UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

Machine learning applications in operations management and digital marketing

Wang, Q.

Publication date
2019
Document Version
Final published version
License
Other

Link to publication

Citation for published version (APA):
Wang, Q. (2019). *Machine learning applications in operations management and digital marketing*. [Thesis, externally prepared, Universiteit van Amsterdam].

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)

Download date:05 Sept 2026

With the rise of data collection also comes the need for methods to analyze the data, and for novel approaches of leveraging the data to improve decision-making. Machine learning is a set of algorithms with the primary purpose of identifying patterns in data to accomplish specific tasks. Since 2012, the number of Google searches containing the term machine learning has increased by 7 fold, and the data scientist, an occupation that often requires machine learning skills, has been rated by Glassdoor as the best job in America three years in a row. Academic publications in machine learning have also enjoyed a significant popularity boost. Although machine learning has been wildly popular, its scientific success has mostly been captured in the fields of computer science and artificial intelligence. The use of machine learning in business research, specifically in the areas of operations management and digital marketing, has been limited. In this dissertation, I study how machine learning can be used to solve prominent problems in operations management and digital marketing. The primary motivation is to show that the application of machine learning can solve problems in ways that existing approaches cannot. In its entirety, this dissertation is a study of four problems—two in operations management and two in digital marketing—and develops solutions to these problems via data-driven approaches by leveraging machine learning. These four problems are distinct, and are presented in the form of individual self-containing essays. Each essay is the result of collaborations with industry partners and is of academic and practical importance. In some cases, the solutions presented in this dissertation outperform existing state-of-the-art methods, and in other cases, it presents a solution when no reasonable alternatives are available.

5 800133 952093

Machine Learning Applications in Operations Management and Digital Marketing

Qingchen Wang

Machine Learning Applications in Operations Management and Digital Marketing

Qingchen Wang

Machine Learning Applications in Operations Management and Digital Marketing

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het College voor Promoties ingestelde commissie,
in het openbaar te verdedigen in de Aula der Universiteit
op vrijdag 14 juni 2019, te 13.00 uur

door **Qingchen Wang**

geboren te Zhengzhou

Promotiecommissie:

Promotores

Prof. dr. M. Salomon	Universiteit van Amsterdam
Prof. dr. G.M. Koole	Vrije Universiteit Amsterdam

Copromotores

dr. M.J. Soomer	Universiteit van Amsterdam
dr. K. Pak	Universiteit van Amsterdam

Overige leden

Prof. dr. R.J.M.M. Does	Universiteit van Amsterdam
dr. S. Rudinac	Universiteit van Amsterdam
dr. A.V. den Boer	Universiteit van Amsterdam
Prof. dr. S. Bhulai	Vrije Universiteit Amsterdam
Prof. dr. R.D. van der Mei	Vrije Universiteit Amsterdam

Faculteit Economie en Bedrijfskunde

Acknowledgements

Wow, it's finally done! It has been a long journey, and I could not have made it alone. I have a lot of people to thank for helping me to get here.

First and foremost, I would like to thank my promotors, Prof. Ger Koole and Prof. Marc Salomon, for giving me the opportunity to complete this thesis. I am very fortunate to have the two of you as my academic advisors, and it is only with your advice and guidance that I can be here today. I am also indebted to Prof. Sandjai Bhulai, Dr. Arnoud den Boer, Dr. Kevin Pak, and Dr. Maarten Soomer for the valuable discussions over the past few years that have helped my research tremendously.

The first three years of my PhD involved working part-time at two companies, ORTEC and Objective Partners. This was a unique experience, and I am very glad to have been a part of this arrangement. Thank you to all of my former colleagues at ORTEC and Objective Partners. I really enjoyed working with all of you, and best of all, our time together—whether it was playing table football or table tennis—was just so much fun.

I must thank all of my colleagues from both the Analytics and Optimization group at the Vrije Universiteit Amsterdam, and the Operations Management group at the Amsterdam Business School. For the past four years, these groups have been my academic home, and I really cannot imagine a better group of students and staff to be around. To the (soon-to-be) Dr. Ruben van de Geer, thank you for being my officemate. It is always nice to know at times that I am not the only one struggling. I wish you the best of luck in your future endeavors.

Finally, I want to acknowledge my former classmates and professors from INSEAD, University College London, and the University of British Columbia. Together, we had a lot of good times and a lot of challenges. Some of you have already, or will in the future complete your PhDs. I hope your journeys were or will be just as enjoyable and fruitful as mine.

To my parents, thank you for being supportive of this complete academic journey.

To Yijun, thank you for being the most important part of my life throughout my PhD. Your companionship has turned my struggles into joy, and together, everyday has been the best it possibly could be.

Contents

1 Introduction **11**

1.1 Contribution and valorization 13

1.2 Data requirements and industry collaboration 14

2 Machine Learning Essentials **17**

2.1 Supervised learning 18

2.1.1 Classification 19

Classification problems studied in this dissertation. 20

2.1.2 Regression 21

Regression problems studied in this dissertation. 21

2.2 The predictive analytics process 22

2.2.1 Performance evaluation and validation 22

Evaluation metrics for binary classification. 22

Evaluation metrics for regression. 26

Validation. 27

2.2.2 Feature engineering 28

Timestamped events. 29

Mathematical feature transformations. 30

Categorical features. 30

Pairwise comparisons. 31

2.2.3 Supervised learning algorithms 31

Linear models. 32

Nonlinear models. 33

Tree-based models. 35

3 Data-driven Consumer Debt Collection **37**

3.1 Introduction 37

3.1.1 Background and motivation 37

3.1.2 Main contributions 39

3.1.3 Relation to literature 40

Debt collection optimization. 40

Credit scoring and valuation. 41
3.1.4 Outline 42
3.2 Problem description 42
3.2.1 High-level overview 42
3.2.2 Collection phase 43
3.3 Model description 44
3.3.1 State space 44
3.3.2 Action space and value function 46
3.4 Value function approximation with machine learning 47
3.4.1 Estimating the predicted repayment probability 48
3.4.2 Approximating the value function 49
3.4.3 Gradient boosted decision trees 50
3.5 Data description 52
3.6 Model estimation results 54
3.6.1 Training and validation data 54
3.6.2 Debt repayment prediction performance 56
3.6.3 Illustration of individual PRP trajectories 57
3.6.4 Feature importance for predicting PRP 58
3.6.5 Marginal effect of phone calls on PRP 61
3.7 Controlled field experiment 63
3.7.1 Experimental setup 63
3.7.2 Experimental results 65
3.7.3 Comparison of GOCP and IP 66
GOCP spends more effort on difficult cases. 67
Timing of phone calls. 68
Similarities between GOCP and IP. 68
3.8 Conclusion 68

4 Contact Center Staffing and Scheduling 71
4.1 Introduction 72
4.1.1 Contribution 74
4.1.2 Related literature 75
Call centers. 75
Machine learning for simulation. 76
Outline. 77
4.2 Problem description 77
4.2.1 High level description of contact centers 77
4.2.2 Mathematical formulation 78

4.3 Simulation setup . . . 80
4.4 Machine learning approach . . . 81
4.4.1 Gradient boosted decision trees . . . 82
4.4.2 Feature construction . . . 83
4.5 Service level prediction performance . . . 84
4.5.1 Train and validation framework . . . 85
Sampling data from simulation. . . . 85
4.5.2 Prediction performance . . . 86
4.6 Numerical experiments . . . 89
4.6.1 Optimization procedure . . . 89
4.6.2 Mathematical approximation and simulation optimization approaches . . . 90
4.6.3 Optimization results . . . 93
Single-skill scenario. . . . 93
Multi-skill (calls only) scenario. . . . 93
Multi-skill (calls plus chat and email) scenario. . . . 94
4.7 Conclusion . . . 94

5 Multi-channel Conversion Attribution 97
5.1 Introduction . . . 97
5.1.1 Background and motivation . . . 97
5.1.2 Shapely value-based attribution . . . 100
5.1.3 Contribution . . . 101
5.1.4 Outline . . . 103
5.2 Overview of existing data-driven attribution methods . . . 103
5.2.1 Probabilistic models . . . 104
5.2.2 Markov models . . . 104
5.2.3 Point processes . . . 105
5.2.4 Vector autoregression models . . . 105
5.3 Attribution framework . . . 105
5.3.1 Shapley value . . . 106
5.3.2 Shapley value modified for an ML predicted conversion rate . . 107
5.3.3 Fairness of Shapley value attribution—five properties and their axioms . . . 108
5.4 Data description . . . 110
5.5 Conversion prediction with machine learning . . . 112
5.5.1 Gradient boosted decision trees . . . 113
5.5.2 Feature engineering . . . 114

5.5.3 Prediction performance . . . 115
5.6 Attribution with machine learning . . . 117
5.6.1 Attribution Results . . . 118
5.6.2 How SVA and MLS differ . . . 119
Rare journey occurrences of SVA. . . . 120
Effect of customer behavior information. . . . 121
5.6.3 Which rule-based attribution methods are reasonable? . . . 122
5.6.4 Which channels are more effective? . . . 123
5.7 Conclusion . . . 125

6 Recombining Customer Journeys 127
6.1 Introduction . . . 128
6.2 Related work . . . 129
6.3 Data description . . . 130
6.4 Supervised learning approach . . . 131
6.4.1 Feature engineering . . . 132
6.4.2 Learning algorithm . . . 133
6.4.3 Evaluation metric . . . 133
6.5 Prediction results and evaluation . . . 133
6.5.1 Performance with feature subsets . . . 135
6.5.2 Using cookies from one city without down-sampling . . . 136
6.6 Conclusion . . . 137

7 Improving Display Advertising 141
7.1 Introduction . . . 141
7.1.1 Relation to literature . . . 144
Structure of this chapter. . . . 146
7.2 Business process and data collection . . . 146
7.2.1 Retargeting advertising . . . 147
7.2.2 Data collection and description . . . 148
7.2.3 Ground truth data generation . . . 148
7.3 Device-matching algorithm . . . 149
7.3.1 NLP approach to modeling website visits . . . 150
URL structure and hierarchy. . . . 150
Term frequency-inverse document frequency (TF-IDF). . . . 151
Doc2Vec representation of URLs. . . . 151
7.3.2 Initial candidate selection . . . 151
7.3.3 Pairwise device matching via machine learning . . . 152
7.3.4 Feature engineering . . . 153

7.4 Device matching performance 153
7.4.1 Candidate selection 154
7.4.2 Pairwise matching 154
7.5 Economic impact of cross-device matching 156
7.6 Conclusion 158

8 Bibliography **159**

Chapter 1

Introduction

As the world becomes more and more digitized, we are now able to collect enormous amounts of data about everything, from how customers shop for groceries to how Formula One racecars behave throughout every millisecond of a 300km race. With the rise of data collection also comes the need for methods to analyze the data, and for novel approaches of leveraging the data to improve decision-making. Machine learning is a set of algorithms with the primary purpose of identifying patterns in data to accomplish specific tasks. It originated in computer science, and due to the increase in available data and advance in computing power, has become immensely popular in recent times. Since 2012, the number of Google searches containing the term "machine learning" has increased by 7 fold, and the "data scientist", an occupation that often requires machine learning skills, has been rated by Glassdoor as the best job in America three years in a row. Academic publications in machine learning have also enjoyed a significant popularity boost. For example, the paper on image classification using deep convolutional neural networks (a machine learning algorithm) by Krizhevsky et al. (2012) has been cited over 35,000 times, an astronomical number by academic standards. Although machine learning has been wildly popular, its scientific success has mostly been captured in the fields of computer science and artificial intelligence. The use of machine learning in business research, specifically in the areas of operations management and digital marketing, has been limited. In this dissertation, I study how machine learning can be used to solve prominent problems in operations management and digital marketing. The primary motivation is to show that the application of machine learning can solve problems in ways that existing approaches cannot. In its entirety, this dissertation is a study of four problems—two in operations management and two in digital marketing—and develops solutions to these problems via data-driven approaches by leveraging machine learning. These four problems are distinct, and are presented in the form of individual self-containing essays. Each essay is the result of collaborations with industry partners and is of academic and practical importance. In some cases, the solutions presented in this dissertation outperform existing state-of-the-art methods, and in other cases it presents a solution when no reasonable alternatives are available.

The first problem studied in this dissertation is on consumer debt collection (Chapter 3). For this problem, I develop a data-driven algorithm to optimize when and to whom phone calls should be made to maximize the collection of delinquent debt cases. This algorithm was tested in a controlled experiment at a Dutch collection agency and was found to have increased the amount of debt collected per call by 47.2%. The second problem studied is on contact center staffing and scheduling (Chapter 4). For this problem, I develop a machine learning approach to accurately approximate a complex simulation of contact centers, leading to a fast and reliable method for identifying high-quality staffing schedules at low costs. Using numerical simulations that represent real-life contact centers, it is found that my approach can improve upon the existing approaches by over 4%, and is able to analyze more complex contact centers than previously possible. The third problem studied in this dissertation is on the attribution of online purchases to digital advertisements (Chapter 5). For this problem I develop a new attribution model that extends a well-known existing framework to incorporate customers' web-browsing behavior when evaluating the effectiveness of digital advertisements. Using data from a Dutch online travel agency, it is shown that customers' web-browsing behavior are highly predictive of purchasing decisions, and thus should be taken into account when attributing purchases. This solution is currently the only attribution model that is able to incorporate web-browsing behavior at the individual customer level. The fourth problem I study focuses on probabilistically matching web-browsing devices (or browser cookies) to users based on browsing behavior. I consider two different instances of this problem, one of devices browsing a single news publishing website (Chapter 6) and another of devices captured by an ad exchange (Chapter 7), and develop solutions to them separately. In both cases, I show that matching can be performed with good reliability, and that display advertising firms can potentially use this technology to improve their advertising effectiveness.

The essays presented in this dissertation (Chapters 3 to 7) assume that readers already have some basic understanding of machine learning and predictive analytics. For readers who are new to machine learning, either conceptually or practically, Chapter 2 aims to provide you with such knowledge and discusses the essentials of machine learning, such as the type of tasks it is able to perform, how it works to perform certain tasks, and how its performance should be evaluated. It also provides a brief overview of how machine learning is used in the remaining chapters of this dissertation. Readers who already have a good understanding of machine learning can skip Chapter 2 and start with Chapter 3.

1.1 Contribution and valorization

This section describes the contributions of myself and my coauthors with respect to the essays presented in this dissertation. Chapter 3 is based on the paper "Data-driven Consumer Debt Collection via Machine Learning and Approximate Dynamic Programming" (van de Geer et al. 2018), which is currently under review at *Management Science*. This paper is jointly authored with my colleague Ruben van de Geer and Prof. Sandjai Bhulai. As agreed upon by myself and Ruben, the authorship of this paper is ordered alphabetically between the two of us to indicate equal contribution as doctoral students. The conception of this project, together with the ideation, mathematical modeling of the optimization approach, and writing are shared between Ruben and me. The development of the machine learning solution, together with the generation and analysis of the results can be solely attributed to me. Prof. Bhulai contributed to verifying our approach and wrote small parts of the paper. This paper has been presented multiple times at practitioner events (KaggleDays in Warsaw, Data Science Meetups in Amsterdam and Utrecht), academic conferences (POMS Sydney 2017, POMS Hong Kong 2018, StochMod 2018, MSOM 2018, INFORMS Annual Meeting 2018), and academic seminars at various research institutions (University of Kansas, Vanderbilt University, Northwestern University, HEC Montreal, McGill University, Chinese University Hong Kong at Shenzhen, City University Hong Kong, University of Hong Kong, York University, University of Maryland).

Chapter 4 is based on the paper "Optimal Contact Center Staffing and Scheduling with Machine Learning" (Li et al. 2018). This paper is currently being edited and finalized for eventual submission to *Management Science*, and is jointly authored by my colleague Siqiao Li, myself, and our doctoral advisor Prof. Ger Koole. As agreed upon by Siqiao and I, the authorship of this paper is ordered alphabetically between the two of us to indicate equal contribution as doctoral students. The conception of this project can be entirely attributed to myself, although the problem of contact center staffing and scheduling is well-studied in the operations management community and was first brought to my attention by Prof. Koole during a seminar. The contact center modeling and simulation part is mostly performed by Siqiao together with CCMath (a company specializing in workforce planning software for contact centers), and I contributed by designing, implementing, and testing of the machine learning algorithm together with the local-search optimization procedure. Prof. Koole's contribution to this paper is primarily high-level supervision. This paper has been presented at the INFORMS Service Science Conference 2018

and the Service Engineering: From Theory to Practice workshop 2019 at the Israel Institute of Technology (Technion).

Chapter 5 is based on the paper "Multi-channel Conversion Attribution: A Machine Learning Approach" (Peeperkorn et al. 2019). This paper is currently being edited and finalized for eventual submission to *Marketing Science*, and is jointly authored by my colleague Piet Peeperkorn, my co-supervisor Dr. Maarten Soomer, and myself. The authorship of this paper is ordered alphabetically. Multi-channel conversion attribution is a well-studied problem in the marketing and computer science fields, and it was initially proposed as the primary topic of my doctorate. Beyond the initial proposal of the problem, most of the work in this paper can be attributed to myself, including modeling, designing, and implementing the machine learning approach, generation and analysis of the results, and most of the writing. Piet provided expertise in understanding and writing about the Shapley value, and Dr. Soomer wrote parts of the introduction of the paper. This paper has been presented at the Digital Marketing and Machine Learning Conference 2018 at Carnegie Mellon University.

Chapter 6 is based on the paper "Recombining Customer Journeys With Probabilistic Cookie Matching: A Supervised Learning Approach" (Wang 2017). This paper was originally submitted to the INFORMS Data Science Workshop 2017 but was unfortunately rejected, primarily because the practical impact of probabilistic cookie matching has not been proven. Future work to extend this paper has been planned for later in 2019. The work in this paper can be entirely attributed to me.

Chapter 7 is based on the paper "Improving Display Advertising With Predictive Device Matching: A Machine Learning Approach" (Wang and Wijnsma 2018). This paper is co-authored by myself and colleague Taco Wijnsma, and a similar version of this paper was written as the MBA thesis of Taco Wijnsma under my supervision. All work in this specific version of the paper can be attributed to me, although discussions between Taco and I played a major role in certain algorithmic choices. Moreover, as the manager of Adscience at the time, Taco provided the data used for this project. This paper was presented at the POMS Hong Kong Conference 2018 at the Chinese University Hong Kong.

1.2 Data requirements and industry collaboration

The research work in this dissertation are data-driven in nature. With the exception of Chapter 4, real-life data is used for algorithm development and evaluation. Even for Chapter 4, the data provided comes from a simulation based on real-life contact

center scenarios. A primary challenge in data-driven research is obtaining the necessary data to study the research question, often limiting the scope of the study or the ability to conduct certain analyses. For this reason, collaboration and cooperation from industry partners are crucial. I would like to acknowledge the industry partners for providing the data needed to conduct the research in this dissertation. Data collection, storage, and maintenance are extraordinarily time-consuming and costly. Without high-quality data this dissertation would not be possible and data-driven research of modern scale problems would remain nothing more than an idea.

Chapter 3 is in collaboration with a mid-sized Dutch collection agency. This company provided data in the form of basic information about a set of debt collection cases, collection actions taken on debtors, and the ultimate collection results. They also accommodated a controlled field experiment that was used to evaluate our algorithm in a real-life setting. As a result of this collaboration, the paper described in Chapter 3 is the first to validate the effectiveness of data-driven debt collection algorithms in practice.

Chapter 4 is in collaboration with a small company that develops contact center workforce planning software, and the simulation scenario used comes from a mid-sized European contact center. This work is only possible because of the detailed contact center simulation software provided by the partner, which generated the data necessary to develop and evaluate our machine learning framework. This would not have been possible with only observational data due to the required sample size.

Chapter 5 is in collaboration with a Dutch online travel agency. Any research related to the prediction or attribution of conversion behavior at the user level requires detailed data from websites that have tracked the browsing histories of their users. The collection of this data is a challenging technical task as millions of users visit the website and browse hundreds of millions of pages each year. This task typically cannot be performed by research teams, and therefore requires industrial level engineering teams to collaborate in data collection. Our partner spent multiple years on this process prior to the start of this research project to ensure that the data collected is correct and reliable.

Chapter 6 is in collaboration with a large Dutch news publisher. The most challenging aspect of probabilistic device matching is obtaining labeled data for training supervised learning algorithms. By combining user data from logged-in accounts across multiple devices, the news publisher is able to deterministically map the browsing histories of multiple devices to the user of the account. This provided the ground truth required for development and evaluation of device matching algorithms. Our partner spent multiple years on data collection, storage, and maintenance prior to the start of this research project.

Chapter 7 is in collaboration with a small Dutch programmatic buyer of display advertisements. Programmatic buying of display advertisements is technically demanding, requiring decisions to be made within milliseconds through an exchange platform. Billions of decision requests are made every day for web users across the Netherlands alone. For this project, our industry partner collected tens of billions of decision requests from the Netherlands, and filtered it down to the hundreds of millions of requests used to develop and evaluate the method used in our paper.

Chapter 2

Machine Learning Essentials

Machine learning generally deals with numerical data that is in tabular form stored in an N-by-M matrix, where each row $n = 1, ..., N$ represents an "observation" and each column $m = 1, ..., M$ represents a "feature". Table 1 presents a simple example of the data format used by machine learning algorithms. An observation is an entity or event that is defined by a set of M features that describe this observation. For example, it could be a customer of a fashion retailer, where relevant features of this customer may include the customer's age, gender, the products he or she has purchased in the past, and loyalty programs the customer has subscribed to. Typically, non-numerical features are converted into numerical representations (e.g., gender can be represented by a binary value: 1 for female and 0 for male).

Based on a dataset, the tasks that machine learning algorithms perform can be broadly categorized into two groups: supervised learning and unsupervised learning. For supervised learning, the algorithm is used to map a dataset's features to an unknown target value of interest, commonly known as a "prediction". Using the example above, we may be interested in predicting which customers of the fashion retailer will visit the store next week, possibly with the goal of sending a coupon to these customers. Using information (the features) about each customer in our dataset, a supervised learning algorithm may be used to predict the probability that a customer will visit the store next week. For unsupervised learning, the algorithm is used to find implicit categorizations of the observations based on their features. Again using the same example, the fashion retailer may want to group its customers based on their past purchases. An unsupervised learning algorithm can be used to

Table 1: Example of a data format for machine learning.

ID	Feature 1	Feature 2	Feature 3	Feature ...	Feature M
1	Value Feature 1	Value Feature 2	Value Feature 3	Value Feature ...	Value Feature M
2	Value Feature 1	Value Feature 2	Value Feature 3	Value Feature ...	Value Feature M
3	Value Feature 1	Value Feature 2	Value Feature 3	Value Feature ...	Value Feature M
...	Value Feature 1	Value Feature 2	Value Feature 3	Value Feature ...	Value Feature M
N	Value Feature 1	Value Feature 2	Value Feature 3	Value Feature ...	Value Feature M

recommend groupings of customers that most distinctively separate them based on the data available. Human analysts can then annotate these groups based on their general characteristics (e.g., high spenders vs. bargain hunters). This dissertation focuses exclusively on supervised learning tasks due to the ability to objectively evaluate and interpret the results. While unsupervised learning is undoubtedly useful, human intuition is required to interpret results and therefore complicates analyses. The next section provides more details about the supervised learning task.

2.1 Supervised learning

Typically, supervised learning is also synonymous with "predictive analytics". A supervised learning algorithm consists of a single or a set of functions that map values for a set of features into a single target value of interest (prediction). The parameters of the functions are initially unknown, and the automated process of finding parameters that best map feature values to predictions is called "training" a model. To train a model, we first require that labels of the target values be provided together with the dataset of features[1]. This is often provided in historical data, where not only the features, but also labels of the target value have been observed. Going back to the example of the fashion retailer, if we record every time a customer visits the store, then we can look back in time and construct the value of whether the customer will visit the store one week in the future (from a particular point in time in the past). The dataset with labeled target values constructed is called the "training set" or "training data". After a training set is provided, the supervised learning algorithm finds parameters that best map the feature values to the labeled target values for all observations. Equation (1) below presents a mathematical definition of the supervised learning problem:

$$\min_{\boldsymbol{k}} \; g\big(f(\boldsymbol{X}, \boldsymbol{k}),\, \boldsymbol{y_{true}}\big)\,, \tag{1}$$

where $g(\cdot)$ is an error function that takes as input a function $f(\cdot)$ to predict the target value from a matrix of feature values $\boldsymbol{X}$ and a vector of parameters $\boldsymbol{k}$, and the vector of true labeled target values $\boldsymbol{y_{true}}$. The goal is thus to find the parameters $\boldsymbol{k}$ that minimize the error. The choice of error function $g(\cdot)$ and prediction function $f(\cdot)$ is decided by the user and is both problem and data specific (i.e., some error functions are more appropriate for some problems than others). The procedure to find the best parameters $\boldsymbol{k}$ can either be mathematical or computational. After an

[1]The term "supervised" in supervised learning refers to the process of an algorithm "learning" from labeled observations.

algorithm is trained, predictions can then be made on a dataset where the feature values are available but not the labeled target values (e.g., customers in the present day). This dataset is called the "test set".

Supervised learning problems or tasks are often divided into two types: "classification" and "regression". The difference in these two types of problems depends on whether the target variable is categorical (i.e., can only be one among a set of numerically unrelated values), or numerical (i.e., can be any continuous quantity in $\mathbb{R}$). In practical settings, classification and regression problems are not very different, and mistakenly using a classification algorithm to solve a regression problem or vice versa can still produce good results[2]. In fact, it is often straightforward to extend many of the well-known supervised learning algorithms to appropriately handle either problem. However, when applying supervised learning to specific problem domains, it is important to understand the distinction between classification and regression, and to use the correct method. In this dissertation, both classification and regression problems play prominent roles so more details and some examples are provided below.

2.1.1 Classification

When we think of classification we usually think of binary yes/no questions. Will it rain tomorrow? Is this email genuine or is it spam? Is this an image of a cat or a dog? Does this CT scan show cancer or healthy cells? Binary questions are the most common form of classification problems. In other cases, there may be multiple classes to choose from (e.g., which animal is this an image of)? This dissertation only focuses on binary problems but it is simple to extend solutions for the binary problem to solve multi-class problems.

In a binary classification problem, the target value can only be a class, typically represented numerically by 1 if the observation belongs to one class, and 0 if it belongs to the other class. The procedure to train an algorithm is based on finding parameters for functions to best map feature values to the correct class. This is usually evaluated as the fraction of correctly classified observations from the training set, where a correct classification is made when the predicted class equals the labeled class.

A different way of framing the binary classification problem is to treat it as a probabilistic prediction problem. In this context, rather than predicting a binary class (1 or 0) for an observation, we instead predict the probability that the observation

[2]Assuming reasonable post-processing of predictions.

belongs to the class. The observed labels are still binary, but by predicting a probability, the algorithm is also able to demonstrate the confidence of its prediction. Evaluation of this problem is done by considering the difference between the predicted probability of the observation belonging to the class and its true class label. This means that if the true class label of an observation is 1, then a prediction of 0.99 will be better than a prediction of 0.65. On the contrary, if the true class label was 0, then the prediction of 0.65 would be better than the prediction of 0.99. In many cases, this approach is equal to the concept of maximum likelihood estimation in statistics, where the algorithm is finding parameters with the objective of maximizing the likelihood of jointly observing the feature and labeled target values in the training set. The probabilistic prediction problem can also be used to give binary predictions. To do so, a threshold between 0 and 1 needs to be set such that all predictions above the threshold are set to 1, and all predictions below the threshold are set to 0.

Classification problems studied in this dissertation. In this dissertation, Chapters 3, 5, 6, and 7 solve classification problems. In all cases, the classification problem is framed as a probabilistic prediction problem, but two of the chapters require actual class predictions so thresholding is also applied.

A major part of Chapter 3 requires the prediction of the probability that a debtor will fully repay his debt based on details of his current state in the debt collection process. Here, the probability of repayment is the focus, and we compare the differences in predicted probabilities between taking two different actions—call or not call. In this chapter, it is shown that even with very limited information (i.e., not very informative features) about debtors and the nature of their debt, a classification algorithm can predict repayment probability of debtors reasonably well across various collection states. This ultimately allows collection decisions to be optimized based on predicted responses.

The classification problem in Chapter 5 is to predict the probability that customers of an online travel agency will purchase a vacation package. In this problem, a customer is defined by information extracted from his visits to the travel agency's website. For every visit made by the customer, his behavior during the visit is recorded and afterward his purchase probability is predicted. The increase in predicted purchase probability after each visit is then interpreted as the effect that the visit had on the customer's purchase decision. Similar to the debt repayment prediction problem in Chapter 3, in the purchase prediction problem here we also focus on the difference in probability between two actions—visit or no visit.

Chapters 6 and 7 study the problem of matching browsing devices belonging to the same user. Different contexts of this problem are considered in the two chapters, but only a single classification problem is solved. The goal of this problem is to compare each pair of browsing devices that have been used to browse a website (or set of websites) and predict whether the pair of devices belong to the same user. Browsing devices are tracked by websites as HTTP cookies, and it is typically assumed that a unique cookie represents a unique user of the website. However, in practice a unique cookie can only track a user on a single device for a certain period of time, so often a user is tracked by multiple cookies at the same time (due to using multiple devices, using multiple browsers, or refreshing cookies). By reliably predicting which devices belong to the same user, tracking cookies can be combined and websites will have more accurate information about their users, either for reporting purposes or to better understand their users' needs. For targeted online advertising, this can also increase the pool of certain desirable targets, potentially increasing effectiveness.

2.1.2 Regression

Regression problems differ from classification in that the target value is not explicitly constrained to within a limited set and can be any real number. There are many cases of regression-based questions asked every day in businesses. How many customers will visit my retail store next week? What will the temperature be tomorrow? How severe in dollar value will an insurance claim be? Because the target value can in principle be any real number[3], it is unlikely for any prediction to be exactly correct, so predictions are evaluated by how close they are to the true label. The regression problem in machine learning is analogous to that of in statistics. However, a key difference is that in machine learning the primary focus is on predictions made by the algorithm, whereas in statistics the primary focus is on the parameters of the functions for the purpose of statistical inference.

Regression problems studied in this dissertation. In this dissertation, Chapter 4 studies a regression problem. In this chapter, a supervised learning algorithm is developed to predict the weekly service level of a contact center given a staffing schedule of customer service agents. The service level is labeled as a continuous value between 0 (poor) and 1 (excellent), and predictions also follow the same format. It is shown that weekly service levels of contact centers can be accurately predicted when customer demand is stationary (i.e., does not exhibit large exogenous fluctuations).

[3]Sometimes there can be implicit constraints, e.g., there cannot be negative numbers of customers visiting a retail store, or a negative dollar value of insurance claims.

With reliable predictions of the service level, it becomes possible to more efficiently compare the performance of different staffing schedules, allowing for the possibility of finding better schedules within practical time constraints.

2.2 The predictive analytics process

Between when raw data is available and when an algorithm can be deployed in practice, a number of steps need to be taken to ensure that predictions can be of high quality for its intended purpose. The papers presented in Chapters 3 to 7 already assume the reader has a basic understanding of this process. For readers who are not familiar with various steps in the predictive analytics process, this section is intended to provide the necessary background understanding.

Predictive analytics consists of three major components: choosing a supervised learning algorithm, constructing features from raw data to best represent the available information, and evaluating the predictive performance of the algorithm. These three components are considered in tandem, and multiple iterations are often required to achieve good results.

2.2.1 Performance evaluation and validation

Although it may seem counterintuitive, it is often best to start a predictive analytics project with performance evaluation. The first question to ask is: what is the objective, and how should predictions be evaluated? Recall function $g(\cdot)$ from Equation (1), this can be called the error metric and it can be used to translate the practical supervised learning problem into a mathematical function. It is important to select the error function that is consistent with how the practitioner wishes to evaluate predictions. For example, in a problem where an advertiser has limited ads he can present to a large group of customers, he needs to select the customers that are most likely to respond positively to his ad. In this problem, it is customary to first predict the probability that each customer will respond to the ad, and then show the ad to those most likely to respond. For this problem, it is not important to know if the exact prediction of responding or not responding is correct, rather the group selected to show the ad should be the most likely to respond. Therefore, to achieve better prediction results, a metric abbreviated as AUC should be used. Some well-known evaluation metrics are explained below.

Evaluation metrics for binary classification. A number of error functions for binary classification problems rely on the "confusion matrix". The confusion matrix is a 2-by-2 table consisting of four elements: "true positive", "true negative", "false

positive", and "false negative". Predictions for each observation can be assigned to one of the elements. True positive (TP) is when the prediction is 1 and the true label is 1, true negative (TN) is when the prediction is 0 and the true label is 0, false positive (FP) is when the prediction is 1 and the true label is 0, and false negative (FN) is when the prediction is 0 and the true label is 1. Within each element of the confusion matrix, the number of prediction outcomes that matches the element is recorded.

The most simple and intuitive evaluation metric for classification problems is "accuracy", defined by:

$$Accuracy := \frac{\#TP + \#TN}{\#TP + \#TN + \#FP + \#FN}. \tag{2}$$

Accuracy is a useful metric because it directly represents how often the algorithm is correct versus how often it is wrong. This is easy to understand for humans, and is a quick way of convincing people of an algorithm's effectiveness. However, in practice accuracy is a poor metric when the classes are imbalanced. In many prediction problems there are few observations in one class compared to the other class. For example, very few customers respond to advertisements in general, so the responder class is much smaller than the non-responder class. In such cases, when using accuracy as the evaluation metric, a naive algorithm would perform very well (e.g., over 90%) by predicting all observations to be the dominant class. This leaves little room for improvement by an intelligent algorithm, potentially decreasing the efficiency of the algorithm to learn correct relationships. In another context, it may be more important to be correct on positive observations than on negative observations or vice versa. Accuracy treats all observations equally, but in problems such as cancer classification, the value of correctly classifying cancer when it is present is far greater than the cost of incorrectly classifying cancer when it is not present. A different set of evaluation metrics are required for such situations: "precision", "recall", and "F_β-score".

Precision can be interpreted as how correct the algorithm is when it predicts positive. Mathematically it is defined by:

$$Precision := \frac{\#TP}{\#TP + \#FP}. \tag{3}$$

Recall can be interpreted as the fraction of true positive observations the algorithm is able to retrieve (i.e., predict as positive). It is mathematically defined as:

$$Recall := \frac{\#TP}{\#TP + \#FN}. \tag{4}$$

Note that with precision, observations where the algorithm predicted to be negative are ignored. This means that if the fraction of the positive class is small, then naively predicting all observations to be negative would yield poor precision. This is also the case for recall, where only the observations that truly belong to the positive class are considered, and only positive predictions can be correct for these observations. Finally, the F_β-score is used to combine precision and recall into a single metric based on the harmonic mean as follows:

$$F_\beta\text{-}score = (1+\beta^2) \cdot \frac{Precision \cdot Recall}{\beta^2 \cdot Precision + Recall}, \tag{5}$$

where β is a parameter used to balance the values of precision and recall. If $\beta = 1$, then precision and recall are equally weighted, and if $\beta < 1$ or $\beta > 1$ then precision is weighted more or less than recall, respectively. Together, the precision, recall, and F_β-score are the most commonly used metrics for binary classification problems that require exact class predictions. They are also used in Chapters 6 and 7 of this dissertation.

For evaluating the quality of probability predictions, two commonly used metrics are "logarithmic loss", also known as "logloss", and area under the Receiver Operating Characteristic curve, typically abbreviated to "AUC". Logloss is defined as follows:

$$Logloss := -\sum_{i=1}^{N} \left(y_i^{true} \cdot \ln(y_i^{pred}) + (1 - y_i^{true}) \cdot \ln(1 - y_i^{pred}) \right), \tag{6}$$

where i denotes an observation, y_i^{true} is the true binary label of observation i (1 or 0), and y_i^{pred} is the predicted probability for observation i. Intuitively, this error metric considers two cases for each observation: the true label is 1, and the true label is 0. When the true label is 1, then the logloss of that prediction is $\ln(y_i^{pred})$. Since a probability prediction is bounded within $[0,1]$, $\ln(y_i^{pred})$ is a negative value and its magnitude increases exponentially as y_i^{pred} decreases (i.e., moves away from 1). On the contrary, when the true label is 0, the magnitude of logloss increases exponentially as y_i^{pred} increases (i.e., moves away from 0). The negative sign is simply there for interpreting the problem as minimizing an error. Overall, logloss ensures that high-confidence correct predictions are rewarded more than low-confidence correct predictions and high-confidence incorrect predictions are penalized more than low-confidence incorrect predictions. Moreover, logloss assumes that it is better to have more cases of being wrong by a little than fewer cases of being wrong by a lot.

Unfortunately, much like accuracy, the value of logloss can also be affected by the distribution of positive and negative classes in the dataset. When the fraction of positive class observations is low, then naively predicting all observations to equal the

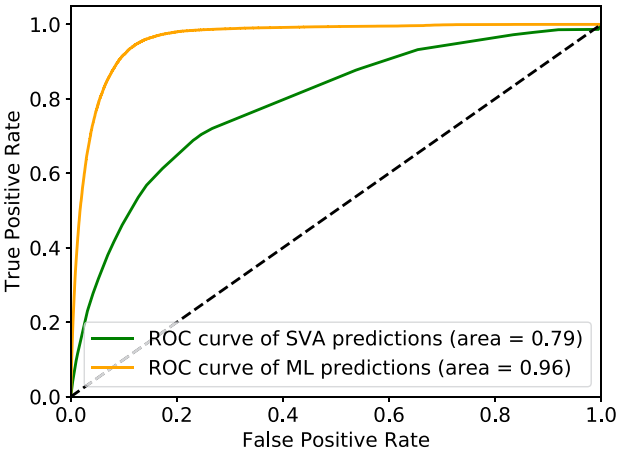


Figure 1: Sample ROC curves of predictions from different classification models.

empirical fraction of positive classes yields low (good) logloss values. While this does not limit the learning efficiency of the algorithm in the way that accuracy does, it makes performance comparisons across datasets very difficult. For easier performance comparisons across datasets, and even across problems, the AUC metric is much more useful. AUC measures how well a classification algorithm is able to differentiate between true positive observations from true negative observations. Specifically, it can be interpreted as: given a randomly selected positive observation and a randomly selected negative observation, what is the probability that the algorithm scores the positive observation higher than the negative observation? The mathematical definition of AUC is complicated and therefore in practice it is observed graphically from the Receiver Operating Characteristic (ROC) curve. Figure 1 presents an example of ROC curves from two classifiers on the same dataset. The ROC curve is a tradeoff between the true positive rate (y-axis) and the false positive rate (x-axis). The true positive rate (TPR) is the fraction of true positives predicted among all positive class observations, and the false positive rate (FPR) is the fraction of false positives predicted among all negative class observations. Since the original predictions are probabilities, a threshold is required to compute true positive and false positive predictions, but will be moved up and down between $[0, 1]$ to the range of TPRs and FPRs. When the threshold is high, all predictions are false and therefore TPR and FPR are both zero. As the threshold is decreased, both TPR and FPR increase, but at different rates. The ROC curve plots the relationship between TPR and FPR, and AUC is the area under this curve. If the model randomly guesses probabilities, then the ROC curve is expected to be linear (i.e., the TPR and FPR rates increase

at equal rates). In Figure 1, ROC curves of two algorithms are presented. The green curve has an area of 0.79, and with this prediction algorithm a TPR of 0.60 can be achieved with an FPR of only 0.20 (i.e., for every three correct positive predictions the algorithm makes one mistake). The orange curve is much better with an area of 0.96, and a 0.90 TPR can be reached with an FPR of 0.10 (i.e., for every ten correct positive predictions the algorithm makes one mistake).

Evaluation metrics for regression. Regression problems give supervised learning algorithms more freedom to make predictions, making evaluation more complicated. However, due to this additional complication, practitioners have actually simplified how regression problems are evaluated and mostly relied on metrics based on the numerical difference between the predicted and true label values. The most commonly used metric is "mean squared error" (MSE), defined by:

$$MSE := \frac{1}{N}\sum_{i=1}^{N}\left(y_i^{true} - y_i^{pred}\right)^2. \tag{7}$$

MSE compares the square of differences between the predicted and true values for each observation, and computes their average. One major advantage of MSE is that it is continuous and differentiable with respect to the predicted values, allowing for supervised learning algorithms to more easily optimize its parameters. Like the logloss metric for classification, MSE favors more smaller errors than fewer larger errors. However, due to the unbounded nature of regression problems, MSE is less robust to outliers as the extreme value can overly affect its predictions. When outliers are a concern, it is often better to use "mean absolute error" (MAE) as the error metric, defined by:

$$MAE := \frac{1}{N}\sum_{i=1}^{N}\left|y_i^{true} - y_i^{pred}\right|. \tag{8}$$

Instead of computing the square of differences between the predicted and true label values, MAE uses the absolute value. This weighs all distances equally, thereby allowing it to be more robust to extreme outliers than MSE. However, unlike the MSE, MAE is not a continuous function so it is more challenging to optimize supervised learning algorithms based on this metric. Both MSE and MAE assume that errors are equal regardless of the magnitude of the true label value. However, in some cases the same error magnitude on smaller values may be worse than for larger values. The metric "mean average percent error" (MAPE) considers the error as a percent of the true label value. For example, a prediction of 8 when the true label is 10

results in 20% error, and is equivalent to a prediction of 80 when the true label is 100. Mathematically, MAPE is defined by:

$$MAPE := \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i^{true} - y_i^{pred}}{y_i^{true}}\right| \cdot 100. \tag{9}$$

One shortcoming of MAPE is that it could lead to precision problems for observations where the true label value is small. If the magnitude of true label values in a dataset ranges from 0 to 100, the supervised learning algorithm may not be precise enough to predict with small error on observations where true value is small (e.g., less than 5). A simple method to deal with this problem is to instead use the average of y_i^{pred} and y_i^{true} in the denominator of Equation (9).

Finally, another representation of MSE is the coefficient of determination (R^2), defined by:

$$R^2 := 1 - \frac{\sum_{i=1}^{N}(y_i^{true} - y_i^{pred})^2}{\sum_{i=1}^{N}(y_i^{true} - \bar{y})^2}. \tag{10}$$

R^2 ranges from $[-1, 1]$ and is typically interpreted as how well an algorithm's predictions are related to the true label values, and the greater the magnitude the stronger the relationship (0 means no relationship and 1 means perfect relationship). A positive or negative R^2 value implies that the relationship is in the same or opposite direction.

Validation. The purpose of supervised learning is to make accurate predictions on new observations without having true label values. By definition, the performance of these predictions on unseen data cannot be evaluated. One option is to evaluate the algorithm's prediction performance on the training dataset, for which the true labels are already provided. However, this can lead to overfitting—the algorithm is over-optimized for the training dataset and fails to generalize to unseen observations.

Recall from Equation (1) that the supervised learning algorithm minimizes an error function with respect to parameters of a prediction function. The prediction function can be as simple as a flat line with only one parameter determining its position on the y-axis, or as complicated as a set of functions with as many parameters as there are observations[4]. The problem with evaluating an algorithm based on its prediction performance on the training set is that error will always decrease as the prediction function increases in complexity. In fact, the most powerful supervised learning algorithms such as gradient boosting and neural networks are complex

[4]It is also possible to have more parameters than the number of observations in the training set.

enough to perfectly predict all training observations if uncontrolled. In practice, the most commonly used method for evaluating algorithms is through a process called "validation". The validation process is intended to simulate real-life deployment of the learning algorithm and appropriately evaluate its performance. It works by splitting the original training set into two parts: a new training set, and a validation set (also called the holdout set or the out-of-sample set[5]). The new training set is used to train the supervised learning algorithm, and the validation set is used to evaluate predictions. Since the validation set was part of the original training set, true label values are provided to evaluate predictions. Moreover, as the validation set is not used to train the supervised learning algorithm, overfitting can be reflected by poor prediction performance.

Since validation is used to simulate real-life deployment, the design of the validation framework is important. For many problems, simply randomly splitting the training data into two equal halves is sufficient. However, for problems where time plays a role and past data is used to train algorithms to predict the future, the validation framework also needs to reflect this, and splitting the training data into halves by time is more appropriate. A cost of validation is that the amount of data that can be used to train the algorithm is significantly decreased. Typically, the more data is available to train the algorithm, the better it is able to generalize to unseen observations. By reducing the size of the training set, prediction error on the validation may be overestimated. To reduce the cost of validation, a splitting ratio that favors training can be used (e.g., 70% for training and 30% for validation). However, if the validation set is too small, then it may not be a good representation of the unseen observations it is intended to simulate. Ultimately, if computational resources are abundant, k-fold cross validation is a good solution. k-fold cross validation works by randomly splitting the original training data into k parts. Then for each part, the other $k-1$ parts are used to train the algorithm and evaluated on that part. Finally, the prediction error is averaged over each validation part. This way, all of the data is used for both training and validation, but without overlap.

2.2.2 Feature engineering

Generally, for all predictive analytics problems, the raw data provided is not directly suitable for training machine learning algorithms. This is because the structure of the raw data is usually not directly translatable to the N-by-M matrix format of one row per observation with M features. For example, in the debt repayment prediction

[5]Often the validation set is also called "test set", but the test set sometimes also has a different meaning.

problem considered in Chapter 3, the raw data contains many rows per debtor, where each row is an action taken by a debt collector on a debtor (e.g., phone call). However, the goal is to make a prediction for each debtor (at a given point in time) and not each action, and the features associated with each debtor should describe the debtor's state in the collection process. To construct an N-by-M dataset that is suitable for the prediction problem, the raw data must be processed in a way that captures the target appropriately, with features containing sufficient information to predict the target value. This process is called "feature engineering". Feature engineering is a creative process that combines both intuition and science, and one of the greatest advantages of having a reliable validation framework is to test the effectiveness of various feature designs and combinations.

Timestamped events. In this dissertation, data in all chapters except one comes in the format of sequences of timestamped events (e.g., collection actions and customer website visits). To engineer a sequence of events into a single row of information, a simple approach is to convert one type of event into one feature, and to record the number of occurrences of that event. For example, when dealing with website visits that came through a set of possible marketing channels, a feature for each channel can be constructed to hold the number of times such a visit occurred for the customer. With this feature, if the visitors through particular channels are more likely to purchase a product, then the supervised learning algorithm could easily capture this relationship.

Since events are usually timestamped, a substantial amount of information can be extracted from when events occur. If predictions are also time-dependent, then the amount of time since specific events can be important. For example, customers may be more likely to purchase a product if their previous visit to the website was recent versus further in the past. In addition, it is even possible to look back at multiple past events and compute the density of their occurrences. Timestamped events also allow the algorithm to capture seasonality effects, such as month of year, day of week, or hour of day. For example, it has been hypothesized that online customers browse during the day while at work, and only make purchases from home in the evening. By including the hour of the most recent visit as a feature, the algorithm might be better able to predict purchase behavior. Alternatively, if the ratio of visits during specific times matter (i.e., many weekend visits may be indicative of bargain hunters), then these specific times can be constructed as features and the number or ratio of occurrences for each customer would be their values.

Mathematical feature transformations. Sometimes, mathematical transformations of features or even the target variable can significantly improve the learning efficiency of algorithms. Most supervised learning algorithms assume a particular relationship between features and the target value. A classic example is the linear relationship, defined by:

$$y_i^{pred} = \beta \cdot x_i + \alpha, \tag{11}$$

where y_i^{pred} is the predicted value and x_i is a feature value. In this mathematical relationship, it is assumed that a unit increase in x_i will always result in β units of increase in y_i^{pred}. However, in practice a unit increase in x at low levels may affect the target value more strongly than when x is already at high levels. This distorts the assumed linear relationship between the feature and the target, resulting in a poorly learned β and inferior prediction outcomes. One solution is to change the algorithm—use a nonlinear relationship instead. However, a simpler approach to improve the results in this case without changing the algorithm is to log-transform the feature, i.e., using $\ln(x_i)$ instead of x_i. Often, log-transformed count features better fit linear relationships than raw counts. Alternative transformations such as taking the square root of x may also be effective. For some algorithms, it can be useful to perform normalization—subtracting by the feature's mean across all observations and then dividing by the standard deviation—on all features. This is because some algorithms are sensitive to the magnitude of feature values, so normalizing all features to a single scale can help.

Relationships across features may also be important. In statistics, this is referred to as "feature interactions", which mostly focuses on the product of features. For supervised learning purposes, other relationships such as the sum and ratio of features can also be considered. For example, website visits across the six hourly periods between 6pm and midnight can be summed into a single feature for visits during the evening. This allows for individual hourly visits to vary but not affect the prediction as long as the evening visits remain constant.

Categorical features. All machine learning algorithms require data to be in numeric form. Therefore, the natural method of encoding categorical features is to assign a unique numeric value to each category. However, assigning numeric values to categories already assumes a numerical relationship between them, which cannot be generalized to all categorical features. A commonly used method to handle categorical features is "one-hot encoding", also known as "dummy encoding". This method transforms a categorical feature into a set of binary features, one for each

category. Then, for each observation, the new feature that corresponds to the category of the original feature has a value of 1, whereas the features that correspond to the other categories all have values of 0. This allows the supervised learning algorithm to learn the relationship between each category and the target value. One problem with one-hot encoding is that a large number of binary features are created from a single feature with many categories. This can potentially result in overfitting, and at the very least introduces complexity to the management of the features. Unfortunately, there are no other feature engineering methods that can more simply represent categorical variables without requiring additional assumptions.

Pairwise comparisons. Many prediction problems rely on the comparison of features across two entities. Some examples include document search (comparing the search query and potential search result), product recommendation (comparing information about a product and user), and device matching (presented in Chapters 6 and 7). For these problems, including features of the two entities independently is a good start, but additional value can be derived from pairwise comparisons. Distance-based metrics across features between the pairs, such as Euclidean or Manhattan distance (for numeric features), cosine and Jaccard similarity (for text-based features), and simple binary differences (e.g., whether two users visited the same website within an hour) can be effective. Finally, for problems that rely on text or large numbers of event occurrences of different types, "feature embeddings" have become powerful tools. Feature embedding is a way of summarizing a large set of features into a much smaller set of features based on similarity or co-occurrence across all observations (e.g., the word2vec embedding). By replacing raw features with values from an embedding, the pairwise distance-based metrics tend to convey more information.

2.2.3 Supervised learning algorithms

Throughout the years many supervised learning algorithms have been developed for prediction problems. However, it should be noted that recency and popularity are not reflective of an algorithm's performance. Due to the exponential increase in data availability and computing power, a number of advanced algorithms (e.g., neural networks, gradient boosting) have become much more effective than they initially were when first developed, and have dominated the space. For this dissertation, gradient boosting is the only algorithm used as it is most suitable for the problems considered. Nevertheless, this section aims to provide the reader with a general understanding and background for some well-known supervised learning algorithms.

Linear models. Historically, it is often accepted that the first supervised learning algorithm was the "perceptron" (Rosenblatt 1958), which is the foundation for modern-day neural networks. The perceptron is a simple linear threshold function for classifying observations, defined by:

$$y_i^{pred} = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x}_i + b > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

where $\mathbf{x}_i$ is a vector of feature values for observation i, $\mathbf{w}$ is a vector of weights, and b is a bias term. For a given training set, there is an algorithmic procedure to determine $\mathbf{w}$ and b that minimizes classification error.

However, even before the existence of computers, "linear regression", the most well-known supervised learning algorithm, was developed[6]. According to Yan and Su (2009), linear regression can be dated back to Legendre in 1805 and Gauss in 1809. Linear regression is very similar to the perceptron and is defined by:

$$y_i^{pred} = \mathbf{w} \cdot \mathbf{x}_i + b. \tag{13}$$

Here, the process of finding $\mathbf{w}$ and b focuses on minimizing the mean squared error as defined in Equation (7), and it can be solved analytically via ordinary least squares or maximum likelihood, or numerically via gradient descent. The weights $\mathbf{w}$ can be interpreted as the magnitude of effect that one unit increase in feature x has on the target value of y. Simple nonlinear transformations are possible by taking powers or logarithms of various features. For classification problems, "logistic regression"—which transforms predictions from linear regression via the sigmoid function—can be used. A probability prediction is made, and the procedure to find $\mathbf{w}$ and b minimizes the logloss function defined in Equation (6). Both linear and logistic regression are fast, flexible, and generally highly effective. For this reason, it is often wise to start with one of these two approaches for any supervised learning problem.

One major problem with linear and logistic regression is the potential for overfitting. Unlike some of the more modern supervised learning algorithms, the exact nature of how linear and logistic regression find $\mathbf{w}$ and b often leads to the detection of spurious relationships in the training data between some features and the target, resulting in overfitting. Moreover, if the number of features in a dataset exceeds the number of observations (common in images and text), linear and logistic regression are guaranteed to perfectly fit the training data, which will almost always result in

[6]Linear regression is better known in statistics and econometrics, but is also regarded by the machine learning community as a supervised learning algorithm.

poor predictions on unseen observations. To combat this problem, a general framework called "regularization" was developed to artificially reduce the magnitude of $\mathbf{w}$ by including a penalty term $\lambda\|\mathbf{w}\|_2$ to the error function. Because the linear and logistic regression algorithms are minimizing an error function, inclusion of the penalty term penalizes larger values of $\mathbf{w}$, thereby preventing the exact minimization of the error function on the training data. When the $L2$-norm is used for the penalty term the resulting algorithm is called "ridge regression" (Hoerl and Kennard 1970), and when the $L1$-norm is used the resulting algorithm is called "LASSO" (Tibshirani 1996). It is also possible to use both penalty terms together, resulting in an algorithm called "elastic net" (Zou and Hastie 2005). λ is a hyperparameter that controls the strength of regularization, and needs to be tuned manually via trial and error during validation. Although there are no theoretical arguments for why regularization is effective, it has been practically useful when a dataset contains large numbers of features.

Nonlinear models. Although supervised learning algorithms based on linear models are generally effective, and their parameters are easy to estimate, nonlinear models provide greater flexibility in capturing the real-life relationships between features and the target. The simplest nonlinear supervised learning algorithm is known as "k-nearest neighbors" (Cover and Hart 1967), or kNN for short. The kNN algorithm predicts observations to equal the target label of its k nearest labeled observations based on feature values. For regression or probability prediction problems, the predicted label is an average of the nearest labels, and for binary classification the prediction is by majority voting. How near observations are to each other is determined by a distance metric, and common measures are: Euclidean distance, Manhattan distance, and cosine similarity. The k in kNN is a hyperparameter, meaning that is should be selected via validation. It generally trades off how local predictions should be, with $k = 1$ suggesting that the closest neighbor should be solely responsible for the predicted label of an observation, and more neighbors are included as k increases. kNN has been shown to work well in some problems, but one major disadvantage is that the distance metrics assume equal importance of the features. In reality, some features are far more predictive of the target than other features, and therefore they should be weighted differently. Unfortunately, kNN does not provide such flexibility.

Since the invention of the perceptron, AI researchers realized that multiple layers of perceptrons where one fed into the next creates a nonlinear algorithm that is able to classify more challenging classification problems. When many layers of perceptrons are connected together, this concept was given the name of "artificial neural networks", or just "neural networks". However, it was a challenge to efficiently learn

the weights of neural networks. Rumelhart et al. (1988) popularized the backpropagation algorithm, which combined with gradient descent allowed the training of neural networks to be more efficient, starting a decade-long interest in this algorithm by AI researchers. Eventually, due to insufficient available data and computing power, neural networks proved to be difficult to use successfully in practice and other algorithms gained traction in the 2000s.

Using a different approach from neural networks, "support vector machines" (Cortes and Vapnik 1995), also known as SVM, became the most popular supervised learning algorithm in the late 90s to the early 00s. Behind SVM is the notion that if there is a binary-class dataset that is linearly separable (i.e., can be perfectly classified by a hyperplane), then it is possible to find a hyperplane that maximizes the distance between it and the closest observation of each class from it. The key to this algorithm is that even when the data is not linearly separable based on the dimensions specified by the feature space (e.g., two-dimensional space when there are two features), it is possible to map the feature space into higher dimensions through the use of kernels[7]. Some well-known kernels include the polynomial kernel, radial basis function kernel, and string kernel. The advantage of kernels is flexibility (able to model many complex relationships between features and the target) and computational efficiency (orders of magnitude faster than explicitly transforming the features).

In the late 2000s and early 2010s, thanks to the massive increase of data availability and the power of graphical processing units, neural networks made a comeback and have again become the most popular supervised learning algorithm. This coincided with the success of two techniques, "convolutional neural networks" (CNN) and "recurrent neural networks" (RNN) that significantly defeated the existing state-of-the-art approaches for the problems of image recognition (Krizhevsky et al. 2012) and speech recognition (Graves et al. 2013). These newly popularized approaches use many more layers than the multi-layer perceptrons studied in the 80s, and as a result have been re-branded as "deep neural networks", or more generally known as "deep learning". As of right now, deep learning approaches are state-of-the-art for some well-known problems in AI, particularly relating to images, text, and speech, and is close to state-of-the-art for the supervised learning problems discussed in this dissertation. However, the performance of neural networks heavily depend on its many hyperparameters (e.g., number of layers, size of layers, types of activation functions), and significant amounts of trial and error is required to find good hyperparameters for any problem. For this reason it is still easier to rely on simpler or more stable algorithms for practical use.

[7]It is also possible to use kernels for other algorithms such as the perceptron or linear regression, but it is typically used in combination with SVM.

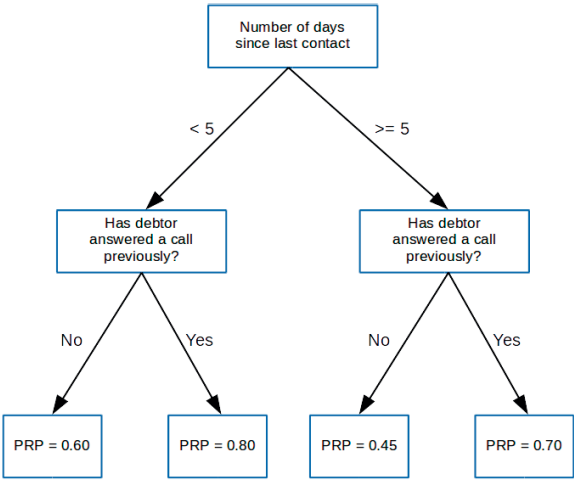


Figure 2: Example of a simple decision tree.

Tree-based models. Based on human intuition, decision trees represent a set of nested binary rules used for making predictions. Decision trees for supervised learning were first popularized by Breiman et al. (1984), under the name "classification and regression trees" (CART). Figure 2 is an example of a decision tree for the prediction of repayment probabilities of debtors. In this decision tree, two features are considered: the number of days since last contact and whether the debtor has answered a call previously. For any observation, its predicted probability can be determined by simply traversing the tree from top to bottom. In the example, it can be seen that if a debtor was last contacted 3 days ago and he did not answer a call previously, his predicted repayment probability is 0.6. Alternatively, if the debtor did answer a call previously, his predicted repayment probability would be 0.8. One of the greatest advantages of decision trees is its flexibility to handle both numerical and categorical data without requiring feature engineering. Moreover, the nested structure of trees can capture interaction effects if present. Although decision trees are intuitive and flexible, the process of generating a tree that minimizes any error function for a given dataset is intractable. Even when a tree has depth of one (i.e., a single layer), all values of all features must be considered to determine the optimal split. Due to the intractability, numerous heuristics have been developed over time for practical use. Nevertheless, the predictive performance of a single classification or regression tree is inconsistent and often quite poor.

The performance of decision trees was significantly improved through a process called "bootstrapped aggregating", also known as "bagging" (Breiman 1996). The

idea of bagging is to train multiple instances of the same supervised learning algorithm on randomly sampled subsets of the training data, and then aggregate their predictions into a single value by averaging. This method works particularly well when the base supervised learning algorithm is decision trees because the intractable training process of decision trees results in large variations among trained trees when small variations are made to the training data. Bagging helps to stabilize the predictions, and surprisingly, improved the overall performance through the wisdom of crowds. Subsequently, Breiman (2001) took bagging one step further and introduced "random forests", which not only subsamples subsets of the training data, but also subsets of the features. This further increased the variation among the learned decision trees, and led to additional performance improvement. For much of the 2000s random forests is known to be one of the best supervised learning algorithms due to ease of use and strong performance.

At the same time of random forests, Friedman (2001) developed the "gradient boosting machine", which combines decision trees with a popular technique known as "gradient boosting". Unlike bagging, gradient boosting follows the idea of iteratively training instances of the same supervised learning algorithm by learning from the errors of the previous iteration. Finally, the instances are aggregated to produce a single prediction. Much like bagging, decision trees are again well-suited for gradient boosting and it is also possible to subsample subsets of features for training each instance of the tree. The resulting algorithm is known as "gradient boosted decision trees" (GBDT). Thanks to fast implementations of GBDT in recent times, it has become the dominant algorithm for most predictive analytics problems (Chen and Guestrin 2016). It is also the algorithm of choice for all of the problems studied in this dissertation. However, GBDT has received only a fraction of the popularity of deep learning algorithms and is likely to be surpassed in the not-too-distant future.

Much of this chapter is only meant to provide readers with a basic understanding of machine learning, supervised learning, and predictive analytics. For more comprehensive and detailed information, a number of excellent books on machine learning have been published: Friedman et al. (2001), Bishop (2006), Barber (2012), Murphy (2012), Goodfellow et al. (2016). These books focus on different aspects of machine learning, but all from a theoretical perspective. Unfortunately, there are no comprehensive documents on the practical aspects of machine learning.

Chapter 3

Data-driven Consumer Debt Collection via Machine Learning and Approximate Dynamic Programming

Chapter Abstract

This chapter develops and tests a framework for the data-driven scheduling of outbound calls made by debt collectors. These phone calls are used to persuade debtors to settle their debt, or to negotiate payment arrangements in case debtors are willing, but unable to repay. We determine on a daily basis which debtors should be called to maximize the amount of delinquent debt recovered in the long term, under the constraint that only a limited number of phone calls can be made each day. Our approach is to formulate a Markov decision process and, given its intractability, approximate the value function based on historical data through the use of state-of-the-art machine learning techniques. Specifically, we predict the likelihood with which a debtor in a particular state is going to settle its debt and use this as a proxy for the value function. Based on this value function approximation, we compute for each debtor the *marginal value* of making a call. This leads to a particularly straightforward optimization procedure, namely, we prioritize the debtors that have the highest marginal value per phone call. We validate our proposed methodology in a controlled field experiment conducted with real debtors. The results show that our optimized policy substantially outperforms the current scheduling policy that has been used in business practice for many years. Most importantly, our policy collects more debt in less time, whilst using substantially fewer resources—leading to a large increase in the amount of debt collected per phone call.

3.1 Introduction

3.1.1 Background and motivation

In the U.S., $605 billion of household debt was delinquent as of March 31, 2018 (Federal Reserve Bank of New York 2018). Of this amount, over $400 billion was

delinquent for more than 90 days. For companies that rely on installment payments, this means it is of great importance to manage the collection of installments efficiently and collect as many payments as possible to assure business continuity and drive profitability. The potential, but also the complexity of doing so, was already recognized half a century ago by Cole (1968):

> "Collection work would be easier and the results better if there were some magic way in which each account could be immediately and accurately classified as to the reason for nonpayment and the collection method which would be most effective with that particular debtor. Sorting devices to perform such miracles unfortunately are not yet available, and until such become economically and mechanically feasible the responsibility for any classification, if made at all, rests with the credit personnel involved." (pp. 314-315).

With the increased availability of data and the development of sophisticated machine learning techniques, such "sorting devices" have now become reality.

In this paper, we present a framework for the data-driven scheduling of outbound phone calls made by debt collectors. That is, we determine on a daily basis which debtors a debt collector should call to maximize the amount of delinquent debt recovered in the long term, under the constraint that only a limited number of phone calls can be made each day. These phone calls are used to persuade debtors to settle their debt, or to negotiate payment arrangements (e.g., a payment plan) in case debtors are willing, but unable to repay their debt. Scheduling these calls is challenging as it is difficult to assess the value of making a phone call to a debtor. This is because a priori the outcome of making a call is uncertain, and the extent to which a call attributes to a repayment is non-trivial. In general, the effect of phone calls on the repayment behavior of debtors depends on numerous interacting features, such as the time since the previous phone call, whether the debtor answered the call before, the amount of debt owed, the time of the month, and the persuasiveness of the agent who is calling. It is unclear what the effect of these (interacting) features is on the outcome of phone calls and, hence, on the effectiveness of a schedule of phone calls. This lack of structure and understanding drives our belief that a flexible non-parametric machine learning method would be most appropriate to leverage data for optimizing actions.

To this end, we show that the problem of scheduling phone calls is naturally formulated as a Markov decision process (MDP), but that a prohibitively large state space is required to capture the dynamics of the collection process appropriately. To alleviate this, we show how state-of-the-art machine learning methods can be used in an approximate dynamic programming (ADP) framework that is interpretable,

highly scalable, and data-driven. We validate our proposed approach by means of a controlled field experiment with real debtors in a real business setting.

This research is carried out in collaboration with an anonymous debt collection agency from the Netherlands, to which we refer to as the Collector. The Collector provided the data required to estimate our models and implemented our methodology to conduct the controlled field experiment. The Collector handles about 250,000 collection cases each year, with a principal (monetary value) of approximately €120 million. Currently, the Collector schedules phone calls according to a static policy in which calls are scheduled based on a one-size-fits-all policy. Given that the Collector has carefully tracked all of its historical efforts and outcomes, we can leverage this data for the purpose of optimizing its collection process.

3.1.2 Main contributions

To the best of our knowledge, the current paper is the first to incorporate modern machine learning methods into an ADP framework that is validated through a controlled field experiment in a real-life business setting. We take the problem of dynamically scheduling outbound calls for a debt collector—as naturally described by an MDP—and approximate state values using supervised machine learning. More precisely, we construct a binary classification problem to predict—based on a debtor's state—the likelihood with which a debtor is going to repay its debt. The debtor's state space is high dimensional and incorporates all static and dynamic information that characterizes a debtor at a given point in time. For the purpose of value function approximation, we multiply the likelihood with which a debtor settles its debt by the size of the debt—thereby obtaining an approximation for the expected value of a debtor given its current state. In doing so, we overcome the curse of dimensionality inherent to this problem by inferring the value of a debtor's state based on historical data in a highly scalable and flexible manner.

Based on our value function approximation, we compute for each debtor the *marginal value* of a phone call, which is defined as the change in the value function if we spend another phone call on this debtor. This leads to a particularly straightforward optimization procedure, namely, we prioritize the debtors that have the highest marginal value per phone call. The result is a policy that is interpretable (debtors with the highest marginal value on the effort are prioritized), highly scalable, and data-driven. In addition, the optimization procedure allows for straightforward implementation in business practice: arrivals of new debtors are naturally incorporated and an appropriate number of phone calls can be determined to be made on a given day, depending on the debt collector's capacity.

We validate our proposed methodology in a controlled field experiment conducted with real debtors. The results show that our optimized policy substantially outperforms the current scheduling policy that has been used in business practice for many years. Most importantly, our policy collects more debt in less time, whilst using substantially fewer resources—leading to a 47.2% increase in the amount of debt collected per phone call. We also identify a key managerial insight, namely that capacity is best spent on debtors that are more difficult to collect from. These are debtors that are in the collection process for a longer period of time, are less likely to pick up the phone, and have not partially repaid or promised to do so. These insights help managers better understand the dynamics of the debt collection process.

In summary, this paper contributes to the existing literature on business analytics, data-driven optimization, and that of ADPs in the following ways: i) we add to the debt collection optimization literature by presenting a novel, scalable, and flexible framework for daily data-driven scheduling of outbound calls; ii) we incorporate state-of-the-art machine learning methods to the ADP framework, which takes advantage of higher-order feature interactions and results in superior out-of-sample model fit for value function approximation compared to benchmark models; iii) we open the proverbial machine learning black box and identify generalizable insights for the improved scheduling of outbound debt collection phone calls; and iv) we validate our methodology by means of a controlled field experiment with real debtors.

The following section contains a review of existing literature on debt collection and debt collection optimization in particular. We discuss relevant literature that concerns approximate dynamic programming in Section 3.4, together with our proposed approximation method.

3.1.3 Relation to literature

Debt collection optimization. More than half a century ago, Mitchner and Peterson (1957) considered the problem of optimizing the collection of delinquent debt at Bank of America for various types of loans, such as car loans, personal loans, and real estate loans. They formulated the problem of collecting debt as an optimal stopping problem, in which the duration with which the collector should pursue the debtor was optimized, taking into account the cost of doing so. Their results show a potential increase in net profit of 33%.

Fifteen years later, Liebman (1972) developed a simple Markov decision process for optimizing credit control policies. They solve an example problem with four delinquency states, two amount owed states, two recent experience states, and three action strategies. However, the curse of dimensionality quickly becomes a significant

challenge and no further progress on this topic was made until more recently by Abe et al. (2010), De Almeida Filho et al. (2010), and Miller et al. (2012).

In Abe et al. (2010) and the accompanying paper Miller et al. (2012), a framework for debt collection optimization is presented that, of the existing work, is closest to the approach considered in this paper. In Abe et al. (2010) the collection process is modeled as a constrained MDP, which explicitly takes business, legal, and resource constraints into account. Subsequently, given the intractability of the MDP, a constrained Q-learning algorithm is proposed by means of which a policy can be obtained. In Miller et al. (2012) the deployment of this methodology at the New York State Department of Taxation and Finance is described for which an increase in collected delinquent debt by 8 percent is reported over the first year, where an increase of 2-4 percent would otherwise have been projected.

Also from the operations domain, De Almeida Filho et al. (2010) present a study on the optimization of debt collection in the context of consumer lending. In their work, a dynamic programming approach is presented in which the monthly decision epochs pertain to deciding which action to take in the month to come. The value function corresponds to the future net discounted recovery rate and the transitions are assumed to be deterministic. Since the model assumes homogeneous debtors, the approach is especially useful to predict collection performance and resource requirements for aggregated portfolios of debtors for which it is reasonable to assume homogeneity. The authors refer to the importance and potential of tailoring the collection process to the individual debtor, but note that the data required for this purpose are hardly ever available in practice.

Credit scoring and valuation. In the field of finance, much research has been done on the credit-granting decision, i.e., whether to grant a loan to a potential new customer. Typically, the credit-granting decision for personal loans is made by means of credit scoring, which is a standardized method of assigning a score to potential customers that represent their creditworthiness—see Crook et al. (2007) for a literature review and Lessmann et al. (2015) for a benchmarking study on existing scoring models.

On the other hand, the valuation of existing credit—and existing unsecured credit in particular—is more closely related to our work since the debt that the Collector is trying to collect is essentially outstanding unsecured credit. Although much work has been done on the valuation of corporate credit and secured customer credit, the literature on unsecured consumer credit is sparse. The work of Chehrazi and Weber (2015) on dynamic valuation of delinquent credit card accounts models stochastic repayment behavior of individual debtors over time. They derive a self-exciting point

process for repayment behavior and estimate the parameters of the process using the generalized method of moments. This model is then used to construct a dynamic collectability score to estimate the probability of collecting from a debt account, thus allowing for the valuation of credit card debt. In a subsequent paper, Chehrazi et al. (2018) formulates a stochastic optimal control problem from the self-exciting point process established in Chehrazi and Weber (2015) and derives a semi-analytic solution. However, this solution was not analyzed empirically nor was it experimentally validated.

3.1.4 Outline

In Section 3.2, we provide an overview of how debt collection works and what the optimization problem is that debt collectors face. In Section 3.3 and Section 3.4, we formulate an MDP and provide an approximation for this MDP, respectively. Section 3.5 contains a description of the data used for model development and validation. In Section 3.6, we present the results of model estimation and validation. Section 3.7 presents the design and results of our controlled field experiment. We conclude in Section 3.8.

3.2 Problem description

We first provide a high-level overview of the operations of a debt collector. Thereafter, we provide more details on the actual collection process. This is all based on the experiences of our industry partner (the Collector), but is illustrative for the debt collection industry in general.

3.2.1 High-level overview

In practice, a client that has overdue debt with a company is placed "in collections", which means that the debtor is transferred to either a specialized debt collection department within the company or to an external debt collecting agency that works on behalf of the company. In this work, we refer to both as a debt collector, i.e., a debt collector can be either the debt owner itself or a third-party debt collection agency working on behalf of the debt owner. The debtor typically incurs a collection fee that is added to the original debt to cover the additional costs of recovering the debt, and is regulated in many countries. In the problem that we consider, the collection fee is independent of the amount of debt owed and constant across debtors.

Once placed in collections, the debt collector pursues the debtor to settle the debt plus the collection fee by sending out letters and e-mails, and through phone

calls made by its agents. Amongst the Collector's clients are utility providers, credit facilitators, and health care providers, which operate in the business-to-consumer market.

The process of the Collector comprises two phases. Upon arrival, a debtor first enters the collection phase, in which the Collector pursues the debtor to repay the debt plus the collection fee through letters, e-mails, and phone calls. During this phase, the collector acts cooperatively towards the debtor, and can offer payment plans if debtors are willing, but not able to pay on a short-term notice. As such, this phase can take from a few days (in case the debtor pays immediately) up to a few months (in case the debtor does not pay at all or gets involved in a payment plan).

When the Collector is unsuccessful in recovering the debt during the collection phase, it chooses to either write off the debt or invoke a legal procedure. The former happens when, for example, the debtor is deceased or has declared bankruptcy. The latter means that a bailiff is invoked, who will send out a subpoena and ultimately can confiscate property if necessary. Whether a debtor is escalated to the legal phase or written off is determined case by case and depends, amongst other things, on the amount of outstanding debt and the likelihood of recovering the debt through legal procedures. Since this phase requires legal assessment by an expert, it is very expensive and the outcome is highly uncertain. Hence, recovering debt before the legal phase is deemed beneficial for both collector and debtor. As such, the legal phase is excluded from the optimization procedures proposed in this work and our objective is to maximize recovered debt during the collection phase, which is described in greater detail in the following section.

3.2.2 Collection phase

The collection phase is characterized by four sequential letters (sent via both post and e-mail simultaneously), where each letter has a seven-day payment notice and communicates with increasing urgency the necessity to repay the debt. The letters are sent between seven to ten days of each other. The fourth and final letter communicates the severe (financial) consequences of the legal procedure that is possibly invoked if the debtor does not settle.

In between the letters (or after the final letter), the Collector is free to call debtors at its discretion. This is considered a vital tool during the collection process, as the phone calls allow the agents to inform the debtor about the situation along with the consequences of non-payment, and to make an assessment of whether the debtor is willing and/or able to pay. Figure 1 provides a schematic illustration of the collection process.

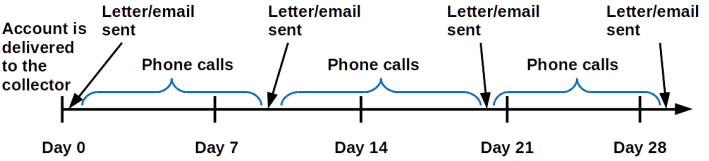


Figure 1: The standard operating timeline of a debt collection agency.

The optimization problem that the Collector encounters, is deciding each day which debtors should be called to maximize recovered debt, given the finite and inflexible capacity of its workforce. In practice, this implies that the Collector has to decide on a prioritization on the debtor portfolio that indicates which debtors should be called first. Currently, the Collector's policy is to schedule a phone call each time a debtor has received a new letter. In addition, if a debtor agreed on a payment plan and failed to comply with its conditions, a call is scheduled as well. In case capacity is insufficient, the Collector's managerial staff makes an assessment of which debtors should be called first. Given the labor-intensive nature of the phone calls, the gains from optimizing the prioritization of calls are potentially substantial.

3.3 Model description

The problem of optimizing debt collection efforts over time in the current context is formulated as an MDP with an infinite time horizon and decision epochs in discrete time. This suits the approach of the Collector, since in principle the Collector operates indefinitely and decisions are made at discrete points in time (i.e., daily). To formalize the MDP, we assume that at any given point in time, the Collector has at most $N \in \mathbb{N}$ debtors in its portfolio. Practically, this means we set N arbitrarily large such that the Collector never has more than N debtors in its portfolio.

3.3.1 State space

We denote the state space of each of the (at most) N debtors by $\mathcal{X}$ and the state space of the portfolio of debtors by $\bar{\mathcal{X}} := \mathcal{X}^N$ (i.e., the N-fold Cartesian product of $\mathcal{X}$). In our formulation, each of the N parts of the state space is utilized by different debtors over time—each part $\mathcal{X}$ of the state space $\bar{\mathcal{X}}$ functions as a slot for storing the information of one of the debtors. A slot becomes available for new arriving debtors once efforts on the existing debtor are terminated because the debt is recovered or written-off.

The state space below is chosen to accommodate for the data to which we apply our methodology (as described in Section 3.5). We divide the debtor state space

$\mathcal{X}$ into debtor-specific features, historical-interaction features, and seasonalities as follows. The superscripts B, I, N, and C indicate whether it is a binary, integer, numerical, or categorical variable, respectively.

Debtor specific: 1) initial debt amount^{N}, 2) customer tenure^{I}, 3) has partially repaid debt^{B}, 4) repayment plan in place^{B}, 5) phone number is available^{B}, 6) e-mail address is available^{B}, 7) product type^{C}, 8) amount repaid already^{N}, 9) Collector collected from debtor before^{B}, 10) average income in the postal code area of the debtor^{N}, 11) share of people under 30 in postal code area of the debtor^{N}, 12) current substatus^{C}, 13) passed final letter^{B}.

Here, 2) indicates when the debtor became a customer of the debt owner: the exact time was not provided, instead we have an integer that represents the inverse order in which the debtor became a customer relative to all customers of the debt owner (a larger value means the debtor was a customer of the debt owner for a longer period of time); 5) pertains to whether the debt owner provided the Collector with a phone number of the debtor. If this is not the case, the Collector may still be able to call the debtor by searching manually in publicly available resources for potential phone numbers that match to the name and address of the debtor; 7) refers to the product or service that the debtor purchased and led to the debt; 12) refers to a debtor's state description used internally by the Collector to characterize a debtor at a given point in time; 13) refers to whether the debtors have received the final (i.e., fourth) letter.

Historical interaction: 1) has answered a phone call^{B}, 2) promised to repay^{B}, 3) number of previous collector-debtor interactions^{I}, 4) number of previous phone calls^{I}, 5) days since promise to repay^{I}, 6) days since last collector-debtor interaction^{I}, 7) days since last phone call^{I}, 8) days since last answered phone call^{I}, 9) days since last incoming contact^{I}, 10) days since last incoming e-mail^{I}, 11) days since last incoming phone call^{I}.

Here, 2) and 5) refer to the event in which the debtor has (verbally) promised the Collector to settle the debt; in 3) and 6) the word 'interaction' includes both collector- and debtor-initiated communication efforts, and also phone calls that did not get through count as an interaction—thereby using it in a broader sense than usual.

Seasonality: day of week^{C}, week of month^{C}.

For features where missing values are possible, a unique integer is used as replacement for missing values. An example of this is the feature *days since last phone call* for cases where no phone calls have previously been made to the debtor.

3.3.2 Action space and value function

Regarding the action space, for a given day let $a \in \{0,1\}^N$ describe which debtors will be called: $a_i = 1$ for $i \in \{1,2,\ldots,N\}$ indicates that a call is made to debtor i, and $a_i = 0$ means no call is made on a given day. In some cases, it is undesirable to call a debtor (e.g., when the debt is currently being further investigated because the debtor disputed the debt). Therefore, we construct the action space as follows. Let $i \in \{1,\ldots,N\}$, $x = (x_1,\ldots,x_N) \in \bar{\mathcal{X}}$ with $x_i \in \mathcal{X}$, and let $\mathcal{A}'(x_i)$ denote the action space pertaining to debtor i, so that $\mathcal{A}'(x_i)$ equals $\{0\}$ if no call is allowed and $\{0,1\}$ when a call to debtor i is allowed. In addition, let $c_t \in \mathbb{N}$ denote the (deterministic) capacity on day t, i.e., the maximum number of phone calls that can be made on day t, where t counts the number of days since the collection process was initiated. Accordingly, we define

$$\mathcal{A}_t(x) := \left\{ (a_1,\ldots,a_N) : a_i \in \mathcal{A}'(x_i),\ i = 1,\ldots,N,\ \sum_{i=1}^{N} a_i \leq c_t \right\}$$

as the action space on day t. In case slot i of the state space is not used, $\mathcal{A}'(x_i) = \{0\}$ for all $x_i \in \mathcal{X}$.

Furthermore, on day t, for $x, y \in \bar{\mathcal{X}}$ and $a \in \mathcal{A}_t(x)$, let $p(x,a,y)$ denote the probability of moving from state x on day t to state y on day $t+1$, when choosing action a and let $r(x,a,y)$ denote the amount of debt recovered (i.e., repaid and received) when moving from state x to state y choosing action a. The possible arrival of new debtors is implicitly incorporated in $p(x,a,y)$. Then, the optimality equation becomes

$$V_t(x) = \max_{a \in \mathcal{A}_t(x)} \sum_{y \in \bar{\mathcal{X}}} p(x,a,y)\left(r(x,a,y) + \gamma V_{t+1}(y)\right), \tag{1}$$

for $t = 0,1,2,\ldots$, where $V_t(x)$ denotes the total expected discounted reward when being in state $x \in \bar{\mathcal{X}}$ at day t, and $\gamma \in (0,1)$ denotes an appropriate discount rate. The function $V_t : \bar{\mathcal{X}} \to \mathbb{R}$ is often referred to as the *value function*.

Since the state space $\bar{\mathcal{X}}$ consists of all debtor information, the formulated MDP has a high-dimensional state space. Moreover, parts of the state space are unbounded (e.g., the number of collector-debtor interactions). This makes it intractable to solve the MDP even numerically. The MDP, however, has structural properties that facilitate the computation of near-optimal policies. First, the debtors in the portfolio behave independently of each other, i.e., changes to the state and repayment probability of one debtor do not affect the repayment probability of the other debtors. Second, the dependence in the problem formulation is only due to the capacity constraint c_t on day t. Hence, a natural approximation that breaks the dependence arises

when the Collector solves a stochastic knapsack problem based on the state of the debtors in the portfolio on that day. The knapsack has size c_t on day t, and the expected value of each item in the knapsack will be given by the expected gain in the value function from calling the debtor. Note that in this formulation, the discount factor naturally disappears since future arrivals do not affect current decisions. In the next section, we elaborate on how to estimate the value function of each debtor.

3.4 Value function approximation with machine learning

We use value function approximation (VFA) to approximate the value of the states of the MDP described in the previous section. Any function can be used to approximate the value function, including radial basis functions, polynomials, neural networks, and decision trees (Bertsekas and Tsitsiklis 1995). VFA has been successfully applied in optimization in a variety of problems, such as large-scale resource allocation (Powell and Topaloglu 2006), multi-priority patient scheduling (Patrick et al. 2008), and autonomous inverted helicopter flight (Ng et al. 2006). Recent breakthroughs in machine learning—notably convolutional neural networks—have sparked the field of deep reinforcement learning, which allows for VFA through visual images. For example, AlphaGo was able to exploit this approach by successfully approximating the 10^{170} state space in the game of Go and defeat the world's best human players (Silver et al. 2016).

In this paper we use another state-of-the-art machine learning algorithm for VFA, namely, gradient boosted decision trees (GBDT). This is a more suitable algorithm for prediction problems that are arranged in the standard tabular structure and has been the dominant algorithm in winning well over half of all machine learning competitions in 2015, including the KDD Cup (Chen and Guestrin 2016). It was also found by Olson et al. (2018) to be the best algorithm when benchmarked against twelve other algorithms for 165 publicly available classification problems. In Section 3.4.3, we provide details on the GBDT algorithm.

We use the GBDT model to construct a mapping $\hat{V} : \bar{\mathcal{X}} \to \mathbb{R}$ that approximates the value function, thereby circumventing the problem of having to solve (1). This approximation is used to optimize the actions, i.e., to determine which debtors are to be called on a given day. In the following two sections, we show how we construct the mapping $\hat{V}$ (Section 3.4.1) and optimize actions based on this approximation (Section 3.4.2).

3.4.1 Estimating the predicted repayment probability

To approximate the value of a debtor being in a particular state, we estimate the debtor's predicted repayment probability (PRP), which is defined as the likelihood of recovering the full debt during the collection phase. Partially repaid cases are considered to be unpaid as the Collector only receives credit for fully collected cases. Our approach is to estimate the PRP based on historical data by means of a GBDT model as follows. Suppose a certain debtor is $k \in \mathbb{N}$ days into the collection process, and consider all closed cases that once were k days into the collection process as well, i.e., all closed cases that either did not settle their debt within k days or were not written off within k days. We use these closed cases to train a GBDT model that predicts the likelihood of recovering the debt of the debtor currently considered. We formalize this procedure as follows.

Let $n \in \mathbb{N}$ be the total number of closed cases in our dataset, i.e., cases for which the debt was either recovered or written off, and for which the debtor is no longer being contacted. Let $i \in \{1, 2, \ldots, n\}$ and define $\tau_i \in \mathbb{N}$ as the total number of days debtor i spent in the collection process. For all $s \in \{1, 2, \ldots, \tau_i\}$, let $x_i^{(s)} \in \mathcal{X}$ be the state of debtor i at s days since arrival. We optimize the phone calls during the first $K \in \mathbb{N}$ days of the collection process of each debtor. Although, theoretically, K is unbounded, in our practical implementation we set K such that virtually all calling efforts take place in the first K days. For all $k \in \{1, \ldots, K\}$, denote by

$$\mathcal{I}_k := \{i : k \leq \tau_i,\ i \in \{1, \ldots, n\}\}$$

the index set containing all the closed cases in the dataset that were still in the collection process k days after arrival.

Furthermore, we denote by $y_i \in \{0, 1\}$ the eventual outcome of the collection process: $y_i = 1$ if the debt of debtor i was fully recovered after τ_i days, i.e., during the collection phase, and $y_i = 0$ otherwise, meaning that the debt was either written off or recovered after legal actions. Hence, $x_i^{(k)}$ and y_i are the state of debtor i after k days and the eventual outcome of the collection process, respectively, for all $k \in \{1, \ldots, K\}$ and all $i \in \mathcal{I}_k$.

Our approach is to train one GBDT model for each number of days since arrival $k \in \{1, \ldots, K\}$ as follows. Let $k \in \{1, \ldots, K\}$. Then, we train model k by using $(x_i^{(k)})_{i \in \mathcal{I}_k}$ as features (or independent variables) and $(y_i)_{i \in \mathcal{I}_k}$ as target (or dependent) variables. We denote the trained GBDT model by $f_k : \mathcal{X} \to (0, 1)$, where f_k maps the state of a debtor after k days to a prediction for the likelihood that the debt is eventually recovered. This likelihood is exactly the PRP that we introduced earlier on, i.e., if debtor $i \in \mathcal{I}_k$ is in state $x \in \mathcal{X}$ after k days, then $f_k(x)$ represents its PRP.

We train a single model for each number of days since arrival because the data is imbalanced in a sense that there are many more observations for debtors that are earlier in the collection process (i.e., $|\mathcal{I}_k| \geq |\mathcal{I}_{k+1}|$ for each $k \in \{1, \ldots, K-1\}$). This is because cases are closed as soon as the debt is fully recovered or written off. If we train a single model, this could cause the GBDT model to be biased toward better predicting the early part of the process at the expense of the later part. To alleviate this, we follow the aforementioned approach in which we split the data by days after arrival into K sets and train K models.

Summarizing, we compute the PRP of a debtor on a given day by considering debtors that once were in a similar situation before, given that the days since arrival is highly correlated with the rest of the collection process.

3.4.2 Approximating the value function

To approximate the value of the state of a particular debtor, we multiply the debtor's PRP by its outstanding debt. More precisely, let $x = (x_1, \ldots, x_N) \in \bar{\mathcal{X}}$ be the state of the debtor portfolio at a certain point in time and let $k_i \in \mathbb{N}$ denote the number of days debtor $i \in \{1, \ldots, N\}$ has been in the collection process. Our approximation for the value of being in state x is

$$\hat{V}(x) := \sum_{i=1}^{N} f_{k_i}(x_i) \cdot \text{debt}_i, \tag{2}$$

where debt_i denotes debtor i's current outstanding debt. When slot $i \in \{1, \ldots, N\}$ of the state space is not used, we set $\text{debt}_i = 0$. Observe that, when the objective is to maximize the number of fully collected cases (irrespective of the amount of debt recovered), we can accommodate for this by setting $\text{debt}_i = 1$ for all $i \in \{1, \ldots, N\}$.

Our proposed approximation in Equation (2) implies that we consider the Collector's portfolio on a particular day as an assortment of independent debtors in different states of the collection process, and compute the value of the portfolio as a sum of their individual values. This approximation allows us to evaluate policies by computing the difference in PRP with and without making a phone call to a particular debtor.

To formalize this, let $\psi : \mathcal{X} \to \mathcal{X}$ be the mapping that takes as input a debtor's state and then updates this state as follows: i) increase the feature *number of previous collector-debtor interactions* by one; ii) increase the feature *number of previous phone calls* by one; iii) set the feature *days since last collector-debtor interaction* to zero; and iv) set the feature *days since last phone call* to zero (see also the state space

description in Section 3.3.1). Our approach is to determine the marginal value of making an additional call to debtor $i \in \{1, \ldots, N\}$ by computing

$$[f_{k_i}(\psi(x_i)) - f_{k_i}(x_i)] \cdot \text{debt}_i. \tag{3}$$

Recall that f_{k_i} maps debtor i's state to the PRP, i.e., to a prediction of the likelihood that debtor i will *eventually* repay, without needing to explicitly consider potential future states. Hence, Equation (3) provides us with a measure to compare the added value of calling different debtors. Naturally, the policy on day t is to call the c_t debtors for which Equation (3) is the highest (recall that c_t denotes the capacity of the Collector on day t). In the following section, we provide background on GBDT and discuss why it works well in this particular case.

3.4.3 Gradient boosted decision trees

GBDT, also called *gradient boosting machines* and *multiple additive regression trees*, falls under the general paradigm of ensemble methods in machine learning (Dietterich 2000). The algorithm works by constructing multiple decision trees using the *classification and regression trees algorithm* (CART, Breiman et al. 1984) and combining these into a so-called committee, in which the predictions of the individual trees are combined to form one prediction (usually via a weighted average). We first describe how CART works and what its drawbacks are. Then, we explain how ensembles of trees overcome these drawbacks. Finally, we describe the GBDT algorithm and discuss why it works for our problem of predicting the repayment of debt.

The CART algorithm works by recursively partitioning the feature space into non-overlapping rectangular subsets and making a prediction for the target variable for each of these subsets. This is done by splitting, in each recursion, the feature that minimizes a certain error metric (e.g., mean squared error or Gini impurity). This procedure is myopic in a sense that the partitioning decision does not consider future partitionings. As a result, CART does not guarantee a globally optimal partitioning.

A major drawback of CART is its propensity to overfit on training data, which results in a model that generalizes poorly to unseen data. Ensembles of CART models have been successfully used to overcome this. Early ensembling techniques, such as bootstrapped aggregating, commonly referred to as *bagging*, work by generating multiple versions of a prediction algorithm by using randomly selected subsamples of the training data (Breiman 1996). The random forest algorithm is an example of a bagging algorithm. Subsampling observations via bootstrapping adds variation to the training data, which leads to significantly different trees being built, resulting in reductions in error rate by 20-89% (Breiman 2001).

Unlike bagging, where trees are built independently, GBDT builds trees sequentially. This is called *boosting* and works as follows. The goal of GBDT is to minimize a loss (or: objective) function that maps the predictions to a score that measures the quality of the predictions. Theoretically, any differentiable function can be used as a loss function. We use the *logarithmic loss* function, which is the standard choice for binary classification problems and is defined as follows. Suppose we are training model f_k for $k \in \{1, \ldots, K\}$, then the logarithmic loss function $L : \mathbb{R}^{|\mathcal{I}_k|} \to \mathbb{R}$ is

$$L(z) := -\left(\sum_{i \in \mathcal{I}_k} y_i \cdot \log(\sigma(z_i)) + (1 - y_i) \cdot \log(1 - \sigma(z_i)) \right), \tag{4}$$

where $\sigma : \mathbb{R} \to (0, 1)$ is defined by $\sigma(u) := (1 + e^{-u})^{-1}$ for $u \in \mathbb{R}$. The GBDT algorithm repeats Step 1-3 below a prespecified number of times, where $\epsilon > 0$ is set as a hyperparameter:

Step 0. Initalize with $z_i \leftarrow \sigma^{-1}\left(\frac{1}{|\mathcal{I}_k|} \sum_{j \in \mathcal{I}_k} y_j\right)$ for all $i \in \mathcal{I}_k$.

Step 1. Compute the gradient of the loss function $\frac{\partial L(z)}{\partial z_i} = \sigma(z_i) - y_i$ for all $i \in \mathcal{I}_k$.

Step 2. Train a regression tree using $-(\sigma(z_i) - y_i)$ for all $i \in \mathcal{I}_k$ as the target variables.

Step 3. Update $z \leftarrow z + \epsilon z'$, where $z' \in \mathbb{R}^{|\mathcal{I}_k|}$ are the predictions from Step 2. Go to Step 1.

When the algorithm terminates, $\sigma(z_i)$ is GBDT's prediction for y_i for all $i \in \mathcal{I}_k$.

By iteratively building regression trees on the negative gradient in Step 2, newly built trees are optimized for observations that are difficult to predict, thereby improving the overall model fit with each iteration. For a more detailed discussion on GBDTs we refer the reader to Friedman (2001) and Friedman et al. (2001).

Improving the model fit of the training data does not guarantee generalization to unseen data. Therefore, a cap in the number of iterations is required to prevent overfitting. The cap in the number of iterations, along with other hyper-parameters, such as maximum depth per tree, can be tuned using a training-validation framework. The implementation of GBDT used in this paper is LightGBM, which is a fast and distributed open source GBDT framework developed by Microsoft (Ke et al. 2017).

CART, and GBDT, in particular, is well suited for our prediction problem for two reasons. First, since CART works by partitioning the data, it is invariant to monotonic transformations of the features. This differs from models such as logistic regression where substantial efforts in finding the best functional transformations of the features are required to tune the model to achieve better prediction performance. This means that we can directly use the debtor's collection state as features in a CART model without performing any functional transformations. Second, as

Table 1: Observable collector-debtor interactions.

Variable	Type	Description
Debtor ID	Integer	-
Date	Date	-
Communication type	Categorical	Letter, e-mail, or phone call
Communication direction	Binary	In- or outbound (regards to 'Communication type')
Reached	Binary	In case of outbound phone call
Document type	Categorical	In case of outbound letter / e-mail
Promised to pay	Binary	If debtor promised to pay

a consequence of recursive partitioning, CART implicitly takes into account feature interactions that can lead to improved prediction accuracy and better state-value approximations. Again, for other models such as logistic regression, the feature interactions must be defined manually.

3.5 Data description

To train and validate our proposed value function approximation method, described in the previous section, we rely on a dataset provided by the Collector. This dataset contains information on 80,138 debtors that arrived between January 1, 2014 and September 30, 2016. All these debtors are individuals, who are clients of the same insurance company. This insurance company offers all kinds of insurance products, such as car and travel insurance plans. The dataset comprises four data sources: i) debtor-specific information: customer tenure, the type of insurance product, whether the Collector has tried to collect from the debtor on a previous occasion, date of arrival, postal code, original debt, and the collection fee; ii) log of historical interactions between the Collector and the debtor, see Table 1; iii) log of incoming payments; and iv) log of status and substatus changes. The status and substatus changes pertain to information that is used by the Collector to characterize the current state of a debtor. The status is active (the debtor is currently being pursued), inactive (the debt has been paid or the debt has been written off), or on hold (the case is currently being investigated, i.e., there is reason to believe that the debt has already been paid or is inadmissible). The substatus describes the status in greater detail—it indicates: in which stage of the process a debtor is (i.e., which document has been sent most recently); if the debtor has agreed to a payment plan; if the debtor has violated a payment agreement; if the contact details are incorrect; etc. In addition, we enriched the dataset by adding for each debtor the average disposable income in the postal code area where the debtor resides plus data on the distribution of age

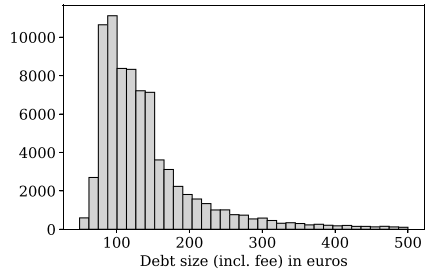


Figure 2: Distribution of debt amounts.

Note. For readability, we capped the debt at €500.

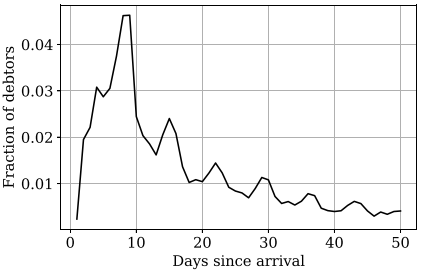


Figure 3: Full repayment over time.

Note. The y-axis pertains to the fraction of the debtors that repaid their debt (in full) on the corresponding day.

groups by leveraging publicly available data[1]. Using these data sources we are able to compile the state space features described in Section 3.3.1.

In Figure 2 to Figure 5, we illustrate the characteristics of the data that we used. In Figure 2, we present a histogram of the initial debt amount including collection fee (2,430 debtors, or 3%, have a higher initial debt than €500, with a maximum of €4,779). The histogram reveals that most debts are in the €50–€200 range, but that the distribution has a heavy right tail, with occasionally large amounts. It holds that the smaller the debt, the greater the likelihood that the debt is recovered by the Collector—of all amounts smaller than €300, for example, approximately 72% is recovered in the first fifty days, whereas for amounts larger than €300 this is only 53%.

Figure 3 illustrates the fraction (or relative frequency) of repaying (the full amount) during each of the first fifty days of the collection process. For example, approximately one percent of all debtors pay off all of its remaining debt on the twentieth day since arrival. The figure reveals that after the first week there is a negative trend observable, but that jumps occur regularly, which are due to letters that are sent and payment due dates that expire.

The bar chart in Figure 4 illustrates the number of outbound calls made on each of the first fifty days of the collection process of all debtors. For example, approximately 20,000 calls are made to debtors on the third day after their arrival. The figure clearly shows that the calls are clustered after letters have been sent—around day 2, 11, 21, and 30. The last cluster that is observable, around day 43, corresponds to the payment due date of the fourth and final letter.

[1]Central Bureau of Statistics, https://www.cbs.nl/nl-nl/maatwerk/2017/15/besteedbaar-inkomen-per-postcodegebied-2004-2014, accessed October 11, 2017.

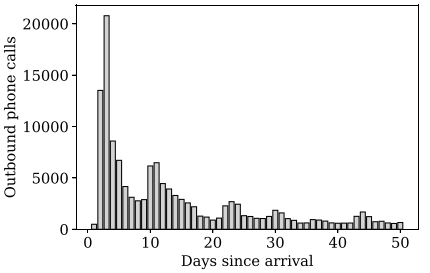


Figure 4: Number of outbound calls.

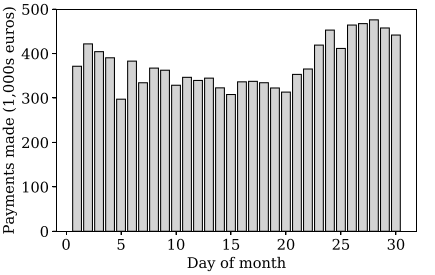


Figure 5: Debt recovered for each day of the month.

Finally, in the bar chart in Figure 5 the inflow of money over different days of the month is illustrated (e.g., the first bar corresponds to the amount of debt that is paid on the first day of the month). The inflow of money peaks at the end of the month after people have received their paycheck, and then gradually decreases again over time.

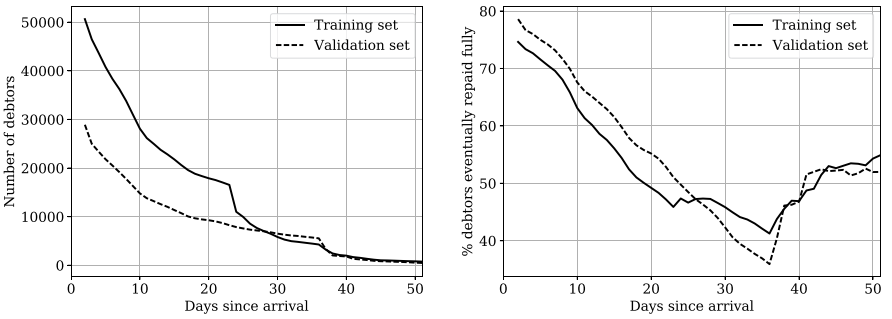
3.6 Model estimation results

In this section, we describe how we train and validate the GBDT binary classification models that predict if debtors are going to repay, based on their current state (see Section 3.4.1). First, we set up a training and validation framework (Section 3.6.1). Then, we evaluate the prediction performance (Section 3.6.2) and illustrate debtor-specific prediction trajectories (Section 3.6.3). Section 3.6.4 contains an analysis of feature importance. Finally, in Section 3.6.5, we analyze the marginal effect of phone calls on PRP.

3.6.1 Training and validation data

We split the dataset into a training and validation set: debtors who arrived between January 1, 2014 and July 19, 2015 are used for training, and debtors who arrived between July 20, 2015 and September 30, 2016 are held out from training and used for validation.[2] The decision to split the training and validation set by date is to mimic practice, where only data of the present is available when making predictions about the future. We chose the specific split date arbitrarily, but in practice the split is often made such that 60-70% of the data is used for training and the remaining 30-40% is used for validation. As a result, the training and validation set contain 50,624

[2]There is often ambiguity between the definition of validation and test sets. In this paper, we define the validation dataset as the holdout set which we use to evaluate our GBDT models.



(a) (b)

Figure 6: Number of debtors (a) and percentage of debtors that repaid (b) for various days since arrival.

and 28,900 debtors, respectively. A small number of debtors had only inadmissible statuses and therefore were not included for training or evaluation, hence the total number of debtors between the training and validation set is less than the total reported in Section 3.5.

Figure 6 (a) shows the number of debtors still in the collection process as time passes for both the training and validation datasets. The figure reveals that the number of debtors decreases as time goes by, which is the result of repayment, writing-off debt, and legal action. There is a sudden drop in the number of debtors around day 23 and day 37 in the training and validation set, respectively. This is due to changes in the collection process on January 1, 2015, where the date of initializing the legal phase was postponed from day 23 to a variable number between days 32 and 42.

Figure 6 (b) illustrates the relative frequency of debtors for various days since arrival from the training and validation sets. For a specific number of days since arrival $k \in \{1, \ldots, 50\}$, the graph illustrates the percentage of debtors that eventually repaid during the collection process given that the debtor is still in the collection process after k days. The percentage of debtors that settles its debt completely initially decreases as their cases are further into the collection process. This suggests that debtors who are more able and willing to repay their debt will do so quickly, while those who are less able or willing to repay may require more effort. Around day 35, the percentage starts increasing again, which can be attributed to the fact that the Collector starts writing-off debtors that are impossible to collect from and, consequently, the remaining debtors are not written-off and are more likely to repay.

3.6.2 Debt repayment prediction performance

We measure the quality of the GBDT binary classification models $f_1, \ldots, f_K$ by their ability to distinguish repaying from non-repaying debtors. Recall that, for all $k \in \{1, \ldots, K\}$, f_k maps the state of a debtor after k days to a prediction for the probability that this debtor repays its debt prior to legal action (i.e., the PRP, see Section 3.4). Due to the low number of debtors remaining late in the collection process, we limit our model to only consider debtors up to 50 days since arrival (where only 1,147 and 620 debtors remain in the training and validation set, respectively), thus we set $K = 50$. Finally, the first phone calls were made starting from day 2 so no debtors are considered for day 1.

The models $f_2, \ldots, f_{50}$ are trained using the training data as described in Section 3.6.1. For details on the construction of the features (independent variables) and the target (dependent) variable for each model, we refer to Section 3.4.1 and Section 3.4.2. Using the trained models, for each debtor in the validation set we compute the PRP for each day this debtor was in the collection process. To compare the PRPs with the actual outcomes, we use the area under the receiving operator curve (AUC). Here, the AUC can be interpreted as the probability that we rank a randomly selected debtor that eventually settles its debt as more likely to repay than a randomly chosen debtor that did not repay (Fawcett 2006). We achieve an AUC of 0.689 where, in comparison, the AUC score of random guessing (i.e., predicting 0 or 1 with equal probability) or naively setting the repayment probability equal to the empirical probability is equal to 0.5.

We use AUC since it measures how well we are able to rank debtors based on their likelihood of repayment, which fits our optimization procedure in which we rank the marginal effect of phone calls based on PRP (see Equation (3) and the discussion below it). Moreover, alternatives like the logarithmic loss and accuracy depend on the distribution of the two classes (paying and non-paying debtors), which is undesirable as it varies over time.

We also compute the AUC over time by computing the AUC of each model $f_2, \ldots, f_{50}$ separately. More precisely, for each $k \in \{2, \ldots, 50\}$, we use f_k to compute the PRP at day k for each debtor in the validation set that was still in the collection process after k days since arrival. Figure 7 shows the AUC scores over time for the GBDT model. We also compare it against two other benchmark models: GBDT without collector-debtor interaction features (e.g., days since last phone call), and logistic regression with all features.[3] GBDT's performance improves as debtors are further in the collection process up to around day 40, and then deteriorates. In contrast,

[3]All categorical features are dummy encoded to ensure consistency across the three models.

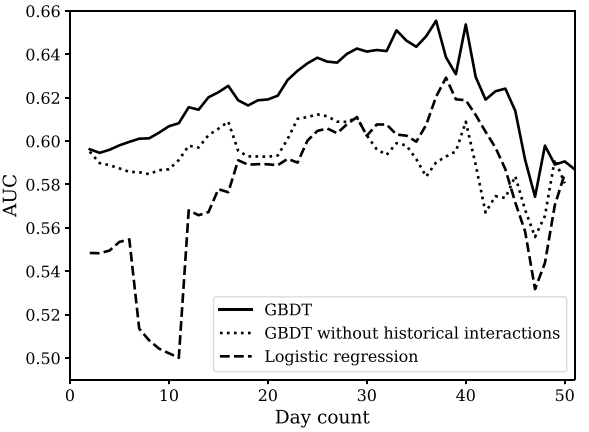


Figure 7: The AUC for GBDT, GBDT without historical interactions, and logistic regression.

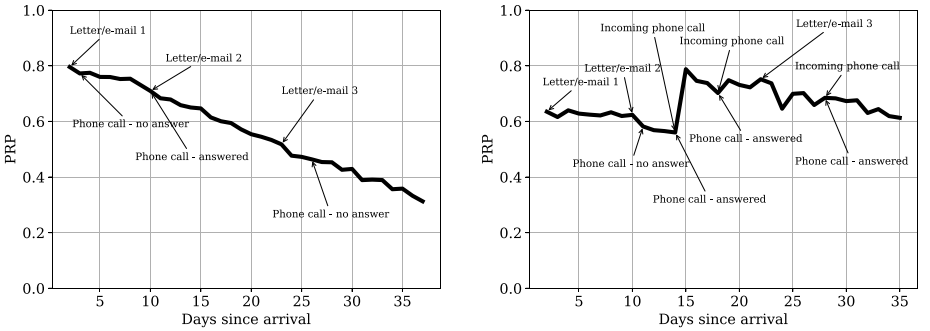
GBDT without collector-debtor interaction features has consistently lower AUC and does not improve for debtors further in the collection process. This suggests that past collector-debtor interactions have explanatory power and that the predictive power increases when more information about the debtor becomes available.

The logistic regression model exhibits a similar trend as the GBDT model in Figure 7. However, it consistently underperforms both GBDT models up to day 29, beyond which it surpasses the GBDT model without collector-debtor interaction features, but still underperforms the GBDT model with the complete feature set. Even though the logistic regression model uses the exact same features as GBDT, it does not perform as well because it imposes a specific parametric specification on the features and cannot use information from feature interactions (e.g., the combined effect of five prior interactions and one prior phone call and eight days since the previous phone call). The AUC scores of all three models deteriorate after day 40. This is likely due to a decreasing number of training observations—by day 40 only 2,141 debtors (4.2%) remain in the training set.

3.6.3 Illustration of individual PRP trajectories

Our framework allows us to look back at individual debtor histories and observe the PRP dynamics with respect to events during the collection process. Figure 8 depicts the collection process for two debtors from the validation set: one which did not repay its debt (panel (a)) and one which did repay (panel (b)).

The debtor in panel (a) starts with a PRP of around 0.80, which then gradually decreases to around 0.30. Over time, no events occur that indicate that the debtor



(a) (b)

Figure 8: PRP over time for a debtor that did not repay (a) and that did repay (b).

is going to settle and, consequently, the PRP declines gradually over time. It turned out that the debtor in panel (a) had moved to a different address and could not be found, explaining the unsuccessful collector-debtor interactions as they likely were not with the actual debtor.

With a PRP of approximately 0.60 initially, the debtor in panel (b) starts with a lower PRP than the debtor from panel (a). However, the PRP does not deteriorate much over time and positive jumps occur frequently. These jumps are due to the fact that events occur that positively influence its PRP. Specifically, the debtor both answers calls and makes phone calls to the Collector itself, which are indications that the debt will likely be recovered.

3.6.4 Feature importance for predicting PRP

A natural question to ask is which features are informative in predicting PRP. In tree-based models, the (relative) importance of each of the features is not immediately observable. Unlike linear models, tree-based methods do not produce a set of coefficients that represent the (linear) effects of the features on the (predicted) outcome. A number of methods have been developed in recent times to help interpret or explain the predictions of complex machine learning models (Ribeiro et al. 2016, Lundberg and Lee 2017), but in this paper, we rely on a simpler approach that is already included as part of the GBDT implementation. Given that trees are built by sequentially partitioning the features that have the most predictive power (see Section 3.4.3), it can be inferred as follows what the most informative features are.

Suppose we are training model f_k for $k \in \{1, \ldots, K\}$ and we are building a regression tree at Step 2 from the algorithm description in Section 3.4.3. Suppose this tree has $T \in \mathbb{N}$ terminal nodes and denote the rectangular, non-overlapping partitioning of

the feature space by $R_1, \ldots, R_T$. Then, the so-called *variance gain* at a terminal node $t \in \{1, \ldots, T\}$ from splitting feature $j \in \{1, \ldots, n_f\}$, where $n_f \in \mathbb{N}$ denotes the number of features, at $d_j \in \mathbb{R}$, is equal to (Friedman 2001, Nielsen 2016, p.62):

$$\frac{\left(\sum_{i:x_i \in R_t, x_{ij} \leq d_j} (\sigma(z_i) - y_i)\right)^2}{2|\{i : x_i \in R_t, x_{ij} \leq d_j\}|} + \frac{\left(\sum_{i:x_i \in R_t, x_{ij} > d_j} (\sigma(z_i) - y_i)\right)^2}{2|\{i : x_i \in R_t, x_{ij} > d_j\}|} - \frac{\left(\sum_{i:x_i \in R_t} (\sigma(z_i) - y_i)\right)^2}{2|\{i : x_i \in R_t\}|}. \tag{5}$$

This expression approximates the gain (i.e., decrease) in the logistic loss function when, at node $t \in \{1, \ldots, T\}$, splitting feature $j \in \{1, \ldots, n_f\}$ at value $d_j \in \mathbb{R}$. The regression tree from Step 2 of Section 3.4.3 is built by iteratively splitting the feature so that (5) is maximized (the implementation of GBDT that we rely on (LightGBM) uses an approximation of (5) that is more computationally efficient).

We use the variance gain, as implemented in LightGBM (Ke et al. 2017), to analyze the importance of features by summing, for each feature, the variance gains of all the nodes in all the trees at which this feature was split. For easier interpretation, we normalize the variance gain of each feature so that the sum of all feature variance gains equals 100. Since we train one GBDT model for each of the forty-nine days (days 2 to 50) into the collection process, there are forty-nine sets of feature importances. Table 2 presents the relative variance gain of each feature for five GBDT models, namely for f_5, f_{15}, f_{25}, f_{35}, and f_{45} (corresponding to 5, 15, 25, 35, and 45 days since arrival, respectively). The average gain of each feature across all forty-nine models is also included. Note that feature importance does not specify in which direction a feature affects the model's predictions. GBDT does not assume monotonic feature effects so the same feature can have positive or negative effects under different conditions.

Two observations can be made from Table 2. First, the features *initial debt amount* and *customer tenure* are highly influential in predicting repayment probability of debtors, especially early in the collection process. A reason for this may be that debtors that owe more money are less likely to repay their debt, and debtors that have been customers for a longer period of time are more likely to repay their debt as they've probably had a better relationship with the debt owner. Indeed, when we compute the correlation between PRP and *initial debt amount* and *customer tenure* for debtors in the validation set for each model across all models we obtain an average of -0.26 and 0.42, respectively. Conversely, a number of features have very low impact, such as *has partially repaid debt*, *repayment plan in place*, and *product type*. This is surprising because we expected the debtors that have already repaid part of their debt or have agreed to a repayment plan would be more likely to fully repay their debt. However, this seems to not be the case. A possibility that these

Table 2: Feature importance.

	Percent of the total variance gain of each feature					
	5 days	15 days	25 days	35 days	45 days	Average
Initial debt amount	22.02	19.16	15.62	11.82	16.30	16.24
Customer tenure	24.07	18.98	15.92	14.78	11.75	15.96
Has partially repaid debt	0.00	0.05	0.00	0.07	0.00	0.08
Repayment plan in place	0.00	0.02	0.79	0.32	0.73	0.52
Phone number is available	19.65	10.73	2.97	0.82	0.18	6.65
E-mail address is available	2.80	1.44	1.11	1.21	0.96	1.43
Product type	0.00	0.00	0.00	0.00	0.00	0.00
Amount repaid already	0.00	0.17	0.05	0.17	0.47	0.44
Collector collected from debtor before	1.80	2.74	2.03	1.03	0.99	1.71
Average income in postal code area	11.44	9.89	8.79	8.62	7.32	9.00
Share of people under 30 in postal code area	5.75	5.21	6.21	6.49	7.84	6.34
Current substatus	0.02	2.66	5.76	5.50	2.59	3.29
Passed final letter	0.00	0.00	0.00	0.40	3.22	0.90
Has answered a phone call	0.67	0.35	0.60	0.14	0.09	0.42
Promised to repay	0.03	0.90	1.55	1.13	0.22	1.12
Number of previous collector-debtor interactions	1.47	2.34	2.38	2.65	2.67	2.32
Number of previous phone calls	0.61	1.23	1.42	1.46	1.83	1.50
Days since promise to repay	0.00	0.95	5.50	2.74	3.32	2.54
Days since last collector-debtor interaction	0.50	1.06	2.75	8.88	9.85	4.93
Days since last phone call	0.55	1.75	2.65	3.22	4.92	2.84
Days since last answered phone call	0.64	2.12	2.76	3.37	4.84	2.73
Days since last incoming contact	0.92	2.92	5.26	7.57	6.29	4.03
Days since last incoming e-mail	0.92	1.56	2.26	2.71	1.16	1.81
Days since last incoming phone call	3.86	10.53	10.77	11.41	5.98	8.95
Day of week	1.29	2.10	1.17	1.78	1.40	1.55
Week of month	0.99	1.15	1.70	1.69	0.87	1.28

The relative percent variance gain of each feature from the state space is tabulated for GBDT models trained on debtors that are 5, 15, 25, 35, and 45 days since arrival. The average gain for each feature across all models (days since arrival of 2 to 50 days) is tabulated in the last column.

features have low impact is because there is little variation in the training data. In particular, *product type* actually has zero variation in the training data as it was first recorded in January of 2016, so that all observations in the training data have the same value for this feature.

The second observation is that the impact of features varies for different days since arrival. Debtor-specific features, such as *phone number available*, tend to have relatively high impact in models that pertain to fewer days since arrival (e.g., five days). On the contrary, features related to collector-debtor interactions, such as *days since last collector-debtor interaction*, have more impact later in the collection process. The GBDT predictions also reflect this, as the correlation coefficient between *phone number available* and PRP in the validation set decreases from 0.50 at five days since arrival down to 0.03 at forty-five days since arrival, and the magnitude of correlation between *days since last collector-debtor interaction* and PRP increases from -0.02 at five days since arrival to -0.19 at forty-five days since arrival. Finally, some features, such as *current substatus* and *days since last incoming phone call*,

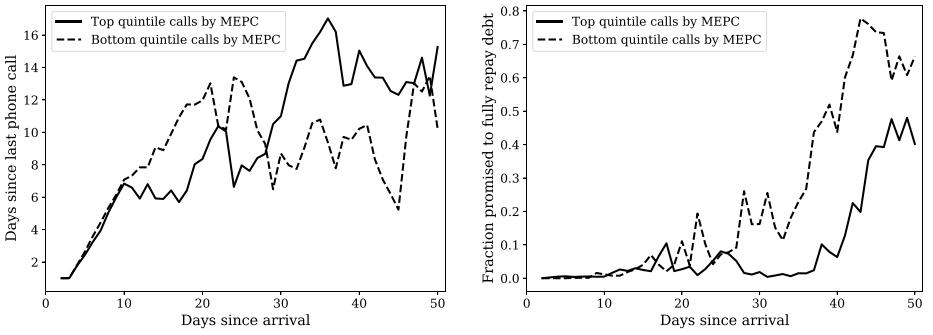
have the bulk of their effect towards the middle of the collection process. This shows that different features predict repayment in different ways throughout the collection process.

3.6.5 Marginal effect of phone calls on PRP

Our approach to maximize the amount of debt collected is to compute the difference in PRP between making and not making an additional phone call to a debtor—see Section 3.4 and Equation (3) in particular. To analyze how the marginal effect of a phone call on PRP (abbreviated to MEPC) depends on the state of a debtor, we compute the MEPC for every debtor in the validation set for every day that the debtor was eligible for receiving a phone call. We identified 432,555 of such *potential* phone calls across the 28,900 debtors in the validation set over 497 days. The average MEPC over these phone calls equals 0.92%, with a standard deviation of 3.73%. This indicates that, on average, phone calls have a positive effect on PRP, but that the effect varies substantially across debtor states.

Analyzing the impact of a feature on the MEPC is non-trivial, given that features interact and correlate with each other. For example, *current substatus* and *number of previous collector-debtor interactions* are highly correlated as they change in similar directions with the amount of time spent in the collection process. This makes it challenging to attribute MEPC to a single feature. Instead, we present some specific insights to illustrate how MEPC can differ under different conditions. To do so, we sort the 432,555 phone calls in the validation set by MEPC and compare average feature values for each day since arrival between the top and bottom quintiles, which represent the most and least effective phone calls. Figure 9 contains plots for two of the features, *days since last phone call* and *promised to repay*.

For *days since last phone call* (Figure 9a), there is no difference between the most and least effective phone calls in the first 10 days of the collection process. From day 11 to 28, the debtors that have been called more recently seem to be better options for the next call. Starting from day 29, the effect flips around and it becomes better to call the debtors that were called less recently. Arguably, this has to do with intrinsic differences in the types of debtors that remain earlier versus later into the collection process. More debtors that are reachable and able to repay their debt remain in collections between 10 and 28 days into the process (i.e., after the first letter but before the final letter). These debtors likely require more persuasion so calling them more often could increase collectibility. As these debtors eventually repay their debt, the debtors that remain later into the collection process are more likely to either be unreachable or unable to repay their debt, and as a result, there is less value in calling them often.



(a) (b)

Figure 9: Average values for the top and bottom quintiles by MEPC for days since last outbound phone call (a) and fraction of debtors that promised to fully repay debt (b).

For *promised to repay* (Figure 9b), it seems in general better to call debtors that have not previously promised to repay their debt. The fraction of debtors that have promised to repay their debt is similar between the top and bottom quintile of calls up to day 26 and differs significantly afterward. The intuitive explanation for this observation is that debtors that have promised to repay their debt are already likely to repay their debt without further intervention. Until they have broken their promise, it is better to accept their promise and simply wait for repayment.

In Figure 10, we plot the MEPC against *days since arrival*. Phone calls seem to have limited effect until day 20, and then become increasingly effective until day 37, after which they become less effective again. Although there may be many reasons causing this, we believe the increase in effectiveness is because the debtors that are able and willing to repay their debt will likely do so early in response to the letters and e-mails, so phone calls are unnecessary and add no value to the collectibility of these cases. On the contrary, debtors that have not already repaid their debt two or three weeks into the collection process are more likely to be unwilling to repay, thus phone calls can add greater value. Phone calls begin to lose effectiveness after day 37 as more of the debtors remaining are either unreachable or are unable to repay the debt. In the former case, the debtor will be escalated to the legal phase, and in the latter case, an agreement might be reached between the Collector and a debt management intermediary. Neither of these cases will result in repayment via the collection process and spending additional phone calls on these debtors will be wasted effort.

We emphasize that both the training and validation datasets are realizations of the Collector's actual collection process and do not reflect any form of randomized experimentation. Therefore, many of the possible states may have been under-observed

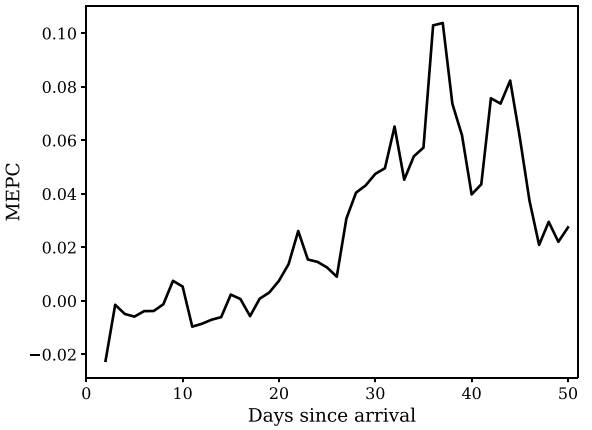


Figure 10: MEPC for each day since arrival.

(e.g., zero phone calls made within the first 15 days of the collection process). Moreover, because the collection process is path-dependent and we do not know how individual collectors select which debtors to call, we do not know what would have happened had different actions been taken and cannot be sure that our MEPC estimates are unbiased. Ultimately, a controlled field experiment is necessary to understand the true value of using a data-driven prediction model to optimize phone calls.

3.7 Controlled field experiment

In this section, we present the results of a controlled field experiment that we conducted. First, in Section 3.7.1, we discuss how we designed the experiment. Then, in Section 3.7.2, we present the results and discuss the collection performance of our proposed policy. Finally, in Section 3.7.3, we analyze how our proposed policy differs from the incumbent policy by inferring how the (states of) debtors that our policy calls differ from those called by the incumbent policy.

3.7.1 Experimental setup

To evaluate the performance of the GBDT-optimized calling policy (GOCP), which is described in Section 3.4.2, we ran a controlled field experiment spanning a period of 102 days starting on January 19, 2018 and ending on April 30, 2018. The experiment regards debtors that are clients of the same insurance company used for the data analysis in Section 3.5 and the model development in Section 3.6.

Starting from January 19, 2018, we assigned each newly arriving debtor randomly to one of the following three collection policies: GOCP, the incumbent policy (IP) currently used by the Collector (described in Section 3.2.2), or a third experimental policy, which is not related to this paper. For our analysis, we consider debtors that arrived between January 19, 2018 and February 28, 2018, since for these debtors we observe the outcome of the collection process (given that the experiment ran until April 30, 2018). As a result, a total of 921 debtors are within the scope of our experiment, of which 455 were exposed to GOCP and 466 to IP.

The implementation of GOCP is as follows. At the start of each day, we train fifty GBDT models, as described in Section 3.4.1, based on all the closed cases from January 1, 2016 up to the present day. Then, according to Equation (3) in Section 3.4.2, we compute the predicted marginal effect of making a phone call for each of the open cases that are no more than fifty days into the collection process and are eligible for receiving phone calls. Debtors are not eligible for receiving phone calls if, e.g., the debt is currently being investigated because the debtor disputed the debt (we use the substatus of debtors, see Section 3.5, to verify if a debtor is eligible for receiving phone calls).

The Collector's objective is to maximize the number of cases that are completely recovered, since only in case of full recovery does the Collector receive a collection fee, which is independent of the amount of debt. To this end, when computing the marginal effect of phone calls according to Equation (3), we set $\text{debt}_i = 1$ for each debtor $i \in \{1, \ldots, N\}$ (note that the amount of debt is still part of the state space of a debtor, since it does affect the likelihood of repayment). In this way, GOCP is designed to maximize the number of fully recovered cases.

The cases are then sorted by the predicted marginal effect of calling, and the top 20% of these debtors is added to the pool of planned phone calls for that day. A number of phone calls following other policies (including IP) are also added to this pool. This is done in a way that the capacity of the Collector on a given day is sufficient to make all the phone calls in the calling pool. Agents in the Collector's call center are free to call debtors from the calling pool at their discretion. To prevent bias originating from behavioral changes of the agents, there is no identifier of which debtors are following which policy within the pool of planned calls and the agents were not aware of this experiment.

The Collector is required to always attempt at least one phone call before sending out the next letter. To conform with this, when a specific debtor is not called for a long time, it is artificially prioritized and scheduled for a phone call, so that the next letter can be sent afterward.

Table 3: Summary statistics on the results of the experiment.

	IP	GOCP	
Key figures			
Number of debtors	466	455	
Number of outbound calls	1,119	876	
Number of inbound calls	236	188	
Fraction of total debt recovered	0.572	0.652	
Amount collected per call (€)	31.80	46.30	
Amount collected per outbound call (€)	38.53	56.72	
Descriptive statistics			
Fraction of debtors that is called at least once	0.916	0.831	***
Fraction of outbound phone calls picked up	0.268	0.223	
Fraction of fully collected cases	0.590	0.626	
Average number of days until first phone call	4.5	7.7	***
Average number of outbound calls per debtor	2.40	1.93	***
Average number of days until full repayment	22.2	20.3	
Average initial debt (€)	161.74	166.17	

***, **, and * indicate a statistically significant difference at a significance level of 0.01, 0.05, and 0.10, respectively. Inbound calls are calls made by debtors to the Collector, e.g., after missing a phone call from the Collector or after receiving one of the letters.

3.7.2 Experimental results

We evaluate the performance of GOCP (relative to IP) by considering the state of each of the 921 debtors in the scope of this experiment at the time the experiment ended on April 30, 2018. In Table 3, we provide summary statistics on the performance of IP and GOCP. In general, GOCP is able to (fully) recover more cases and collect more debt in shorter time, whilst requiring much fewer resources.

To see this, observe that IP recovered only 57.2% of the total outstanding debt, whilst GOCP was able to recover 65.2%, which is a substantial relative increase of 14.0%. This corresponded with 62.6% of the cases being fully recovered by GOCP and 59.0% by IP. Of the fully recovered cases, the average number of days until complete repayment is 22.2 for IP and 20.3 for GOCP, suggesting that GOCP is not only more effective at collecting more debt, but that it also does so in less time.

The total number of phone calls made by GOCP is 21.8% lower than IP (876 vs 1,119 outbound calls). Together with the increased amount of debt recovered, this leads to a 47.2% increase in the monetary amount collected per call made by GOCP as compared to IP (€56.72 vs €38.53). When also including the inbound phone calls—usually made by debtors after missing a call or receiving a letter from the Collector—we see the same picture, namely an increase in 45.6% in euros collected per call. Overall, this shows that the calls under our data-driven policy are considerably more effective in terms of return on effort than the incumbent policy.

The (decrease in) number of outbound phone calls under GOCP is mostly the consequence of the implementation of GOCP, in which the 20% of debtors is called

that have the highest predicted marginal increase in PRP (see Section 3.7.1). Hence, the decrease in number of phone calls on itself is not a direct consequence of GOCP. However, it is due to GOCP that the most effective phone calls can be selected, which, in this case, leads to more debt being recovered, whilst requiring fewer phone calls.

Apart from the number of phone calls, the fraction of debtors that receive at least one phone call is also substantially lower for GOCP than for IP, namely 83.1% vs 91.6%. Presumably, this is due to the fact that, on average, GOCP makes the first phone call to a debtor only after 7.7 days, whilst IP does so after 4.5 days (recall that IP always schedules a phone call immediately after sending out the first letter). Consequently, IP calls debtors that would have settled on short-term notice without requiring the persuasion of one of the Collector's agents—thereby wasting expensive capacity. On the other hand, GOCP postpones the first call, thereby allowing debtors to settle and spending only effort on debtors that are not able or willing to pay.

All in all, the results reveal that the data-driven approach to debt collection that we propose substantially outperforms the existing policy that is based on business rules. As such, it shows that the historical data available contains sufficient information to make better decisions in a highly automated manner and that our framework is capable of operationalizing and monetizing this data.

3.7.3 Comparison of GOCP and IP

The previous section revealed that GOCP is much more effective as a calling policy than IP. In this section, we analyze how the debtors that were selected by GOCP differ from those selected by IP. To this end, we identified all phone calls that were the first phone call to a debtor on a given day. These calls are most likely a result of the respective policies and were not in response to an incoming call or a chain of phone communication within a day. Based on these calls, we made a comparison between the state of debtors called by GOCP and by IP, of which the results are presented in Table 4 (see Section 3.3 for a full description of the state space). For example, GOCP calls debtors that are on average 18.6 days into the collection process, whilst for IP this is 15.6 days. Recall that *telephone number available* regards to whether the phone number was known at the time the debtor arrived, i.e., whether it was provided by the debt owner. In case it is not available, phone calls can still be made if a phone number can be retrieved through, e.g., public telephone directories. In the following sections, we discuss insights derived from Table 4.

Table 4: Comparison of the state of debtors that IP and GOCP call.

Debtor state feature	IP	GOCP	
Debtor specific			
Days since arrival	15.6	18.6	***
Initial debt (€)	167.67	173.08	
Has partially repaid debt	0.005	0.006	
Repayment plan in place	0.036	0.045	
Phone number is available	0.712	0.634	***
E-mail address is available	0.765	0.706	***
Amount already repaid (percent of total)	0.002	0.001	
Collector collected from debtor before	0.368	0.326	*
Average income in postal code area (thousands)	33.35	34.28	**
Share of people under 30 in postal code area	0.361	0.361	
Passed final letter	0.001	0.000	
Historical interaction			
Has answered call before	0.231	0.162	***
Promised to repay	0.104	0.069	**
Number of previous collector-debtor interactions	3.9	3.6	**
Number of previous phone calls	1.0	0.9	*
Days since promise to repay	19.5	22.1	
Days since last collector-debtor interaction	2.8	5.9	***
Days since last phone call	13.3	13.5	
Days since last answered phone call	17.1	17.9	
Days since last incoming contact	14.6	17.4	*
Days since last incoming e-mail	12.0	16.7	**
Days since last incoming phone call	18.0	20.8	

Based on 830 and 642 calls initiated by IP and GOCP, respectively. ***, **, and * indicate a statistically significant difference at a significance level of 0.01, 0.05, and 0.10, respectively.

GOCP spends more effort on difficult cases. The results in Table 4 indicate that GOCP, as compared to IP, focuses on debtors that are more difficult to collect from. To see this, first observe that, compared to IP, GOCP calls more debtors that did not answer a phone call before. This suggests that GOCP calls debtors that are more difficult to reach and likely more difficult to collect from. Second, GOCP calls less frequently to debtors that have previously promised to settle their debt. Hence, GOCP spends relatively more effort on debtors that have not yet shown willingness to pay by acknowledging their debt to the Collector. Third, GOCP calls debtors that have been in the process for a longer period of time (18.6 days vs 15.6 days), which also suggests that GOCP focuses on the more difficult cases—given that debtors with a high willingness to pay settle early in the process. Finally, compared to IP, GOCP calls more often to debtors for which less information is available. Namely, the fraction of debtors of which the phone number and e-mail address were known is lower for GOCP. The unavailability of this information makes collection work more difficult, but GOCP suggests that it is worthwhile pursuing those debtors (e.g., by searching for their phone number elsewhere).

Timing of phone calls. IP schedules debtors immediately after they receive one of the four letters that are sent by the Collector, and are then "forgotten" until the next letter has been sent. On the other hand, GOCP calls debtors at its discretion based on the debtors' states. As a result, from Table 4, we clearly see that the timing of phone calls made by GOCP differs from that of IP: GOCP makes on average the first phone call to a debtor 3.2 days later than IP (see Section 3.7.2) and also calls debtors that have been in the collection process longer (18.6 days for GOCP as compared to 15.6 days for IP, see Table 4). In addition, the number of days since last collector-debtor interaction is much higher for GOCP (5.9 days for GOCP vs 2.8 days for IP).

These observations, together with the results from Section 3.7.2, suggest the following. After sending letters, it is worthwhile to give debtors some slack to settle their debt, before spending expensive phone calls on these debtors. Hence, by waiting longer before making the first call and waiting longer after the previous interaction with the debtor, automatically only debtors that are unable or unwilling to pay remain—these are exactly the debtors that the Collector would want to spend capacity on.

Recall that the Collector requires that a phone call is made before debtors can be escalated to the next letter, thereby setting an upper bound on the number of days without a phone call (see Section 3.7.1). Without this business constraint, we expect the results to be even more profound.

Similarities between GOCP and IP. Although GOCP and IP call debtors that differ in several aspects, they are not significantly different in other aspects. It is surprising to see that debtors called by both policies do not significantly differ in outstanding debt (€173.08 vs €167.67 for GOCP and IP respectively). The difference is almost identical to the difference in initial outstanding debt for debtors assigned to the two policies. However, both policies actually select debtors with greater debt than average (€163.93). This is likely due to the fact that debtors with smaller debt amounts tend to repay earlier, and thus more calls are eventually made to debtors with larger debt amounts.

3.8 Conclusion

This paper considers the problem of deciding on a daily basis which debtors a debt collection agency should call, given that only a limited number of calls can be made by its agents. This is a challenging optimization problem since, at any given time, a debtor portfolio consists of a large collection of heterogeneous debtors that are

at different stages in the collection process. Our approach is to formulate an MDP and approximate it through data-driven machine learning methods—thereby circumventing dimensionality issues by relying on historical data. This approach revolves around computing, for each debtor at each state, the predicted repayment probability (PRP) and inferring the marginal increase in PRP when making an additional phone call. We find that in a holdout sample our machine learning technique achieves 0.689 AUC for the PRP estimates. Furthermore, we were able to obtain an improved scheduling policy for our industry partner. We validated this policy in a controlled field experiment conducted with real debtors. The results show that our data-driven policy substantially outperforms the current scheduling policy. Most notably, our policy leads to an increase in the amount of debt collected per outbound call from €38.53 to €56.72, leading to a 47.2% improvement in return on calling effort. The improvement comes mostly from selecting debtors that have been in the collection process longer, have not been contacted recently, and have not previously answered calls nor promised to repay their debt. In general, the proposed policy puts more emphasis on debtors that are harder to collect from and calls are scheduled later in the collection process.

Chapter 4

Optimal Contact Center Staffing and Scheduling with Machine Learning

Chapter Abstract

A fundamental challenge in staffing and scheduling of service systems is ensuring high quality of service (QoS) at minimum costs. This challenge is particularly complex when considering modern contact centers that have multi-skill agents and multi-class customers with heterogeneous arrival rates. Moreover, the inclusion of modern channels such as chat and email further increases the complexity. In this setting, there are no closed-form expressions for QoS measurements and well-known approximation techniques cannot integrally solve the staffing and scheduling problem, resulting in significant over-staffing. Simulations are widely used to accurately provide QoS expectations for staffing schedules, but they are computationally demanding and reliable optimization procedures cannot meet the time demands of practical use. A new approach is required to efficiently find near-optimal staffing schedules for real-life use in service systems. We show how machine learning can be used in combination with simulation to solve the integral staffing and scheduling problem for multi-skill contact centers. This has the potential to provide better solutions for the workforce planning problem faced by contact center managers, and can also be applied to other service systems. We develop a machine learning framework to approximate QoS expectations by predicting simulation outcomes, allowing us to quickly produce a look-up table of QoS for all candidate schedules. The QoS approximations are highly accurate, even when the contact center includes non-traditional channels such as chat and email. We then implement a simple deterministic optimization procedure to obtain schedules that can satisfy QoS targets at low costs. Using numerical experiments based on a real-life contact center scenario, we show that under reasonable time constraints, our method substantially improves upon the best schedules obtained via simulation optimization for multi-skill contact centers, both with and without chat and email. Implementation of our method could result in significant reduction of staffing costs for complex modern multi-skill contact centers.

4.1 Introduction

Contact centers (or call centers), as one of the classical service systems, have been a mainstay in operations management research for many years. One reason for the popularity of contact centers is their importance to the success of businesses. According to a report by Global Industry Analysts, Inc. (2018), the contact center industry is estimated to have generated $200 billion in revenue worldwide in 2017 and is expected to reach over $400 billion by 2022. It is estimated that over 2 million people work in contact centers in the U.S., with millions more working in other countries such as India or the Philippines. Contact centers play an important role in customer service as the first point of contact for customer inquiries and complaints, and sometimes serve as the only method for customers to complete critical tasks (e.g., re-booking flights after unexpected delays or terminating a mobile phone plan). Moreover, with direct access to customers, contact centers also engage in marketing via cross selling of products and customer churn prevention.

As technology progresses, customers are using more communication channels than just the telephone, sometimes preferring to use self-help options, email, social media, or chat. Moreover, customers rarely rely on a single channel, often selecting a channel based on their needs of the moment. For urgent requests that are difficult to explain and require real-time response, a telephone call may still be the ideal option. However, for problems where written instructions will suffice, email or chat could be better as the customer does not have to hold for an agent and can refer back to the discussion in the future. Communication preferences also differ for customers across age groups. Young people, for example, prefer chatting online over talking to strangers, while older people, on the other hand, are still more familiar with phone conversations.

From the point of view of management, the diversity of channels can help smooth workload, offloading non-urgent services to non-real time channels such as email or text message so that agents can handle them during periods of lower calling load. However, the operational challenges faced by contact centers are also more complicated when the system offers more flexibility to customers. Handling telephone requests already requires agents with different skills such as language (e.g., English and Spanish) and product understanding (e.g., flight information and baggage rules). The inclusion of new channels adds another dimension to the system, increasing the difficulty of guaranteeing high quality of service at manageable costs. Furthermore, customers arriving in non-real time channels behave differently, requiring all new metrics of measuring quality of service.

In this chapter, we investigate the staffing and scheduling problem for modern multi-channel contact centers with multiple classes of customers and multi-skilled agents. The problem is as follows—given a contact center with customers arriving from different channels employing agents with different skills, how can the manager staff and schedule agents to meet quality of service (QoS) objectives at minimum costs? This is a fundamental challenge faced by contact center managers due to the labor-intensive nature of customer service. Approximately 60-80% of the operating budget of a contact center is comprised of agent costs (Aksin et al. 2007), thus it is crucial to deploy the right number of agents with the right skills to the right schedules so as to meet the uncertain, time-varying demand of service.

This task of staffing and scheduling agents is generally carried out weekly (or bi-weekly) by the contact center's workforce management team based on the following steps. First, using historical data, a *forecast* of arrival requests for the next week is obtained. The forecast predicts the incoming volume of each type of service request for every interval (usually between 15 and 60 minutes). Then for each service type *independently*, the forecasted call volume is translated to staffing level requirements *per interval* such that a pre-specified QoS can be met. This step is known as *staffing*. In practice, staffing level requirements are usually computed in one of two simple ways:

1. multiply average handling time (AHT) with the forecasted call volume and then offset by an experience-based safety ratio; or
2. leverage Erlang formulas from queuing theory and solve a linear program with advanced call center software.

The first method is a heuristic often used in practice, and the second method is based on the stationary independent period-by-period (SIPP) model of contact centers and is well known in the literature (Gans et al. 2003). Once the interval-independent staffing levels are determined, scheduling software is used to determine the optimal shift configurations. This step is known as *shift scheduling*. Finally, the last step, *rostering*, assigns the agents to shifts. Some extra agents may be added (hired) on short notice if necessary.

While this approach works well in simple contact centers with only calls and single-skilled agents, it becomes unreliable in modern contact centers with multiple channels, multi-skilled agents and heterogeneous customers. First, by calculating staffing levels per service type independently, it is assumed that agents are fully dedicated to each service type (i.e., only single-skill agents are employed). However, most agents are capable of performing multiple skills (e.g., bilingual or can answer both calls and emails) and are shared by a blend of customers based on a routing policy. Thus, the QoS of a service type is not only affected by the number of agents with

the right skill, but also by the configuration of other multi-skilled agents and the routing policy. A second problem arises from treating each interval independently. The QoS targets are usually set for a sequence of many intervals (e.g., 48 intervals for a day of 30-minute intervals or 336 intervals for a week), so optimizing staffing independently for each interval can lead to over-staffing. This problem is amplified when considering non-real time channels such as emails, where interval-independent approaches perform especially poorly because emails are often handled in a different interval from when they arrive. Finally, neglecting the transient effects between consecutive intervals (i.e., not accounting for busy servers and queues at the end of intervals when starting new intervals) also introduces errors (Ingolfsson et al. 2007).

To appropriately choose staffing levels so that scheduling costs can be minimized, the staffing and shift scheduling problems should be considered integrally. Due to complexity, few papers have studied the integrated problem in a setting with multi-skill agents and heterogeneous customers. The key difficulty of this problem is the lack of closed-form expressions for QoS measurements. Consequently, the papers that study the integrated problem either rely on simulation (Cezik and L'Ecuyer 2008, Avramidis et al. 2010) or approximate the QoS by modeling the system into certain existing mathematical models such as fluid models or queuing models (Bodur and Luedtke 2017). However, both methods are difficult to implement in practice.

The approximation methods often need strict conditions such as heavy-traffic systems, short service times, impatient customers, and specific routing policies. Thus they can be less robust and in practice it is difficult to meet all of the requirements. On the contrary, the advantages of using simulation are multi-folded: it tends to produce more reliable results (Ingolfsson et al. 2010, Bodur and Luedtke 2017), takes into consideration the transient effects between intervals, allows for more diversified QoS measurements, makes fewer assumptions so as to be more realistic, and most importantly is the only way to include non-traditional channels such as chat and email. Unfortunately, simulation methods can be very time consuming, which is a significant obstacle in practice as it is common for practitioners to iterate multiple times between shift scheduling and rostering. As a result, we study whether there is an approach that can efficiently (i.e., quickly and with acceptable accuracy) estimate the QoS measurements without strict assumptions.

4.1.1 Contribution

In this chapter, we combine simulation with function approximation to accurately and efficiently estimate QoS and integrally solve the staffing and shift scheduling problems. However, instead of mathematically modeling the system, we develop a framework to approximate the simulation by training a machine learning model using

simulation outcomes. More specifically, we rely on a general simulation model for contact centers, and under any given scenario, we use the simulation to randomly generate a number of possible schedules and record their corresponding QoS. The computational burden in this stage is still manageable as it does not need to be performed in real time. Subsequently, we train a machine learning model on these schedules and are then able to quickly produce a deterministic look-up table with the *predicted* QoS of all possible schedules. Finally, we develop a simple local search algorithm to search the look-up table of schedules to find optimal or near-optimal staffing schedules in real time for the provided scenario.

Using numerical experiments, we show that first, our machine learning framework is able to approximate QoS with very high accuracy, even though only the shift schedule is provided. This is important because errors in QoS predictions propagate to the subsequent optimization problem, potentially leading to falsely optimal results. Next, we show that our framework performs well in finding near-optimal staffing schedules. For the single-skill scenario, our framework is able to find staffing schedules that are 3.8% better compared against the stationary independent period-by-period (SIPP) model. For the multi-skill scenario, our framework is able to find staffing schedules that are between 4.3% and 28.7% better compared to a simulation-based optimization approach. Finally, For the multi-skill scenario that also includes non-traditional channels (chat and email), our framework is able to find staffing schedules that are between 7.5% and 72.5% better than simulation optimization. Moreover, our local search algorithm is developed to work similarly to how contact center managers manually optimize staffing schedules, thus providing locally optimal schedules and removes the need for managers to perform additional checks.

4.1.2 Related literature

Call centers. Telephone call centers (we refer back to "call centers" instead of "contact centers" to maintain consistency with existing literature) are prominent applications of research in the operations management field. Two popular review papers and countless research papers were written in the past twenty years, and the term "call center" has appeared in over 350 INFORMS publications from 1998–2018. Gans et al. (2003) review the literature on the operations and management of call centers, including forecasting demand, routing of calls, and personnel planning. In a more recent review paper, Aksin et al. (2007) discuss contemporary issues such as the design of multi-skill call centers, outsourcing, and service contracting, and interfaces with marketing.

The focus of this chapter is on the staffing and scheduling problems for multi-skill contact centers, thus we review the existing literature on this problem and

refer the reader to Gans et al. (2003) for work on single-skill contact centers or other related problems such as demand forecasting, agent rostering, and call routing. An effective staffing method for multi-skill call centers was proposed in Pot et al. (2008). In their paper, evaluation under different configurations of multi-skill agents for a single interval is approximated quickly based on the block queuing model. The shift scheduling problem is solved later in Bhulai et al. (2008), and Bhulai and Koole (2003) study how to balance emails and calls. There are many papers studying the shift scheduling problem and most of them are based on the classical set-covering problem proposed by Dantzig in 1954. A hybrid method that combines scheduling heuristics with simulation to simultaneously solve both the scheduling and the rostering problem is proposed in Fukunaga et al. (2002). Instead of considering the objective function for each interval, Koole and Van Der Sluis (2003) develop a scheduling methodology that meets only an overall service level objective over the entire scheduling period.

However, the literature in this area has mostly focused on either determining the staffing requirements or solving the scheduling problem separately. Integrally solving the staffing and shift scheduling problem has only been studied recently as discussed in the earlier part of this paper.

Machine learning for simulation. The use of models to model simulations for faster execution is a well-studied topic called "simulation metamodeling" (Barton 2015). The term was popularized in Kleijnen (1975), and a number of methods have been developed, such as polynomial regression, splines, and radial basis functions (Barton and Meckesheimer 2006). Overlapping with the machine learning perspective, Sabuncuoglu and Touhami (2002) experimented with neural networks for metamodeling. A particularly successful approach was developed by Ankenman et al. (2010), using stochastic kriging (also known as Gaussian processes) to model simulations of queuing systems. Kriging is inherently non-linear and has the flexibility to model complex response functions. Unfortunately it is computationally expensive—scaling cubically with the number of samples—and is inappropriate for problems with high throughput.

The development of deep learning has also led to powerful simulation metamodels in the natural sciences. Tompson et al. (2016) use deep learning to obtain fast and realistic simulations of Navier-Stokes equations for fluid flow. Pang et al. (2018) use deep convolutional neural networks for relativistic hydrodynamic simulations of heavy ion collisions in quantum physics. Finally, Gastegger et al. (2017) apply machine learning to speed up molecular dynamics simulations in quantum chemistry. Our approach uses a different state-of-the-art machine learning method—gradient boosted decision trees. Details of our method are discussed later.

Outline. The remainder of this paper is organized as follows. In Section 4.2 we provide a description of contact center operations and a mathematical formulation of our problem. In Section 4.3, we explain the simulation setup that is used to generate data for training and testing our machine learning framework. Section 4.4 describes our machine learning framework, including the algorithm used and corresponding feature construction. Section 4.5 presents the QoS prediction performance of our machine learning framework. Finally, we describe our optimization procedure and show the results of our numerical experiments in Section 4.6, and conclude in Section 4.7.

4.2 Problem description

We first provide a high-level description of the operation of inbound contact centers. Thereafter, we formulate the integral staffing and scheduling problem mathematically as an optimization problem.

4.2.1 High level description of contact centers

A contact center consists mostly of agents, whose major responsibility is handling customer requests from different channels (e.g., telephone, chat, email). Customers are categorized to different service types according to their requirements, and each service type corresponds to a specific skill that agents possess. For many contact centers the most common type of skill is language (e.g., English, Spanish, French, etc), and customers will select the language of the agent they wish to speak to. Multi-skill agents are able to handle more than one service type, so in the language example they would be multi-lingual. Often times multi-skill agents exhibit different levels of proficiency among their skills, so there can be different priorities in the service types that the agent will handle. An agent group is a set of agents who possess the same skill set.

When a customer calls the contact center, various routing technologies can direct the call to an available agent (if any) who has the skill to handle this call. However, often there are no agents available and the customer needs to hold the call and wait in a queue. The waiting time is usually unknown in advance and the customer may abandon the queue at any time without receiving service. The length of time a customer is willing to wait before abandonment is known as *patience*, which is usually no more than a few minutes. Once connected to an agent, the customer will be served until his/her problem is resolved and the time it takes to do so is called *handling time*. While the call is being handled, an interruption by another call with higher priority may arrive. This diverts the agent to the interrupting call, forcing

the previous customer to hold until the agent has finished handling the interrupting call. However, this situation is not preferred/allowed by most companies since it can lead to long handling times or abandonment. When the call is completed, the agent becomes available again and will handle the next routed call. The *waiting time* of a customer is the time spent in the queue until he/she is connected to an agent, and it plays a major role in the QoS indicators.

When a chat request arrives, again it will be routed to an agent who has the corresponding skill. However, differing from calls, multiple chats can be handled in parallel (up to a maximum number) by a single agent. This is because customers also need time to reply and this time can be used by the agent to respond to other customers. Therefore, a chat can be handled by an agent as long as his/her maximum level of concurrency has not been reached. Although chats can be more efficient, the handling time of ongoing chats can become longer when a new chat request is accepted. In practice, the maximum number of parallel chats is limited to a number (e.g., 2 or 3) such that the service time distributions will not be negatively affected by the level of concurrency.

Emails, in contrast to the other two channels, are not necessarily answered in real time since customers cannot abandon. We can consider the patience of an email customer to be very long, thus the average waiting time of emails is much longer than calls and chats. As a result, emails tend to have lower priorities than the other channels and are often interrupted. When emails are not handled within the day of arrival, *backlogs* are generated, which is an important measurement of email QoS. Additionally, the waiting time of an email customer differs from phone and chat customers in that it also includes the handling time since they must wait until receiving the reply.

All customers share the same feature that arrivals are random and arrival rates are time-varying. For most purposes, the arrival rate of each service type is usually considered to be constant within a time interval (e.g., 15 or 30 minutes). On the other hand, agents are working (and paid) by shifts instead of intervals. A common shift type over a day is 8 hours long with a 30 minute break in between. Different shift types according to the length, start/finish time, and other variables usually have different costs. Multi-skill agents are also paid more than single-skill agents.

4.2.2 Mathematical formulation

We consider a contact center where arriving customers are categorized into I types of service: $\{1, \ldots, I\}$. These customers can be calls, emails or chats that are served by agents divided into G groups: $\{1, \ldots, G\}$. Agents within the same group have the same skill set, which gives the subset of service types that this agent can serve. To

distinguish the costs of agent groups, we introduce c_g. Finally, there are K different shifts: $\{1, \ldots, K\}$ with the corresponding costs c_k. The schedule can be represented easily by the decision variable $n_{g,k}$, which is the number of agents staffed to group g, and shift k.

We do not assume any particular arrival process, handling time distribution, or patience distribution, instead we only need to be able to simulate them. In practice, the arrival rates are normally derived from the forecast results, and redials and reconnects can also be included. We only assume that service within the same type will be served on a first-come-first-served (FCFS) basis.

In the multi-skill multi-class environment, the choice of routing rules is also important for achieving good system performance. Again, we do not restrict our model to a specific routing policy. In the built simulation model, on the basis of the longest-idle-agent-first policy, static priority is used and preemption is allowed for some queues: once an agent starts serving a customer, there can be an interruption by other calls with higher priorities.

As mentioned in the previous section, the objective of our optimization problem is reaching QoS targets at minimum costs. We choose service level (SL) as the QoS measurement. It is defined as the proportion of customers who wait less than a given time threshold among all arrived customers over a time period. The given time threshold is also known as the acceptable waiting time (AWT). In practice, managers often pay attention to daily or weekly SLs and set a target to each service type or sometimes a set of service types. All of them are considered in our problem. The QoS target is called service level agreement (SLA), and it is the minimum SL a service type (or set) needs to meet. For emails, an additional QoS target related to the backlog is considered. The backlog suggests the maximum percentage of emails that can be transferred to the next day (or week). Other QoS measurements such as abandonment and occupancy can also be evaluated.

The objective function is composed of two parts: the penalty cost caused by breaching the QoS targets C_{QoS} and the agent cost C_{agent}. It can be written as:

$$C_{\text{overall}} = C_{\text{QoS}} + C_{\text{agent}}, \tag{1}$$

$$C_{\text{QoS}} = w_1 C_{\text{SLA}^{\text{week}}} + w_2 C_{\text{SLA}^{\text{day}}} + w_3 C_{\text{backlog}}, \tag{2}$$

$$C_{\text{SLA}^{\text{week}}} = \sum_{i=1}^{I} \mathbb{E}(\max\{\text{SLA}_i^{\text{week}} - SL_i, 0\}) + \sum_{s=1}^{S} \mathbb{E}(\max\{\text{SLA}_s^{\text{week}} - SL_s, 0\}), \tag{3}$$

$$C_{\text{agent}} = \sum_{g=1}^{G} \sum_{k=1}^{K} c_g c_k n_{g,k}, \tag{4}$$

where w_1, w_2 and w_3 are weights to adjust the importance between different targets and s in Equation (3) represents the index of the set of service types. $C_{\text{SLA}^{\text{day}}}$ and C_{backlog} can be calculated in a similar way as $C_{\text{SLA}^{\text{week}}}$, therefore we do not write down all the details for simplicity. Instead of setting the QoS targets as hard constraints, we put it into the objective function as penalty costs. In this way, the local search algorithm is easier to develop, and it helps to guarantee that a solution can be found. Note that the weight of C_{agent} is set to 1 as a reference. In order to ensure that the QoS targets are met, the weights in Equation (2) should be set much higher than the agent costs. Finally, the optimization problem can be modeled as

$$\begin{aligned} \min \quad & C_{\text{QoS}} + C_{\text{agent}} \\ \text{s.t.} \quad & n_{g,k} \in \mathbb{N}, \quad \forall k \in \{1, \ldots, K\}, g \in \{1, \ldots, G\} \end{aligned} \tag{5}$$

4.3 Simulation setup

The operation of a contact center is modeled via a discrete-event simulation, where the system can be described by a discrete sequence of events in time. The change of system states is triggered by events, but different states also affect the time at which an event happens. The input of the simulation model is the setting of the contact center, and for a given staffing schedule the QoS is evaluated over a predefined period of time. This period can be either a day or a week, divided into a number of intervals. Contact centers in practice often prefer a weekly schedule due to the use of the weekly shift patterns.

In the simulation model, three lists are constructed representing events, working agents, and queues. All events are stored in the event list sequentially in time. We first generate all customer arrivals according to the given arrival distribution per interval. The end of an interval is also considered as an event and is inserted in the list. When an arrival event happens, the simulation checks whether an agent with the right skill is idle in the current agent list. If an agent is available, the arrival event (i.e., customer) is handled and a departure event will be inserted into the event list after a random handling time. If no agents are available, the customer will be added to the queue and then an abandonment event will be inserted after a random amount of patience to the event list. A customer will leave the queue if he/she has not been served by the time the abandonment event occurs. When a departure event happens, the formerly occupied agent becomes idle and then, according to a given routing policy, the first customer from the selected queue will be removed from the queue and occupy the agent. Re-dials are generated when abandonment (departure) happens with given probabilities. During the simulation, all states related to QoS

are recorded, and at the end of a single simulation run, the average performance of each QoS is reported.

Similar simulation models that only consider calls have been built in other research. However, this is the first one to deal with a blend of calls, emails, and chats. A number of special points need to be treated differently when considering chat and email. When a chat agent finishes a chat while handling other chats, he/she will not become idle but search the queue to start a new chat (if any). If an agent can handle both chats and calls, calls can be handled only if the agent is idle. Regarding email, an initial backlog needs to be added to the queue at the beginning of the simulation to avoid cold start.

4.4 Machine learning approach

In order to optimize staffing schedules, we need a method that can approximate contact center service levels (SL) as accurately as possible given only information about the staffing schedule. To do so, we train a machine learning model on a sample of simulated results and use it to predict various service level metrics for each staffing schedule. The simulated results correspond to a specific scenario so in principle we train a machine learning model for each scenario. The method should be generalizable to different scenarios and be invariant to simulation details, thus we can only use information derived directly from the staffing schedule, which is defined by the number of agents of each skill group assigned to each shift.

Since SL is a continuous variable between 0 and 1, we model this as a regression problem and minimize the sum of squared error (SSE) between the true and predicted SL values of simulated samples during the training process. The general formulation of our problem can be written as

$$\arg\min_{f} \sum_{s=1}^{S} \left(f(n^s_{1,1}, n^s_{1,2}, ..., n^s_{1,k}, n^s_{2,1}, n^s_{2,2}, ..., n^s_{2,k}, ..., n^s_{G,1}, n^s_{G,2}, ..., n^s_{G,K}) - SL^s_{true} \right)^2 \tag{6}$$

where S is the total number of simulated samples, and $n^s_{g,k}$ is the number of agents belonging to skill group g assigned to shift k. f is a function that produces a SL prediction for inputs of a given staffing schedule. An example of a function f that can minimize the SSE is least squares regression, where optimal weights for a linear combination of the staffing variables are computed. However, while we expect changes in SL to be monotonic with changes in the number of staff scheduled, least squares

regression is not a suitable function to approximate the contact center staffing problem because SL does not respond linearly to the staffing number.

We use Gradient Boosted Decision Trees (GBDT) for this problem (Friedman 2001). It is also known by other names such as Multiple Additive Regression Trees (MART), Gradient Boosting Machine (GBM) or Tree Boosting. They all refer to the same technique which applies a framework called *gradient boosting* that uses regression trees as base learners (Breiman et al. 1984). GBDT is one of the most powerful machine learning algorithms and has been used to win most of the recent predictive analytics competitions (Chen and Guestrin 2016).

4.4.1 Gradient boosted decision trees

GBDT works by starting with a naive prediction and then iteratively fitting regression trees on the residuals to obtain a linear combination of predictors for any regression (or classification) problem. A single regression tree works by finding partitions in the feature space (i.e., independent variables) such that the target (dependent) variable within the partition is as homogeneous as possible. A single value is derived to label all observations within that partition and it becomes the model's prediction for that partition. This value is often the mean, median, or mode of the target variable values for all observations within the partition. The measure of homogeneity is defined by a loss function (e.g., mean squared error for regression or Gini impurity for classification).

Building the optimal regression tree (i.e., determining the globally best possible partitions) for a given dataset is an intractable problem because all possible combinations of partitions must be evaluated to find the best tree. Instead, in practice regression trees are fit myopically, meaning that partitions are created one-by-one, and the best partition from any point in time is determined without considering the possibility of future partitions. As a result of this approach, a single regression tree is likely to poorly fit the data, and could exhibit large variations with small changes to the data. Gradient boosting alleviates the under-fitting problems of regression trees by iteratively fitting trees on the residual (i.e., error) of previous trees. This allows for greater focus on the difficult observations. Moreover, GBDT uses a method called *bagging*, which subsamples observations and features for fitting regression trees every iteration. This process leverages the large variations of myopically built regression trees to further improve prediction power.

For more details on GBDT we refer the reader to Friedman (2001) and Friedman et al. (2001). In this chapter, we use a fast and accurate implementation of GBDT called LightGBM, which is currently used in most solutions to data science challenges (Ke et al. 2017).

4.4.2 Feature construction

The predictive capabilities of any machine learning algorithm depend on the features that are provided to the algorithm. Equation (6) assumes that GBDT would use as features the number of agents of each skill group assigned to each shift. These features alone do not provide enough information to GBDT to yield sufficiently accurate SL predictions due to two weaknesses of regression trees that are not adequately addressed by gradient boosting. First, the splitting of regression trees cannot directly account for non-binary operations across features such as summation, subtraction, or division. For example, the total number of agents employed could be an important predictor of SL as an insufficient number of agents will result in low SL regardless of how the agents are assigned across skill groups and shifts. Compounding the first weakness, regression trees also cannot directly account for deeper meaning of subsets of features such as skill groups or shift types. A schedule with many agents assigned to one skill group while neglecting another skill group is also likely to result in low SL. The current formulation assumes that shifts within the same skill group are unrelated, limiting the effectiveness of GBDT.

To account for these weaknesses and help GBDT more accurately model the relationship between the staffing schedule and SL, we construct an additional five sets of features derived from staffing schedule information. First, to ensure that a staffing schedule has a sufficiently large number of agents, we include the feature *total agent count* which is the total number of agents in the schedule irrespective of group or shift. Agents belong to different groups (i.e., have different skills) and work different shifts (e.g., 9am to 5pm or 1am to 9am), therefore it is likely for staffing schedules that have too few agents for specific skills or shifts to have low SL. To help GBDT identify whether staffing schedules have sufficient capacity for specific skills and shifts we include two sets of features, *number of agents per agent group* and *number of agents per shift*. The feature set *number of agents per agent group* is constructed by summing together the agents belonging to the same group (e.g., English speaking or French speaking) across different shifts, so there are G new features added, where G is the number of agent groups. Alternatively, the feature set *number of agents per shift* is constructed by summing together the agents working the same shift (e.g., 9am to 5pm) across different agent groups, so there are K new features added, where K is the number of all possible shifts.

One problem with shifts is that it is difficult to directly identify when a combination of shifts leaves a particular time period poorly covered by a staffing schedule, resulting in poor SL. To solve this problem we discretize shifts into a combination of time intervals (of, e.g., 30 minutes) and compute the feature set *number of agents*

Table 1: Features constructed from staffing schedule information

Feature set name	Definition	Number of features
Number of agents per shift per group	$n_{g,k}$	$G \times K$
Total agent count	$\sum_{g \in G} \sum_{k \in K} n_{g,k}$	1
Number of agents per group	$\sum_{k \in K} n_{g,k}$	G
Number of agents per shift	$\sum_{g \in G} n_{g,k}$	K
Number of agents per interval	$\sum_{G \in G} n_{g,i}$	I
Number of agents per group per interval	$n_{g,i}$	$G \times I$

per interval. The number of features under *number of agents per interval* depends on the time period under consideration and the length of each shift. For example, if the length of each interval is 30 minutes and the time period under consideration is one week, then there are 336 features, one for each time interval. Finally, we also compute the *number of agents per group per interval* to further split by agent groups. This increases the number of features by the number of intervals multiplied by the number of agent groups.

Table 1 provides a summary of the complete feature set used in our GBDT model. One concern is the large number of features that will be constructed for problems of considerable size, potentially leading to significant overfitting of the training samples. Another concern is that the additionally constructed features are correlated with each other and the initial *number of agents per shift per group* due to monotonic relationships. Fortunately, unlike linear regression or neural networks, the tree generating process of GBDT is iterative and does not optimize the loss function for all features at once. During the GBDT tree generation process, a split is made for a single feature on a single value without considering effects of other features. This approach has the advantage of being more robust against the overfitting risk that arises from having large numbers of features. To further protect against overfitting in general, an appropriate train and validation framework should be used to select hyperparameters (e.g., learning rate, max tree depth, feature subsampling frequency) that maximizes generalizability.

4.5 Service level prediction performance

We train GBDT models on staffing schedules sampled from the simulation described in Section 4.3. Three scenarios are considered: single-skill over one day, multi-skill (only calls) with five agent groups over one week (seven days), and multi-skill (calls plus chat and email) with five agent groups over one week (seven days). For the single-skill scenario, we predict only one SL metric, for the multi-skill scenario with

only calls we predict seven SL metrics, one for each skill and two set-level metrics, and for the multi-skill scenario with chat and email, we predict nine SL metrics, one for each skill plus chat and email, and two set-level metrics. One key aspect of the multi-skill scenario is that the skill requirements differ systematically. All agent groups can serve customers requiring the first skill, but only one agent group can serve customers requiring the other four skills. Moreover, demand for the skills is also different. The first skill is the most popular, accounting for over ten times the demand for the other skills. Skills two, three, and four are similar and the fifth skill is the least popular, accounting for about five percent of the demand as the others. Finally, for the multi-skill scenario with calls plus chat and email, all agent groups can also serve customers via chat and email, but with the same priority as calls of the first skill.

4.5.1 Train and validation framework

An appropriate train and validation framework is required to evaluate model performance for any prediction problem. In our framework, the complete set of simulated staffing schedules (and outcomes) are randomly assigned to either a *training* or *validation* dataset. The training dataset is used to train or fit the GBDT model and the validation dataset is used as an out-of-sample test to evaluate the predictive performance of the model. It is generally good practice to have more samples in the training dataset so that the GBDT model can have more data to work with, but it is also important to have enough samples in the validation dataset to ensure that the performance scores generalize.

Sampling data from simulation. We simulate 40,731 sample outcomes for the single-skill scenario, 66,217 sample outcomes for the multi-skill scenario with only calls, and 90,000 sample outcomes for the multi-skill scenario with chat and email. To ensure that the SL distributions are usable for our machine learning framework, we generate simulation samples using the following procedure. First, two pairs of parameters must be decided: (U, L), which is the upper and lower bound of the total number of agents scheduled, respectively; and (u, l), which is the upper and lower bound, respectively, of the number of agents scheduled for each agent group and shift combination. We set $U = 200$, $L = 0$, $u = 100$, and $l = 0$ for the single skill scenario, and $U = 400$, $L = 0$, $u = 100$, and $l = 0$ for the multi-skill scenarios. Next, a random number of agent group and shift combinations are chosen to add agents, so the combinations that are not chosen have zero agents. The maximum value for this number is 50 for the multi-skill scenario and 10 for the single-skill scenario. Then, for each of the agent group and shift combinations that have been chosen to

add agents we randomly assign agents to a combination within the bounds of $[u, l]$. Finally, after all chosen combinations have been assigned agents, we check whether the total number of agents assigned is between $[U, L]$, if yes, the derived schedule is simulated and otherwise it is thrown away.

4.5.2 Prediction performance

Following Section 4.5.1, we randomly select a $2:1$ split of training and validation, where 66.7% (24,774 for the single-skill scenario, 44,145 for the multi-skill scenario with only calls, and 60,000 for the multi-skill scenario with chat and email) of the simulated samples are used to train the GBDT model and 33.3% (15,956 for the single-skill scenario, 22,073 for the multi-skill scenario with only calls, and 30,000 for the multi-skill scenario with chat and email) of the samples are used to evaluate prediction performance. Three different evaluation metrics—root mean squared error (RMSE), weighted mean absolute percent error (ωMAPE), and coefficient of determination (R^2)—are used to measure how closely the predicted SL values relate to the observed SL value from the simulation. The three evaluation metrics are as follows: let $\hat{SL}_s$ and SL_s denote the predicted and observed service levels for observation s, respectively, we compute the RMSE of the predictions for observations $\{1, ..., S\}$ as

$$RMSE := \sqrt{\frac{1}{S}\sum_{s=1}^{S}(SL_s - \hat{SL}_s)^2}.$$

RMSE is the most commonly used error metric for evaluating prediction performance for regression problems (i.e., when the target variable is numeric). To present the error in the form of a percent value, we also compute the ωMAPE as

$$\omega MAPE := 100 * \frac{\sum_{s=1}^{S}(\omega_s * |SL_s - \hat{SL}_s|)}{\sum_{s=1}^{S}(\omega_s * SL_s)}, \quad \omega_s = \frac{SL_s}{\sum_{r=1}^{S} SL_r}.$$

The weight ω_s is used weigh the samples to prevent errors made on samples with low SL from dominating the average. Finally, we compute the R^2 as

$$R^2 = \left(\frac{\sum_{s=1}^{S}(SL_s - \bar{SL})(\hat{SL}_s - \bar{\hat{SL}})}{\sqrt{\sum_{s=1}^{S}(SL_s - \bar{SL})^2}\sqrt{\sum_{s=1}^{S}(\hat{SL}_s - \bar{\hat{SL}})^2}}\right)^2, \quad \bar{SL} = \frac{1}{S}\sum_{r=1}^{S} SL_r, \quad \bar{\hat{SL}} = \frac{1}{S}\sum_{r=1}^{S} \hat{SL}_r.$$

Table 2 presents the prediction performance of GBDT for both the single- and multi-skill scenarios with only calls. As a baseline, we also include the prediction performance of a linear regression model estimated using the same data and features.

Table 2: Prediction performance on single-skill and multi-skill scenarios with only calls.

	s-SL	m-SL1	m-SL2	m-SL3	m-SL4	m-SL5	m-Set1	m-Set2
GBDT								
RMSE	0.019	0.013	0.017	0.018	0.018	0.055	0.013	0.014
ωMAPE(%)	1.9	1.1	2.2	2.2	2.3	5.5	1.1	1.9
R^2	0.993	0.998	0.998	0.998	0.998	0.981	0.998	0.998
Linear regression								
RMSE	0.132	0.143	0.197	0.203	0.202	0.239	0.143	0.170
ωMAPE(%)	15.8	14.2	23.8	24.1	24.0	25.6	14.2	22.0
R^2	0.694	0.704	0.698	0.690	0.691	0.650	0.704	0.703

Only a single SL value is reported for the single-skill scenario and five skill-level plus two set-level SLs are reported for the multi-skill scenario. Overall, GDBT predicts SL very well, with RMSE under 0.02 and R^2 over 0.99 for all measures with the exception of SL5 for the multi-skill scenario. ωMAPE is also fairly consistent with RMSE and R^2, suggesting that the predictions are not too skewed in favor of either the high or low SL samples. In comparison to GBDT, the linear regression predictions are far worse, with RMSE and ωMAPE almost an order of magnitude higher than the GBDT predictions. R^2 is also lower by around 30% on average. These results suggest linear models are likely to be poor approximations for SL, and that GBDT, given its high accuracy in predicting SL, could be a suitable alternative to simulations.

Another observation is that both models perform differently for the different skills in the multi-skill scenario. We believe this is due to the differences in demand for the different skills. Calls requiring the first skill has the largest volume and thus results in the most accurate SL predictions. Skills two, three, and four have similar volumes so prediction accuracy of their SL is similar as well. Finally, skill five has very low demand, likely resulting in large variations in SL and worse prediction performance. It should be noted that the prediction performance for m-Set1 is exactly the same as for m-SL1. This is because in this scenario, m-Set1 contains only m-SL1 and therefore they are the same. We keep this set as a performance metric because it plays its own role in calculating the objective for the ensuing optimization procedure. m-Set2 is a combination of m-SL2, m-SL3, m-SL4, and m-SL5. We believe prediction performance for m-Set2 is better than each of the individual SLs it comprises as a result of larger aggregated volume.

Next we investigate the effect of adding different features to the base prediction model for GBDT and linear regression. Table 3 shows how the prediction performance changes as we include new feature sets for GBDT and linear regression. We first start with the base prediction model specified in Equation (6), using only the *number of agents per shift per group* features (96 features for the single-skill scenario and

Table 3: Prediction performance by feature sets.

	s-SL	m-SL1	m-SL2	m-SL3	m-SL4	m-SL5	m-Set1	m-Set2
GBDT								
Number of agents per shift per group	0.077	0.102	0.056	0.059	0.057	0.106	0.102	0.050
+Total agent count	0.079	0.069	0.055	0.057	0.056	0.097	0.069	0.044
+Number of agents per group	NA	0.066	0.044	0.047	0.047	0.085	0.066	0.035
+Number of agents per shift	NA	0.045	0.047	0.049	0.049	0.085	0.045	0.037
+Number of agents per interval	0.019	0.012	0.047	0.049	0.049	0.084	0.012	0.036
+Number of agents per group per interval	NA	0.013	0.017	0.018	0.018	0.055	0.013	0.014
Linear regression								
Number of agents per shift per group	0.132	0.143	0.197	0.203	0.202	0.239	0.143	0.170
+All other feature sets	0.132	0.143	0.197	0.203	0.202	0.239	0.143	0.170

480 features for the multi-skill scenario). Then, we cumulatively add feature sets one-by-one and obtain the prediction performance for the two models. The feature sets are added in the following sequence: (1) *total agent count* (one feature for both single- and multi-skill scenarios); (2) *number of agents per group* (not applicable for the single-skill scenario and 5 features for the multi-skill scenario); (3) *number of agents per shift* (not applicable for the single-skill scenario and 96 features for the multi-skill scenario); (4) *number of agents per interval* (48 features for the single-skill scenario and 336 features for the multi-skill scenario); (5) *number of agents per group per interval* (not applicable for the single-skill scenario and 1,680 features for the multi-skill scenario)

The first thing that can be observed is that linear regression is unaffected by the inclusion of new features. Prediction performance remained constant with the inclusion of each feature set (when rounded to the third decimal), thus we collapsed them into a single row. This is likely due to the fact that our new features are mostly linear transformations of the baseline features, resulting in multicollinearity. GBDT on the other hand is greatly improved by the new features and its response to the inclusion of new features seems to depend on the specific SL type under consideration. For example, when we added *total agent count* to the base model we saw large improvements for m-SL1 and m-Set1, moderate improvements for m-SL5 and m-Set2, very small improvements for m-SL2, m-SL3, and m-SL4, and even a small decline for s-SL. Alternatively, inclusion of the final feature set, *number of agents per group per interval*, greatly improved m-SL2, m-SL3, m-SL4, m-SL5, and m-Set2, while causing slight declines in performance for m-SL1 and m-Set1. This suggests that different types of information are important for different skills. For consistency we use the same features to train models for the different SL types, but it is possible to use different features or even develop different models for each skill individually.

Table 4: Prediction performance on multi-skill scenario with chat and email.

	m-SL1	m-SL2	m-SL3	m-SL4	m-SL5	m-SLch	m-SLem	m-Set1	m-Set2
GBDT									
RMSE	0.014	0.023	0.024	0.024	0.067	0.034	0.019	0.013	0.018
ωMAPE(%)	1.3	2.8	2.9	3.0	6.6	4.2	2.8	1.4	2.3
R^2	0.997	0.994	0.994	0.994	0.962	0.989	0.995	0.998	0.996
Linear regression									
RMSE	0.102	0.095	0.098	0.098	0.134	0.110	0.079	0.083	0.070
ωMAPE(%)	10.4	11.9	12.1	12.1	14.8	14.0	11.7	9.6	9.6
R^2	0.866	0.900	0.897	0.897	0.848	0.879	0.916	0.904	0.937

Surprisingly, the inclusion of chat and email slightly decreases the prediction performance of GBDT and improves the performance of linear regression. However, the prediction performance of GBDT is still substantially better than linear regression. Table 4 presents the results. The same pattern remains when comparing prediction performance by skill. We can see that SL for skill one is fairly easy to predict, and for skill five is particularly difficult. SL for chat also seems difficult to predict. This may be due to the routing policy, where agents are assigned to serve customers via chat for certain periods of time only, resulting in service level fluctuations even when sufficient numbers of agents are present. Finally, we can also see that m-Set1 differs from m-SL1 because this set also contains chat and email.

4.6 Numerical experiments

In this section, we show how the trained GBDT model can be used to optimize staffing schedules. We present three numerical experiments, one for the single-skill scenario and one each for the multi-skill scenario with and without chat and email (see scenario definitions in Section 4.3). We first describe our optimization procedure in Section 4.6.1, then we present the results in Section 4.6.3 and compare our results against the SIPP and simulation optimized staffing schedules.

4.6.1 Optimization procedure

One disadvantage of GBDT is that there is no easy method to optimize a target variable from the trained model. Because GBDT is non-parametric and does not enforce any functional relationships between the features and the target variable, finding the optimal staffing schedule from a trained GBDT is a combinatorial optimization problem. We develop a simple algorithm (for details see Algorithm 1) based on local search to find near-optimal staffing schedules. This algorithm iterates over a staffing number (i.e., the number of agents), and performs local search over possible staffing

schedules for the given staffing number. Note that instead of performing local search from a single schedule, we search from N randomly sampled schedules to improve diversity.

For practical purposes, five heuristics are introduced to Algorithm 1. First, we terminate the local search procedure prematurely when it seems that the given staffing number is insufficient to meet the SLAs. This is because the effect of not meeting the SLAs far outweighs all other aspects of the objective, so staffing schedules that cannot meet the SLAs are almost guaranteed to score poorly on the objective. Second, since meeting the SLAs is so important, we increase the SLAs used in the objective function of the optimization algorithm by two percent. This is to account for the potential prediction error of GBDT and to ensure that the staffing schedules found by the optimization algorithm have sufficient likelihood of meeting the SLAs. The third heuristic is that instead of returning only the best schedule found by the algorithm, we return the final locally-searched schedule for each of the N schedules that are searched from. Fourth, we continue the algorithm for an additional three staffing number values after the stopping criteria has been met. The fifth and final heuristic is that we return the best schedules found from the last five staffing number values that were evaluated. Together, the third, fourth, and fifth heuristics ultimately result in a set of 50 schedules returned from the optimization procedure. The 50 schedules are then fed back into the simulation for a final performance verification, and the best staffing schedule from the verification is returned.

4.6.2 Mathematical approximation and simulation optimization approaches

To verify the effectiveness of our machine learning optimization framework we compare our solution against a mathematical approach—the stationary independent period-by-period (SIPP) model—for the single-skill scenario and against a simulation optimization approach for the multi-skill scenarios.

The SIPP model is based on the set-covering problem, proposed by Dantzig in 1954. In the single-class single-skill scenario, the result from the SIPP model is a very good approximation of the optimal solution, as the only error is that introduced by ignoring the transient effects between consecutive intervals. Using the same notation as in Section 4.2, the SIPP model for the integral staffing and shift scheduling problem can be written as

$$\min \quad \sum_{k=1}^{K} c_k n_k$$

Algorithm 1 Local search algorithm for finding near optimal staffing schedules

```
1: procedure localSearch(GBDT)
2:     initialize best_overall_score
3:     for staffing_number in 1 : inf do
4:         for n in 1 : N do
5:             staffing_schedule[n]← choice(shift_groups, staffing_number)
6:             staffing_schedule_SL[n]← predictSL(GBDT, staffing_schedule[n])
7:             staffing_schedule_score[n]← objective(staffing_schedule_SL[n])
8:         best_schedule[n]← staffing_schedule[n]
9:         best_schedule_score[n]← staffing_schedule_score[n]
10:        best_staffing_score← max(best_schedule_score)
11:
12:        while best_staffing_score not surpassed for 100 iterations do
13:            for n in 1:N do
14:                staffing_schedule_new[n]← swapShift(staffing_schedule[n])
15:                staffing_schedule_new_SL[n]← predictSL(GBDT, staffing_schedule_new[n])
16:                staffing_schedule_new_score[n]← objective(staffing_schedule_new_SL[n])
17:
18:                if staffing_schedule_new_score[n] better than staffing_schedule_score[n] then
19:                    staffing_schedule_score[n]← staffing_schedule_new_score[n]
20:                    staffing_schedule[n]← staffing_schedule_new[n]
21:                    best_schedule_score[n]← staffing_schedule_score[n]
22:            if max(best_schedule_score) better than best_staffing_score then
23:                best_staffing_score← max(best_schedule_score)
24:
25:        if max(best_staffing_score) better than best_overall_score then
26:            best_overall_score← max(best_staffing_score)
27:        else
28:            return best_schedule corresponding to the best_overall_score
29:
30: procedure choice(shift_groups, staffing_number)
31:     randomly assign the number of agents specified by staffing_number to shift_groups
32:
33: procedure swapShift(staffing_schedule)
34:     randomly swap one agent in the staffing_schedule from one shift group to another
35:
36: procedure predictSL(GBDT, staffing_schedule)
37:     use the given GBDT to predict the SL for the given staffing_schedule
38:
39: procedure objective(staffing_schedule_SL)
40:     compute the objective value for a staffing schedule based on the given staffing_schedule_SL
41:     see Equation 5 for the mathematical formulation
```

$$
\begin{aligned}
\text{s.t.} \quad & a_t = \sum_{k=1}^{K} x_{k,t} n_k, && \forall t \in \{1,\dots,T\} \\
& \sum_{t=1}^{T} \lambda_t \mathrm{SL}_t(a_t) / \sum_{t=1}^{T} \lambda_t \geq \mathrm{SLA}^T \\
& n_k \in \mathbb{N}. && \forall k \in \{1,\dots,K\}
\end{aligned}
$$

Since there is only one service type with a single-skill agent group, the decision variable $n_{k,g}$ is simplified to n_k, and the binary parameter $x_{k,t}$ equals to 1 if shift k works in interval t. The main constraint guarantees that the SLA over T intervals is met by the schedule n_k, where a_t is the number of agents scheduled in interval t. However, this model is non-linear because the main constraint is non-linear. A formulation with only linear constraints is needed to solve the problem:

$$
\begin{aligned}
\min \quad & \sum_{k=1}^{K} c_k n_k \\
\text{s.t.} \quad & \sum_{k=1}^{K} x_{k,t} n_k = \sum_{m=1}^{M} y_{m,t} m, && \forall t \in \{1,\dots,T\} \\
& \sum_{m=1}^{M} \sum_{t=1}^{T} y_{m,t} \mathrm{SL}_{m,t} \geq \mathrm{SLA}^T (\sum_{t=1}^{T} \lambda_t) \\
& \sum_{m=1}^{M} y_{m,t} = 1, && \forall t \in \{1,\dots,T\} \\
& n_k \in \mathbb{N}, && \forall k \in \{1,\dots,K\} \\
& y_{m,t} \in \{0,1\}, && \forall m \in \{1,\dots,M\}, \forall t \in \{1,\dots,T\}
\end{aligned}
$$

where a binary auxiliary variable $y_{m,t}$ is added. It equals to 1 if m agents are scheduled in interval t. The basic idea of the formulation is calculating beforehand the SL of each interval for all $m \in \{1,\dots,M\}$: $\mathrm{SL}_{m,t}$. For this single-skill scenario, $\mathrm{SL}_{m,t}$ can be easily calculated via Erlang formulas. There is a good excel add-on offering various Erlang formulas with regard to different scenarios and can be downloaded via `http://software.ccmath.com/erlang/ErlangCCmath.xla`. Since Erlang C does not take abandonment into consideration, we use Erlang A here.

For the multi-skill setting the SIPP model becomes inappropriate and simulation optimization is used to compare against our approach. Simulation optimization works by intelligently exploring different decision variable values to eventually find the best results for a problem. For the integral staffing and shift scheduling problem, the decision variables are $n_{k,g}$ and the result is the objective in Equation (5). Given the complexity of this problem, we use a heuristic based simulation optimization method in the following way:

1. a max flow problem is solved to obtain an initial schedule;
2. a heuristic-based algorithm is designed to add agents to one of the agent group/shift combinations until either all SLAs are satisfied or the marginal cost for adding new agents is too high.

The simulation optimization approach is reliable but, unfortunately, it requires a large number of iterations to find good schedules. It is important for contact center managers to get good staffing schedules in real time with minimal wait times.

4.6.3 Optimization results

Recall from Equation (5) that our objective is to minimize the cost of failing to meet the SLAs plus the cost of agents. In our experiment, we use SLAs of 0.68 for each individual skill, 0.65 for chat, 0.80 for email, and 0.74 for the sets (see Equation (3)). The cost of agents is 1 unit for the first group and 1.5 units for each of the other four groups. We set w_1 to be 25 and only consider the SLA at the week level, meaning that w_2 and w_3 are both set to zero (see Equation (2)). The parameters for the single-skill objective are equivalent to the multi-skill case with the exception that there is only one SLA for the first skill (no set SLA) and the cost is also 1 unit for all agents.

Single-skill scenario. Using the SIPP model we are able to find a staffing schedule with the objective score of 52. This schedule has an SL of 0.71 and employs 52 agents. Since the SLA was 0.68, the schedule has likely over-staffed as its SL was 0.03 higher than the agreement. Using a single run of our machine learning framework, we are able to find 9 staffing schedules that are at least as good as the schedule found by the SIPP model. Of the 9 schedules, 6 are better than the schedule found by the SIPP model and the best schedule has an objective score of 50. This schedule has an SL of 0.68—exactly meeting the SLA—and employs 50 agents. This result improves upon the SIPP model by 3.8% and the runtime was approximately 50 seconds.

Multi-skill (calls only) scenario. To get a representative set of staffing schedules from the simulation optimization approach, we ran the procedure 100 times to obtain 100 different schedules. On average each run took three minutes, with the objective scores ranging from 104.5 to 140.2 (mean = 109.9 and median = 108.3). The best schedule employs 73 agents and in all of the schedules, there were fewer agents of the first group than the other four groups even though they cost 0.5 units less. Using a single run of our machine learning framework (runtime of approximately 2 minutes), we are able to find 11 staffing schedules that are at least as good the best schedule found by simulation optimization, and 18 schedules that are better than the median

schedule found by simulation optimization. The best schedule found by our machine learning framework has an objective score of 100 and employs a total of 71 agents. Given that an agent costs at most 1.5 units, the machine learning framework is able to reduce costs not only through employing fewer agents but also by shifting agents from the more expensive group to the cheaper group. Specifically, the best schedules found via machine learning recognizes that the fifth skill is of low demand and therefore employs fewer agents with that skill in favor of the first group which is 0.5 units cheaper. This is an expected feature of the machine learning framework.

Multi-skill (calls plus chat and email) scenario. Similar to the calls-only scenario, again we ran the simulation optimization procedure 100 times to obtain 100 different schedules. On average each run took three minutes, with the objective scores ranging from 129 to 207.2 (mean = 140.5 and median = 135.5). The best schedule employs 96 agents which is reasonable because the increase in workload from chat and email customers is already equivalent to 21 agents when compared to the scenario with only calls. Using a single run of our machine learning framework (runtime of approximately 2 minutes), we are able to find 21 staffing schedules that are at least as good as the best schedule found by simulation optimization, and 24 schedules that are better than the median schedule found by simulation optimization. The best schedule found by our machine learning framework has an objective score of 120 and employs a total of 92 agents. We can see that the inclusion of chat and email substantially increases the variance of simulation optimization results, decreasing stability. On the contrary, our machine learning framework still performs very well and is able to find a large number of high-performing staffing schedules.

4.7 Conclusion

In this chapter, we show how machine learning can be used in combination with simulation to solve the integral staffing and scheduling problem for multi-skill contact centers. A machine learning model is trained on data samples generated via simulation, and is then used to predict deterministic quality of service values for all possible staffing schedules. Subsequently, an optimization procedure based on local search is used to find near-optimal staffing schedules for a given objective with a variety of QoS targets. We test our machine learning framework using simulation parameters derived from a real-life contact center scenario and find that the predictive performance of QoS metrics is excellent. Subsequently, we compare the best staffing schedules found via our optimization procedure against the best staffing schedule

found by the stationary independent period-by-period model in the single-skill scenario, and against a simulation optimization approach in two multi-skill scenarios. The results show that our method outperforms both existing approaches. For the single-skill scenario, it finds a staffing schedule that is 3.8% better than the best schedule found by the stationary independent period-by-period model. For the multi-skill (with only calls) scenario, it finds a staffing schedule that is 4.3% better than the best schedule found via 100 simulation optimization runs, and is 7.7% better than the median schedule of the 100 runs. Moreover, upon closer examination, we find that the machine learning framework is able to realize the low demand for one of the skills in the multi-skill scenario, and thus shift agents from that skill to a highly demanded skill. This strategy was not observed from the schedules found by simulation optimization. Finally, for the multi-skill (with calls plus chat and email) scenario, it finds a staffing schedule that is 7.5% better than the best schedule found via 100 simulation optimization runs, and is 12.9% better than the median schedule of the 100 runs.

Chapter 5

Multi-channel Conversion Attribution: A Machine Learning Approach

Chapter Abstract

This chapter presents a novel machine learning approach to the problem of attributing online conversion credit. By incorporating customer behavior information that is highly effective in predicting whether a customer journey will result in a conversion, this approach achieves conversion prediction quality that significantly exceeds existing attribution models. Conversion credits are then assigned to different marketing channels based on their relative contribution as defined by the Shapley value framework, which is proven to ensure fairness and is easy to interpret. Our approach also allows for attribution at the individual journey level instead of in aggregate at the channel level. This provides practitioners with the ability to observe the contribution of each individual touchpoint, allowing for more precise measurements of return on investment of advertising by linking with the exact cost of the touchpoint. We test our method on a real-life dataset of customer journeys for an online travel agency and compare its attribution outcomes to four existing attribution methods. Our findings suggest that the inclusion of customer behavior information greatly affects attribution outcomes, and that the intuition supporting rule-based attribution methods are reasonable, but indeed overlooks important information.

5.1 Introduction

5.1.1 Background and motivation

The last few years have seen an enormous rise in the popularity of data-driven conversion attribution (CA) models. This is no wonder considering the need for insights into the effectiveness of individual digital marketing channels and campaigns. Advertising spend in online advertising channels, such as search engine advertising, display advertising, and social media advertising has become increasingly popular, finally surpassing traditional offline spending (Recode 2017). With this increase, the call for greater justification of digital advertising effectiveness is also apparent.

Digital advertising is naturally more measurable than offline advertising. Detailed customer-level data is usually available on the number of views (impressions) and clicks on an advertisement and even the number of conversions (e.g., online purchases) following a view or click. This allows marketers to gain detailed insights regarding the effectiveness of digital advertising campaigns.

The insight in effectiveness is hampered by the fact that an individual consumer often encounters multiple digital advertisements of the same advertiser and pays multiple visits to the advertiser's website before making a purchase. Suppose Janice is looking to buy a new camera. In the process, she might visit the same e-commerce website a few times before buying a camera (on that particular website or elsewhere). First, she might search Google for 'digital cameras', and one of the *search engine advertisements* shown on top of the search results leads her to the e-commerce website. She then browses around to get familiar with the range of cameras they sell but is not ready to buy yet. She also browses some other websites and compiles a short list of cameras she is interested in. After a second Google search for a specific type of camera, again a search engine advertisement leads her to the e-commerce website. She takes note that this website sells the camera for an attractive price, but still is not ready to make the purchase yet. Subsequently, the e-commerce company starts targeting her with display banners because they have noticed that she seems to be interested. So when she visits the New York Times website a few days later, she sees a display banner from the e-commerce site for the camera she was interested in earlier. This reminds Janice that she still wanted to buy the camera, so she clicks on the banner ad and purchases the camera on the e-commerce website. From the perspective of the e-commerce site, Janice's *customer journey* consisted of 3 *touchpoints*:

1. A website visit initiated by the Google search ad on the broad search term ('digital cameras')
2. A website visit initiated by the Google search ad on the narrow search term (specific type of camera)
3. A website visit initiated by the display advertisement on the New York Times website. This website visit included the purchase of a camera (conversion).

To establish the effectiveness of the different advertisements, it is essential to know what each of their contribution towards the eventual purchase decision is. Would Janice have made the purchase if she did not see the first advertisement? Or the last? How much of the credit should we assign to each of the three? The e-commerce website has to pay for all three advertisements so they need to answer this question to establish the *return on marketing investment* (ROMI) and *cost per conversion* (CPA) of these advertisements. These and similar KPIs are then used as inputs to

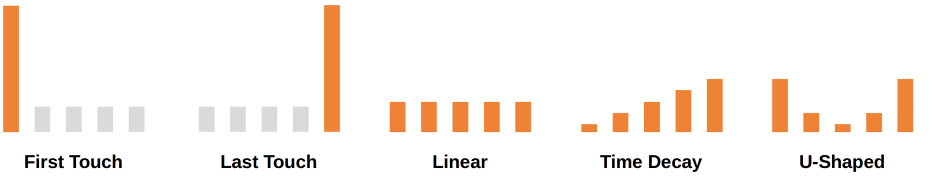


Figure 1: Five rule-based conversion attribution methods.

make decisions on future investments and allocation of the advertising spend that maximize return. Should spend on display banners be increased? Should the search advertising budget be reallocated from narrow search terms to broad search terms?

The problem of determining the contribution of individual advertisements towards a conversion is known as conversion attribution (or multi-touch attribution). This contribution cannot be directly observed from the customer journey unless the journey only has one touchpoint. All we know is which touchpoints were included in each customer journey and whether the journey resulted in a conversion or not.

There are multiple methods to perform conversion attribution. We can distinguish between assumption-based models and data-driven models. In an assumption-based (or rule-based) model, attribution is performed based on a predefined rule. The most commonly used method thus far in the industry is the *Last Touch* attribution rule (LTA), where 100% of the credit is given to the last touchpoint of the customer journey. LTA, because of its simplicity, has been the industry standard for a long time. In our example customer journey, the display banner would receive all the credit for the conversion based on this method. Figure 1 provides an overview of some well-known assumption-based attribution methods. *First Touch* and *Last Touch* attribution rules award 100% of the credit to the first and last touchpoints in the customer journey, respectively. The *Linear* attribution rule awards equal credit to all touchpoints in the journey regardless of their position. The *Time Decay* rule awards credit based on recency, with the more recent touchpoints receiving more credit, and finally the *U-Shaped* rule assumes that the first and last touchpoints are both equally important while the other touchpoints contribute less.

The problem with assumption-based attribution models is, naturally, that they are based on assumptions. It is not known nor can it be verified whether these assumptions are correct, and some of the methods have very basic assumptions that are not realistic. Specifically, LTA is a poor method because: 1) it disregards all other touchpoints in the journey, assuming that they add no value towards a conversion and 2) it leads to free-riding or customer exploitation (Berman 2018). Because spending budgets and allocation by advertisers are often determined by the amount of conversion credit that a channel accounts for, ad suppliers are incentivized

to be the last touchpoint in a customer journey more so than to affect conversion when LTA is the attribution method.

In recent years, data-driven conversion attribution models have become popular in marketing research and are increasingly adopted by practitioners. The primary goal is to create an attribution model that can accurately explain the contribution of channels, based on the observed data (customer journeys) instead of on assumptions. Different modeling approaches have been proposed, such as regression-based models (Shao and Li 2011, Nottorf 2014, Danaher and van Heerde 2018), Markov models (Abhishek et al. 2015, Anderl et al. 2016), and Shapley value-based models (Dalessandro et al. 2012, Li and Kannan 2014). In Section 5.2 these type of models will be discussed in more detail.

5.1.2 Shapely value-based attribution

Among the existing data-driven attribution models, the Shapley value attribution model (SVA) has become widely adopted in practice, primarily due to Google's offering of it in its Google Analytics 360 software (Google 2018). Nevertheless, SVA offers three highly attractive advantages that differentiate itself from other models. First, it is characterized by axioms to ensure fairness and is based on the causal attribution framework (Dalessandro et al. 2012). The Shapley value is a concept from game theory, providing a method to assign an average value to players of a game representing their contribution. When applied to the attribution problem, the players are channels in a customer journey, and SVA estimates the marginal contribution of each channel by computing the conversion rates of the journey with the channel to the journey without the channel. This means that if a channel positively contributes toward conversions, directly or indirectly (e.g., via spill-over or carry-over effects), regardless of its position in the customer journey, SVA is able to appropriately estimate its contribution. The second attractive property of SVA is its ability to estimate heterogeneous contributions of channels for different customer journeys, accounting for the possibility that channels may affect conversion behavior differently under different conditions (e.g., affiliate channels may be more effective in longer journeys where it is the last touchpoint). On the contrary, statistical models that rely on regression or Markov models can only estimate channels' contributions for a population of customer journeys, assuming that a channel's contribution towards conversion is constant across all customer journeys. Finally, the third, but perhaps most attractive property of SVA is its relative ease to implement and evaluate in practice. A number of the other data-driven models require complicated estimation techniques, whereas SVA only requires comparisons of empirical conversion rates of customer

journeys, resulting in simple estimation procedures that can also be parallelized for computational efficiency.

Although SVA is well-received by practitioners, two disadvantages result from its reliance on empirical conversion rates of customer journeys:

1. Empirical conversion rates are not reliable when based on a small number of observations. This often is the case in conversion attribution, because usually we have a large number of unique customer journeys (unique sequence/set of channels), of which most are only observed a few times. This effect is further amplified by the fact that conversion rates are typically low (e.g. 1%), which increases the number of observations required to make a reliable estimate.
2. Empirical conversion rates take into account only the number of conversions compared to the number of observations for each customer journey. They do not consider underlying factors that influence (and/or explain) the observed conversion rates, such as properties of the website visits (e.g., time on site, pages visited), promotions, prices and external factors (weather, competition etc).

While the first disadvantage is unique to SVA, the second disadvantage is common across all existing data-driven attribution models.

To illustrate the inadequacy of attribution models that only consider touchpoints in customer journeys, Figure 2 presents two example journeys, Journey A and Journey B, both of which contain the exact same touchpoints and resulted in conversions. Both journeys started with the customer clicking on a search ad, followed by a visit via a display banner click, and finally a direct visit that culminated in a conversion. However, we can see that the customers' behaviors in the two journeys differ. In Journey A, the customer browsed 6 pages for 15 minutes during the first visit, then only browsed 1 page for 20 seconds during the second visit, and finally browsed 17 pages for 34 minutes in the third visit. Based on the customer's behavior, we can posit that the last visit likely contributed the most to the conversion. On the contrary, in Journey B, the customer browsed 2 pages for 2 minutes during the first visit, browsed 22 pages for 41 minutes during the second visit, and browsed 3 pages for 2 minutes in the third visit. We posit that for this customer it was likely the second visit that contributed the most to the conversion. Under existing attribution models, the three channels would be awarded the same amount of conversion credit for Journey A as Journey B, based on the effectiveness of the path and sub-paths of touchpoints in both journeys across all customer journeys of the website, without considering individual differences in behavior across consumers.

5.1.3 Contribution

In this chapater, we extend the Shapley value attribution model by substituting empirical conversion rates of customer journeys with predicted conversion rates made

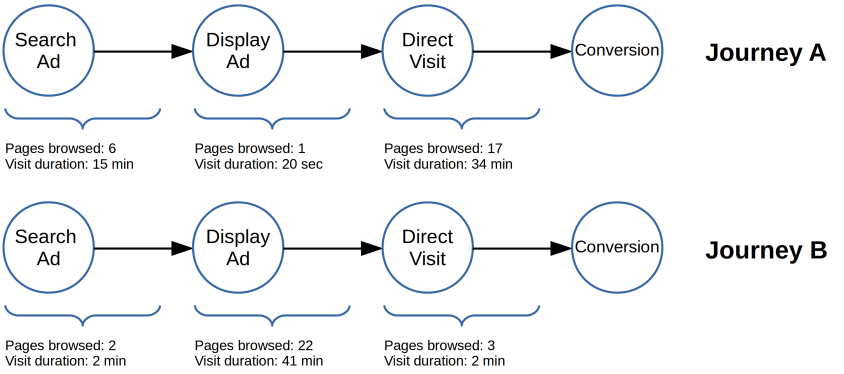


Figure 2: Two example customer journeys.

via a state-of-the-art supervised machine learning prediction model. The machine learning model is able to automatically extract predictive characteristics (features) from customer journeys under a general formulation, thereby providing confident predictions of conversion rates even for rare-occurring journeys. Moreover, it allows for the inclusion of customer behavior information such as visit duration or the number of pages browsed within a visit. In sum, our approach eliminates the disadvantages of SVA while still maintaining its three attractive properties: fairness, able to estimate heterogeneous contributions of channels, and ease of implementation.

Using data provided by a mid-sized online travel agency, we show that by incorporating information about customer behavior during a visit into our machine learning model, we are able to increase the quality of conversion rate predictions on out-of-sample customer journeys by 15% as compared to both the empirical conversion rates based on SVA and the predicted conversion rates based on the Markov chain attribution model. We find that customer behavior during website visits, namely the number of pages browsed and visit duration, is far more predictive of conversions than other available information. The resulting attribution values of our method differs from SVA by 31% across all customer journeys within a six-month period, and by 40% for customer journeys with at least two unique channels.

Our method allows us to investigate the shortcomings of SVA, namely the attribution distortion effects from rare-occurring customer journey compositions and the negligence of customer behavior information. We find that over 20% of the converted journeys in our dataset have a sub-journey that is relatively rare-occurring, and for these journeys the average attribution difference between SVA and our proposed machine learning-based attribution method is significantly larger than others, suggesting a strong effect of rare-occurring journey compositions. Regarding the negligence of customer behavior information, well over 40% of converting customer

journeys contained a visit that was at least 90^{th} percentile in visit duration or pages viewed among visits by all customers. These journeys also had considerably different attributions between the two methods, albeit not as pronounced as the effect of rare-occurring sub-journeys.

Finally, using our method we can test the soundness of the rule-based attribution models. By comparing the estimated individual touchpoint effects of the first, middle, and last touchpoints in converting customer journeys, we show that first touch, last touch, and all touch attribution are all reasonable to varying degrees. The last touchpoint in the converting customer journeys has a strong effect overall, and the first touchpoint has a slightly stronger effect than the average of the middle touchpoints. However, the sum of effects from all middle touchpoints is stronger than even the last touchpoint. This suggests that a substantial amount of information is omitted by each of the rule-based attribution models.

5.1.4 Outline

The remainder of this chapter is organized as follows. In Section 5.2 we provide an overview of existing data-driven attribution models. Next, in Section 5.3 we explain the Shapley value attribution framework, and show how it can be extended in combination with a machine learning prediction model to incorporate customer behavior information during website visits. Section 5.4 presents the dataset used to develop, implement, and evaluate our attribution framework. In Section 5.5, we describe how machine learning, namely Gradient Boosted Decision Trees is used to predict conversion probability of customer journeys. Section 5.6 presents the results of our attribution approach, together with a comparison against the results of some other attribution methods. Finally, Section 5.7 concludes this paper.

5.2 Overview of existing data-driven attribution methods

In the literature, many methods to solve the attribution problem exist. This section provides an overview of data-driven conversion attribution models. We analyze various papers in the literature that have focused on the attribution problem, and discuss their performance, data usage, and disadvantages. The approaches can generally be categorized into four groups: probabilistic models, Markov models, point processes, and vector autoregression models. Most of these approaches focus on a specific part of attribution such as spill-over, carry-over and dynamic effects, and consumer states. They try to analyze the incremental impact of an advertisement directly, which is

problematic due to the absence of validation. Only complete customer journeys have some form of validation in the form of a conversion. Our method differs from the existing papers by not assuming specific structures of conversion behavior, and relies only on the data and out-of-sample conversion predictions to guide attribution.

5.2.1 Probabilistic models

The first data-driven attribution model presented in the literature is a bagged logistic regression model (Shao and Li 2011). In this model, bagged logistic regression is fitted on customer journeys with a conversion dummy as the dependent variable and touchpoint occurrences as independent variables. The coefficients of the estimated logistic regression model serve to explain the contribution of each marketing channel. Next to the logistic regression model, Shao and Li (2011) also propose a simple probabilistic model. The bagged logistic regression model has better predictive power, but the probabilistic model has more stable estimates. To model spill-over effects in the simple probabilistic model, they implemented first and second order probability estimations. A disadvantage of both approaches is that they do not specify structure and therefore may be inaccurate models of reality. Dalessandro et al. (2012) propose a model that fits the Shapley value solution concept from cooperative game theory, and show that it is a generalization of the probabilistic model introduced by Shao and Li (2011). The Shapley value of a marketing channel equals its average impact across all customer journeys. We discuss more details on the Shapley value approach in Section 5.3. Li and Kannan (2014) develop an individual-level probabilistic model to attribute online channels using estimates of spill-over and carry-over effects. They model visit and purchase decisions via a two-level nested logit framework, and the Shapley value is implemented to allocate value among channels in the customer journey. A major drawback of this approach lies in the complexity of both specifying and estimating the model. Moreover, it is unclear how well the model is able to predict future conversion behavior.

5.2.2 Markov models

Abhishek et al. (2015) model the conversion funnel through a hidden Markov model, suggesting that there are three latent states (disengaged, active, and engaged) which represent customer interest before ultimately converting. The latent states are inferred from the customers research intensity as proxied by the number of page views, where the disengaged state is represented by low intensity, the active state by medium intensity, and the engaged state by high intensity. From each of the three states, the customer has a probability of converting as defined by his/her state, advertising related activities (i.e., touchpoints), and the number of pages viewed. Another

Markov model with customer states is proposed by Anderl et al. (2016). Higher order Markov chains are constructed, where a state in a k^{th} order Markov chain represents k consecutive channels observed in the customer journey. They implement the removal effect to measure channel contribution, and find that the predictive power increases with higher order Markov chains. The Markov chains also provide stable attribution results, especially for lower order Markov chains. A drawback of Markov-based models is that attribution can only be performed at the population level.

5.2.3 Point processes

Chandler-Pepelnjak (2010) and Zhang et al. (2014) propose additive hazard models based on survival theory for attribution analyses. They find that their method makes good inference of the parameters. Ji et al. (2016) use the Weibull distribution to describe the observed conversion delay and use the hazard rate of conversions to measure the influence of an ad exposure. The primary advantage of models based on point processes is the incorporation of time effects between visits. Both models perform well in both effectiveness and conversion prediction analysis compared with logistic regression, simple probabilistic, and standard Shapley value models. Xu et al. (2014) develop a stochastic model based on mutually exciting point processes. It models advertisement clicks and purchases as outcomes of univariate point Poisson processes in continuous time. Furthermore, customer heterogeneity is accounted for by incorporating individual random effects. In general, a limitation of point process models is that the intensity of events is only defined by the occurrence and type of touchpoints, and no effect is assumed from unobserved factors (e.g., customer behavior).

5.2.4 Vector autoregression models

Wiesel et al. (2011), De Haan et al. (2016), and Kireyev et al. (2016) construct vector autoregression (VAR) models to examine the long-term effectiveness and spill-overs across online and offline advertisements. This approach allows for the capturing of both carry-over and spill-over effects of advertising in online and offline channels. Unfortunately, it can only be estimated on the aggregate data level rather than individual customer journeys.

5.3 Attribution framework

In this section, we describe the Shapley value as an attribution framework. First, the Shapley value for marketing attribution is introduced. Then, we adjust the Shapley

value to incorporate user-level customer behavior information and combine it with the use of machine learning prediction to estimate the conversion probabilities of customer journeys. Finally, we show that the resulting modified Shapley value still exhibits the set of desirable properties that maintains its 'fairness'.

5.3.1 Shapley value

A situation in which a finite set of players can generate certain payoffs by cooperation can be described as a *cooperative game with transferable utility* (TU-game). A famous solution for TU games is the *Shapley value* (Shapley 1953). Each player is awarded a value, which equals the average of his or her contribution to each subgroup in the game. In this chapter, we consider weighted Shapley values as a marketing attribution model, where players in the model are replaced by marketing channels, customer journeys are the subgroups, and weights are assigned to specific subgroups based on the number of conversions resulting from journeys containing that specific set of channels. The Shapley value of a channel is thus estimated by computing the total impact of a marketing channel across all converting journeys.

Formally, the game consists of a set of marketing channels C, where $C = \{C_1, ..., C_n\}$, and the conversion probability v for any given subset of marketing channels. A customer journey S is defined as a set of k different channels $S = \{C_1, .., C_k\}$, where $S \in C$. In game theoretical terms, a customer journey can be referred to as a partnership of marketing channels. Naturally, the proportion of customers that encounter and convert from specific journeys differ, and the conversion probability of a specific journey is referred to as its *worth*. The worth of a journey is defined by the characteristic function v which is typically estimated for a journey S using its empirical conversion rate among a set of observed customer journeys. Let the weight of a customer journey S be ω_S, defined as the number of total conversions resulting from journey S, then the weighted Shapley value for channel C_i is given by:

$$\psi_{C_i}^{Sh}(C, v) = \sum_{S \subseteq C \setminus \{C_i\}} \omega_S \, E[v(S \cup C_i) - v(S)], \quad \forall C_i \in C. \tag{1}$$

The Shapley value is 'fair' in that it is the unique value that satisfies certain desirable properties: *efficiency*, *linearity*, *additivity* and the *null-player* property. Under different assumptions however, the Shapley value is often characterized by different properties (Young (1985); van den Brink and van der Laan (1998); van den Brink (2002)). Since the weighted Shapley values differ by having a weight function for every subgroup of players (or subsets of marketing channels), under this condition two properties, *positivity*, and *partnership*, are added, and the linearity property is omitted. In Section 5.3.3 we explain these properties in detail and generalize them to marketing attribution.

5.3.2 Shapley value modified for an ML predicted conversion rate

One problem with the straightforward application of the Shapley value for marketing attribution is that the empirical conversion rate of customer journeys, as represented by its set of marketing channels, gives inaccurate conversion predictions at the user level. This is because the straightforward application of the Shapley value assumes that the worth of a customer journey is equal to the number of customers that converted divided by the number of customers that did not. However, each user can experience his path to purchase differently. The Shapley value can be rewritten by including heterogeneous customers, and the characteristic function can thus be transformed into a summation of the conversion probabilities of individual customers for this journey. The inclusion of heterogeneous customers significantly increases the complexity of the customer journey, and therefore a machine learning (ML) model is implemented to map all instances of customer journeys at the user level to a single conversion rate prediction. The Shapley value is then extended to incorporate the predictions from the ML model. More specifically we introduce weighted Shapley values on user-level. In Section 5.3.3 we show that the fairness properties still hold under this extension.

Let $u \in U$ define a user from the set of all users and $U_S \subseteq U$ be the set of users that encountered a given converted customer journey S. Each converted customer journey has been encountered by multiple users and each of these users has a specific conversion rate for that experience. Such a characteristic function is denoted as $v_u \in \mathcal{D}$, where $\mathcal{D}$ is the set of characteristic functions for all users and their encountered journeys. For each S, ω_S characteristic functions are defined. Thus, under this extension, the worth of customer journey S can be defined as:

$$\bar{v}(S) = \sum_{u \in U_S} v_u(S), \tag{2}$$

where $\bar{v}(S)$ is the sum of all individual characteristic functions of users that converted via journey S. The term $v_u(S)$ represents the conversion probability of journey S for user u. As such, the weighted Shapley value when considering customer journeys on the user level is given by:

$$\psi_{C_i}^{Sh}(C, \bar{v}) = \sum_{S \subseteq C \setminus \{C_i\}} E[\bar{v}(S \cup C_i) - \bar{v}(S)], \quad \forall C_i \in C. \tag{3}$$

Note that the weight factor ω_S from Equation (1) is omitted, because $\bar{v}(S)$ already sums over all converting journeys for a given subset of marketing channels.

In this construction of the weighted Shapley value, the subset S only consists of information about the set of marketing channels encountered by the user. To account

for user-level behavior information, such as the number of pages browsed or the time spent on site, we include extra features denoted by X. Then, we rely on a machine learning prediction model $f_u^{ML}(S, X_S)$ to estimate the user level conversion probability $v_u(S)$ for journey S and the corresponding behavior features X_s for user u while experiencing journey S. The Shapley value on the user level, where each worth is estimated by an ML model, can be written as:

$$\psi_{C_i}^{ShML}(C, \bar{v}) = \sum_{S \subseteq C \setminus \{C_i\}} \sum_{u \in U_{S \cup C_i}} f_u^{ML}(S \cup C_i, X_{S \cup C_i, u}) - f_u^{ML}(S, X_{S,u}), \quad \forall C_i \in C. \quad (4)$$

5.3.3 Fairness of Shapley value attribution—five properties and their axioms

In this section we show that the constructed ML-extended Shapley value, namely $\psi_{C_i}^{ShML}(C, \bar{v})$, is a unique solution and is in fact still the weighted Shapley value. This means that it satisfies the following five properties: *efficiency*, *additivity*, *positivity*, *partnership*, and the *null player* property. Here we define the properties and generalize them for the purpose of marketing attribution.

The property of *efficiency* entails that the total gain made by all players cooperating together is fully distributed among all individual players. For marketing attribution, this means that the sum of the Shapley values of each marketing channel should equal the total number of conversions. Thus, the sum of the attributed value awarded to the marketing channels for a user should equal 1. This is fulfilled if the following axiom is satisfied:

$$\sum_{C_i \in C} \psi_{C_i}^{Sh}(C, \bar{v}) = \bar{v}(C), \quad (5)$$

where the worth $\bar{v}$ of the power set C equals all the conversions in the game. The ML model estimates the incremental value of each channel in a customer journey, therefore it does not fully attribute 100% of the total worth to its marketing channels. Thus, the axiom of efficiency is not fulfilled as is. Ruiz et al. (1998) show that it is possible to distribute the unattributed value among the players uniformly. We construct an alternative approach to normalize the incremental values of marketing channels within the customer journey on the user level. The normalized weighted Shapley value is given by:

$$\psi_{C_i}^{Sh}(C, \bar{v}) = \sum_{S \subseteq C \setminus \{C_i\}} E\left[\frac{\bar{v}(S \cup C_i) - \bar{v}(S)}{\bar{v}(C)}\right], \quad \forall C_i \in C. \quad (6)$$

The property of *additivity* states that if we can describe the game with a joint characteristic function $\psi^{Sh}(C, v, w)$, then the distributed games of these characteristic

functions should correspond to the games of these characteristic functions independently:

$$\psi_{C_i}^{Sh}(C, v, w) = \psi_{C_i}^{Sh}(C, v) + \psi_{C_i}^{Sh}(C, w). \tag{7}$$

A characteristic function can be regarded as a model to estimate the conversion rate of customer journeys. This axiom implies that if we have two models to predict conversion rates, then the sum of their predicted conversion rate should equal the conversion rate of both models together. This axiom can become useful when ensemble learning is implemented to better estimate the worth of specific customer journeys. In our application, we divide the worth of a customer journey into individual values for each user who has encountered that specific path. The axiom is fulfilled because $\psi_{C_i}^{Sh}(C, \sum_{u \in U_s} v_u(S)) = \psi_{C_i}^{Sh}(C, \bar{v})$.

Positivity states that if the characteristic function $\bar{v}$ is monotonic, then the worth of R, where $S \subseteq R$, should always be equal to or greater than the worth of S. Hence, if an additional marketing channel is added to an existing customer journey, then the conversion probability for the customer journey with the additional channel can only be equal to or greater than the conversion probability of the original journey. Using the ML model, we estimate the incremental value of a channel C_i as $\max(0, E[v_u(S \cup C_i) - v_u(S)])$, thus the characteristic function is monotonic and the axiom is fulfilled.

Partnership assumes that a coalition of partners can behave as one individual. This reduces the size of the game. Kalai and Samet (1987) prove that if S is a coalition of p partners with a single characteristic function v, then the group of partners receives a share in total and individuals receive their marginal contribution. While this axiom is more suitable when decision making of players is considered, one may apply it to a group of marketing channels, which forms a partnership (i.e., customer journey) and are treated as an individual entity. Our approach estimates the conversion probability per customer journey on the user level. The weighted Shapley value then allocates contribution to individual channels within the customer journey. Since the allocation is based on the conversion probabilities caused by the channels, the axiom is fulfilled.

Finally, the *null player* property requires that the Shapley value of a null-player o in a game is zero. A null-player is a player who makes no marginal contribution to any coalition in the game. For marketing attribution it means that if a marketing channel does not contribute towards conversion in any customer journey, then its Shapley value should also be zero. In our set of channels, no null-players exist. However, it is straightforward that if a marketing channel with zero impact was included, then the Shapley value of this marketing channel as computed from Equation (6) would be zero. This axiom is satisfied if the Shapley value for null-player C_o is $\psi_{C_o}^{Sh} = 0$.

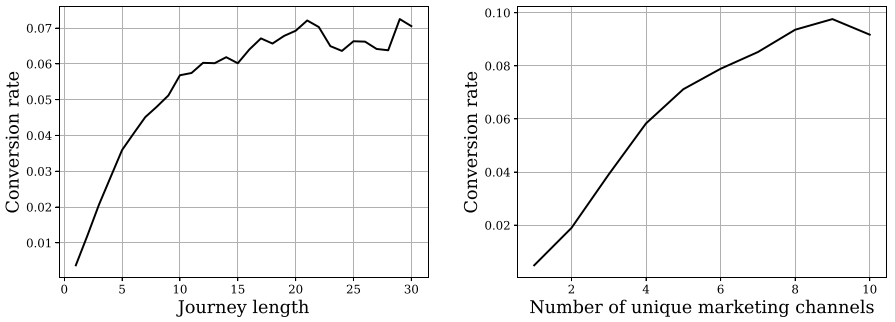
5.4 Data description

To test our approach, we rely on a dataset provided by a Dutch online travel agency. This agency sells all-inclusive vacation trips to a large number of destinations, with flights mostly departing from the Netherlands. The products are divided into two groups: summer holidays and winter holidays. We collected clickstream data aggregated at the user (cookie) level for all users from January 1, 2018 to September 31, 2018. The users are then split into individual customer journeys, with the end of the journey defined as either a conversion (purchase) or 30 days of inactivity. Customer journeys contain only visits to the website of the online travel agency, and a visit is characterized by the marketing channel that led to the visit, the number of pages the user browsed (both visited and viewed), and the total duration of visit on the site (difference in seconds between the first and last pages browsed). We do not have information on marketing actions that did not result in a visit (e.g., display banner views or Google searches). For analysis, we consider all customer journeys that ended in the six-month period between April 1, 2018 and September 31, 2018. Table 1 provides some basic summary statistics of the customer journeys in our dataset.

Table 1: Summary statistics.

Total number of journeys	5,597,382
Total number of visits	14,851,971
Total number of pages browsed	97,047,153
Average visit duration	330 seconds
Total number of conversions	67,757

In Figure 3 we explore how to the number of visits and the number of unique marketing channels in a customer journey relates to the likelihood of conversion. Figure 3a plots the conversion rate of journeys with 1 to 30 visits. We observe consistent increases of conversion rate as journeys get longer up to 10 visits. For journeys of only one visit, the conversion rate is under 0.5%. On the contrary, journeys with 10 or more visits have conversion rates of over 5.5%. The relationship seems to still be positive up to 30 visits, but becomes noisier as there are few journeys of such length. This suggests that, not accounting for other possible factors, longer journeys are more likely to convert. We believe that this may be due to the nature of the product sold by the travel agency. Holiday packages are generally expensive and are unlikely to be purchased impulsively. Customers tend to spend a significant amount of time deliberating not only where to go for holiday, but also must consider a number of other factors such as which dates to travel, and which hotel resort to stay at, and



(a) (b)

Figure 3: The conversion rate of customer journeys with respect to the number of visits (a) and the number of visits from unique marketing channels (b).

which vendor to purchase from. We do not expect such a purchase to be completed during a single visit, hence interested customers are more likely to return multiple times to browse a variety of products offered by the travel agency.

The focus of our paper is on marketing channels, and as such we consider a set of fifteen channels defined by the travel agency. Table 2 lists the channels and their occurrence frequencies. The most commonly occurring channel is *direct* at 38.6% of the visits, followed by *generic paid search* at 19.4%. There is significant variation in the position of visits in a customer journey from various channels. Table 2 also presents the distribution of the position of visits by marketing channel, splitting positions into four general categories: visit in a one-visit journey, the first visit in a multi-visit journey, the last visit in a multi-visit journey, and a middle visit (neither first nor last) in a multi-visit journey. *Display* visits predominantly occurred in single-visit journeys, suggesting that users may have clicked on the advertisement due to spur of the moment or by mistake. On the contrary, *email*, *referral*, and *branded paid search* visits mostly occurred in multi-visit journeys, suggesting a greater relationship between overall user engagement and these marketing channels. It should be noted that only 2.6% of *retargeting* visits occurred in single-visit journeys, which is expected because retargeting banners can only trigger for users who have previously visited the website.

Another interesting observation is that direct visits are far more likely to be the first visit in a journey than the last visit, whereas most of the other channels are much more likely to be the last visit. It is not clear why this may be the case, but a common belief is that when the referral source of a visit cannot be tracked properly the visit is assigned to be a direct visit. This may include sources that are highly

Table 2: Marketing channel statistics.

		Frequencies				Averages per visit	
Marketing channel	Num. occ.	Single visit	First visit	Last visit	Middle visit	Pages	Duration
Affiliates	481,467	33.1%	13.0%	17.2%	36.7%	6.3	231.9s
Branded Paid Search	1,583,691	16.4%	11.0%	17.0%	55.6%	13.7	515.1s
Email	798,168	11.5%	8.7%	10.1%	69.6%	5.3	183.5s
Direct	5,737,059	23.2%	13.9%	6.9%	55.9%	9.2	369.2s
Direct Deals	15,014	28.2%	11.4%	14.3%	46.1%	4.0	134.2s
Display	283,586	60.4%	8.7%	15.1%	15.8%	2.3	67.8s
Generic Paid Search	2,882,968	27.4%	13.7%	18.1%	40.9%	6.9	253.7s
Organic Search	1,624,718	34.5%	11.4%	19.3%	34.8%	9.6	374.5s
Paid Search	124,194	22.3%	4.1%	17.0%	56.5%	11.0	388.3s
Paid Social	625,680	31.8%	12.1%	16.8%	39.2%	5.1	141.2s
Redirects	15,221	25.5%	12.6%	14.7%	47.3%	9.5	354.3s
Referral	288,705	17.8%	13.2%	12.5%	56.5%	11.8	488.4s
Retargeting	271,042	2.6%	18.6%	1.9%	76.8%	4.8	153.2s
Social	72,676	36.3%	14.3%	13.7%	35.7%	5.0	168.1s
(Other)	47,782	33.1%	5.9%	19.0%	42.0%	6.5	241.4s

exploratory in nature. A primary reason for why 18.6% of the retargeting visits are the first visit of multi-visit journeys is due to how the journeys are defined. When the user clicks on a retargeting banner soon after converting or after 30 days since the previous visit, the retargeting visit will be recorded as the first visit in the journey. However, it is surprising that only 1.9% of retargeting visits are at the end of a journey.

Finally, Table 2 also includes information on user engagement during visits from each marketing channel. There is a large variation in both the number of pages browsed and the total time on site for visits from different channels. branded paid search and referral visits are effective channels for users who want to engage more with the travel agency, and display banners are generally poor. We expect that this is because different channels attract users from different parts of the purchase funnel. Display banners are prospective in nature and do little in selecting users that are more likely to engage. On the contrary, channels like branded search or referral are attracting users that have already demonstrated an interest in the travel agency thus visitors are more likely to engage. Surprisingly, email, direct deals, and social tend to exhibit lower visitor engagement. We believe that users may visit the website from promotional emails or direct deals just to browse the promotional products, but will leave the website when no products of interest are found.

5.5 Conversion prediction with machine learning

Since every customer journey either ends in a conversion or a non-conversion, predicting the conversion probability of customer journeys can naturally be formulated

as a classification problem. In a classification problem, a set of explanatory variables, called *features*, are used to predict a target variable. To predict the target variable, a machine learning model is *trained* by finding a functional relationship that best maps a set of feature values to the corresponding target value in a *labeled* dataset. For conversion prediction, the features include information about the customer journey (e.g., which touchpoints occurred in the journey), and the target value is binary: one if the journey converted and zero otherwise. Although the target value is binary, the predictions made by machine learning models are typically numeric between zero and one, corresponding to the likelihood with which the model believes a journey will convert. To evaluate the quality of the model's predictions, a *holdout* dataset is used. This dataset is also labeled but does not contain any observations used to train the model. To evaluate prediction quality, the machine learning model predicts the conversion likelihood of each customer journey in the holdout dataset, and the predictions are compared against the true labels based on a pre-specified error function. In this section, we first describe the machine learning model used to predict conversions. Then we provide the details of how features are engineered from the raw data to be used by the model. Finally, we present the prediction performance of the model and compare against other attribution methods.

5.5.1 Gradient boosted decision trees

We use Gradient Boosted Decision Trees (GBDT), also known as Gradient Tree Boosting (GTB), Gradient Boosting Machines (GBM), and Multiple Additive Regression Trees (MART). GBDT is widely known by the competitive predictive analytics community to be the best performing prediction algorithm for datasets that are arranged in tabular format (Chen and Guestrin 2016). More recently, Olson et al. (2018) tested 13 algorithms on 165 public datasets and found GBDT to be the best performing algorithm.

GBDT is a non-parametric prediction method and it works by starting with a naive prediction for each observation in the training data and iteratively fits regression trees on the residuals of the predictions made in the previous iteration. The newly fitted regression tree's predictions are then added to the predictions made from the previous iteration to form a new prediction. Two properties of GBDT make it particularly suitable for conversion prediction. First, the tree structure of GBDT allows for the model to implicitly capture feature interaction effects if they hold predictive power. This means that the model can segment customer journeys into different groups based on combinations of feature values if doing so can help predict conversion. In comparison, other approaches such as defining interaction terms in a logistic regression model or explicitly clustering customer journeys do not guarantee

improvements in prediction performance, potentially leading to overfitting. Second, since regression trees find specific points in the data to split on, it does not require any functional relationships between the features and the conversion probability. In particular, the effect that features such as the number of pages browsed or time spent on website have on conversion likelihood may not follow a well-defined distribution. Finally, as compared to other tree-based methods such as classification trees or random forests, the iterative nature of GBDT allows individual trees to focus on predicting different subsets of the data rather than having to predict all of the data together.

5.5.2 Feature engineering

All machine learning or statistical models require data to be in the standard tabular M-by-N format before it can be used to train (or estimate) the model and make predictions. The purpose of feature engineering is two-fold. First, raw data in the form of customer journeys must be converted into the M-by-N format, and second, important predictive information must be encoded into the features so it can be efficiently learned by the machine learning model. For some models (e.g., neural networks or logistic regression), this can include certain mathematical transformations (e.g., standardization or log-transformation), but this is not required for GBDT.

Using the data described in Section 5.4, we engineer 44 features that in combination represent a customer journey. These features will be used by GBDT to predict the likelihood that a customer journey will convert. We first construct a feature for each marketing channel, where the value for each feature is the total number of times a visit from that channel occurred in the customer journey. There are a total of 15 features in this group and we call it *marketing channel count vectors*. This set of features encode the basic information of the customer journey as common to all data-driven attribution models. A journey that has a single display visit and two direct visits would have values of 1 and 2 in their respective vectors, and 0 in the other thirteen vectors. Note that this formulation does not consider the order of which the visits occur. The order of visits is a particularly difficult attribute to encode in the tabular format without having to create an additional exponential number of features. As such, we omit this attribute for the prediction of conversion rates.

From the *marketing channel count vectors*, we then construct two features, *number of visits* and *number of unique marketing channels*, that better help GBDT use the information from customer journeys. *Number of visits* is simply the row-wise sum of *marketing channel count vectors*, representing the total number of visits a journey contains. *Number of unique marketing channels* is the number of non-zero values in each row of *marketing channel count vectors*. From Figure 3 we can see positive

relationships between conversion rate and both the *number of visits* and *number of unique marketing channels*, suggesting that they could help predict conversion likelihood.

Next, we construct three features relating to the level of engagement of the user with the travel agency's website. For each journey, we compute the *visit duration*, *number of pages browsed*, and *number of pages viewed* by the user. The values for these features were already computed in the raw data, but for each visit. Here, we sum over the values across all visits of the journey. From Table 2 we can see significant variation for both the pages browsed and visit duration across visits from different channels. Moreover, a quick calculation of the correlation between these three features and conversion shows correlation coefficients of 0.14, 0.20, and 0.20 for *visit duration*, *number of pages browsed*, and *number of pages viewed*, respectively. Taken together, the three user engagement features could add predictive value on top of the marketing channel related features.

Finally, we construct a set of features that correspond to the number of times for each hour of day that visits in each session started from. There are a total of 24 features in this group and we call it *hour of visit count vectors*. These features might not directly affect conversions, but could provide additional information about the customer. We posit that visits during work hours are more likely to be for orientation or product searches, whereas visits during evening hours result in greater likelihood of purchase. We also include the feature *number of unique hours of visit* to differentiate between users who always visit the website at the same time versus those who visit at different times.

There are other features that we do not include. The primary reason for not including them is their lack of predictive power. Since the focus of this step is on prediction performance, we do not include features that do not improve prediction quality on the validation dataset.

5.5.3 Prediction performance

To evaluate how well our model predicts conversion we randomly assigned 70% of the customer journeys to a training set, and the other 30% of the customer journeys to a held-out validation set. We compute the area under the receiving operator curve (AUC) on our predictions for the validation set to assess the performance of the model. Furthermore, we also compute the AUC on predictions made by six other models: last touch, first touch, Markov model (orders 2, 3, and 4), and the empirical journey conversion rate from the standard Shapley value attribution approach. Figure 4 presents the AUC scores of each of the models. It should be noted that a model that predicts conversions well does not necessarily perform attribution well.

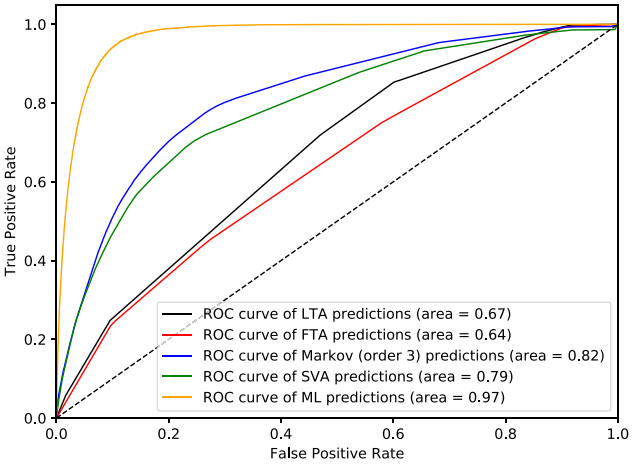


Figure 4: Conversion prediction performance.

However, being able to predict conversions well is evidence that the model fits users' conversion behavior well, and thus the attribution values derived from this model are also likely to be better.

In terms of conversion prediction, both the last touch and first touch attribution models perform poorly. This is both expected and surprising. Recall in Figure 3a that there is a strong relationship between the journey length and conversion rate. Naturally, by only considering a single visit, both first and last touch attribution models are agnostic to this relationship, and thus predict conversions poorly. However, the fact that these two models have predictive power at all indicates that the marketing channel for which visitors arrive from can indicate the journey's conversion likelihood. Indeed, given the large variation of user engagement in visits from different channels, we could also expect variations in conversion behavior.

Compared to the first and last touch attribution models, the Markov models and empirical journey conversion rate (used for SVA) performs substantially better at predicting conversions. These models capture both the effect of marketing channels, but also to a lesser degree the number of visits. For example, the second-order Markov model computes the conversion likelihood from the sequence of the last two visits in the journey. This approach can differentiate between single and multi-visit journeys. Finally, these models can also capture the effect of having multiple unique marketing channels in the journey.

The GBDT model performs considerably better compared to the other models, with an AUC of 0.97. This shows that beyond journey length and marketing channels, information about user engagement plays a significant role in determining whether or

Table 3: Feature importance.

Feature name	Relative importance
Time on site	48.71
Number of pages browsed	25.26
Number of visits	9.90
Number of direct channel occurrences	4.66
Number of pages viewed	3.61
Number of organic search channel occurrences	1.61
Number of visits starting between 15-16h	1.38
Number of generic paid search channel occurrences	0.68
Number of unique marketing channels	0.53
Number of visits starting between 10-11h	0.45

not a customer journey will convert. From a trained GBDT model, we can compute the relative importance of features on predicting the target variable. This is derived by computing the decrease in variance (known as variance gain) from each split in each regression tree of the GBDT and normalizing it by 100. Table 3 presents the relative importance of the ten most important features in our trained GBDT model. Three of the top five features are related to user engagement within visits, combining for 77.58% of the total variance gain. In particular, *time on site* alone accounts for almost half of the variance gain and *number of pages browsed* accounts for over a quarter.

One caveat about the AUC score is that it evaluates how well the model is able to differentiate between converting and non-converting journeys. A high AUC score does not mean that the model is as accurate in exactly predicting whether a conversion will occur. This is because most customer journeys in our dataset have a very low likelihood of converting, and the GBDT model is able to identify them with confidence. However, for the journeys that have a reasonable predicted likelihood of converting (e.g., between 5-50% chance), there is great uncertainty in whether they actually will convert.

5.6 Attribution with machine learning

Using the trained GBDT, we can then proceed with attributing conversion credit to the marketing channels. Recall from Section 5.3.2 that while the Shapley value attribution method attributes conversion credit directly to channels, our method attributes credit to touchpoints (i.e., visits), and then aggregates the credit from visits to the channels that led to the visit. As defined in Equation (4), the relative contribution of a touchpoint in a customer journey is proportional to the difference in conversion likelihood of that journey immediately before and after its occurrence. While the true conversion likelihood is unknown, predictions made by the trained

GBDT can serve as a replacement. For all converted customer journeys, sub-journeys of lengths 1 to $K-1$, where K is the full length of the complete journey, must be constructed together with their engineered features as described in Section 5.5.2.

We illustrate our attribution procedure using a simple example in Figure 5. Suppose we have a converted journey with four touchpoints, occurring in the following order led by the marketing channels: search ad, display ad, search ad, and direct. For these four touchpoints, the customer browsed 6 pages for 3 minutes in the first visit, browsed 3 pages for 1 minute in the second visit, browsed 14 pages for 12 minutes in the third visit, and browsed 2 pages for 3 minutes in the fourth visit. From this journey, we can create three sub-journeys. Sub-journey 1 contains information about only the first touchpoint (search ad, browsed 6 pages for 3 minutes), sub-journey 2 contains information about the first two touchpoints (search ad, display ad, browsed 9 pages for 4 minutes), and sub-journey 3 contains information about the first three touchpoints (search ad, display ad, search ad, browsed 23 pages for 16 minutes). In this example, the relative contribution of the first touchpoint (search ad) is f_{GBDT}(sub-journey 1), where f_{GBDT} represents the predicted conversion likelihood from the trained GBDT model. The relative contribution of the second (display ad), third (search ad), and fourth (direct) touchpoints are:

$$\begin{aligned}
\text{Contribution display} &:= f_{GBDT}(\text{sub-journey 2}) - f_{GBDT}(\text{sub-journey 1}),\\
\text{Contribution search (second)} &:= f_{GBDT}(\text{sub-journey 3}) - f_{GBDT}(\text{sub-journey 2}),\\
\text{Contribution direct} &:= f_{GBDT}(\text{complete journey}) - f_{GBDT}(\text{sub-journey 3}),
\end{aligned}$$

respectively. We can interpret the relative contribution of a touchpoint as the marginal increase in conversion likelihood of the customer as a result of the visit. Note that since the channel search ad occurs twice in this journey, its contribution is then the sum of the contributions for the two search ad touchpoints in the journey.

5.6.1 Attribution Results

We compute the attribution results for all converted journeys in the travel agency dataset for five attribution methods: last touch attribution (LTA), all touch attribution (ATA), first touch attribution (FTA), Shapley value attribution (SVA), and machine learning-SV attribution (MLS). Table 4 presents the attribution results for each marketing channel. In the first block, all converted journeys are attributed. Since attribution results for single-channel journeys are the same regardless of attribution method, in the second block we present the attribution results for only the multi-channel journeys.

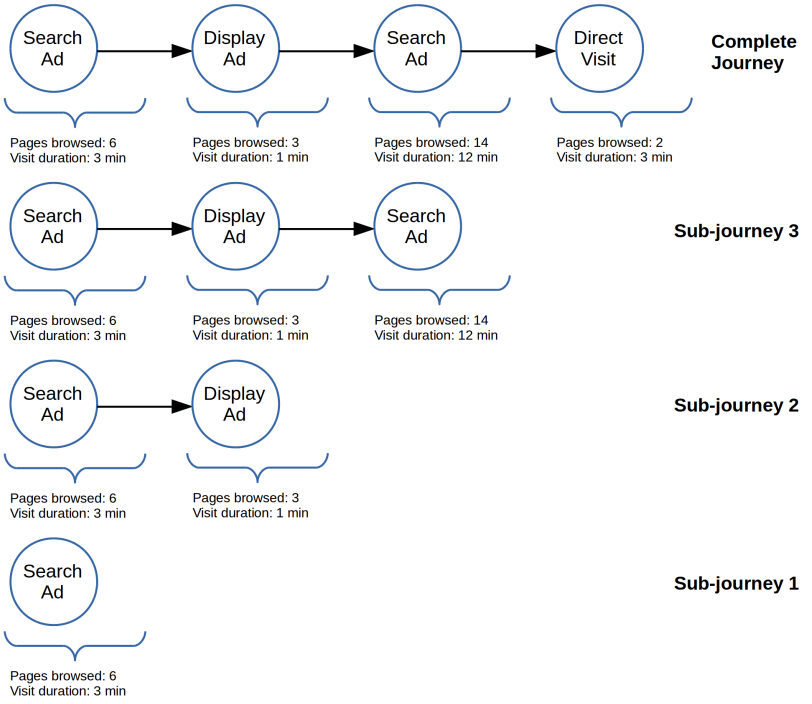


Figure 5: Illustration of attribution procedure.

A clear observation from Table 4 is how different the attribution results are across the attribution methods. When focusing on the multi-channel journey attributions, we can see that some channels result in greater variation across the methods than others. For example, referral has a large spread, where the difference in attributed conversions between the two most similar methods (FTA and ATA) is over 22%, and the difference between the two most dissimilar methods (FTA and SVA) is almost six-fold. While the difference between the three rule-based methods are by design and therefore could simply be a characteristic of the position of touchpoints in customer journeys, it is surprising that both SVA and MLS would produce such different results from both the rule-based methods and each other.

5.6.2 How SVA and MLS differ

Since both MLS and SVA are data-driven attribution methods, and moreover MLS is an extension of SVA, one might expect these two methods to produce the most similar attribution results. However, we can see from the absolute differences in

Table 4: Attribution results.

	All converted journeys					Multi-channel converted journeys				
Marketing channel	LTA	ATA	FTA	SVA	MLS	LTA	ATA	FTA	SVA	MLS
Affiliates	1,143	1,419	1,676	1,745	1,194	857	1,133	1,390	1,459	908
Branded Paid Search	12,992	13,729	15,820	16,275	14,682	7,836	8,573	10,664	11,119	9,526
Email	784	1,359	1,860	706	1,075	468	1,043	1,544	390	759
Direct	31,898	28,407	20,373	25,330	28,781	21,367	17,880	9,846	14,803	18,254
Direct Deals	12	18	31	21	15	9	15	28	18	12
Display	29	107	162	80	53	17	95	150	68	41
Generic Paid Search	6,996	9,564	12,262	5,590	8,147	4,468	7,036	9,734	3,062	5,619
Organic Search	9,091	9,543	12,656	10,912	9,532	5,876	6,328	9,441	7,697	6,317
Paid Search	198	351	845	359	415	107	260	754	268	324
Paid Social	113	206	272	138	129	65	158	224	89	81
Redirects	39	73	79	167	62	27	61	67	155	50
Referral	3,891	2,135	1,247	5,582	3,094	3,517	1,761	873	5,208	2,720
Retargeting	230	487	54	235	209	225	482	49	230	204
Social	145	142	138	266	137	100	97	93	221	92
Other	131	148	213	283	164	101	118	183	253	134

attributed conversions at the channel level that MLS is actually more similar to ATA and LTA than it is to SVA. We believe that a primary reason for the difference is due to the fact that SVA relies on empirical conversion rates of customer journeys.

Rare journey occurrences of SVA. A major part of the SVA's calculation of channel contribution relies on comparisons of conversion rates between sub-journeys. Conversion rate is simply an empirical point estimate of the likelihood that a journey will convert, and therefore is noisy when the occurrences of specific journeys (as defined by the marketing channels) is low. The noise can lead to abnormally high empirical conversion rates for some sub-journeys, and can greatly outweigh the signal when overall conversion rates are low. Since the overall conversion rate of customer journeys in our dataset is only 1.2%, an abnormally high empirical conversion rate (e.g., 10%) for a sub-journey can result in an exceptionally large share of contribution awarded to the last channel in that sub-journey. Since attributed conversions for channels in a journey are proportional to their relative contributions, when an abnormally large share of contribution is awarded to one channel, then that channel is attributed an abnormally large share of the conversion, and the other channels are attributed an abnormally low share of the conversion.

To investigate the effect of rare-occurring sub-journeys, we identified cases where the occurrence of a sub-journey is less than 10, 50, 100, 500, and 1000. The converted multi-channel journeys where these sub-journeys occurred are extracted and we compared the attribution results between SVA and MLS for these journeys. Table 5 presents the findings. 3.8% of the converted journeys have a sub-journey that

Table 5: Effect of rare sub-journeys on attribution differences between SVA and MLS.

	Journeys with rare sub-journey		All journeys
Def. rare sub-journey	Percent of journeys	Avg attr diff	Avg attr diff
Fewer than 10 occurrences	3.8%	1.52	
Fewer than 50 occurrences	7.5%	1.36	
Fewer than 100 occurrences	10.0%	1.30	0.86
Fewer than 500 occurrences	16.6%	1.20	
Fewer than 1000 occurrences	20.8%	1.15	

occurred fewer than 10 times. For these journeys, the average sum of absolute differences in attribution across channels between SVA and MLS is 1.52, which is over 75% higher than the average difference for all journeys. 20.8% of the converted journeys have a sub-journey that occurred fewer than 1,000 times. While 1,000 occurrences allow for much greater precision in the estimation of conversion rates, there is still sufficient noise to greatly affect attribution. For these journeys, the sum of absolute differences in attribution across channels between SVA and MLS is 1.15, which is still 33% higher than the average difference for all journeys.

Effect of customer behavior information. The second challenge of relying on empirical conversion rates of customer journeys is the difficulty of incorporating customer behavior information during visits. For two converting journeys with touchpoints led by identical marketing channels, the customers are likely to have behaved differently during their visits, suggesting that the conversion should be attributed differently. SVA cannot easily incorporate customer behavior information, as doing so would require creating multiple levels for the channels to reflect behavior, exponentially increasing the number of combinations in the system and further exacerbating the problem caused by estimating conversion rates.

MLS is able to incorporate customer behavior information by intelligently extracting useful information that predicts conversion without needing to explicitly encode all possible combinations of the information. Therefore, for the remainder of the converted customer journeys in our dataset that did not have a rare sub-journey, we can investigate the effect that customer behavior information has on attribution results. From the feature importance values presented in Table 3, we can see that the customer behavior features *visit duration* and *number of pages browsed* are most important for predicting conversions. We posit that when these two features are high, i.e., customers are more engaged with the travel agency website, the conversion probability is also high. Thus, touchpoints where visit duration is long or many pages were browsed should contribute more to the conversion and be attributed more credit. However, because SVA is unable to consider customer behavior in visits, there

Table 6: Effect of customer behavior information on attribution differences between SVA and MLS.

Percentile	Value at percentile	Percent journeys with	Avg attr diff above	Avg attr diff below
Visit duration 90^{th}	3,701 sec	52.9%	0.87	0.69
Visit duration 95^{th}	6,077 sec	29.5%	0.92	0.73
Visit duration 99^{th}	14,301 sec	6.5%	1.03	0.77
Pages viewed 90^{th}	64 pages	42.7%	0.88	0.72
Pages viewed 95^{th}	103 pages	25.6%	0.92	0.74
Pages viewed 99^{th}	205 pages	6.9%	1.00	0.77

would be a significant discrepancy in attributed conversions between SVA and MLS for journeys where a single touchpoint had exceptionally long visit duration or many page views.

From the converted customer journeys that did not have a rare-occurring sub-journey, we identified journeys with at least one touchpoint that exceeds the threshold for customer engagement. The average sum of absolute differences in attribution across channels between SVA and MLS are then compared. We consider the two behavior features (visit duration and page views) separately. Table 6 presents the findings. First, we set customer engagement thresholds at three levels for both features: 90^{th}, 95^{th}, and 99^{th} percentile of converting journeys. For visit duration, the 90^{th} percentile is at 3,701 seconds, or just over one hour. 52.9% of the remaining converting journeys had a touchpoint with at least this visit duration, and the average attribution difference for these journeys between SVA and MLS is 26% higher than the journeys without such a touchpoint (0.87 versus 0.69, respectively). The 99^{th} percentile for visit duration is much longer, at 14,301 seconds, or almost four hours. 6.5% of the remaining converting journeys had a touchpoint with at least this visit duration, and the average attribution difference for these journeys between SVA and MLS is 34% higher than the journeys without such a touchpoint (1.03 versus 0.77 respectively). Note that a visit of 3,701 seconds does not necessarily imply that the user was browsing the website for this whole duration, but that the user had kept the browser/tab open and at least intermittently interacted with the website. Page views exhibit similar effects as visit duration, which is expected as the two features are highly correlated. A user that had a long visit is also likely to have browsed many pages, so a touchpoint that falls into the high engagement category based on visit duration probably also falls into the same category based on page views as well.

5.6.3 Which rule-based attribution methods are reasonable?

Rule-based attribution methods were developed from a variety of marketers' intuitions. Using our dataset and GBDT predictions of touchpoint contributions, we can

Table 7: Effect of first, middle, and last touchpoints.

	Average touchpoint effect (L = journey length)								
Touchpoint(s)	L = 2	L = 3	L = 4	L = 8	L = 12	L = 16	L = 20	L = 24	Overall
First	0.048	0.043	0.039	0.033	0.025	0.027	0.024	0.021	0.037
Middle		0.045	0.040	0.023	0.016	0.011	0.008	0.007	0.029
Last	0.129	0.122	0.102	0.062	0.039	0.034	0.031	0.020	0.090

test which rule-based attribution methods, and therefore their corresponding marketing intuition, are more reasonable. To do so, we compute the GBDT predicted touchpoint contributions for each touchpoint in journeys of length 2 to 24. These touchpoints are grouped into three categories: first touchpoint (the first touchpoint in a journey), middle touchpoint (touchpoints that are neither first nor last in a journey), and last touchpoint (the final touchpoint in a journey). The average contribution as measured by gain in conversion probability of these touchpoints for a variety of journey lengths is presented in Table 7.

A number of observations can be made from Table 7. Supporting the last touch attribution method, the last touchpoint tends to contribute the most towards conversion when compared against other touchpoints. The first touchpoint is significantly worse than the last touchpoint, but is still slightly better than the middle touchpoints, providing some justification for the first touch attribution method. However, also supporting the notion against the first and last touch attribution methods, the middle touchpoints when combined contributes substantially to conversion. If we focus on journeys where the middle touchpoint exists (i.e., lengths 3 to 24), the total contribution of the middle touchpoints is almost equivalent to the total contribution of the first and last touchpoints together. This suggests that linear attribution, also known as all touch attribution (ATA), would be the best option among the rule-based attribution methods. Indeed, if we refer back to the attribution results in Table 4, the results from ATA is the most similar to that of MLS.

Finally, it can be observed that touchpoints decrease in average contribution as the customer journey becomes longer. This is because only converted journeys are considered. For shorter journeys to convert, it is expected for their touchpoints would have greater effect towards conversion than that of longer journeys.

5.6.4 Which channels are more effective?

The goal of conversion attribution is to facilitate return on investment (ROI) calculations of digital advertisement spend. From the attribution results, we compute the attributed conversions per touchpoint occurrence for each channel, thereby providing the net value of each channel. The findings are presented in Table 8.

Table 8: Attributed conversions per 10,000 touchpoints.

Marketing channel	LTA	ATA	FTA	SVA	MLS
Affiliates	24	29	35	36	25
Branded Paid Search	82	87	100	103	93
Email	10	17	23	9	13
Direct	56	50	36	44	50
Direct Deals	8	12	21	14	10
Display	1	4	6	3	2
Generic Paid Search	24	33	43	19	28
Organic Search	56	59	78	67	59
Paid Search	16	28	68	29	33
Paid Social	2	3	4	2	2
Redirects	26	48	52	110	41
Referral	135	74	43	193	107
Retargeting	8	18	2	9	8
Social	20	20	19	37	19
Other	27	31	45	59	34

Per touchpoint occurrence, we can see that there is great variation in effectiveness across the channels. The worst performing channels are display and paid social, accounting for only 2 conversions per 10,000 occurrences as per the MLS attribution. Moreover, because a touchpoint is a visit, each occurrence represents a click and not an impression. The best performing channels are referral and branded paid search, accounting for around 100 conversions per 10,000 touchpoints as per the MLS attribution. With the exception of these two channels, the other paid channels do not outperform organic channels (direct and organic search) on a per-visit basis for the travel agency. This is reasonable as organic channel visits are made by users who either already knew about the travel agency or have searched for a product that the agency offers, demonstrating prior interest in either the company or its product. However, other channels such as affiliates, email, generic paid search, and paid social are engineered to identify the users that are likely to be interested in the agency or its products. For these channels to perform worse than the organic channels suggest improvements are possible with regards to targeting the right potential customers at the right time.

Since MLS relies heavily on the customer engagement features for attribution, we can actually already predict the two best and worst channels by looking at the average duration per visit for the channels in Table 1. Just as the channel performance, display and paid social visits have the shortest duration while branded paid search and referral visits have the longest duration.

5.7 Conclusion

The most commonly used data-driven conversion attribution algorithm is based on the Shapley value. This approach is attractive because Shapley values have some desirable properties that ensure fair and logical attribution results. Integral to the Shapley value attribution approach is the use of conversion probabilities. Typically, empirical probabilities (i.e., observed conversion rates) of customer journeys are used to estimate conversion rates. However, empirical probabilities are not reliable due to the size of the power set of customer journey combinations. Even with 5 million customer journeys, we show empirically that many converting journeys (20.8%) have rare-occurring sub-journeys that significantly distort the attribution results. Additionally, empirical probabilities do not consider underlying behavioral factors that influence (and/or explain) the observed conversion rates, such as as properties of the web-site visits (e.g., time on site, number of pages views).

In this chapter, we developed an approach that extends the Shapley value based attribution. This approach entails estimating the conversion probabilities by a Gradient Boosted Decision Tree (GBDT) model. Using data from a Dutch online travel agency, we have shown that using this approach we can take into account additional factors that explain conversion rates. This approach shows a considerable increase in predictive power compared to other well-known attribution models, including the Shapley value model with empirical conversion probabilities. The attribution results of our model differ noticeably from the standard Shapley value attribution model and three rule-based models. Finally, we show that all three rule-based models, last touch, first touch, and all touch attribution models make reasonable assumptions but all fail to capture the complete attribution picture.

Chapter 6

Recombining Customer Journeys With Probabilistic Cookie Matching: A Supervised Learning Approach

Chapter Abstract

The gathering and analysis of online customer journey data is central to web-based business success. Having an accurate understanding of customer behavior allows web businesses to offer the right product or promotion to the right person at the right time. Unfortunately, customer journey data is often gathered by following cookie histories on web browsers or mobile apps. Although this works well when users access the web with only one device, it results in fragmented customer journeys when multiple devices are used. Such fragmentation biases models of customer behavior, leading to incorrect user metrics, poor ad-targeting, and unintended promotions or product recommendations. In this chapter, we introduce a supervised learning approach to recombine customer journeys in the presence of multiple device usage and frequent cookie wiping. We model customer journey recombination as a predictive cookie matching problem and use machine learning methods to link different cookies of the same user probabilistically based on commonly available information such as time of use, IP address, location, device, and page content. Using a deterministically matched dataset of cookies, we create cookie pairs and train on 50% of the data while testing on the remaining 50% with masked labels. We find that performance can be high, with over 0.9 F_1-score when users are geographically sparsely populated and moderate with over 0.5 F_1-score when users are densely populated. We show that the sharing of IP address has the strongest predictive power while other features can still predict well in the absence of IP address information. Finally, we find that logistic regression performs on par with the state-of-the-art machine learning method when users are sparsely populated but significantly underperforms when user population is dense.

6.1 Introduction

As more devices become connected to the internet, consumers' web behavior becomes increasingly fragmented. The user can connect to the web through her mobile phone, tablet, personal laptop, work desktop, TV, smart home device, and even car. Traditionally, web users have been tracked by storing cookies in the browser and analyses are often conducted with the cookie representing the user. Since a cookie is connected to the browser of a device, different devices will have different cookies, leading to biases in online customer journeys when users are using multiple devices or are frequently deleting cookies. In order to obtain a complete view of online customer behavior, it is necessary to match cookies to their users so accurate browsing history can be captured when no user identification is available.

Consider the following problem: user u_1 visits a news content website with her smartphone in the morning and browses article a_1. A cookie c_1 is placed in her phone and tracks that she has browsed article a_1. At night, she reads another article a_2 on her laptop, but because it is not via her smartphone a new cookie c_2 is placed in her laptop, tracking that she has browsed article a_2. The news content website is unable to know that cookies c_1 and c_2 belong to the same user u_1, and therefore stores into its database that two users visited the website, one browsed article a_1 and the other browsed article a_2. As a result, simple metrics such as the average browsed articles per user per day becomes biased, and support tools such as recommendation systems would not be able to use information from c_2 when recommending articles to her smartphone and use information from c_1 when recommending articles to her laptop, thus hindering its effectiveness.

There are two methods of matching cookies to users: (1) deterministic matching through login history (e.g., if cookie c_1 and cookie c_2 both logged into the same account then they belong to the same user) and (2) probabilistic matching through user behavior (e.g., the probability of cookie c_1 and cookie c_2 belonging to the same user is probabilistic with respect to some predictors). While deterministic matching is naturally precise, the number of cookies with user logins is usually small. Alternatively, probabilistic matching is able to assign match probabilities to all cookie combinations but is dependent on the quality of the predictors. In this chapter, we model probabilistic matching as a supervised classification problem. Using a dataset of deterministically matched cookies from an online news content platform, we develop and test a supervised learning approach to match cookies to users. We find that using common information available to most websites (e.g., IP address, location, timestamp, browser, device, content) it is possible to achieve over 95% precision and 90% recall scores for cookies in sparsely populated geographic areas

and over 50% precision and 65% recall scores for cookies in densely populated areas. Furthermore, we show that in the absence of some information our approach can still achieve reasonable performance.

6.2 Related work

As the use of non-personal computer devices (e.g., mobile phone, tablet, TV) became more prevalent for web use, a number of papers have studied the behavior of users across different channels, especially mobile. Ghose and Han (2011) look at how user content generation relates to usage behavior on the mobile internet, Ghose et al. (2012) and Tossell et al. (2012) explore how mobile browsing behavior differs from personal computers, and Kamvar et al. (2009) and Manchanda (2015) analyze mobile search behavior. Müller et al. (2012) try to understand and improve design for tablet use, and more recently, Xu et al. (2016) look at the impact of the availability of tablets on digital commerce. While these papers discover a number of differing aspects of mobile device use behavior from personal computers, they do not consider the possibility of combining mobile use with personal computers to paint a more complete picture of customer behavior.

In the computer science and social network analysis literature, link prediction bears strong resemblance to the probabilistic cookie matching problem. Link prediction focuses on inferring links between nodes in a graph, usually some form of social network, using graph-based features formulated as a supervised learning problem (Taskar et al. (2004), Al Hasan et al. (2006), Liben-Nowell and Kleinberg (2007), Lichtenwalter et al. (2010)). A major difference between link prediction and probabilistic cookie matching is that link prediction attempts to predict future association between two distinct nodes whereas probabilistic cookie matching looks to connect nodes from past behavior.

In joint efforts between Drawbridge and the 2015 IEEE International Conference on Data Mining (ICDM), and between the Data-Centric Alliance and the 2016 International Conference on Information and Knowledge Management (CIKM), two public challenges were held with the goal of matching multiple cookies to the same user. In the ICDM challenge, Drawbridge released a dataset of cookies and devices with a set of anonymized features to describe attributes and behaviors of both the cookies and devices. The CIKM challenge was simpler as competitors were asked to find matches in a set of user ids provided by the Data-Centric Alliance with only anonymized URLs, HTML titles, and timestamps. The top-performing teams in the two challenges scored 0.895 ($F_{0.5}$) and 0.420 (F_1), respectively (Phan et al. 2017).

Although both challenges were held in conjunction with academic conferences and drew interest from researchers and practitioners alike, the winning solutions were heavily engineered towards optimal performance for the specific datasets provided and therefore holds limited value towards solving a general form of this problem. Furthermore, the lack of information regarding how the data was generated along with heavy anonymization of variables results in additional difficulty when analyzing both the methods and results discussed in the challenges. This paper aims to provide a clear understanding of how to perform probabilistic cookie matching and analyzes the effect of various features on match performance.

6.3 Data description

Our data comes from a five-month use log of a Dutch online news content platform from the start of December 2016 to the start of May 2017. Content from a number of newspapers are aggregated to this platform and it is available as an application accessible from the web, through a smartphone, or a tablet. The newspapers are all in the Dutch language and users come primarily from The Netherlands and Belgium. Login is required to use this application so all cookies and devices are deterministically matched to a user ID. Using this dataset we are able to develop our probabilistic cross device matching method and test it against the ground truth. All page views for each user are tracked with timestamps, including pages such as "my home page" and the "payment portal". Since the purpose of probabilistic cookie matching is to match users who are not logged in, we keep only content page views, resulting in a total of 124,883 users utilizing 453,574 cookies (3.63 cookies per user) across 347,551 IP addresses (3.81 per user). In total there are 9,735,894 page views. Figure 1 provides the distribution of cookies and IP addresses per user and Figure 2 provides the distribution of page views per user, per cookie, and per IP address.

Beyond page views the dataset also includes information about the device (e.g., device type, operating system, browser, browser language, screen size) along with the derived location coordinates of the device at the time of the page view. There are 54 unique operating systems (OS) and 109 unique browsers (e.g., Chrome and Mobile Safari) used. The most popular OS are Windows 10 and Mac OS X (iPad) with 1,974,562 and 1,812,372 page views, respectively, while the most commonly used browsers are Mobile Safari and Chrome with 2,273,871 and 2,076,049 page views, respectively. There are a total of 2,245 unique device screen sizes ranging from mobile (360-by-640) to standard laptop/monitor (1920-by-1080) to ultra-wide monitors (3440-by-1440). Our users come from 11,863 different (recorded) cities with Amsterdam being the most popular.

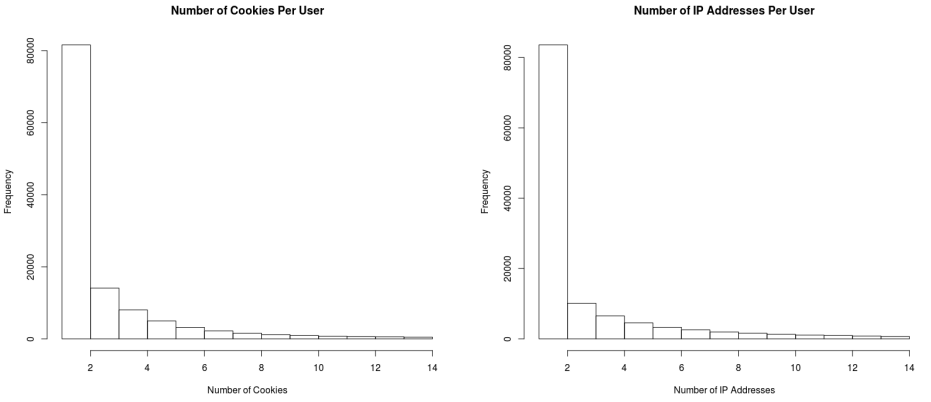


Figure 1: Histograms of the number of cookies and IP addresses per user (maximum is constrained at 15).

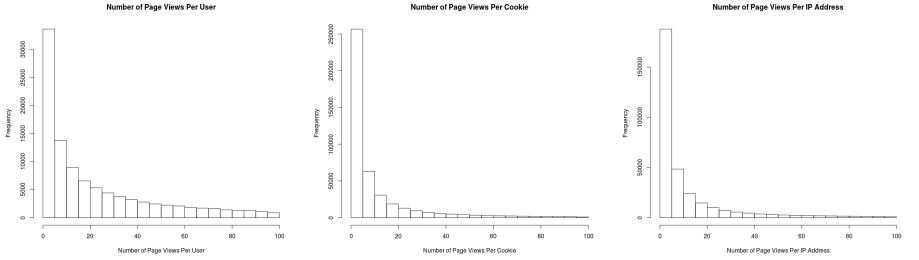


Figure 2: Histogram of the number of page views per user, cookie, and IP (maximum is constrained to 100).

6.4 Supervised learning approach

Probabilistic cookie matching can be viewed as a pairwise classification problem, where given information about two cookies as explanatory variables, a model is trained to predict the binary response variable: 1 if the two cookies belong to the same user, and 0 otherwise. This was also the predominant approach taken in the high-performing solutions of the ICDM and CIKM competitions. First, a set of 4,100,814 matched cookies are created through matching their deterministic user IDs. From this set of matched cookie pairs, we build a training and test dataset where 50% of the pairs are randomly selected to be in either set based on their user IDs. To most closely resemble real-life applications, we ensure that all matched cookie pairs of the same user are in the same set to prevent user-level information from leaking between the training and test sets. Finally, due to the large number of non-matching cookie pairs as this is a 2-combination problem of 453,574 cookies, we randomly sample

Table 1: The complete set of features used for the supervised learning algorithms and their descriptions.

Feature name	Feature description
num_events1	Number of events total for cookie 1
num_events2	Number of events total for cookie 2
num_events_diff	Absolute difference of number of events for cookie 1 and cookie 2
num_unique_ip1	Number of unique IP addresses for cookie 1
num_unique_ip2	Number of unique IP addresses for cookie 2
date_overlap	Dummy for whether the first and last dates of cookie 1 and cookie 2 overlaps
num_ip_shared	Number of unique IP addresses appearing in both cookie 1 and 2
num_lat_shared	Number of unique latitudes appearing in both cookie 1 and 2
num_long_shared	Number of unique longitudes appearing in both cookie 1 and 2
avg_diff_lat	Absolute difference between the mean latitudes of cookie 1 and cookie 2
avg_diff_long	Absolute difference between the mean longitudes of cookie 1 and cookie 2
num_br_shared	Number of unique browsers appearing in both cookie 1 and 2
num_br_lang_shared	Number of unique browser languages appearing in both cookie 1 and 2
num_os_shared	Number of unique operating systems appearing in both cookie 1 and 2
num_screen_width_shared	Number of unique screen widths appearing in both cookie 1 and 2
num_screen_height_shared	Number of unique screen heights appearing in both cookie 1 and 2
num_content_author_shared	Number of unique content authors appearing in both cookie 1 and 2
num_content_length_shared	Number of unique content lengths appearing in both cookie 1 and 2
num_content_source_shared	Number of unique content sources appearing in both cookie 1 and 2
num_events_overlap1	Number of events for cookie 1 which falls in the period of cookie 2's activity
num_events_overlap2	Number of events for cookie 2 which falls in the period of cookie 1's activity

10,251,247 observations (0.01%) from the set of all possible non-matching cookie pairs and evenly split them between the training and test datasets.

6.4.1 Feature engineering

A key to the supervised learning approach is feature engineering—creation of the requisite features which hold predictive power on the response variable. For this problem, simply using the original variables as explanatory variables is insufficient as it is the comparison of these variables for both cookies which drives whether they belong to the same user. For example, the set of locations of two cookies are unimportant when considered independently, but from that we can derive the number of common locations between the two cookies, which can have high predictive power given that two cookies that are always in different cities are unlikely to have the same user. We engineer a set of 21 features related to information about timestamps, IP address, location, device, and content. Table 1 lists each feature and describes how they are created.

6.4.2 Learning algorithm

Although a number of supervised learning methods can be used for this task, we select gradient boosted decision trees (with the logloss objective function for binary classification) for its overall effectiveness in supervised learning problems in general (Friedman 2001). It was also the dominant method used in the ICDM and CIKM competitions and has been the most used method in winning data science competitions (Chen and Guestrin 2016). Gradient boosted decision trees achieves top performance from being able to readily exploit interactions in the features and does not require monotonic transformations. Gradient boosting allows the algorithm to focus on difficult-to-predict observations to more rapidly improve model estimation, and overfitting is reduced by aggregating a large number of decision trees trained on subsets of both features and observations. The specific implementation used is LightGBM (Ke et al. 2017), which performs on par with other implementations such as XGBoost (Chen and Guestrin 2016) but is an order of magnitude faster[1]. We also compare LightGBM with logistic regression to more easily compute feature effects and to give a general performance comparison between the two methods.

6.4.3 Evaluation metric

The key evaluation metrics of this problem are precision, recall, and the F_1-score which is the harmonic mean of precision and recall. In the scope of probabilistic cookie matching, precision is the number of correct predictions of cookie matches divided by the total number of predicted matches and recall is the number of correctly predicted cookie matches divided by the total number of true matches. Both metrics are important because wrong predictions of matched cookies can result in false inference of user behavior while failing to detect matched cookies limits the completeness of the customer journey picture we want to infer. The F_1-score provides a balance between the two metrics.

6.5 Prediction results and evaluation

Using our complete set of features on the test set of 7,166,089 observations (0.01% down-sampling of unmatched pairs) and setting cutoff threshold to 0.5, we achieve F_1-scores of 0.933 and 0.928 for LightGBM and logistic regression, respectively. The corresponding precision and recall scores are 0.968 and 0.900 for LightGBM, and 0.956 and 0.901 for logistic regression. We can see that logistic regression performs on

[1]Performance comparisons at `https://github.com/Microsoft/LightGBM/wiki/Experiments#comparison-experiment`

Table 2: The complete set of features used for the supervised learning algorithms and their descriptions.

Feature name	Coef estimate	Std. Err	z-statistic	Significance
num_events1	-0.009	0.000	-28.113	***
num_events2	-0.009	0.000	-27.864	***
num_events_overlap1	-0.004	0.000	-15.577	***
num_events_overlap2	-0.003	0.000	-12.398	***
num_events_diff	0.006	0.000	18.333	***
date_overlap	-1.695	0.008	-217.875	***
num_unique_ip1	-0.006	0.001	-5.317	***
num_unique_ip2	-0.002	0.001	-1.664	
num_ip_shared	10.638	0.074	144.532	***
num_lat_shared	1.055	0.023	44.998	***
num_long_shared	0.507	0.024	21.316	***
avg_diff_lat	0.016	0.000	35.783	***
avg_diff_long	0.005	0.000	23.491	***
num_br_shared	1.074	0.005	227.978	***
num_br_lang_shared	0.898	0.004	204.287	***
num_os_shared	1.542	0.004	351.254	***
num_screen_width_shared	0.402	0.012	33.439	***
num_screen_height_shared	0.601	0.012	50.549	***
num_content_author_shared	0.171	0.003	55.788	***
num_content_length_shared	-0.075	0.002	-37.670	***
num_content_source_shared	0.078	0.002	33.711	***
intercept	-4.081	0.005	-906.650	***

par with LightGBM, suggesting that the feature set is relatively simple and the features have strong predictive power independently. Table 2 shows the logistic regression coefficients to provide further analysis of the feature effects. The features that positively predict matched cookies are the number of shared IP addresses, locations, browsers, operating systems, and screen sizes. These features focus on detecting cases where the same device is used but due to either frequent cookie churning or privacy restrictions, the cookie is removed resulting in new cookies added to the same device many times. Conversely, when the two cookies have time overlaps in their activities and as the number of page views increases in the overlapping time, the pair is less likely to belong to the same user. This makes sense because many different users are using the platform at the same time, but nevertheless we expect that cross-device use would result in time overlaps of activity between cookies. This suggests that cross-device use may be rarer than expected.

Since both LightGBM and logistic regression makes probabilistic predictions and both precision and recall are computed with binary predictions, we must set cutoff thresholds (e.g., 0.5) for which a prediction is 1 or 0. The thresholds greatly affect precision and recall scores as increasing the threshold usually leads to increased precision but decreased recall. Figure 3 plots the precision and recall scores for both

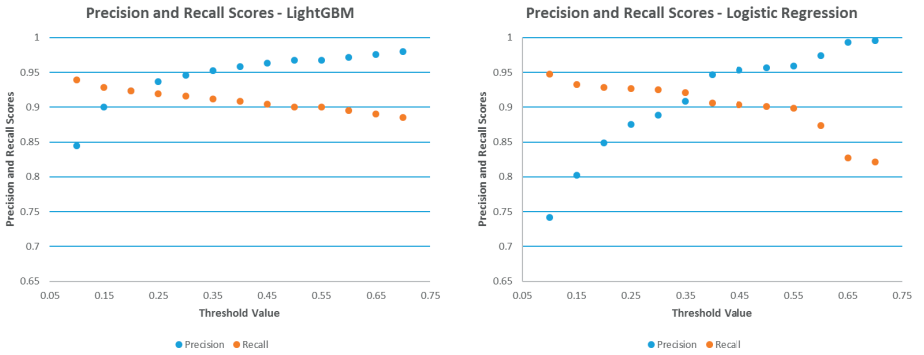


Figure 3: Precision and recall for LightGBM and logistic regression across different cutoff threshold values.

LightGBM and logistic regression across a range of thresholds. Both algorithms exhibit the same patterns of inverse relationship between precision and recall. At low cutoff thresholds, precision is low and recall is high, and as the threshold increases, precision increases and recall decreases. While the F_1-score favors balanced values of precision and recall, selecting the optimal threshold depends on the relative costs of false positives versus false negatives. If cookie matching is used to send personalized promotions then it is probably worse to send the promotion to the wrong customer than to miss a customer. Alternatively, if it is used to improve retargeting advertisement campaigns then targeting the wrong user would be akin to a prospecting advertisement which has minimal consequences. In both cases, the cutoff threshold can be selected to best suit business needs.

6.5.1 Performance with feature subsets

Our complete feature set includes information about timestamps, IP address, location, device, and content. However, in many practical settings, some of these features may not be available. For example, due to privacy reasons, it may not be possible to obtain the complete IP address or location of the user, and in other cases, information about the webpage's content may not be available. We carry out supervised learning on subsets of the feature groups to test the performance of our approach under different feature availability. First, we assume that timestamps will always be available. This is reasonable as any web-based application can always record the time of an event. Next, we consider cases when each of the other features types (IP address, location, device, content) are unavailable. Location is sometimes inferred through the IP address so we also consider their combined unavailability. Finally, in a more pessimistic setting, we also assume that content is unavailable so we consider the

Table 3: Precision, recall, and F_1-score of LightGBM and logistic regression when types of feature information are not available. The scores are based on the best thresholds selected for each algorithm and each case.

	LightGBM			Logistic Regression		
Feature Setting	Precision	Recall	F_1-score	Precision	Recall	F_1-score
All Features	0.9633	0.9046	0.9330	0.9594	0.8988	0.9281
No IP Address	0.9042	0.8730	0.8883	0.9058	0.8253	0.8637
No Location	0.9671	0.8911	0.9276	0.9617	0.8936	0.9264
No Device	0.9715	0.8241	0.8917	0.9997	0.7978	0.8874
No Content	0.9620	0.9049	0.9326	0.9585	0.9000	0.9283
No IP Address, Location	0.8890	0.7941	0.8389	0.8635	0.8012	0.8312
No IP Address, Location, Content	0.8888	0.7925	0.8379	0.8731	0.7979	0.8338

case with only timestamps and device features. Table 3 summarizes the performance of LightGBM and logistic regression for the above cases. We see that LightGBM is consistently better than logistic regression but only slightly. Among the feature types, IP address, contains the most predictive power as the F_1-score decreased by 0.045 for LightGBM and 0.054 for logistic regression when it was removed. Another important feature is device information as the F_1-score decreased by 0.041 for both LightGBM and logistic regression when it was removed. The other features have negligible importance in the presence of IP address and device information, but in the absence of IP address both location and content information play important roles as removing them individually decreased F_1-score by 0.05 for LightGBM and 0.03 for logistic regression. Finally, removing both location and content in the absence of IP address decreases F_1-score down to under 0.55, suggesting that they still hold strong predictive power.

6.5.2 Using cookies from one city without down-sampling

Unlike other evaluation metrics (e.g., area under the receiver operating curve), the F_1-score can be affected by the distribution of true and false observations in the data. Because we down-sampled the false observations by 10,000 times, we expect that our prediction performance could be significantly different from the performance of practical implementation of the cookie matching problem where all possible pairs must be considered. Unfortunately, creating a full set of 104,582,453,185 cookie pairs is infeasible so we constrain the set geographically, only including cookies that have appeared in a specific city. This is reasonable because we do not expect two cookies that have not both appeared in a single city to be a matching pair, therefore allowing us to create a set of cookie pairs that is more reflective of practical use. The city we choose is Haarlem, The Netherlands, with a population of approximately 150,000. Constraining our dataset to only cookies that have appeared in Haarlem, we have

Table 4: Precision, recall, and F_1-score of LightGBM and logistic regression for the complete cookie pair set for cookies from the city of Haarlem, Netherlands. Rows indicate when the specified feature information type is not available. The scores are based on the best thresholds selected for each algorithm and each case.

	LightGBM			Logistic Regression		
Feature Setting	Precision	Recall	F_1-score	Precision	Recall	F_1-score
All Features	0.4263	0.7280	0.5377	0.4340	0.5602	0.4891
No IP Address	0.3701	0.3502	0.3599	0.0210	0.0001	0.0002
No Location	0.5351	0.6941	0.6043	0.5444	0.3831	0.4497
No Device	0.4887	0.6355	0.5525	0.3541	0.0038	0.0075
No Content	0.4259	0.7242	0.5363	0.5258	0.3449	0.4166
No IP Address, Location	0.0976	0.2194	0.1351	0.0036	0.0623	0.0068
No IP Address, Location, Content	0.0570	0.5153	0.1026	0.0273	0.2703	0.0496

a total of 9,556 cookies, 83,568 matched cookies pairs, and 45,570,222 unmatched pairs. Similar to earlier, we randomly assign 50% of the cookie pairs to the training set and evaluate performance on the other 50%, but we ensure that cookie pairs from the same user are assigned to only the training or test set without any cross leakage.

Table 4 shows the performance of LightGBM and logistic regression with the full dataset of Haarlem. In comparison to that of the down-sampled dataset of all cities, performance is significantly worse for both algorithms. Furthermore, while LightGBM is still able to produce reasonable predictions when IP address or device is not available, logistic regression fails to discriminate between matched and unmatched pairs. Table 5 shows the coefficients of logistic regression. The direction of effects is largely similar to when trained on the down-sampled data with the exception of the number of content length shared, which changed from insignificant to negatively significant. The consistency of logistic regression coefficients suggests that the same pattern still remains. Unfortunately, because the geographic density of cookies is now 10,000 times higher, there is a larger proportion of unmatched cookies that share IP addresses (through common workplaces or businesses open to public) and have similar location. In order to improve prediction performance under higher density, we look to include additional features such as content topic information which may help discriminate between matched and unmatched cookie pairs.

6.6 Conclusion

This chapter attempts to recombine broken customer journeys via probabilistic cookie matching. We formulate this problem as a supervised learning problem similar to that of link prediction in social network analysis, and develop features from commonly accessible information to predict pairwise matching of cookies. Using data

Table 5: Logistic regression coefficients for the complete cookie pair set for cookies from the city of Haarlem, Netherlands. All coefficients except for num_content_length_shared are significant at least at the 0.05 level.

Feature name	Coef estimate	Std. Err	z-statistic	Significance
num_events1	-0.017	0.001	-17.468	***
num_events2	-0.016	0.001	-16.018	***
num_events_overlap1	-0.004	0.001	-4.729	***
num_events_overlap2	-0.002	0.001	-2.991	**
num_events_diff	0.013	0.001	14.430	***
date_overlap	-1.663	0.027	-60.766	***
num_unique_ip1	-0.049	0.004	-12.900	***
num_unique_ip2	-0.042	0.003	-13.057	
num_ip_shared	3.843	0.016	239.687	***
num_lat_shared	0.370	0.174	2.127	**
num_long_shared	0.994	0.174	5.719	***
avg_diff_lat	0.024	0.001	16.317	***
avg_diff_long	0.011	0.001	19.394	***
num_br_shared	1.137	0.018	63.143	***
num_br_lang_shared	0.438	0.016	27.694	***
num_os_shared	0.998	0.016	61.255	***
num_screen_width_shared	0.581	0.045	12.891	***
num_screen_height_shared	0.800	0.044	18.047	***
num_content_author_shared	0.091	0.009	10.084	***
num_content_length_shared	0.009	0.006	1.420	
num_content_source_shared	0.134	0.007	19.402	***
intercept	-8.404	0.017	-486.996	***

from a Dutch online new content platform, we train and test two supervised learning algorithms—gradient boosted decision trees with the LightGBM implementation and logistic regression. We find that both algorithms perform excellently when the geographic density of cookies is low from down-sampling (over 0.95 precision and over 0.90 recall) but performance decreases as geographic density increases (down to over 0.50 precision and over 0.65 recall). Moreover, LightGBM is able to better distinguish between matched and unmatched pairs than logistic regression when the signal is weaker. Finally, we show that information about IP address and device are highly predictive and location and content are not important in their presence. However, when IP address or device information are absent, location and content are moderately predictive and thus become necessary. For future work, further improvements can be made on developing predictive features to better discriminate matched and unmatched cookies in close proximity. Our current feature set does not take full advantage of usage patterns and content browsing, leaving two areas for investigation. We recommend that business practitioners consider the geographical density of cookies, information available, and the precision vs. recall tradeoff prior to

implementation of this approach as they can all have profound effects on optimizing performance for business needs.

Chapter 7

Improving Display Advertising With Predictive Device Matching: A Machine Learning Approach

Chapter Abstract

Retargeting is a highly effective strategy of targeting display advertisements to potential online customers who have recently visited the advertiser's website. Johnson et al. (2017) estimate that retargeting campaigns can increase an online retailer's website visits and purchases by over 17% and 10%, respectively. However, retargeting advertisements rely on tracking users with HTTP cookies which is unable to link different devices belonging to the same user, thus treating all devices as different users. This limitation in tracking constrains the effectiveness of retargeting campaigns by losing retargeting candidates when they switch devices or refresh cookies. To link devices and expand the pool of retargeting candidates, we develop a machine learning framework to predict device matches based on user browsing behavior. Using data from an e-commerce website and a programmatic buying platform, we show that our method is able to identify device matches for 10.5% of the e-commerce customers, resulting in an increase in retargeting candidates by 75.4% and total retargeting advertisements served by 104.4%.

7.1 Introduction

Display advertising is an online marketing method that consists of presenting banners of advertisers on publisher websites (e.g., `www.msn.com`) to potential customers. The banners can range in different sizes, be placed in different locations of the publisher's website (top-of-page or on the sidewall), and be embedded in different forms of media such as text, video, or mobile apps. In 2016, the Interactive Advertising Bureau (IAB) estimated that out of a total of $72.5 billion U.S. spending on internet advertising, $31.7 billion was on display advertising, with $13.6 billion on desktop and $18.1 billion on mobile devices (Interactive Advertising Bureau 2016).

A particularly effective strategy of display advertising is retargeting, which identifies customers who have recently visited the advertiser's website but did not make a purchase (convert), and subsequently serves banners to them in hopes of maintaining engagement and inducing an eventual conversion. Retargeting campaigns have the advantage of targeting individuals who are already aware of the advertiser and have demonstrated purchase intent. This results in higher rates of customer response (e.g., clicks) and sales, leading to increased efficiency in advertising spend. The effectiveness of retargeting is well known in the industry, with advertisers investing significant amounts of their online budget in this approach (Adroll 2017). Recent academic studies have agreed with this notion, with Johnson et al. (2017) and Sahni et al. (2017) finding that retargeting campaigns caused website visits to increase by 17.2% and 14.6%, respectively, and Johnson et al. (2017) finding a causal increase in sales by 10.5%.

Although retargeting campaigns have proven to be effective in boosting website traffic and sales, its practical implementation is unable to recognize retargeting candidates when customers use different devices, thus failing to take full advantage of all available candidates and leave potential revenue on the table. Retargeting campaigns work by placing HTTP cookies into the web browser of customers when they visit the advertiser's website and then serve banners to those tracked customers after they leave. However, because HTTP cookies are unique to a single browser, different devices used by the same person are treated as different customers. If the customer first visits the advertiser's website on her mobile phone and subsequently browses the web with her laptop, the retargeting campaign is unable to recognize her laptop and will not serve banners to her. Between desktop and laptop computers, mobile phones and tablets, and alternative connected devices such as video game consoles and televisions, various industry measures have estimated the average user to own more than three devices in 2016 (GlobalWebIndex 2016). Wang (2017) shows that for a single Dutch news aggregation platform across a period of six months, the mean number of devices per account was 3.63 and over 55% of the accounts were accessed with at least two different devices.

Another problem with retargeting campaigns is the negative effect of harassing customers with advertisements after he or she has recently made a purchase. This is wasteful because a customer who has just bought a product is unlikely to buy from the advertiser again within a short period of time. While existing technology allows advertisers to avoid retargeting customers who have recently made purchases, it also relies on HTTP cookies and only works on the device from which the purchase was made. Therefore, if the customer visited the advertiser's website on both her desktop and mobile phone but made a purchase on her desktop, she will continue

to be shown retargeting banners on her phone, leaving her feeling irritated while still costing the advertiser money. Taken together, these two problems demonstrate a strong need to link devices to users whether it is to maximize the potential for retargeting effectiveness or to minimize the potential for customer frustration.

The tracking limitations of HTTP cookies have been well-known by digital marketing practitioners and a number of commercial services to match devices have been offered. There are generally two types of device matching methods: deterministic and probabilistic. Deterministic matching is offered by online platforms with large logged-in user-bases such as Facebook or Gmail. These platforms are able to match devices to owners when they share a sign in to the same user account on the platform. Probabilistic matching vendors gather data about device usage and develop models to match devices based on information such as IP address, browsing behavior, and geographical location. A probabilistic model is estimated on a small set of deterministically matched data and then is used to predict matches on all available devices. The end product sold by both deterministic and probabilistic matching vendors is a graph of matched devices, where nodes represent devices and edges between two nodes represents a pairwise match.

Unfortunately, the device graphs provided by commercial services are rarely useful for retargeting purposes due to recency requirements of matching active and track-able devices. Delivering retargeting banners requires fast response to customer visits so only a limited time window is allowed to gather sufficient data to predict device matches. Existing services often need large numbers of user browsing sessions to identify matches between devices, but this is inappropriate for retargeting as HTTP cookies often do not remain active for very long. A sample of one day's display impressions from a programmatic buying platform show that over 66.6% of online browsing events belong to devices which were not previously known, suggesting a high degree of cookie churn and anonymous browsing behavior. Therefore, the portion of the device graphs which do not contain active and track-able devices is outdated for real-time use. Matching an active device to inactive or non-track-able devices does not increase the number of retargeting candidates as display advertisements can only be served to users during active browsing.

This paper asks the following two questions. First, with only a limited amount of data due to recency requirements of matching active devices, how well can devices belonging to the same user be matched based on only browsing behavior? This is further complicated as the tracking of browsing behavior is limited to websites on which display advertisements are auctioned so visits to many major websites are excluded (e.g., Google and Facebook). Second, how many more devices can we add

to the retargeting pool through matching devices to their users, and will the amount and matching quality sufficiently improve the overall effect of retargeting campaigns?

To answer these questions, we develop a machine learning framework to identify active devices belonging to customers of a retargeting advertiser, allowing us to serve retargeting banners to devices of customers which were not used to visit the advertiser's website. These devices would not have been targeted by the retargeting campaign otherwise, hence the pool of retargeting candidates is expanded through device matching. Due to the lack of ground truth device matches with which to train and validate our framework, we simulate ground truth by artificially splitting browsing history of existing devices. As a result, a number of device-related features such as IP address, browser type, and operating system could not be used for device matching. We base our framework on only the user's browsing history which is representative of the data available to programmatic buying platforms and also serves as a lower bound on predictive ability.

In collaboration with ORTEC Adscience, a Dutch programmatic buying platform, we test the effectiveness of our device matching framework and its impact on retargeting campaigns. Using browsing data obtained via bid requests from various ad exchanges, we collect browsing histories of devices which have previously visited Adscience's advertising client in the past 14 days—actual retargeting candidates. We train and validate our device matching framework by randomly assigning 50% of devices to a train and validation set, achieving validation scores of 51% recall and 67% precision. We then collect another dataset of potential retargeting candidates—devices which have not visited Adscience's advertising client but shared an IP address with the actual retargeting candidates. Our device matching framework is then used to predict pairwise matches between the actual and potential retargeting candidates, identifying new potential retargeting candidates for 10.5% of the actual retargeting candidates, consequently increasing the pool of retargeting candidates by 75.4% and total retargeting banner volume by 104.4%.

7.1.1 Relation to literature

This paper broadly falls under three streams of literature related to display advertising. A large body of literature have studied real-time bidding, with the focus on developing algorithms to either optimize budget over time (Cai et al. (2017), Ren et al. (2018), Wang et al. (2017)) or to improve targeting through predicting user response to individual banner ads (McMahan et al. (2013), Juan et al. (2017), Vasile et al. (2017)). While this stream of literature is highly relevant to any programmatic buying platform, ORTEC Adscience already employs variants of the proposed state-of-the-art implementations.

Another stream of literature focuses on understanding how display advertisements work and under which conditions they perform better or worse. Specifically, the effectiveness of retargeting campaigns has been of great interest due to widespread industrial success. Using an econometric model, Ghose and Todri-Adamopoulos (2016) found that retargeting increases the customer's propensity to convert by 26%. Further studies by Johnson et al. (2017) and Sahni et al. (2017) use controlled field experiments to demonstrate the effectiveness of retargeting campaigns. Johnson et al. (2017) used "ghost ads" to identify control-group counterparts of exposed consumers, finding that retargeting ads lifted website visits by 17% and purchases by 11%. Sahni et al. (2017) randomly assigned retargeting candidates to one group which sees the relevant advertisement and another group which gets a public service announcement that allows for tracking of the users in place of retargeting banners. They found that 14.6% more users in the product-viewers retargeting campaign returned to the website.

The third relevant stream of literature focuses on cookie churn and cross-device matching. Dasgupta et al. (2012) noted that HTTP cookies are not persistent for the same user, recognizing that internet users frequently delete cookies from their browsers. They dubbed it the "cookie churn" problem and presented a constrained clustering framework based on interval graph coloring and cookie similarities. While their method performs reasonably well, it can only be applied to cases of deleted cookies on the same device with the same browser. Others have also identified problems resulting from cookie churn, specifically with regards to measuring the effectiveness of online advertising (Coey and Bailey (2016), Johnson et al. (2017)). As the cookie churn problem became well-known and cross-device use became more popular, algorithms for cross-device matching (also referred to as probabilistic cookie matching or cookie stitching) have become a popular topic among practitioners and academics (Roy et al. (2015), Kim et al. (2017), Volkova (2017), Zimmeck (2017)). Two international data science competitions were held in conjunction with the 2015 IEEE International Conference on Data Mining (ICDM) and the 2016 International Conference on Information and Knowledge Management (CIKM Cup) to engineer novel approaches for cross-device matching. Hundreds of teams competed in the two competitions and a number of methods were published in the proceedings of the two conferences (Cao et al. (2015), Díaz-Morales (2015), Kejela and Rong (2015), Kim et al. (2015), Landry et al. (2015), Lian and Xie (2016), Tran (2016), Phan et al. (2017)).

With the exception of the CIKM Cup solutions, all of the other papers on cross-device matching rely heavily on IP address as a key feature in device match prediction. In this chapter, we do not have direct access to IP address in our data and

therefore deviate from those solutions. The dataset provided by the CIKM Cup is most similar to ours and therefore we base our cross-device matching methodology on that of Phan et al (2017). However, we make modifications to improve match quality and to satisfy the runtime constraints of industrialization.

Finally, while there has been a large body of literature on cross-device matching methodology, to the best of our knowledge this is the first paper to study the implications of cross device matching, specifically with respect to retargeting advertising—one of the most effective form of display advertising.

Structure of this chapter. The remainder of this chapter is structured as follows. In Section 7.2 we introduce background about how display advertising and retargeting work in practice. We also describe the data used to develop and test our device matching algorithm. Section 7.3 presents the machine learning framework for cross-device matching. Details on candidate selection, feature engineering, and the supervised learning algorithm for pairwise matching classification are provided. Section 7.4 reports performance evaluations of the matching framework on a validation dataset. Section 7.5 presents the impact of cross device matching on retargeting campaigns, suggesting potential economic gains in practice. Finally, we conclude in Section 7.6.

7.2 Business process and data collection

Display advertising works by showing banners to consumers when they surf the web. When a consumer lands on a website with display advertising, called a publisher, the publisher selects a banner from one of its advertisers to show the user in hopes of inducing the consumer to click on the banner, redirecting the consumer to the advertiser's landing page. An individual showing of a display banner is termed an "impression". Traditionally advertisers have bought impressions in bulk directly from a single or network of many publishers. In the late 2000s, the prevalence of real-time bidding (RTB) technology, also called programmatic buying allowed all display impressions to be bought and sold individually through ad exchanges. While some impressions are still sold in bulk directly to advertisers, most of the impressions are now sent to various ad exchanges through a sell-side intermediary, called the supply side platform (SSP). On the other end of the exchange is the buy-side intermediary, the demand side platform (DSP) which buys impressions for its advertiser clients.

When a consumer lands on a publisher's website but before loading into her browser, a bid request is auctioned through the SSP to various ad exchanges requesting DSPs to bid for the available impressions. Often included with the bid request

is a set of information including an SSP-specific identification code, the IP address of the consumer, the browser and device type, geographical location, and the URL of the website which is about to load. With this information, the DSPs have under 50 milliseconds to make bids representing the amount of money they are willing to pay to show the ad of their clients in this impression. The highest bidder wins the auction and pays the publisher money equal to the second highest bid or the floor price, whichever is higher.

There are generally two strategies for which DSPs follow to purchase impressions, prospecting and retargeting. In prospecting, the DSP evaluates all bid requests and computes its attractiveness using a targeting algorithm. One popular targeting approach is contextual targeting, where advertisers focus on specific types of websites to purchase impressions. For example, an automobile seller would prefer to advertise on the car-related pages of news outlets under the notion that consumers who read about automobiles are more likely to be interested in buying a car. Another approach is behavioral targeting, where the consumer is tracked and her browsing history and previous response to display banners are used to decide whether to continue to serve banners to her. While there is tremendous potential in both targeting methods, they are difficult to implement in practice as DSPs usually receive limited information about the website and the consumer in a bid request. A commonly used metric for measuring effectiveness of display advertising is the click-through rate (CTR) of banners. The CTR is the number of times customers clicked on the display banner divided by the total number of times the banner is served to customers. Although CTR varies by type of banner (e.g., top-of-page or on the sidewall), the overall CTR of display banners is estimated to be 0.05%. Adscience's targeting algorithm for prospecting strategies is estimated to yield a CTR of 0.10%.

7.2.1 Retargeting advertising

The second strategy DSPs follow to purchase impressions is retargeting, which is a special type of behavioral targeting. Retargeting is a simple but highly effective strategy, requiring minimal user browsing history and no contextual information. When consumers visit the advertiser's website, HTTP cookies are placed in their browsers by the advertiser's DSP. Subsequently, when consumers enter another website which has exchange-traded impressions, the DSP is able to identify them as customers of its client, and bids to win those impressions, serving them ads of the client which they recently visited. The advantage of retargeting is that consumers who have visited a website are already far into the conversion funnel by having demonstrated interest in the website's brand or products. These consumers are more likely to remain interested in the advertiser and respond to the banner. Based on

experience at Adscience, the CTR and conversion rates of retargeting banners are on average five times greater than that of banners in general, and more than twice that of its prospecting campaigns.

7.2.2 Data collection and description

Our data comes from bid requests (events) of all visitors of an advertising client of Adscience over a period of six weeks from July 15 to August 31 of 2017. These visitors are true retargeting candidates, and the goal is to match other devices which belong to these visitors so they can be served retargeting banners as well. Due to the massive number of devices and events received from the ad exchange we only collect events of devices which have shared an IP address with a true retargeting candidate over the six week period. All devices accessing the internet from the same network (e.g., in the same house, company, or public wifi) will share the same IP so it is highly likely that devices belonging to the same user will have shared an IP address within the six weeks. Kim et al (2016) and Wang (2017) also found sharing an IP address to be the greatest predictor of belonging to the same user.

In total, we obtained 10,000,000 events from 50,000 unique visitors to the client (true retargeting candidates) and 15,000,000 events from 65,000 visitors who have shared an IP address with the true retargeting candidates (potential candidates). In order to account for cases of users idling and refreshing pages, we combine consecutive events from the same URL within a 30-minute time frame. This reduced our dataset to 6,000,000 events for the true retargeting candidates and 8,000,000 for the potential candidates. We set aside the first two weeks of events from the true retargeting candidates for developing (training and validating) our machine learning model, and use the third week of events from both the true and potential retargeting candidates to match devices. Although events data from longer periods are available, we deliberately select a shorter period to align with the response time constraints of matching active and track-able devices.

7.2.3 Ground truth data generation

In order to train a machine learning model for device matching prediction, it is necessary to first have a dataset with device matching labels. These labels often come from a deterministically matched dataset where users have self-identified their devices by logging into online platforms (e.g., Facebook or Google) with multiple devices. Unfortunately, we do not have access to deterministically matched device labels with our bid request data so we simulate cross-device use by artificially splitting long single-device browsing histories into multiple devices.

From our first two weeks of events for the true retargeting candidates, we create a device matching training set with cookies of over 100 events. The events are grouped into sessions, characterized by consecutive browsing without a 30-minute break in between. At the start of a new session, we simulate whether the session would be assigned to the same pseudo-device or a new pseudo-device. A single cookie can only belong to one device, so we use the term *pseudo-device* to represent the simulated device after splitting a cookie. As part of the CIKM Cup 2016, the Data-Centric Alliance released anonymized cross-device browsing events data with matched labels. This dataset allows us to calibrate simulation parameters for device splitting. We sample a two-week period of browsing events and compute the probability that a user is using two or more devices based on the number of events from 100 to over 500 by increments of 50. We also compute the probability of whether the devices are used across overlapping or distinct periods. These probabilities are used to determine the number of pseudo-devices a cookie will be split into, and by which type (overlapping versus non-overlapping). After simulating device splitting, we have a training set of 6,000,000 events from 50,000 unique visitors and 175,000 pseudo-devices.

7.3 Device-matching algorithm

Our device-matching algorithm primarily uses two types of information: the URL of the websites visited, and the timing of web browsing behavior. The URL of websites visited provides information about the content that is being browsed. We posit that a single user will have overlaps in the websites he/she browses across his/her devices. Therefore, we take a natural language processing (NLP) approach to handling the URLs and construct metrics which can reflect the similarity of browsing pattern across devices. The timing of web browsing behavior is also an indicator of whether devices belong to the same user. For some users, different devices are never used together at the same time (e.g., work versus home desktop) while other users tend to use multiple devices together (e.g., mobile phone and laptop). We construct metrics of browsing behavior for each device across the seven day-of-week and twenty-four hour-of-day blocks.

Using the URLs visited information, we first filter pairwise device combinations based on websites visited to reduce the number of potential device matches. Next, we engineer a set of 1,200 predictive features suitable for a state-of-the-art decision-tree based machine learning algorithm. Finally, we split our dataset of events from the pseudo-devices to training and validation datasets for tuning model parameters and estimating predictive performance. After training and tuning the machine learning matching algorithm, we apply pairwise predictions to the devices in the test set,

which consists of the set of devices that have previously visited the advertiser's website (actual retargeting candidates) and another set of devices which have not visited the advertiser's website but have shared an IP address with those who have (potential retargeting candidates). The devices from the potential retargeting candidates set that are matched to the devices from the true retargeting candidates set are thus included for retargeting campaigns in the future.

7.3.1 NLP approach to modeling website visits

Individual devices are characterized by a sequence of events which contains the URLs and timestamps of the website visits. An example of an URL is `http://www.msn.com/nl-nl/nieuws/binnenland`, which is the Dutch landing site of MSN's domestic news page. From the URLs, we can infer a user's behavior on a device and compute similarity metrics with behavior on other devices. Device pairs which share more URLs in common are defined to have greater similarity than pairs that share fewer URLs in common. This is analogous to many applications in the fields of *natural language processing* and *information retrieval* where text documents are first transformed into numerical vectors and then are classified or retrieved via operations on those vectors. We model devices' event histories as documents of URLs and transform them into numerical values using *term frequency inverse document frequency* (Manning et al. 2010) and Doc2Vec (Le and Mikolov 2014). The result of these transformations are used to compute similarity metrics and ranks for the initial candidate selection and as features for the machine learning algorithm. We describe the transformation process in more detail below.

URL structure and hierarchy. Due to the structure of URLs where common information is stored in different parts of the URL (e.g., `www.msn.com/nl-nl` and `www.msn.com/en-us` are both landing pages for MSN but one is for the Netherlands and the other for the U.S.). It is not desirable to remove the common information before the "/" symbol and assume that the Dutch and American landing pages for MSN are completely different nor to remove the different information after the "/" and assume that they are exactly the same. To account for the structural information in URLs we first convert each URL into a set of four URLs using the hierarchical structure illustrated in Phan et al. (2017). An URL is split into segments via the "/" symbol and converted into a set of four URLs (H0-H3) as specified:

- H0: only keeping the domain (e.g., `www.msn.com`)
- H1: domain plus one additional segment (e.g., `www.msn.com/nl-nl`)
- H2: domain plus two additional segments (e.g., `www.msn.com/nl-nl/nieuws`)

- H3: domain plus three additional segments (e.g., `www.msn.com/nl-nl/nieuws/binnenland`)

URLs with fewer than four segments do not include the higher level hierarchies and URLs with more than four segments are truncated as the information after the fourth segment is often too specific. We use the converted URLs to perform both TF-IDF and Doc2Vec transformations and compute similarity features.

Term frequency-inverse document frequency (TF-IDF). TF-IDF is a method which creates a single vector for each URL and computes a numeric value for the presence of this URL in each document. The value of each URL visited by a device is defined by the frequency of occurrence of the URL in the device (term frequency) multiplied by the logarithm of the inverse of the number of devices the term appeared in (inverse document frequency). This transforms a device's URLs into an $m \times 1$ vector where m is the total number of unique URLs visited by all devices. TF-IDF is preferred over simply keeping a vector of counts of each URL in a device due to its ability to account for the overall popularity. Since we are using TF-IDF vectors to compute behavioral similarity between two devices, URLs that are less popular among devices should be awarded greater weight than more popular URLs. The inverse document frequency part of TF-IDF accounts for this.

Doc2Vec representation of URLs. Doc2Vec (Le and Mikolov 2014) is the document level analog of Word2Vec (Mikolov et al. 2013), a neural network based state-of-the-art method of embedding words with a continuous vector representation. Using the co-occurrence of words (and context words) in documents, Doc2Vec is able to identify semantic structure in words such that the vector for "king" minus the vector for "man" plus the vector for "woman" is equal to the vector for "queen". Similar relationships such as countries and capital cities can also be detected. Doc2Vec creates the vector representation using rolling windows of context words so different URL hierarchy levels result in significantly different representations of URL vectors. Therefore, we apply Doc2Vec to each of the four URL hierarchies. Two major parameters of Doc2Vec need to be set by the user, the number of vectors in the space and the window size for how many URLs are included as context URL for a given word. We use 300 for the number of vectors and a window size of 5 for hierarchy levels H0 and H1, and size of 10 for H2 and H3.

7.3.2 Initial candidate selection

A major challenge of device matching is the large number of device pairs which can be created from a set of users. There are $\binom{n}{2}$ unique pairs in a set of n cookies,

resulting in prohibitively large numbers of pairs for even a small number of devices (e.g., 50,000 devices results in almost 1.25 billion pairs). Furthermore, the number of true matches is linear with the number of devices so the ratio of false to true matches grows quadratically, leading to a severely imbalanced dataset when the number of devices is large. To reduce the data size to a linear factor of the number of devices, we follow the method of Phan et al. (2017) and perform candidate selection using a k-nearest neighbors (kNN) algorithm on TF-IDF and Doc2Vec transformations of visited website URLs to filter out unlikely device pairs. For each device, (kNN) identifies the k most similar devices to it based on the specified features and similarity metric. Here our features are the numerical vectors computed either from TF-IDF or Doc2Vec (resulting in two sets of candidates), and our similarity metric is the Euclidean distance, as denoted by $\|dev_i - dev_j\|_2$ for all devices i and j where the device is characterized by a numerical vector. We select the hyperparameter k which maximizes the device matching performance on our validation dataset. Beyond using kNN to do candidate selection we also use distances and ranks from both TF-IDF and Doc2Vec transformations as features for pairwise prediction.

A drawback of filtering out unlikely device pairs is that we are implicitly assuming that the excluded pairs are highly unlikely to be from the same user based on our filtering algorithm. While this step lacks precision (i.e., it is weak at identifying true device matches), it is still able to find over ninety percent of matched pairs in a linear factor of the number of devices. This also leads to the advantage of providing our pairwise matching algorithm with more difficult cases to train, thus improving its discriminative power.

7.3.3 Pairwise device matching via machine learning

After the initial candidate selection step, we create pairwise features for each candidate pair and train a machine learning algorithm to predict whether the pair of devices belong to the same user. We use gradient boosted decision trees (GBDT), also called gradient boosting machines (GBM) and multiple additive regression trees (MART). GBDT is a general purpose classification algorithm and has been the dominant algorithm in winning machine learning competitions such as the KDD Cup in 2015 (Chen and Guestrin 2016). Although a number of other machine learning algorithms can be used for pairwise device matching (e.g., logistic regression, neural networks, support vector machines), we rely exclusively on GBDT in this chapter, due to its superior performance and flexibility.

GBDT is based around the classification and regression trees (CART) algorithm (Breiman et al. 1984). The CART algorithm works by finding splits in the data myopically to minimize the classification error in the training sample. It is prone

to overfitting as a tree that is deep enough can perfectly the training sample while generalizing poorly to the unseen test sample. It is also prone to underfitting because myopic splitting works on one level at a time and is unable to reconsider better splits from earlier levels. GBDT improves on the CART algorithm by combining many decision trees, each trained with bootstrapped aggregation of observations and features (Breiman 1996, 2001). Furthermore, gradient boosting (Friedman 2001) is applied which trains new decision trees using the residuals of previous trees, thus guiding the algorithm towards better performance on observations which are more difficult to predict. This indirectly circumvents the overfitting and underfitting problems of the CART algorithm.

7.3.4 Feature engineering

A key component of any successful machine learning implementation is feature engineering. Using the combination of websites visited and timing of web browsing behavior, we create six groups of features used to predict device matches: similarity between devices from numerical vectors of browsed URLs (Euclidean distance, Manhattan distance, cosine similarity), rank of similarity between devices, indicator for whether the candidate set was selected from the TF-IDF, Doc2Vec, or both transformations, the number of events for both devices by day-of-week, hour-of-day, and hour-of-day-of-week, the difference in number of events between the device pair by day-of-week, hour-of-day, and hour-of-day-of-week, and the number of events the pair of devices overlap in 5, 10, and 60 minute periods. In total, 1,200 features are used.

7.4 Device matching performance

There are two steps to the process of measuring the performance of our framework. First, to evaluate our ability to identify device matches, we randomly split (evenly) the set of pseudo-devices into train and validation sets. For this, we perform the complete process separately for the two sets, including TF-IDF and Doc2Vec transformations, filtering for initial candidate selection, and feature engineering. We subsequently train the GBDT using the train set and then predict device matches on the validation set. This is the only method to evaluate match performance as true device matches are unobserved. In the training set we have 17,476 pseudo-devices from 7,390 pseudo-users and 13,456 true pseudo-device pairs. In the validation set we have 17,356 pseudo-devices from 7,361 pseudo-users and 13,273 true pseudo-device pairs.

	Predicted True	Predicted False
Actual True	6,792	6,481
Actual False	3,365	468,120

Table 1: Confusion matrix of device match predictions of the validation set

7.4.1 Candidate selection

From the 17,476 and 17,356 pseudo-devices in the train and validation sets, a total number of over 150 million pairs are possible in each set. This is prohibitively large for practical purposes as the number of devices often exceed hundreds of thousands. To reduce the number of potential device pairs we apply the (kNN) algorithm on TF-IDF and Doc2Vec transformations of visited website URLs to filter out unlikely device pairs. This results in a reduction to 491,225 candidate pairs for the train set and 484,758 candidate pairs for the validation set. Unfortunately, the cost of the reduction in candidate pairs is that some matched pairs are lost in the candidate pairs which were filtered out. From the 13,456 and 13,273 matched pairs in the train and validation sets, the total number of matched pairs retained in the selected candidate pairs are 10,996 and 10,834, respectively. Overall, the initial candidate selection process is successful as 99.7% of the potential candidate pairs were removed while still recovering 81.7% of the true matched pairs.

7.4.2 Pairwise matching

The GBDT algorithm predicts a match probability for each of the candidate pairs in the validation set. To evaluate the quality of the predictions we rely on the following metrics: accuracy, precision, recall, and F_1-score. These metrics reflect how good the predictions are and they all rely on the confusion matrix which is a 2-by-2 table (Table 1) of the number of occurrences of:

- True positives (TP): Correctly predicting a match between two devices (cell (1, 1))
- True negatives (TN): Correctly predicting a non-match between two devices (cell (1, 2))
- False positives (FP): Wrongly predicting a match between two devices (cell (2, 1))
- False negatives (FN): Wrongly predicting a non-match between two devices (cell (2, 2))

Threshold	Accuracy	Precision	Recall	F_1	AUC
0.20	0.977	0.577	0.562	0.569	0.973
0.25	0.979	0.627	0.534	0.577	0.973
0.30	0.980	0.669	0.512	0.580	0.973
0.35	0.980	0.706	0.491	0.579	0.973
0.40	0.981	0.738	0.468	0.573	0.973

Table 2: Evaluation metrics on the validation set

Using the cells from Table 1 we compute the accuracy, precision, recall, and F_1-score as follows:

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \qquad Precision = \frac{TP}{TP+FP}$$
$$Recall = \frac{TP}{TP+FN} \qquad F_1 = 2 * \frac{Precision*Recall}{Precision+Recall}$$

Another evaluation metric is the area under the receiver operator characteristic curve (AUC). The AUC focuses on the relative ranking of predicted match probabilities of the candidate pairs, and can be interpreted as the probability that a randomly selected true match pair will have predicted match probability higher than a randomly selected true non-match pair.

The evaluation metrics are presented in Table 2. Note that other than AUC the evaluation metrics require a threshold to be set, which determines at which predicted probability the candidate pair is deemed to be a match versus non-match. Increasing the threshold causes a decrease in the predicted match/non-match ratio. This often has an indirect effect on the number of TP, TN, FP, and FN, and naturally affects the accuracy, precision, recall, and F_1-score. AUC is unaffected as it is based on the probabilities and Table 2 shows that with the thresholds set in a range from 0.20 to 0.40, accuracy and precision increases as threshold is raised, while recall decreases. When the threshold is increased the bar for predicted matches also increases, significantly decreases the number of FP and increases the number of TN, thus resulting in improved accuracy and precision. Conversely, increasing the bar for predicted matches also increases the number of FN which lowers recall. In practice, the metrics of precision and recall are inversely related and they are often of equal value to the business problem. The F_1-score is used to find the optimal balance between precision and recall and thereafter in this chapter we use 0.30 as our threshold of choice which results in the optimal F_1-score.

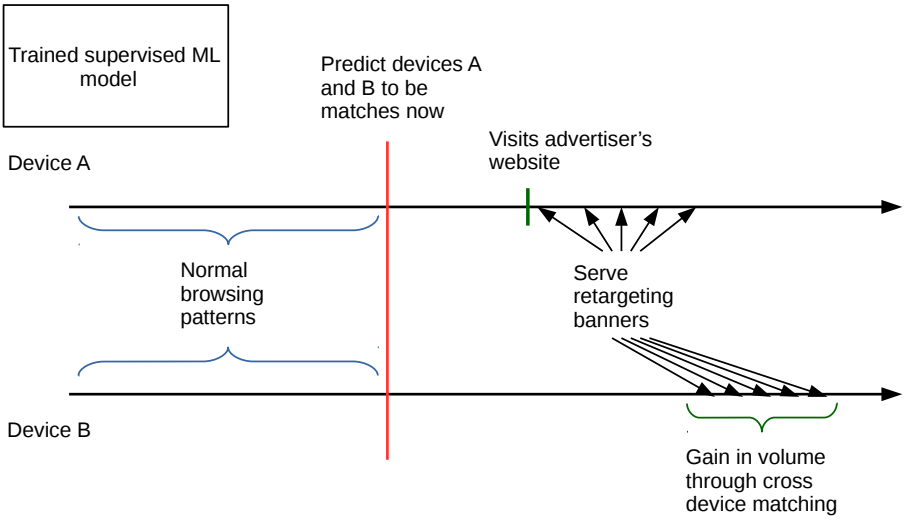


Figure 1: Schematic of retargeting matched devices

7.5 Economic impact of cross-device matching

The primary purpose of this chapter is to use cross-device matching to identify a wider audience for retargeting campaigns. To show the potential economic impact of matching devices, we consider the number of new devices that can be served retargeting banners and the number of banners that can be served as part of the matched devices that were not previously retargeting candidates. Figure 1 provides an example of the process for two devices. Let device A and device B be different devices belonging to the same user. Using one week of browsing data, we match device A and B using our machine learning framework. Afterward, when device A visits the advertiser's website we are able to subsequently serve retargeting banners not only to device A but also to device B which did not visit the advertiser's website, thus increasing the reach and effectiveness of the retargeting campaign.

To quantify this effect, we apply cross-device matching to devices that have visited a client of ORTEC Adscience. This client is the e-commerce division of a Dutch electronics store that focuses on selling cameras, which we will refer to as the "e-tailer". There are over 100,000 weekly unique visitors to this e-tailer, of which about

25,000 can be tracked through the ad-exchange. Devices that are not track-able by the ad-exchange are excluded from all consideration as they are unavailable for all display advertisements. These devices may be using browsers which are not track-able or employ ad-blockers.

With our machine learning framework already fully trained, we first extract three weeks of events data from all devices that have visited the e-tailer in the past two months. Next, we also extract three weeks of events data from all devices that have shared an IP address with devices that have visited the e-tailer, where the shared IP address cannot be shared by more than 5 devices to exclude public wifi hotspots. These devices are hereafter called "IP candidates". We assume that devices that have not shared an IP address with the e-tailer's visitors are not shared devices. While this assumption is strong, it prevents us from attempting to match an intractable number of devices. Moreover, the sharing of an IP address was found to be the most predictive feature for device matches (Kim et al 2016, Volkova 2017). In total, there are 156,196 unique past visitors of the e-tailer and 446,089 IP candidates over the two months. Using one week of browsing data we predict matches between all past visitors of the e-tailer with all IP candidates. This resulted in 164,396 matched pairs that exceeded the 0.3 match probability threshold.

We use the second and third weeks of browsing data to assess the volume of retargeting candidates. In the second week, we find 25,962 unique devices to visit the e-tailer that can be tracked on the ad-exchange. These devices made 32,350 total visits to the e-tailer and a total of 130,584 potential retargeting banners can be served to them—assuming that a device can be retargeted within a seven-day window of their most recent visit to the e-tailer and a highly conservative maximum of 5 retargeting banners per device. Of the 25,962 devices to visit the e-tailer, we find that 2,731 (10.5%) have matches to other devices (IP candidates) that did not visit the e-tailer. A total of 29,261 device matches are found for those original 2,731 devices and these matches are associated with 48,918 visits to the e-tailer, resulting in an additional 203,855 potential retargeting banners to be served following the same rules for the original visitors.

In total, with cross-device matching we are able to potentially reach an additional 113% more devices and serve 156% more retargeting banners for a given retargeting campaign. Following the precision score based on the validation dataset, we believe that 66.9% of the additionally reached devices are true matches to the original retargeting candidates and the other 33.1% are not. However, since device matching is performed based on similarity in browsing profiles, the devices that were wrongly matched can still be excellent targeting candidates as they share similar browsing profiles with visitors of the e-tailer. Therefore, even though they are not true retargeting candidates in the strict sense, the choice to target them may still be of great benefit to the advertiser.

7.6 Conclusion

In this chapter, we analyze the effect that predictive device matching could have on retargeting banner advertisements. First, we show how HTTP cookies can be matched based on user web browsing behavior information extracted from the ad-exchange. Relying on a combination of distance metrics and natural language processing techniques, we construct browsing patterns of HTTP cookies and make pairwise comparisons of the browsing patterns to determine whether two devices belong to the same user. Using a dataset of cookies that visited an e-commerce website and other cookies that shared an IP address with these visitors, we trained a supervised learning algorithm to predict device matches. Results showed that our approach is able to precisely identify device matches, although many potential matches are lost due to the initial filtering process. Finally, we show that the economic impact of cross-device matching can be substantial, leading to a potential increase of 75.4% in the number of unique retargetable devices and 104.4% in the number of retargeting advertisements served. Given the relative success of retargeting advertisements in driving customers to purchase from e-commerce websites, predictive device matching can significantly increase the overall advertising effectiveness of companies.

Chapter 8

Bibliography

Abe, N., Melville, P., Pendus, C., Reddy, C. K., Jensen, D. L., Thomas, V. P., Bennett, J. J., Anderson, G. F., Cooley, B. R., Kowalczyk, M., et al. (2010). Optimizing debt collections using constrained reinforcement learning. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 75–84. ACM.

Abhishek, V., Fader, P., and Hosanagar, K. (2015). Media exposure through the funnel: A model of multi-stage attribution. *Working paper*.

Adroll (2017). State of performance marketing 2017. Available on: `https://www.adroll.com/assets/pdfs/guides-and-reports/adroll-state-of-performance-marketing-17.pdf`.

Aksin, Z., Armony, M., and Mehrotra, V. (2007). The modern call center: A multi-disciplinary perspective on operations management research. *Production and Operations Management*, 16(6):665–688.

Al Hasan, M., Chaoji, V., Salem, S., and Zaki, M. (2006). Link prediction using supervised learning. In *SDM06: Workshop on Link Analysis, Counter-terrorism and Security*.

Anderl, E., Becker, I., Von Wangenheim, F., and Schumann, J. H. (2016). Mapping the customer journey: Lessons learned from graph-based online attribution modeling. *International Journal of Research in Marketing*, 33(3):457–474.

Ankenman, B., Nelson, B. L., and Staum, J. (2010). Stochastic kriging for simulation metamodeling. *Operations research*, 58(2):371–382.

Avramidis, A. N., Chan, W., Gendreau, M., LEcuyer, P., and Pisacane, O. (2010). Optimizing daily agent scheduling in a multiskill call center. *European Journal of Operational Research*, 200(3):822–832.

Barber, D. (2012). *Bayesian reasoning and machine learning*. Cambridge University Press.

Barton, R. R. (2015). Tutorial: simulation metamodeling. In *Proceedings of the 2015 Winter Simulation Conference*, pages 1765–1779.

Barton, R. R. and Meckesheimer, M. (2006). Metamodel-based simulation optimization. *Handbooks in Operations Research and Management Science*, 13:535–574.

Berman, R. (2018). Beyond the last touch: Attribution in online advertising. *Marketing Science*, 37(5):771–792.

Bertsekas, D. P. and Tsitsiklis, J. N. (1995). Neuro-dynamic programming: an overview. In *Proceedings of the 34th IEEE Conference on Decision and Control*, volume 1, pages 560–564. IEEE.

Bhulai, S. and Koole, G. (2003). A queueing model for call blending in call centers. *IEEE Transactions on Automatic Control*, 48(8):1434–1438.

Bhulai, S., Koole, G., and Pot, A. (2008). Simple methods for shift scheduling in multiskill call centers. *Manufacturing & Service Operations Management*, 10(3):411–420.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Bodur, M. and Luedtke, J. R. (2017). Mixed-integer rounding enhanced benders decomposition for multiclass service-system staffing and scheduling with arrival rate uncertainty. *Management Science*, 63(7):2073–2091.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Cai, H., Ren, K., Zhang, W., Malialis, K., Wang, J., Yu, Y., and Guo, D. (2017). Real-time bidding by reinforcement learning in display advertising. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 661–670. ACM.

Cao, X., Huang, W., and Yu, Y. (2015). Recovering cross-device connections via mining IP footprints with ensemble learning. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, pages 1681–1686. IEEE.

Cezik, M. T. and L'Ecuyer, P. (2008). Staffing multiskill call centers via linear programming and simulation. *Management Science*, 54(2):310–323.

Chandler-Pepelnjak, J. W. (2010). Modeling conversions in online advertising.

Chehrazi, N., Glynn, P., and Weber, T. A. (2018). Dynamic credit-collections optimization. *Management Science, Forthcoming*.

Chehrazi, N. and Weber, T. A. (2015). Dynamic valuation of delinquent credit-card accounts. *Management Science*, 61(12):3077–3096.

Chen, T. and Guestrin, C. (2016). XGBOOST: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM.

Coey, D. and Bailey, M. (2016). People and cookies: Imperfect treatment assignment in online experiments. In *Proceedings of the 25th International Conference on World Wide Web*, pages 1103–1111.

Cole, R. (1968). *Consumer and Commercial Credit Management (Third Edition)*. Richard D. Irwin; Nobleton: Irwin-Dorsey.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Cover, T. M. and Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

Crook, J. N., Edelman, D. B., and Thomas, L. C. (2007). Recent developments in consumer credit risk assessment. *European Journal of Operational Research*, 183(3):1447–1465.

Dalessandro, B., Perlich, C., Stitelman, O., and Provost, F. (2012). Causally motivated attribution for online advertising. In *Proceedings of the Sixth International Workshop on Data Mining for Online Advertising and Internet Economy*, page 7. ACM.

Danaher, P. J. and van Heerde, H. J. (2018). Delusion in attribution: Caveats in using attribution for multimedia budget allocation. *Journal of Marketing Research, forthcoming*.

Dasgupta, A., Gurevich, M., Zhang, L., Tseng, B., and Thomas, A. O. (2012). Overcoming browser cookie churn with clustering. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, pages 83–92. ACM.

De Almeida Filho, A. T., Mues, C., and Thomas, L. C. (2010). Optimizing the collections process in consumer credit. *Production and Operations Management*, 19(6):698–708.

De Haan, E., Wiesel, T., and Pauwels, K. (2016). The effectiveness of different forms of online advertising for purchase conversion in a multiple-channel attribution framework. *International Journal of Research in Marketing*, 33(3):491–507.

Díaz-Morales, R. (2015). Cross-device tracking: Matching devices and cookies. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, pages 1699–1704. IEEE.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International Workshop on Multiple Classifier Systems*, pages 1–15. Springer.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874.

Federal Reserve Bank of New York (Q1, 2018). Quarterly report on household debt and credit. Available on: `https://www.newyorkfed.org/microeconomics/hhdc.html`.

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer Series in Statistics New York, NY, USA:.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232.

Fukunaga, A., Hamilton, E., Fama, J., Andre, D., Matan, O., and Nourbakhsh, I. (2002). Staff scheduling for inbound call and customer contact centers. *AI Magazine*, 23(4):30.

Gans, N., Koole, G., and Mandelbaum, A. (2003). Telephone call centers: Tutorial, review, and research prospects. *Manufacturing & Service Operations Management*, 5(2):79–141.

Gastegger, M., Behler, J., and Marquetand, P. (2017). Machine learning molecular dynamics for the simulation of infrared spectra. *Chemical Science*, 8(10):6924–6935.

Ghose, A., Goldfarb, A., and Han, S. P. (2012). How is the mobile internet different? search costs and local activities. *Information Systems Research*, 24(3):613–631.

Ghose, A. and Han, S. P. (2011). An empirical analysis of user content generation and usage behavior on the mobile internet. *Management Science*, 57(9):1671–1691.

Global Industry Analysts, Inc. (April, 2018). Call centers - a global strategic business report. Available on: `https://www.strategyr.com/Call_Centers_Market_Report.asp`.

GlobalWebIndex (2016). Globalwebindex multi-device ownership. Available on: `https://blog.globalwebindex.com/chart-of-the-day/digital-consumers-own-3-64-connected-devices/`.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Google (2018). Data-driven attribution methodology. `https://support.google.com/analytics/answer/3191594?hl=en`. Accessed: 2018-07-25.

Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649. IEEE.

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Ingolfsson, A., Akhmetshina, E., Budge, S., Li, Y., and Wu, X. (2007). A survey and experimental comparison of service-level-approximation methods for non-stationary M(t)/M/s(t) queueing systems with exhaustive discipline. *INFORMS Journal on Computing*, 19(2):201–214.

Ingolfsson, A., Campello, F., Wu, X., and Cabral, E. (2010). Combining integer programming and the randomization method to schedule employees. *European Journal of Operational Research*, 202(1):153–163.

Interactive Advertising Bureau (2016). IAB internet advertising revenue report. Available on: `https://www.iab.com/wp-content/uploads/2016/04/IAB_Internet_Advertising_Revenue_Report_FY_2016.pdf`.

Ji, W., Wang, X., and Zhang, D. (2016). A probabilistic multi-touch attribution model for online advertising. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1373–1382. ACM.

Johnson, G. A., Lewis, R. A., and Nubbemeyer, E. I. (2017). Ghost ads: Improving the economics of measuring online ad effectiveness. *Journal of Marketing Research*, 54(6):867–884.

Juan, Y., Lefortier, D., and Chapelle, O. (2017). Field-aware factorization machines in a real-world online advertising system. In *Proceedings of the 26th International Conference on World Wide Web*, pages 680–688.

Kalai, E. and Samet, D. (1987). On weighted Shapley values. *International Journal of Game Theory*, 16(3):205–222.

Kamvar, M., Kellar, M., Patel, R., and Xu, Y. (2009). Computers and iPhones and mobile phones, oh my!: a logs-based comparison of search users on different devices. In *Proceedings of the 18th International Conference on World Wide Web*, pages 801–810. ACM.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3149–3157.

Kejela, G. and Rong, C. (2015). Cross-device consumer identification. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, pages 1687–1689. IEEE.

Kim, M. S., Liu, J., Wang, X., and Yang, W. (2015). Connecting devices to cookies via filtering, feature engineering, and boosting. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, pages 1690–1694. IEEE.

Kim, S., Kini, N., Pujara, J., Koh, E., and Getoor, L. (2017). Probabilistic visitor stitching on cross-device web logs. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1581–1589.

Kireyev, P., Pauwels, K., and Gupta, S. (2016). Do display ads influence search? attribution and dynamics in online advertising. *International Journal of Research in Marketing*, 33(3):475–490.

Kleijnen, J. P. (1975). A comment on Blanning's "Metamodel for sensitivity analysis: the regression metamodel in simulation". *Interfaces*, 5(3):21–23.

Koole, G. and Van Der Sluis, E. (2003). Optimal shift scheduling with a global service level constraint. *IIE Transactions*, 35(11):1049–1055.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Landry, M., Chong, R., et al. (2015). Multi-layer classification: ICDM 2015 Drawbridge cross-device connections competition. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, pages 1695–1698. IEEE.

Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Lessmann, S., Baesens, B., Seow, H.-V., and Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, 247(1):124–136.

Li, H. and Kannan, P. (2014). Attributing conversions in a multichannel online marketing environment: An empirical model and a field experiment. *Journal of Marketing Research*, 51(1):40–56.

Li, S., Wang, Q., and Koole, G. (2018). Optimal contact center staffing and scheduling with machine learning. *Working paper*.

Lian, J. and Xie, X. (2016). Cross-device user matching based on massive browse logs: The runner-up solution for the 2016 CIKM Cup. *arXiv preprint arXiv:1610.03928*.

Liben-Nowell, D. and Kleinberg, J. (2007). The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031.

Lichtenwalter, R. N., Lussier, J. T., and Chawla, N. V. (2010). New perspectives and methods in link prediction. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 243–252. ACM.

Liebman, L. H. (1972). A Markov decision model for selecting optimal credit control policies. *Management Science*, 18(10):B–519.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774.

Manchanda, P. (2015). Consumer search behavior on the mobile internet: An empirical analysis. *Working paper*.

Manning, C., Raghavan, P., and Schütze, H. (2010). Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103.

McMahan, H. B., Holt, G., Sculley, D., Young, M., Ebner, D., Grady, J., Nie, L., Phillips, T., Davydov, E., Golovin, D., et al. (2013). Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1222–1230. ACM.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Miller, G., Weatherwax, M., Gardinier, T., Abe, N., Melville, P., Pendus, C., Jensen, D., Reddy, C. K., Thomas, V., Bennett, J., et al. (2012). Tax collections optimization for New York state. *Interfaces*, 42(1):74–84.

Mitchner, M. and Peterson, R. P. (1957). An operations-research study of the collection of defaulted loans. *Operations Research*, 5(4):522–545.

Müller, H., Gove, J., and Webb, J. (2012). Understanding tablet use: A multi-method exploration. In *Proceedings of the 14th International Conference on Human-Computer Interaction with Mobile devices and Services*, pages 1–10. ACM.

Murphy, K. P. (2012). *Machine learning: A probabilistic perspective*. MIT press.

Ng, A. Y., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E., and Liang, E. (2006). Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics IX*, pages 363–372. Springer.

Nielsen, D. (2016). Tree boosting with XGBoost-why does XGBoost win "every" machine learning competition? Master's thesis, NTNU.

Nottorf, F. (2014). Modeling the clickstream across multiple online advertising channels using a binary logit with bayesian mixture of normals. *Electronic Commerce Research and Applications*, 13(1):45–55.

Olson, R. S., La Cava, W., Mustahsan, Z., Varik, A., and Moore, J. H. (2018). Data-driven advice for applying machine learning to bioinformatics problems. In *Pacific Symposium on Biocomputing*, volume 23, page 192. NIH Public Access.

Pang, L.-G., Zhou, K., Su, N., Petersen, H., Stöcker, H., and Wang, X.-N. (2018). An equation-of-state-meter of quantum chromodynamics transition from deep learning. *Nature Communications*, 9(1):210.

Patrick, J., Puterman, M. L., and Queyranne, M. (2008). Dynamic multipriority patient scheduling for a diagnostic resource. *Operations research*, 56(6):1507–1525.

Peeperkorn, P., Soomer, M., and Wang, Q. (2019). Multi-channel conversion attribution: A machine learning approach. *Working paper*.

Phan, M. C., Sun, A., and Tay, Y. (2017). Cross-device user linking: url, session, visiting time, and device-log embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 933–936. ACM.

Pot, A., Bhulai, S., and Koole, G. (2008). A simple staffing method for multiskill call centers. *Manufacturing & Service Operations Management*, 10(3):421–428.

Powell, W. B. and Topaloglu, H. (2006). Approximate dynamic programming for large-scale resource allocation problems. In *Models, Methods, and Applications for Innovative Decision Making*, pages 123–147. INFORMS.

Recode (2017). 2017 was the year digital ad spending finally beat tv. `https://www.recode.net/2017/12/4/16733460/2017-digital-ad-spend-advertising-beat-tv`. Accessed: 2018-09-07.

Ren, K., Zhang, W., Chang, K., Rong, Y., Yu, Y., and Wang, J. (2018). Bidding machine: Learning to bid for directly optimizing profits in display advertising. *IEEE Transactions on Knowledge and Data Engineering*, 30(4):645–659.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). Why should I trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386.

Roy, R. S., Sinha, R., Chhaya, N., and Saini, S. (2015). Probabilistic deduplication of anonymous web traffic. In *Proceedings of the 24th International Conference on World Wide Web*, pages 103–104. ACM.

Ruiz, L. M., Valenciano, F., and Zarzuelo, J. M. (1998). The family of least square values for transferable utility games. *Games and Economic Behavior*, 24(1-2):109–130.

Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1988). Learning representations by back-propagating errors. *Cognitive Modeling*, 5(3):1.

Sabuncuoglu, I. and Touhami, S. (2002). Simulation metamodelling with neural networks: an experimental investigation. *International Journal of Production Research*, 40(11):2483–2505.

Sahni, N. S., Narayanan, S., and Kalyanam, K. (2017). An experimental investigation of the effects of retargeted advertising: The role of frequency and timing. *Working paper*.

Shao, X. and Li, L. (2011). Data-driven multi-touch attribution models. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 258–264. ACM.

Shapley, L. S. (1953). A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Taskar, B., Wong, M.-F., Abbeel, P., and Koller, D. (2004). Link prediction in relational data. In *Advances in Neural Information Processing Systems*, pages 659–666.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

Tompson, J., Schlachter, K., Sprechmann, P., and Perlin, K. (2016). Accelerating Eulerian fluid simulation with convolutional networks. *arXiv preprint arXiv:1607.03597*.

Tossell, C., Kortum, P., Rahmati, A., Shepard, C., and Zhong, L. (2012). Characterizing web use on smartphones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2769–2778. ACM.

Tran, N. K. (2016). Classification and learning-to-rank approaches for cross-device matching at CIKM cup 2016. *arXiv preprint arXiv:1612.07117*.

van de Geer, R., Wang, Q., and Bhulai, S. (2018). Data-driven consumer debt collection via machine learning and approximate dynamic programming. *Working paper*.

van den Brink, R. (2002). An axiomatization of the shapley value using a fairness property. *International Journal of Game Theory*, 30(3):309–319.

van den Brink, R. and van der Laan, G. (1998). Axiomatizations of the normalized banzhaf value and the shapley value. *Social Choice and Welfare*, 15(4):567–582.

Vasile, F., Lefortier, D., and Chapelle, O. (2017). Cost-sensitive learning for utility optimization in online advertising auctions. In *Proceedings of the ADKDD'17*, page 8. ACM.

Volkova, E. (2017). Cross-device tracking with machine learning. Master's thesis.

Wang, J., Zhang, W., Yuan, S., et al. (2017). Display advertising with real-time bidding (RTB) and behavioural targeting. *Foundations and Trends® in Information Retrieval*, 11(4-5):297–435.

Wang, Q. (2017). Recombining customer journeys with probabilistic cookie matching: a supervised learning approach. *Working paper*.

Wang, Q. and Wijnsma, T. (2018). Improving display advertising with predictive device matching: A machine learning approach. *Working paper*.

Wiesel, T., Pauwels, K., and Arts, J. (2011). Practice prize papermarketing's profit impact: Quantifying online and off-line funnel progression. *Marketing Science*, 30(4):604–611.

Xu, K., Chan, J., Ghose, A., and Han, S. P. (2016). Battle of the channels: The impact of tablets on digital commerce. *Management Science*, 63(5):1469–1492.

Xu, L., Duan, J. A., and Whinston, A. (2014). Path to purchase: A mutually exciting point process model for online advertising and conversion. *Management Science*, 60(6):1392–1412.

Yan, X. and Su, X. (2009). *Linear regression analysis: Theory and computing*. World Scientific.

Young, H. P. (1985). Monotonic solutions of cooperative games. *International Journal of Game Theory*, 14(2):65–72.

Zhang, Y., Wei, Y., and Ren, J. (2014). Multi-touch attribution in online advertising with survival theory. In *2014 IEEE International Conference on Data Mining*, pages 687–696. IEEE.

Zimmeck, S. (2017). *Using machine learning to improve internet privacy*. PhD thesis, Columbia University.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.

Summary

With the rise of data collection also comes the need for methods to analyze the data, and for novel approaches of leveraging the data to improve decision-making. Machine learning is a set of algorithms with the primary purpose of identifying patterns in data to accomplish specific tasks. Since 2012, the number of Google searches containing the term machine learning has increased by 7 fold, and the data scientist, an occupation that often requires machine learning skills, has been rated by Glassdoor as the best job in America three years in a row. Academic publications in machine learning have also enjoyed a significant popularity boost. Although machine learning has been wildly popular, its scientific success has mostly been captured in the fields of computer science and artificial intelligence. The use of machine learning in business research, specifically in the areas of operations management and digital marketing, has been limited. In this dissertation, I study how machine learning can be used to solve prominent problems in operations management and digital marketing. The primary motivation is to show that the application of machine learning can solve problems in ways that existing approaches cannot. In its entirety, this dissertation is a study of four problems—two in operations management and two in digital marketing—and develops solutions to these problems via data-driven approaches by leveraging machine learning. These four problems are distinct, and are presented in the form of individual self-containing essays. Each essay is the result of collaborations with industry partners and is of academic and practical importance. In some cases, the solutions presented in this dissertation outperform existing state-of-the-art methods, and in other cases, it presents a solution when no reasonable alternatives are available.

Chapter 2 provides an introduction to the basics of machine learning. Chapter 3 studies the problem of consumer debt collection. For this problem, I develop a data-driven algorithm to optimize when and to whom phone calls should be made to maximize the collection of delinquent debt cases. This algorithm was tested in a controlled experiment at a Dutch collection agency and was found to have increased the amount of debt collected per call by 47.2%. Chapter 4 studies the problem of contact center staffing and scheduling. For this problem, I develop a machine learning approach to accurately approximate a complex simulation of contact centers, leading to a fast and reliable method for identifying high-quality staffing schedules at low costs. Using numerical simulations that represent real-life contact centers, it is found that my approach can improve upon the existing approaches by over 4%, and is able to analyze more complex contact centers than previously possible.

Chapter 5 studies the problem of attribution of online purchases to digital advertisements. For this problem, I develop a new attribution model that extends a well-known existing framework to incorporate customers' web-browsing behavior when evaluating the effectiveness of digital advertisements. Using data from a Dutch online travel agency, it is shown that customers' web-browsing behavior are highly predictive of purchasing decisions, and thus should be taken into account when attributing purchases. This solution is currently the only attribution model that is able to incorporate web-browsing behavior at the individual customer level. Finally, Chapters 6 and 7 studies the problem of probabilistically matching web-browsing devices (or browser cookies) to users based on browsing behavior. I consider two different instances of this problem, one of devices browsing a single news publishing website (Chapter 6) and another of devices captured by an ad exchange (Chapter 7), and develop solutions to them separately. In both cases, I show that matching can be performed with good reliability, and that display advertising firms can potentially use this technology to improve their advertising effectiveness.

All in all, I hope that this dissertation can help the reader better understand how machine learning can be used to solve operations management or digital marketing problems.

Samenvatting

Met de opkomst van dataverzameling ontstaat ook de behoefte voor methodes om de data te analyseren, en voor innovatieve benaderingen van het gebruik van de gegevens om de besluitvorming te verbeteren. Machine learning is een verzameling van algoritmes met als primair doel het identificeren van patronen in gegevens om specifieke taken uit te voeren. Sinds 2012 is het aantal Google-zoekopdrachten met de term machine learning met een factor 7 toegenomen en het beroep data scientist, dat vaak vaardigheden in machine-learning vereist, is door Glassdoor, voor het derde jaar op rij, beoordeeld als het beste beroep in Amerika. Academische publicaties in machine learning hebben ook een significante populariteits-groei genoten. Hoewel machine learning razend populair is geweest, is het wetenschappelijke succes ervan voornamelijk behaald op het gebied van computerwetenschap en kunstmatige intelligentie. Het gebruik van machine learning in bedrijfsonderzoek, met name op het gebied van operations management en digitale marketing, is tot nu toe beperkt gebleven. In dit proefschrift bestudeer ik hoe machine learning toegepast kan worden om prominente problemen in operations management en digitale marketing op te lossen. De primaire motivatie is om te laten zien dat men met de toepassing van machine learning problemen op kan lossen die bestaande methoden niet kunnen oplossen. Dit proefschrift is een studie van in totaal vier problemen—twee in operations management en twee in digital marketing—en dit proefschrift ontwikkeld oplossingen voor deze problemen via data-gedreven benaderingen met het gebruik van machine learning. Deze vier problemen zijn verschillend, en worden gepresenteerd in de vorm van opzichzelfstaande essays. Elke essay is het resultaat van samenwerkingen met partners uit de industrie, en zijn van praktisch en academisch belang. In sommige gevallen overtreffen de oplossingen in dit proefschrift de bestaande state-of-the-art-methoden en in andere gevallen bieden ze een oplossing wanneer er geen redelijke alternatieven beschikbaar zijn.

Hoofdstuk 2 biedt een introductie tot de basisprincipes van machine learning. Hoofdstuk 3 bestudeert het probleem van incasso van consumenten. Voor dit probleem ontwikkel ik een data-gedreven algoritme dat optimaliseert wanneer en met wie telefoongesprekken moeten worden gevoerd om het verzamelde schuldbedrag van delinquenten te maximaliseren. Dit algoritme is getoetst in een gecontroleerd experiment bij een Nederlands incassobureau en bleek een de hoeveelheid geïncasseerde schuld per oproep met 47,2% te hebben doen toenemen. Hoofdstuk 4 bestudeert het probleem van bezetting en planning van contactcentra. Voor dit probleem ontwikkel ik een machine learning benadering om de complexe situatie van call centers nauwkeurig te benaderen, wat leidt tot een snelle en betrouwbare methode voor

het identificeren van hoogwaardige personeelsplanning tegen lage kosten. Met het gebruik van numerieke simulaties die realistische call centers representeren, is het gebleken dat mijn aanpak de bestaande benaderingen met meer dan 4% kan verbeteren en in staat is om meer complexe call centers te analyseren dan voorheen mogelijk was.

Hoofdstuk 5 bestudeert het probleem van attributie van online aankopen aan digitale advertenties. Voor dit probleem ontwikkel ik een nieuw model dat een bekend bestaand kader voor het evalueren van de effectiviteit van digitale advertenties uitbreidt door het surfgedrag van klanten te integreren. Aan de hand van gegevens van een Nederlands online reisbureau wordt aangetoond dat het surfgedrag van klanten zeer goed voorspelbaar is bij aankoopbeslissingen en daarmee rekening moet worden gehouden bij attributie van aankopen. De oplossing is momenteel het enige model dat toewijzingen van aankopen aan advertenties doet, dat in staat is om surfgedrag van individuele klanten te incorporeren. Tenslotte, hoofdstukken 6 en 7 bestuderen het probleem van het probabilistisch vergelijken van mobiele apparaten (of browser cookies) voor gebruikers, op basis van surfgedrag. Ik beschouw twee verschillende voorbeelden van dit probleem, een waarbij ik mobiele apparaten beschouw die browsen op een enkele website voor nieuwsuitgeverijen (hoofdstuk 6) en een waarbij ik een mobiel apparaat beschouw dat is vastgelegd door een advertentie-uitwisseling (hoofdstuk 7). Ik ontwikkel voor beide aparte oplossingen en laat in beide gevallen zien dat matches met goede betrouwbaarheid kunnen worden gemaakt en dat display-reclamebureaus deze technologie mogelijk kunnen gebruiken om de effectiviteit van hun advertenties te verbeteren.

Al met al hoop ik dat dit proefschrift de lezer kan helpen om beter te begrijpen hoe machine learning kan worden gebruikt om problemen in operations management of digitale marketing op te lossen.