, and S. Wager, “Qini curves for multi-armed treatment rules,” *arXiv preprint arXiv:2306.11979*, 2023.
- [375] C. Burges, T. Shaked, E. Renshaw, *et al.*, “Learning to rank using gradient descent,” in *Proceedings of the 22nd International Conference on Machine learning*, 2005, pp. 89–96.
- [376] C. J. Burges, “From ranknet to lambdarank to lambdamart: An overview,” *Learning*, vol. 11, no. 23-581, p. 81, 2010.
- [377] P. Donmez, K. M. Svore, and C. J. Burges, “On the local optimality of lambdarank,” in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, 2009, pp. 460–467.
- [378] M. Jaskowski and S. Jaroszewicz, “Uplift modeling for clinical trial data,” in *ICML workshop on clinical data analysis*, vol. 46, 2012, pp. 79–95.

-
- [379] K. Rudaś and S. Jaroszewicz, “Linear regression for uplift modeling,” *Data Mining and Knowledge Discovery*, vol. 32, no. 5, pp. 1275–1305, 2018.
- [380] H. Bang and J. M. Robins, “Doubly robust estimation in missing data and causal inference models,” *Biometrics*, vol. 61, no. 4, pp. 962–973, 2005.
- [381] E. H. Kennedy, “Towards optimal doubly robust estimation of heterogeneous causal effects,” *Electronic Journal of Statistics*, vol. 17, no. 2, pp. 3008–3049, 2023.
- [382] X. Nie and S. Wager, “Quasi-oracle estimation of heterogeneous treatment effects,” *Biometrika*, vol. 108, no. 2, pp. 299–319, 2021.
- [383] P. M. Robinson, “Root-n-consistent semiparametric regression,” *Econometrica: Journal of the Econometric Society*, pp. 931–954, 1988.
- [384] K. Hillstrom, “Minethatdata e-mail analytics and data mining challenge,” *MineThatData blog*, 2008.
- [385] G. Ke, Q. Meng, T. Finley, *et al.*, “Lightgbm: A highly efficient gradient boosting decision tree,” *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [386] J. H. Friedman, “Stochastic gradient boosting,” *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002.
- [387] D. Sculley, “Combined regression and ranking,” in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010, pp. 979–988.
- [388] L. Yan, Z. Qin, X. Wang, M. Bendersky, and M. Najork, “Scale calibration of deep ranking models,” in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 4300–4309.
- [389] A. Bai, R. Jagerman, Z. Qin, *et al.*, “Regression compatible listwise objectives for calibrated ranking with binary relevance,” in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023, pp. 4502–4508.
- [390] T. Vanderschueren, B. Baesens, T. Verdonck, and W. Verbeke, “A new perspective on classification: Optimally allocating limited resources to uncertain tasks,” *Decision Support Systems*, vol. 179, p. 114 151, 2024.
- [391] V. Veitch and A. Zaveri, “Sense and sensitivity analysis: Simple post-hoc analysis of bias due to unobserved confounding,” *Advances in neural information processing systems*, vol. 33, pp. 10 999–11 009, 2020.

References

- [392] E. J. T. Tchetgen and T. J. VanderWeele, “On causal inference in the presence of interference,” *Statistical methods in medical research*, vol. 21, no. 1, pp. 55–75, 2012.
- [393] K. W. De Bock, K. Coussement, A. De Caigny, *et al.*, “Explainable ai for operational research: A defining framework, methods, applications, and a research agenda,” *European Journal of Operational Research*, vol. 317, no. 2, pp. 249–272, 2024.
- [394] D. Zha, Z. P. Bhat, K.-H. Lai, *et al.*, “Data-centric artificial intelligence: A survey,” *arXiv preprint arXiv:2303.10158*, 2023.
- [395] J. Crabbé, A. Curth, I. Bica, and M. van der Schaar, “Benchmarking heterogeneous treatment effect models through the lens of interpretability,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 12 295–12 309, 2022.
- [396] S. Goethals, D. Martens, and T. Calders, “Precof: Counterfactual explanations for fairness,” *Machine Learning*, vol. 113, no. 5, pp. 3111–3142, 2024.
- [397] C. Cortes, G. DeSalvo, and M. Mohri, “Learning with rejection,” in *Algorithmic Learning Theory: 27th International Conference, ALT 2016, Bari, Italy, October 19-21, 2016, Proceedings 27*, Springer, 2016, pp. 67–82.
- [398] H. Mozannar and D. Sontag, “Consistent estimators for learning to defer to an expert,” in *International conference on machine learning*, PMLR, 2020, pp. 7076–7087.
- [399] G. Petrides, D. Moldovan, L. Coenen, T. Guns, and W. Verbeke, “Cost-sensitive learning for profit-driven credit scoring,” *Journal of the Operational Research Society*, pp. 1–13, 2020.
- [400] L. Biewald, *Experiment tracking with weights and biases*, Software available from wandb.com, 2020. [Online]. Available: <https://www.wandb.com/>.
- [401] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2015.
- [402] J. Rothfuss, F. Ferreira, S. Walther, and M. Ulrich, “Conditional density estimation with neural networks: Best practices and benchmarks,” *arXiv preprint arXiv:1903.00954*, 2019.
- [403] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and accurate deep network learning by exponential linear units (elus),” *arXiv preprint arXiv:1511.07289*, 2015.

-
- [404] Y. Xu, Y. Xu, and S. Saria, “A bayesian nonparametric approach for estimating individualized treatment-response curves,” in *Machine learning for healthcare conference*, PMLR, 2016, pp. 282–300.
- [405] P. Schulam and S. Saria, “Reliable decision support using counterfactual models,” *Advances in neural information processing systems*, vol. 30, 2017.
- [406] J. Roy, K. J. Lum, and M. J. Daniels, “A bayesian nonparametric approach to marginal structural models for point treatments and a continuous or survival outcome,” *Biostatistics*, vol. 18, no. 1, pp. 32–47, 2017.
- [407] H. Soleimani, A. Subbaswamy, and S. Saria, “Treatment-response models for counterfactual reasoning with continuous-time, continuous-valued interventions,” *arXiv preprint arXiv:1704.02038*, 2017.
- [408] A. Bellot and M. Van Der Schaar, “Policy analysis using synthetic controls in continuous-time,” in *International Conference on Machine Learning*, PMLR, 2021, pp. 759–768.
- [409] Z. Qian, Y. Zhang, I. Bica, A. Wood, and M. van der Schaar, “Synctwin: Treatment effect estimation with longitudinal outcomes,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 3178–3190, 2021.
- [410] J. Robins, “A new approach to causal inference in mortality studies with a sustained exposure period—application to control of the healthy worker survivor effect,” *Mathematical modelling*, vol. 7, no. 9–12, pp. 1393–1512, 1986.
- [411] J. M. Robins, “Causal inference from complex longitudinal data,” in *Latent variable modeling and applications to causality*, Springer, 1997, pp. 69–117.
- [412] J. M. Robins, M. A. Hernan, and B. Brumback, *Marginal structural models and causal inference in epidemiology*, 2000.
- [413] A. M. Alaa, S. Hu, and M. Schaar, “Learning from clinical judgments: Semi-markov-modulated marked hawkes processes for risk prognosis,” in *International Conference on Machine Learning*, PMLR, 2017, pp. 60–69.
- [414] V. Jeanselme, G. Martin, N. Peek, M. Sperrin, B. Tom, and J. Barrett, “Deepjoint: Robust survival modelling under clinical presence shift,” *arXiv preprint arXiv:2205.13481*, 2022.
- [415] S. Yu, B. Krishnapuram, R. Rosales, and R. B. Rao, “Active sensing,” in *Artificial Intelligence and Statistics*, PMLR, 2009, pp. 639–646.

- [416] J. Yoon, W. R. Zame, and M. Van Der Schaar, “Deep sensing: Active sensing using multi-directional recurrent neural networks,” in *International Conference on Learning Representations*, 2018.
- [417] A. M. Alaa and M. Van Der Schaar, “Balancing suspense and surprise: Timely decision making with endogenous information acquisition,” *Advances in Neural Information Processing Systems*, vol. 29, 2016.
- [418] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, “Neural ordinary differential equations,” *Advances in neural information processing systems*, vol. 31, 2018.
- [419] Y. Rubanova, R. T. Chen, and D. K. Duvenaud, “Latent ordinary differential equations for irregularly-sampled time series,” *Advances in neural information processing systems*, vol. 32, 2019.
- [420] J. Morrill, P. Kidger, L. Yang, and T. Lyons, “Neural controlled differential equations for online prediction tasks,” *arXiv preprint arXiv:2106.11028*, 2021.
- [421] R. J. Little and D. B. Rubin, *Statistical analysis with missing data*. John Wiley & Sons, 2019, vol. 793.
- [422] I. Mayer, E. Sverdrup, T. Gauss, J.-D. Moyer, S. Wager, and J. Josse, “Doubly robust treatment effect estimation with missing attributes,” *Annals of Applied Statistics*, vol. 14, no. 3, pp. 1409–1431, 2020.
- [423] S. R. Cole and M. A. Hernán, “Constructing inverse probability weights for marginal structural models,” *American journal of epidemiology*, vol. 168, no. 6, pp. 656–664, 2008.
- [424] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, “Recurrent neural networks for multivariate time series with missing values,” *Scientific reports*, vol. 8, no. 1, pp. 1–12, 2018.
- [425] S. Feuerriegel, D. Frauen, V. Melnychuk, *et al.*, “Causal machine learning for predicting treatment outcomes,” *Nature Medicine*, vol. 30, no. 4, pp. 958–968, 2024.
- [426] A. Franks, A. D’Amour, and A. Feller, “Flexible sensitivity analysis for observational studies without observable implications,” *Journal of the American Statistical Association*, 2020.
- [427] V. S. Lo, “The true lift model: A novel data mining approach to response modeling in database marketing,” *ACM SIGKDD Explorations Newsletter*, vol. 4, no. 2, pp. 78–86, 2002.
- [428] S. Powers, J. Qian, K. Jung, *et al.*, “Some methods for heterogeneous treatment effect estimation in high dimensions,” *Statistics in Medicine*, vol. 37, no. 11, pp. 1767–1787, 2018.

-
- [429] D. G. Horvitz and D. J. Thompson, “A generalization of sampling without replacement from a finite universe,” *Journal of the American statistical Association*, vol. 47, no. 260, pp. 663–685, 1952.
- [430] S. Athey and G. W. Imbens, “Machine learning methods for estimating heterogeneous causal effects,” *stat*, vol. 1050, no. 5, pp. 1–26, 2015.
- [431] C. A. Rolling and Y. Yang, “Model selection for estimating treatment effects,” *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 76, no. 4, pp. 749–769, 2014.
- [432] K. Larsen, *Information: Data exploration with information theory methods*, R package version 0.2.1, Accessed: 2024-09-26, 2023. [Online]. Available: <https://cran.r-project.org/package=Information>.
- [433] A. Sharma and E. Kiciman, “Dowhy: An end-to-end library for causal inference,” *arXiv preprint arXiv:2011.04216*, 2020.
- [434] T. Geffner, J. Antoran, A. Foster, *et al.*, “Deep end-to-end causal inference,” *arXiv preprint arXiv:2202.02195*, 2022.
- [435] I. Teinemaa, J. Albert, and N. Pham, *UpliftML: A Python Package for Scalable Uplift Modeling*, <https://github.com/bookingcom/upliftml>, Version 0.0.1, 2021.
- [436] M. A. Hernán and J. M. Robins, “Estimating causal effects from epidemiological data,” *Journal of Epidemiology & Community Health*, vol. 60, no. 7, pp. 578–586, 2006.

Part VI

APPENDICES



A

**PREDICT-THEN-OPTIMIZE OR
PREDICT-AND-OPTIMIZE? AN
EMPIRICAL EVALUATION OF
COST-SENSITIVE LEARNING
STRATEGIES**

A.1 Data

The data sets that are used in the experiments presented in this paper are publicly available online (names are clickable links for the online version):

- [Kaggle Credit Card Fraud \[107\]](#)
- [Kaggle IEEE Fraud Detection](#)
- [UCI KDD98 Direct Mailing](#)
- [UCI Bank Marketing \[106\]](#)
- [Kaggle Telco Customer Churn](#)
- [TV Subscription Churn \[59\]](#)
- [Kaggle Give Me Some Credit](#)
- [UCI Default of Credit Card Clients \[105\]](#)
- [VUB Credit Scoring \[399\]](#)

A.2 Training with instance-dependent or class-dependent costs: results per dataset

Detailed results comparing training with instance-dependent and class-dependent costs per dataset can be found in Tables B1 and B2.

A.2. Instance- or class-dependent cost training: additional results

Table B1: **Instance-dependent or class-dependent costs: cost-insensitive metrics per dataset.** Significantly better results are denoted in **bold** (5%) and *italic* (10%).

Metric	Costs	KCCF	GMSC	KIFD	KTCC	KDD	TSC	UBM	DCCC	VCS
AUC	ID	0.96	0.81	0.89	0.82	0.51	0.61	0.73	0.72	0.76
	CD	0.96	0.81	<i>0.90</i>	0.82	0.53	0.62	0.76	<i>0.75</i>	<i>0.77</i>
AP	ID	0.72	0.30	0.45	0.61	0.05	0.08	0.29	0.46	0.38
	CD	0.77	0.31	<i>0.51</i>	0.60	0.06	0.08	<i>0.37</i>	0.49	0.39
Brier score	ID	0.00	0.16	0.05	0.25	0.40	0.19	0.17	0.21	0.24
	CD	0.00	0.17	0.06	0.25	0.42	0.19	0.16	0.23	0.25
F1 IDCS	ID	0.41	0.22	0.27	0.54	0.10	0.12	0.31	0.43	0.40
	CD	<i>0.46</i>	0.22	0.28	<i>0.55</i>	0.10	0.12	0.33	<i>0.44</i>	0.40
F1 IDCS*	ID	0.40	0.30	0.28	0.57	0.06	0.12	0.28	0.37	0.40
	CD	<i>0.49</i>	<i>0.31</i>	0.38	0.57	0.06	0.12	<i>0.31</i>	<i>0.44</i>	0.43
F1 CDCS	ID	0.63	0.22	0.27	0.54	0.10	0.12	0.32	0.44	0.40
	CD	0.74	0.22	0.27	<i>0.55</i>	0.10	<i>0.12</i>	0.32	0.45	0.39
F1 CDCS*	ID	0.75	0.31	0.40	0.57	0.10	0.13	0.36	0.48	0.43
	CD	0.81	0.31	0.45	0.57	0.10	0.14	0.42	0.51	0.44
F1 Emp ID	ID	0.70	<i>0.32</i>	0.40	0.56	0.10	0.13	0.36	0.49	0.44
	CD	0.81	0.31	0.45	0.57	0.10	<i>0.14</i>	0.42	<i>0.51</i>	0.44
F1 Emp CD	ID	0.75	0.31	0.40	0.57	0.10	0.13	0.36	0.48	0.43
	CD	0.81	0.31	0.45	0.57	0.10	<i>0.14</i>	0.42	0.51	<i>0.44</i>
F1 Emp F1	ID	0.77	0.39	0.49	0.61	0.10	0.13	0.37	0.49	0.44
	CD	0.82	0.39	0.54	0.61	<i>0.10</i>	0.13	0.43	<i>0.53</i>	0.46
F1 CI	ID	0.50	0.22	0.24	0.56	0.10	0.12	0.31	0.45	0.40
	CD	0.53	0.22	0.23	<i>0.57</i>	0.10	0.12	0.31	0.45	0.40
F1 0.5	ID	0.74	0.33	0.43	0.59	0.10	0.13	0.37	0.49	0.44
	CD	0.81	0.33	0.47	0.59	<i>0.10</i>	0.14	<i>0.41</i>	<i>0.51</i>	<i>0.45</i>

Appendix A: An Empirical Evaluation of Cost-sensitive Learning Strategies

Table B2: **Instance-dependent or class-dependent costs: cost-sensitive metrics per dataset.** Significantly better results are denoted in **bold** (5%) and *italic* (10%).

Metric	Costs	KCCF	GMSC	KIFD	KTCC	KDD	TSC	UBM	DCCC	VCS
AEC	ID	0.08	458.90	<i>2.53</i>	82.05	<i>0.72</i>	60.22	0.52	15674.65	<i>0.08</i>
	CD	0.08	460.81	3.05	81.32	0.72	60.42	0.67	16724.90	0.09
Spearman's ρ	ID	0.17	-0.04	<i>0.09</i>	0.12	0.03	-0.35	<i>0.55</i>	<i>0.05</i>	0.36
	CD	-0.07	-0.15	-0.17	0.12	-0.14	-0.30	0.18	-0.30	0.11
Savings IDCS	ID	0.54	<i>0.24</i>	0.47	0.21	-0.01	-0.09	0.57	0.29	<i>0.36</i>
	CD	0.68	0.23	0.42	0.22	-0.01	-0.09	0.51	0.21	0.34
Savings IDCS*	ID	0.66	0.47	0.60	0.26	<i>-0.08</i>	0.07	0.61	0.32	0.42
	CD	<i>0.70</i>	0.48	0.61	0.26	-0.09	0.06	0.62	0.36	<i>0.43</i>
Savings CDCS	ID	0.62	0.23	0.38	0.21	-0.01	-0.10	0.45	0.22	<i>0.34</i>
	CD	0.65	0.22	0.28	0.22	-0.01	-0.10	0.26	0.16	0.30
Savings CDCS*	ID	0.67	<i>0.47</i>	<i>0.59</i>	0.26	<i>-0.01</i>	0.06	0.55	<i>0.35</i>	<i>0.41</i>
	CD	0.67	0.47	0.50	0.26	-0.01	0.06	0.42	0.29	0.38
Savings Emp ID	ID	0.67	<i>0.47</i>	0.59	0.26	-0.01	0.06	0.56	0.35	0.42
	CD	0.67	0.47	0.50	0.26	-0.02	0.06	0.42	0.29	0.38
Savings Emp CD	ID	0.67	0.47	<i>0.59</i>	0.26	<i>-0.01</i>	0.06	0.55	<i>0.35</i>	<i>0.41</i>
	CD	0.67	0.47	0.50	0.26	-0.02	0.06	0.42	0.29	0.38
Savings Emp F1	ID	0.66	0.40	<i>0.52</i>	0.10	-0.03	0.05	<i>0.54</i>	0.35	0.39
	CD	0.64	0.39	0.41	0.10	-0.08	0.05	0.38	0.30	0.33
Savings CI	ID	-2.58	0.23	0.24	0.22	-0.01	-0.10	0.43	0.25	<i>0.35</i>
	CD	-3.05	0.23	0.14	0.23	-0.02	-0.10	0.24	0.18	0.31
Savings 0.5	ID	0.66	<i>0.47</i>	<i>0.59</i>	0.20	<i>-0.03</i>	0.06	0.55	0.35	0.41
	CD	0.66	0.47	0.49	0.21	-0.05	0.06	0.43	0.30	0.37

B

OPTIMIZING THE PREVENTIVE MAINTENANCE FREQUENCY WITH CAUSAL MACHINE LEARNING

B.1 Hyperparameter optimization

To make our work more transparent and facilitate the application of our approach, we provide more information regarding the training and hyperparameter optimization of the neural networks used in this work. Table B1 shows training settings and ranges for the different hyperparameters that were searched over, differentiating between general hyperparameters, hyperparameters for the GAN, and hyperparameters for the MLP. For the MLP and MLP-ITE benchmarks, only the general and MLP hyperparameters were searched over. For all models, hyperparameter optimization was done using grid search based on the mean squared error on the observed outcomes in the validation set. For more details regarding SCIGAN’s training and optimization, we refer to [169] and the accompanying repository available at <https://github.com/ioanabica/SCIGAN>.

Table B1: **Model training.** We show the training settings and hyperparameter ranges that were searched, differentiating between general, GAN-related and MLP-related hyperparameters.

Name	Range
General	
Batch size	[32, 64]
Optimizer	Adam
Learning rate	0.001
GAN	
Hidden neurons	[16, 32]
Dosage samples	2
Training iterations	50,000
MLP	
Hidden neurons	[32, 64]
Training iterations	10,000

C

NOFLITE: LEARNING TO PREDICT INDIVIDUAL TREATMENT EFFECT DISTRIBUTIONS

C.1 Data sets and associated data generating processes

This section gives more extensive details on the different semi-synthetic data sets used in this work. We describe each data set and the underlying data generating process that were used to simulate it.

IHDP [$n = 747, d = 25$; **205**]. We use data based on the standard response surface B. In this setting, potential outcomes are generated as

$$Y_i^{(0)} = \exp((x_i + A)\beta) + \varepsilon \quad \text{and} \quad Y_i^{(1)} = x_i\beta - \omega + \varepsilon \quad (\text{C.1})$$

where $\varepsilon \sim \mathcal{N}(0, 1)$, W is a (fixed) offset matrix with each value equal to 0.5, β is a sparse coefficient vector with each element sampled from $(0, 0.1, 0.2, 0.3, 0.4)$ with corresponding probabilities $(0.6, 0.1, 0.1, 0.1, 0.1)$. For each iteration of the data set, ω is set to ensure that the conditional average treatment effect for the treated (CATT) and conditional average treatment effect on the controls (CATC) both equal 4 on average. We use the 100 replications from github.com/clinicalml/cfrnet [201].

EDU [$n = 8,627, d = 32$; **219**]. For each treatment group, a neural network $f_{y^{(t)}}$ is trained based on the observed outcomes. The potential outcomes are then simulated as

$$Y_i^{(0)} = f_{y^{(0)}}(x_i) + (2 - x_i^{23})\varepsilon_0 \quad \text{and} \quad Y_i^{(1)} = f_{y^{(1)}}(x_i) + (2 - x_i^{23})\varepsilon_1, \quad (\text{C.2})$$

with $\varepsilon_0 \sim \mathcal{N}(0, 0.5^2)$ and $\varepsilon_1 \sim \exp(2)$. x^{23} refers to the 23rd covariate, which is a binary covariate indicating whether the instance’s mother has received previous education.

News [$n = 5,000, d = 3,477$; **200**]. First, based on a topic model z , two treatment-specific centroids z_0^c and z_1^c are estimated in the topic space $z(x)$. Each potential outcome is then generated as the similarity between $z(x_i)$ and z_t^c as

$$Y_i^{(t)} = C(z(x_i)^\top z_0^c + t_i \cdot z(x_i)^\top z_1^c) + \varepsilon,$$

with scaling factor C and $\varepsilon \sim \mathcal{N}(0, 1)$. Treatment assignment is modeled as $p(t_i = 1 | x_i) = \frac{\exp(\kappa \cdot z(x_i)^\top z_1^c)}{\exp(\kappa \cdot z(x_i)^\top z_1^c) + \exp(\kappa \cdot z(x_i)^\top z_0^c)}$ with $\kappa = 10$.

C.2 Hyperparameter optimization

We add more information on the chosen hyperparameters in Table B1. Given the absence of the ground truth in observational data, hyperparameter tun-

ing was done using wandb [400] using the potential outcome’s loglikelihood, PEHE, and IoU of a validation set. Additionally, to tune the hyperparameter λ for the de-biasing term, we use an inverse propensity weighted loglikelihood, allowing us to assess performance under covariate shift. Training is done using gradient descent with the Adam optimizer [401]. We use several types of regularization: ℓ_1 , ℓ_2 , and noise regularization [402]. The encoder networks uses exponential linear units (ELU) as activation functions [403].

Table B1: **Hyperparameter tuning.** We show the optimal hyperparameters for the different data sets. The flow type ‘SigmoidX’ refers to the deep sigmoidal flow (DSF) of [258] conditioned on the balanced representation ϕ of x . During inference, drawing samples from a truncated normal distribution was observed to be more stable for the News data.

Hyperparameter	IHDP	EDU	News
— <i>General</i> —			
Metalearner	T	T	T
— <i>Encoder</i> —			
Hidden layers balancer	2	1	3
Hidden layers encoder – shared	3	0	0
Hidden layers encoder – separate	2	2	3
Hidden neurons encoder	8	8	32
— <i>Flow</i> —			
Number of flow transformations k	0	4	1
Flow type	—	SigmoidX	SigmoidX
Hidden neurons transformer	—	4	2
Hidden neurons conditioner	—	16	32
Hidden layers conditioner	—	2	1
— <i>Training settings</i> —			
Learning rate	5e-4	5e-4	5e-4
Batch size	128	512	128
Training steps	5,000	5,000	10,000
Regularization λ_{ℓ_1}	1e-3	0	5e-4
Regularization λ_{ℓ_2}	5e-4	1e-3	5e-3
λ_{mmd}	1	1e-2	1e-2
Noise regularization x	0	0	1
Noise regularization y	5e-1	1e-1	5e-1
Truncation probability	0	0	1e-2

Appendix C: Learning to Predict Individual Treatment Effect Distributions

The optimal hyperparameters additionally give some insights into how `NOFLITE` can deal with a diversity of data generating processes. For instance, the best metalearner was the T-learner for all data sets. The potential outcomes were generated separately for all data sets under consideration, which corresponds more closely to the T-learner’s hypothesis space. Moreover, the complexity of the distribution can be tweaked depending on the data generating process: `NOFLITE` does not use any flows for IHDP, uses only one flow for News, and uses four flow transformations for EDU. These settings match the increasingly more complex distributions being used in the corresponding the data generating processes.

D

ACCOUNTING FOR INFORMATIVE SAMPLING WHEN LEARNING TO FORECAST TREATMENT OUTCOMES OVER TIME

D.1 Extended Related Work

This section provides a more extensive discussion of several related areas of work.

D.1.1 Forecasting treatment effects over time

There is a growing interest in the ML literature in estimating personalized treatment effects over time. To this aim, different types of neural networks have been explored, including RNNs [272], [285]–[287], transformers [288], and Neural ODEs [289], [290] or Neural CDEs [291]. All existing work in this area has implicitly relied upon assumptions of the observation process, assuming regular observations or completely random observation intervals, see Table B1. Conversely, our work relies on the less strict SAR assumption.

Table B1: An overview of existing work. We categorize the related work according to the assumptions made regarding the observation process. Sampling is either assumed to be regular, completely at random (SCAR), or at random (SAR).

Reference	Sampling
Lim, Alaa, and Schaar [285]	Regular
Bica, Alaa, Jordon, <i>et al.</i> [272]	Regular
Berrevoets, Curth, Bica, <i>et al.</i> [287]	Regular
Li, Shahn, Li, <i>et al.</i> [286]	Regular
Melnichuk, Frauen, and Feuerriegel [288]	Regular
Gwak, Sim, Poli, <i>et al.</i> [289]	SCAR
Seedat, Imrie, Bellot, <i>et al.</i> [291]	SCAR
De Brouwer, Gonzalez, and Hyland [290]	SCAR
This work	SAR

In addition to existing work leveraging neural networks, there is another line of work in the ML literature that uses Gaussian processes to estimate treatment effects over time in the presence of irregular samples [404]–[407]. Other work has looked at using synthetic controls to estimate the effect of a (single) intervention over time [408], [409]. Similar to the ML literature on estimating treatment effects over time using neural networks discussed above, these also rely on regular samples or SCAR. To the best of our knowledge, no existing work in the ML literature has studied the problem of estimating treatment effects given SAR, which is the focus of this work.

Outside the ML literature, several approaches have been proposed for estimating *average* treatment effects over time, most notably the seminal

work using g -computation and marginal structural models [410]–[412]. More specifically, several approaches have been proposed in the (bio)statistics literature to learn causal effects under informative sampling, see Gasparini, Abrams, Barrett, *et al.* [280] for an overview. Existing work on estimating causal effects under SAR can be categorized based on (1) whether they use inverse visiting weights or random effects, and (2) whether they assume discrete or continuous time. (1) The first group uses the *inverse probability of visiting* or its continuous-time equivalent, the inverse intensity of visiting, as weights in the objective function of the estimator [274], [277], [294]. (2) The second uses *shared random effects* to jointly model the observation and outcome processes [279]. However, these approaches assume a certain parametric form or latent variable(s), which might not reflect the actual (unknown) data generating procedure and usually focus on population average effects. Conversely, our approach is conceptually similar to inverse intensity of visit weighting, but leverages the use of flexible ML methods that do not require these assumptions.

D.1.2 Informative sampling in ML

Various other works in ML consider related problem settings. For example, informative sampling has been considered as a source of information for prognosis in health care [413] and as a challenge to robustness of predictive models to distribution shifts [414], though this line of work does not consider the estimation of treatment effects. Moreover, the literature on active sensing views takes observing as an active role in which the decision-maker controls the sampling mechanism [415]. The key question addressed in this line of work is what and when to measure [416] and, potentially, also when to stop measuring [417]. This is in contrast to our setting, where data is not actively sampled, but *observed* over time and the observer has a passive role.

D.1.3 Neural ODEs

Neural ordinary differential equations (ODEs) have recently emerged as a novel class of machine learning models combining neural networks and differential equations [418]. Due to their ability of handling irregular observations, Neural ODEs have been applied for time series, either directly or combined with a recurrent neural network [e.g., 419]. Neural controlled differential equations (CDEs) additionally allow for modeling covariates as a control, making them suitable for dealing with irregularly sampled time series [303], [420], as in the setting considered in this work.

D.1.4 Missing data

Dealing with missing data is an important and established field in statistics and machine learning [295], [421]. This literature is related to our setting, as we are interested in learning a continuous latent path based on irregular observations over time, which could also be seen as a form of missing data imputation. Moreover, the sampling mechanisms considered in this work are similar to missing data mechanisms. Several recent works explore dealing with missing data in the context of treatment effect estimation [329], [422].

D.2 List of Mathematical Symbols

We compile a list of mathematical symbols and their explanation in Table B2. Moreover, we use a real-world example of a health care application to illustrate their meaning.

Table B2: **List of symbols.** We compile a list of the main mathematical symbols used and their explanation. The final column illustrates each symbol for the case of a cancer patient.

Symbol	Explanation	Cancer patient example
X	Covariate path	Blood pressure, heart rate, etc.
A	Treatment path	Chemotherapy, radiotherapy, etc.
Y	Outcome path	Tumor size
t	Time	
$N(t)$	Counting process over time	Five observations after two weeks
$\lambda(t)$	Observation intensity over time	Probability of observing tumor size at time t
$\mu_{a,t}(\tau)$	Expected outcome at time $t + \tau$ given treatment path a	Tumor size next week absent any treatment
$\mu_{a,t}(\tau)$	Expected outcome at time $t + \tau$ given treatment path a	Tumor size next week absent any treatment
$\hat{y}_{i,t}(t')$	Instance i 's predicted treatment outcome at time $t' = t + \tau$	Patient i 's tumor size next week
$\hat{\lambda}_{i,t}(t')$	Instance i 's predicted observation intensity at time $t' = t + \tau$	Patient i 's observation intensity next week

D.3 TESAR-CDE: Multitask Training Procedure

We include a more detailed training procedure for the multitask configuration of TESAR-CDE in Algorithm 1.

Algorithm 2: Pseudo-code for the TESAR-CDE (Multitask) training procedure

Input: Observational data $\mathcal{D} = \{t_j^{(i)}, x_{t_j}^{(i)}, a_{t_j}^{(i)}, y_{t_j}^{(i)}\}$ for $i \in \{0, \dots, n\}$ and $j \in \{0, \dots, m_i\}$, weighted MSE loss $\mathcal{L}^{\text{WMSE}}$, cross-entropy loss \mathcal{L}^{CE} , total epochs E , learning rate η , and batch size b . TESAR-CDE architecture consisting of four networks: an embedding network g with weights W_g , an encoder CDE function f_θ with weights W_θ , a decoder CDE function f_ϕ with weights W_ϕ , a final intensity map f_ψ^λ with weights W_ψ^λ , and a final outcome map f_ψ^y with weights W_ψ^y .

for epochs = 1 to E **do**

 Sample batch $i_0, i_1, \dots, i_b \subset \{0, \dots, n\}$

 Encode the first observation $z(t_0)^{(i)} = g(t_0^{(i)}, x_{t_0}^{(i)}, a_{t_0}^{(i)}, y_{t_0}^{(i)})$ for each i in batch

 Encode the history up to time t :

$z(t) = \text{ODESolve}(f_\theta, z(t_0), \bar{X}(t), \bar{A}(t), \bar{Y}(t))$

 Decode the history up to time $t + \tau$:

$z_t(t + \tau) = \text{ODESolve}(f_\theta, z(t), \bar{A}_t(t + \tau))$

 Map to forecast the outcome at $t + \tau$: $\bar{y}_t(t + \tau) = f_\psi^y(z_t(t + \tau))$

 Map to forecast the intensity at $t + \tau$: $\bar{\lambda}_t(t + \tau) = f_\psi^\lambda(z_t(t + \tau))$

 Compute $\text{grad}_g = \nabla_{W_g} \frac{1}{n} \sum_i \mathcal{L}_i^{\text{WMSE}}$

 Compute $\text{grad}_\theta = \nabla_{W_\theta} \frac{1}{n} \sum_i \mathcal{L}_i^{\text{WMSE}}$

 Compute $\text{grad}_\phi = \nabla_{W_\phi} \frac{1}{n} \sum_i \mathcal{L}_i^{\text{WMSE}}$

 Compute $\text{grad}_{\psi^y} = \nabla_{W_{\psi^y}} \frac{1}{n} \sum_i \mathcal{L}_i^{\text{WMSE}}$

 Compute $\text{grad}_{\psi^\lambda} = \nabla_{W_{\psi^\lambda}} \frac{1}{n} \sum_i \mathcal{L}_i^{\text{CE}}$

 Update weights $W_g \leftarrow W_g - \eta \text{grad}_g$

 Update weights $W_\theta \leftarrow W_\theta - \eta \text{grad}_\theta$

 Update weights $W_\phi \leftarrow W_\phi - \eta \text{grad}_\phi$

 Update weights $W_{\psi^y} \leftarrow W_{\psi^y} - \eta \text{grad}_{\psi^y}$

 Update weights $W_{\psi^\lambda} \leftarrow W_{\psi^\lambda} - \eta \text{grad}_{\psi^\lambda}$

if $\frac{1}{n} \sum_i \mathcal{L}_i^{\text{MT}} = \frac{1}{n} \sum_i ((1 - \alpha)\mathcal{L}_i^{\text{WMSE}} + \alpha\mathcal{L}_i^{\text{CE}})$ did not improve for 50 epochs **then**

 Break {Early stopping}

end if

end for

D.4 TESAR-CDE: Implementation

This section provides more details on the implementation of TESAR-CDE.

D.4.1 Weight truncation

For more stable training, we truncate the estimated intensities at $c_{\min} = 0.001$, such that the maximal importance weight is equal to 1000. This is similar to what is typically done with propensity scores when adjusting for confounding bias. The truncation constant c_{\min} allows for trading off bias and variance, with $c_{\min} = 1$ corresponding to the unweighted variant [423]. We did not tune the cutoff rate c_{\min} .

D.4.2 Hyperparameter optimization

To allow for a fair comparison between the models, we do not tune hyperparameters for each model separately, but rather find the best configuration for the baseline TE-CDE model at a level of informativeness $\gamma = 0$ and use this for all models. We show the ranges and final values for all hyperparameters in Table B3. Hyperparameter optimization was done using wandb’s Bayesian optimization [400]. For each network in the CDE (f_{θ} and f_{ϕ}), we use a final tanh activation layer, as recommended by Kidger, Morrill, Foster, *et al.* [303]. All models are trained with a batch size of 128 and learning rate of $5e - 4$ for a maximum of 1000 epochs. Learning was terminated if the training loss did not improve for 50 epochs. For the multitask configuration, we use $\alpha = 0.8$ to balance the loss terms, though this is only used for early stopping as each part of the network has a different optimizer, see also ?? 2. For all models, we construct a control path for the Neural CDEs using a cubic interpolation.

Table B3: **Hyperparameter optimization.** We show the range for each hyperparameter that was optimized. The optimal value is shown in bold.

Parameter	Range
Latent state z dimension	{8, 16, 32 }
Encoder layers	{1, 2, 3 }
Decoder layers	{1, 2 , 3}
Map layers	{ 1 , 2}
Encoder hidden neurons	{4, 8 , 16}
Decoder hidden neurons	{4, 8 , 16}
Map hidden neurons	{4, 8 , 16}

D.4.3 A note on adding counts

A frequent practice in time series forecasting when observation times may be informative is to add observation counts to the data [303], [424]. However, in the context of estimating treatment effects, this is problematic. First, adding counts is complicated because estimating counterfactual treatments would then require counterfactual count data, which is not observed. Moreover, adding count data may itself introduce confounding or collider bias [273]. Therefore, we do not add observation counts in this work.

D.5 Tumor Growth Simulation

We use the tumor growth simulation of Geng, Paganetti, and Grassberger [310], which was also used in the previous ML literature on estimating treatment effects over time [e.g., 272], [285], [288], [291]. We refer to these works for more details.

The tumor size is modelled as:

$$\frac{dY(t)}{dt} = \left[\underbrace{1 + \rho \log\left(\frac{K}{Y(t)}\right)}_{\text{Tumor growth}} - \underbrace{\beta_c C_t}_{\text{Chemotherapy}} - \underbrace{(\alpha_r d(t) + \beta_r d(t)^2)}_{\text{Radiotherapy}} + \underbrace{\epsilon_t}_{\text{Noise}} \right] Y(t).$$

Parameters are obtained as follows. Carrying capacity K is set equal to 30. Growth parameter ρ is sampled from a normal distribution $\rho \sim \mathcal{N}(7.00 \times 10^{-5}, 7.23 \times 10^{-3})$. β_c is also sampled from a normal distribution $\beta_c \sim \mathcal{N}(0.028, 0.0007)$. Finally, α_r and β_r are obtained as $\alpha_r \sim \mathcal{N}(0.0398, 0.168)$ and $\beta_r = \frac{\alpha_r}{10}$. Noise is added by sampling $\epsilon(t) \sim \mathcal{N}(0, 0.01)$.

Following earlier work [272], [285], [291], we create heterogeneity in the treatment effects by creating three patient groups. Patient group 1 has a larger radiotherapy effect, achieved by multiplying $\mu(\alpha_r)$ with 1.1. Similarly, patient group 3 has a larger chemotherapy effect by increasing α_c with 10%.

We consider two types of treatment plans: a sequential and a concurrent plan [307]–[309]. We show simulated tumor paths and intensities for several patients in Figure D.1. For the informative observation process Equation (6.7), we visualize the intensity distribution at different levels of γ in Figures D.2 and D.3. For the uninformative observation process, we show the intensity distributions in Figure D.4.

For all experiments, we generate patient trajectories over 120 days. For training, we split the data for forecasting to have a lookback window of seven days and a maximum forecasting horizon of five days.

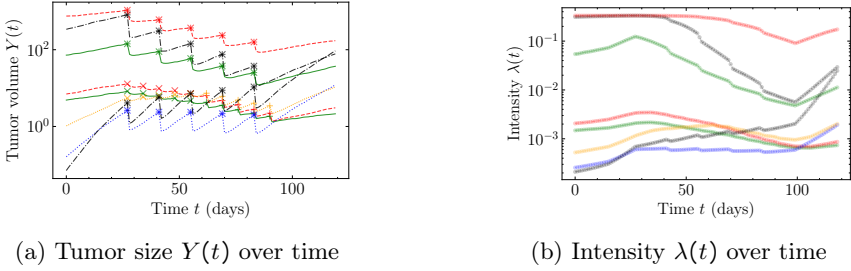


Figure D.1: **Tumor and intensity evolution.** We show the simulated tumor size and corresponding intensity over time for several (randomly selected) patients. The intensity is simulated based on Equation (6.7) with an informativeness $\gamma = 4$.

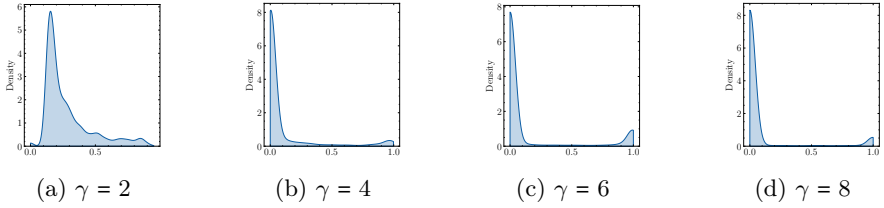


Figure D.2: **Informative sampling – intensity distribution.** We show the distribution of intensities $\lambda(t)$ over all patients for different levels of informativeness γ . At $\gamma = 0$ (not shown), all intensities are equal $\lambda_i(t) = 0.5$.

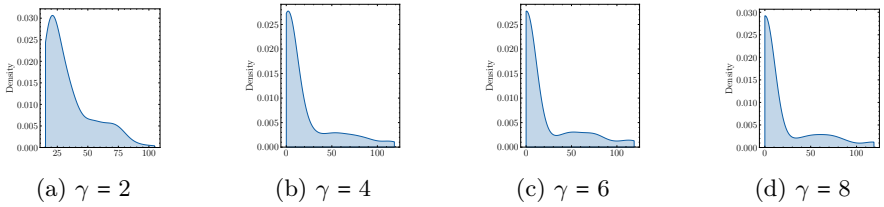


Figure D.3: **Informative sampling – expected observations.** We show the expected observations over the entire time period considered over all patients for different levels of informativeness γ . At $\gamma = 0$ (not shown), all intensities are equal $\lambda_i(t) = 0.5$ and all patients have 60 expected observations.

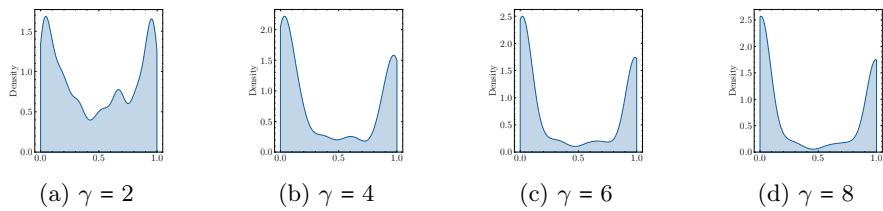


Figure D.4: **Uninformative sampling – intensity distribution.** We show the distribution of intensities $\lambda(t)$ over all patients for different levels of “informativeness” γ . At $\gamma = 0$ (not shown), all intensities are equal $\lambda_i(t) = 0.5$.

D.6 Additional Results

In this section, we present additional results to further validate the proposed TESAR-CDE. First, we evaluate the predicted observation intensities. Second, we analyze the sensitivity of the multitask model to hyperparameter α .

We evaluate how accurate TESAR-CDE predicts the observation intensities in terms of the Brier Score: $BS = \sum_{t=0}^T \sum_{\tau=0}^{\tau_{\max}} (\lambda_i(t + \tau) - \hat{\lambda}_{i,t}(t + \tau))^2$, see Figure D.5. Generally, both versions of our method can learn to accurately predict the observation intensities, with the two-step TESAR-CDE performing slightly better than the multitask configuration. These findings are consistent with our motivation of the multitask setup: while the two-step model learns a (generally better) model of the intensity itself using all available information, these more accurate intensities do not help with potential outcome prediction. Additionally, we find that the Brier score increases with informativeness for both models. This trend indicates that more informativeness makes it harder to learn to predict the observation intensity. We hypothesize that this is due to observing in general becoming more rare as increases.

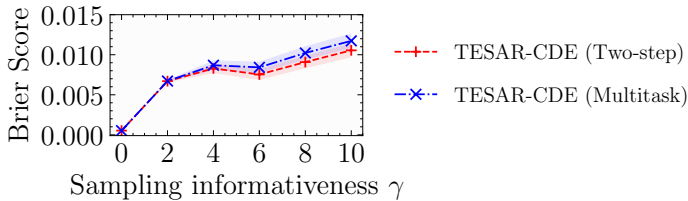


Figure D.5: **Evaluating the intensity prediction at varying informativeness γ .** We show the Brier Score \pm SE (lower is better) over ten runs at increasing levels of informativeness γ , keeping the forecasting horizon $\tau = 1$.

Next, we evaluate performance for the multitask TESAR-CDE for different values of its hyperparameter α , see Figure D.6. As the shared representation is only trained for outcome prediction, the only point of having the hyperparameter is to scale both terms such that they roughly influence the early stopping in the same way. We see that our method is robust to different values of this hyperparameter and that scaling them to approximately the same magnitude (using 0.8) results in good performance in practice. (Note that the loss terms are weighted by α and $(1 - \alpha)$, which is why we restrict to be strictly between 0 and 1.)

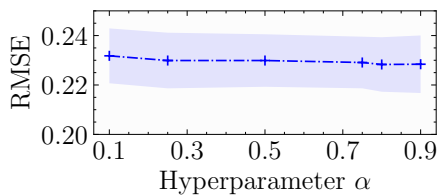


Figure D.6: **Evaluating the multitask configuration’s outcome prediction for different levels of hyperparameter α .** We show the RMSE \pm SE over ten runs, while keeping the level of informativeness fixed at $\gamma = 6$ and averaging over $\tau \in \{1, \dots, 5\}$.

E

AutoCATE: TOWARDS END-TO-END,
AUTOMATED TREATMENT EFFECT
ESTIMATION

The appendix starts with a more detailed introduction and background to CATE estimation in Appendix E.1. The next sections provide more details on `AutoCATE` (Appendix E.2), describe the data sets used in this work (Appendix E.3), and present additional empirical results (Appendix E.4). Finally, we compare `AutoCATE` with other packages for CATE estimation in Appendix E.5.

E.1 Background on CATE Estimation

This section provides a more detailed introduction and background on treatment effect estimation. In accordance to the main body, we denote an instance by a tuple (x, t, y) , with covariates $X \in \mathcal{X} \subset \mathbb{R}^d$, a treatment $T \in \mathcal{T} = \{0, 1\}$, and an outcome $Y \in \mathcal{Y} \subset \mathbb{R}$. Following the potential outcomes framework [163], [181], we describe an instance’s potential outcome Y for a given treatment $T = t$ as $Y(t)$. The Conditional Average Treatment Effect (CATE) is then defined as the expected difference in outcomes between treating and not treating:

$$\mathbb{E}\left[Y(1) - Y(0)|X\right]. \tag{E.1}$$

Knowing this effect is crucial in a variety of domains, such as education [199], healthcare [425], and maintenance [10]. Estimating the CATE from observational data involves significant *challenges* (Appendix E.1.1), requires standard *assumptions* (Appendix E.1.2), and tailored ML methods (Appendix E.1.3). We explain these in the following.

E.1.1 Challenges: The Fundamental Problem and Confounding

The fundamental problem of causal inference [7] is that, for each instance, we only observe either $Y(0)$ or $Y(1)$, depending on what treatment was administered. We refer to the observed outcome as the factual outcome and the unobserved outcome as the counterfactual outcome. Because one outcome is always unobserved, we never know the true CATE τ , which means that there is *no ground truth* CATE available for training or validation.

In observational data, the outcome that was observed is typically not random: some instances were more likely to be treated, while other instances were more likely not to receive treatment. For example, in healthcare, patients may be more likely to receive a new treatment if they have access to better healthcare, have no pre-existing conditions, and are younger. The covariates that influence both the outcome and treatment assignment are

called *confounders*, with the resulting non-random treatment assignment sometimes referred to as confounding.

Confounding presents an additional challenge for CATE estimation and validation as it results in *covariate shift*. Some instance-treatment pairs (the counterfactuals) will be absent in the observational training data compared to the hypothetical test data that contains all instance-treatment pairs (both factials and counterfactuals). Because of this, an ML model may focus too much on the observed data points at the cost of worse predictions for the counterfactuals and, as such, the test data overall.

E.1.2 Assumptions For Identifiability

Identifying the causal effect from observational data requires making standard assumptions: consistency, overlap, and unconfoundedness. This section explains these assumptions in more detail.

Assumption 14 (Consistency). *The observed outcome given a treatment is the potential outcome under that treatment: $Y|X, t = Y(t)|X$.*

Assumption 15 (Overlap). *For each instance, there is a non-zero probability of receiving each treatment given their covariates: $\forall x \in \mathcal{X}$ and $t \in \mathcal{T} : P(T = t|X = x) > 0$. This condition ensures that there is sufficient variability in the treatment assignment.*

Assumption 16 (Unconfoundedness). *Given an instance's covariates, its potential outcomes are independent of the treatment assignment: $Y(0), Y(1) \perp\!\!\!\perp T | X$. This condition implies that all factors influencing both the treatment assignment and outcome are included in X . In other words, there are no unobserved confounders.*

There has recently been much interest in CATE estimation under violation of these assumptions. For example, by quantifying the uncertainty or sensitivity of an estimate to a possible violation [190], [193], [426], characterizing overlap violations [188], or developing metalearners that can deal with unobserved confounders [271]. We believe that extending AutoCATE to deal with these settings and to incorporate these methods will improve its potential for real-world applicability even further. As such, we consider it an important direction for future versions.

E.1.3 CATE Estimation: Meta- and Baselearners

We briefly describe the approach of estimating the CATE with a metalearner here. A straightforward way of estimating the CATE is using a single ML model, where the treatment variable is considered an ordinary input variable.

This metalearner is called the *S-Learner* and can be implemented with a wide variety of baselearners (i.e., ML algorithms that predict an outcome based on data, such as a decision tree or neural network). An alternative metalearner, the *T-learner*, fits two models—one model for each treatment group. Both models can use the same baselearner or a different one. More information on the metalearners in `AutoCATE` is provided in Appendix E.2.1. For more extensive overviews, we refer to [173], [345], and [425].

E.2 AutoCATE: Additional Information

This section presents information on metalearners (Appendix E.2.1), risk measures for evaluation (Appendix E.2.2), and `AutoCATE`'s search spaces for preprocessors and baselearners (Appendix E.2.3).

E.2.1 Metalearners

We describe the metalearners implemented in `AutoCATE` in more detail below. We first define the estimates that make up the building blocks of these models: the estimated propensity score $\hat{e}(x) = \mathbb{E}(t|x)$, the treatment-group specific outcome $\hat{y}_0(x) = \mathbb{E}(y|x, t = 0)$ and $\hat{y}_1(x) = \mathbb{E}(y|x, t = 1)$, and the treatment-unaware outcome $\hat{\mu}(x) = \mathbb{E}(y|x)$. In the following, the function f describes a model that is learned with a base learner such as a neural network or gradient boosting.

S-Learner. The *S-Learner*, or *single learner*, simply uses the treatment as a variable: $f_S(x, t) = \mathbb{E}(y|x, t)$. The CATE τ is then estimated as $\hat{\tau} = \hat{y}_1 - \hat{y}_0 = f_S(x, t = 1) - f_S(x, t = 0)$.

***Lo-Learner* [427].** The *Lo-Learner* is similar to an *S-Learner*, in the sense that it uses the treatment as a variable, but it adds interaction terms between the covariates x and treatment t : $f_{Lo}(x, t) = \mathbb{E}(y|x, t, x \cdot t)$. The CATE τ is then estimated as $\hat{\tau} = \hat{y}_1 - \hat{y}_0 = f_{Lo}(x, t = 1) - f_{Lo}(x, t = 0)$.

T-Learner. The *T-Learner* constructs *two* models—one per treatment group: $f_T^0(x) = \mathbb{E}(y|x, t = 0)$ and $f_T^1(x) = \mathbb{E}(y|x, t = 1)$, and predicts the CATE as $\hat{\tau} = \hat{y}_1 - \hat{y}_0 = f_T^1(x) - f_T^0(x)$.

***X-Learner* [208].** The *X-Learner* first learns two treatment-specific outcome models: $\hat{y}_0(x)$ and $\hat{y}_1(x)$. It then uses these to impute the counterfactual outcome for each instance and, as such, obtain a pseudo-outcome $\tilde{\tau}_X$ for the treatment effect: $\tilde{\tau}_X^0 = \hat{y}_1(x) - y$ if $t = 0$, and $\tilde{\tau}_X^1 = y - \hat{y}_0(x)$

else. For each treatment group, a model is then learned on these pseudo-outcome: $f_X^0(x) = \tilde{\tau}_X^0$ and $f_X^1(x) = \tilde{\tau}_X^1$. The final effect model then estimates $f_X(x) = g(x)f_X^0 + (1 - g(x))f_X^1$ and predicts the treatment effect as $\hat{\tau} = f_X(x)$. $g(x) \in [0, 1]$ is a weighting function, typically the estimated propensity score $g(x) = \hat{e}(x)$.

RA-Learner [209]. The *RA-Learner* or *regression-adjusted learner* is similar to an *X-Learner*, but directly learns the final model on the pseudo-outcomes: $f_{RA}(x) = \mathbb{E}(\tilde{\tau}_X|x)$, predicting the treatment effect as $\hat{\tau} = f_{RA}(x)$.

Z-Learner. The transformed outcome approach [378], [428] or inverse propensity weighted estimator [209] uses a pseudo-outcome based on the Horvitz-Thompson transformation [429]: $\tilde{\tau}_Z = \left(\frac{t}{\hat{e}(x)} - \frac{1-t}{1-\hat{e}(x)}\right)y$. The *Z-Learner* then estimates $f_Z(x) = \mathbb{E}(\tilde{\tau}_Z|x)$ and predicts the treatment effect as $\hat{\tau} = f_Z(x)$.

U-Learner. The *U-Learner* is based on a pseudo-outcome $\tilde{\tau}_U = \frac{y-\hat{\mu}(x)}{t-\hat{e}(x)}$. The final model fits $f_U(x) = \mathbb{E}(\tilde{\tau}_U|x)$ and predicts the treatment effect as $\hat{\tau} = f_U(x)$.

F-Learner [430]. The *F-Learner* uses the pseudo-outcome $\tilde{\tau}_F = \frac{t-\hat{e}(x)}{\hat{e}(x)(1-\hat{e}(x))}y$. The final model fits $f_F(x) = \mathbb{E}(\tilde{\tau}_F|x)$ and predicts the treatment effect as $\hat{\tau} = f_F(x)$.

DR-Learner [381]. The *DR-Learner* is a robust version of the *Z-Learner*, based on the pseudo-outcome $\tilde{\tau}_Z = \left(\frac{t}{\hat{e}(x)} - \frac{1-t}{1-\hat{e}(x)}\right)y + \left(1 - \frac{t}{\hat{e}(x)}\right)\hat{y}_1(x) + \left(1 - \frac{1-t}{1-\hat{e}(x)}\right)\hat{y}_0(x)$. The final model is $f_{DR}(x) = \mathbb{E}(\tilde{\tau}_{DR}|x)$ and predicts the treatment effect as $\hat{\tau} = f_{DR}(x)$.

R-Learner [382]. The *R-Learner*, based on Robinson’s decomposition [383], fits a model $f_R(x)$ using a weighted loss function with pseudo-outcomes $\tilde{\tau}_R = \frac{y-\hat{\mu}(x)}{t-\hat{e}(x)}$ and weights $w = (t - \hat{e}(x))^2$. The treatment effect is then predicted as $\hat{\tau} = f_R(x)$.

E.2.2 Evaluation Risk Measures

Metalearner pseudo-outcomes. An instance’s true CATE τ is unknown, but we can use the pseudo-outcomes $\tilde{\tau}$ used by the *T*-, *Z*-, *U*-, *F*-, *DR*-, and *R*-Learners (see above) as ground truth.

Table B1: **Preprocessor search spaces.** We describe the search spaces for the different preprocessors. If a hyperparameter is not mentioned, we use its default. All preprocessors are implemented with `scikit-learn` [335]; we refer to their documentation for more information.

Hyperparameter	Range	Hyperparameter	Range
	<i>VarianceThreshold</i>		<i>StandardScaler</i>
<code>threshold</code>	[0, 0.04]		—
	<i>SelectPercentile</i>		<i>RobustScaler</i>
<code>k</code>	[5, <code>n_dim</code>]		—
<code>score_func</code>	<code>mutual_info_{regression, classif}</code>		

(a) Feature Selection
(b) Feature Scaling

Influence Function (IF) [185]. The influence function criterion gives an estimate of an ML pipeline’s estimation error. It is based on a pseudo-outcome of the treatment effect $\tilde{\tau}$, estimated with a T -Learner. This pseudo-outcome is then debiased using the influence function. The final criterion is:

$$(1 - B) \tilde{\tau}^2 + B y(\tilde{\tau} - \hat{\tau}) - D(\tilde{\tau} - \hat{\tau})^2 + \tilde{\tau}^2$$

with $D = t - \hat{e}(x)$, $C = \hat{e}(x)(1 - \hat{e}(x))$, and $B = 2tDC^{-1}$.

k -Nearest Neighbor (k NN) [431]. The nearest neighbor matching measure finds the nearest neighbor in the opposite group, defined using the Euclidean distance, and uses its outcome as the counterfactual outcome. As such, it is essentially a T -Learner pseudo-outcome where the baselearner is restricted to a nearest neighbor model. We extend upon this by allowing alternative versions to be constructed by increasing k .

E.2.3 Preprocessor and Baselearner Search Spaces

Preprocessors. ML pipelines include three (optional) steps to preprocess the data before being fed to a model: feature selection, transformation, and scaling. For feature selection, include `VarianceThreshold`, `SelectPercentile`, or no selection. For feature scaling, we include `StandardScaler`, `RobustScaler`, or no scaling. Finally, we include feature transformation algorithms in our software package (`SplineTransformer`, `PolynomialFeatures`, `KBinsDiscretizer`), but do not include them in the experiments as they significantly slowed down training times. Other steps for feature selection and scaling from `scikit-learn` are similarly supported, but not included in the experiments, which is why we do not discuss them here. Table B1 provides detailed information on the search spaces.

Table B2: **Baselearner search spaces.** We describe the search spaces for each baselearner. If a hyperparameter is not mentioned, we use its default. All baselearners are implemented with `scikit-learn` [335]; we refer to their documentation for more information.

Hyperparameter	Range	Hyperparameter	Range
<i>Gradient Boosting</i>		<i>Linear/Logistic Regression</i>	
n_estimators	[50, 2000]	alpha	[1e-6, 1e6]
subsample	[0.4, 10]	<i>Gaussian Process</i>	
min_samples_split	[2, 500]	n_restarts_optimizer	[0, 5]
learning_rate	[0.05, 0.5]	normalize_y	[True, False]
n_iter_no_change	[5, 100]	alpha	[1e-5, 1e2]
max_leaf_nodes	None	max_iter_predict	[100, 1000]
max_depth	None	<i>Support Vector Machine</i>	
<i>Random Forest</i>		c	[1e-6, 1e6]
n_estimators	[50, 500]	kernel	[linear, poly, rbf, sigmoid]
max_depth	None	degree	[1, 10]
min_samples_split	[2, 100]	<i>k-Nearest Neighbors</i>	
max_features	[0.4, 1.0]	n_neighbors	[1, 30]
<i>Extra Trees</i>		weights	[uniform, distance]
n_estimators	[50, 500]	<i>Neural Network</i>	
max_depth	None	hidden_layers	[1, 3]
min_samples_split	[2, 100]	hidden_neurons	[8, 64]
max_features	[0.4, 1.0]	alpha	[1e-6, 1e1]
<i>Decision Tree</i>		learning_rate_init	[5e-4, 1e-2]
max_depth	[1, 2000]	batch_size	[16, 64]
min_samples_split	[2, 500]	activation	[tanh, relu]
min_samples_leaf	[1, 500]	max_iter	200
max_features	[0.4, 1.0]	solver	adam
		early_stopping	True

Baselearners. We present the search spaces for all baselearners’ hyperparameters in Table B2. These are based largely upon existing AutoML packages (e.g., FLAML [323]) and some (limited) experimentation, so these may be improved in future versions.

AutoCATE’s resulting search space of ML pipelines for CATE estimation is vast, with 2,187 possible pipelines even *without considering hyperparameters*:

$$3 \text{ feature selection} \times 3 \text{ scaling} \times 27 \text{ metalearner-baselearner configurations} \times 9 \text{ baselearners} \quad (\text{E.2})$$

with $27 = 1 (S) + 2 (T) + 4 (DR) + 5 (X) + 4 (R) + 3 (RA) + 1 (Lo) + 2 (Z) + 3 (U) + 2 (F)$, i.e., the sum of all baselearners required per metalearner.

E.2.4 Example ML Pipeline

We give an example of a pipeline built by AutoCATE, excluding baselearner hyperparameters. *Evaluation* using a T -Risk evaluation, with control outcomes estimated with gradient boosting and treatment outcomes estimated using a neural network. *Estimation* by first selecting a top percentile of features based on the F-value between the label and feature, followed by a DR -Learner where propensity scores are estimated with a support vector machine, control outcomes with gradient boosting, treatment outcomes with a linear regression, and the final effect with a random forest. This example illustrates the complexity of an ML pipeline for CATE estimation—in this case, there are six different ML models with several hyperparameters each. If an *ensemble* is used for estimation, this complexity increases even more.

E.2.5 AutoCATE’s API: Additional Information

We give more information on AutoCATE’s initialization arguments in Listing E.1.

```
1 class AutoCATE:
2     def __init__(
3         self,
4         # evaluation_metrics: Risk measures to evaluate the
5           performance
6         evaluation_metrics=None,
7         # preprocessors: Preprocessors to try (defaults added
8           later)
9         preprocessors=None,
10        # base_learners: Baselearners to try (defaults added
11          later)
12        base_learners=None,
13        # metalearners: Metalearners to try (defaults added
14          later)
15        metalearners=None,
16        # task: Type of task ('regression' or 'classification
17          ')
18        task="regression",
19        # metric: Metric used to evaluate the model (e.g., '
20          MSE')
21        metric="MSE",
22        # ensemble_strategy: Strategy for selecting a final
23          model
24        ensemble_strategy="toplverage",
25        # single_base_learner: Use only one base learner
26        single_base_learner=False,
27        # joint_optimization: Same hyperparameters for
28          baselearners
29        joint_optimization=False,
30        # n_folds: Number of folds for cross-validation
31        n_folds=1,
```

```

24     # n_trials: How many trials to optimize the estimation
        pipeline
25     n_trials=50,
26     # n_eval_versions: Number of versions of each risk
        measure
27     n_eval_versions=1,
28     # n_eval_trials: Number of trials for evaluating the
        model
29     n_eval_trials=50,
30     # seed: Random seed for reproducibility
31     seed=42,
32     # visualize: Whether to visualize results
33     visualize=False,
34     # max_time: Maximum time allowed for fitting the model
35     max_time=None,
36     # n_jobs: Number of parallel jobs to run
37     n_jobs=-1,
38     # cross_val_predict_folds: Folds for cross-validated
        estimates
39     cross_val_predict_folds=1,
40     # holdout_ratio: Ratio of data for validation (if
        single fold)
41     holdout_ratio=0.3
42 ):
43
44     # Initialization code (not included here)
45     ...

```

Listing E.1: **Arguments for the AutoCATE class initialization.** We describe each argument and its default initialization.

E.3 Data: Additional Information

This section describes the data used in this work in more detail.

IHDP [205]. The data come from the Infant Health and Development Program, describing the impact of child care and home visits on children’s cognitive development. Treatments and outcomes were simulated for a total of 100 data sets. Each version contains $n = 747$ instances and $d = 25$ covariates.

ACIC [337]. The data from the ACIC 2016 competition was based on data from the Collaborative Perinatal Project, studying drivers of developmental disorders in pregnant women and their children. 77 distinct data sets were created, each with $n = 4,802$ instances and $d = 58$ covariates. 100 iterations were originally created for each data set, but we use only the first one for each.

Twins [217]. The Twins data studies the effect of being the heavier twin on mortality. $n = 11,984$ pairs of twins are included, with $d = 46$ features each. Only one version of this data set exists, so we run 10 iterations of each experiment.

News [200]. This data simulates a reader’s reading experience (y) based on the device they use for reading (t) and the news article (x). There are 50 distinct data sets, each with $n = 5,000$ instances with and $d = 3,477$ covariates.

Below, we include results for two data sets on uplift modeling:

Hillstrom [384]. This data contains records of customers ($n = 64,000$) that were contacted by a marketing campaign over e-mail. Originally, customers received either no mail, a mail with men’s merchandise, or one with women’s merchandise, but we convert it to not contacted ($t = 0$) or contacted ($t = 1$). For each customer, $d = 10$ covariates are available. As the outcome y , we consider whether the customer visited the website or not.

Information [432]. The information data set comes from the R Information package. It describes customers ($n = 10,000, d = 68$) in the insurance industry, as well as whether they were contacted with a marketing campaign and whether they made a purchase.

E.4 Additional Results

E.4.1 Stage 1: Evaluation

Table B3 shows results for evaluating with k -fold cross validation for different values of k .

Table B3: **The effect of k in k -fold cross validation.** For each data set, we show result for a varying number of cross-validation folds. Results for 50 evaluation trials with a T -risk and 50 estimation trials with a T -Learner and gradient boosting.

	1	2	3	4	5	10
<i>IHDP</i>	2.15 \pm .35	2.16 \pm .35	2.10 \pm .35	2.07 \pm .33	2.29 \pm .42	2.25 \pm .41
<i>ACIC</i>	1.52 \pm .09	1.58 \pm .08	1.48 \pm .08	1.51 \pm .09	1.50 \pm .08	1.53 \pm .09
<i>Twins</i>	.323 \pm .00	.324 \pm .00	.322 \pm .00	.324 \pm .00	.344 \pm .00	.346 \pm .00
<i>News</i>	2.42 \pm .07	2.40 \pm .07	2.41 \pm .06	2.41 \pm .07	2.45 \pm .07	2.45 \pm .07

E.4.2 Stage 2: Estimation

Figure E.1 shows how often each metalearner gets picked in `AutoCATE`'s `BestMeta` configuration. The difference in metalearner selection rates illustrates the importance of data-driven metalearner selection, as facilitated by `AutoCATE`. Interestingly, other metalearners are preferred for a binary outcome (Twins) than for continuous outcomes (all others). This finding suggests that different `BestMeta` configurations may be optimal for different outcomes.

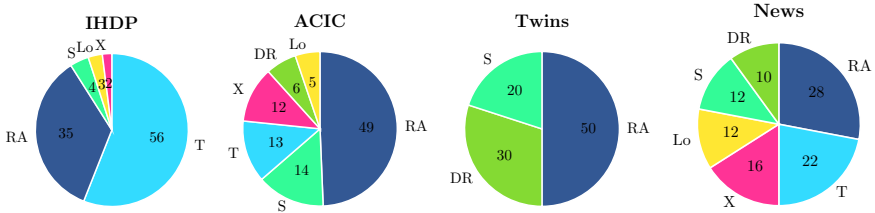


Figure E.1: **Metalearner selection.** We show how many times a metalearner gets picked (in % of all data set iterations) for a given data set. Results for `AutoCATE`'s `BestMeta` configuration, including the *S*-, *T*-, *Lo*-, *X*-, *RA*-, *DR*-, and *U*-Learners, with 50 evaluation and 500 estimation trials.

We compare different metalearners in terms of $\sqrt{\text{PEHE}}$ in Table B4. These results show that searching across metalearners typically significantly improves precision compared to using only one metalearner. Moreover, some metalearners can result in very poor performance even after 200 optimization trials. Typically, these results are due to exceptionally poor performance in some iterations (e.g., the *R*-Learner). Additionally, we compare the performance trade-off in terms of time and precision for best metalearners in Figure E.2. These results show that the *S*-, *T*-, and *Lo*-Learner are often the fastest to train and the most precise in terms of $\sqrt{\text{PEHE}}$. These results illustrate the potential of improving `AutoCATE`'s time efficiency by considering these trade-offs.

We can also apply explainability techniques to understand what drives a pipeline's predictions. Figure E.3 illustrates this and shows how permutation feature importance can be used with `AutoCATE`.

E.4.3 Stage 3: Ensembling

The ensemble built by `AutoCATE` can be used to gauge the uncertainty regarding a prediction, by highlighting the spread of predictions. We illustrate such an analysis in Figure E.4.

Table B4: **Comparing metalearner precision.** For each data set, we compare the different metalearner’s performance in terms of $\sqrt{\text{PEHE}}$, with the best result highlighted in **bold**. We also include a comparison with searching over all metalearners (AllMeta) and, in brackets, show how much this outperforms the best single metalearner. For each result, AutoCATE uses a T -risk with 50 evaluation trials, 200 estimation trials, and top 1 average model selection.

	S	T	DR	X	R	RA	Lo	Z	U	F	AllMeta
IHDP	4.52 _{±.74}	2.52 _{±.37}	5.91 _{±.98}	5.46 _{±.87}	2752.36 _{±1613.91}	5.80 _{±.89}	2.47 _{±.34}	50.09 _{±6.21}	7.45 _{±1.12}	9.58 _{±.95}	1.54_{±.25} (-37.5%)
ACIC	4.00 _{±.24}	4.26 _{±.14}	3.61 _{±.22}	3.09 _{±.16}	477325.02 _{±87957.53}	3.27 _{±.19}	3.07 _{±.13}	150829.14 _{±56790.59}	5.75 _{±.43}	4.65 _{±.35}	1.62_{±.09} (-47.3%)
Twins	.318_{±.00}	.345 _{±.01}	.320 _{±.00}	.333 _{±.00}	77.408 _{±33.07}	.323 _{±.00}	.360 _{±.00}	.546 _{±.01}	.418 _{±.01}	.376 _{±.00}	.321_{±.00} (+ 0.9%)
News	2.89 _{±.14}	2.53 _{±.07}	3.38 _{±.15}	2.93 _{±.13}	36448.74 _{±13452.34}	3.14 _{±.13}	2.57 _{±.08}	16.06 _{±1.80}	2.74 _{±.13}	3.41 _{±.11}	2.40_{±.08} (- 5.0%)

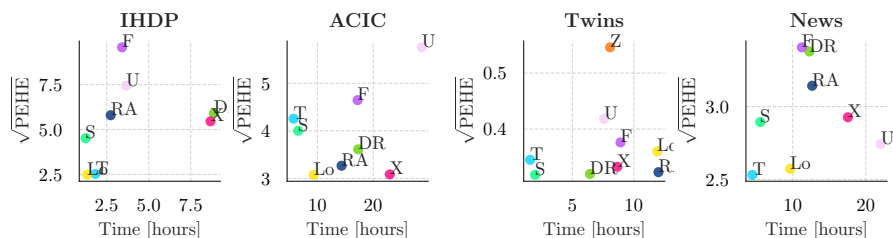


Figure E.2: **Comparing metalearner precision and time efficiency.** We show each metalearner’s performance in precision ($\sqrt{\text{PEHE}}$) and time (excluding outliers, see Table B4). For each, AutoCATE uses a T -risk with 50 evaluation trials, 200 estimation trials, and top 1 average model selection.

E.4.4 Benchmarking AutoCATE

Table B5 presents results for additional benchmarks: S- and T-Learners based on linear or logistic models (without regularization).

Figure E.5 shows additional results for two data sets for uplift modeling (see Appendix E.3 for more information on the data). The effectiveness of AutoCATE is related to at least three factors. First, by using the AUQC metric, the search is aligned with the downstream task: prioritizing instances for treatment [13]. Second, the search space for AutoCATE includes more meta- and baselearners than the benchmarks. Third, the top five ensemble seems to improve the stability and accuracy of the predicted ranking.

E.4.5 Analyzing AutoCATE’s Results

We analyze the results of AutoCATE’s optimized pipelines in Figure E.6. These results illustrate how AutoCATE can facilitate a higher-level, comprehensive analysis of methods for CATE estimation and model validation.

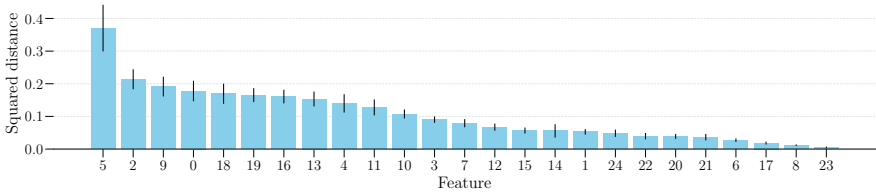


Figure E.3: **Analyzing AutoCATE’s feature importance.** We can analyze how much each feature contributes to treatment effect heterogeneity. We illustrate this analysis for the first iteration of IHDP using permutation feature importance, showing the squared distance to the original prediction when permuting a feature column.

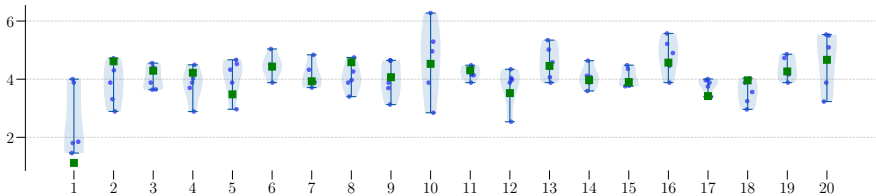


Figure E.4: **Assessing uncertainty with AutoCATE.** The ensemble returned by AutoCATE can be used to analyze uncertainty regarding the prediction. We illustrate this for the first 20 instances of the first iteration of the IHDP data. For each instance, the (usually unknown) ground truth is shown in green, while the predictions from the top five pipelines are shown in blue and with a violinplot.

E.5 Comparing Software Packages for CATE Estimation

Table B6 lists software packages for CATE estimation, comparing their functionalities with AutoCATE. Notably, *no other package* is focused on automated, end-to-end CATE estimation.

Table B5: **Comparing AutoCATE with common benchmarks on CATE estimation.** We compare performance in terms of $\sqrt{\text{PEHE}}$, with the best result highlighted in **bold**. AutoCATE results for a T -risk with 50 evaluation trials and 50 estimation trials with the BestMeta-BestBase configuration.

	AutoCATE		Benchmarks					
	<i>Top 1</i>	<i>Top 5</i>	<i>S-RF</i>	<i>T-RF</i>	<i>S-GB</i>	<i>T-GB</i>	<i>S-LR</i>	<i>T-LR</i>
<i>IHDP</i>	1.25 \pm .18	1.38 \pm .21	3.30 \pm .57	2.61 \pm .45	3.02 \pm .52	1.86 \pm .29	5.73 \pm .89	2.41 \pm .39
<i>ACIC</i>	1.52 \pm .09	1.45 \pm .10	1.67 \pm .08	1.65 \pm .09	1.48 \pm .10	1.38 \pm .09	4.13 \pm .25	3.08 \pm .15
<i>Twins</i>	.315 \pm .00	.314 \pm .00	.318 \pm .00	.331 \pm .00	.319 \pm .00	.334 \pm .00	.320 \pm .00	.335 \pm .00
<i>News</i>	2.33 \pm .06	2.29 \pm .06	2.46 \pm .09	2.39 \pm .07	2.68 \pm .11	2.40 \pm .06	3.68 \pm .17	2.93 \pm .12

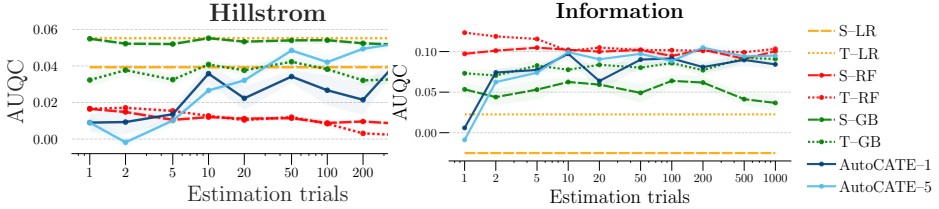


Figure E.5: **Benchmarking AutoCATE for treatment prioritization.** We present additional results in terms of AUQC for two uplift data sets, Hillstrom and Information. These show that AutoCATE is a useful tool for prioritizing instances for treatment, and highlight that its optimization is more effective at optimizing AUQC compared to the benchmarks based on μ -risk. AutoCATE uses a T -risk with 50 evaluation trials and the AUQC metric, the BestMeta-BestBase search space, and Top 1 or Top 5 ensembling.

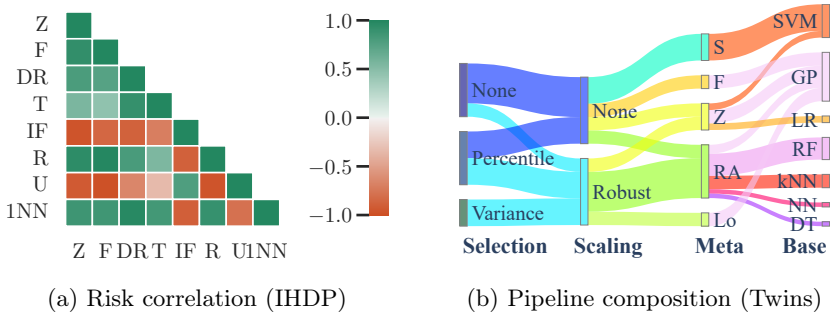


Figure E.6: **Analyzing AutoCATE’s results.** We present results analyzing pipelines optimized by AutoCATE. Figure (a) shows the correlation between risk measures for a single IHDP iteration. Surprisingly, risk measures can be strongly *negatively correlated*, suggesting potential for more advanced multi-objective approaches that adaptively learn which objectives are reliable for a given data set. Figure (b) visualizes the *optimal pipelines* learned across ten iterations for the Twins data.

Table B6: **Software package comparison.** We provide an overview of commonly used packages for CATE estimation and compare their functionalities with AutoCATE, showing whether they support (1) evaluation, (2) estimation, (3) ensembling, and (4) automated, end-to-end optimization—as provided by AutoCATE or similar.

PACKAGE Name	FUNCTIONALITIES				GENERAL INFORMATION		
	(1)	(2)	(3)	(4)	Language	Reference	Link
CausalML	\times^*	✓	\times	\times	Python	[334]	GitHub
EconML	✓ [§]	✓	✓ [§]	\times	Python	—	GitHub
DoWhy	\times^\dagger	✓	\times	\times	Python	[433]	GitHub
Causica	\times	✓	\times	\times	Python	[434]	GitHub
UpliftML	\times	✓	\times	\times	Python	[435]	GitHub
scikit-uplift	\times	\times	\times	\times	Python	—	GitHub
grf	\times	✓	✓ [‡]	\times	R	[207]	CRAN
AutoCATE	✓	✓	✓	✓	Python	This work	GitHub

*CausalML offers provides some tools for internal validity, such as comparing results across segments.

§EconML includes an R -risk and can provide an ensemble based on this risk measure.

†DoWhy includes robustness checks for assumption violations.

‡The grf package allows for evaluation based on the Targeting Operating Characteristics curve.

F

**METALEARNERS FOR RANKING
TREATMENT EFFECTS**

F.1 Problem Formulation: Identifiability Assumptions

As introduced in the main body, we require the standard assumptions from causal inference to identify the causal effect. In this work, we assumed that historical data comes from a randomized controlled trial:

Assumption 17 (Consistency). *When $Y = y$ and $T = t$, we assume that $Y(T = t) = y$. This implicates that, for each instance, when given treatment t , the outcome we observe is the potential outcome associated to that treatment $Y(t)$.*

Assumption 18 (No interference). *An instance's outcome given a treatment is independent of treatments administered to other instances: $Y_i(t_0, \dots, t_i, \dots, t_n) = Y_i(t_i)$.*

Assumption 19 (Unconfoundedness). *We assume $Y(T) \perp\!\!\!\perp T$, i.e., past treatment decisions were made at random, i.e., not based on the instance's characteristics.*

If we do not have data from a randomized trial, we require a stronger assumption called strong ignorability or no hidden confounding: $Y(T) \perp\!\!\!\perp T|x$, i.e., past treatment decisions were exclusively based on the instance's observed characteristics x . In this case, we also require positivity: for each instance, the probability of administering each treatment has to be larger than zero, i.e., $P(T|x) > 0$. We do not consider this scenario in our work. However, our ranking metalearners can easily be extended to these scenarios: while some metalearners already integrate the propensity score in their construction, others may be improved by inverse propensity score weighting [429], [436].

F.2 Empirical Results: Additional Experiments

This section presents additional experimental results. We describe additional results for RQ1 in Appendix F.2.1, and for RQ2 and RQ3 in Appendix F.2.2 where we analyze the effect of several design choices and hyperparameters.

Table B1: *Ranking Quality for Different Objectives and Metalearners.* For each metalearner, we compare three different objectives: point-, pair-, and listwise. We show performance in terms of AUQC (with standard error in brackets), for the Synthetic, Criteo, and Hillstrom (Women (φ) and Men (σ) e-mail), and Promotion data.

Meta	Objective			Data				
	Point	Pair	List	Synthetic	Criteo	Hillstrom φ	Hillstrom σ	Promotion*
Z	✓	✗	✗	0.183 (0.027)	0.038 (0.018)	0.057 (0.030)	0.013 (0.033)	0.407 (0.111)
	✗	✓	✗	0.137 (0.016)	-0.014 (0.010)	0.016 (0.039)	0.022 (0.044)	0.370 (0.296)
	✗	✗	✓	0.242 (0.026)	0.048 (0.009)	0.092 (0.018)	0.027 (0.033)	0.593 (0.037)
S	✓	✗	✗	0.051 (0.050)	0.030 (0.020)	0.011 (0.048)	0.025 (0.041)	0.370 (0.074)
	✗	✓	✗	0.052 (0.039)	0.002 (0.008)	0.042 (0.014)	-0.021 (0.039)	0.593 (0.296)
	✗	✗	✓	0.037 (0.027)	0.048 (0.016)	0.039 (0.054)	0.021 (0.035)	0.222 (0.148)
T	✓	✗	✗	0.054 (0.049)	0.043 (0.021)	-0.004 (0.036)	0.015 (0.009)	0.333 (0.148)
	✗	✓	✗	0.046 (0.044)	-0.015 (0.010)	0.044 (0.014)	-0.017 (0.045)	0.630 (0.185)
	✗	✗	✓	0.058 (0.043)	0.039 (0.013)	0.021 (0.037)	0.032 (0.028)	0.556 (0.222)
X	✓	✗	✗	0.052 (0.049)	0.034 (0.017)	-0.000 (0.031)	-0.028 (0.039)	1.000 (0.296)
	✗	✓	✗	0.054 (0.042)	0.002 (0.009)	0.028 (0.039)	0.029 (0.034)	0.704 (0.259)
	✗	✗	✓	0.061 (0.038)	0.036 (0.012)	0.033 (0.051)	0.018 (0.033)	0.852 (0.111)
DR	✓	✗	✗	0.055 (0.046)	0.044 (0.007)	-0.101 (0.044)	0.043 (0.025)	0.667 (0.111)
	✗	✓	✗	0.048 (0.047)	0.022 (0.012)	0.027 (0.019)	0.006 (0.054)	-0.481 (0.222)
	✗	✗	✓	0.056 (0.038)	0.039 (0.014)	0.005 (0.055)	0.039 (0.033)	-0.148 (0.222)
R	✓	✗	✗	0.054 (0.046)	0.029 (0.018)	-0.034 (0.042)	0.033 (0.028)	0.593 (0.222)
	✗	✓	✗	0.050 (0.047)	0.021 (0.022)	0.036 (0.013)	0.012 (0.043)	-0.296 (0.111)
	✗	✗	✓	0.056 (0.039)	0.038 (0.014)	0.048 (0.048)	0.034 (0.031)	-0.259 (0.259)

*For the Promotion data, we only present scaled results such that the best AUQC = 1 due to reasons of confidentiality.

F.2.1 Comparing Performance for the Different Objectives and Metalearners (RQ1): Additional Results

We display the results presented in the main body of the text without normalizing the best value to 1 in Figure F.1 and provide them in table format in Table B1.

F.2.2 Analyzing Alternative Metrics (RQ2) and Design Choices (RQ3): Additional Results

First, we present additional results to support our investigation surrounding RQ2. For each metalearner, we visualize the trade-off between different metrics: the MSE (i.e., pointwise accuracy), Kendall τ (i.e., pairwise rank correlation), and AUQC (i.e., listwise ranking quality). This shows that models that perform well in terms of AUQC, also perform well in terms of Kendall τ . Conversely, performance in terms of MSE does not seem related

Table B2: *Analyzing Performance Trade-offs on Synthetic Data.* For each metalearner, we compare three different objectives: pointwise, pairwise, and listwise. Using the `Synthetic` data set, we compare performance in terms of MSE (measuring pointwise accuracy), Kendall τ (measuring pairwise rank correlation), and AUQC (measuring global, listwise decision quality). For each, we show the standard error in brackets.

Meta	Objective			Metric		
	Point	Pair	List	MSE	Kendall τ	AUQC
Z	✓	×	×	2.474 (0.255)	0.128 (0.021)	0.183 (0.027)
	×	✓	×	2.867 (0.263)	0.101 (0.013)	0.137 (0.016)
	×	×	✓	6.465 (1.260)	0.164 (0.020)	0.242 (0.026)
S	✓	×	×	37.916 (3.557)	0.044 (0.033)	0.051 (0.050)
	×	✓	×	483.712 (159.054)	0.050 (0.028)	0.052 (0.039)
	×	×	✓	562.509 (300.946)	0.038 (0.022)	0.037 (0.027)
T	✓	×	×	41.014 (3.852)	0.045 (0.033)	0.054 (0.049)
	×	✓	×	213.265 (30.716)	0.049 (0.030)	0.046 (0.044)
	×	×	✓	513.908 (94.592)	0.057 (0.030)	0.058 (0.043)
X	✓	×	×	40.417 (3.697)	0.043 (0.033)	0.052 (0.049)
	×	✓	×	88.987 (16.651)	0.054 (0.030)	0.054 (0.042)
	×	×	✓	162.146 (40.750)	0.058 (0.028)	0.061 (0.038)
DR	✓	×	×	42.007 (4.017)	0.043 (0.031)	0.055 (0.046)
	×	✓	×	61.014 (10.268)	0.049 (0.032)	0.048 (0.047)
	×	×	✓	311.236 (105.607)	0.055 (0.028)	0.056 (0.038)
R	✓	×	×	41.263 (4.760)	0.042 (0.031)	0.054 (0.046)
	×	✓	×	51.008 (12.094)	0.051 (0.032)	0.050 (0.047)
	×	×	✓	124.400 (18.588)	0.055 (0.028)	0.056 (0.039)

to AUQC or Kendall τ . Finally, we also display these results in table format in Table B2.

F.2.3 Sensitivity Analysis

To single out the effect of the analyzed design choice, we train with default hyperparameters for each training objective and metalearner. We explore three hyperparameters: (1) the number of sampling iterations k , i.e., the number of sampled pairs per instance, (2) the sigmoid parameter σ controlling the steepness of the comparison in the construction of the pairwise score (see Equation (8.3)), and (3) whether we normalize the ranking score by feeding it to a logistic sigmoid and constraining it to $[0, 1]$. Related work has shown that this normalization might effectively serve as a form of regularization and help with overfitting [362].

In Figure F.3, we vary the number of sampled pairs k in the ranking objectives, for each ranking metalearner, and compare it to the pointwise model on the **Synthetic** data. We also show results for the metalearners with normalization of the score (i.e., constraining the model to outputs between zero and one) and without normalization. Somewhat surprisingly, with normalization, we see that increasing k does not yield better results. Conversely, without normalization, increasing k does in fact improve performance for most metalearners. Nevertheless, for the Z-, X-, DR-, and R-Learner, the best performance is achieved with normalization and $k = 1$ —the same settings used in the experiments in the main body (i.e., Table B1). For the S- and T-Learner, deviating from these settings might improve performance. Overall, we see that for each metalearner and objective, we can obtain the same performance or better than the pointwise equivalent given that the right hyperparameters are chosen. This insight provides another validation of our proposed approach.

In Figure F.4, we vary the sigmoid parameter σ , controlling the steepness of the comparison of the instance scores in the construction of the pairwise score (see Equation (8.3)). Generally, we obtain good performance for smaller values ($\sigma \leq 1$). When using normalization, our method seems more sensitive to this hyperparameter compared to training without score normalization. Although there may be some benefit of tuning this hyperparameter, we observe that fixing the sigmoid parameter at $\sigma = 1$ seems like a good choice overall—this was the setting used to generate the experimental results in the main body.

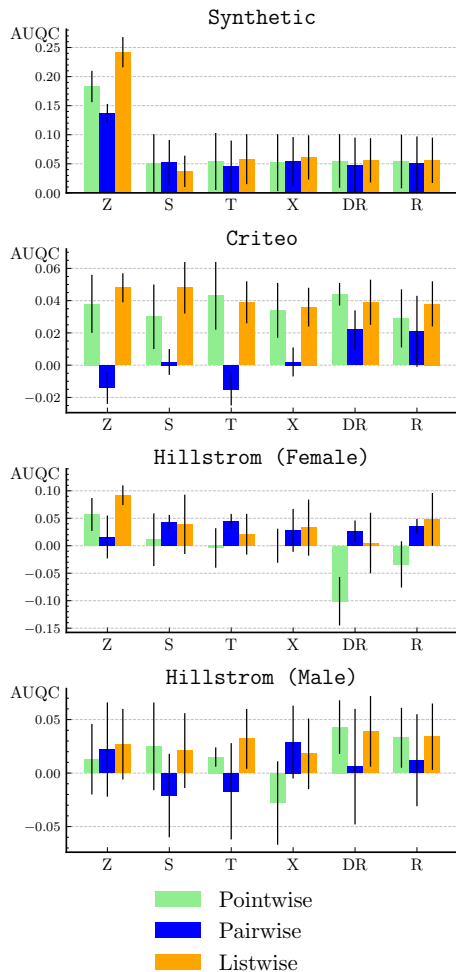


Figure F.1: *Ranking Quality for Different Objectives and Metalearners.* For each metalearner, we compare three different objectives: point-, pair-, and listwise. We show performance in terms of AUQC \pm one standard error, for five different data sets. As opposed to the figure in the main body, we do not scale the results here. Due to confidentiality reasons, we cannot share the raw results for the Promotion data set.

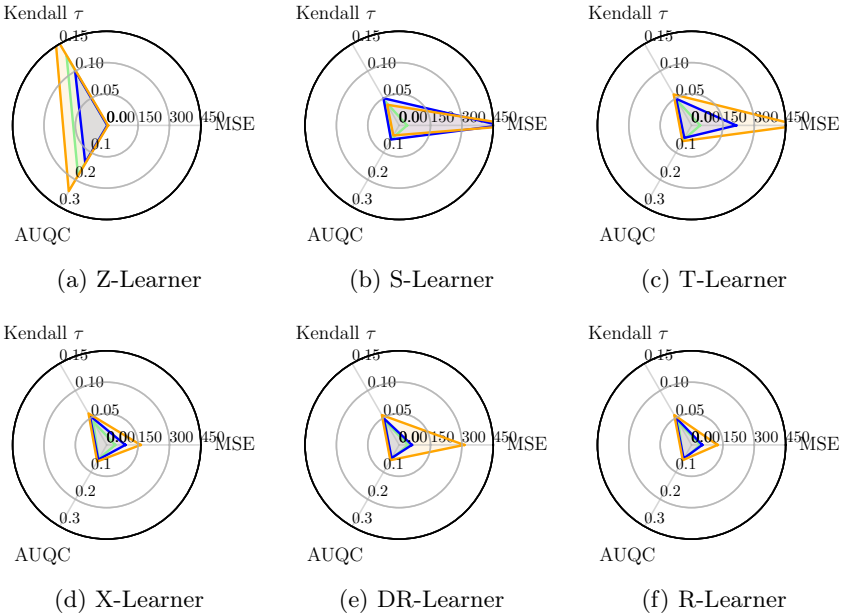


Figure F.2: *Analyzing Performance Trade-offs on Synthetic Data.* For each metalearner, we compare three different objectives: **pointwise**, **pairwise**, and **listwise**. Using the **Synthetic** data set, we compare performance in terms of MSE (measuring pointwise accuracy), Kendall τ (measuring pairwise rank correlation), and AUQC (measuring global, listwise decision quality).

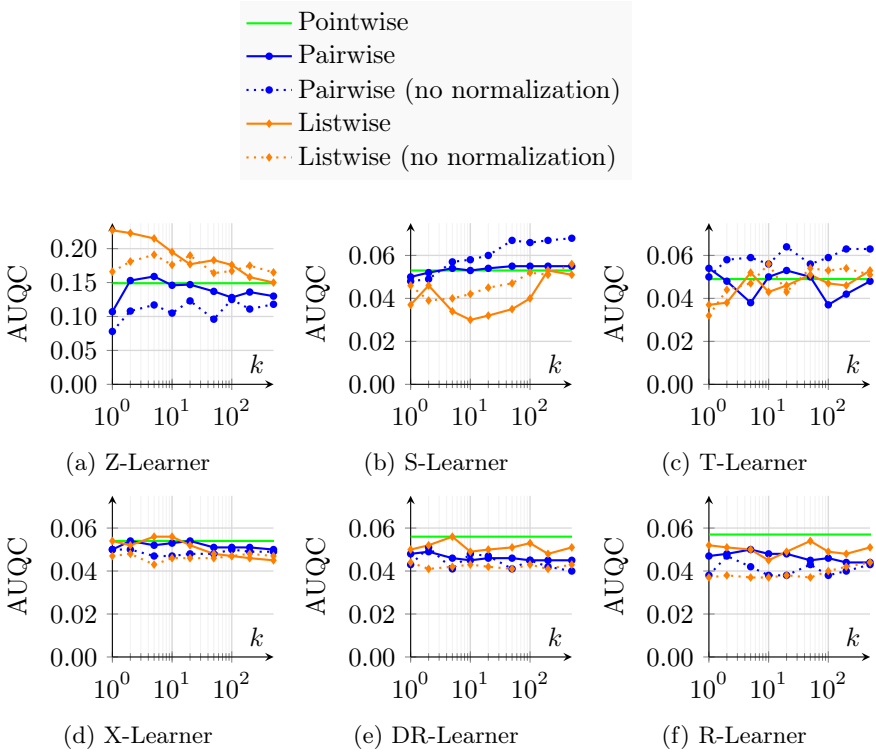


Figure F.3: *What Is the Effect of the Number of Sampling Iterations k ?* We show performance in terms of AUQC (higher is better) for the different metalearners on the Synthetic data set. We fix the sigmoid parameter $\sigma = 1$ and train with default hyperparameters.

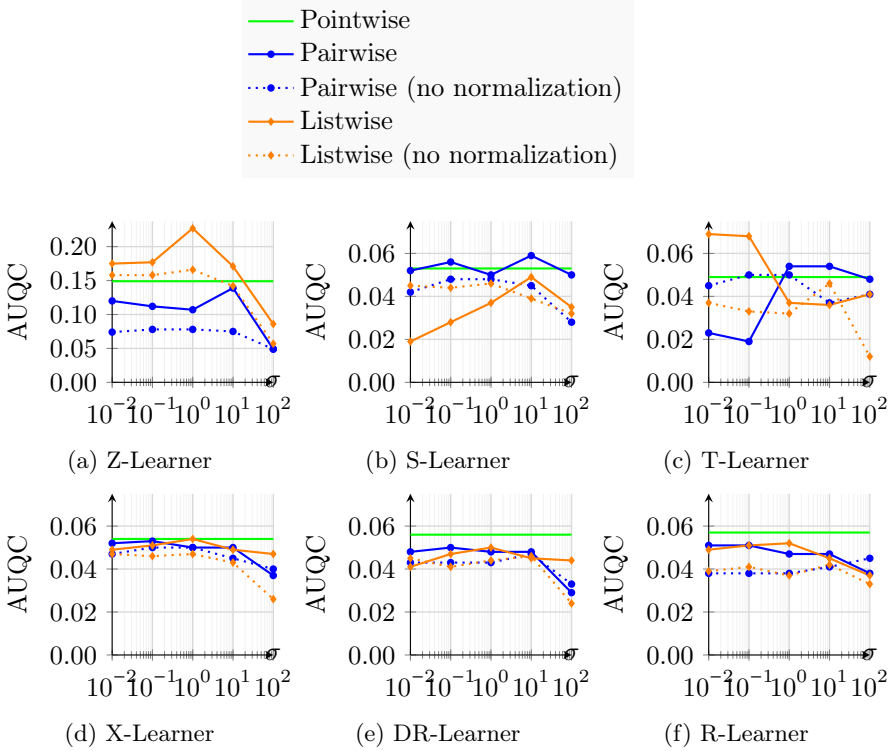


Figure F.4: *What Is the Effect of the Sigmoid Parameter σ ?* We show performance in terms of AUQC (higher is better) for the different metalearners on the Synthetic data set. We fix the number of sampling iterations $k = 1$ and train with default hyperparameters.

LIST OF TABLES

2.1	Cost-sensitive learning overview. We present an overview of various cost-sensitive learning methods in terms of the type of costs, place with respect to model training and classifier(s) used when applicable. <i>Costs</i> —CD: class-dependent, ID: instance-dependent; <i>Classifiers</i> —boosting, DR: decision rule, DT: decision tree, LR: logistic regression, NB: Naive Bayes, NN: neural network, SVM: support vector machine, -: classifier-agnostic.	18
2.2	Cost matrix. Extending the confusion matrix (2.2a) to a class- (2.2b) and instance-dependent cost matrix (2.2c).	19
2.3	Overview of the different models. These are obtained by combining the different objective functions with the different types of classifiers.	27
2.4	Overview of the datasets. Size (N), dimensionality (D) and degree of class imbalance (% Pos) are shown.	30
2.5	Cost matrices for the different applications. For each application, we present the different costs associated with different outcomes. A_i , Int_i , c_i^{FN} and c_i^{FP} represent instance-dependent costs, and c_f is a fixed cost.	31
2.6	Savings: comparison of the different thresholding strategies (averaged across all datasets). Best and second-best result for each model are denoted in bold and <i>italic</i>	33
2.7	Brier score before and after calibration for the different models (averaged across all datasets). The Brier score of models trained with a cost-sensitive objective function improves considerably, whereas it is stable for the models trained with a cross-entropy loss.	34

List of Tables

2.8 **F1 Score: comparison of the different thresholding strategies (averaged across all datasets).** Best and second-best result for each model are denoted in **bold** and *italic*. 35

2.9 **Instance-dependent or class-dependent costs: cost-insensitive metrics per model.** Significantly better results are denoted in **bold** (5%) and *italic* (10%). 36

2.10 **Instance-dependent or class-dependent costs: cost-sensitive metrics per model.** Significantly better results are denoted in **bold** (5%) and *italic* (10%). AEC is normalized between 0 and 1 per dataset (lower is better). 37

2.11 **Summary of the key findings.** We present a summary of the results per research question and hypothesis. Performance is judged in terms of costs and errors. Each question is answered with yes (\checkmark), no (\times) or inconclusive (?). 38

3.1 **Notation table.** We give an overview of the notation used in this work. For each symbol, we give both the general name and its role in our running example of fraud detection. 47

3.2 **Data sets overview.** For each data set, we present the application area, abbreviation, number of instances (N), class imbalance in terms of proportion of positive instances (% Pos), and corresponding reference. 54

3.3 **Cost matrices for the different application areas.** For each application, we present the costs for all outcomes in terms of predicted (\hat{y}) and actual (y) labels. A_i , c_i^{FN} , c_i^{FP} and Int_i represent instance-dependent costs and c_f is a fixed cost. 55

3.4 **Evaluation metrics overview.** We present an overview of the evaluation metrics. The average and standard deviation over all data sets are shown, with the best result denoted in **bold**. Results that are not significantly different from the best result are underlined ($\alpha = 0.05$). This is based on a Friedman test on the rankings with Bonferroni–Dunn post hoc correction. For both expected precision and profit, the ranking models perform best in their respective category. For the classification metric, average precision, the cost-insensitive classifier, xgboost, performs best. Conversely, for the ranking metrics, namely, Spearman correlation and the area under the cumulative profit curve, the ranking models outperform their classifying counterparts. 56

4.1	Data overview. Overview of the available contract information on machine and contract characteristics, preventive maintenance interventions, overhauls, and failures. For confidentiality, we present PM interventions, overhauls, and failures per running period, which is an undisclosed number of running hours. Similarly, the costs are averaged and re-scaled.	76
4.2	Data excerpt. We present an excerpt of the data set, showing examples of covariates related to the machine and contract \mathbf{x}_i , and maintenance related events per running period: the observed PM frequency t_i , the overhaul rate $o_i(t_i)$, and the failure rate $f_i(t_i)$.	77
4.3	Methodologies overview. Our proposed, individual policy, SCIGAN-ITE, prescribes the PM frequency based on the individual treatment effect (ITE) estimated using SCIGAN. This proposed approach is analyzed using an ablation study and compared with two variants. The first, MLP-ITE, does not account for selection bias. The second, SCIGAN-ATE, is a general policy based on the average treatment effect (ATE) and is not individualized towards each individual machine.	81
4.4	Empirical evaluation. We compare performance for the different policies over five simulation runs. We evaluate each model’s ability to accurately predict the potential outcomes $o_i(t)$ and $f_i(t)$ using the MISE, as well as each model’s ability to accurately prescribe PM frequencies (PE) and to minimize costs (PCR). For all metrics, a lower value is better.	83
5.1	Literature table. We compare NOFLITE with existing methodologies for estimating individual treatment effect distributions. Three dimensions are considered: (1) whether a neural network is used, (2) whether the model’s likelihood \mathcal{L}_θ is optimized directly, and (3) whether it adjust for confounding.	93
5.2	Empirical results. We compare NOFLITE against a variety of existing methods for three data sets: (a) IHDP, (b) EDU, and (c) News. For each metric, arrows indicate whether a lower (\downarrow) or higher (\uparrow) value is better; the ideal coverage is 90%. We show the best result in bold . We highlight the log-likelihood (LL) in gray to emphasize that this is our main metric of interest, as it evaluates the quality of the predicted distributions. For all data sets, GANITE achieves a loglikelihood of less than -10 , indicated by a dash (—).	106

7.1 **Performance for validation based on different risk measures.** Results in $\sqrt{\text{PEHE}}_{\pm\text{SE}}$ (lower is better). **Bold** highlights the best results, with underlined values falling within 1 standard error. Results for 50 evaluation trials and 50 estimation trials with a T -Learner and gradient boosting. 137

7.2 **Comparing different evaluation metrics.** We compare model selection with different evaluation metrics. For the Twins data set, MAPE cannot be calculated, as the true CATE can be zero. **Bold** highlights the best results, with underlined values falling within 1 standard error. Colored cells show the hypothesis that matching metrics will yield the best performance. Results for 50 evaluation trials with a T -risk and 50 estimation trials with a T -Learner and gradient boosting. 139

7.3 **Ensemble strategies.** We compare ensembling strategies for a single or multiple objectives in terms of $\sqrt{\text{PEHE}}$. **Bold** highlights the best results, underlined values lie within 1 standard error. Results for 50 evaluation trials and 50 estimation trials with a T -Learner and gradient boosting. 141

8.1 *Treatment Allocation Examples.* We highlight several applications where treatment allocation is required, characterized by (1) an estimated treatment effect, (2) an optimization objective, and (3) operational constraints. In *marketing*, the goal is to target customer segments and drive conversions, while adhering to budget constraints. In *healthcare and epidemiology*, optimal vaccine allocation during pandemics aims to minimize population mortality, subject to vaccine supplies. In *maintenance*, technicians are assigned to maintain assets, prevent failures, and maximizing operational uptime. Finally, in *economics and policy design*, subsidy usage needs to be optimized by efficiently allocating public funds and maximizing health care impact. 148

8.2	<i>Comparing Pointwise Estimation and Effect Ranking for Treatment Allocation.</i> This illustrative example shows the true treatment effect for several instances, ordered from largest to smallest. The first model estimates each instance’s treatment effect fairly accurately in terms of MSE. However, the ranking of instances based on these estimates differs significantly from their true order, which will result in suboptimal treatment allocation when only some instances can be treated. For the second model, we observe the opposite scenario: estimates are poor in terms of MSE, but their ranking respects the true order.	150
8.3	<i>Literature Table.</i> We categorize related work on effect ranking by differentiating between different (1) ranking approaches (point-, pair-, and listwise) and (2) metalearners.	152
B1	Instance-dependent or class-dependent costs: cost-insensitive metrics per dataset. Significantly better results are denoted in bold (5%) and <i>italic</i> (10%).	219
B2	Instance-dependent or class-dependent costs: cost-sensitive metrics per dataset. Significantly better results are denoted in bold (5%) and <i>italic</i> (10%).	220
B1	Model training. We show the training settings and hyperparameter ranges that were searched, differentiating between general, GAN-related and MLP-related hyperparameters. . .	222
B1	Hyperparameter tuning. We show the optimal hyperparameters for the different data sets. The flow type ‘SigmoidX’ refers to the deep sigmoidal flow (DSF) of [258] conditioned on the balanced representation ϕ of x . During inference, drawing samples from a truncated normal distribution was observed to be more stable for the News data.	225
B1	An overview of existing work. We categorize the related work according to the assumptions made regarding the observation process. Sampling is either assumed to be regular, completely at random (SCAR), or at random (SAR).	228
B2	List of symbols. We compile a list of the main mathematical symbols used and their explanation. The final column illustrates each symbol for the case of a cancer patient.	231
B3	Hyperparameter optimization. We show the range for each hyperparameter that was optimized. The optimal value is shown in bold.	234

List of Figures

B1 **Preprocessor search spaces.** We describe the search spaces for the different preprocessors. If a hyperparameter is not mentioned, we use its default. All preprocessors are implemented with `scikit-learn` [335]; we refer to their documentation for more information. 246

B2 **Baselearner search spaces.** We describe the search spaces for each baselearner. If a hyperparameter is not mentioned, we use its default. All baselearners are implemented with `scikit-learn` [335]; we refer to their documentation for more information. 247

B3 **The effect of k in k -fold cross validation.** For each data set, we show result for a varying number of cross-validation folds. Results for 50 evaluation trials with a T -risk and 50 estimation trials with a T -Learner and gradient boosting. 250

B4 **Comparing metalearner precision.** For each data set, we compare the different metalearner’s performance in terms of $\sqrt{\text{PEHE}}$, with the best result highlighted in **bold**. We also include a comparison with searching over all metalearners (AllMeta) and, in brackets, show how much this outperforms the best single metalearner. For each result, AutoCATE uses a T -risk with 50 evaluation trials, 200 estimation trials, and top 1 average model selection. 252

B5 **Comparing AutoCATE with common benchmarks on CATE estimation.** We compare performance in terms of $\sqrt{\text{PEHE}}$, with the best result highlighted in **bold**. AutoCATE results for a T -risk with 50 evaluation trials and 50 estimation trials with the BestMeta–BestBase configuration. 254

B6 **Software package comparison.** We provide an overview of commonly used packages for CATE estimation and compare their functionalities with AutoCATE, showing whether they support (1) evaluation, (2) estimation, (3) ensembling, and (4) automated, end-to-end optimization—as provided by AutoCATE or similar. 255

B1 *Ranking Quality for Different Objectives and Metalearners.* For each metalearner, we compare three different objectives: point-, pair-, and listwise. We show performance in terms of AUQC (with standard error in brackets), for the **Synthetic**, **Criteo**, and **Hillstrom** (Women (♀) and Men (♂) e-mail), and **Promotion** data. 259

B2 *Analyzing Performance Trade-offs on Synthetic Data.* For each metalearner, we compare three different objectives: pointwise, pairwise, and listwise. Using the **Synthetic** data set, we compare performance in terms of MSE (measuring pointwise accuracy), Kendall τ (measuring pairwise rank correlation), and AUQC (measuring global, listwise decision quality). For each, we show the standard error in brackets. 260

LIST OF FIGURES

1.1	Dissertation overview. This dissertation discusses seven chapters. For each chapter, we indicate whether it contributes to decision-focused learning, causal inference, or both.	9
2.1	Toy example with class-dependent (top) and instance-dependent costs (bottom). (Left) Two classes and the probability distribution are shown, with the instance size proportional to its misclassification cost. (Middle) The resulting decision boundary for a <i>cost-insensitive</i> classifier mimics the underlying probability distribution. (Right) For a <i>cost-sensitive</i> classifier, the decision boundary lies further from the more costly class when costs are class-dependent. With instance-dependent costs, the decision boundary is not only related to the probability distribution, but also the cost distribution.	21
2.2	Overview of the experimental design using the two-staged framework. In the first stage, a predictive model is built by training a type of classifier with an objective function, which can be both cost-sensitive or cost-insensitive. In the second stage, the predictions of this model are used to make decisions. Both the scores and decisions are evaluated.	26
2.3	Cost-insensitive metrics: critical difference diagrams for the AUROC and AP	31
2.4	Cost-sensitive metrics: critical difference diagrams for the AEC and Spearman ρ	32

List of Figures

3.1 **Problem overview.** Our setting concerns a type of linear assignment problem with two sources of uncertainty: stochastic worker capacity and uncertain task outcomes. To account for stochastic capacity in the assignment problem, the capacity distribution is converted to workers with decreasing processing probabilities. Task outcomes are also uncertain and need to be predicted. The key objective is to assign workers to tasks to maximize the resulting expected profit. 46

3.2 **Comparing the methodologies in terms of expected precision and profit.** We plot each methodologies' ranking in terms of expected profit and expected precision on each data set. For each method, the average ranking is shown with a star (\star). Moreover, the ranking density is fitted with a Gaussian kernel; for visual clarity, only probabilities greater than 0.5 are shown. On average, csLambdaMART performs best in terms of expected profit, while LambdaMART performs best in terms of expected precision. 58

3.3 **Evaluating the top k ranked instances.** Precision (**a**) and profit (**b**) for obtained by the top k instances in the ranking for the different models averaged over all data sets. The ranking models outperform the classifiers in the metric they optimize for: LambdaMART is the best in terms of precision; csLambdaMART has the best profit. 59

4.1 **Diagram illustrating the assumed causal relationships between the different variables.** X : Asset characteristics, T : PM frequency, O : Overhaul rate, and F : Failure rate. . . 70

4.2 **Methodology overview.** We present a high-level overview of our methodology. First, machine characteristics \mathbf{x}_i are used to predict the potential outcomes in terms of overhauls $o_i(t)$ and failures $f_i(t)$. Based on these estimates, the total cost for different PM frequencies $t \in T$ can then be estimated. Finally, the PM frequency \hat{t}_i^* is chosen to minimize the total expected cost. 72

4.3 **Semi-synthetic data.** We represent the observed outcomes for contracts in the training and validation set by dots and the potential outcomes for contracts in the test set by a line. The bold lines illustrate the overhaul rate, failure rate, and total cost averaged across all contracts. 78

4.4 **Simulating selection bias.** (4.4a) We simulate a training data set where each contract’s (observed) PM frequency is drawn from its probability distribution, shown in green, using Equation (4.7) for different values of λ . When $\lambda = 0$, each contract has equal probabilities of receiving each PM frequency between 0 and 20, corresponding to randomly assigned PM frequencies. Increasing λ makes the distributions more dependent on contract characteristics and therefore more diverse. This way, certain contracts will more likely receive less frequent PM, resulting in selection bias. Higher values of λ imply more diversity in the distributions and, consequently, more selection bias. (4.4b) We show how the PM frequency is distributed among the different contracts, both in reality and as a result of different values of λ . Larger values of λ result in more selection bias, with a value of 30 resulting in an overall PM frequency distribution close to the original. 80

4.5 **Evaluating the policies’ decisions.** We compare the accuracies and costs of the prescribed PM frequencies by looking at each model’s performance over all contracts. (Left) We show how the differences between the prescribed and optimal PM frequency are distributed per model. (Right) We show the distribution of all contracts’ policy cost ratios resulting from each model. Results are shown for one representative iteration. 84

4.6 **Results for varying levels of selection bias.** We show results for different levels of selection bias in terms of λ (see Equation (4.7)). Although SCIGAN-ITE performs similar to MLP-ITE for lower values of λ , it has better performance for stronger levels of bias in terms of MISE, PE, and PCR. . . . 85

- 5.1 **Optimizing treatment decisions requires knowing the treatment effect distribution.** Predicting the treatment effect distribution $p(\tau)$ allows for assessing a treatment’s utility by pairing it with a utility function $u(\tau)$ and obtaining its expected utility $\mathbb{E}(u(\tau))$. Consequently, the individual treatment effect distribution is instrumental to analyzing treatment utilities (on the left) and comparing treatment preferences (on the right). *Left:* For a given treatment and the corresponding treatment effect distribution $p(\tau)$, we can compare different utility functions $u(\tau)$ and their expected utilities. *Right:* Similarly, for a given utility function $u(\tau)$, different treatment effect distributions $p(\tau)$ incur different expected utilities. These types of analyses are not possible using only the individual’s expected treatment effect $\mathbb{E}(\tau)$, illustrating the importance of estimating the entire treatment effect distribution $p(\tau)$ for personalized decision-making. 91
- 5.2 **NOFLITE architecture overview.** We visualize the S-learner configuration of our method. *Optimization:* NOFLITE learns (1) a conditional prior $p(z|x,t)$ for each instance, and (2) an invertible mapping $\mathbf{g}(y)$ from that conditional prior to the empirical distribution $p(y^{(t)}|x)$, possibly conditioned on x and/or t . Both the encoder and normalizing flow use neural networks and are trained with gradient descent. *Inference:* Based on input (x,t) , the conditional prior $p(z|x,t)$ is estimated. Samples z are drawn from this prior and transformed using the inverse flow $\mathbf{g}^{-1}(z)$ to estimate the distribution of each potential outcome $p(y^{(t)}|x)$ 96
- 5.3 **NOFLITE illustration.** We visualize NOFLITE’s output for selected test instances for one iteration of the News data set. *Individual treatment effect distributions:* Figures (a-c) visualize one particular instance’s treatment effect distribution and related statistics based on samples from the model. *Distribution heterogeneity:* Figure (d) compares the predicted individual treatment effect distributions of several instances. 104

6.1	Problem illustration: sampling mechanisms.	We show an instance’s latent trajectory ($Y(t)$) and sampling intensity ($\lambda(t)$) over time t , along with administered treatments (■) and observations (○) resulting from different sampling mechanisms. (a) Regular. Samples are obtained at regular intervals over time. (b) SCAR. Samples are irregular, drawn at completely random intervals over time. (c) SAR. Sampling times are <i>irregular, but not completely random</i> : e.g., there might be more samples when the outcome is large. We refer to the dependence of the sampling intensity on an instance’s covariates, treatments, and/or outcomes as informative sampling. Whereas existing work in the ML literature assumes regular sampling or SCAR, this work is, to the best of our knowledge, the first to consider learning to forecast treatment outcomes given SAR.	109
6.2	TESAR-CDE: Adapting TE-CDE for learning given SAR.	The history of observations (○) and treatments (■) up to time t is first encoded as a continuous latent path $z(t)$. Based on a future treatment plan $\bar{A}_t(t')$, the decoder then forecasts a future latent path $z_t(t')$, with $t' = t + \tau$. In contrast to TE-CDE, TESAR-CDE (1) uses the latent path $z_t(t')$ to forecast both the outcome $\hat{y}_t(t')$ and intensity $\hat{\lambda}_t(t')$, and (2) uses the intensity to weight the outcome loss using $\mathcal{L}^{\text{WMSE}}$	118
6.3	Comparing TESAR-CDE to TE-CDE.	We show TE-CDE and our proposed alternative, TESAR-CDE, in its two-step and multitask configuration. Arrows indicate the input (⋯→), forward pass (→) and backpropagation (←⋯). The multitask model uses the intensity loss only to train the intensity map, but not the shared encoder or decoder. The dashed arrow (- ->) indicates that the intensities are used as weights λ_t^{-1} in $\mathcal{L}^{\text{WMSE}}$, but not backpropagated as part of this loss.	120
6.4	Results for varying informativeness γ and different forecasting horizons τ.	We show the RMSE \pm SE over ten runs. (Left) RMSE for increasing levels of informativeness γ , keeping the forecasting horizon fixed at $\tau = 1$. (Right) RMSE for an increasing forecasting horizon τ up to five days, keeping informativeness fixed at $\gamma = 6$	123
6.5	Observation scarcity.	We show the RMSE \pm SE over ten runs at increasing observation scarcity S_λ for fixed informativeness ($\gamma = 4$) and forecasting horizon ($\tau = 1$).	124

6.6 **Outcome-unrelated sampling.** We show the RMSE \pm SE over ten runs for a sampling mechanism unrelated to the outcome $Y(t)$ in function of informativeness γ 126

7.1 **AutoCATE enables insights into CATE estimation.** We analyze hundreds of pipelines optimized by AutoCATE (see Section 7.5). *Metalearners*—(a) Different metalearners can be optimal for a data set, highlighting the need for searching across them. (b) The top five pipelines often feature a mix of different metalearners (e.g. $\{T, T, RA, RA, DR\}$: 3 unique types), showing that different metalearners can perform well and suggesting potential for combining them. *Baselearners*—(c) The chosen baselearners are also diverse, and (d) different model types favor different ones. Using a single baselearner is thus likely suboptimal, supporting our choice to tune submodels independently. 130

7.2 **AutoCATE overview.** We estimate treatment effects in three stages: (1) *Evaluation*—learning the appropriate risk measure(s), (2) *Estimation*—tuning a CATE estimation pipeline, and (3) *Ensembling*—selecting a final model or constructing an ensemble. We build ML pipelines for evaluation and estimation based on a collection of *preprocessing algorithms* and *ML baselearners*. 133

7.3 **Evaluation framework.** We show two possible frameworks for validating pipelines based on a single split or a cross-validation procedure. For each, the data is split in three groups to (1) train the estimation pipelines, (2) train the validation pipelines, and (3) validate the validation pipelines. . . 134

7.4 **How many iterations should we tune evaluation models?** We compare downstream results, based on different number of trials, used to tune the models underlying the evaluation metrics. Results for a T -risk and 50 estimation trials with a T -Learner and gradient boosting. 138

7.5 **How much data to use for evaluation?** We show results for different holdout ratios and fit a polynomial function for each data set to gain insight into the optimal ratio. Results for 50 evaluation trials with a T -risk and 50 estimation trials with a T -Learner and gradient boosting. 138

7.6 **What meta- and baselearners to include?** We compare different search spaces for AutoCATE, either including all metalearners (AllMeta) or only the best (BestMeta), as well as all baselearners (AllBase) or only the best (BestBase). Results for 50 evaluation trials with a T -risk. 140

7.7	Comparing AutoCATE with tuning based on μ-risk. We compare tuning a T -Learner with gradient boosting using either AutoCATE (based on a T -risk) or tuning based on the MSE on the observed outcome. AutoCATE uses a T -risk with 50 evaluation trials and top 1 model selection.	142
7.8	Benchmarking AutoCATE. We compare AutoCATE with common benchmarks using S - and T -Learners with random forests and gradient boosting. AutoCATE uses a T -risk with 50 evaluation trials and BestMeta-BestBase search spaces, with either Top 1 or Top 5 model selection.	142
8.1	<i>Evaluating a Treatment Allocation Policy.</i> We compare targeting policies using a Qini curve, depicting the cumulative total effect of a policy for a number of treated instances, summarized by the area under the Qini curve (AUQC).	155
8.2	<i>Ranking Quality for Different Objectives and Metalearners.</i> For each metalearner, we compare a point-, pair-, and listwise version. We show the AUQC \pm one standard error, scaled to have the best result = 1, for five different data sets.	166
8.3	<i>Analyzing Performance Trade-offs on Synthetic Data.</i> We compare the three different objectives (point-, pair-, and listwise) across metalearners. Using the Synthetic data set, we compare performance in terms of MSE (i.e., pointwise accuracy), Kendall τ (i.e., pairwise rank correlation), and AUQC (i.e., listwise decision quality). For each, we show the correlation ρ	167
8.4	<i>What Is the Effect of the Number of Sampling Iterations k?</i> We show performance in terms of AUQC for the different metalearners on the Synthetic data set. We fix the sigmoid parameter $\sigma = 1$ and train with default hyperparameters. . .	168
D.1	Tumor and intensity evolution. We show the simulated tumor size and corresponding intensity over time for several (randomly selected) patients. The intensity is simulated is based on Equation (6.7) with an informativeness $\gamma = 4$	237
D.2	Informative sampling – intensity distribution. We show the distribution of intensities $\lambda(t)$ over all patients for different levels of informativeness γ . At $\gamma = 0$ (not shown), all intensities are equal $\lambda_i(t) = 0.5$	237

List of Figures

D.3 **Informative sampling – expected observations.** We show the expected observations over the entire time period considered over all patients for different levels of informativeness γ . At $\gamma = 0$ (not shown), all intensities are equal $\lambda_i(t) = 0.5$ and all patients have 60 expected observations. 237

D.4 **Uninformative sampling – intensity distribution.** We show the distribution of intensities $\lambda(t)$ over all patients for different levels of “informativeness” γ . At $\gamma = 0$ (not shown), all intensities are equal $\lambda_i(t) = 0.5$ 238

D.5 **Evaluating the intensity prediction at varying informativeness γ .** We show the Brier Score \pm SE (lower is better) over ten runs at increasing levels of informativeness γ , keeping the forecasting horizon $\tau = 1$ 239

D.6 **Evaluating the multitask configuration’s outcome prediction for different levels of hyperparameter α .** We show the RMSE \pm SE over ten runs, while keeping the level of informativeness fixed at $\gamma = 6$ and averaging over $\tau \in \{1, \dots, 5\}$. 240

E.1 **Metalearner selection.** We show how many times a metalearner gets picked (in % of all data set iterations) for a given data set. Results for AutoCATE’s BestMeta configuration, including the *S*-, *T*-, *Lo*-, *X*-, *RA*-, *DR*-, and *U*-Learners, with 50 evaluation and 500 estimation trials. 251

E.2 **Comparing metalearner precision and time efficiency.** We show each metalearner’s performance in precision ($\sqrt{\text{PEHE}}$) and time (excluding outliers, see Table B4). For each, AutoCATE uses a *T*-risk with 50 evaluation trials, 200 estimation trials, and top 1 average model selection. 252

E.3 **Analyzing AutoCATE’s feature importance.** We can analyze how much each feature contributes to treatment effect heterogeneity. We illustrate this analysis for the first iteration of IHDP using permutation feature importance, showing the squared distance to the original prediction when permuting a feature column. 253

E.4 **Assessing uncertainty with AutoCATE.** The ensemble returned by AutoCATE can be used to analyze uncertainty regarding the prediction. We illustrate this for the first 20 instances of the first iteration of the IHDP data. For each instance, the (usually unknown) ground truth is shown in green, while the predictions from the top five pipelines are shown in blue and with a violinplot. 253

E.5	Benchmarking AutoCATE for treatment prioritization. We present additional results in terms of AUQC for two uplift data sets, Hillstrom and Information. These show that AutoCATE is a useful tool for prioritizing instances for treatment, and highlight that its optimization is more effective at optimizing AUQC compared to the benchmarks based on μ -risk. AutoCATE uses a T -risk with 50 evaluation trials and the AUQC metric, the BestMeta-BestBase search space, and Top 1 or Top 5 ensembling.	254
E.6	Analyzing AutoCATE’s results. We present results analyzing pipelines optimized by AutoCATE. Figure (a) shows the correlation between risk measures for a single IHDP iteration. Surprisingly, risk measures can be strongly <i>negatively correlated</i> , suggesting potential for more advanced multi-objective approaches that adaptively learn which objectives are reliable for a given data set. Figure (b) visualizes the <i>optimal pipelines</i> learned across ten iterations for the Twins data.	255
F.1	<i>Ranking Quality for Different Objectives and Metalearners.</i> For each metalearner, we compare three different objectives: point-, pair-, and listwise. We show performance in terms of AUQC \pm one standard error, for five different data sets. As opposed to the figure in the main body, we do not scale the results here. Due to confidentiality reasons, we cannot share the raw results for the Promotion data set.	262
F.2	<i>Analyzing Performance Trade-offs on Synthetic Data.</i> For each metalearner, we compare three different objectives: pointwise , pairwise , and listwise . Using the Synthetic data set, we compare performance in terms of MSE (measuring pointwise accuracy), Kendall τ (measuring pairwise rank correlation), and AUQC (measuring global, listwise decision quality).	263
F.3	<i>What Is the Effect of the Number of Sampling Iterations k?</i> We show performance in terms of AUQC (higher is better) for the different metalearners on the Synthetic data set. We fix the sigmoid parameter $\sigma = 1$ and train with default hyperparameters.	264
F.4	<i>What Is the Effect of the Sigmoid Parameter σ?</i> We show performance in terms of AUQC (higher is better) for the different metalearners on the Synthetic data set. We fix the number of sampling iterations $k = 1$ and train with default hyperparameters.	265

PUBLICATION LIST

Articles in peer-reviewed scientific journals:

- Vanderschueren, T., Baesens, B., Verdonck, T., and Verbeke, W. “Predict-then-optimize or predict-and-optimize? An empirical evaluation of cost-sensitive learning strategies.” *Information Sciences* (2022)
- Vanderschueren, T., Boute, R., Verdonck, T., Baesens, B., and Verbeke, W. “Optimizing the preventive maintenance frequency with causal machine learning.” *International Journal Of Production Economics* (2023)
- Vanderschueren, T., Berrevoets, J., and Verbeke, W. “NOFLITE: Learning to Predict Individual Treatment Effect Distributions.” *Transactions on Machine Learning Research* (2023)
- Vanderschueren, T., Baesens, B., Verdonck, T., and Verbeke, W. “A new perspective on classification: Optimally allocating limited resources to uncertain tasks.” *Decision Support Systems* (2024)

Peer-reviewed conference proceedings:

- Vanderschueren, T., Verdonck, T., Baesens, B., and Verbeke, W. “Instance-dependent cost-sensitive learning: do we really need it?” *Hawaii International Conference on System Sciences* (2022)
- Vanderschueren, T., Curth, A., Verbeke, W., and van der Schaar, M. “Accounting for informative sampling when learning to forecast treatment outcomes over time.” *International Conference on Machine Learning* (2023)

Publication list

Articles currently under review:

- Vanderschueren, T., Verbeke, W., Moraes, F., and Proença, H.M. “Met-learners for Ranking Treatment Effects” *Under Review at The ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (2024).
- Vanderschueren, T., Verdonck, T., van der Schaar, M. and Verbeke, W. “AutoCATE: End-to-End, Automated Treatment Effect Estimation” *Under Review at the International Conference on Learning Representations* (2025).

Abstracts presented at scientific conferences:

- Vanderschueren, T., Berrevoets, J., and Verbeke, W. “NOFLITE: Learning to Predict Individual Treatment Effect Distributions.” *Presented at the DSSV-ECDA 2023, Antwerp, Belgium* (2023).
- Vanderschueren, T., Boute, R., Baesens, B., Verdonck, T., and Verbeke, W. “Prescriptive maintenance with causal machine learning.” *The Control Room of the Future: AI Empowered Dashboards, Gent, Belgium* (2022).
- Vanderschueren, T., Boute, R., Baesens, B., Verdonck, T., and Verbeke, W. “Prescriptive maintenance with causal machine learning.” *Presented at the Belgian Operational Research Society (ORBEL), 36th Annual Conference, Gent, Belgium* (2022).
- Vanderschueren, T., Boute, R., Baesens, B., Verdonck, T., and Verbeke, W. “Optimizing maintenance by learning individual treatment effects.” *Presented at the Workshop on Spurious Correlations, Invariance, and Stability at the International Conference on Machine Learning, Baltimore, Maryland* (2022).
- Vanderschueren, T., Boute, R., Verdonck, T., Baesens, B., and Verbeke, W. “Prescriptive maintenance with causal machine learning.” *Presented at the European Conference on Operational Research (EURO), Helsinki* (2022).
- Vanderschueren, T., Boute, R., Verdonck, T., Baesens, B., and Verbeke, W. “Failure Prediction vs. Maintenance Prescription: Optimizing Maintenance Interventions by Learning Individual Treatment Effects.” *Presented at the 22nd International Working Seminar on Production Economics, Innsbruck, Austria (online)* (2022).
- Vanderschueren, T., Baesens, B., Verbeke, W., and Verdonck, T. “Instance-dependent cost-sensitive learning: Do we really need it?” *Presented at*

the 31st European Conference on Operational Research, Athens (2021).

Vanderschueren, T., Baesens, B., Verbeke, W., and Verdonck, T. “An empirical evaluation of cost-sensitive learning strategies.” *Presented at the Eastern European Summer School 2021, Budapest (2021).*

Vanderschueren, T., Baesens, B., Verbeke, W., and Verdonck, T. “Instance-dependent cost-sensitive learning: an empirical evaluation.” *Presented at the ORBEL 2021 Workshop on Business Analytics (2021).*

A full list of the doctoral dissertations from the Faculty of Economics and Business can be found at:

[https://www.kuleuven.be/english/research/
doctoraldefences/archive](https://www.kuleuven.be/english/research/doctoraldefences/archive)