Essays on Algorithms for Customer Acquisition and Retention in SaaS Business Model

Ebrahim Barzegary

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2021

Reading Committee:

Hema Yoganarasimhan, Chair

Oliver J. Rutz

Simha Mummalaneni

Program Authorized to Offer Degree:
Foster School of Business

University of Washington

Abstract

Essays on Algorithms for Customer Acquisition and Retention in SaaS Business Model

Ebrahim Barzegary

Chair of the Supervisory Committee:
Michael G. Foster Professor Hema Yoganarasimhan
Marketing and International Business

In recent years, software firms have migrated from the traditional licensing business model to the "Software as a Service" (SaaS) model, where consumers subscribe to the software on monthly, quarterly, or annual contracts. This change has created new opportunities and challenges in the domain of customer acquisition and retention for firms. SaaS firms have access to an unprecedented amount of data since offering software as a service enables them to capture users' behavioral and contextual data at a granular level. Nevertheless, utilizing this amount of data requires a set of high-dimensional friendly tools and methodologies that may not be available.

In this dissertation, I try to address the challenge of high-dimensionality in modeling customer acquisition and retention in the SaaS business model. The first chapter, following the introduction, outlines high-dimensional data algorithms available to design and evaluate optimal free trials, the most commonly used customer acquisition strategy in SaaS. Using data from a field experiment, I showcase how companies can design optimal trial policies and understand the underlying mechanism for the effect of trial length on customer acquisition. In the second chapter, I discuss the problem of modeling customer retention as a dynamic discrete choice model. I offer a new algorithm that let researchers incorporate the high-dimensional usage data when modeling users' subscription decision. I run several simulations using the canonical bus engine replacement problem to show the performance of the proposed algorithm. Then, I discuss the limitations of the new algorithm and explain how researchers can use it to model customer retention in SaaS. The algorithm's source code is available on Github to be used and further developed for other applications.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to take this opportunity to express my gratitude to those who helped me to pursue my doctoral studies at the University of Washington. I wish to express my sincere appreciation to my Ph.D. advisor Professor Hema Yoganarasimhan. Her patience and continuous support empowered me to explore the research questions that interested me and encouraged me to think more independently. She has been my mentor and generously helped me in my hard days. I am grateful and inspired by her research enthusiasm, methodological rigor, and her dedication towards research.

I also want to thank other members of my advisory committee, Professor Oliver Rutz and Professor Simha Mummalaneni, for their insightful comments and support along the way. I would like to express my sincere gratitude to Professors Amin Sayedi and Natalie Mizik, who saw potentials in me and opened the door of Foster School of Business. I would like to express special thanks to Jaime Banaag and Beau Kirkeby who helped me patiently in different administrative processes and supported me like members of a caring family in tough times. I would also want to thank Omid Rafieian, Majid Majzoubi, Eugene Pavlov, Shahryar Doosti, Sareh Nabi, and Behnaz Bojd, who guided me like a mentor during my Ph.D. I want to especially thank my dear friend Bita Hajihashemi who helped me survive the pandemic and gave me motivation during the days I wanted to quit! Finally, I want to thank my friends in Mackenzie Hall: Evelyn Smith, Karo Solat, Amin Kazemi, Mily Wang, Jisu Kim, Mohammad Arbabian, Maria Mitkina, Amin Fazli, and Emisa Nategh. I am grateful for all the things you taught me and for making the Ph.D. years fun.

Finally, I would like to express my deepest gratitude to my family, whose unconditional love, patience, and support encouraged me to pursue my Ph.D. in long years far from them. They dedicated their lives to providing me with the opportunities they did not have. I would not be here today if it were not for their continuous support during my journey.

DEDICATION

to my beloved parents, Tahereh and Ahmad
&
to my dear brothers, Iman and Arman.

Chapter 1

INTRODUCTION

SaaS, or software as a service, is a business model in which software is licensed to customers via a subscription plan instead of perpetual licensing. Thanks to the power of cloud computing, the SaaS industry has grown immensely. The global SaaS business model end-user spending is forecasted to reach 145 billion dollars by 2022 [Gartner, 2021]. Today, the SaaS business model takes up a fair share of B2B offerings in the tech industry. In contrast to the exponential growth of SaaS, customer acquisition and retention in SaaS is a less investigated topic in the marketing literature. In this dissertation, I try to discuss algorithms for customer acquisition and retention in the SaaS business model, focusing on one of the biggest challenges of the model in the SaaS setting: the high-dimensionality of data.

This dissertation is composed of two main essays: i) *Design and evaluation of optimal free trials* and ii) *A recursive partitioning algorithm for dynamic discrete choice modeling in high-dimensional data settings.* Offering free trials is one of the most commonly used methods for customer acquisition in SaaS business model. In the first essay, I cover the customer acquisition part of the dissertation. I try to investigate the following research questions:

- How can we find the optimal free trial length?
- Does it worth it for companies to personalize the free trial length?
  - How can we design personalized free trial length assignment policies?
  - How can we evaluate free trial length assignment policies without implementing them?
- What are the factors that explain the heterogeneous effect of free trial length on post-trial subscriptions?
  - Which users benefit from longer trials and which users benefit from shorter trials?
  - Why does a more extended free trial increase the subscription rate for some users while it reduces the subscription rate for some other groups?

I answer these questions using data from a large field experiment, where more than 300,000 users were randomly assigned to get 7, 14, or 30 days of free trials. I use prediction and conditional treatment effect estimation algorithms (such as GRF) to design personalized free trial length assign-

ment policies. Next, I utilize the IPS reward estimation method to evaluate the performance of any trial length assignment policy using only the random experiment data. I compare the performance of policies designed by different algorithms and show that the best performing algorithm in my data can increase the post-trial subscription by 6.8%. I segment users based on their optimal trial length and highlight the differences between different segments. I use usage and dormancy length[1] as mediators to explain the underlying mechanism for the effect of free trial length on subscription. Finally, I include users' segments as moderators in my model to explain how this underlying mechanism works differently for different segments, i.e., why different trial lengths are optimal for different groups of users.

In my second essay, I address modeling customer retention in the SaaS business model. Modeling consumers' subscription decisions in the SaaS business model has two requirements that make it challenging.

1. Companies usually offer different subscription tiers, each of which has some limitations in terms of functionalities or features that the plan subscriber can use. Therefore, any modeling approach requires incorporating the usage characteristics of the subscription into the model. In addition, software usage is correlated with users' characteristics. Neglecting usage into modeling would bias the estimated coefficient. Nevertheless, usage characteristics can be very high-dimensional, especially for companies that offer a suite of software packages as a product, such as Microsoft 365.

2. Consumers are forward-looking in their subscription decisions to SaaS products. The forward-looking behavior is more prominent in settings where companies offer yearly contract discounts, where consumers' utility from the subscription changes over time, or in a B2B case where there is a high cost associated with teaching employees to use the new software. Software packages usually have a learning curve, and users get more utility as they become more experienced. Thus, any modeling approach that ignores the forward-looking nature of users' decisions would give biased estimates.

Dynamic discrete choice modeling approaches are used for modeling forward-looking decisions in consumers. These methods require calculating the value for being in each state in the space of the problem. Nevertheless, conventional approaches are not applicable in modeling users' subscription

[1]Dormancy length is the length between the last day that the software is used during the trial and the end of the trial. For example, if the last time the user lunches the software is on the 10th day of the trial, and the trial length is 30 days, the dormancy length would be 20 days.

decisions due to the dimensionality of the usage data. The time complexity of conventional methods grows exponentially with the addition of each variable, and therefore, using them is infeasible in high-dimensional settings. The second essay offers a novel two-step semi-parametric approach that enables researchers to estimate dynamic discrete choice models in such high-dimensional settings. I model the setting in which there are two sets of variables:

1. A low dimensional set of independent variables X: This is the set of features that researchers usually deal with in dynamic choice modeling. Researchers assume a functional form for the effect of these variables on the utility function. One can add variables such as price and promotion dummy to this set of features.
2. A high-dimensional set of independent variables Q: This is a set of features that researchers do not have any parametric assumptions on their effect on utility. We use a non-parametric approach for estimating the impact of this set of features. Usage features can be added to this set of features.

We crafted a recursive partitioning-based approach to discretize the high-dimensional variable set Q to a one-dimensional variable. The proposed method discretizes the high-dimensional variable set Q such that observations in the same partition behave similarly in terms of decisions and state transition probabilities. After the dimension reduction in the first step, researchers can use any conventional dynamic discrete choice modeling approach to model the variable set X and the discretized one-dimensional variable Q in the second step.

The proposed approach has several desirable properties that make it appealing, especially for estimation in the industry settings:

- It is insensitive to scale, irrelevant variables, and outliers.
- It is fast and scalable (can be parallelized).
- The time complexity of the algorithm is linear with respect to the dimension of Q.
- Flexible to be used with any other currently available method in the second stage.

I uploaded a well-documented version of the algorithm with a simple example on Github to be further developed and used by others.

While this dissertation is motivated and positioned for customer acquisition and retention in the SaaS business model, one can use the proposed algorithms and tools in other settings. The methodologies discussed in the first essay can be used to personalize any marketing intervention, such as promotions, direct mail marketing, and even advertising. The algorithm proposed in the second paper can solve the accuracy-feasibility trade-off that researchers encounter in any dynamic

discrete choice modeling. Researchers can control for variables that are not the main focus of the research project, and consequently, reduce the potential biases for the variables of interest using the proposed method in the second essay.

Chapter 2

DESIGN AND EVALUATION OF OPTIMAL FREE TRIALS

Free trial promotions are a commonly used customer acquisition strategy in the Software as a Service (SaaS) industry. We use data from a large-scale field experiment to study the effect of trial length on customer-level outcomes. We find that the 7-days trial is the best average treatment that maximizes customer acquisition, retention, and profitability. In terms of mechanism, we rule out the demand cannibalization theory, find support for the consumer learning hypothesis, and show that long stretches of inactivity at the end of the trial are associated with lower conversions. We then develop a framework for personalized targeting policy design and evaluation. We first learn a lasso model of outcomes as a function of users' pre-treatment variables and treatment. Next, we use individual-level predictions of the outcome to assign the optimal treatment to each user. We then evaluate the personalized policy using the inverse propensity score reward estimator. We find that a personalization based on lasso leads to 6.8% improvement in subscription compared to a uniform 30-days for all policy. It also performs well on long-term customer retention and revenues in our setting. Segmentation analysis suggests that skilled and experienced users are more likely to benefit from longer trials. Finally, we show that personalized policies do not always outperform uniform policies, and one should be careful when designing and evaluating personalized policies. In our setting, personalized policies based on other outcomes and heterogeneous-treatment effects, estimators (e.g., causal forests, random forests) perform worse than a simple 7-days for all policy.

2.1 Introduction

2.1.1 SaaS Business Model and Free Trials

Over the last few years, one of the big trends in the software industry has been the migration of software firms from the perpetual licensing business model to the "Software as a Service" (SaaS) model. In the SaaS model, the software is sold as a service, i.e., consumers can subscribe to the software based on monthly or annual contracts. Global revenues for the SaaS industry now exceed 200 billion USD [Gartner, 2019]. This shift in the business model has fundamentally changed the marketing and promotional activities of software firms. In particular, it has allowed firms to leverage

a new type of customer acquisition strategy: free trial promotions, where new users get a limited time to try the software for free.

Free trials are now almost universal in the SaaS industry because software is inherently *experience good*, and free trials allow consumers to try the software product without risk. However, we do not understand how long these trials should be or the exact mechanism through which they work. In the industry, we observe trial lengths ranging anywhere from one week to three months; e.g., Microsoft 365 offers a 30 days free trial, whereas Google's G Suite offers a 14 days free trial. There are pros and cons associated with both long and short trials. A short trial period is less likely to lead to free-riding or demand cannibalization and is associated with lower acquisition costs. On the other hand, an extended trial period can enhance consumer learning by giving consumers more time to learn about product features and functionalities. Longer trials can also create stickiness/engagement and increase switching-back costs. That said, if users do not use the product more with a longer trial, they are more likely to conclude that the product is not useful or forget about it. Thus, longer trials lack the deadline or urgency effect [Zhu et al., 2018].

While the above arguments make a global case for shorter/longer trials, the exact mechanism at work and the magnitude of its effect can be heterogeneous across consumers. In principle, if there is significant heterogeneity in consumers' response to the length of free trials, SaaS firms may benefit from assigning each consumer a different trial length depending on her/his demographics and skills. The idea of personalizing the length of free trial promotions is akin to third-degree price discrimination because we effectively offer different prices to different consumers over a fixed period. Indeed, SaaS free trials are particularly well-suited to personalization because of a few reasons. First, software services have zero marginal costs, and there are no direct cost implications of offering different trial lengths to different consumers. Second, it is easy to implement a personalized free trial policy at scale for digital services, unlike physical products. Finally, consumers are less likely to react adversely to receiving different trial lengths (unlike prices). However, it is unclear whether personalizing the length of free trials improves customer acquisition and firm revenues. If yes, what is the best approach to design and evaluate personalized free trials.

2.1.2 Research Agenda and Challenges

In this paper, we are interested in understanding the role of trial length on customer acquisition and profitability for digital experience goods. We focus on the following research questions. First, does the length of a free trial promotion affect customer acquisition, and if so, what is the ideal trial length? Second, what is the mechanism through which trial length affects conversions? Third, is

there heterogeneity in users' responsiveness to trial lengths? If yes, how can we personalize the assignment of trial lengths based on users' demographics and skills, and what are the gains from doing so? Further, what types of customers benefit from shorter trials vs. longer trials? Finally, how do personalized targeting policies that maximize short-run outcomes (i.e., customer acquisition) perform on long-run metrics such as consumer retention and revenue?

We face three main challenges in answering these questions. First, from a data perspective, we need a setting where trial length assignment is exogenous to user attributes. Further, to understand the mechanism through which trial length affects conversions, we need to observe usage and activity during the trial period. Second, from a methodological perspective, we need a framework to design and evaluate personalized targeting policies in high-dimensional settings and identify the optimal policy. A series of recent papers at the intersection of machine learning and causal inference provide heterogeneous treatment effects estimators, which we can use to personalize treatment assignment [Athey and Imbens, 2016, Wager and Athey, 2018]. Similarly, a series of papers in marketing have combined powerful predictive machine learning models with experimental (or quasi-experimental) data to develop personalized targeting policies [Rafieian and Yoganarasimhan, 2021, Simester et al., 2020]. However, the optimal policy that each of these papers/methods arrive at in a given empirical context can differ. Thus far, we have little to no understanding of how these methods compare in their ability to design effective targeting policies. This brings us to the third challenge. We need to be able to evaluate the performance of each policy *offline* (without deploying it in the field). Evaluation is essential because deploying a policy in the field to estimate its effectiveness is costly in time and money. Moreover, given the size of the policy space, it is simply not feasible to test each policy in the field.

2.1.3 Our Approach and Findings

To overcome these challenges and answer our research questions, we combine a three-pronged framework to design and evaluate personalized targeting policies with data from a large-scale free trial experiment conducted by a major SaaS firm. The firm sells a suite of related software products (e.g., Microsoft 365, Google G Suite) and is the leading player in its category, with close to monopolistic market power. At the time of this study, the firm used to give users a 30-days free trial for each of its software products, during which they had unlimited access to the software suite. Then, the firm conducted a large-scale field experiment, where new users who started a free trial for one of the firm's products were randomly assigned to one of 7, 14, or 30-days trial length conditions. It also monitored the subscription and retention decisions of the users in the experiment for two

years. The firm also collected data on users' pre-treatment characteristics (e.g., skill level and job) and post-treatment product usage during the trial period.

First, we quantify the average treatment effect of trial length on subscription. We find that the firm can do significantly better by simply assigning the 7-days trial to all consumers (which is the best uniform policy). This leads to a 5.59% gain in subscriptions over the baseline of 30 days for all policy in the test data. In contrast, the 14-days for all policy does not significantly increase subscriptions. This finding suggests that simply shortening the trial length to 7 days will lead to higher subscriptions. At the time of the experiment, the firm offered a standard 30-day free trial to all its consumers. So better performance of the much shorter 7-day trial was surprising, especially since the reasons proposed in the analytical literature for the efficacy of free trials mostly support longer trials, e.g., switching costs, consumer learning, software complexity, and signaling. (See §2.2 for a detailed discussion of the analytical literature on free trials). Therefore, we next examine the mechanism through which trial length affects conversion and present some evidence for why a shorter trial works better in this setting and examine the generalizability of these results. To that end, we leverage the usage data during the trial period to understand the mechanism through which trial length affects subscriptions. We show that there are two opposing effects of trial length. On the one hand, as trial length increases, product usage and consumer learning about the software increases. This increase in usage can have a positive effect on subscriptions. On the other hand, as trial length increases, the gap between the last active day and the end of the trial increases, while the average number of active days and usage per day reduces. These factors are associated with lower subscriptions. In our case, the latter effect dominates the former, and shorter trials are better.

Our analysis presents three key findings relevant to the theories on the role of free trials for experience goods. First, we rule out the demand cannibalization or free riding hypothesis advocated by many theoretical papers by showing that users who use the product more during the trial are more likely to subscribe [Cheng and Liu, 2012]. Second, we provide empirical support for the consumer learning hypothesis, since we show that longer trials lead to more usage, which in turn is associated with higher subscriptions [Dey et al., 2013]. Third, we identify a novel mechanism that plays a significant role in the effectiveness of free trials – the negative effect of long stretches of inactivity at the end of the trial on subscription.

Next, we develop a two-step approach to personalized policy design since an unstructured search for the optimal policy is not feasible in our high-dimensional setting. In the first stage, we learn a lasso model of outcomes (subscription) as a function of the users' pre-treatment demographic variables and their trial length. Then in the second stage, we use the individual-level predictions

of the outcome to assign the optimal treatment for each user. Then, we use the Inverse Propensity Score (IPS) reward estimator, popular in the counterfactual policy evaluation literature in computer science, for offline policy evaluation [Horvitz and Thompson, 1952, Dudík et al., 2011].

Based on this approach, we show that the personalized free trial policy leads to over 6.8% improvement in subscription compared to the baseline uniform policy of giving a 30-day trial for all. That said, the magnitude of gains from personalization (over the best uniform policy of 7 days for all) are modest (which is in line with the recent findings on personalization of marketing interventions in digital settings; e.g., Rafieian and Yoganarasimhan [2021]). Further, we find that customers' experience and skill level affect their usage, which affects their subscription patterns. Beginners and inexperienced users show only a small increase in usage with longer trial periods. Further, when given longer trials, they end up with long periods of inactivity at the end of the trial period, which negatively affects their likelihood of subscribing. Thus, it is better to give them short trials. In contrast, long trials are better for experienced users because it allows them to use the software more and they are not as negatively influenced by periods of inactivity later in the trial period. Overall, our findings suggest that simpler products and experienced users are more likely to benefit from longer trials.

Next, we find that the personalized policy, designed to optimize subscriptions, also performs well on long-term metrics, with a 7.96% increase in customer retention (as measured by subscription length) and 11.61% increase in revenues. We also consider two alternative personalized policies designed to maximize subscription length and revenues and compare their performance with that of the subscription-optimal policy. Interestingly, we find that the subscription-optimal policy always performs the best, even on long-run outcomes. While this finding is specific to this context, it nevertheless shows that optimizing low-variance intermediate outcomes (i.e., statistical surrogates) can be revenue- or loyalty-optimal in some settings.

Finally, we consider counterfactual policies based on four other outcome estimators: (1) linear regression, (2) CART, (3) random forests, and (4) XGBoost, and two heterogeneous treatment effect estimators: (1) causal tree, and (2) generalized random forests. We find our lasso-based personalized policy continues to perform the best, followed by the policy based on XGBoost (6.17% improvement). However, policies based on other outcome estimators (e.g., random forests, regressions) perform poorly. Interestingly, policies based on the recently developed heterogeneous treatment effects estimators (causal tree and causal forest) also perform poorly. Causal tree is unable to personalize the policy at all. Causal forest personalizes policy by a small amount, but the gains from doing so are marginal. While our findings are specific to this context, it nevertheless suggests

that naively using these methods to develop personalized targeting policies can lead to sub-optimal outcomes. This is particularly important since these methods are gaining traction in the marketing literature and are being used without evaluation using off-policy methods; see for example Guo et al. [2017] and Fong et al. [2019].

Our research makes three main contributions to the literature. First, from a substantive perspective, we present the first empirical study that establishes the causal effect of trial length on conversions and provides insight into the mechanisms at play. Second, from a methodological perspective, we present a framework that managers and researchers can use to design and evaluate personalized targeting strategies applicable to a broad range of marketing interventions. Finally, from a managerial perspective, we show that the policies designed to optimize short-run conversions also perform well on long-run outcomes in our setting, and may be worth considering in other similar settings. Importantly, managers should recognize that many popular estimators can give rise to poorly designed personalized policies, which are no better than simple uniform policies. Offline policy evaluation is thus a critical step before implementing any policy.

2.2 Related Literature

Our paper relates to the research that examines the effectiveness of free trials on the purchase of experience goods, especially digital and software products. Analytical papers in this area have proposed a multitude of theories capturing the pros and cons of offering free trials. Mechanisms such as switching costs, network effects, quality signaling, and consumer learning are often proposed as reasons for offering free trials. In contrast, free-riding and demand cannibalization are offered as reasons against offering free trials. See Cheng and Liu [2012], Dey et al. [2013], and Wang and Özkan-Seely [2018] for further details. In spite of this rich theory literature, very few empirical papers have examined whether and how free trials work in practice. In an early paper, Scott [1976] uses a small field experiment to examine if users given a two-week free trial are more likely to purchase a newspaper subscription. Interestingly, she finds that free trials do not lead to more subscriptions compared to the control condition. While the number of participants may not have been sufficient to detect small effects and the context was very different from digital SaaS products, it nevertheless raises the question of whether free trials can be an effective marketing strategy. More recently, two empirical papers study free trials using observational data. Foubert and Gijsbrechts [2016] build a model of consumer learning and show that while free trials can enhance adoption, ill-timed free trials can also suppress adoption. Using a bayesian learning approach, Sunada [2018] compares the profitability of different free trial configurations. However, neither of these papers

examines how trial length affects subscriptions/revenues because they lack variation in the length of the free trials in their data. In contrast, we use data from a large-scale field experiment with exogenous variation in the length of free trials to identify the optimal trial length for each user. In addition, we contribute to this literature by leveraging the individual-level software usage data during the trial period to rule out some of the earlier theories proposed in this context, e.g., free riding. To the best of our knowledge, our paper provides the first comprehensive empirical analysis of how trial length affects the purchase of digital experience goods.

Second, our paper relates to the marketing literature on real-time customization and personalization of digital products and promotions using machine learning methods. This literature has used a wide range of methods for the personalization tasks in a variety of contexts: website design using dynamic programming and adaptive experiments [Hauser et al., 2009], display ads using multi-arm bandits [Schwartz et al., 2017], ranking of search engine results using feature engineering, and boosted trees [Yoganarasimhan, 2020], mobile ads using behavioral and contextual features [Rafieian and Yoganarasimhan, 2021], and the sequence of ads shown in mobile apps using batch reinforcement learning and optimal dynamic auctions [Rafieian, 2019a,b]. We add to this literature in two ways. First, we document the gains from personalizing the duration of a new type of promotional strategy: the length of time-limited free trials for digital experience goods using a targeting framework based on data from a large-scale field experiment. Second, we show that while personalization can help, it may not always be the case. Indeed, in our setting, many commonly used methods for personalization often perform worse than a robust uniform policy based on average treatment effects. While these findings are specific to our context, it nevertheless suggests that managers should be careful in designing and evaluating personalized targeting policies.

Our paper also relates to the theoretical and empirical research on personalized policy design and evaluation in computer science and economics. In an early theoretical paper, Manski [2004] presents a method that finds the optimal treatment for each observation by minimizing a regret function. Recent theoretical papers in this area include Swaminathan and Joachims [2015a] and Swaminathan et al. [2017], Kitagawa and Tetenov [2018], and Athey and Wager [2017]. There is also a small but growing list of marketing papers in this area. Hitsch and Misra [2018] propose a heterogeneous treatment effects estimator based on kNN, develop targeting policies based on it, and evaluate the performance of their policies using the IPS estimator on a test data. However, their estimator does not work in our setting because it requires all the covariates to be continuous since it based on Euclidean distance. Simester et al. [2020] examine how managers can evaluate targeting policies efficiently. They compare two types of randomization approaches: (a) randomization by action and

(b) randomization by policy. The provide two valuable insights. First, they note that randomization by action is preferable to randomization by policy because it allows us use off policy evaluation to evaluate any policy. Second, they note that when comparing two policies we should recognize that if the policies recommend the same action for some customers then the difference in the performance of the policy for these customers is exactly zero. In another particularly relevant paper, Simester et al. [2019] investigate how data from field experiments can be used to design targeting policies for new customers or new regimes, and also use the IPS estimator to evaluate the peformance of a series of personalized policies. They present comparisons for a broad range of methods and show that model-based methods in general (and lasso in particular) offers the best performance, though this advantage vanishes if the setting and/or consumers change significantly. Our paper also echoes this finding: the lasso-based personalized policy performs the best in our setting too. Further, we also provide comparisons to personalized policies based on the newly proposed heterogeneous treatment effects estimators (e.g., causal forest), and show that the lasso-based policy continues to perform the best.

Our paper is relevant to the literature on statistical surrogates [Prentice, 1989, VanderWeele, 2013]. In our setting, subscription can be interpreted as an intermediate outcome or surrogate for long-run retention and revenue. Interestingly, we find that personalized policies optimized on the short-term outcome or surrogate do well (or better than) policies optimized directly on the long-term outcomes. We attribute this to the fact that long-term outcomes have higher variance and fewer observations in our setting. In a recent paper, Yang et al. [2020] use surrogates to impute the missing long-term outcomes and then use the imputed long-term outcomes to develop targeting policies. Their results confirm our broader finding that short-term outcomes can be sufficient to derive targeting policies that are optimal from a long-run perspective.

More broadly, our work relates to the large stream of marketing literature that has examined and contrasted the short vs. long run effects of promotions [Mela et al., 1997, Pauwels et al., 2002]. The main takeaway from this literature is that consumers who are exposed to frequent price promotions become price sensitive and engage in forward buying over time. While these early papers focused on consumer packaged goods, Anderson and Simester [2004] conduct a field experiment on price promotions in the context of durable goods sold through catalogs. They find evidence in support of both forward-buying and increased deal sensitivity. Our paper adds to this literature by examining the long-run effect of free-trial promotions on long-run subscription and revenue for digital SaaS products. While free-trial promotions can be viewed as a price discount (i.e., zero price for a fixed period), forward-buying is not feasible in our setting and consumers are exposed to the promotion

only during the sign-up period (i.e., no expectation of future free trials). In this case, we find that targeted free-trial promotions that maximize short-run revenue (or subscriptions) also perform well on long-run outcomes (two-year revenue).

2.3 Setting and Data

In this section, we describe our application setting and data.

2.3.1 Setting

Our data come from a major SaaS firm that sells a suite of software products. The suite includes a set of related software products (similar to Excel, Word, PowerPoint in Microsoft's MS-Office). The firm is the leading player in its category, with close to monopolistic market power. Users can either subscribe to single-product plans that allow them access to one software product or to bundled plans that allow them to use several products at the same time. Bundles are designed to target specific segments of consumers and consist of a set of complementary products. The prices of the plans vary significantly and depend on the bundle, the type of subscription (regular or educational), and the length of commitment (monthly or annual). Standard subscriptions run from $\$30$ to $\$140$ per month depending on the products in the bundle and come with a monthly renewal option. (To preserve the firm's anonymity, we have multiplied all the dollar values in the paper by a constant number.) If the user is willing to commit to an annual subscription, they receive over 30% discount in price. However, users in annual contracts have to pay a sizable penalty to unsubscribe before the end of their commitment. The firm also offers educational licenses at a discounted rate to students and educational institutions, and these constitute 20.8% of the subscriptions in our data.

2.3.2 Field Experiment

At the time of this study, the firm used to give users a 30-day free trial for each of its software products, during which they had unlimited access to the product.[1] In order to access the product after the trial period, users need a subscription to a plan or bundle that includes that product.

To evaluate the effectiveness of different trial lengths, the firm conducted a large-scale field experiment that ran from December $1^{st}2015$ to January $6^{th}2016$ and spanned six major geographic markets – Australia and New Zealand, France, Germany, Japan, United Kingdom, and United States

[1]This free trial is at the software product level, i.e., users start a separate trial for each software product, and their trial for a given product expires 30 days from the point at which they started the free trial for it.

	7 Days Trial	14 Days Trial	30 Days Trial	Total
Number of observations (N)	51,040	51,017	235,667	337,724
Percent of total observations	15.11	15.11	69.78	100
Number of subscriptions	7,835	7,635	34,564	50,034
Percent of total subscriptions	15.66	15.26	69.08	100
Subscription rate within group (in %)	15.36	14.96	14.67	14.81

Table 2.1: Summary statistics of treatment assignment and subscription rates.

of America. During the experiment period, users who started a free trial for any of the firm's four most popular products were randomly assigned to one of 7, 14 or 30 days free trial length buckets. These three trial lengths were chosen because they are the most commonly used ones in the industry and represent a vast majority of the SaaS free trials. Treatment assignment was at user level, i.e., once a user was assigned to a trial length, her/his trial length for the other three popular products was also set at the same length. The length of the free trial for other products during this period remained unchanged at 30 days. The summary statistics for the treatment assignment and subscriptions are shown in Table 2.1.

The experiment was carefully designed and implemented to rule out the possibility of self-selection into treatments, a common problem in field experiments. In our setting, if users can see which treatment (or free trial length) they are assigned to prior to starting their trial, then users who find their treatment undesirable may choose to not start the trial. In that case, the observed sample of users in each treatment condition would no longer be random, and this in turn would bias the estimated treatment effects. Moreover, since the experimenter cannot obtain data on those who choose to not to start their free trials, there is no way to address this problem econometrically. To avoid these types of self-selection problems, the firm designed the experiment so that users were informed of their trial-length only after starting their trial. In order to try a software product, users had to take the following steps: (1) sign up with the firm by creating an ID, (2) download an app manager that manages the download and installation of all the firm's products, and (3) click on an embedded *start trial* button to start the trial for a given product. Only at this point in time, they are shown the length of their free trial as the time left before their trial expires (e.g., "Your free trial expires in 7 days"). While users can simply quit or choose to not use the product at this point, their identities and actions are nevertheless captured in our data and incorporated in our analysis.

Finally, it is important to note that treatment assignment was unconfounded with other marketing mix variables. In this context, it is useful to discuss prices since they can vary across products and

Variable	Number of sub-categories	Share of top sub-categories			
		1^{st}	2^{nd}	3^{rd}	4^{th}
Geographic region	6	55.02%	13.66%	9.12%	8.83%
Operating system	8	28.97%	21.4%	14.04%	13.98%
Signup channel	42	81.56%	8.14%	3.47%	0.81%
Job	14	28.20%	21.90%	20.34%	8.46%
Skill	5	69.05%	12.75%	10.77%	7.38%
Business segment	7	35.41%	32.74%	18.40%	7.81%

Table 2.2: Summary statistics for the pre-treatment categorical variables.

users. The price that a user gets for a product/bundle depends only on two user-level observables – the geographic location of the user and her/his job (students get a discount). Both of these variables are observed in the data, and treatment assignment is orthogonal to these variables. Thus, price is not confounded with treatment.[2]

In sum, the design satisfies the two main conditions necessary for the experiment to be deemed clean – (1) unconfoundedness and (2) compliance [Mutz et al., 2019].

2.3.3 Data

We have data on 337,724 users who were part of the experiment. For each user i, we observe the following information – (1) Treatment assignment (W_i), (2) Pre-treatment demographic data (X_i), and (3) Post-treatment behavioral data (Z_i). The treatment variable denotes the trial length that the user was assigned to – 7, 14, or 30 days. The variables under the latter two categories are described in detail below.

Pre-treatment demographic data

1. Geographic region: The geographic region/country that the user belongs to (one of the six described in §2.3.1). It is automatically inferred from the user's IP address.

2. Operating system: The OS installed on the user's computer. It can take eight possible values,

[2]While the distribution of prices shown to users is the same across all treatment arms, the distribution of prices paid by the subscribers can be different under each treatment arm. This is because the length of the free trial can influence which types of consumers subscribe and the products/bundles that they subscribe to. These differences can lead to downstream differences in the revenues under treatments and targeting policies. We discuss these issues in detail in §2.7.

e.g., Windows 7, Mac OS Yosemite. It is inferred by the firm based on the compatibility of the products downloaded with the user's OS.

3. Sign-up channel: The channel through which users came to sign-up for the free trial. In total, there are 42 possible sign-up channels, e.g., from the legacy version of the software, from the firm's website, through third-parties, and so on.

4. Skill: A self-reported measure of the user's fluency in using the firm's software suite. This can take four possible values – beginner, intermediate, experienced, and mixed.

5. Job: The user's job-title (self-reported). The firm gives users 13 job-titles to pick from, e.g., student, business professional, hobbyist.

6. Business segment: The self-reported business segment that the user belongs to. Users can choose from six options here, e.g., educational institution, individual, enterprise.

The last three variables are self-reported though not open-ended, i.e., users are required to pick one option from a list provided by the firm. However, users may choose not to report these values, in which case, the missing values are recorded as "unknown". We treat this as an additional category for each of these three variables in our analysis.[3] The six pre-treatment variables and their summary statistics are shown in Table 2.2.

Post-treatment behavioral data

For all the users in our data, we observe their subscription and renewal decisions for approximately 24 months (from December 2015 till November 2017). We have data on:

1. Subscription information: We have data on whether a user subscribes or not, and the date and type of subscription (product or bundle of products) if she does subscribe.

2. Subscription length: Number of months that the user is a subscriber of one or more products/bundles during the 24-month observation period. If a user does not subscribe to any of

[3]Only a small fraction of people choose to not report these data. For example, the percentage of users with "unknown" Skill and Job is 7.4% and 21.9%, respectively.

Variable	Mean	Standard Deviation	Min	25%	50%	75%	Max	Number of Observations
Subscription	0.148	0.355	0	0	0	0	1	337,724
Subscription length (all)	2.23	6.37	0	0	0	0	108	334,223
Subscription length (subscribers)	16.02	8.43	0	10	17	22	108	46,533
Revenue (all)	79.13	285	0	0	0	0	20,208	334,223
Revenue (subscribers)	568	552	0	242	420	666	20,208	46,533

Table 2.3: Summary statistics of subscription, subscription length, and revenue outcomes. All the revenue numbers are scaled by a constant factor to preserve the firm's anonymity.

the firm's products during the observation period, then this number is zero by default.[4]

3. Revenue: The total revenue (in scaled dollars) generated by the user over the 2-year observation period. This is a function of the user's subscription date, the products and/or bundles that she subscribes to, the price that she pays for her subscription, and subscription length.

The summary statistics of these outcome variables are presented in Table 2.3. Both subscription length and revenue are shown for: (a) all users and (b) the subset of users who subscribed. There are a couple of points to note here. First, we do not have access to the subscription length and revenue data for team subscriptions and government subscriptions (which constitute a total of 3501 subscriptions). Hence, the number of observations used to calculate the summary statistics for subscription length and revenue for subscribers is lower. Second, the minimum subscription length observed in the data for subscribers is zero because we have a few users (58 users) who immediately unsubscribed after subscribing (within one month), in which case the firm returns their money and records their subscription length and revenue as zero. Based on Table 2.3, we see that approximately 14.8% users who start a free trial subscribe, and the average subscription length of subscribers is about 16 months (which is a little over a year).

We also observe the following product download and usage data for the duration of a user's trial period.

1. Products downloaded: The date and time-stamp of each product downloaded by the user.

2. Indicator for software use: An indicator for whether the user used the software at all.

[4]If a user unsubscribes for a period of time and then comes back, her subscription length is the total number of months that she was a paying customer of the firm. If a user subscribes to two or more plans, we aggregate the length of subscription all plans and report the total. So the subscription length can be greater than 24 months for such users.

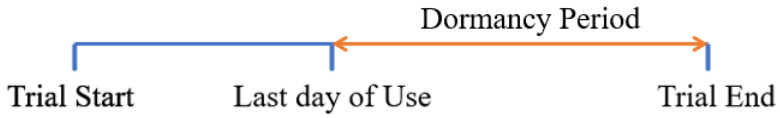


Figure 2.1: Dormancy length: Number of days between the last active day and the end of the trial period.

3. Number of active days: Total number of days in which the user used the software during the trial period. For example, if a user with a 7-day trial uses the software on the first and third day, this variable is two.

4. Usage during trial: Each product in the software suite has thousands of functionalities. Functionalities can be thought of as micro-tasks and are defined at the click and key-stroke level; e.g., save a file, click undo, and create a table. The firm captures all this information and we have data on the total count of the functionalities used by the user during her trial period.

5. Dormancy length: The number of days between the last active day and the last day of trial, as shown in Figure 2.1. For example, if a user with a 30-days trial last used the software on day 20, then her dormancy length is 10.

We present the summary statistics of these usage variables in Table 2.4. The usage data are also missing (at random) for a subset of users and we report the summary statistics for non-missing observations. As we can see, most users download only one software product; only 13.6% of people download more than one product. Further, 83% of users try the software at least once. However, the number of active days is relatively small; the average user uses the software for only three days during the trial period. Next, we see that an average user uses $1,733$ functionalities during her trial. However, notice that this variable is very skewed with the variance much higher than the mean. So we use the natural log of this variable in all our analyses going forward. Finally, we see that the average dormancy length is close to 17 days, which means that many users stop using the software much before the end of trial period.

Finally, we refer interested readers to Tables A.1 and A.2 in Appendix A.1 for the summary statistics of outcome and usage variables by trial length, respectively.

Variable	Mean	Std	Min	25%	50%	75%	Max	N
Total downloaded packages	1.17	0.41	1.0	1.00	1.00	1.00	4.00	337,724
Indicator for software use	0.83	0.37	0.0	1.00	1.00	1.00	1.00	303,514
Number of active days	3.03	3.94	0.0	1.00	2.00	4.00	30.00	303,514
Usage during trial	1,733	7,220	0	47	257	1,086	488,666	303,514
Log usage during trial	5.09	2.74	0.0	3.87	5.55	6.99	13.10	303,514
Dormancy length	16.87	11.23	0.0	6.00	15.00	29.00	30.00	303,514

Table 2.4: Summary statistics for usage features.

Training and Test Data

To design and test counterfactual free trial policies, we partition the data into two independent samples.

- **Training Data:** This is the data that is used for both learning the model parameters as well as model selection (or hyper-parameter optimization through cross-validation).
- **Test Data:** This is a hold-out data on which we can evaluate the performance of the policies designed based on the models built on training data.

We use 70% of the data for training (and validation) and 30% for test. See Table A.3 in the Appendix §A.1 for a detailed breakdown of how the data are split across the two data-sets. Note that while the joint distributions of the variables in the two samples should be the same theoretically, there will be some minor differences between the two data-sets due to the randomness in splitting in a finite sample. It is important to keep this in mind when comparing results *across* the two data-sets.

2.4 Main Effect of Trial Length on Subscription

We now document the main effect of trial length on subscription and present some evidence for the mechanism behind this effect. For expositional simplicity, we focus on subscription here and present a detailed analysis long-run outcomes such as revenue and retention in §2.7.

2.4.1 Average Treatment Effect

In a fully randomized experiment (such as ours), the average effect of a treatment can be estimated by simply comparing the average of the outcome of interest across treatments. We set the 30-day condition as the control and estimate the average effects of the 14- and 7-days trials on subscriptions for training and test data. The results from this analysis are shown in Table 2.5.

Data	Treatment	Subscription rate	Subscription rate difference	t-statistics	Percentage gain over baseline
Training data	7 days	0.1532	0.0064	3.08	4.34
	14 days	0.1490	0.0021	1.03	1.45
	30 days	0.1468	—	—	—
Test data	7 days	0.1544	0.0082	2.58	5.59
	14 days	0.1511	0.0048	1.51	3.28
	30 days	0.1463	—	—	—

Table 2.5: Average effect of the 7- and 14-day treatments on subscription; compared to the control condition of 30-day free-trial. Baseline subscription rate (for 30-day case): 14.68 in training data and 14.63 in test data.

The 7-day trial increases the subscription rate by 4.34% over the baseline of the 30-day condition in the training data and by 5.59% in the test data. However, in both data sets, the effect of the 14-day trial is not significantly different from that of the 30-day trial. These results suggest that a uniform targeting policy that gives the 7-day treatment to all users can significantly increase subscriptions.[5] We also see that the average treatment effect is fairly small compared to the outcome, which is either zero or one. This is understandable since the effect of trial length is likely to be small compared to other factors that affect customer acquisition. Finally, note that the gains and subscription rates in the training and test data are slightly different. As discussed earlier, this is due to the randomness in the splitting procedure.

Next, to ensure that these results are not driven by any problems with randomization, we conduct a series of randomization checks. We present the details of these tests in Appendix §A.2.2 and discuss them briefly here. First, we conduct a joint test of orthogonality of pre-treatment variables and treatment assignment [McKenzie, 2017]. This is done by regressing the treatment variable on the entire set of pre-treatment variables (with dummies for each sub-category shown in Table 2.2). We find that the pre-treatment variables have no predictive power when it comes to predicting treatment, which suggests that randomization was done correctly. Note that this approach to checking for potential issues with randomization is preferable to the old practice of showing tables of means for pre-treatment variables across treatment arms and running a battery of t-tests for a variety of reasons; see Bruhn and McKenzie [2009] and Mutz et al. [2019] for detailed discussions.[6] Next, we regress

[5] In general, it is a better practice to obtain ATEs directly based on mean comparisons without using regression-based approaches [Imbens and Rubin, 2015]. Nevertheless, in Appendix A.2.1, we present the ATEs based on regressions (with and without controls) and they are statistically indistinguishable from those shown in the main text.

[6] Further, presenting tables of means for each pre-treatment variable is not feasible in our case since all our

Outcome variable	Intercept	14 days trial	30 days trial	R^2	N
Total downloaded packages	1.137 (0.002)	0.01 (0.002)	0.017 (0.002)	0.000	337724
Indicator for software use	0.828 (0.002)	0.004 (0.002)	0.009 (0.002)	0.000	303514
Number of active days	1.747 (0.018)	0.625 (0.026)	1.711 (0.02)	0.028	303514
Number of active days/trial length	0.25 (0.001)	-0.08 (0.001)	-0.134 (0.001)	0.078	303514
Log usage during trial	4.77 (0.013)	0.196 (0.018)	0.411 (0.014)	0.003	303514
Log average daily usage during trial	3.197 (0.009)	-0.357 (0.012)	-0.737 (0.009)	0.022	303514
Dormancy length	4.631 (0.043)	5.135 (0.06)	16.432 (0.047)	0.337	303514

Table 2.6: Regression of usage features on trial length. Standard errors in parentheses.

the outcome variable (subscription outcome) on the treatment variable and all the pre-treatment variables. We find that the treatment effects are very similar to those in Table 2.5, which again suggests that there are no issues with randomization.[7]

2.4.2 Mechanism

At the time of the experiment, the firm offered a standard 30-day free trial to all its consumers. The better performance of the much shorter 7-day trial was both surprising and inexplicable for many reasons. First, the firm sells a complicated suite of software with multiple products and functionalities. So we would have expected that giving consumers more time to familiarize themselves with it and learn the software would produce better outcomes. Second, the reasons proposed in the theory literature for the efficacy of free trials largely support longer free trials, e.g., switching costs, consumer learning, software complexity, and signaling. Thus, it is not obvious why a shorter trial works better. Therefore, we now examine how trial length affects conversion and present some evidence for why a shorter trial works better in this setting. In the process, we also discuss the generalizability of our findings and the mechanisms proposed.

Intuitively, trial length can affect how consumers download, use, and interact with the software; and differences in these usage variables can lead to different subscription outcomes. So we first examine whether and how trial length affects usage. We regress each of the usage variables shown in Table 2.4 on trial length and present the results in Table 2.6. Since trial length is randomly assigned,

pre-treatment variables are categorical with a large number of sub-categories.

[7]Later in the paper, we use the empirical propensities to evaluate the gains from our models. So any minor discrepancies in the propensities of treatment allocation are taken care of; see Equation (2.4) and the discussion around it.

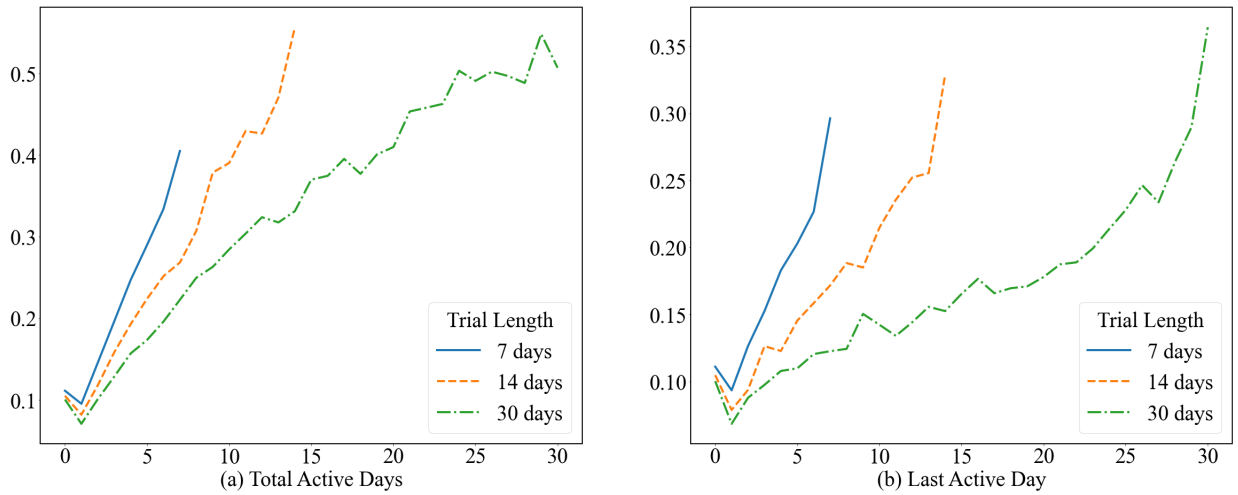


Figure 2.2: (a) The subscription rate based on the last day of trial use for different trial lengths. (b) The subscription rate based on the number of active days for different trial lengths.

we can interpret these results causally. First, we find that longer free trials lead to more product downloads and more usage. Further investigation suggests that this increase in downloads mainly comes from the higher downloads of products 1 and 3, which are complements (see Figure A.1 in Appendix A.3). This suggests that giving longer trial lengths to users increases their probability of exploring other complementary products. Next, we see that a larger fraction of people try the software at least once with a longer trial, and the number of active days and log usage also increases with trial length. However, the rate of increase in the number of active days and usage is sub-linear compared to the increase in trial length. For instance, going from 7 to 14 days increases the number of active days by 0.625, which is much smaller than 7 days (the increase in trial length). Thus, when we normalize the number of active days by trial length, the average number of days during which a user is active during her trial reduces as trial length increases. The same pattern holds for log usage; while total usage increases as trial length increases, average daily usage falls. Finally, we find that the dormancy period increases as trial length increases. While the average dormancy length is 4.6 days for the 7-days trial, it is over 21 days for the 30-days trial.

Next, we examine whether and how usage is associated with subscription. The left panel of Figure 2.2 shows the probability of subscription as a function of the total number of active days for each trial length. As we can see, users who are active for more days are also more likely to subscribe, and this pattern is true for all three trial lengths. However, given the same level of activity, shorter trial lengths are associated with higher conversion. For example, users who were active for

	coef	std err	z	$P > \|z\|$	[0.025	0.975]
Indicator for using the software	-0.5145	0.036	-14.252	0.000	-0.585	-0.444
Total downloaded packages	0.5632	0.013	43.789	0.000	0.538	0.588
Number of active days	0.0440	0.002	19.241	0.000	0.040	0.049
Log usage during trial	0.0620	0.005	11.267	0.000	0.051	0.073
Dormancy length	-0.0297	0.001	-30.141	0.000	-0.032	-0.028

Table 2.7: Regression of subscription on usage features and trial length, with all the pre-treatment variables included as controls (not shown in the table above).

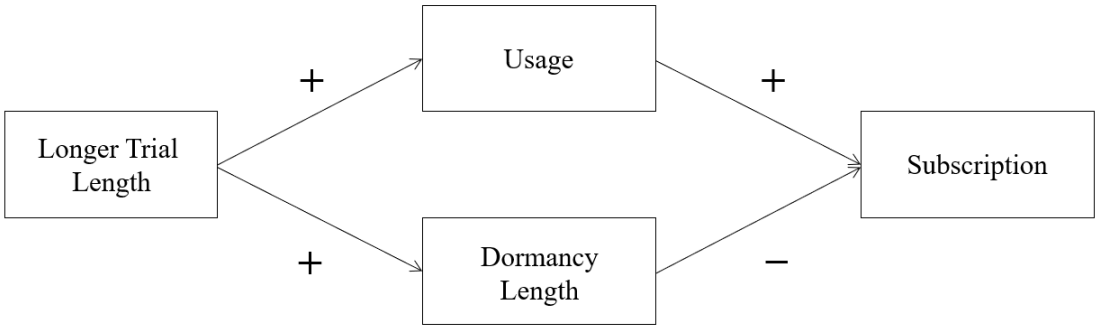


Figure 2.3: The effect of trial length on usage and dormancy length, and subsequently subscription.

five days are more likely to subscribe when they are in the 7-day condition, compared to the 14 or 30-days condition. Next, in the right panel of Figure 2.2, we show the probability of subscription as a function of the last active day for all three trial lengths. We see that users whose last active day is earlier in the trial period are less likely to subscribe. Further, for the same last active day, users with shorter trials are more likely to subscribe. Recall that dormancy length is defined as trial length minus the last active day. So this suggests that users who have not used the product for long periods at the end of the trial period are less likely to subscribe.

We now check if the preliminary patterns shown in Figure 2.2 hold after we control for other usage and user-specific observables. In Table 2.7, we present the results from a regression with the user's subscription decision as the outcome variable and her trial length and usage variables as explanatory variables. We find that after controlling for everything else, users who have more active days and use the product more are more likely to subscribe. Further, users who have longer dormancy periods are less likely to subscribe. This is understandable because a user who has not used the software for a long time by the end of her trial period is likely to forget about it and/or

conclude that the product is not useful [Zhu et al., 2018].

Together, the above findings suggest that two opposing effects of trial length on usage and subscription. We depict these effects in Figure 2.3. On the one hand, as trial length increases, product usage and consumer learning about the software increases. This increase in usage can have a positive effect on subscriptions. On the other hand, as trial length increases, the gap between the last active day and the end of the trial increases, while the average number of active days and usage per day reduces. These factors are associated with lower subscriptions. In our case, it seems that the latter effect dominates the former, and hence shorter trials are better.[8]

Our analysis presents three key findings relevant to the broader theories on the role of free trials for experience goods. First, we rule out the well-known demand cannibalization hypothesis advocated by many theoretical papers [Cheng and Liu, 2012, Dey et al., 2013]. These papers argue that, with longer trials, free-riders can use the product extensively during the trial, get their project/job done, and avoid subscribing. However, the results in Figure 2.2 and Table 2.7 rule out the free-riding hypothesis because users who use the product heavily during the trial are also more likely to subscribe. However, this evidence is for the full population of users. Second, we provide empirical support for the consumer learning hypothesis proposed in analytical papers (e.g., [Dey et al., 2013]) since we find that longer trials lead to more usage, which in turn is associated with higher subscription. Third, we identify a novel mechanism that plays a significant role in the effectiveness of free trials – the negative effect of long dormancy periods on subscription. We provide more evidence in support of these ideas in §2.6, where we discuss the heterogeneous response of different types of users.

2.4.3 Heterogeneity in Users' Responsiveness

So far, we have shown that the 7-days trial is the best average treatment and provided some intuition for why. However, the effect of trial length could be heterogeneous across users and the mechanisms discussed earlier could be differentially important for different types of users. We now examine if this is indeed the case.

In the top left panel of Figure 2.4, we partition the data into six sub-groups based on the user's geographic region and present the average subscription rates for the three trial lengths for each region. The results suggest that there is some heterogeneity in response rates by region. For example,

[8]The results in Table 2.7 should only be interpreted as suggestive evidence for the second half of the mechanism shown in Figure 2.3 (and not causally). This is because the unobserved attributes of the user that drive usage may also drive subscription.

the 14-day trial is more effective in Germany while the 7-day trial is more effective in the United States of America. Next, we perform a similar exercise on skill-level and job (see the top right and bottom panels in Figure 2.4). Again, we find that users' responsiveness to the treatment is a function of their skill level and job. For instance, the 7-day trial is significantly better for Beginners, whereas the 14-day trial is more effective for Mixed-skill users.

These results suggest that users' responsiveness to trial lengths is heterogeneous on many pre-treatment variables. If the firm can successfully exploit the different sources of heterogeneity and personalize its free trial assignment at the individual-level, then it may be able to further improve subscriptions.

2.5 Counterfactual Analysis: Personalized Policy Design and Evaluation

Given that the preliminary evidence above suggests that benefits the firm can benefit from personalizing free trial assignment. In §2.5.1, we describe the procedure we use to design the personalized policy. Next, in §2.5.2, we present the gains from the personalized policy in our setting. Next, in §2.5.3, we compare the performance of our approach to other personalized policies. Finally, in §2.6, we examine why some consumers respond better to shorter trials (vs. longer trials) and tie the policy-prescribed segmentation to mechanisms discussed in §2.4.2.

2.5.1 Optimal Policy Design

Let $i \in \{1, \ldots, N\}$ denote the set of independent and identically distributed users, where each user is characterized by a pre-treatment covariate vector $X_i \in X$ of dimension D. Let $W_i \in \mathcal{W}$ denote the treatment or intervention that i receives. $\mathcal{W} = \{0, \ldots, W-1\}$ refers to the set of treatments, and the total number of treatments is W. Finally, let $Y(X_i, W_i)$ denote the outcome for a user i with pre-treatment variables X_i when she is allocated treatment W_i.

A personalized treatment assignment policy, π, is defined as a mapping between users and treatments such that each user is allocated one treatment, $\pi : X \to \mathcal{W}$. The firm's goal is to choose a policy π such that it maximizes the expectation of outcomes, $\frac{1}{N}\,\mathbb{E}\left[\sum_{i=1}^{N} Y(X_i, W_i^{\pi})\right]$. Thus, for policy π and outcome of interest Y, we can write our reward function as $R(\pi, Y) = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\left[Y(X_i, \pi(X_i))\right]$. Thus, given a reward function $R(\pi, Y)$, the optimal personalized policy is given by:

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}}\left[R(\pi, Y)\right], \tag{2.1}$$

where Π is the set of all possible policies.

The problem of finding the optimal personalized policy is equivalent to one of finding the policy π^* that maximizes the reward function $R(\pi, Y)$. As discussed in §2.1, this is a non-trivial problem since the cardinality of the policy space can be quite large.[9] So, a direct search over the policy space to find the optimal policy is infeasible. Therefore, we adopt a two-step approach to find the optimal policy π^* that avoids this problem. To do so, we make the three standard assumptions on: (1) unconfoundedness, (2) SUTVA, and (3) positivity. Given that our data comes a fully randomized experiment assumptions (1) and (2) are automatically satisfied. Further, assumption 2 is satisfied because we do not expect any network effects in our setting (since the experiment was run on unconnected users distributed all over the world).

With these assumptions in place, we can design the optimal personalized policy if we either have estimates of the outcome of interest or pairwise treatment effects. In the main analysis, we design our personalized policy based using outcome estimates based on lasso [Tibshirani, 1996, Friedman et al., 2010]. That is, we model the subscription outcome using as $f(x, w) = \mathbb{E}[Y|X_i = x, W_i = w]$, where $f(\cdot)$ is a lasso model. Lasso estimates a linear regression that minimizes the MSE with an additional term to penalize model complexity as shown below:

$$(\hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3) = \sum_{i=1}^{n} (Y_i - X_i\beta_1 - W_i\beta_2 - X_iW_i\beta_3)^2 + \lambda(||\beta_1||_1 + ||\beta_2||_1 + ||\beta_3||_1), \tag{2.2}$$

where $||\beta_i||_1$ is the L1 norm of the vector β_i and is equal to the sum of the absolute value of the elements of vector β_i. Intuitively, if there are multiple weak (and correlated) predictors, lasso will pick a subset of them and force the coefficients of others to zero, thereby estimating a simpler model. Model-selection in lasso is data-driven, i.e., λ is a hyper-parameter that is learned from the data (and not assumed). Please see Appendix A.4 for details of our lasso estimation.

Next, using estimate of the expected outcome, $\hat{y}(x = X_i, w)$, from the lasso model, we obtain the optimal personalized policy based on for observation i as:

$$\pi_{lasso}(X_i) = w^*, \quad \text{where} \quad w^* = \underset{w \in \mathcal{W}}{\operatorname{argmax}}\, \hat{y}(x = X_i, w) \tag{2.3}$$

Our personalized free trial policy, π_{lasso} partitions the population into three segments: 7-, 14- and 30-days optimal segments, which constitute 68.9%, 23.2%, and 7.9% of the population, respectively.

---

[9] The total number of possible policies is $W^{\prod_{d=1}^{D} c_d}$, when we have D pre-treatment variables and the d-th variable can take c_d different values. This number can be extremely high even in simple settings. In our application, the cardinality of the policy space is equal to $3^{987,840}$.

2.5.2 *Empirical Policy Evaluation: Gains from Personalization*

We now empirically evaluate and quantify the gains from the personalized free trial policy π_{lasso} over non-personalized policies. To do so, we first define three uniform (one length for all) policies:

- π_{30} – This policy prescribes the 30-days treatment for all users. It was used by the firm at the time of the experiment and we therefore use it as the baseline policy in all our comparisons.
- π_{14} – This policy prescribes the 14-day treatment for all users.
- π_7 – This policy prescribes the 7-day treatment for all users. Since we found that 7 days is the best average treatment in §2.4.1, this is the best uniform policy.

We evaluate the expected reward from the policies (both personalized and uniform) using the Inverse Propensity Score (IPS) estimator, that has been extensively used in the off-policy evaluation literature [Horvitz and Thompson, 1952, Dudík et al., 2011], and has recently been applied in the marketing too [Simester et al., 2019, Rafieian and Yoganarasimhan, 2021, Hitsch and Misra, 2018]. For any given policy π, this estimator takes all the observations where the user received the policy-prescribed treatment and scales them up by their propensity of receiving the treatment assigned to them. This scaling gives us a pseudo-population that received the policy-prescribed treatment. Thus, the average of the outcome for this pseudo-population gives us an unbiased estimate of the reward for the full population, if we were to implement the proposed policy in the field. Formally:

$$\hat{R}_{IPS}(\pi, Y) = \frac{1}{N}\sum_{i=1}^{N} \frac{1[W_i = \pi(X_i)]Y_i}{\hat{e}_{\pi(X_i)}(W_i)}, \tag{2.4}$$

where $\hat{e}_{\pi(X_i)}(W_i)$ is the probability that a user whom the policy prescribes treatment $\pi(X_i)$ is given W_i.[10]

We present the expected rewards (or subscription rates) from all the three uniform policies and π_{lasso} in the top panel of Table 2.8. The key finding is that personalization based on pre-treatment demographic variables leads to over 6.8% improvement in subscription compared to the baseline uniform policy of giving a 30-days trial for all. Further, we see that the personalized policy also does better than the best uniform policy of 7 days for all. To examine if this difference is significant, we conduct a paired t-test based on bootstrapping as follows. We repeatedly (20 rounds) split the

[10] In theory, in a randomized experiment, the propensity of treatment assignment is orthogonal to the treatment prescribed by any policy π. Thus, $e(W_i = w, X_i) = e(W_i = w)\ \forall\ w\ \in \mathcal{W}$ is known and constant for all observations. However, in practice, within the set of users for whom policy π prescribes w, the empirical treatment propensities might not be the same as that in the full data. So we use the empirical propensity, defined as: $\hat{e}_{\pi(X_i)}(W_i) = \frac{\frac{1}{N}\sum_{j=1}^{N} 1[W_j=W_i, \pi(X_j)=\pi(X_i)]}{\frac{1}{N}\sum_{j=1}^{N} 1[\pi(X_j)=\pi(X_i)]}$.

Policy category	Policy	Estimated Subscription (%)		Increase in subscription (%)	
		Training Set	Test Set	Training Set	Test Set
Personalized based on lasso	π_{lasso}	15.85	15.62	7.97	6.81
Uniform	π_7	15.32	15.44	4.34	5.59
	π_{14}	14.90	15.11	1.45	3.28
	π_{30} (Baseline)	14.68	14.63	—	—
Alternative personalized policies	π_{reg}	15.89	15.33	8.21	4.83
	π_{cart}	15.32	15.44	4.34	5.59
	π_{r_forest}	17.42	14.82	18.67	1.32
	$\pi_{xgboost}$	16.00	15.53	8.98	6.17
	π_{c_forest}	15.58	15.46	6.09	5.71
	π_{c_tree}	15.32	15.44	4.34	5.59

Table 2.8: Gains in subscription from implementing different counterfactual free-trial policies. The results for policies π_{cart}, π_{c_tree}, and π_7 are the same since they prescribe the 7-days treatment to all users.

entire data into training and test (in the same proportion used in the main analysis, i.e., $0.7/0.3$). Then, in each round, we train a lasso model on the training set using a five-fold cross-validation and calculate the IPS-rewards (based on Equation 2.4) for both π_7 and π_{lasso} on the test data. Finally, we run a two-sided paired t-test to compare lasso's performance with the uniform all 7-days policy. The t-statistic and p-value for the two-sided test are 3.123 and 0.0056, respectively, which confirms that the personalized policy π_{lasso} is better than the best uniform policy π_7.

That said, notice the magnitude of gains from personalization (over the best uniform policy) is modest. This is understandable since the personalized policy assigns about 70% of users to the 7-day treatment, and the gains from personalization only accrue from the remaining 30% of users who are allocated the 14- or 30-days treatments. As Simester et al. [2020] point out, this is because the difference in the the performance of the two policies for users assigned to the 7-day trial is exactly zero. Further, our treatment effect is small compared to the outcome – a common occurence for marketing interventions such as advertising or promotions [Lewis and Rao, 2015]. These findings are consistent with the recent literature on personalization [Yoganarasimhan, 2020, Rafieian and Yoganarasimhan, 2021], which demonstrate positive but moderate gains from personalization digital interventions. They also provide partial support for the growing literature on perils of personalization [Zhang, 2011, Hajihashemi et al., 2021].

2.5.3 Comparisons and Robustness Checks

So far, we used lasso as the outcome estimator to design our personalized policy. We now examine whether counterfactual personalized policies based on other outcome and heterogeneous treatment effects estimators perform better. Specifically, we consider policies based on four additional outcome estimators: (1) linear regression, (2) CART, (3) random forests, and (4) XGBoost, and two heterogeneous treatment effect estimators: (1) causal tree, and (2) generalized random forests. The technical details of these models and their tuning details are shown in Appendices A.5 and A.6.

First, we find that each of these policies behaves quite differently when it comes to treatment assignment (see Table A.8 in Appendix A.7 for details). Interestingly, we find that policies based on CART and causal tree do not personalize treatment assignment and end up giving the 7-day treatment to all users, that is: $\pi_7 \equiv \pi_{cart} \equiv \pi_{c\_tree}$. We also find that in $\pi_{xgboost}$ is quite similar to π_{lasso}. Both prescribe the 7-days trial to $\approx 70\%$ of users, the 14-days trial to $\approx 20\%$ of users, and the 30-days trial to $\approx 10\%$ of users. In contrast, π_{reg} and π_{r_forest} prescribe the 7-day treatment to the least number of users while π_{c_forest} prescribes the 7-day treatment to 91% of users (and the 30-day treatment to no one).

Next, in the bottom panel of Table 2.8, we present the performance of these policies. We find that π_{lasso} continues to be the best, and the second-best policy is $\pi_{xgboost}$. There are two main takeaways here. First, poorly designed personalized policies (e.g., those based on regression and random forest) can actually do worse than the best uniform policy on the test data. Second, we do not find much correlation between an outcome estimator's predictive ability and its efficacy in policy design. For instance, random forest has a lower mean squared error on the test data compared to lasso, but π_{r_forest} is much worse than π_{lasso} (see Table A.9 in Appendix A.7). This is likely because the objective function in outcome estimation methods is predictive ability, which is different from policy design or performance. In sum, our findings suggest that managers should be careful in both designing and evaluating personalized policies. It is critical to: (1) not conflate a model's predictive ability with its ability to form policy, and (2) evaluate the performance of each policy on an independent test data with appropriate policy evaluation metrics.

Next, we find that the recently proposed heterogeneous treatment effects estimators, causal tree and causal forest, perform poorly when it comes to personalized policy design. Our results suggest that managers may be better off adopting the best uniform policy instead of investing resources in personalizing policies based on these methods. This is an important finding since these methods are gaining traction in the marketing literature and researchers are starting to use them (e.g., Guo et al. [2017], Fong et al. [2019]). Our findings suggest that relying on heterogeneous treatment effects

estimators can be sub-optimal.

Finally, we examine if there is any relationship between the estimates of treatment effects based on a specific method and the performance of the policy based on it. Figure 2.5 shows the CDF of $\tau_{7,30}$ for all the methods used for policy design.[11] [12] The first pattern that stands out is that treatment effects estimates based on CART, causal tree, and casual forest show very little heterogeneity (see the three vertical lines to the right of zero). This explains why policies based on these methods perform poorly – they are unable to personalize the policy sufficiently to optimize users' response at the individual level. In contrast, the treatment effect estimates based on linear regression and random forest show the maximum amount of heterogeneity (see the two rightmost curves). This pattern, in combination with the poor performance of π_{reg} and π_{r_forest} on test data (and their extremely good performance on training data) hints at overfitting problems. That is, these models seem to infer much more heterogeneity than is true in the data. Interestingly, we see that the CDFs of treatment effects based on lasso and XGBoost lie somewhere in between the above two groups. They show sufficient heterogeneity, but not too much. Hence, policies based on these methods are able to personalize the treatment sufficiently without overfitting. Recall that the dispersion in treatment assignment for these two policies is higher than that in π_{cart}, π_{c_tree}, and π_{c_forest}, but lower than that in π_{reg} and π_{r_forest}. Thus, the ideal estimators for policy design are those that are able to capture sufficient heterogeneity to personalize effectively without overfitting (i.e., capture spurious heterogeneity).

2.6 Segmentation Analysis and Additional Evidence for Mechanism

So far, we focused on the question of "Who (should get a treatment)". We now examine the question of "Why (should s/he get a specific treatment)". Understanding why some users respond well to longer trials while others respond better to shorter trials can give us insight into consumers' preferences and decision-making process. These insights are valuable for two reasons. First, from the firm's perspective, they can be leveraged to improve other marketing interventions such as advertising and pricing. Second, from a research perspective, this gives us a better understanding of the sources of heterogeneity in the effectiveness of trial length on conversion and mechanisms at

[11] The estimated distributions of $\tau_{14,30}$, and $\tau_{7,14}$ are shown in Figure A.2 in Appendix A.7 and their interpretations are largely similar to that presented here for $\tau_{7,30}$.

[12] For the outcome estimators, we can estimate treatment effects from outcome estimates as , $\mathbb{E}[Y|X_i = x, W_i =$ 7 days$] - \mathbb{E}[Y|X_i = x, W_i =$ 30 days$]$. For heterogeneous treatment methods, these estimates are directly available (see Equation (A.4)).

play, which can be generalized to other settings.

We now correlate a user's optimal treatment with her pre-treatment demographic and post-treatment behavioral variables. In the process, we present additional evidence for the mechanism through which trial length affects conversion, as discussed in §2.4.2. We conduct three sets of analyses to understand the mechanism and characterize the three segments. First, we quantify the differential effect of trial length on the download and usage behavior of the three segments. Second, we characterize the heterogeneity in the effect of usage on subscription across the three segments. Finally, we correlate a user's optimal treatment with her/his pre-treatment demographics and post-treatment outcomes to characterize the three segments. We refer readers to Appendix A.8 for the details of these analyses and provide a summary of the three segments below.

- 7-day optimal segment: A vast majority of these users are beginners or students, and they are the least likely to subscribe. These users use the product more when given longer trials but don't scale up their usage as much as the 14-day optimal segment. This is understandable because most of them lack the skills to use the product extensively, even if given the opportunity to do so. Further, the negative effect of long dormancy periods is the most severe for this segment. This is understandable since these users are less familiar with the product, so when they go for a long period without using the software, they are more likely to conclude that the software is not useful and choose not to subscribe.

  Overall, we find that short trials are more effective for beginners and new users because even though there are some positive effects of learning and usage, extended periods of inactivity at the end of long trials can have a strong negative effect on their subscription. One might wonder if this result simply stems from the fact that beginners have short tasks that require more than a week to complete (but still less time than 14/30 days), which leads them to have lower subscription rates when assigned the 14 and 30-day trial (i.e., a more complex version of the demand cannibalization hypothesis). However, if this explanation is true, then we should find that beginners/7-day optimal users who are assigned to the 14- and 30-day trial should be less likely to subscribe if they use the product more. However, we find the opposite; see Appendix A.8.4.
- 14-day optimal segment: These users are more likely to be mixed-skill, and they have the highest usage and subscription rates. This segment takes the most advantage of longer trials, i.e., they use the product the most and have the shortest periods of dormancy when given longer trials. It seems like these users actually try the product's features and evaluate the product carefully before deciding whether to subscribe or not. However, the effect of usage on subscriptions is

lower for these users (compared to the other two segments). This is likely because they are figuring out whether the software is a good fit or not, and more usage may lead some users to learn that it is not a good fit. Further, the magnitude of the negative effect of dormancy length on subscription is also high for them. That is why the 30-days trial is not optimal for these users: the higher usage that comes with a more extended trial does not translate to big differences in subscription, but they still get hit by the increase in dormancy length with the 30-days trial. On the other hand, when they are given only 7-days, they cannot use the product much, and the benefit from higher usage is not realized. Thus, 14 days is ideal for these users.

- 30-day optimal segment: These users are more experienced than average are less likely to be students and hobbyists, and more likely to sign up through the app manager instead of the website. These factors suggest that these are more likely to be experienced/legacy users who are already familiar with the software. Long dormancy periods have the least negative effect on these users. This is understandable because these users are likely to be already aware of and experienced with the software. Thus, they are unlikely to infer that the product is not useful if they don't use it for a few days at the end of the trial. Further, longer trials lead to more usage for these users, and the effect of usage on subscription is also high. Thus, giving them 30-days for trial is good.

One interesting pattern in the above findings is the non-monotonicity of usage across the three segments. We find that 7-day optimal users use the product the least, followed by the 30-day optimal users, while the 14-day optimal users use the product the most. This can be explained by the relative expertise-levels of the three groups. The extent to which a user uses the product depends on two factors: (1) how much do they need to evaluate the product?, and (2) how much can they evaluate product? The 7-day optimal users, who are pre-dominantly beginners have the least ability to explore the product features, and hence use it the least. In contrast, the 30-day optimal users, who are more likely to be experts and legacy users, have the highest ability to evaluate the product. However, given their expertise and familiarity with the software, they can do this without extensive usage. Finally, the 14-day optimal users, who are more likely to be mixed-skill users, have both high need to evaluate the product and sufficient ability to explore it. Hence, they have the highest usage.

It worth mentioning that our findings provide partial support to the theories proposed in the literature on the relationship between users' skill-level/experience and the effectiveness of free trials. For example, Dey et al. [2013] argue that longer trials are beneficial only when the learning rate is sufficiently large. We find that this is true in our case as well. However, this prior analytical research does not consider the negative effect of dormancy length on subscription, especially for beginners

and new users. They argue that beginners should be given longer free trials because longer trials allow them to learn about the product, which increases their likelihood of subscription. In contrast, we find that short trials are optimal for beginners. While longer trials have a positive impact on the usage and subscription of this group, they are also the group whose subscription is most negatively affected by longer dormancy periods. Thus, ignoring the negative impact of dormancy length can lead us to make sub-optimal allocations of trial lengths for different segments.

Our findings suggest that firms and managers should take into account the heterogeneity in the evolution of usage and inactivity (as trial length increases) for different consumer types and customize trial lengths based on these patterns. In our setting, users require some skill and need to invest the effort to learn and use the software effectively. In particular, beginners and inexperienced users are unable to scale up their usage with longer trials, and therefore have longer periods of inactivity later in the trial period (which has a detrimental effect on subscription). However, if the software is simple and easy to use, we would not see such periods of inactivity. Interestingly, this suggests that simpler products may benefit from longer trials (especially for beginners), whereas more complex products may benefit from shorter trials. In sum, both the complexity of the product and the skill of the user jointly determine usage and activity (or inactivity), which then affects subscription. Our results provide some general guidelines to firms on how to pick the right trial length for different products and segments.

2.7 Long-term Outcomes: Consumer Loyalty and Profitability

So far we have focused on short-run outcomes in our policy design and evaluation. However, a policy that maximizes subscriptions (or short-run conversions) may not be the best long-run policy if it brings in users who are less profitable or less loyal. For example, a policy that increases subscriptions among students (who get a significant educational discount and hence pay lower prices) and/or users who subscribe to lower-end products/bundles (that are priced much lower than the all-inclusive software suite) at the expense of high-end users can lead to lower revenues. Similarly, a policy designed to maximizes subscriptions may do so at the expense of long-term retention, i.e., it may bring in the less loyal consumers who churn within a short period. Thus, a subscription-optimal policy may in fact be sub-optimal from the perspective of long-run outcomes [Gupta et al., 2006, Fader and Hardie, 2009, McCarthy et al., 2017]. In this section, we therefore examine two important post-subscription outcomes of interest for the firm.

- Consumer loyalty, as measured by subscription length or the number of months a user subscribes to the service over the two year period after the experiment.

- Consumer profitability, as measured by the revenue generated by the user over the two years after the experiment. (In SaaS settings, revenues and profits can be treated as equivalent since the marginal cost of serving an additional user is close to zero.)

2.7.1 Gains in Retention and Revenue from Counterfactual Policies

We first show the average treatment effect of the three trial lengths on retention and revenue in Table 2.9.[13] We find that the 7-days trial continues to be the best. In the test data, it increases retention by 6.4% and revenue by 7.91%. The average effect of the 14-days trial is both smaller in magnitude and not significant in the training data.[14] These results largely mirror our findings on the average treatment effect of subscription, i.e., the 7-days trial is the best treatment.

Next, we examine how the uniform and personalized targeting policies described in §2.5.2 perform on the two long-term outcomes of interest. To derive the the IPS estimates of average subscription length and revenue under policy π, we first segment users into three groups based on the policy-prescribed treatment: (1) $\pi(X_i) = 7$ days, (2) $\pi(X_i) = 14$ days, and (3) $\pi(X_i) = 30$ days. Then, we use the observed subscription lengths and revenues as the outcome variables (Y_i) in Equation (2.4) to estimate the IPS rewards for these outcomes. Table 2.10 shows the results from this analysis. The main takeaway is that the personalized policy, π_{lasso}, which was designed to maximize subscriptions also does well on consumer loyalty and revenue compared to the other uniform policies. This is valuable from the the firms' perspective because it suggests that policies optimized for short-run outcomes are largely aligned with long-run outcomes as well.

---

[13] A minor point is that we do not have access to subscription length and revenue data for all subscribers (recall the discussion in §2.3.3). So we treat the missing observations as zero in calculations. The results remain qualitatively unchanged if we instead work with the subset of users for whom we have non-missing revenue data.

[14] Note that 14-days trial outperforms the 7-days trial in the test data (on revenue) even though the 7-days trial is the best policy in the training data. We present a brief explanation for this discrepancy now. In general, estimates from one data set are valid in another data set only when the joint distribution of outcomes and covariates are similar in both data sets. However, in finite samples, there are usually some minor differences in training and test data due to the randomness in the splitting procedure. In our case, the main difference is this – the distributions of subscription length for the 14-day condition in the training and test data are different. This is however not the case for the 7- or 30-day conditions; see Table A.16 in Appendix A.9. Thus, the estimate of subscription length from the training data does not translate well to test data, and this leads to the large difference in the subscription length and revenue estimates across the training and test data sets.

Data	Treatment	Average Subscription Length	Retention difference	t-statistics	Percentage gain over baseline
Training data	7 days	2.32	0.16	4.27	7.27
	14 days	2.22	0.06	1.54	2.59
	30 days	2.17	—	—	—
Test data	7 days	2.33	0.14	2.42	6.28
	14 days	2.32	0.13	2.27	5.91
	30 days	2.19	—	—	—

Data	Treatment	Average Revenue	Revenue difference	t-statistics	Percentage gain over baseline
Training data	7 days	82.65	6.17	3.75	8.06
	14 days	79.08	2.59	1.58	3.38
	30 days	76.49	—	—	—
Test data	7 days	83.72	6.14	2.42	7.92
	14 days	84.02	6.44	2.53	8.30
	30 days	77.58	—	—	—

Table 2.9: Average effect of the 7- and 14-day treatments on long-term variables (subscription length and revenue) compared to the control condition of 30-day free-trial.

2.7.2 Gains in Short-run vs. Long-run Outcomes

An interesting empirical pattern here is that the gains in subscription length and revenues are quantitatively different from the gain in subscription (compare the percentage increases in Tables 2.8 and 2.10). We now discuss the source of this difference.

We can write down the expected subscription length (denoted by Y_i^l) conditional on treatment W_i as:

$$\mathbb{E}(Y_i^l|W_i) = \Pr(Y_i^s|W_i) \cdot \mathbb{E}\left[T_{end} - T_{start}|W_i, Y_i^s = 1\right], \tag{2.5}$$

where $\Pr(Y_i^s|W_i)$ is the probability that user i will subscribe conditional on receiving treatment W_i and $\mathbb{E}\left[T_{end} - T_{start}|W_i, Y_i^s = 1\right]$ is i's expected length of subscription conditional on receiving treatment W_i and subscribing ($Y_i^s = 1$). The reason for the discrepancy in the gains on the two outcomes – subscription and subscription length – becomes apparent from Equation (2.5). If trial length affects not just subscription, but also how long a subscriber will remain loyal to the firm, then the gains in Y_i^l will be naturally different from the gains in subscription. To examine if this is true in our data, we present the summary statistics for $\mathbb{E}\left[T_{end} - T_{start}|W_i, Y_i^s = 1\right]$ for the three trial lengths in Table A.16 in Appendix A.9. We see that there are some small differences in this metric

Category	Policy	Subscription Length				Revenue			
		Estimate (Months)		Increase (%)		Estimate ($)		Increase (%)	
		Training	Test	Training	Test	Training	Test	Training	Test
Personalized	π_{lasso}	2.39	2.36	10.42	7.96	85.96	86.67	12.38	11.72
Uniform	π_7	2.32	2.33	7.27	6.28	82.65	83.72	8.06	7.92
	π_{14}	2.22	2.32	2.59	5.91	79.08	84.02	3.38	8.30
	π_{30} (Baseline)	2.17	2.19	—	—	76.49	77.58	—	—

Table 2.10: IPS estimates of the average subscription length and revenue under counterfactual policies (three uniform and one personalized).

across the three trial lengths, which account for the differences between the gains in subscription and gains in subscription length.

Similarly, we can write the expected revenue (denoted by Y_i^r) conditional on treatment W_i as:

$$\mathbb{E}(Y_i^r|W_i) = \Pr(Y_i^s|W_i) \cdot \mathbb{E}\left[T_{end} - T_{start}|W_i, Y_i^s = 1\right] \cdot \mathbb{E}\left[\text{Price}_i|W_i, X_i, Y_i^s = 1\right]. \tag{2.6}$$

This is similar to Equation (2.5), with the additional $\mathbb{E}\left[\text{Price}_i|W_i, X_i, Y_i^s = 1\right]$ term. It suggests that trial length can influence revenues through three channels – (1) subscriptions, (2) length of subscription, and (3) price of the product subscribed. The first two were already discussed in the paragraph above. We now examine whether the products that consumers subscribe to and the prices that they pay are also a function of trial length. That is, we examine whether $\mathbb{E}\left[\text{Price}_i|W_i, X_i, Y_i^s = 1\right]$ is indeed a function of W_i in our data. Note that the price that a subscriber pays is a function of both the product that s/he subscribes to (e.g., single product, all-inclusive bundle) as well her/his demographics (e.g., students pay lower prices for the same product). In Table A.17 in Appendix A.9, we present the distribution of products and subscription type by trial length for all the subscribers in our data. Again, we see that there are some minor differences in product and subscription types across trial lengths, which explain the differences in revenue gains.

2.7.3 Optimizing on Long-run Outcomes

So far we saw that a personalized policy designed to optimize short-run conversions also does well on long-run outcomes. However, this still begs the question of how it compares to policies directly optimized to maximize long-run outcomes. In practice, the problem with using retention/revenues until some period T (e.g., two years) is that we have to wait till T to identify the best policy and

Dataset	Policy optimized on	Subscription	Total Revenue	Subscription Length
Training data	Subscription	15.85	85.96	2.39
	Total Revenue	15.60	86.41	2.35
	Subscription Length	15.71	84.77	2.40
Test data	Subscription	15.62	86.67	2.36
	Total Revenue	15.45	84.28	2.33
	Subscription Length	15.53	84.86	2.35

Table 2.11: Expected mean of the three outcomes of interest under policies optimizing each outcome.

then implement it. This is both sub-optimal and impractical from a firm's perspective. In contrast, using a short-term outcome such as subscriptions to design policy and then projecting the policy gains on long-term objectives (e.g., revenues) is both practical and feasible. However, a policy optimized on short-run outcomes may still perform worse than one directly optimized on long-term outcomes. Therefore, we examine and compare the performance of the policy designed to maximize subscriptions with policies designed to maximize customer loyalty or profitability, and see which performs better in our context.

To that end, we now design two other personalized policies designed to maximize: (1) subscription length and (2) revenue. The policy design follows the same procedure described in §2.5.1, but with revenue (Y_i^r) and subscription length (Y_i^l) as our outcome variables. That is, we first estimate two separate lasso models with the above two variables as outcome variables, and then assign policy based on them.

Table 2.11 compares the performance of the three personalized policies on the three outcome variables of interest to the firm: subscriptions, subscription length, and revenue.[15] Interestingly, we find that the policy optimized on short-run conversions (subscriptions) also performs the best on retention and revenue. There are three reasons for this. First, as we saw in the previous section, conditional on subscription, the differences in retention length and products purchased are relatively minor. Hence, optimizing on subscription is largely consistent with optimizing on retention/revenue. Second, recall that the long-run outcomes are missing for about 7% of the subscribers. So the policies based on these outcome have less information for training, which compromises their generalizability and hampers their performance on the test data. Third, subscription is a binary outcome and as such has no variance in the positive realizations. In contrast, the variance in the

[15] See Appendix A.10 for a discussion of how the treatment allocation varies across the three policies.

long-run outcome variables (subscription length and revenue) can be quite high. This variance makes it harder to generalize models based on these outcomes, which in turn adversely affects the performance of policies based on them.

In summary, our findings suggest that if there are no significant differences in customer loyalty and profitability as a function of the promotional channel through which the user converted (trial length in this case), then optimizing low-variance short-run conversions will also lead to more generalizable policies that will also perform well on long-run outcomes.

2.8 Conclusions

Free trials are now a commonly used promotional strategy for SaaS products and other digital experience goods. In this paper, we examine the effect of trial length on consumers' subscription and retention decisions using data from a large-scale field experiment run by a leading SaaS firm, where the firm randomly assigned new users to 7, 14, or 30 days of free trial. We leverage two unique features of the data in our study: (a) the exogenous assignment of trial length and (2) the user's post-treatment product download and usage data during the trial period.

We find that the shortest trial length (7-days) is the best average treatment and maximizes both short- and long-run outcomes, customer acquisition, retention, and profitability. While this result is likely to be specific to our setting, we examine the behavioral underpinning of these findings and provide some evidence on the mechanisms at play. We rule out the demand cannibalization or free riding theory, find support for the consumer learning hypothesis, and identify a novel mechanism that plays a significant role in the effectiveness of free trials – the negative effect of long stretches of inactivity at the end of the trial on subscription.

We then develop a personalized targeting policy based on lasso and show that it can lead to over 6.8% improvement in subscription compared to the baseline uniform policy of giving a 30-day trial for all. Further exploration of usage within different consumer segments in our personalization scheme suggests that simpler products and experienced users are more likely to benefit from longer trials. Finally, we find that the personalized policy designed to optimize subscriptions also performs well on long-term metrics such as customer retention and revenues in our setting. We also compare the performance of our benchmark personalized policy with alternative personalized policies developed based on other well-known outcome estimators (e.g., random forests) and the recently developed heterogeneous treatment effects estimators (e.g., generalized random forests). We find that many alternative personalized policies perform poorly, and are often worse than the simple uniform 7-days for all policy. A key managerial takeaway is that firms should not naively

assume that personalization based on the most advanced estimators always helps. Instead, they should develop personalized policies based on a number of methods and carefully evaluate them using offline IPS estimators before investing resources in deploying personalized policies in the field.

Our paper opens many avenues for future research. First, while our analysis indicates that product usage during the trial period affects users' subscription decisions, we do not causally tie usage to subscriptions because usage is endogenous. Nevertheless, future research may be able to use treatment assignment (e.g., trial length) as an instrument that exogenously shifts usage and directly estimate the effect of usage on purchase. This can provide further insight into the question of whether encouraging usage (either through free trial or product improvements) can lead to better purchase outcomes. Second, our analysis suggests that personalized policies do not always perform better than a simple uniform policy. One interesting finding is that outcome estimators that have high predictive ability do not necessarily do well on personalized policy design (compare the performance of models in Table A.9 in Web Appendix A.7 with Table 2.8). Moreover, the finding that recently developed CATE estimators such as causal forest do not perform well in our setting is also surprising. Further investigation into the question of whether these results are generalizable would be a useful next step.

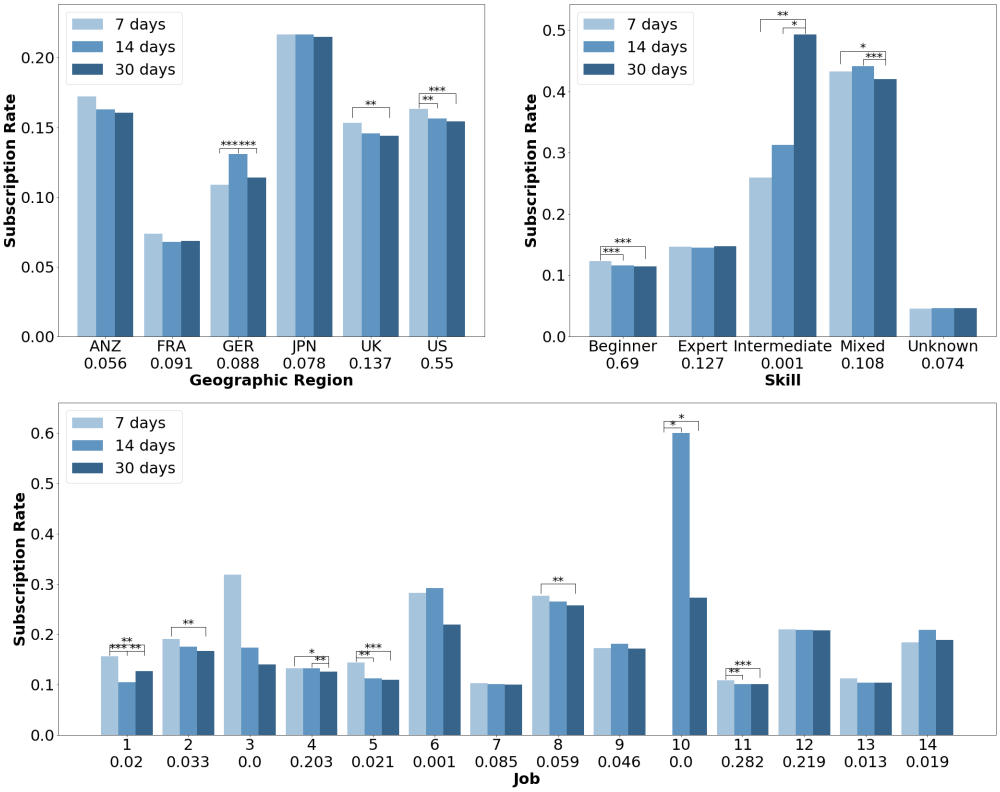


Figure 2.4: Heterogeneity in consumers' response to the three trial lengths within three categories – Geographic region, Skill, and Job. The six geographic regions shown are: Australia and New Zealand, France, Germany, Japan, and United States of America (in that order). Under each sub-category, the fraction of users in that sub-category are shown. We do not include sub-category names for Job to preserve the firm's anonymity.(* $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$)

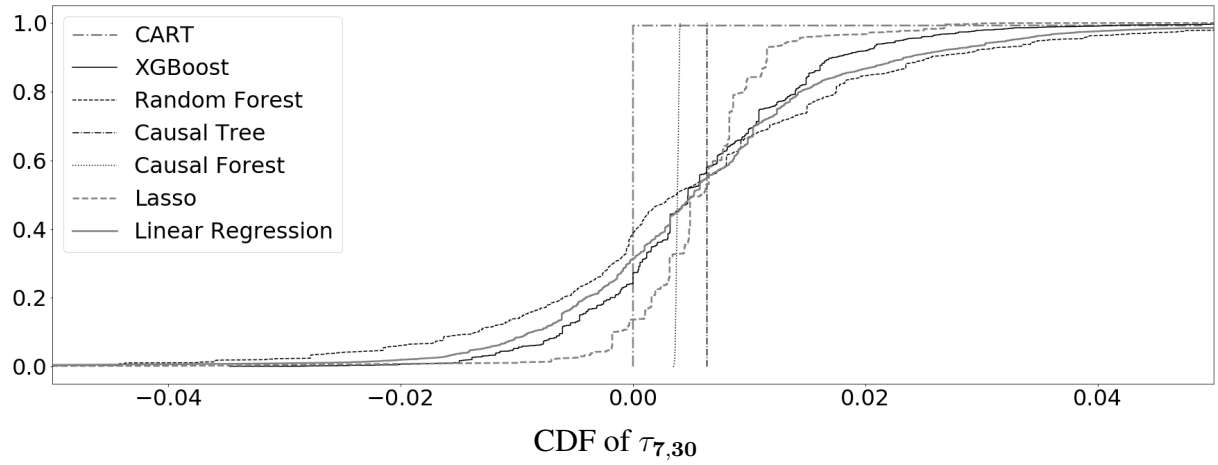


Figure 2.5: The CDF of estimated CATEs for 7 vs 30 days of free trial from using different methods (for test data).

Chapter 3

A RECURSIVE PARTITIONING APPROACH FOR DYNAMIC DISCRETE CHOICE MODELING IN HIGH DIMENSIONAL SETTINGS

Dynamic discrete choice models are widely employed to answer substantive and policy questions in settings where individuals' current choices have future implications. However, estimation of these models is often computationally intensive and/or infeasible in high-dimensional settings. Indeed, even specifying the structure for how the utilities/state transitions enter the agent's decision is challenging in high dimensional settings when we have no guiding theory. In this paper, we present a semi-parametric formulation of dynamic discrete choice models that incorporates a high-dimensional set of state variables, in addition to the standard variables used in a parametric utility function. The high-dimensional variable can include all the variables that are not the main variables of interest, but may potentially affect people choices and must be included in estimation procedure, i.e., control variables. We present a data-driven recursive partitioning algorithm that reduces the dimensionality of the high-dimensional state space by taking the variation in choices and state transition into account. Researchers can then use the method of their choice to estimate the problem using the discretized state space from the first stage. Our approach can reduce the estimation bias and make estimation feasible at the same time. We present Monte Carlo simulations to demonstrate the performance of our method compared to standard estimation methods where we ignore the high-dimensional explanatory variable set.

3.1 Introduction

Consumers' choices are not solely a function of the utility they get in the current stage in many settings. They might forgo a choice with higher current utility for a better stream of utilities in the future. Buying a new car instead of keeping an old car, getting higher education, and investing in retirement plans are examples of such choices. Estimating the underlying primitives of an agent's behavior in these settings requires incorporating the current utility she gets and her future stream of utilities from a choice.

In the conventional dynamic discrete choice modeling approaches, researchers calculate the value of being in each state of the problem space. This value, a.k.a value function, is equal to the

discounted sum of all the future utilities an agent gets from making optimal choices in the future starting from that given state. Researchers calculate the value functions by solving a dynamic programming problem using the Bellman equation [Smammut and Webb, 2010]. Two limitations make the estimation of these models challenging. First, solving the dynamic programming problem is computationally expensive and infeasible in high-dimensional data settings. Second, researchers must make some assumptions on the data generating process in the unobserved part of the state space to make the estimation possible. For example, one common assumption is that the observable part of the state in time t is independent of the unobservable part of the state in the previous time period. As a result, it becomes essential to incorporate all the potential variables that affect agents' decisions and state transitions to avoid violation of these assumptions, even if these variables are not the main focus of the research problem at hand.

These two limitations force an accuracy-computation trade-off to researchers. On the one hand, limiting the number of variables in the estimation procedure, in the favor computational feasibility, increases the chance of violating the required estimation assumption. On the other hand, adding more explanatory variables makes the computation infeasible as adding each variable increases the size of the state space exponentially [Rust, 1997]. In addition to the accuracy concerns, model specification choices such as selecting the appropriate covariates and discretizing the state space are challenging, especially in complex and high-dimensional settings. Researchers usually have little intuition about which covariates to select or how to discretize the state space [Semenova, 2018]. As a result, many estimation approaches avoid this trade-off by proposing value function estimation procedures that do not require solving a dynamic programming problem [Hotz and Miller, 1993, Hotz et al., 1994, Keane and Wolpin, 1994, Norets, 2012, Arcidiacono et al., 2013, Semenova, 2018].

Besides the concerns for estimation assumptions, the data-gathering trends in the industry calls for more high-dimensional friendly approaches in dynamic choice modeling. Gathering data has become a necessary part of businesses of all sizes. Companies create massive databases of users' behavioral and contextual data, hoping to turn the data into knowledge and enhance the quality of their services, and marketing interventions. Demographic and behavioral data is used for designing personalized services, promotions, and prices. The advent of high-dimension friendly approaches has made it possible for companies to exploit all the available information to extract knowledge from consumers. The hyper-parameter optimization procedure implemented within these algorithms ensures that the model is generalizable to the data-generating process, not the training dataset. Overfitting is of foremost importance in high-dimensional settings as it is easier for the algorithm to

model noise instead of signal [Zimek et al., 2012]. These methods, nevertheless, are not appropriate for modeling dynamic choice modeling settings, as they do not incorporate the future implications of a decision into account. Estimation of dynamic choice models in high-dimensional settings has remained a rewarding research topic open to investigation.

This paper proposes a novel approach that let researchers control for a high-dimensional variable set $\mathbf{Q}$, in addition to the conventional independent variable set $\mathbf{X}$ in dynamic discrete choice modeling. We reduce the dimensionality of $\mathbf{Q}$ using a data-driven discretization approach based on recursive partitioning. In our framework, we distinguish between the state space discretization and estimation, and separate these two steps in our framework. We define the term *perfect discretization* as a discretization where all the points in the same partition have a similar decision and transition probabilities. We reformulate the DDC problem using the perfect discretization definition. We then propose an algorithm for discretization and prove that the discretization offered by our algorithm converges to a perfect discretization. Researchers can then use any conventional algorithm for the estimation of parameters in for the estimation stage using the discretized state space offered in the first stage.

Our dimension reduction method has several desirable properties that makes it convenient to use and applicable in many settings. First, we separate the estimation and discretization tasks and define a general discretization criterion that does not depend on parametric assumptions involved in the estimation step. This property makes the algorithm robust to the parametric assumption of the estimation step. Furthermore, the discretization algorithm does not impose any limitation on the estimation method one can use in the second stage. The discretization algorithm converts the high-dimensional variable set $\mathbf{Q}$ to a categorical variable $\mathcal{P}$. Researchers can then use the new independent variable set $\{\mathbf{X}, \mathcal{P}\}$ with any conventional DDC estimation method that can handle categorical variables. In addition, the algorithm can be used for discretization in both finite and infinite time horizon settings.

Second, our discretization algorithm inherits the desirable properties of recursive partitioning-based algorithms. The time complexity of our method is linear with respect to the dimensionality of $\mathbf{Q}$. If the number of dimensions in $\mathbf{Q}$ doubles, the discretization time of our algorithm doubles at most. It is a substantial computational saving compared to conventional methods such as nested fixed-point, the time complexity of which is exponential with respect to the dimensionality of the independent variable set. In addition, our algorithm is robust to scale and irrelevant variables. Our discretization algorithm offers the same discretization for $\mathbf{Q}$ and any transformation of $\mathbf{Q}$ that does not change the ordinality of observations. Furthermore, the algorithm can be implemented in a

parallelized way, making its execution pretty fast over distributed systems and servers with many cores. These desirable properties make it possible for researchers and industry users to benefit from all their available data in the discretization procedures without domain expertise or computational concerns.

Third, in addition to the agents' choice data, our algorithm uses the rich information available in the state transition data to reduce the state space dimension. It is a novel approach given that researchers usually regard the state space transition as a nuisance parameter; they non-parametrically estimate it in the first step of DDC modeling. The algorithm's capability to use the variation in agents' decision and state transition for dimension reduction makes it a more efficient algorithm than when only the decision part is used for discretization. Additionally, our algorithm learns the relative importance of these two parts of agents' behavior during hyperparameter optimization and assigns an optimal weight to each part of the data. As we show in our simulation study section, optimizing the relative importance of these two can lead to considerable gains in optimal discretization.

Our paper is organized as the following. In section 3.2, we review the current literature on the estimation of dynamic discrete choice models and proposed methods for their estimation in high-dimension. We also touch upon the recursive partitioning algorithm and its extensions designed for estimation in different modeling settings. In section 3.3, we formulate the problem at hand by explaining the components of dynamic discrete choice models, pointing to the curse of dimensionality, defining perfect discretization, and reformulating the estimation of DDC problems in the discretized space. In section 3.4 we describe our discretization approach, discuss hyper-parameter optimization and model selection, and highlight the properties of the algorithm. We run two simulation studies to take a deeper look into this problem in section 3.5. In the first simulation study, we highlight the importance of controlling for estimation assumptions. In the second simulation study, we highlight the value of state transition data and our algorithm's ability to exploit the rich information in state transition for discretization. Section 3.7 concludes.

3.2 Related Literature

First, our paper is related to the literature of estimating dynamic discrete choice modeling in high-dimensional settings and breaking the curse of dimensionality in DDC modeling. The Bellman's equation in DDC modeling rarely has an analytical solution, and the dynamic programming problem of calculating the value function is usually solved numerically [Rust, 1997]. Unfortunately, the computational complexity of this numerical estimation increases exponentially as the number of explanatory variables increase. Many solutions have been proposed to break this course of

dimensionality, and make estimation of DDC models in high-dimensional settings feasible.

Hotz and Miller [1993] shows that the value function can sometimes be estimated from the probability that a specific choice occurs in any given state space. This so-called Conditional Choice Probability (CCP) method makes the DDC estimation problem possible in high dimensions. However, CCP techniques have two limitations: i) using CCP methods is not always possible since they rely on some particular structures in the state space transition, ii) to conduct counterfactual analysis, researchers usually need to solve the full model once using NFXP. Several methods used simulation to estimate the approximation of full solution methods [Keane and Wolpin, 1994, Hotz et al., 1994]. One can use the non-parametric estimations of choice probabilities and state transitions to draw a choice and a realized state given a choice and use simulations to solve the DDC problem.

Another stream of research tries to resolve the dimensionality problem by approximating the value function. Methods such as parametric policy iteration [Benitez-Silva et al., 2000], and sieve value function iteration [Arcidiacono et al., 2012] estimate the value function by assuming a flexible functional form for it. Nonetheless, these methods work well when there are a set of basis functions that provide a good approximation of the value function [Rust, 2000]. [1] A more recent body of literature has tried to use the advances in machine learning and methods such as neural networks to solve DDC problems. Norets [2012] uses an artificial neural network to estimate the value function taking state variables and parameters of interest as input. Semenova [2018] offers a simulation-based method using machine learning. She estimates the state transition and decision probabilities by machine learning models in the first stage and uses them to find the underlying decision parameters in the second stage.

Su and Judd [2012] proposes yet another approach to solve the curse of dimensionality problem in dynamic discrete choice modeling called MPEC. They formulate the dynamic discrete choice problem as a constrained maximization problem where likelihood function is the maximization objective, and the bellman equations are the constraints. Dubé et al. [2012] show that MPEC is applicable to a broader set of problems. MPEC method reduces the computational complexity as it does not need to solve the bellman equation and calculate the value function at each guess of the structural parameters. Nevertheless, MPEC algorithm is still not applicable in high-dimensional settings without proper discretization of the state space. Although it do not solve the structural equation at each iteration of MPEC algorithm, it estimates the value function for each point of the state space. As the number of variables increases, the size of the state space increases exponentially, and as a result, using MPEC becomes computationally infeasible.

---

[1] See Powell [2007] for a summary of the related literature on approximating in dynamic programming.

Our algorithm adds to this literature by offering a method to break the curse of dimensionality through discretization rather than value function approximation. We argue that state-space discretization and parameter estimation are two separate tasks. In fact, our discretization algorithm can be used together with any of the above algorithms: researcher can use our algorithm to reduce the dimensionality of a high-dimensional state space $\mathbf{Q}$ to a one-dimensional categorical variable $\mathcal{P}$ in the first stage, and then use $\mathcal{P}$ in addition to other independent variables $\mathbf{X}$ for estimation of parameters using any of the above algorithms in the second stage. This procedure lets the researcher control for a high-dimensional covariate $\mathbf{Q}$ at a low computational cost.

Second, our paper contributes to the literature of estimation using recursive partitioning. Breiman et al. [1984a] work gave birth to the Classification and Regression Trees (CART) algorithm, one of the earliest and well-known algorithms for estimation using recursive partitioning. Ensemble methods combine several trees to produce better performance than a single tree. The Random Forest algorithm [Breiman, 2001] is based on the idea of generating thousand of such trees, each on a subsample of data and covariates, and average the estimates of trees for prediction. While recursive partitioning has been used for prediction tasks, there has been a recent development for using recursive partitioning for different purposes. Athey and Imbens [2016] has used the recursive partitioning approach for the heterogeneous causal effect estimation task. They offer a method to partition the data into subgroups that differ in the magnitude of their treatment effects. Athey et al. [2019b] propose the Generalized Random Forests (GRF) algorithm, a method for non-parametric statistical estimation that can be used to fit any quantity of interest. They use their approach to develop new estimation methods for different statistical tasks, including non-parametric quantile regression, conditional average partial effect estimation, and heterogeneous treatment effect estimation via instrumental variables. However, their method is limited to the static estimation settings. Our algorithm adds to this literature by proposing a novel use of recursive partitioning to estimate dynamic models. We develop a new approach for formulating the objective function that enables us to use the state transition data during recursive partitioning.

Third, our paper indirectly contributes to the literature of unobserved heterogeneity in dynamic discrete choice modeling. A potential solution for concerns regarding DDC modeling assumptions on the unobservable part of state space is to use latent class models. Arcidiacono and Jones [2003] and Arcidiacono and Miller [2011] have suggested EM-based algorithms to account for unobservable parts of the state space and their transitions. Nevertheless, these algorithms suffer from several limitations that make them impractical in circumstances where we have quite a few unobservable states. Besides, these algorithms are not guaranteed to converge to the global maximum [Wu, 1983].

Even though our algorithm does not directly capture the effect of unobservables, its ability to capture the effects of a high-dimensional variable set alleviates the concerns from the unobservable part of the state space. Similar to the latent class models in DDC estimation, our algorithms assign a category to each observation. However, in contrast to the EM-based algorithms, which use a latent class to capture the explained variation in the dependent variable, our algorithm uses a high-dimensional variable set to potentially explain the variation.

3.3 Problem Definition

We consider the discrete choice problem from the perspective of a forward-looking single agent, denoted by $i \in \{1 \dots N\}$. In every period t, the agent chooses between $j = 1 \dots J.$ options. i's decision in period t is denoted by d_{it}, and $d_{it} = j$ indicates that agent i has chosen option j in period t. The agent's decision is not only the function of her utility in current state (s_{it}), but also her expectation of her utility in all her future states given decision d_{it}. We assume that the agent's state is composed of three sets of variables.

1. A set of observable low-dimensional state variables $x_{it} \in \mathbf{X}$.
2. A set of observable high-dimensional state variables $q_{it} \in \mathbf{Q}$.
3. Unobservable state variable ϵ_{it}, which is a $J \times 1$ vector each associated with one of the alternatives observed by the agent, but not by the researcher.

$\mathbf{X}$ represents state variables for which we have some a priori theory, i.e., we know how the parametric form in which they enter the utility function. The structural parameters associated with these variables form the main estimands of interest. $\mathbf{Q}$ denotes state variables that act as nuisance variables in our estimation exercise – they are not the main variables of interest, and we do not have a theory for if and how they influence users' decisions and state transitions. However, ignoring them can potentially bias the estimates of interest.

In each period t, agent i derives an instantaneous flow utility $u(s_{it}, d_{it})$, which is a function of her decision d_{it} and her state variables $s_{it} = \{x_{it}, q_{it}, \epsilon_{it}\}$. The per period utility is additively separable as follows:

$$u(s_{it}, d_{it} = j) = \bar{u}(x_{it}, q_{it}, d_{it} = j; \theta_1) + \epsilon_{itj}, \tag{3.1}$$

where ϵ_{itj} is the error term associated with j^{th} option at time t, $\bar{u}(x_{it}, q_{it}, d_{it} = j; \theta_1)$ is the deterministic part of the utility from making decision j in state $\{x_{it}, q_{it}\}$, and θ_1 is the set of structural parameters associated with the deterministic part of utility. The state s_{it} transitions

into new, but not necessary different, values in each period following decision d_{it}. We make three standard assumptions on the state transition process – first order markovian, conditional independence, and IID error terms. These assumptions imply that: (i) $\{s_{it}, d_{it}\}$ are sufficient statistics for s_{it+1}, (ii) error terms are independent over time, and (iii) errors in the current period affect states tomorrow only through today's decisions. Thus, we have:

$$\Pr(x_{it+1}, q_{it+1}, \epsilon_{it+1}|x_{it}, q_{it}, \epsilon_{it}, d_{it}) = \Pr(\epsilon_{it+1}) \Pr(x_{it+1}, q_{it+1}|x_{it}, q_{it}, d_{it}) \tag{3.2}$$

We denote the state transition function $\Pr(x_{it+1}, q_{it+1}|x_{it}, q_{it}, d_{it})$ as $g(x_{it+1}, q_{it+1}|x_{it}, q_{it}, d_{it}; \theta_2)$, where θ_2 captures the parameters associated with state transition.[2]

Each period, the agent takes into account the current period payoff as well as how her decision today will affect the future, with the per-period discount factor given by β. She then chooses d_{it} to sequentially maximize the expected discounted sum of payoffs $\mathbb{E}\left[\sum_{\tau=t}^{\infty} \beta^{\tau} u(s_{i\tau}, d_{i\tau})\right]$. Our goal is to estimate the set of structural parameters $\theta = \{\theta_1, \theta_2\}$ that rationalizes the observed decisions and the states in the data, which are denoted by $\{(x_{i1}, q_{i1}, d_{i1}), \ldots, (x_{it}, q_{it}, d_{it}), \ldots, (x_{iT}, q_{iT}, d_{iT})\}$ for agents $i \in \{1, \ldots, N\}$ for T time periods.

3.3.1 Challenges

The standard solution is to use a maximum likelihood method and estimates the set of parameters that maximizes the likelihood of observing the data. Given the first-order Markovian and conditional independence assumptions, we can write the likelihood function and estimate the structural parameters as follow

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \log \hat{p}(d_{it}|x_{it}, q_{it}; \theta_1) + \sum_{t=2}^{T} \log \hat{g}(x_{it}, q_{it}|x_{it-1}, q_{it-1}, d_{it-1}; \theta_2) \right) \tag{3.3}$$

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, \mathcal{L}(\theta)$$

where $\hat{p}(.)$ is the predicted choice probabilities.

However, there are three main challenges in estimating a model in this setting:

- **Theory:** First, the researcher may lack theory on how the high-dimensional variables q enter the agents' utility function. For instance, if we consider the example of high-dimensional usage

[2]Both the utility function and state transition can also be estimated non-parametrically if we do not wish to parametrize them. In that case, θ_1 and θ_2 would simply denote the utility and state transition at a given combination of state variables.

variables in a subscription model, we do not have much theoretical guidance on which of these variables affect users' utility and how. As such, we cannot make parametric assumptions on the effect of $\mathbf{Q}$ on decisions and transitions or hand-pick a subset of these variables to include in our model.

- **Data:** Second, in a high-dimensional setting, we may not have sufficient data in all areas of the state space to model the flow utility and the transitions to/from that area.
- **Estimation:** Finally, estimation of a discrete choice dynamic model in an extremely high dimensional setting is often computationally infeasible and/or costly. Rust [1987]'s nested-fixed point algorithm requires us to calculate the value function at each combination of the state variables at each iteration of the estimation, which is infeasible in a large state space setting. While two-step methods can overcome estimation challenges in large state-spaces by avoiding the value function iteration, they nevertheless need non-parametric estimates of Conditional Choice Probablities (CCPs) at all values at each state [Hotz and Miller, 1993]. This is not possible in a very high dimensional setting with finite data.

Thus, in a finite data, high-dimensional setting where we lack guiding theory, it is not feasible to specify a utility function over q and/or estimate a dynamic discrete choice model using conventional methods. Therefore, we need a data-driven approach that reduces dimensionality of $\mathbf{Q}$ in an intelligent fashion.

3.3.2 Dimensionality Reduction using Discretization

Our solution is to recast the problem by mapping $\mathbf{Q}$ to a lower-dimensional space through data-driven discretization such that $\mathbf{\Pi} : \mathbf{Q} \to \mathcal{P}$, where $\mathcal{P} = \{1, \ldots, k\}$. The goal is similar in spirit to that of Classification and Regression Trees (CART) algorithm used for outcome prediction [Breiman et al., 1984a] and the Causal Tree algorithm for estimation of conditional average treatment effects [Athey and Imbens, 2016]. These algorithms discretize the covariate space to minimize the heterogeneity of the statistic of interest within a partition. For example, the CART algorithm discretizes the covariate space into disjoint partitions such that observations in the same partition have similar outcome values/class. Similarly, the Causal Tree algorithm discretizes the state space such that observations within the same partition have similar treatment effects. However, the high-level intuition from the static estimation of CART/causal tree cannot be directly translated to dynamic discrete choice models. Therefore, our first step is to outline the characteristics of a suitable discretization. In §3.3.2 we formally define the term **perfect discretization** as a discretization wherein observations with similar behavior are pooled together in the same partition. Then in, §3.3.2, we present the

reformulated problem in the lower-dimensional space.

Properties of a Good Discretization

We now discuss some basic ideas that a good discretization should capture. First, some variables in $\mathbf{Q}$ may be irrelevant to our estimation procedure, i.e., they have no effect on utilities or state transitions. A good data-driven discretization should be able to neglect these variables and thus be robust to irrelevant variables. Second, our discretization approach has to be entirely non-parametric since we not have any theory on how variables in $\mathbf{Q}$ affect agents' utilities and state transitions. Finally, the discretization should be generalizable, i.e., it should be valid outside the training data. Formally, we define the term *perfect discretization* as follows:

Definition 1. A discretization $\mathbf{\Pi}^* : \mathbf{Q} \to \mathcal{P}$ is perfect if all the points in the same partition have the same decision and incoming and outgoing transition probabilities. That is, for any two points q, q' in a partition $\pi \in \mathcal{P}$, we have:

$$\forall x, x' \in \mathbf{X}, q'' \in \mathbf{Q}, j \in \mathbf{J} : \begin{cases} \Pr(j|x,q) = \Pr(j|x,q') & \text{(3.4a)} \\ \Pr(x',q''|x,q,j) = \Pr(x',q''|x,q',j) & \text{(3.4b)} \\ \Pr(x,q|x',q'',j) = \Pr(x,q'|x',q'',j) & \text{(3.4c)} \end{cases}$$

The first equality ensures that the decision probabilities are similar for data points within a given partition $\pi \in \mathcal{P}$. The second equality asserts the equality of transition probabilities *from* any two points q, q' within the same partition $\pi \in \mathcal{P}$. Finally, the last equality implies that the transition probabilities *to* any two points within a partition π are equal. Together, these three equalities imply that all the observations within a same $\pi \in \mathcal{P}$ are similar from both modeling and estimation perspective. Therefore, we do not need to model the heterogeneity within the partition π. Instead, modeling agents' behavior at the level of $\mathcal{P}$ is sufficient.

Formulation of DDC in a discretized space

We now use the definition of perfect discretization and translate the DDC estimation in the $\mathbf{Q}$-space to the $\mathcal{P}$-space. Because observations within each partition in a perfect discretization behave similarly, we can project the problem from the $\mathbf{Q}$-space to the $\mathcal{P}$-space. That is, $\mathbf{\Pi}^*$ is a sufficient statistic for estimation of state transition and decision probabilities. We can therefore write the

probabilities of choices and state transitions in the $\mathcal{P}$-space as follows:

$$\forall x, x' \in \mathbf{X}, q'' \in \mathbf{Q}, j \in \mathbf{J} : \begin{cases} \Pr(j|x,q) = \Pr(j|x, \mathbf{\Pi}^*(q)) & \text{(3.5a)} \\ \Pr(x', q''|x, q, j) = \Pr(x', q''|x, \mathbf{\Pi}^*(q), j) & \text{(3.5b)} \\ \Pr(x, q|x', q'', j) = \dfrac{\Pr(x, \mathbf{\Pi}^*(q)|x', q'', j)}{N(x, \mathbf{\Pi}^*(q))} & \text{(3.5c)} \end{cases}$$

where $N(x, \mathbf{\Pi}^*(q))$ is the number of observations in state $\{x, \mathbf{\Pi}^*(q)\}$ in our data.

These three equalities are the counterparts of the relationships shown in Equation (3.4) in the $\mathcal{P}$-space. According to the first equality in Equation (3.5), if the choice probabilities for the observations within a partition $\pi \in \mathcal{P}$ are the same, then $\{\mathbf{X}, \mathcal{P}\}$ is a sufficient statistic to capture the choice probabilities. The same argument is true for out-going state transitions. If the probabilities of transition to other states from all points in a partition are similar, we can use the partition instead of points to specify transition (as shown in the second relationship in Equation (3.5)). Finally, when the probability of transition into all the points within a partition are similar, the probability of moving to a specific point is equal to the probability of transitioning into the partition divided by the number of observations within that partition. That is:

$$\begin{aligned} \Pr(x, \mathbf{\Pi}^*(q)|x', q', j) &= \sum_{q'' \in \mathbf{\Pi}^*(q)} \Pr(x, q''|x', q', j) \\ &= N(x, \mathbf{\Pi}^*(q)) \Pr(x, q|x', q', j), \end{aligned}$$

which gives us the third relationship in Equation (3.5).

We now use the relationships in Equation (3.5) to reformulate the log likelihood in Equation (3.3). Given a perfect partitioning $\mathbf{\Pi}^*$, the log likelihood can be written as:

$$\begin{aligned} \mathcal{L}(\theta, \mathbf{\Pi}^*) = &\sum_{i=1}^{N} \sum_{t=1}^{T} \log p(d_{it}|x_{it}, \mathbf{\Pi}^*(q_{it}); \theta_1, \mathbf{\Pi}^*) \\ &+ \sum_{i=1}^{N} \sum_{t=2}^{T} \log \frac{g(x_{it}, \mathbf{\Pi}^*(q_{it})|x_{it-1}, \mathbf{\Pi}^*(q_{it-1}), d_{it-1}; \theta_2, \mathbf{\Pi}^*)}{N(x_{it}, \mathbf{\Pi}^*(q_{it}); \mathbf{\Pi}^*)} \end{aligned} \tag{3.6}$$

Note that the second term in the log-likelihood is obtained by combining the second and third terms in Equation (3.5) as: $\Pr(x', q''|x, q, j) = \Pr(x', q''|x, \mathbf{\Pi}^*(q), j) = \frac{\Pr(x', \mathbf{\Pi}^*(q'')|x', \mathbf{\Pi}^*(q), j)}{N(x, \mathbf{\Pi}^*(q''))}$.

Finally, since agents within a given partition in $\mathbf{\Pi}^*$ have similar state and choice probabilities, we can conclude that their utility function are also similar. We can formulate the utility function in

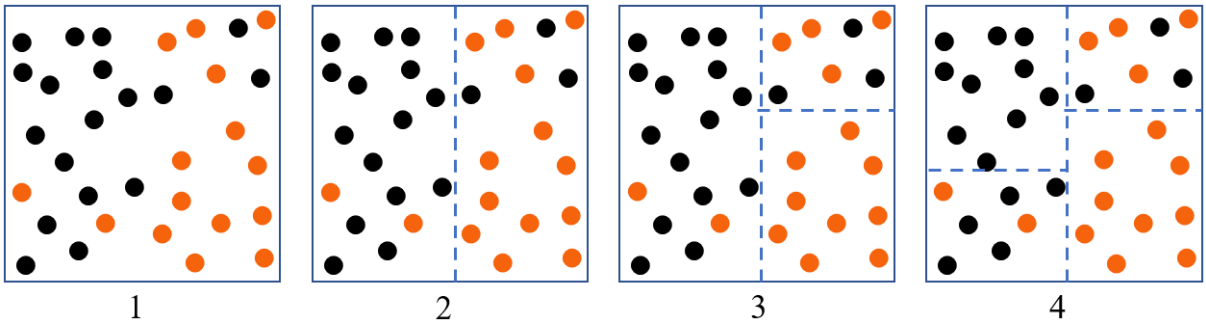


Figure 3.1: An example of recursive partitioning for a classification task with two explanatory variables and two outcome classes (denoted by orange and black dots).

the $\mathcal{P}$-space as follows:

$$u(s_{it}, d_{it}) = \bar{u}(x_{it}, \mathbf{\Pi}^*(q_{it}), d_{it}; \theta_1, \mathbf{\Pi}^*) + \epsilon_{itj} \tag{3.7}$$

Similarly, we can also write the value function and the choice-specific value function in terms of the discretized state space. Conceptually, once we have a perfect discretization $\mathbf{\Pi}^*$, we can treat $\mathbf{\Pi}^*(q)$ as a categorical variable in addition to $\mathbf{X}$, and ignore q. As such, all the methods available for the estimation of dynamic discrete choice models (e.g., nested fixed point, two-step estimators) are directly applicable here, with $\{\mathbf{X}, \mathcal{P}\}$ as the state-space. Thus, all the consistency and efficiency properties of the estimator used would directly translate to this setting.

3.4 Our Discretization Approach

We now present our discretization algorithm. We first explain the recursive partitioning method for a general objective function in §3.4.1. Next, in §3.4.2, we formulate a recursive partitioning scheme for the dynamic discrete choice model discussed above. We summarize the properties of our algorithm in §3.4.2. Finally, we discuss model selection and hyper-parameter optimization in §3.4.3.

3.4.1 Discretization using Recursive Partitioning

Recursive partitioning is a meta-algorithm for partitioning a covariate space into disjoint partitions to maximize some objective function. In each iteration, the algorithm uses an objective function as the criterion for selecting the next split among all the potential candidate splits. A split, divides a partition into two along based one of the variables in the covariate space. The following pseudo-code

presents a general recursive partitioning algorithm, where the goal is to maximize the objective function $\mathcal{F}(\mathbf{\Pi})$.

- Inputs: Objective function $\mathcal{F}(\mathbf{\Pi})$ that takes a partitioning $\mathbf{\Pi}$ as input and outputs a score.
- Initialize $\mathbf{\Pi}_0$ as one partition equal to the full covariate space.
- Do the following until the stopping criterion is met, or $\mathcal{F}(\mathbf{\Pi}_t) = \mathcal{F}(\mathbf{\Pi}_{t-1})$
  - For every $\pi \in \mathbf{\Pi}_t$, every $q \in \mathbf{Q}$, and every value $v \in \text{range}(q)$ in π
    - $\Delta(\pi, q, v) = \mathcal{F}(\mathbf{\Pi}_t + \{\pi, q, v\}) - \mathcal{F}(\mathbf{\Pi}_t)$
  - $\{\pi', q^*, v^*\} = \text{argmax}_{\{\pi,q,v\}} \Delta(\pi, q, v)$
  - $\mathbf{\Pi}_{t+1} = \mathbf{\Pi}_t + \{\pi', q^*, v^*\}$

Figure 3.1 shows four iterations of recursive partitioning applied to a classification task. The partitioning $\mathbf{\Pi}$ maps the two-dimensional covariate space into four different partitions.

3.4.2 Recursive Partitioning for Dynamic Discrete Choice Models

The ultimate goal of our discretization exercise is to estimate θ by maximizing the likelihood function in Equation (3.6). To do so, we first need to find a perfect discretization, i.e., a discretization that satisfies Equation (3.4). Thus, an intermediate goal is to find the partitioning $\mathbf{\Pi}^*$. The key question then becomes what should be the objective function for the recursive partitioning algorithm that helps us achieve the two goals discussed above. One naive approach would be to simply use the log-likelihood shown in Equation (3.6). However, this is problematic because of three reasons. First, this likelihood is a function of both $\mathbf{\Pi}^*$ and θ. A naive implementation of a recursive partitioning algorithm requires estimating the optimal θ in every iteration for a given $\mathbf{\Pi}_t$. However, estimating θ is computationally expensive in a DDC setting because we need to calculate the discounted future utility associated with a given choice to estimate the primitives of agents' utility function fully. Doing this at every *potential* split in each iteration of the algorithm is very computationally expensive (and infeasible when the $\mathbf{Q}$-space is large). Second, the likelihood function contains two sets of outcomes: (i) choice probabilities and (ii) state transition probabilities. Therefore, any data-driven approach to discretize the state space must account for both of these outcomes. This makes the splitting process more complex than usual recursive partitioning algorithms such as CART/causal tree, where there is only one set of estimands to be considered. Third, unlike a

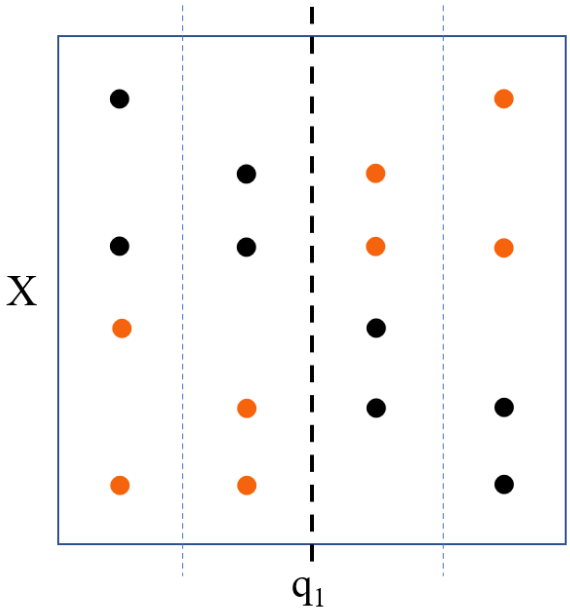


Figure 3.2: The dashed lines are candidate splits. Orange and black dots are observations with decision 1 and 2 respectively. Agents' choices across different Xs in the left side of the thick dashed line are different from their choices in the right side.

standard recursive partitioning algorithm, where we split on all the state variables, here we only split on $\mathbf{Q}$ since $\mathbf{X}$s are known (or assumed) to influence outcomes by definition. As such, we need an algorithm which only splits on a subset of state variables ($\mathbf{Q}$), but considers all the state variables ($\{\mathbf{X}, \mathbf{Q}\}$) to estimate the choice probabilities and state transitions; see Figure 3.2 for an example.

We design a recursive partitioning method that addresses all these three problems. To address the first problem, we separate the discretization problem from estimation and suggest an objective function for recursive partitioning algorithm that is fully non-parametric, i.e., is not dependent on θ – thus, its calculation is computationally cheap. Next, to address the second problem, we split our objective function into two sub-objectives, whose relative importance can be learnt from the data using model selection. Finally, to address the third problem, we customize the splitting procedure such that it only splits on $\mathbf{Q}$ and not $\mathbf{X}$. We discuss these ideas in detail below.

Nonparametric Objective Function

As we discussed in §3.3.2, we can estimate the primitives of agents' utilities and state transitions by maximizing the log-likelihood shown in Equation (3.6). This likelihood is a function of both

the primitive parameters and the discretization $\mathbf{\Pi}^*$. Nevertheless, we do not need to solve the maximization problem on both dimensions simultaneously. Conceptually, the discretization goal is to separate $\mathbf{Q}$ into buckets such that observations within the same bucket behave similarly. A key insight here is that we can achieve this objective without estimating θ, by simply using the non-parametric estimates of choice probabilities and state transitions observed in the data. That is, we can re-formulate the objective function from Equation (3.6) in non-parametric terms. Our proposed objective function is thus a weighted sum of the nonparametric equivalents of the first and second components of Equation (3.6) as shown below.

$$\mathcal{F}(\mathbf{\Pi}) = \mathcal{F}_{dc}(\mathbf{\Pi}) + \lambda\, \mathcal{F}_{tr}(\mathbf{\Pi}) \tag{3.8}$$

where $\mathbf{\Pi}$ is a partitioning function, and $\mathcal{F}_{dc}(\mathbf{\Pi})$ and $\mathcal{F}_{tr}(\mathbf{\Pi})$ are:

$$\begin{aligned}\mathcal{F}_{dc}(\mathbf{\Pi}) &= \sum_{i=1}^{N}\sum_{t=1}^{T} \log \frac{N(x_{it}, \mathbf{\Pi}(q_{it}), d_{it}; \mathbf{\Pi})}{N(x_{it}, \mathbf{\Pi}(q_{it}); \mathbf{\Pi})} \\ &= \sum_{x\in\mathbf{X}}\sum_{\pi\in\mathbf{\Pi}}\sum_{j\in\mathbf{J}} N(x, \pi, j; \mathbf{\Pi}) \log \frac{N(x, \pi, j; \mathbf{\Pi})}{N(x, \pi; \mathbf{\Pi})}\end{aligned} \tag{3.9}$$

$$\begin{aligned}\mathcal{F}_{tr}(\mathbf{\Pi}) &= \sum_{i=1}^{N}\sum_{t=2}^{T} \log \frac{N(x_{it}, \mathbf{\Pi}(q_{it}), x_{it-1}, \mathbf{\Pi}(q_{it-1}), d_{it-1}; \mathbf{\Pi})}{N(x_{it-1}, \mathbf{\Pi}(q_{it-1}), d_{it-1}; \mathbf{\Pi}) N(x_{it}, \mathbf{\Pi}(q_{it}))} \\ &= \sum_{x\in\mathbf{X}}\sum_{\pi\in\mathbf{\Pi}}\sum_{j\in\mathbf{J}}\sum_{x'\in\mathbf{X}}\sum_{\pi'\in\mathbf{\Pi}} N(x, \pi, x', \pi', j; \mathbf{\Pi}) \log \frac{N(x, \pi, x', \pi', j; \mathbf{\Pi})}{N(x, \pi; \mathbf{\Pi}) N(x', \pi', j; \mathbf{\Pi})}\end{aligned} \tag{3.10}$$

The function $N(.)$ counts the number of observations for a given condition. For example, $N(x, \pi, j)$ is the number of observations where the agent chose decision j in state $\{x, \pi\}$ and $N(x, \pi, x', \pi', j)$ is the number of observations that chose decision j in state $\{x', \pi'\}$, and transitioned to $\{x, \pi\}$.

The weighting parameter λ is a multiplier that specifies the relative importance of the state transition likelihood in comparison to decision likelihood in our algorithm. To make this parameter more intuitive and generalizable across different datasets, we decompose it as follows:

$$\lambda = \lambda_{adj} \times \lambda_{rel}, \quad \text{where } \lambda_{adj} = \frac{\mathcal{F}_{dc}(\mathbf{\Pi}_0)}{\mathcal{F}_{tr}(\mathbf{\Pi}_0)}. \tag{3.11}$$

Here, $\mathbf{\Pi}_0$ is the full covariate space of $\mathbf{Q}$ as one partition. λ_{rel} is a hyperparameter that captures the

relative importance state transition and decision likelihoods in our objective function and should be learnt from the data using cross-validation. For example, $\lambda_{rel} = 2$ implies that the recursive partitioning algorithm values one percentage lift in $\mathcal{F}_{tr}$ twice as much a one percentage lift in $\mathcal{F}_{dc}$ when selecting the next split. The optimal λ_{rel} can vary with the application. In settings where there is more information in the state transition, the optimal λ_{rel} will be higher whereas a smaller λ_{rel} is better when the choice probabilities are more informative. Therefore, it is important to choose the right value of λ_{rel} to prevent over-fitting and learn a good discretization.

A couple of additional points of note regarding our proposed objective function. First, the first lines in both Equations (3.9) and (3.10) are aggregating observations over individuals and time whereas the second lines are aggregating over all possible decisions and state transitions weighted by their occurrence. While the two representations are equivalent, we use the latter one in the rest of the paper since it is more convenient. Second, the objective function in Equation (3.8) is the non-parametric version the log-likelihood in Equation (3.6) (with the hyperparameter λ added in for data-driven optimization). The terms $\frac{N(x_{it},\mathbf{\Pi}(q_{it}),d_{it};\mathbf{\Pi})}{N(x_{it},\mathbf{\Pi}(q_{it});\mathbf{\Pi})}$, and $\frac{N(x_{it},\mathbf{\Pi}(q_{it}),x_{it-1},\mathbf{\Pi}(q_{it-1}),d_{it-1};\mathbf{\Pi})}{N(x_{it-1},\mathbf{\Pi}(q_{it-1}),d_{it-1};\mathbf{\Pi})}$ are the non-parametric counterparts of $\Pr(d_{it}|x_{it},\mathbf{\Pi}(q_{it});\theta_1,\mathbf{\Pi})$ and $\Pr(x_{it},\mathbf{\Pi}(q_{it})|x_{it-1},\mathbf{\Pi}(q_{it-1}),d_{it-1};\theta_2,\mathbf{\Pi})$, respectively. Therefore, maximizing $\mathcal{F}(\mathbf{\Pi})$ is equivalent to maximizing the original likelihood function.

Algorithm Properties

We now establish two key properties of the recursive partitioning algorithm proposed here. First, as shown in Appendix B.1, the non-parametric log-likelihood shown in Equation (3.6) is non-decreasing at each iteration of the algorithm. Formally, we have:

$$\mathcal{L}(\theta_t^*,\mathbf{\Pi}_t) \leq \mathcal{L}(\theta_{t+1}^*,\mathbf{\Pi}_{t+1})$$
$$\text{where} \begin{cases} \theta_t^* = \text{argmax}_\theta\, \mathcal{L}(\theta,\mathbf{\Pi}_t) \\ \theta_{t+1}^* = \text{argmax}_\theta\, \mathcal{L}(\theta,\mathbf{\Pi}_{t+1}) \end{cases}$$

Second, as shown in Appendix B.2, for $\lambda > 0$, the final discretization $\mathbf{\Pi}^* = \text{argmax}_{\mathbf{\Pi}}\, \mathcal{F}(\mathbf{\Pi})$ converges to a perfect discretization[3]. Together, these two properties ensure that: (a) the algorithm will increase the likelihood at each step and converge, and (b) upon convergence, it will achieve a perfect discretization.

[3]Under some conditions that we discuss in the proofs

In addition to the above two properties, our proposed algorithm shares the desirable properties of other recursive partitioning-based algorithms. First, it has linear time complexity with respect to the dimensionality of $\mathbf{Q}$ [Sani et al., 2018] – if the number of dimensions in $\mathbf{Q}$ doubles, the requires time to discretize the model doubles at most. This is in contrast to conventional DDC estimation methods such as the Nested Fixed Point algorithm, where the execution time increases exponentially in observable state variables. Second, the proposed algorithm is robust to the scale of the state variables and only depends on the ordinality of the state variable. As such, any changes in the scale of variables in $\mathbf{Q}$ does not change the estimated partition. Finally, since the algorithm splits on a variable only if it increases the log-likelihood shown in Equation (3.8), it is robust to the presence of irrelevant state variables.

Together, these properties allow the researcher to include all potential observable variables that might affect the agents' decisions in $\mathbf{Q}$ without significantly increasing the compute cost. This is valuable in the current data-abundant era, where firms have massive amounts of user-level data but lack theoretical insight on the effect (if any) of these variables on users' decisions. Indeed, one of the advantages of the method is that it allows the researcher to make post-hoc inference or theory-discovery in a data-rich environment. In sum, our proposed algorithm allows researchers and industry practitioners to reduce the estimation bias associated with ignoring relevant state variables, improve the fit of their models, and draw post-hoc inference in a high-dimensional setting without theoretical guidance, all at relatively low compute cost.

3.4.3 Hyperparameter Optimization and Model Selection

We now discuss the details of the hyperparameters used and the tuning procedure for model selection. Like other machine learning approaches, our recursive partitioning algorithm also needs to address bias-variance trade-off. If we discretize $\mathbf{Q}$ into too many small partitions, the set of partitions (and the corresponding estimates of choice and state transition probabilities) will not generalize beyond the training data. Thus, we need a set of hyperparameters that constrain or penalize model complexity. We can then tune these hyper-parameters using a validation procedure.[4]

[4]See Hastie et al. [2009] for a detailed discussion of the pros-cons of different validation procedure.

Set of Hyperparameters

We implemented two hyperparameters that are commonly used in other recursive partitioning-based models: *minimum number of observations* and *minimum lift*.[5] The former stops the algorithm from making very small partitions by ruling out splits (at any given iteration) that produce partitions with observations fewer than the *minimum number of observations*. The latter prevents over-fitting by stopping the partitioning process if the next split does not increase the objective function by the *minimum lift*, which is defined as $\frac{\mathcal{F}(\mathbf{\Pi}_{t+1})-\mathcal{F}(\mathbf{\Pi}_t)}{\mathcal{F}(\mathbf{\Pi}_t)}$. In addition to these two standard hyperparameters, we also include another one: *maximum number of partitions*, which stops the recursive partitioning after the algorithm has reached the *maximum number of partitions*. This hyper-parameter not only controls overfitting, but can also help with identification concerns because it allows the researchers to restrict the number of partitions, and hence the number of estimands. Together, these three hyperparameters ensure that the algorithm does not overfit and produces a generalizaeble partition that is valid out-of-sample.

In addition to these three hyper-parameters that constrain partitioning, we have another key hyperparameter, λ_{rel}, that shapes the direction of partitioning. As discussed earlier, λ_{rel} captures the relative importance of the two parts of the objective function when selecting the next split. If λ_{rel} is set close to zero, the discretization procedure prioritizes splits that explain the choice probabilities. As λ_{rel} increases, the recursive partitioning tends to choose splits that explain the variation in the state transition. λ_{rel} can be either tuned using a validation procedure or set manually by the researched based on their intuition or the requirements of the problem at hand. We present some simulations on the importance of picking the right value of λ_{rel} in §3.5.3.

Score function

Let η denote the set of all hyperparameters. To pick the right η for a given application setting, we need a measure of performance at a given η. Then, we can consider different values of η and select the one that maximizes out-of-sample performance (on the validation data). The model's performance is measured by calculating our objective function on a validation set using the training

[5] See the hyperparameters in Random Forest [Breiman, 2001], XGBoost [Chen and Guestrin, 2016], and Generalized Random Forest [Athey et al., 2019b].

set's estimated values. Formally, our score function for a given set of hyperparameters is:

$$score(\eta) = \frac{1}{1+\lambda_{rel}} \left(\sum_{x \in \mathbf{X}} \sum_{\pi \in \mathbf{\Pi}^*} \sum_{j \in \mathbf{J}} N^{val}(x, \pi, j; \mathbf{\Pi}^*) \log \frac{N^{trn}(x, \pi, j; \mathbf{\Pi}^*)}{N^{trn}(x, \pi; \mathbf{\Pi}^*)} \right. \tag{3.12}$$

$$\left. + \lambda_{rel} \lambda_{adj}^{val} \sum_{x \in \mathbf{X}} \sum_{\pi \in \mathbf{\Pi}^*} \sum_{j \in \mathbf{J}} \sum_{x' \in \mathbf{X}} \sum_{\pi' \in \mathbf{\Pi}^*} N^{val}(x, \pi, x', \pi', j; \mathbf{\Pi}^*) \log \frac{N^{trn}(x, \pi, x', \pi', j; \mathbf{\Pi}^*)}{N^{trn}(x, \pi; \mathbf{\Pi}^*) N^{trn}(x', \pi', j; \mathbf{\Pi}^*)} \right)$$

where $\mathbf{\Pi}^* = \text{argmax}_{\mathbf{\Pi}} \mathcal{F}(\mathbf{\Pi}; \eta)$ and $N^{trn}(.)$ and $N^{val}(.)$ are counting functions within the training and validation data, respectively. Notice that the main differences between Equations (3.8) and (3.12) is that here the model is *learnt* on the training data, but evaluated on the validation data. As such, the weight terms ($N(\cdot)$) and λ_{adj} are calculated on the validation dataset, while the probability terms are based on the training set. In addition, one minor difference is that this score is normalized by $\frac{1}{1+\lambda_{rel}}$. Without this normalization, any hyperparameter optimization procedure will tend to select larger λ_{rel} since without normalization, a larger λ_{rel} leads to a higher score.

In the validation procedure, we select the hyperparameter values that have the highest $score$ on the validation dataset. Note that having a separate validation set is not necessary. We can also use other hyper-parameter optimization techniques such as cross-validation, which is implemented in the accompanying software package.

Zero Probability Outcomes in Training Data

A final implementation issue that we need to address is the presence of choices and state transitions in the validation set that have not been seen in the training set.[6] For example, suppose that we have no observations where the agent chooses action j in state $\{x, \pi\}$ in the training data, but there exists such an observation in the validation data. Then the score in Equation (3.12) becomes negative infinity. This problem makes the hyperparameter optimization very sensitive to outliers.

To solve this problem, we turned to the Natural Language Processing (NLP) literature, which also deals with high-dimensional data and has studied this problem extensively. Several smoothing techniques have been proposed to resolve this issue in NLP models [Chen and Goodman, 1999]. One of the simplest smoothing methods that used in practice and can be applied to our setting is additive smoothing [Johnson, 1932], which assumes that we have seen all possible observations (i.e., choices and state-transitions) at least δ times, where usually $0 \leq \delta \leq 1$. This smoothing technique

[6]These kind of zero-probability occurrences are more likely in a finite sample as the dimensionality of the data increases.

removes the possibility of zero probabilities

A typical value for δ in the NLP literature is 1; however, the optimal value for δ depends on the data-generating process. A low value for δ penalizes the model more heavily for potential outliers. It also prevents the recursive partitioning step from creating small partitions since it increases the probability of having observations in the validation set that are not observed in the training set. On the other hand, a high value for δ may help with reducing overfitting by adding more noise to the data. In our simulations we set δ to 10^{-5}, which is a relatively small value. However, the researcher can use a different number depending on their data and application setting.

3.5 Monte Carlo Experiments

We now present two simulation studies to illustrate the performance of our algorithm. We use the canonical bus engine replacement problem introduced by Rust [1987] as the setting for our experiments. Rust's framework has been widely used as a benchmark to compare the performance of newly proposed estimators/algorithms for single-agent dynamic discrete choice models [Hotz et al., 1994, Aguirregabiria and Mira, 2002, Arcidiacono and Miller, 2011, Semenova, 2018]. We start by describing the original bus engine replacement problem and our high-dimensional extension of it in §3.5.1. In our first set of experiments in §3.5.2, we document the extent to which our algorithm is able to recover parameters of interest and compare its performance to a benchmark case where the high-dimensional state variables are ignored. In the second set of experiments in §3.5.3, we examine the importance of optimizing of the key hyperparameter, λ_{rel}.

3.5.1 Engine replacement problem

In Rust's model, a single agent (Harold Zurcher) chooses whether to replace the engine of a bus or continue maintaining it in each period. The maintenance cost is linearly increasing in the mileage of the engine while replacement constitutes a one-time lump-some cost. The inter-temporal trade-off is as follows – by replacing the bus engine, he pays a high replacement cost today and has lower maintenance costs in the future. If he chooses to not replace the bus engine, he forgoes the replacement cost, but will continue to pay higher maintenance cost in the future.

We start with the basic version of the model without the high-dimensional state variables. Here, the per-period utility function of two choices is given by:

$$\begin{aligned} u(x_{it}, d_{it} = 0) &= -c_m x_{it} + \epsilon_{i0t} \\ u(x_{it}, d_{it} = 1) &= -c_r + \epsilon_{i1t}, \end{aligned} \tag{3.13}$$

where x_{it} is the mileage of bus i at time t, c_m is the per-mile maintenance cost, c_r is the cost of replacing the engine, and $\{\epsilon_{i0t}, \epsilon_{i1t}\}$ are the error terms associated with the two choices. Next, we assume that the mileage increases by one unit in each period [7]. Formally:

$$\begin{aligned} &\text{if } \; d_{it} = 0, \;\; \text{then } \; x_{it+1} = x_{it} + 1 \;\; \text{if } \; x_{it} + 1 < 20 \;\; \text{, else } \; x_{it+1} = 19 \\ &\text{if } \; d_{it} = 1, \;\; \text{then } \; x_{it+1} = 1 \end{aligned} \tag{3.14}$$

The maximum mileage is capped at 20, i.e., after the mileage hits 19, it continues to stay there.

We now extend the problem to incorporate a set of high-dimensional state variables $\mathbf{Q}$ that can affect utility function and state transition. $\mathbf{Q}$ can include all the potential variables that can affect utilities and state transitions. For example, the mileage accrued may vary depending on the bus route, weather of the day, etc. Similarly, the replacement costs may vary by bus brands and/or economic conditions. A priori, it can be hard to identify which of these state variables and their combinations matter. Our method allows us to include all potential variables in the utility and state transition, and identify the partitions that matter.

In our simulations, we expand the utility function and state transition to include $\mathbf{Q}$ as follows:

$$\begin{aligned} u(x_{it}, \mathbf{\Pi}^*(q_{it}), d_{it} = 0) &= c_m x_{it} + \epsilon_{i0t} \\ u(x_{it}, \mathbf{\Pi}^*(q_{it}), d_{it} = 1) &= f_{dc}(\mathbf{\Pi}^*(q_{it})) + \epsilon_{i1t} \end{aligned} \tag{3.15}$$

$$\begin{aligned} &\text{if } \; d_{it} = 0, \;\; \text{then } \; x_{it+1} = x_{it} + f_{tr}(\mathbf{\Pi}^*(q_{it})) \;\; \text{if } \; x_{it} + f_{tr}(\mathbf{\Pi}^*(q_{it})) < 20, \;\; \text{else } \; x_{it+1} = 19 \\ &\text{if } \; d_{it} = 1, \;\; \text{then } \; x_{it+1} = f_{tr}(\mathbf{\Pi}^*(q_{it})) \end{aligned} \tag{3.16}$$

where q_{it} is the set for bus i at time t, and $f_{dc}(\mathbf{\Pi}^*(q_{it}))$ and $f_{tr}(\mathbf{\Pi}^*(q_{it}))$ are functions that specify the effect of q_{it} in the replacement cost, and state transition respectively. Note that we use $\mathbf{\Pi}^*(q_{it})$ instead of q_{it} since $\mathbf{\Pi}^*(q_{it})$ is a perfect discretization (and conveys the same information) as q_{it}. Finally, we set $c_m = -0.2$ in both our simulation studies.[8]

[7] This choice is just for the sake of increasing simplicity. However, our framework can easily handle stochastic state transitions as well.

[8] Note that the specific functional form is chosen for convenience, and the algorithm works even with a fully non-parametric utilities within a partition and non-parametric state transitions across partitions.

3.5.2 First simulation study

In the first set of experiments, our goal is to demonstrate the algorithm's performance and document the extent of bias when we do not account for $\mathbf{Q}$.

Data generating process

In this simulation study, $\mathbf{Q}$ consists 10 variables: $q^1, \ldots, q^{10}$, where $q^i \in \{0, 1, \ldots, 9\}$. However, only the first two variables affect the data generating process, and the rest of them are irrelevant. In principle, our algorithm is robust to inclusion of irrelevant state variables. Therefore, the extra eight variables in $\mathbf{Q}$ allows us to examine if this is indeed the case. In our simulations, q^1 and q^2 partition $\mathbf{Q}$ into four regions $\{\pi_1, \pi_2, \pi_3, \pi_4\}$ such that all the observations within a partition have the same choice and state transition probabilities. The partitions are given by:

$$\mathbf{\Pi}^*(q) = \begin{cases} \pi_1, & \text{if } q^1 < 5 \text{ and } q^2 < 5 \\ \pi_2, & \text{if } q^1 < 5 \text{ and } q^2 \geq 5 \\ \pi_3, & \text{if } q^1 \geq 5 \text{ and } q^2 < 5 \\ \pi_4, & \text{if } q^1 \geq 5 \text{ and } q^2 \geq 5 \end{cases}$$

The mileage transitions are as described in Equation (3.16). We also need to define the state transition in the $\mathbf{Q}$ space. Note that since we have a perfect discretization, only the state transition between πs matter – condition on the partition π the exact q is completely random. We consider three different state transition models in our simulations:

- No transition: There is only within-partition transition. This implies that the agent remains within the same partition in each period: $\mathbf{\Pi}^*(q_{it}) = \mathbf{\Pi}^*(q_{it+1})$.
- Random transition: Agents' transitions in the $\mathbf{Q}$-space are completely random. In each period, an agent's randomly transition from one partition to another such that: $\mathbf{\Pi}^*(q_{it}) \perp \mathbf{\Pi}^*(q_{it+1})$.
- Sparse transition: After each period, the agent remains in the same partition with probability 0.5 an moves to the next partition with probability 0.5.[9]

  Next, we consider two different cases for f_{tr} and f_{dc} in Equations (3.15) and (3.16) as follows:

$$f_{tr}([\pi_1, \pi_2, \pi_3, \pi_4]) = \begin{cases} [0, 1, 2, 3] \text{ in the dissimilar mileage transition case} \\ [1, 1, 1, 1] \text{ in the similar mileage transition case} \end{cases} \tag{3.17}$$

[9]The order of partitions is as follows: π_2 is after π_1, π_3 is after π_2, π_4 is after π_3, and π_1 is after π_4.

$$f_{dc}([\pi_1, \pi_2, \pi_3, \pi_4]) = \begin{cases} [-7, -6, -5, -4] \text{ in the dissimilar replacement costs case} \\ [-5, -5, -5, -5] \text{ in the similar replacement costs case} \end{cases} \tag{3.18}$$

Together, this gives us $3 \times 2 \times 2$ possible scenarios for the data generating process. We simulate data and recover the parameters for each of these cases. While doing so, we set the hyper-parameters as follows: we minimum lift to 10^{-10}, minimum number of observations to 1, and λ_{rel} to 1. Further, for each case, we run the algorithm (and then perform the estimation) for four different values of *maximum partitions*: $1, 2, 4,$ or 6. The single partition case is equivalent to ignoring the high-dimensional state variable $\mathbf{Q}$. Setting *maximum partitions* to 2 and 6 allows us to examine how our algorithm performs when we allow for under-discretization and over-discretization, respectively.

Results

We run 12 Monte Carlo simulations, each of which employs a different data generating process. Each simulation consists of 100 rounds of data generation from a given data-generating process. And in each round, we generate data for 400 buses for 100 time periods. For each round of simulated data, we first use our algorithm to discretize the high-dimensional state space $\mathbf{Q}$. Then, as we discuss in §3.3.2, we simply treat each of the partitions in $\mathbf{\Pi}^*(q)$ as an extra categorical variable in addition to $\mathbf{X}$ at the estimation stage. As such, we provide the partitions obtained from our algorithm and the mileage to the nested fixed-point algorithm and estimate the utility parameters.

There are two different cases for f_{tr}, two different cases for f_{dc}, and three different cases for transition in $\mathbf{Q}$. We simulate all combinations of these cases, which result in a total of 12 data-generating process Monte Carlo simulations. For each data generation process case, we run 100 rounds of data generation to form bootstrap confidence intervals around our estimates. In each round, we simulate 4 different number of allowed partition and parameter estimation. The results of estimated value for c_m in these simulations are presented in Table 3.1. Table 3.2 presents the estimated replacement cost in cases where we neglect $\mathbf{Q}$ by allowing one partition and the case where we allow four partitions.

As the table of results depicts, neglecting $\mathbf{Q}$ leads to biased estimates in many settings, even when q_{it} does not directly affect the flow utility function, and there is no serial correlation in q over time. The bias arises because by neglecting $\mathbf{Q}$, we violate the DDC modeling assumptions in favor of computational feasibility. To be more specific, the conditional independence assumption states that ϵ_{it+1} and x_{it+1} are independent of ϵ_{it}. When we neglect $\mathbf{Q}$ in the estimation procedure, it is captured in the error term. Mileage transition is a function of q, in the cases where $\mathbf{Q}$ has a diverse

Effect on Replacement Cost	Effect on Mileage Transition	Transition in $\Pi^*(\mathbf{Q})$ Sapce	Number of Allowed Partitions			
			1	**2**	**4**	**6**
Dissimilar	Dissimilar	No transition	-0.135 (-0.144, -0.127)	-0.173 (-0.18, -0.167)	-0.194 (-0.204, -0.187)	-0.193 (-0.204, -0.186)
Dissimilar	Dissimilar	Random transition	-0.149 (-0.154, -0.142)	-0.186 (-0.191, -0.18)	-0.2 (-0.206, -0.193)	-0.199 (-0.206, -0.192)
Dissimilar	Dissimilar	Sparse transition	-0.123 (-0.129, -0.118)	-0.182 (-0.198, -0.167)	-0.199 (-0.207, -0.191)	-0.199 (-0.207, -0.191)
Dissimilar	Similar	No transition	-0.165 (-0.172, -0.158)	-0.191 (-0.2, -0.183)	-0.199 (-0.207, -0.19)	-0.198 (-0.207, -0.19)
Dissimilar	Similar	Random transition	-0.171 (-0.177, -0.162)	-0.193 (-0.201, -0.184)	-0.2 (-0.209, -0.192)	-0.199 (-0.209, -0.192)
Dissimilar	Similar	Sparse transition	-0.17 (-0.176, -0.164)	-0.096 (-0.119, -0.067)	-0.201 (-0.211, -0.189)	-0.201 (-0.211, -0.188)
Similar	Dissimilar	No transition	-0.178 (-0.188, -0.17)	-0.193 (-0.201, -0.187)	-0.199 (-0.208, -0.192)	-0.199 (-0.208, -0.192)
Similar	Dissimilar	Random transition	-0.185 (-0.194, -0.179)	-0.197 (-0.205, -0.189)	-0.2 (-0.208, -0.192)	-0.2 (-0.207, -0.191)
Similar	Dissimilar	Sparse transition	-0.174 (-0.182, -0.168)	-0.236 (-0.247, -0.224)	-0.2 (-0.209, -0.193)	-0.2 (-0.209, -0.193)
Similar	Similar	No transition	-0.2 (-0.205, -0.192)	-0.199 (-0.205, -0.192)	-0.199 (-0.204, -0.189)	-0.198 (-0.204, -0.188)
Similar	Similar	Random transition	-0.2 (-0.207, -0.193)	-0.199 (-0.207, -0.193)	-0.199 (-0.207, -0.192)	-0.198 (-0.206, -0.191)
Similar	Similar	Sparse transition	-0.199 (-0.209, -0.192)	-0.199 (-0.208, -0.192)	-0.198 (-0.208, -0.19)	-0.198 (-0.207, -0.19)

Table 3.1: The estimated mileage maintenance cost coefficient , c_m, and their 96% bootstrap confidence interval around the mean in each of the 12 Monte Carlo simulations. Each row is a result of 100 rounds of simulation.

Effect on Replacement Cost	**Effect on Mileage Transition**	**Transition in $\Pi^*(\mathbf{Q})$ Sapce**	Incorporating discretized $\mathbf{Q}$				Neglecting $\mathbf{Q}$
			$f_{dc}(\pi_1)$	$f_{dc}(\pi_2)$	$f_{dc}(\pi_3)$	$f_{dc}(\pi_4)$	c_r
Dissimilar	Dissimilar	No transition	-6.812	-5.845	-4.843	-4.435	-5.408
			(-7.115, -6.589)	(-6.121, -5.635)	(-5.084, -4.642)	(-4.724, -3.908)	(-5.76, -5.148)
Dissimilar	Dissimilar	Random transition	-7.006	-6.008	-4.998	-3.995	-4.962
			(-7.179, -6.834)	(-6.132, -5.853)	(-5.165, -4.831)	(-4.13, -3.852)	(-5.099, -4.846)
Dissimilar	Dissimilar	Sparse transition	-7.015	-5.987	-4.984	-3.984	-4.886
			(-7.352, -6.773)	(-6.201, -5.804)	(-5.139, -4.846)	(-4.222, -3.773)	(-5.046, -4.766)
Dissimilar	Similar	No transition	-6.975	-5.964	-4.977	-4.0	-4.701
			(-7.252, -6.69)	(-6.162, -5.74)	(-5.142, -4.82)	(-4.13, -3.86)	(-4.906, -4.548)
Dissimilar	Similar	Random transition	-7.025	-5.996	-4.997	-3.995	-4.661
			(-7.327, -6.775)	(-6.211, -5.83)	(-5.183, -4.829)	(-4.165, -3.842)	(-4.769, -4.507)
Dissimilar	Similar	Sparse transition	-7.061	-6.028	-5.021	-4.012	-4.731
			(-7.394, -6.769)	(-6.253, -5.757)	(-5.244, -4.813)	(-4.221, -3.803)	(-4.858, -4.618)
Similar	Dissimilar	No transition	-5.064	-5.019	-4.977	-4.925	-5.323
			(-5.218, -4.901)	(-5.163, -4.857)	(-5.138, -4.811)	(-5.083, -4.732)	(-5.524, -5.115)
Similar	Dissimilar	Random transition	-5.041	-5.01	-4.986	-4.957	-5.039
			(-5.191, -4.905)	(-5.151, -4.883)	(-5.14, -4.856)	(-5.115, -4.84)	(-5.221, -4.9)
Similar	Dissimilar	Sparse transition	-5.059	-5.021	-4.994	-4.956	-5.045
			(-5.237, -4.905)	(-5.196, -4.854)	(-5.149, -4.847)	(-5.112, -4.812)	(-5.219, -4.864)
Similar	Similar	No transition	-5.043	-4.998	-4.963	-4.911	-4.994
			(-5.196, -4.886)	(-5.131, -4.861)	(-5.086, -4.823)	(-5.039, -4.779)	(-5.11, -4.869)
Similar	Similar	Random transition	-5.033	-4.994	-4.975	-4.936	-5.0
			(-5.25, -4.874)	(-5.162, -4.851)	(-5.12, -4.831)	(-5.099, -4.772)	(-5.143, -4.863)
Similar	Similar	Sparse transition	-5.005	-4.982	-4.964	-4.939	-4.989
			(-5.166, -4.842)	(-5.118, -4.817)	(-5.102, -4.813)	(-5.094, -4.808)	(-5.129, -4.827)

Table 3.2: The estimated replacement cost and their 96% bootstrap confidence interval around the mean in each of the 12 Monte Carlo simulations. Each row is a result of 100 rounds of simulation.

effect on mileage transition. Thus, when we do not incorporate $\mathbf{Q}$ as an observable, x_{it+1} would be correlated with ϵ_{it} – a violation of our estimation assumption. Similarly, in the "No transition" and "Sparse transition" cases, neglecting $\mathbf{Q}$ leads to a correlation between ϵ_{it+1} and ϵ_{it}, another violation of our estimation assumptions. These violations lead to much more severe bias in the case where $\mathbf{Q}$ affects the flow utility.

Interestingly, serial correlation in error terms does not seem to bias the estimated parameters if it does not affect the utility function or the observable part of state variables. As the first three rows of the table show, estimation by neglecting $\mathbf{Q}$ recovered the data generating parameters without bias. However, serial correlation worsens the bias in the presence of correlation between x_{it+1} and ϵ_{it} or when $\mathbf{Q}$ affects the flow utility.

Another interesting finding is the effect of over-discretization and under-discretization. As expected, over-sampling does not introduce any bias in the estimation procedure, since it does not violate any estimation assumption. However, the estimation variance increases slightly since we estimate more parameters when we over-discretize. As Table 3.1 presents under-discretization has lower bias than ignoring $\mathbf{Q}$. However, in two cases allowing only two partitions leads to a higher bias than having only one partition, making under-discretization worse than no discretization.

While this research aims not to calculate the amount of misspecification bias, and our results are not generalizable, our simulation results show that the amount of bias from neglecting $\mathbf{Q}$ can be huge. Even when $\mathbf{Q}$ is not directly involved in the flow utility, neglecting it can bias our estimation as much as 13%. These results highlight the importance of controlling for potential variables that can affect our DDC modeling procedure. In addition, our results shows that over-discretization is not an issue, especially in setting where the number of observations is large enough to reduce the estimation variance concerns. It also shows that we can benefit from discretization even when we under-discretize $\mathbf{Q}$. These two results argue in favor of using the discretization algorithm compared to neglecting $\mathbf{Q}$.

3.5.3 Second Simulation Study

Selecting an optimal λ_{rel} is important for proper discretization. For example, assigning a big λ_{rel} in settings where the state transition is noisy and less informative makes the algorithm pick up noise as a signal in its discretization. Thus, selecting an optimal λ_{rel} becomes more important in settings where we have few observations. Also, when the dimension of $\mathbf{Q}$ is high, the algorithm is more likely to find noisy patterns as we check many more variables for a potential split. The goal of this simulation study is to dig deeper into λ_{rel} selection by running a series of Monte Carlo simulations

in different data generating process settings.

Data generating process

The data generating processes in this study are similar to the previous study except for some minor differences. First, the dimension of $\mathbf{Q}$ is 30, and only the first 10 variables affect the data-generating process, and the rest are irrelevant. We randomly discretize $\mathbf{Q}$ into 15 partitions using the process explained in the appendix B.3. We generate 100 random discretizations to evaluate the performance of the algorithm across different settings. Each discretization is used in eight different data-generating scenarios, as we explain next.

Similar to the first simulation study we simulate two cases for f_{tr}: i) $f_{tr}(\pi) = 1$ for all 15 partitions, or ii) $f_{tr}(\pi)$ is a random number from set $\{0, 1, 2, 3\}$. The transition of state in $\mathbf{\Pi}^*$ has two scenarios: i) random transition or ii) sparse transition as defined in the previous simulation study. The only difference is that in the sparse transition case, the state remains the same with probability $1/3$ or goes to either of the next two states with the same probability[10]. Variation across these two dimensions lets us vary the amount of information available in the state transition. For example, there is less information in the state transition in the case where the transition across partitions in $\mathbf{\Pi}^*$ is random, and $f_{tr}(\pi) = 1$. In addition, we vary the amount of data by simulating two scenarios: i) 100 buses in 100 periods and ii) 100 buses in 400 periods. Therefore, in total, we simulate eight different data-generating scenarios on each partitioning.

Finally, we test the effect of λ_{rel} by varying it across different values and evaluate the performance of the resulting partition. To remove the effect of other hyper-parameters and only capture the effect of λ_{rel}, we do not run hyperparameter optimization. We set the minimum number of observations and minimum lift to 1, and 10^{-10} respectively. We set the total number of partitions to 15 – the actual total number of partitions in the true partitioning. We then evaluate the out-of-sample performance of different value for $\lambda_{rel} \in \{0, 0.2, 0.5, 1, 2, 5, 100\}$ across different scenarios using the score function defined in equation 3.12. We also calculate the number of matched partitions between the true data-generating discretization and the estimated discretization generated by the algorithm.

[10]The defined ordinality of states is completely random, and is not defined in a meaningful way.

Number of Periods	Transition in $\mathbf{\Pi}^*(Q)$ space	$\mathbf{\Pi}^*(Q)$ affect mileage transition	Value of λ_{rel} 0	0.2	0.5	1	2	5	100
100	Sparse	Yes	-3722	-3481	-3274	-3121	-2977	-2864	**-2766**
	Sparse	No	-3762	-3536	-3407	-3323	-3245	**-3200**	-3228
	Random	Yes	-3743	-3762	**-3730**	-3768	-3831	-3913	-4062
	Random	No	**-3698**	-3781	-3867	-3959	-4063	-4192	-4373
400	Sparse	Yes	-13287	-12969	-12606	-12239	-11864	-11519	**-11217**
	Sparse	No	-13815	-13650	-13469	-13264	-13059	**-12909**	-12987
	Random	Yes	**-13518**	-13620	-13693	-13770	-13863	-13974	-14187
	Random	No	**-13613**	-13921	-14231	-14542	-14856	-15177	-15556

Table 3.3: The calculated score for different values of λ_{rel} in different data-generating processes. A bigger λ_{rel} is better when there are more information in the state transition data. Scores are calculated using validation data.

Results

In total we run 100 (partitioning) $\times 8$ (scenarios)$\times 7$ (values for λ_{rel}) $= 5600$ partitioning simulations, the result of which are presented in tables 3.3 and 3.4. The bold numbers in each row in Table 3.3 are the λ_{rel}s with best score in the validation dataset. Please note that while the result for matched partitions is presented in Table 3.4, it is not a good scoring function for finding the optimal value for λ_{rel} for a couple of reasons: i) score is a measure of out-of-sample performance, while matched partitions in an in-sample performance measure, and ii) a higher partition match does not guarantee a better discretization since this measure does not incorporate the quality of unmatched partitions.

According to the results, the optimal value for λ_{rel} is significantly different from one data-generating process to another. However, as expected, the optimal value for λ_{rel} is bigger when the state transition is more informative. When the transition in $\mathbf{\Pi}^*(\mathbf{Q})$ space is sparse compared to the random case, the state transition is more informative. Also, the state transition is more informative when $\mathbf{\Pi}^*(\mathbf{Q})$ affects transition in mileage compared to the case where mileage transition is similar across different partitions in $\mathbf{\Pi}^*(\mathbf{Q})$. This pattern is presented in the results as the optimal value for λ_{rel} is highest when the transition in $\mathbf{\Pi}^*(\mathbf{Q})$ is sparse, and $\mathbf{\Pi}^*(\mathbf{Q})$ affects the mileage transition. The optimal λ_{rel} is zero when transition in $\mathbf{\Pi}^*(\mathbf{Q})$ is random, thus not informative for finding the discretization, and $\mathbf{\Pi}^*(\mathbf{Q})$ does not affect mileage transition.

Table 3.4 exhibits the potential value of information that is available in the state transition probabilities part of the data. If we only use the decision probabilities part of observations, i.e.,

Number of Periods	Transition in $\mathbf{\Pi}^*(Q)$ space	$\mathbf{\Pi}^*(Q)$ affect mileage transition	Value of λ_{rel} 0	0.2	0.5	1	2	5	100
100	Sparse	Yes	1.39	4.81	8.07	9.54	10.30	10.05	9.88
	Sparse	No	1.63	6.68	9.29	10.01	10.18	9.09	7.37
	Random	Yes	1.86	3.43	6.10	6.32	6.03	5.21	3.83
	Random	No	2.54	2.63	2.71	3.01	3.19	2.87	0.00
400	Sparse	Yes	3.62	6.74	8.73	9.68	10.25	10.14	10.16
	Sparse	No	5.04	8.42	9.40	9.98	10.08	9.44	7.56
	Random	Yes	4.16	6.20	7.36	7.06	6.55	5.56	4.24
	Random	No	7.28	7.45	7.47	7.55	7.64	7.28	0.29

Table 3.4: The number of matched partition between the true discretization and the discretization generated by different values of λ_{rel} in different data-generating processes.

set λ_{rel} to zero, the algorithm's ability to recover the partitions is weak. Increasing the number of observations helps with recovering the discretization; however, it is not as efficient as using the state transition part of the data. In cases where the state transition holds information about the discretization (either transition in the $\mathbf{\Pi}^*(\mathbf{Q})$ environment is sparse, or it affects mileage transition), setting a non-zero value for λ_{rel} boosts the performance of the discretizing algorithm substantially. Adding more data does not increase the performance of the algorithm for bigger values of λ_{rel} in these cases.

While not comprehensive, this experiment highlights the value of state transition information for finding the optimal state-space discretization. Using state transition information makes the state space discretization more efficient in using data than methods that only use the decision probability part of the likelihood function. In addition, we show that the optimal weight for the two parts of the likelihood function varies in different data-generating processes. Therefore, researchers should optimize the λ_{rel} and select the one with higher out-of-sample performance.

3.6 Limitations and Suggestions for Application

We prove that the discretization offered by the algorithm converges to a perfect discretization. Nevertheless, it might not happen in empirical applications. We might not have enough observations to fully recover the accurate discretization of the state transition or utility structure in the high-dimensional variable set $\mathbf{Q}$. This problem is highly likely in empirical settings where we have limited observations and too many variables in $\mathbf{Q}$. Additionally, state transition or utility structure

in $\mathbf{Q}$ might not be discrete. Theoretically, we can get very close to a perfect discretization in this empirical setting, but there is no perfect discretization to find.

In such scenarios, the proposed algorithm under-discretizes $\mathbf{Q}$, and therefore, the estimation step would still suffer from violation of dynamic discrete choice models assumptions. We expect even an under-discretized $\mathbf{Q}$ to improve the estimated parameters since we reduce the number of assumption violations by capturing more variation as observables. Nevertheless, as Table 3.1 presents, there are cases where neglecting $\mathbf{Q}$ leads to a better estimation than incorporating an under-discretized $\mathbf{Q}$. This issue arises because our objective function in the discretization step is different from our objective function in the estimation step. The amount and direction of bias depend on various reasons, including the parametric assumptions on $\mathbf{X}$, the state transition form, and the utility structure. The complexity of the situation makes it hard to find a debiasing solution.

There are some steps that scientists and practitioners can take to check the robustness of the estimates and prevent potential issues raised by this limitation. First, it is necessary to use hyper-parameter optimization to ensure that the generated discretization is generalizable to the data generating process, not just the training sample. We can check the performance of estimated parameters in both training and validation sets and the extent of the difference as a measure of discretization quality. Another method is to check the sensitivity of discretization and estimated parameters to the number of observations. For example, we can discretize $\mathbf{Q}$ and estimate the model with 80% of the data, and check how close the discretization and estimated parameters are to the case where we use all the data. We can ensure that we have enough data to find a suitable discretization if the estimates are close. Finally, we can check the validity of our model by analyzing its ability to predict counter-factual scenarios. Companies can run field experiments and use their results to measure the model's quality for predicting counter-factual scenarios.

We suggest using this algorithm in big data settings where we have many observations. We originally designed this algorithm to model users' subscription decisions in subscription-based software firms. These companies have tens of thousands of users who make the subscription decision either monthly, quarterly or yearly – the number of observations is enormous. These companies capture how users interact with the software and use functionalities offered by the software. Researchers can use our approach to model the subscription decision of users in this setting by adding price and promotion in $\mathbf{X}$, and the high-dimensional usage features and demographics of users in $\mathbf{Q}$.

3.7 Conclusion

Dynamic discrete choice modeling is used to estimate the underlying primitives of agents' behavior in settings where actions have future implications. Traditional methods for estimation of these models are computationally expensive, thus, inapplicable in high-dimensional data settings. Nonetheless, neglecting potential variables that affect agents' decisions or state transitions may bias the estimated parameters. As our simple experiment shows, this bias can be as high as 13% even when the neglected variables do not directly affect agents' flow utility.

This paper offers a new algorithm to estimate these methods in high-dimensional settings by dimension reduction using discretization. More specifically, our recursive partitioning-inspired approach let us control for a high-dimensional variable set $\mathbf{Q}$ in addition to the conventional independent variable set $\mathbf{X}$. We define the term *perfect discretization* and reformulate the conventional likelihood equation in the discretized space. We prove that the discretization offered by our algorithm converges to a perfect discretization[11]. In addition, we discuss some desirable properties of our algorithm, such as having linear time complexity with respect to the dimension of $\mathbf{Q}$ and being robust to scale and irrelevant variables.

Finally, we run two sets of simulation experiments, consisting of a series of Monte Carlo simulations using an extended version of the canonical Rust's bus engine problem. In the first simulation set, we show that our algorithm can recover the data generating process by controlling for the high-dimensional state space. We also point that the estimated parameters can be biased as much as 40% if we violate the DDC estimation assumptions by neglecting potential variables that affect the dynamic problem at hand. In the second simulation, we test the ability of our algorithm to recover the data generating discretization in complex and high-dimensional data generating settings. We show that our algorithm is more data-efficient due to its ability to capture the data variations in both agents' decisions and the state transitions. Surprisingly, the valuable data variation that exists in the state transition is usually neglected in traditional estimation methods.

Everyday companies are gathering more and more contextual and behavioral data from consumers, and consequently, are more inclined toward using high-dimensional friendly algorithms that do not require theoretical assumptions. Our proposed method allows companies to use these data to reduce the estimation bias in DDC modeling settings and help them understand new dimensions of agent behavior through post-hoc analysis. Besides, the algorithm helps researchers get around the accuracy-computational feasibility trade-off by offering an efficient method that lets them control

[11] Under some conditions standard in any recursive partitioning algorithm.

for a high-dimensional variable set.

BIBLIOGRAPHY

V. Aguirregabiria and P. Mira. Swapping the nested fixed point algorithm: A class of estimators for discrete markov decision models. *Econometrica*, 70(4):1519–1543, 2002.

E. T. Anderson and D. I. Simester. Long-run effects of promotion depth on new versus established customers: three field studies. *Marketing Science*, 23(1):4–20, 2004.

A. Ansari and C. F. Mela. E-customization. *Journal of marketing research*, 40(2):131–145, 2003.

P. Arcidiacono and J. B. Jones. Finite mixture distributions, sequential likelihood and the em algorithm. *Econometrica*, 71(3):933–946, 2003.

P. Arcidiacono and R. A. Miller. Conditional choice probability estimation of dynamic discrete choice models with unobserved heterogeneity. *Econometrica*, 79(6):1823–1867, 2011.

P. Arcidiacono, P. Bayer, F. A. Bugni, and J. James. Approximating high-dimensional dynamic models: Sieve value function iteration. Technical report, National Bureau of Economic Research, 2012.

P. Arcidiacono, P. Bayer, F. A. Bugni, and J. James. *Approximating high-dimensional dynamic models: Sieve value function iteration*. Emerald Group Publishing Limited, 2013.

E. Ascarza. Retention futility: Targeting high-risk customers might be ineffective. *Journal of Marketing Research*, 55(1):80–98, 2018.

S. Athey and G. Imbens. Recursive partitioning for heterogeneous causal effects. *Proceedings of the National Academy of Sciences*, 113(27):7353–7360, 2016.

S. Athey and S. Wager. Efficient policy learning. *arXiv preprint arXiv:1702.02896*, 2017.

S. Athey and S. Wager. Estimating treatment effects with causal forests: An application. *arXiv preprint arXiv:1902.07409*, 2019.

S. Athey, R. Chetty, G. W. Imbens, and H. Kang. The surrogate index: Combining short-term proxies to estimate long-term treatment effects more rapidly and precisely. Technical report, National Bureau of Economic Research, 2019a.

S. Athey, J. Tibshirani, S. Wager, et al. Generalized random forests. *The Annals of Statistics*, 47(2): 1148–1178, 2019b.

P. J. Barwick and P. A. Pathak. The costs of free entry: an empirical study of real estate agents in greater boston. *The RAND Journal of Economics*, 46(1):103–145, 2015.

H. Benitez-Silva, G. Hall, G. J. Hitsch, G. Pauletto, and J. Rust. A comparison of discrete and parametric approximation methods for continuous-state dynamic programming problems. *manuscript, Yale University*, 2000.

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

J. Bergstra, D. Yamins, and D. D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. *Journal of Machine Learning Research*, 2013.

J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554, 2011.

G. Biau, L. Devroye, and G. Lugosi. Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research*, 9(Sep):2015–2033, 2008.

A. Blum, A. Kalai, and J. Langford. Beating the hold-out: Bounds for k-fold and progressive cross-validation. In *Proceedings of the twelfth annual conference on Computational learning theory*, pages 203–208. ACM, 1999.

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

L. Breiman. Arcing Classifier. *The Annals of Statistics*, 26(3):801–849, 1998.

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

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

L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees*. CRC press, 1984b.

M. Bruhn and D. McKenzie. In Pursuit of Balance: Randomization in Practice in Development Field Experiments. *American Economic Journal: Applied Economics*, 1(4):200–232, 2009.

P. Büchlmann and B. Yu. Analyzing bagging. *Annals of Statistics*, pages 927–961, 2002.

A. Buja and W. Stuetzle. Observations on bagging. *Statistica Sinica*, pages 323–351, 2006.

S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394, 1999.

T. Chen and C. Guestrin. 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, 2016.

H. K. Cheng and Y. Liu. Optimal software free trial strategy: The impact of network externalities and consumer uncertainty. *Information Systems Research*, 23(2):488–504, 2012.

K. X. Chiong, A. Galichon, and M. Shum. Duality in dynamic discrete-choice models. *Quantitative Economics*, 7(1):83–115, 2016.

T. M. Cover and J. A. Thomas. Entropy, relative entropy and mutual information. *Elements of information theory*, 2:1–55, 1991.

J. Davis and S. B. Heller. Using causal forests to predict treatment heterogeneity: An application to summer jobs. *American Economic Review*, 107(5):546–50, 2017.

D. Dey, A. Lahiri, and D. Liu. Consumer learning and time-locked trials of software products. *Journal of Management Information Systems*, 30(2):239–268, 2013.

T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923, 1998.

J.-P. Dubé and S. Misra. Personalized pricing and customer welfare. 2019.

J.-P. Dubé, J. T. Fox, and C.-L. Su. Improving the numerical performance of static and dynamic aggregate discrete choice random coefficients demand estimation. *Econometrica*, 80(5):2231–2267, 2012.

M. Dudík, J. Langford, and L. Li. Doubly robust policy evaluation and learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 1097–1104. Omnipress, 2011.

C. S. Dweck. *Mindset: The new psychology of success*. Random House Digital, Inc., 2008.

D. Dzyabura and J. R. Hauser. Active machine learning for consideration heuristics. *Marketing Science*, 30(5):801–819, 2011.

D. Dzyabura and J. R. Hauser. Recommending products when consumers learn their preference weights. *Marketing Science*, 38(3):417–441, 2019.

D. Dzyabura and H. Yoganarasimhan. 11. machine learning and marketing. page 255. Edward Elgar Publishing, 2018.

B. Efron. Estimation and accuracy after model selection. *Journal of the American Statistical Association*, 109(507):991–1007, 2014.

P. S. Fader and B. G. Hardie. Probability Models for Customer-base Analysis. *Journal of Interactive Marketing*, 23(1):61–69, 2009.

M. H. Farrell, T. Liang, and S. Misra. Deep neural networks for estimation and inference. *Econometrica*, 89(1):181–213, 2021.

E. M. Feit and R. Berman. Profit-maximizing a/b tests. *Available at SSRN 3274875*, 2019.

N. Fong, Y. Zhang, X. Luo, and X. Wang. Targeted promotions on an e-book platform: Crowding out, heterogeneity, and opportunity costs. *Journal of Marketing Research*, 56(2):310–323, 2019.

B. Foubert and E. Gijsbrechts. Try it, you'll like it—or will you? the perils of early free-trial promotions for high-tech service adoption. *Marketing Science*, 35(5):810–826, 2016.

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

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

J. H. Friedman and P. Hall. On bagging and nonlinear estimation. *Journal of statistical planning and inference*, 137(3):669–683, 2007.

Gartner. Forecast: Public cloud services, worldwide, 2016-2022, 4q18 update., 2019. URL https://www.gartner.com/en/newsroom/press-releases/2019-04-02-gartner-forecasts-worldwide-public-cloud-revenue-to-g.

Gartner. Gartner forecasts worldwide public cloud end-user spending to grow 23% in 2021, 2021. URL https://www.gartner.com/en/newsroom/press-releases/2021-04-21-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-

L. Guelman, M. Guillén, and A. M. Pérez-Marín. Uplift random forests. *Cybernetics and Systems*, 46(3-4):230–248, 2015.

T. Guo, S. Sriram, and P. Manchanda. The effect of information disclosure on industry payments to physicians. *Available at SSRN 3064769*, 2017.

S. Gupta, D. Hanssens, B. Hardie, W. Kahn, V. Kumar, N. Lin, N. Ravishanker, and S. Sriram. Modeling customer lifetime value. *Journal of Service Research*, 9(2):139–155, 2006.

B. Hajihashemi, A. Sayedi, and J. D. Shulman. The perils of personalized pricing with network effects. *Marketing Science*, 2021. doi: 10.1287/mksc.2021.1323.

B. Hansotia and B. Rukstales. Incremental value modeling. *Journal of Interactive Marketing*, 16(3): 35–46, 2002.

T. Hastie, R. Tibshirani, and J. Friedman. The elements of statistical learnin. *Cited on*, page 33, 2009.

J. R. Hauser, G. L. Urban, G. Liberali, and M. Braun. Website morphing. *Marketing Science*, 28(2): 202–223, 2009.

A. Heiman and E. Muller. Using demonstration to increase new product acceptance: Controlling demonstration time. *Journal of Marketing Research*, pages 422–430, 1996.

I. Hendel and A. Nevo. Measuring the implications of sales and consumer inventory behavior. *Econometrica*, 74(6):1637–1673, 2006.

G. J. Hitsch and S. Misra. Heterogeneous treatment effects and optimal targeting policy evaluation. *Available at SSRN 3111957*, 2018.

T. K. Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

D. G. Horvitz and D. J. Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American statistical Association*, 47(260):663–685, 1952.

V. J. Hotz and R. A. Miller. Conditional choice probabilities and the estimation of dynamic models. *The Review of Economic Studies*, 60(3):497–529, 1993.

V. J. Hotz, R. A. Miller, S. Sanders, and J. Smith. A simulation estimator for dynamic models of discrete choice. *The Review of Economic Studies*, 61(2):265–289, 1994.

IDC. Worldwide semiannual public cloud services tracker. International Data Corporation, 2017. URL `https://www.businesswire.com/news/home/20171106005140/en/Worldwide-Public-Cloud-Services-Revenue-Growth-Remains`.

G. W. Imbens and D. B. Rubin. *Causal inference in statistics, social, and biomedical sciences*. Cambridge University Press, 2015.

INVESP. The state of a/b testing [infographic]. URL `https://www.invespcro.com/blog/the-state-of-ab-testing/`.

R. John. Maximum likelihood estimation of discrete control processes. *SIAM journal on control and optimization*, 26(5):1006–1024, 1988.

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

W. E. Johnson. Probability: The deductive and inductive problems. *Mind*, 41(164):409–423, 1932.

N. Kallus. Recursive partitioning for personalization using observational data. *arXiv preprint arXiv:1608.08925*, 2016.

M. P. Keane and K. I. Wolpin. The solution and estimation of discrete choice dynamic programming models by simulation and interpolation: Monte carlo evidence. *the Review of economics and statistics*, pages 648–672, 1994.

J.-H. Kim. Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. *Computational statistics & data analysis*, 53(11):3735–3745, 2009.

T. Kitagawa and A. Tetenov. Who should be treated? empirical welfare maximization methods for treatment choice. *Econometrica*, 86(2):591–616, 2018.

R. Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

S. R. Künzel, J. S. Sekhon, P. J. Bickel, and B. Yu. Meta-learners for estimating heterogeneous treatment effects using machine learning. *arXiv preprint arXiv:1706.03461*, 2017.

A. Lambrecht and C. Tucker. When does retargeting work? information specificity in online advertising. *Journal of Marketing Research*, 50(5):561–576, 2013.

D. Lefortier, A. Swaminathan, X. Gu, T. Joachims, and M. de Rijke. Large-scale validation of counterfactual learning methods: A test-bed. *arXiv:1612.00367*, 2016.

R. Lewis, J. M. Rao, and D. H. Reiley. Measuring the effects of advertising: The digital frontier. In *Economic Analysis of the Digital Economy*, pages 191–218. University of Chicago Press, 2015.

R. A. Lewis and J. M. Rao. The unfavorable economics of measuring the returns to advertising. *The Quarterly Journal of Economics*, 130(4):1941–1973, 2015.

G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts. Understanding variable importances in forests of randomized trees. In *Advances in neural information processing systems*, pages 431–439, 2013.

C. F. Manski. Statistical treatment rules for heterogeneous populations. *Econometrica*, 72(4): 1221–1246, 2004.

D. M. McCarthy, P. S. Fader, and B. G. Hardie. Valuing subscription-based businesses using publicly disclosed customer data. *Journal of Marketing*, 81(1):17–35, 2017.

D. McKenzie. Should we require balance t-tests of baseline observables in randomized experiments?, 2017. URL `https://blogs.worldbank.org/impactevaluations/should-we-require-balance-t-tests-baseline-observables-randomized-experim`

N. Meinshausen. Quantile regression forests. *Journal of Machine Learning Research*, 7(Jun): 983–999, 2006.

C. F. Mela, S. Gupta, and D. R. Lehmann. The long-term impact of promotion and advertising on consumer brand choice. *Journal of Marketing research*, 34(2):248–261, 1997.

L. Mentch and G. Hooker. Quantifying uncertainty in random forests via confidence intervals and hypothesis tests. *The Journal of Machine Learning Research*, 17(1):841–881, 2016.

P. Milgrom and J. Roberts. Price and Advertising Signals of Product Quality. *Journal of Political Economy*, 94(4):796–821, 1986.

K. Murphy. Machine learning, a probabilistic perspective, 2012.

D. C. Mutz, R. Pemantle, and P. Pham. The perils of balance testing in experimental design: Messy analyses of clean data. *The American Statistician*, 73(1):32–42, 2019.

C. Nadeau and Y. Bengio. Inference for the generalization error. *Machine learning*, 52(3):239–281, 2003.

X. Nie and S. Wager. Quasi-oracle estimation of heterogeneous treatment effects, 2017.

A. Norets. Inference in dynamic discrete choice models with serially orrelated unobserved state variables. *Econometrica*, 77(5):1665–1682, 2009.

A. Norets. Estimation of dynamic discrete choice models using artificial neural network approximations. *Econometric Reviews*, 31(1):84–106, 2012.

K. Pauwels, D. M. Hanssens, and S. Siddarth. The long-term effects of price promotions on category incidence, brand choice, and purchase quantity. *Journal of Marketing Research*, pages 421–439, 2002.

W. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley Series in Probability and Statistics. Wiley, 2007. ISBN 9780470182956. URL `https://books.google.com/books?id=WWWDkd65TdYC`.

R. L. Prentice. Surrogate endpoints in clinical trials: definition and operational criteria. *Statistics in medicine*, 8(4):431–440, 1989.

O. Rafieian. Optimizing user engagement through adaptive ad sequencing. Technical report, Working paper, 2019a.

O. Rafieian. Revenue-optimal dynamic auctions for adaptive ad sequencing. Technical report, Working paper, 2019b.

O. Rafieian and H. Yoganarasimhan. How does variety of previous ads influence consumer's ad response? 2020.

O. Rafieian and H. Yoganarasimhan. Targeting and privacy in mobile advertising. *Marketing Science*, 2021.

P. M. Robinson. Root-n-consistent semiparametric regression. *Econometrica: Journal of the Econometric Society*, pages 931–954, 1988.

P. E. Rossi, R. E. McCulloch, and G. M. Allenby. The value of purchase history data in target marketing. *Marketing Science*, 15(4):321–340, 1996.

D. B. Rubin. Matching to remove bias in observational studies. *Biometrics*, pages 159–183, 1973.

D. B. Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of educational Psychology*, 66(5):688, 1974.

J. Rust. Optimal replacement of gmc bus engines: An empirical model of harold zurcher. *Econometrica: Journal of the Econometric Society*, pages 999–1033, 1987.

J. Rust. Numerical dynamic programming in economics. *Handbook of computational economics*, 1: 619–729, 1996.

J. Rust. Using randomization to break the curse of dimensionality. *Econometrica: Journal of the Econometric Society*, pages 487–516, 1997.

J. Rust. Parametric policy iteration: An efficient algorithm for solving multidimensional dp problems? Technical report, Mimeo, Yale University, 2000.

P. Rzepakowski and S. Jaroszewicz. Decision trees for uplift modeling with single and multiple treatments. *Knowledge and Information Systems*, 32(2):303–327, 2012.

N. S. Sahni, S. C. Wheeler, and P. Chintagunta. Personalization in email marketing: The role of noninformative advertising content. *Marketing Science*, 37(2):236–258, 2018.

N. S. Sahni, S. Narayanan, and K. Kalyanam. An experimental investigation of the effects of retargeted advertising: The role of frequency and timing. *Journal of Marketing Research*, 56(3): 401–418, 2019.

H. M. Sani, C. Lei, and D. Neagu. Computational complexity analysis of decision tree algorithms. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 191–197. Springer, 2018.

E. M. Schwartz, E. T. Bradlow, and P. S. Fader. Customer acquisition via display advertising using multi-armed bandit experiments. *Marketing Science*, 36(4):500–522, 2017.

E. Scornet, G. Biau, J.-P. Vert, et al. Consistency of random forests. *The Annals of Statistics*, 43(4): 1716–1741, 2015.

C. A. Scott. The effects of trial and incentives on repeat purchase behavior. *Journal of Marketing Research*, 13(3):263–269, 1976.

V. Semenova. Machine learning for dynamic discrete choice. *arXiv preprint arXiv:1808.02569*, 2018.

S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965.

D. Simester, A. Timoshenko, and S. I. Zoumpoulis. Targeting prospective customers: Robustness of machine learning methods to typical data challenges. *Management Science*, 2019.

D. Simester, A. Timoshenko, and S. I. Zoumpoulis. Efficiently evaluating targeting policies: Improving on champion vs. challenger experiments. *Management Science*, 66(8):3412–3424, 2020.

C. Smammut and G. I. Webb, editors. *Bellman Equation*, pages 97–97. Springer US, Boston, MA, 2010. ISBN 978-0-387-30164-8. doi: 10.1007/978-0-387-30164-8_71. URL https://doi.org/10.1007/978-0-387-30164-8_71.

J. H. Steckel and W. R. Vanhonacker. Cross-validating regression models in marketing research. *Marketing Science*, 12(4):415–427, 1993.

M. A. Stephens. Edf statistics for goodness of fit and some comparisons. *Journal of the American statistical Association*, 69(347):730–737, 1974.

C. Strobl, J. Malley, and G. Tutz. An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests. *Psychological methods*, 14(4):323, 2009.

C.-L. Su and K. L. Judd. Constrained optimization approaches to estimation of structural models. *Econometrica*, 80(5):2213–2230, 2012.

T. Sunada. Customer learning and revenue-maximizing trial design. Technical report, Working Paper, 2018.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

A. Swaminathan and T. Joachims. Counterfactual risk minimization: Learning from logged bandit feedback. In *International Conference on Machine Learning*, pages 814–823. PMLR, 2015a.

A. Swaminathan and T. Joachims. The self-normalized estimator for counterfactual learning. In *Advances in Neural Information Processing Systems*, pages 3231–3239. Citeseer, 2015b.

A. Swaminathan, A. Krishnamurthy, A. Agarwal, M. Dudik, J. Langford, D. Jose, and I. Zitouni. Off-policy evaluation for slate recommendation. In *Advances in Neural Information Processing Systems*, pages 3632–3642, 2017.

A. Sweeting. Dynamic product positioning in differentiated product markets: The effect of fees for musical performance rights on the commercial radio industry. *Econometrica*, 81(5):1763–1803, 2013.

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

Totango. 2012 saas free trial, freemium and pricing benchmark, 2012. URL `http://www.totango.com/wp-content/uploads/2012/04/2012-SaaS-Free-Trial-Freemium-Pricing-Benchmark.pdf`.

O. Toubia, D. I. Simester, J. R. Hauser, and E. Dahan. Fast polyhedral adaptive conjoint estimation. *Marketing Science*, 22(3):273–303, 2003.

T. J. VanderWeele. Surrogate measures and consistent surrogates. *Biometrics*, 69(3):561–565, 2013.

S. Wager and S. Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 0(0):1–15, 2018. doi: 10.1080/01621459.2017.1319839.

S. Wager, T. Hastie, and B. Efron. Confidence intervals for random forests: the jackknife and the infinitesimal jackknife. *Journal of Machine Learning Research*, 15(1):1625–1651, 2014.

S. Wang and G. F. Özkan-Seely. Signaling product quality through a trial period. *Operations Research*, 66(2):301–312, 2018.

G. I. Webb, M. J. Pazzani, and D. Billsus. Machine learning for user modeling. *User modeling and user-adapted interaction*, 11(1-2):19–29, 2001.

B. L. Welch. The generalization of student's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947.

C. J. Wu. On the convergence properties of the em algorithm. *The Annals of statistics*, pages 95–103, 1983.

J. Yang, D. Eckles, P. Dhillon, and S. Aral. Targeting for long-term outcomes. *arXiv preprint arXiv:2010.15835*, 2020.

J. Yi, Y. Chen, J. Li, S. Sett, and T. W. Yan. Predictive Model Performance: Offline and Online Evaluations. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1294–1302. ACM, 2013.

H. Yoganarasimhan. Search personalization using machine learning. *Management Science*, 66(3): 1045—-1070, 2020.

J. Zhang. The perils of behavior-based personalization. *Marketing Science*, 30(1):170–186, 2011.

Y. Zhao, X. Fang, and D. Simchi-Levi. Uplift modeling with multiple treatments and general response types. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 588–596. SIAM, 2017.

M. Zhu, Y. Yang, and C. K. Hsee. The mere urgency effect. *Journal of Consumer Research*, 45(3): 673–690, 2018.

A. Zimek, E. Schubert, and H.-P. Kriegel. A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 5(5):363–387, 2012.

Appendix A

DESIGN AND EVALUATION OF OPTIMAL FREE TRIALS DETAILS

A.1 Appendix for Data Section

Trial Length	Variable	Mean	Standard Deviation	Min	25%	50%	75%	Max	Number of Observations
7 Days	Subscription	0.154	0.360	0.00	0.00	0.00	0.00	1.00	51,017
	Subscription length (all)	2.35	6.59	0.00	0.00	0.00	0.00	73	50,504
	Subscription length (subscribers)	16.19	8.70	0.0	10.00	18.00	23.00	73.00	7,322
	Revenue (all)	83	295	0	0	0	0	6,219	50,504
	Revenue (subscribers)	578	562	0	239	424	679	6,219	7,322
14 Days	Subscription	0.150	0.357	0.0	0.00	0.00	0.00	1.00	51,040
	Subscription length (all)	2.27	6.45	0.0	0.00	0.00	0.000	67	50,543
	Subscription length (subscribers)	16.09	8.48	0.0	10.00	18.00	22.00	67.00	7,138
	Revenue (all)	81	289	0	0	0	0	6,799	50,543
	Revenue (subscribers)	576	554	0	252	420	679	6,799	7,138
30 Days	Subscription	0.147	0.354	0.00	0.00	0.00	0.00	1.00	235,667
	Subscription length (all)	2.20	6.31	0.00	0.00	0.00	0.00	108	233,176
	Subscription length (subscribers)	15.96	8.36	0.00	10.00	17.00	22.00	108	32,073
	Revenue (all)	77	281	0	0	0	0	20,208	233,176
	Revenue (subscribers)	564	550	0	241	420	660	20,208	32,073

Table A.1: Summary statistics of subscription, subscription length, and revenue outcomes in each trial length. All the revenue numbers are scaled by a constant factor to preserve the firm's anonymity.

A.2 Appendix for ATE and Randomization Checks

A.2.1 ATE using Regressions

A.2.2 Randomization Checks

First, we show the distribution of treatment assignment across the five skill levels in Table A.5. As we can see, the treatment assignment is pretty even; and none of the differences in assignment probabilities within each sub-category are significant. However, as discussed in the main text, this

Trial Length	Variable	Mean	Standard Deviation	Min	25%	50%	75%	Max	Number of Observations
7 Days	Total downloaded packages	1.0	0.0	1.0	1.0	1.0	1.0	4.0	51,017
	Indicator for software use	1.0	0.0	0.0	0.0	1.0	1.0	1.0	51,017
	Number of active days	2.0	2.0	0.0	1.0	1.0	2.0	7.0	45,810
	Ratio of active days	0.0	0.0	0.0	0.0	0.0	0.0	1.0	45,810
	Usage during trial	985	3,567	0.0	37	184	730	223,179	45,810
	Log usage during trial	5.0	3.0	0.0	4.0	5.0	7.0	12.0	45,810
	Dormancy length	1.0	0.0	0.0	0.0	1.0	1.0	1.0	45,810
14 Days	Total downloaded packages	1.0	0.0	1.0	1.0	1.0	1.0	4.0	51,040
	Indicator for software use	1.0	0.0	0.0	0.0	1.0	1.0	1.0	51,040
	Number of active days	2.0	3.0	0.0	1.0	1.0	3.0	14.0	45,902
	Ratio of active days	0.0	0.0	0.0	0.0	0.0	0.0	1.0	45,902
	Usage during trial	1,397	5,938	0.0	44	227	919	400,705	45,902
	Log usage during trial	5.0	3.0	0.0	4.0	5.0	7.0	13.0	45,902
	Dormancy length	1.0	0.0	0.0	0.0	1.0	1.0	1.0	45,902
30 Days	Total downloaded packages	1.0	0.0	1.0	1.0	1.0	1.0	4.0	235,667
	Indicator for software use	1.0	0.0	0.0	1.0	1.0	1.0	1.0	235,667
	Number of active days	3.0	4.0	0.0	1.0	2.0	4.0	30.0	211,802
	Ratio of active days	0.0	0.0	0.0	0.0	0.0	0.0	1.0	211,802
	Usage during trial	1,968	8,007	0.0	51	286	1,227	488,666	211,802
	Log usage during trial	5.0	3.0	0.0	4.0	6.0	7.0	13.0	211,802
	Dormancy length	1.0	0.0	0.0	0.0	1.0	1.0	1.0	211,802

Table A.2: Summary statistics for usage features in each trial length.

Trial Length	Training Data	Test Data	Total
7 days	35,743	15,274	51,017
14 days	35,901	15,139	51,040
30 days	165,056	70,611	235,667
Total	236,700	101,024	337,724

Table A.3: The number of observations for each trial length in each dataset.

	Training Dataset		Test Dataset		All Dataset	
	No control	Control	No control	Control	No control	Control
7 Days	0.0064	0.0064	0.0082	0.0069	0.0069	0.0065
	(0.0021)	(0.0018)	(0.0032)	(0.0027)	(0.0017)	(0.0015)
14 Days	0.0021	0.0018	0.0048	0.0038	0.0029	0.0024
	(0.0021)	(0.0018)	(0.0032)	(0.0027)	(0.0017)	(0.0015)

Table A.4: The coefficients of 7 days and 14 days in regression analysis. In the control case, we control for all the pre-treatment variables.

approach of testing for randomness in assignment within each sub-category is not considered the best practice for a variety of reasons; see Bruhn and McKenzie [2009] and Mutz et al. [2019].

Trial Length	Beginner	Experienced	Intermediate	Mixed	Unknown
7 days	0.6899	0.1295	0.0005	0.1077	0.0723
14 days	0.6887	0.1282	0.0006	0.1082	0.0743
30 days	0.6910	0.1269	0.0006	0.1075	0.0740

Table A.5: The proportion of individuals with a given skill in different trial lengths.

Therefore, we present two more formal randomization checks. First, we regress the treatment variable (W_i) on the pre-treatment variables separately for both the 7 day and 14 day treatment using the 30 day treatment as the base in both regressions. We present these results in Table A.6. The top panel shows the results of the regressions for the training data and the bottom panel shows the results for the test data. In all the four regressions, we see that the pre-treatment variables have no predictive power (all the R-squareds are zero) and F-statistics are not significant. This suggests that the pre-treatment variables are not systematically correlated with treatment assignment. Second, we regress the outcome variable (subscription decision) on the treatment variable and all the pre-treatment variables and present the results in Table A.7. The treatment effects are very similar to those in Table 2.5, which suggests that there are no issues with randomization.

Dataset	Comparison Treatment	R-Squared	Adj. R-squared	F-statistic	p-value of the F-statistic
Training	7 days	0.000377	0.000019	1.052991	0.356526
	14 days	0.000439	0.000081	1.225721	0.093690
Test	7 days	0.000865	0.000026	1.030504	0.406522
	14 days	0.000963	0.000125	1.149222	0.181734

Table A.6: The results of regressing the treatment (W_i) on the pre-treatment variables. In all the regressions, the baseline is the 30 day treatment. The F-statistic tests the null hypothesis that coefficients are jointly statistically significantly different from zero.

Dataset	Trials Length	Coefficient	std err	t-stat	$P > \|t\|$	[0.025	0.975]
Training	14 days	0.0018	0.002	0.999	0.318	-0.002	0.005
	7 days	0.0064	0.002	3.581	0.000	0.003	0.010
Test	14 days	0.0038	0.003	1.385	0.166	-0.002	0.009
	7 days	0.0069	0.003	2.516	0.012	0.002	0.012

Table A.7: Effect of trial length calculated by regressing the subscription decision on all the pre-treatment variables and trial length.

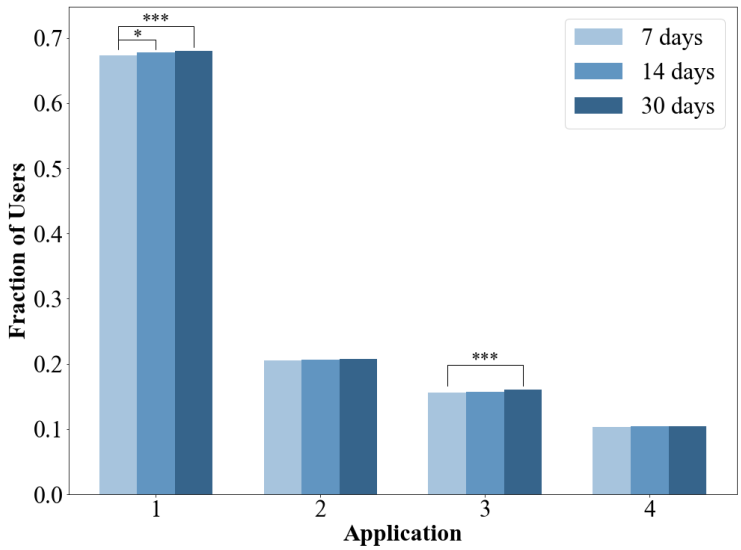


Figure A.1: Fraction of users downloading products 1–4 under each trial length. Products 1 and 3 are complements.

A.3 Appendix for Mechanism

A.4 Lasso Implementation Details

Next, we briefly describe our implementation of the lasso model discussed in §2.5.1. First, note that we use the subscription indicator as outcome variable (Y_i). Next, in our setting, all the pre-treatment variables, X_is, are categorical. So we transform each categorical variable into a set of dummy variables for each sub-category (also referred to as one-hot encoding in the machine learning

literature). After this transformation, we have 82 dummies for the pre-treatment variables, three treatment dummies, and 246 first-order interaction variables. This gives us a total of 331 explanatory variables for the lasso model. Recall that there is one hyper-parameter that we need to tune in the lasso model, which is the L1 regularization parameter, λ. We use the standard cross-validation procedure implemented in the *glmnet* package in R. It searches over 98 different values of λ ranging from 1.8×10^{-5} to 1.5×10^{-1}, and picks the one that gives us the best out-of-sample performance (based on five-fold cross-validation). In our case, it $\lambda = 3.1 \times 10^{-4}$. The final model achieves a MSE of 0.0933 in both the training and test data.

A.5 Design of Alternative Personalized Policies

We discuss the design of a series of alternative personalized policies. First, in §A.5.1, we present with the policies based on other outcome estimators. Next, in §A.5.2, we discuss policies based on CATE estimators.

A.5.1 Policies based on Outcome Estimators

Recall that to design outcome estimators, we first need learn a model $f(x, w) = \mathbb{E}[Y|X_i = x, W_i = w]$. Then, using estimate of the expected outcome, $\hat{f}(x = X_i, w)$, we obtain the optimal personalized policy (based on outcome estimator f) for observation i as:

$$\pi_f(X_i) = w^*, \quad \text{where} \quad w^* = \underset{w \in \mathcal{W}}{\operatorname{argmax}} \hat{f}(x = X_i, w) \tag{A.1}$$

We now consider four commonly used outcome estimators (or models for f): (1) linear regression, (2) CART, (3) random forest, and (4) boosted regression trees. We focus on these because of a few key reasons. First, linear regression is the simplest and most commonly used method to model any outcome of interest. Second, lasso is worth exploring because it was designed to improves on the predictions from a linear regression by reducing the out-of-sample variance using variable selection. Third, CART is a semi-parametric way to model outcomes by partitioning the covariate space into areas with the highest within-group similarity in outcomes. Finally, both random forest and boosted regression trees are improvements of CART and have been shown to offer much higher predictive ability than simpler models such as CART, regression, or lasso. We discuss each of these briefly below.

Linear Regression

A linear regression model with first-order interaction effects can be used to predict an individual i's observed outcome Y_i as a function of her pre-treatment variables, treatment variable, and the interaction of both, as follows:

$$Y_i = X_i\beta_1 + W_i\beta_2 + X_iW_i\beta_3 + \epsilon_i. \tag{A.2}$$

β_1 is a vector that captures the effect of the pre-treatment variables (X_i) on the outcome. β_2 is a vector that captures the main effect of the different treatments on Y. Finally, the vector β_3 captures the interaction effect of treatment and pre-treatment variables. This interaction term is important

because it helps us in personalizing the policy by capturing how the effectiveness of different treatments varies across individuals (as a function of their pre-treatment attributes).

However, in a high dimensional covariate space, linear regressions with first-order interaction effects will have a large number of explanatory variables. This usually leads to poor out-of-sample fit. That is, such regressions tend to have low bias, but high variance – they tend to overfit on the training sample but perform poorly in new samples, especially if the data generating process is noisy. Since our goal is to design optimal policies for counterfactual situations, the out-of-sample fit is the only metric of importance.[1] The lasso model used in the main text addresses the above problem by learning a simpler model that uses fewer variables [Tibshirani, 1996, Friedman et al., 2010].

Tree-based Methods

Next, we discuss tree-based models, starting with CART. CART recursively partitions the covariate space into sub-regions, and the average of Y in a given region ($E(Y)$) is the predicted outcome for all observations in that region [Breiman et al., 1984b]. This type of partitioning can be represented by a tree structure, where each leaf of the tree represents an output region. A general CART model can be expressed as:

$$y = f(x, w) = \sum_{m=1}^{M} \rho_m I(x, w \in R_m), \tag{A.3}$$

where x, w is the set of explanatory variables, R_m is the m^{th} region of the M regions used to partition the space, ρ_m is the predicted value of y in region m.

Trees are trained by specifying a cost function (e.g., mean squared error) that is minimized at each step of the tree-growing process using a greedy algorithm. To avoid over-fitting, usually a penalty term is added to the cost function, whose weight is a hyper-parameter learnt from the data using cross-validation.

While CART has some good properties[2], it often has poor predictive accuracy because it is discontinuous in nature and is sensitive to outliers. Therefore, even with regularization, CART models tend to over-fit on the training data and under-perform on out-of-sample data. We can

[1] We can also add higher-order interaction effects into the regression model. However, we refrain from doing so because it will increase model complexity significantly and exacerbate this problem.

[2] CART can accept both continuous and discrete explanatory variables, is not sensitive to the scale of the variables, and allows any number of interactions between features [Murphy, 2012]. It therefore has ability to capture rich nonlinear patterns in the data. Further, CART can do automatic variable selection, i.e., it uses only those variables that provide better accuracy in the prediction task for splitting.

address these problems using two different techniques – (1) Bagging and (2) Boosting. We discuss both of these techniques and the resulting estimators below.

In general, deep trees have high in-sample fit (low bias), but high out-of-sample variance because of over-fitting. However, we can improve the variance problem by bagging or averaging deep trees using bootstrapping. Ho [1995] formalized this idea and proposed random forests. Random forest usually consists of hundreds or thousands of trees, each of which is trained on a random sub-sample of columns and rows. Each tree is thus different from other trees and the average of these random trees is a better predictor than one single tree trained on the full data.

Another approach is to start with shallow trees. Shallow trees have poor in-sample fit (high bias), but low out-of-sample variance. Additively, adding a series of weak trees that minimize the residual error at each step by a process known as boosting can improve the bias problem, while retaining the low variance. This gives us boosted regression trees. Conceptually, boosting can be thought of as performing gradient descent in function space using shallow trees as the underlying weak learners [Breiman, 1998, Friedman, 2001].[3] In this paper, we use XGBoost, a version of boosted trees proposed by Chen and Guestrin [2016] because it is superior to earlier implementations both in terms of accuracy and scalability.

Please see Appendix §A.6 for the set of hyper-parameters that need to be tuned in CART, random forest, and XGBoost.

A.5.2 Heterogeneous Treatment Effect Estimators

The second approach to designing a personalized policy is to first obtain consistent estimates of heterogeneous treatment effects for each pair of treatments for all users, and then use them to assign treatments. This method also follows a two-step procedure. In the first step, we can obtain consistent estimates of individual-level treatment effects, $\tau_{w_j,w_{j'}}(x)$, for each pair of treatments $w_j, w_{j'} \in \{0, \ldots, W-1\}$. Under the standard assumptions of (1) unconfoundedness, (2) SUTVA, and (3) positivity, $\tau_{w_j,w_{j'}}(x)$, can be defined as [Rubin, 1974, Imbens and Rubin, 2015]:

$$\tau_{w_j,w_{j'}}(x) = \mathbb{E}\Big[Y(X_i, W_i = w_j) - Y(X_i, W_i = w_{j'}) \mid X_i = x\Big]. \tag{A.4}$$

So if we have W treatments, we have to build $\frac{W(W-1)}{2}$ pairwise models, where each model gives us an estimate of individual-level treatment effects for a given pair of treatments, $w_j, w_{j'}$.

[3] It should be noted that bagging is not a modeling technique; it is simply a variance reduction technique. Boosting, however, is a method to infer the underlying model $y = f(x, w)$. Thus, they are conceptually completely different.

In the second step, we use the estimated treatment effects to derive the optimal policy as[4]:

$$\pi^*(X_i) = w_j \quad \text{if and only if} \quad \forall \ j' \neq j \quad \tau_{w_j,w_{j'}}(x = X_i) \geq 0 \tag{A.5}$$

Next, we discuss the These methods are based on the potential outcomes framework and estimate the treatment effect for any pair of treatments $(w_j, w_{j'})$ at each point x as shown in Equation (A.4). However, this equation is not very useful in practical settings (where the covariate space is high-dimensional and data are finite) because we would not have sufficient observations at each X_i to estimate precise treatment effects. Therefore, the general idea behind modern heterogeneous treatment effects estimators is to pool observations that are close in the covariate space and estimate the conditional average treatment effect for sub-populations instead of estimating the treatment effect at each point in the covariate space. That is, we can modify Equation (A.4) as:

$$\tau_{w_j,w_{j'}}(x) = \frac{\sum_{X_i \in l(x), W_i = w_j} Y_i}{\sum 1[X_i \in l(x), W_i = w_j]} - \frac{\sum_{X_i \in l(x), W_i = w_{j'}} Y_i}{\sum 1[X_i \in l(x), W_i = w_{j'}]}, \tag{A.6}$$

where $l(x)$ is the set of covariates that are fairly similar to x. Intuitively, for each point x, we use the observations in $l(x)$ to estimate treatment effects.

The main question in these methods then becomes how to find the optimal $l(x)$ around each x. On the one hand, if $l(x)$ is too small, then we will not have sufficient observations within $l(x)$, which would result in noisy estimates of treatment effects. On the other hand, if $l(x)$ is too large, then we will not capture all the heterogeneity in the treatment effects, which is essential for personalizing policy. Indeed, if $l(x)$ is the entire data, then we simply have one Average Treatment Effect for all users (which will give us one global policy). Thus, finding the optimal $l(x)$ involves the classic bias-variance trade-off.

Starting with Rzepakowski and Jaroszewicz [2012], a growing stream of literature has focused on developing data-driven approaches to finding the optimal $l(x)$ based on ideas from the machine learning literature. Among these methods, the recently developed causal tree and causal forest (or Generalized Random Forest) have been shown to have superior performance. So we focus on them.

---

[4]In practice, we can have situations where three or more pairs of the estimated treatments form a loop, i.e., $\hat{\tau}_{w_j,w_{j'}}(x = X_i) > \hat{\tau}_{w'_j,w_{j''}}(x = X_i) > \hat{\tau}_{w''_j,w_j}(x = X_i)$. This happens if the estimated treatment effects are noisy (usually due to lack of sufficient data). For these observations, we can simply assign the best average treatment (across all observations) as the policy-prescribed treatment. Further, there can be multiple optimal policies because two or more top treatments can have the same effect on the outcome. That is, for some combinations of X_i and $j' \neq j$, we can have: $\tau_{w_j,w_{j'}}(x = X_i) = 0$. However, notice that all the solutions give the same reward, i.e., the optimal reward is unique.

Causal Tree

Causal tree builds on CART. The only difference is that instead of partitioning the space to maximize predictive ability, the objective here is to identify partitions with similar within-partition treatment effect. Athey and Imbens [2016] show that maximizing the variation in the estimated treatment effects (with a regularization term added to control complexity) achieves this objective. Their algorithm consists of two steps. In the first step, it recursively splits the covariate space into partitions. In the second step, it estimates the treatment effect within each partition ($l(x)$) using Equation (A.6). Intuitively, this algorithm pools observations with similar treatment effects into the same partition because splitting observations that have similar treatment effects does not increase the objective function (i.e., variation in the post-split treatment effect).

The main idea behind causal tree is that we can use recursive partitioning to estimate heterogeneous treatment effects if we can come up with an objective function that can identify partitions with similar within-partition treatment effect.[5] Athey and Imbens [2016] show that maximizing the variation in the estimated treatment effects achieves this objective, which can be written as:

$$Var[\hat{\tau}(X)] = \frac{1}{N}\sum_{i=1}^{N}\hat{\tau}^2(X_i) - \left(\frac{1}{N}\sum_{i=1}^{N}\hat{\tau}(X_i)\right)^2 \tag{A.7}$$

Since $\left(\frac{1}{N}\sum_{i=1}^{N}\hat{\tau}(X_i)\right)^2$ remains constant with respect to any possible next split, the objective function can be also described as maximizing the average of the square of estimated treatment effects. In practice, the algorithm chooses the split that maximizes $Var[\hat{\tau}(X)] - \zeta T$, where the second term is added for complexity control and is analogous to the regularization term in CART.

The algorithm consists of two steps. In the first step, it recursively splits the covariate space into partitions. In the second step, it estimates the treatment effect within each partition ($l(x)$) using Equation (A.6). Intuitively, this algorithm pools observations with similar treatment effects into the same partition because splitting observations that have similar treatment effects does not increase the objective function (i.e., variation in the post-split treatment effect).

---

[5]The first example of such a criterion was proposed by Hansotia and Rukstales [2002]: for each potential split, the algorithm calculates the difference between the average outcome of the treatment and control in the right (ΔY_r) and left (ΔY_l) branches of the tree. Then, it selects the split with the highest value of $\Delta\Delta Y = |\Delta Y_r - \Delta Y_l|$. Other splitting criteria include maximizing the difference between the class distributions in the treatment and control groups [Rzepakowski and Jaroszewicz, 2012].

Generalized Random Forest

The causal tree algorithm nevertheless suffers from weaknesses that mirror those of CART. To resolve these issues, Wager and Athey [2018] proposes causal forest, which builds on random forests [Breiman, 2001]. More broadly, Athey et al. [2019b] show that the intuition from random forests can be used to flexibly estimate any heterogeneous quantity, $\theta(x)$, from the data, including heterogeneous treatment effects. They suggest a Generalized Random Forest (GRF) algorithm that learns problem-specific kernels for estimating any quantity of interest at a given point in the covariate space. For treatment estimation, the method proceeds in two steps. In the first step, it builds trees whose objective is to increase the variation in the estimated treatment effects. Each tree is built on a random sub-sample of the data and random sub-sample of covariates. In the second step, the algorithm uses the idea of a weighted kernel regression to calculate the treatment effect at each point x using weights from the first step.

The Generalized Random Forest (GRF) algorithm takes intuition from causal tree and combines it with ideas from predictive random forests to learn problem-specific kernels for estimating any quantity of interest at a given point in the covariate space. For the estimation of heterogeneous treatment effects, the method proceeds in two steps.

In the first step, it builds trees whose objective is to increase the variation in the estimated treatment effects. Each tree is built on a random sub-sample of the data and random sub-sample of covariates. At each step of the partitioning process (when building a tree), the algorithm first estimates the treatment effects in each parent leaf P by minimizing the R-learner objective function. This objective function is motivated by Robinson's method [Robinson, 1988, Nie and Wager, 2017] and can be written as follows:

$$\hat{\tau}_P(\cdot) =_\tau \left[\frac{1}{n_P} \sum_{i=1}^{n_P} \left(\left(Y_i - \hat{m}^{(-i)}(X_i)\right) - (W_i - \hat{e}^{(-i)}(X_i))\hat{\tau}(X_i) \right)^2 + \Lambda_n(\tau(.)) \right], \tag{A.8}$$

where n_P refers to the number of observations in the parent partition, $\Lambda_n(\tau(.))$ is a regularizer that determines the complexity of the model used to estimate $\tau_P(.)$s, and $\hat{m}^{(-i)}(X_i)$ and $\hat{e}^{(-i)}(X_i)$ are out-of-bag estimates for the outcome and propensity score, respectively. While any method can be used for estimating $\hat{m}^{(-i)}(X_i)$ and $\hat{e}^{(-i)}(X_i)$, the GRF implementation uses random forests to estimate these values.

Then, it chooses the next split such that it maximizes the following objective function:

$$\frac{n_L \cdot n_R}{(n_P)^2} \left(\hat{\tau}_L - \hat{\tau}_R\right)^2, \tag{A.9}$$

where n_L and n_R refer to the number of observations in the post-split left and right partitions, respectively. However, instead of calculating the exact values of $\hat{\tau}_L$ and $\hat{\tau}_R$ for each possible split (and for each possible parent), the causal forest algorithm uses a gradient-based approximation of $\hat{\tau}$ for each child node to improve compute speed; see Athey et al. [2019b] for details. Thus, at the end of the first step, the method calculates weights, $\alpha_i(x)$, that denote the frequency with which the i-th training sample falls into the same leaf as x in the first step. Formally, $\alpha_i(x)$ is given by $\alpha_i(x) = \frac{1}{B}\sum_{b=1}^{B} \alpha_{bi}(x)$, where $\alpha_{bi}(x) = \frac{\mathbf{1}(X_i \in l_b(x))}{|l_b(x)|}$, B is the total number of trees built in the first step, and $l_b(x)$ is the partition that x belongs to in the b-th tree.

In the second step, the algorithm uses the idea of a weighted kernel regression to calculate the treatment effect at each point x using weights $\alpha_i(x)$ as follows:

$$\hat{\tau}(x) = \frac{\sum_{i=l}^{N} \alpha_i(x) \left(Y_i - \hat{m}^{(-i)}(X_i)\right) \left(W_i - \hat{e}^{(-i)}(X_i)\right)}{\sum_{i=l}^{N} \alpha_i(x) \left(W_i - \hat{e}^{(-i)}(X_i)\right)^2}, \tag{A.10}$$

As with all supervised learning models, we need to do hyper-parameter optimization to prevent causal forest from overfitting. We refer readers to Appendix §A.6 for details on this.[6]

[6] Athey and Imbens [2016] propose an additional approach to avoid over-fitting in causal tree and causal forest – honest splitting. The main idea behind honesty is to split the data into two parts and use one part for growing each tree (i.e., generating the partitions) and the other part for estimating the treatment effects given a partition. Since the two data-sets are independent of one another, there is a lower likelihood of over-fitting. However, honest splitting comes with its own costs – it reduces the amount of data we have for learning in both stages by half. In settings where the magnitude of the treatment effects is small (such as ours), honest splitting can adversely affect the performance of models. Indeed, we found that models based on honest splitting lead to worse policies compared to models without honest splitting in our setting. So we do not employ honest splitting in our analysis.

A.6 Hyper-parameter Optimization for the Models Estimated

For each alternative model that we use, we describe the hyper-parameters associated with it and the optimal hyper-parameters that we derive after tuning. In all cases, we use five-fold cross-validation to optimize the hyper-parameters. We then train a model on the entire training data using the optimal hyper-parameters, and report the performance of this model on both the training and test data.

- Linear regression does not use any hyper-parameters and hence does not require validation. In this case, we simply train the model on the full training data to infer the model parameters and report the model's performance on both the training and test data.
- For the remaining tree-based models, we directly feed in the 82 dummy variables for the pre-treatment characteristics and the three treatment dummies. Specifically for CART, we use the package *rpart* in R, which implements a single tree proposed by [Breiman et al., 1984b]. We only need to pick the complexity parameter (ζ) using cross validation in this case. We search over 3886 different values for ζ ranging from 8.6×10^{-11} to 1.7×10^{-1}, and derive the optimal complexity parameter as 5.4^{-5}.
- For Random Forest, we use the package *sklearn* in Python. There are three hyper-parameters in this case – (1) n_{tree}, the number of trees over which we build our ensemble forest, (2) $\max_f$, the maximum number of features the algorithm try for any split (it can be either all the features or the squared root of the number of features), and (3) $n_{\min}$, the minimum number of samples required to split an internal node.

  The standard method for finding hyper-parameters is grid-search. However, grid-search is very costly and time-consuming when we have to tune many hyper-parameters. So we use the hyperopt package for tuning the hyper-parameters in this model. Hyperopt provides an automated and fast hyper-parameter optimization procedure that is less sensitive to researcher's choice of searched hyper-parameter values; see Bergstra et al. [2011, 2013] for details.

  For each of these hyper-parameters, we now define the range over which we search as well as the optimal value of the hyper-parameter are shown below:

  - $n_{tree} \in [100, 1200]$ and $n^*_{tree} = 1000$
  - $\max_f \in \{n, sqrt(n)\}$ and $\max^*_f = n$
  - $n_{\min} \in [10, 300]$ and $n^*_{\min} = 70$
- XGBoost also has many hyper-parameters that need tuning. However, we found that our results is sensitive to only three of the parameters: α, η, and $d_{\max}$. The first parameter, α, is an L1 regularization parameter, η is the shrinkage parameter or learning rate, $d_{\max}$ is maximum depth

of trees. Again, we use the hyperopt package to search over a wide range of parameter values. The optimal values are shown below:

- $\alpha \in \{0.1, 0.2, 0.5, 1, 2, 5, 10, 15, 20, 25\}$ and $\alpha^* = 20$
- $\eta \in [0, 1]$ and $\eta^* = 0.59$
- $d_{\max} \in \{6, 8, 10, 12\}$ and $d^*_{\max} = 12$

- Causal tree has two hyper-parameters that needs tuning – (1) the complexity parameter (ζ) and the minimum number of treatment and control observations in each leaf (q). We use the cross-validation procedure in the "causalTree" package in R for tuning ζ. We manually tune q using grid-search over the range $[100, 1000]$ in increments of 100. We search over all possible values of ζ for each q.

  The optimal hyper-parameters for the three trees (one for each pair of treatments) are:

  - The tree for 7 and 14 days pair: $\zeta = 1.8e-05$ and $q = 100$.
  - The tree for 7 and 30 days pair: $\zeta = 8.0e-06$ and $q = 100$.
  - The tree for 14 and 30 days pair: $\zeta = 3.0e-06$ and $q = 900$.

- Causal forest has five hyper-parameters that need to be tuned[7]: (i) $frac$, the fraction of data that is used fr training each tree, (ii) $mtry$, the number of variables tried for each split, (iii) max_imb the maximum allowed imbalance of a split, (iv) imb_pen, a penalty term for imbalanced splits, (v) q, the minimum number of observations per condition (control, treatment) in each partition.[8] We used the hyper-parameter optimization procedure available in the grf package for tuning these hyper-parameters. The optimal hyper-parameters for each model are shown below:

  - 7-14 days pair: $frac = 0.5$, $max\_imb = 0.11$, $imb\_pen = 2.03$, $mtry = 13$, and $q = 1651$.
  - 7-30 days pair: $frac = 0.5$, $max\_imb = 0.13$, $imb\_pen = 2.49$, $mtry = 1$, and $q = 4$.
  - 14-30 days pair: $frac = 0.5$, $max\_imb = 0.20$, $imb\_pen = 5.11$, $mtry = 7$, and $q = 121$.

[7]For more information please visit https://github.com/grf-labs/grf/blob/master/REFERENCE.md

[8]The GRF algorithm estimates outcomes and propensities in the first step. However, in our setting, we do not need to estimate propensity scores since they are known and constant for each observation since treatment assignment is fully randomized. Our qualitative results remain the same if we instead estimate them from the data.

A.7 Appendix for Empirical Results on Alternative Personalized Policies

Policy-prescribed Treatment	π_7	π_{14}	π_{30}	π_{reg}	π_{lasso}	π_{cart}	π_{r_forest}	$\pi_{xgboost}$	π_{c_tree}	π_{c_forest}
7 Days	1	0	0	0.521	0.689	1	0.444	0.697	1	0.911
14 Days	0	1	0	0.322	0.232	0	0.269	0.185	0	0.089
30 Days	0	0	1	0.157	0.079	0	0.287	0.118	0	0
Total	1	1	1	1	1	1	1	1	1	1

Table A.8: Fraction of users assigned the 7-, 14-, and 30-day trials under counterfactual policies (in test data).

First, in Table A.8, we show the distribution of the three treatments (7, 14, and 30 days) varies under the alternative policies.

Method	**Mean Squared Error**	
	Training Set	Test Set
Linear Regression	0.0932	0.0933
Lasso	0.0933	0.0933
CART	0.0916	0.0920
Random Forest	0.0904	0.0915
XGBoost	0.0905	0.0911

Table A.9: Comparison of the predictive performance of the five outcome estimation methods. The MSE for any method are calculated as $\frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{N}$, where $\hat{y}_i$ is the prediction of y_i and N is the number of data-points in the data-set being considered.

Next, in Table A.9, we present the MSE for the five outcome estimators on both training and test data. We find that linear regression and lasso are the worst in terms of model-fit, with XGBoost performing the best. This finding is consistent with earlier papers that have found XGBoost to be the best outcome prediction method in tasks involving prediction of human behavior [Rafieian and Yoganarasimhan, 2020, Rafieian, 2019a]. However, the tree-based models – CART, Random Forest, and XGBoost – suffer from over-fitting in spite of hyper-parameter tuning. That is, their performance on the test data is worse than that on training data. For the heterogeneous treatment effects estimators, we cannot compare the estimates of treatment effects with any ground truth since we (as researchers/managers) do not know the true treatment effects.

Finally, in Figure A.2, we show the CDFs of the estimated CATEs for the 7 vs. 14 day and 14 vs. 30 treatments for all the methods.

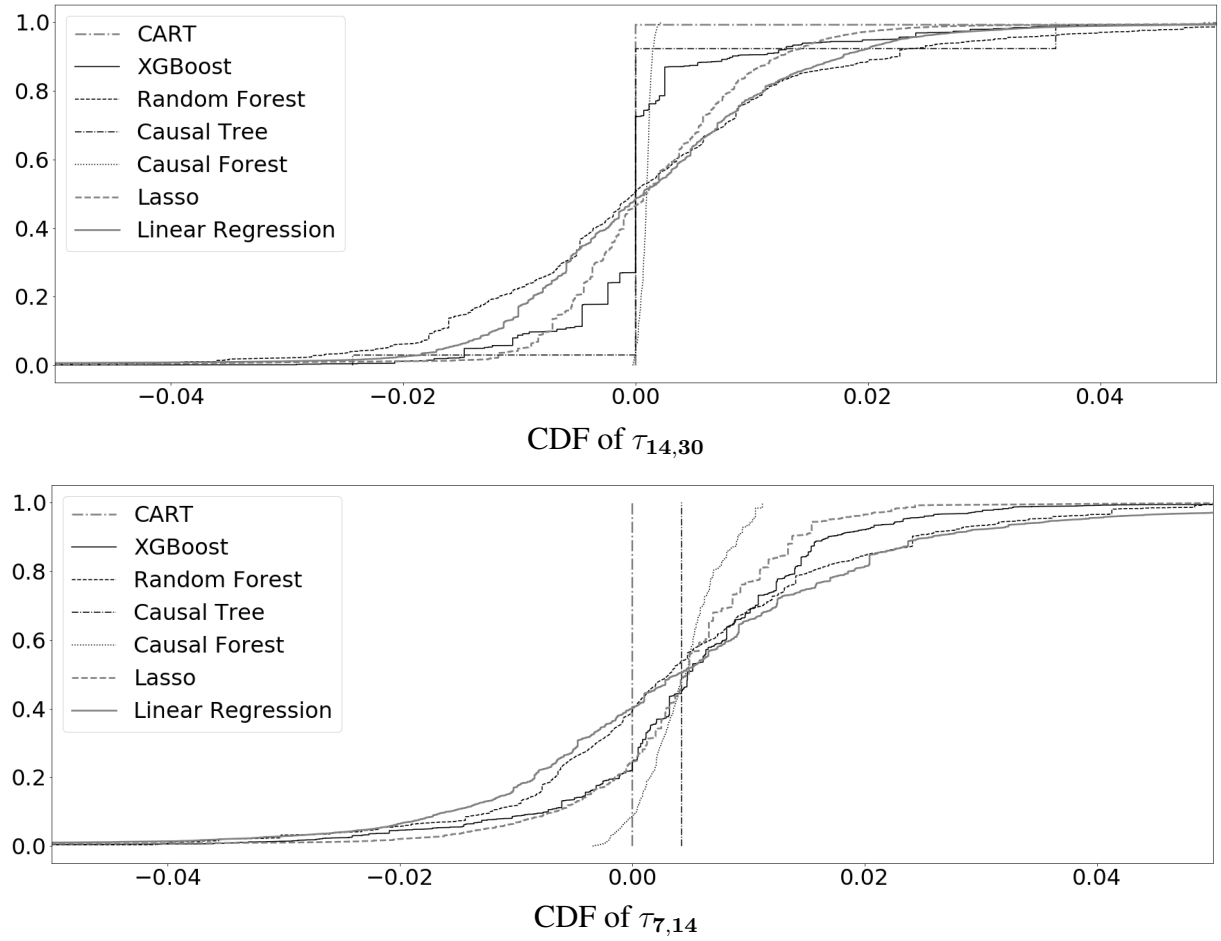


Figure A.2: The CDF of estimated CATEs for 7 vs 14 days, and 14 vs 30 days of free trial from using different methods (for test data).

A.8 Appendix for §2.6

This section is organized as follows. First, in §A.8.1, we quantify the heterogeneity in the effect of trial length on post-treatment usage. Next, in §A.8.2, we quantify the heterogeneity in the effect of usage on subscription. Then, in §A.8.3, we show how these heterogeneous responses are consistent with the user's pre-treatment demographic variables. Finally, in §A.8.4, we provide additional evidence to rule out the demand cannibalization hypothesis.

A.8.1 Heterogeneity in Post-treatment Usage Across Segments

Outcome variable	Intercept	14-days optimal segment	30-days optimal segment	R^2	N
Total downloaded products	1.135 (0.001)	0.065 (0.002)	0.002 (0.003)	0.005	337,724
Indicator for software use	0.833 (0.001)	0.015 (0.002)	-0.03 (0.003)	0.001	303,514
Number of active days	2.75 (0.008)	1.116 (0.017)	0.445 (0.027)	0.014	303,514
Log usage during trial	4.992 (0.006)	0.43 (0.012)	-0.034 (0.019)	0.004	303,514
Dormancy length	17.465 (0.024)	-2.304 (0.049)	-0.953 (0.077)	0.007	303,514

Table A.10: Regressions of different usage variables on the users' optimal trial length. Each row denotes a separate regression, with the first column showing the outcome variable of that regression. Standard errors in parentheses.

Variable	**Number of active days**	**Log usage during trial**	**Dormancy length**
Intercept	1.676 (0.022)	4.732 (0.015)	4.732 (0.051)
14-days trial	0.519 (0.03)	0.148 (0.022)	5.343 (0.072)
30-days trial	1.427 (0.024)	0.341 (0.017)	17.09 (0.056)
14-days optimal segment	0.297 (0.044)	0.222 (0.031)	-0.422 (0.103)
14-days optimal segment · 14-days trial	0.41 (0.062)	0.175 (0.044)	-0.791 (0.145)
14-days optimal segment · 30-days trial	1.086 (0.048)	0.261 (0.034)	-2.519 (0.114)
30-days optimal segment	0.05 (0.068)	-0.158 (0.049)	-0.079 (0.162)
30-days optimal segment · 14-days trial	0.176 (0.097)	0.118 (0.068)	-0.394 (0.228)
30-days optimal segment · 30 days trial	0.523 (0.075)	0.15 (0.053)	-1.21 (0.178)
R-squared	0.044	0.008	0.346
Number of observations	303,514	303,514	303,514

Table A.11: Regression of different usage features on the interaction of trial length, and optimal trial length. Standard errors in parentheses.

We start with Table A.10, which shows how the three segments differ in their usage behavior

based on five different regressions. Each regression in this table uses a specific usage feature as the outcome variable and the user's segment (or optimal treatment) as the explanatory variable. The 7-day optimal segment is the baseline in all the regressions (and is denoted by the intercept).[9] We find that the 14-day optimal users download and use the product more compared to the 7-day optimal users (positive coefficients in the regressions using log usage and number of active days) and they tend to remain active longer than both the 7- and 30-day optimal users (negative coefficient of dormancy length). In contrast, the 30-day optimal users do not use the product significantly more than the 7-day optimal users, but they have somewhat shorter dormancy periods. Overall, this suggests that 7-day optimal users use the product the least and become inactive the soonest, while the 14-day optimal use it the most and are active for the longest.

Next, we examine how these three segments behave under different trial lengths. In Table A.11, we present the results from three regressions, where the outcome variable in each regression is a usage feature (Number of active days, Log usage, or Dormancy length) and the explanatory variables are the user's optimal treatment, the actual treatment assigned to her, and the interaction effects. In all these regressions, the intercept refers to the baseline of 7-day optimal users when they are assigned the 7-day trial. We find that, compared to the 7-day optimal users, the 14-day optimal users use the product more and have shorter period of inactivity at the end, as their trial length increases. Next, consider the 30-day optimal users: these users use the product less when they get 7 days (compared to the 7-day optimal users). Further, when these users get 30 days, they use it more and their dormancy period is shorter. However, when these users are given 14 days, we don't see any significant improvement in their usage or reduction in their dormancy period. Thus, these users really need the longer 30-day trial to register higher usage and have shorter periods of dormancy.

[9] We do not control for the actual treatment assignment here since it is randomly assigned, and therefore orthogonal to a user's optimal treatment.

A.8.2 Heterogeneity in the Effect of usage on Subscription

So far, we have shown how the three segments differ in their usage behavior as a function of trial length; i.e., we have focused on the left side of Figure 2.3. Now we examine the heterogeneity in the effect of usage on subscription across the three segments, i.e., we focus on the right hand side of Figure 2.3. Table A.12 presents the results from regressing the user's subscription outcome on her optimal treatment/segment and her usage features, and the interactions of these two sets of variables. There are two key findings here. First, For 7- and 30-day optimal users, the effect of usage on subscription is similar. However, notice that for the 14-day optimal segment, the effect of usage on subscription is much smaller (the interaction between 14-days optimal segment and log usage is negative). This implies that while longer trials have a positive impact on usage for 14-day optimal segment, more usage doesn't lead to higher conversions in this segment. Second, for 7-days optimal users, dormancy length has a large negative effect on subscription. For the 14-days optimal users, the negative effect of dormancy length is still there, but is somewhat smaller. For the 30 day people, the negative effect is the smallest.

	coef	std err	z	$P > \|z\|$	[0.025	0.975]
Intercept	-2.3522	0.140	-16.838	0.000	-2.626	-2.078
14-days trial	0.0301	0.023	1.297	0.195	-0.015	0.076
30-days trial	0.1780	0.025	7.205	0.000	0.130	0.226
14-days optimal segment	0.2007	0.063	3.182	0.001	0.077	0.324
30-days optimal segment	0.0237	0.094	0.252	0.801	-0.161	0.208
Number of active days	0.0497	0.003	16.984	0.000	0.044	0.055
Number of active days · 14-days optimal segment	-0.0126	0.004	-3.030	0.002	-0.021	-0.004
Number of active days · 30-days optimal segment	0.0001	0.007	0.019	0.985	-0.013	0.013
Log usage during trial	0.0712	0.007	9.723	0.000	0.057	0.086
Log usage during trial · 14-days optimal segment	-0.0303	0.012	-2.572	0.010	-0.053	-0.007
Log usage during trial · 30-days optimal segment	0.0016	0.019	0.086	0.931	-0.035	0.038
Dormancy length	-0.0317	0.001	-27.513	0.000	-0.034	-0.029
Dormancy length · 14-days optimal segment	0.0039	0.002	2.590	0.010	0.001	0.007
Dormancy length · 30-days optimal segment	0.0102	0.002	4.344	0.000	0.006	0.015
Indicator for software use	-0.5910	0.047	-12.463	0.000	-0.684	-0.498
Indicator for software use · 14-days optimal segment	0.2351	0.076	3.075	0.002	0.085	0.385
Indicator for software use · 30-days optimal segment	0.0076	0.118	0.064	0.949	-0.224	0.240
Total downloaded products	0.6077	0.017	35.734	0.000	0.574	0.641
Total downloaded products · 14-days optimal segment	-0.1124	0.027	-4.098	0.000	-0.166	-0.059
Total downloaded products · 30-days optimal segment	-0.0640	0.045	-1.422	0.155	-0.152	0.024

Table A.12: Regressing subscription on usage features interacted with optimal trial length. We also control for pre-treatment variables. The number of observations is $303,514$, and the pseudo R-squared is 0.292.

A.8.3 Pre- and Post-Treatment Attributes of the Segments

We now show the distributions of the pre-treatment demographics and post-treatment subscription outcomes for the three segments in Tables A.13 and A.14.

Variable	Value	Population	Policy Assigned Treatment		
			30 Days	14 Days	7 Days
Country	*United States*	54.9%	55.9%	45.8%	57.9%
	Germany	8.9%	8.2%	16.0%	6.6%
	Japan	7.9%	9.7%	11.5%	6.4%
	Other	28.3%	26.2%	26.8%	29.1%
	Total	100%	100%	100%	100%
Operating System	*Windows 10*	29.0%	9.0%	11.9%	37.0%
	Windows 7	21.5%	46.8%	21.7%	18.5%
	Windows 8.1	14.1%	0.9%	22.6%	12.7%
	El Capitan	13.9%	30.8%	18.0%	10.6%
	Yosemite	13.4%	3.3%	22.0%	11.7%
	Other	8.2%	9.2%	4.0%	9.4%
	Total	100%	100%	100%	100%
Skill	*Beginner*	68.9%	33.8%	41.1%	82.3%
	Experienced	12.8%	41.6%	22.0%	6.4%
	Mixed	10.7%	2.0%	35.0%	3.6%
	Unknown	7.5%	21.8%	1.9%	7.7%
	Intermediate	0.1%	0.7%	0.0%	0.0%
	Total	100%	100%	100%	100%
Job	*Student*	28.1%	12.7%	13.1%	34.9%
	Unknown	22.0%	69.8%	16.2%	18.5%
	Hobbyist	20.0%	7.3%	20.0%	21.5%
	Other	29.8%	10.3%	50.7%	25.1%
	Total	100%	100%	100%	100%
Signup Channel	*Website*	81.6%	65.2%	76.1%	85.3%
	App Manager	8.2%	17.7%	5.3%	8.1%
	Other	10.2%	17.1%	18.7%	6.6%
	Total	100%	100%	100%	100%

Table A.13: Distribution of users' pre-treatment attributes for the three segments: those assigned to the 7-day condition, those assigned to the 14-day condition, and those assigned to the 30-day condition. (For each categorical variable, we show the fractions only for the top few categories in the interest of space.)

Variable	Population	Policy Assigned Treatment		
		30 Days	14 Days	7 Days
	Mean			
Subscription rate	14.8%	17.0%	29.1%	9.8%
Revenue	$536	$545	$546	$524
Retention (months)	16.1	17.3	16.0	16.1
	Fraction			
Purchased Bundle				
Bundle I	55.9%	57.5%	55.4%	55.9%
All inclusive	21.5%	20.3%	21.6%	21.7%
Single Product	19.2%	16.7%	19.2%	19.8%
Other	3.4%	5.5%	3.8%	2.6%
Total	100%	100%	100%	100%
Subscription Type				
Commercial	79.1%	79.0%	81.0%	77.3%
Education	20.7%	20.6%	18.9%	22.7%
Other	0.1%	0.4%	0.1%	0.1%
Total	100%	100%	100%	100%

Table A.14: Means and distributions of users' post-treatment attributes for the three segments: users assigned to the 7-day condition, users assigned to the 14-day condition, and users assigned to the 30-day condition.

A.8.4 Additional Evidence to Rule Out Demand Cannibalization

We now provide additional evidence to rule out the demand cannibalization hypothesis. In Figure A.3, we show that beginners who use the product more when assigned to the 14– and 30-day condition are more likely to subscribe when they use the product more. This rules out the possibility that these users are less likely to subscribe when assigned to the 14 and 30-day condition because they have already obtained their use for the product. Similarly, Table A.15 shows that beginners who use the product more are more likely to subscribe, even after controlling for all the other user-specific variables.

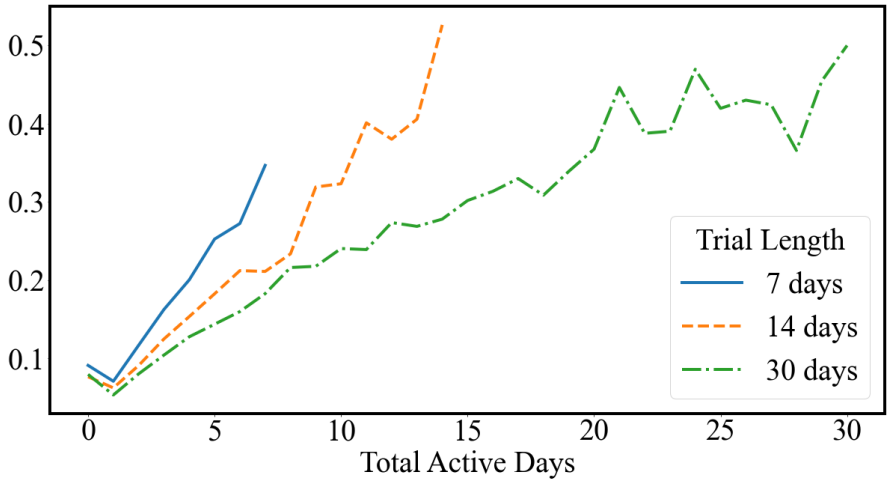


Figure A.3: Subscription probability of *beginner* users with different total active days. The subscription probability increases as the total active days increases.

	coef	std err	z	$P > \|z\|$	[0.025	0.975]
Indicator for using the software	-0.5764	0.035	-16.432	0.000	-0.645	-0.508
Total downloaded packages	0.5714	0.012	46.194	0.000	0.547	0.596
Number of active days	0.0477	0.002	21.734	0.000	0.043	0.052
Log usage during trial	0.0728	0.005	13.688	0.000	0.062	0.083
Dormancy length	-0.0322	0.001	-33.883	0.000	-0.034	-0.030

Table A.15: Regression of subscription on usage features and trial length, with all the pre-treatment variables included as controls (not shown in the table above) for beginner users.

A.9 Appendix for §2.7.2

Trial Length	Subscription Length						
	Mean	**Std**	**Min**	**25%**	**50%**	**75%**	**Max**
7 Days	15.13	9.31	0	8	16	22	73
14 Days	15.04	9.11	0	8	16	22	67
30 Days	14.81	9.05	0	8	16	22	108

Table A.16: The summary statistics of the subscribed users' total months of subscription.

Trial	Subscribed Bundle					Subscription Type		
Length	**1**	**All Inclusive**	**Single Product**	**4**	**5**	**Commercial**	**Education**	**Government**
7 Days	55.24	22.02	19.44	1.68	1.62	78.57	21.30	0.13
14 Days	55.99	21.76	19.12	1.79	1.34	79.62	20.25	0.13
30 Days	55.98	21.64	18.95	1.82	1.62	79.02	20.85	0.13

Table A.17: The fraction of each subscription bundle and type. We do not reveal the names of some of the bundles to preserve's the firm's anonymity.

A.10 Appendix for §2.7.3

Personalized policy based on	7 Days	14 Days	30 Days	Total
Subscription	68.87	23.28	7.85	100
Subscription Length	84.17	15.25	0.58	100
Revenue	63.08	33.50	3.42	100

Table A.18: The percentage of users allocated to each trial length in the three policies based on: (1) subscription, (2) subscription length, and (3) revenue.

Table A.18 presents the proportion of users assigned to each trial length under the three different policies (optimized on the three different outcome variables). We see two interesting patterns here. First, when we personalize the policy to optimize subscription length, the policy has a tendency to assign shorter free trials more often. This is because users who get shorter trials are likely to subscribe sooner. The average number of days to subscription, from the start of the free-trial, is 121, 129, and 144 days for users who receive 7-, 14-, and 30- days trials, respectively. Further, we see that shorter trials lead to higher same-day subscriptions. 2.5% of users who received the 7-days trial and subscribed in the first day, whereas this number for the 14- and 30-days trials is 2% and 1.9%, respectively. Therefore, the subscription length is higher for users who received shorter trial lengths (see Table A.16 in Appendix A.9). Hence, when a policy that optimizes subscription length will emphasize shorter trials. Next, we see that the policy designed to optimize revenues allocates a significantly larger proportion of users to the 14-days trial. This is because when we give 14 day trials, a slightly larger fraction of users subscribe to commercial licenses (Table A.17 in Appendix A.9). Commercial licenses are significantly more expensive; so a revenue-optimizing policy tends to assign 14 days to a larger fraction of users in order increase the likelihood of commercial subscriptions. In sum, we see that the proportion of users assigned to different trial lengths can vary depending on the outcome being optimized.

Appendix B

RECURSIVE PARTITIONING ALGORITHM CONVERGENCE AND SIMULATION DETAILS

B.1 The Proof for Likelihood Increase

In this section, we prove that the likelihood function $\mathcal{L}(\theta^*, \mathbf{\Pi})$ increases at each iteration of the recursive partitioning. Formally, we prove

$$\mathcal{L}(\theta_t^*, \mathbf{\Pi}_t) \leq \mathcal{L}(\theta_{t+1}^*, \mathbf{\Pi}_{t+1}) \tag{B.1}$$

where $\mathbf{\Pi}_t$ and $\mathbf{\Pi}_{t+1}$ are the discretizations generated by the proposed recursive partitioning in steps t and $t+1$ respectively. In addition, $\theta_t^* = \text{argmax}_\theta \, \mathcal{L}(\theta, \mathbf{\Pi}_t)$, and $\theta_{t+1}^* = \text{argmax}_\theta \, \mathcal{L}(\theta, \mathbf{\Pi}_{t+1})$. We assume a fully non-parametric form for the utility and state transition function to avoid having a parametric-form dependent proof[1], i.e., $\bar{u}(x, \pi, j; \theta, \mathbf{\Pi})$ and $g(x, \pi | x', \pi', j; \mathbf{\Pi})$ are constant coefficients for each combination of states and decisions. Thus $\theta = C$, where C is 3 dimensional vector, and $C_{x\pi}^d$ is the utility from decision d in observable state $\{x, \pi\}$.

First we need to define few terms that are necessary for estimation of DDC models. Let the value function and choice specific value function be as follows

$$\bar{V}(x, \pi; \theta, \mathbf{\Pi}) = \log \sum_{j \in \mathbf{J}} \exp v(x, \pi, j; \theta, \mathbf{\Pi}) \tag{B.2}$$

$$v(x, \pi, j; \theta, \mathbf{\Pi}) = \bar{u}(x, \pi, j; \theta, \mathbf{\Pi}) + \beta \sum_{x' \in \mathbf{X}} \sum_{\pi' \in \mathbf{\Pi}} \bar{V}(x', \pi'; \theta, \mathbf{\Pi}) g(x', \pi' | x, \pi, j; \theta, \mathbf{\Pi}) \tag{B.3}$$

where β is the discounting factor usually set by the researcher. We assume that error terms are drawn from Type 1 Extreme Value distribution. Consequently, the predicted choice probabilities

[1] One can argue that a fully non-parametric form can be regarded as a parameterization form itself. However, we believe it is the most flexible form, and any functional form can be drawn from its results. In addition, researchers usually calculate the state transition function non-parametricly.

can be calculated as the following

$$\hat{p}(j|x,\pi;\theta,\mathbf{\Pi}) = \frac{\exp v(x,\pi,j;\theta,\mathbf{\Pi})}{\exp \bar{V}(x,\pi;\theta,\mathbf{\Pi})} \tag{B.4}$$

Finally, we split the likelihood function into two parts, the decision part and the likelihood part denoted by $\mathcal{L}_{dc}(\theta,\mathbf{\Pi})$ and $\mathcal{L}_{st}(\mathbf{\Pi})$ respectively, such that $\mathcal{L}(\theta,\mathbf{\Pi}) = \mathcal{L}_{dc}(\theta,\mathbf{\Pi}) + \lambda\,\mathcal{L}_{st}(\mathbf{\Pi})$. Note that the decision part of the likelihood is a function of the utility function parameters, θ. We prove that each part of the likelihood increases for any additional split to a given partitioning $\mathbf{\Pi}$, which is more general than what we intended in this section.

Theorem 1. For any candidate additional split, noted by $\{k,q,z\}$, to a discretization $\mathbf{\Pi}$, where k is a partition in $\mathbf{\Pi}$, q is a feature in Q, and z is a value within the range of possible values for q in k, the following inequalities hold

$$\exists\theta' : \mathcal{L}_{dc}(\theta^*,\mathbf{\Pi}) \leq \mathcal{L}_{dc}(\theta',\mathbf{\Pi}') \tag{B.5}$$

$$\mathcal{L}_{st}(\mathbf{\Pi}) \leq \mathcal{L}_{st}(\mathbf{\Pi}') \tag{B.6}$$

where $\mathbf{\Pi}' = \mathbf{\Pi} + \{k,q,z\}$ is the discretization after adding the candidate split, and $\theta^* = \operatorname{argmax}_\theta \mathcal{L}(\theta,\mathbf{\Pi})$ is the optimal parameters given discretization $\mathbf{\Pi}$.

We prove inequalities B.5 and B.6 separately. However, we first need to prove some lemmas in order to proceed.

Lemma 2. For any given vector $a : \sum_i a_i = 1$, the answer to the following maximization problem is equal to a.

$$\begin{aligned} \max_b \quad & f(b) = \sum_i a_i \ln b_i \\ \text{s.t.} \quad & \sum_i b_i = 1 \end{aligned}$$

Proof. It is a constrained optimization problem that can be solved by maximizing the Lagrangian

function.

$$\begin{aligned}
\mathcal{L}(a, b, \lambda) &= \sum_i a_i \ln b_i - \lambda(\sum_i b_i - 1) \\
\nabla\mathcal{L}(b, \lambda) &= 0 \\
\frac{\partial\mathcal{L}}{\partial b_i} &= \frac{a_i}{b_i} - \lambda = 0 \Rightarrow a_i = \lambda b_i \Rightarrow \sum a_i = \lambda \sum b_i \\
\frac{\partial\mathcal{L}}{\partial\lambda} &= \sum_i b_i - 1 = 0 \Rightarrow \sum_i b_i = 1 \\
&\Rightarrow \lambda = 1 \Rightarrow b_i = a_i
\end{aligned}$$

□

Lemma 3. There is a flow utility coefficient set θ' such that for any $x \in \mathbf{X}$, $q \in \mathbf{Q}$ and $j \in \mathbf{J}$, the following equation holds

$$\begin{aligned}
\exp v(x, \mathbf{\Pi}'(q), j; \theta', \mathbf{\Pi}') &= \exp v(x, \mathbf{\Pi}(q), j; \theta^*, \mathbf{\Pi}) \\
&+ \Big[\Pr(j|x, \mathbf{\Pi}'(q)) - \Pr(j|x, \mathbf{\Pi}(q))\Big] \exp \bar{V}(x, \mathbf{\Pi}(q); \theta^*, \mathbf{\Pi})
\end{aligned} \tag{B.7}$$

Proof. Using the equality $\log(a+b) = \log(a) + \log(1+b/a)$, we write equation B.7 as the following

$$\begin{aligned}
v(x, \mathbf{\Pi}'(q), j; \theta', \mathbf{\Pi}') &= v(x, \mathbf{\Pi}(q), j; \theta^*, \mathbf{\Pi}) + \log\Big(1 + \frac{\Pr(j|x, \mathbf{\Pi}'(q)) - \Pr(j|x, \mathbf{\Pi}(q))}{\frac{\exp v(x, \Pi(q), j; \theta^*, \mathbf{\Pi})}{\exp \bar{V}(x, \Pi(q); \theta^*, \mathbf{\Pi})}}\Big) \\
&= v(x, \mathbf{\Pi}(q), j; \theta^*, \mathbf{\Pi}) + \log\Big(1 + \frac{\Pr(j|x, \mathbf{\Pi}'(q)) - \Pr(j|x, \mathbf{\Pi}(q))}{\hat{p}(j|x, \mathbf{\Pi}(q); \theta^*, \mathbf{\Pi})}\Big)
\end{aligned} \tag{B.8}$$

Using the formulation of the choice specific value function (equation B.3), we can write equation B.8 in term of the flow utility and expected value functions as follow

$$\begin{aligned}
u(x, \mathbf{\Pi}'(q), j; \theta', \mathbf{\Pi}') &= u(x, \mathbf{\Pi}(q), j; \theta^*, \mathbf{\Pi}) \\
&+ \sum_{x' \in X} \sum_{\pi \in \mathbf{\Pi}'} \bar{V}(x', \pi; \theta^*, \mathbf{\Pi}) g(x', \pi | x, \mathbf{\Pi}(q), j) \\
&- \sum_{x' \in X} \sum_{\pi \in \mathbf{\Pi}'} \bar{V}(x', \pi; \theta', \mathbf{\Pi}') g(x', \pi | x, \mathbf{\Pi}'(q), j) \\
&+ \log\Big(1 + \frac{\Pr(j|x, \mathbf{\Pi}'(q)) - \Pr(j|x, \mathbf{\Pi}(q))}{\hat{p}(j|x, \mathbf{\Pi}(q); \theta^*, \mathbf{\Pi})}\Big)
\end{aligned} \tag{B.9}$$

We assume a non-parametric functional form for the utility function; therefore, we can calculate a set of new utility values for each observable state $\{x, \pi\}$ based on the above equation that satisfies equation B.7. However, the right-hand side of equation B.9 uses θ', which we are trying to calculate. If we want to use equation B.9, we need to prove that this equation has a unique answer. Alternatively, we prove that if equation B.7 holds, the value functions in the new partitioning and old partitioning are equal.

$$
\begin{aligned}
&\bar{V}(x, \mathbf{\Pi}(q); \theta', \mathbf{\Pi}') \\
&\quad = \log \sum_{j \in \mathbf{J}} \exp v(x, \mathbf{\Pi}(q), j; \theta', \mathbf{\Pi}') \\
&\quad = \log \sum_{j \in \mathbf{J}} \Big(\exp v(x, \mathbf{\Pi}(q), j; \theta^*, \mathbf{\Pi}) + \big[\Pr(j|x, \mathbf{\Pi}'(q)) - \Pr(j|x, \mathbf{\Pi}(q)) \big] \bar{V}(x, \mathbf{\Pi}(q); \theta^*, \mathbf{\Pi}) \Big) \\
&\quad = \log \Big(\sum_{j \in \mathbf{J}} \exp v(x, \mathbf{\Pi}(q), j; \theta^*, \mathbf{\Pi}) + \bar{V}(x, \mathbf{\Pi}(q); \theta^*, \mathbf{\Pi}) \sum_{j \in \mathbf{J}} \big[\Pr(j|x, \mathbf{\Pi}'(q)) - \Pr(j|x, \mathbf{\Pi}(q)) \big] \Big) \\
&\quad = \log \sum_{j \in \mathbf{J}} \exp v(x, \mathbf{\Pi}(q), j; \theta^*, \mathbf{\Pi}) \\
&\quad = \bar{V}(x, \mathbf{\Pi}(q); \theta^*, \mathbf{\Pi})
\end{aligned}
\tag{B.10}
$$

where the equality from line 3^{rd} to line 4^{th} comes from $\sum_{j \in \mathbf{J}} \big[\Pr(j|x, \mathbf{\Pi}'(q)) - \Pr(j|x, \mathbf{\Pi}(q)) \big] = 0$.

We can now write equation B.9 as the following:

$$
\begin{aligned}
u(x, \mathbf{\Pi}'(q), j; \theta', \mathbf{\Pi}') = u(x, \mathbf{\Pi}(q), j; \theta^*, \mathbf{\Pi}) \\
+ \sum_{x' \in X} \sum_{\pi \in \mathbf{\Pi}'} \bar{V}(x', \pi; \theta^*, \mathbf{\Pi}) \big(g(x', \pi|x, \mathbf{\Pi}(q), j) - g(x', \pi|x, \mathbf{\Pi}'(q), j) \big) \\
+ \log \Big(1 + \frac{\Pr(j|x, \mathbf{\Pi}'(q)) - \Pr(j|x, \mathbf{\Pi}(q))}{\hat{p}(j|x, \mathbf{\Pi}(q); \theta^*, \mathbf{\Pi})} \Big)
\end{aligned}
\tag{B.11}
$$

This equation is not dependent on θ', and can be used to calculate a set of utility function values (θ') for partitioning $\mathbf{\Pi}'$. □

Now we use lemmas 2 and 3 to prove the inequality B.5 in theorem 1.

Lemma 4. For partitionings $\mathbf{\Pi}'$ and $\mathbf{\Pi}$ in theorem 1, and θ' generated in lemma 3, inequality B.5 holds.

Proof. We show in lemma 3 that there is a θ' such that equality B.7 holds, and for any $x \in X$ and $q \in \mathbf{Q}$ we have $\bar{V}(x, \mathbf{\Pi}'(q); \theta', \mathbf{\Pi}') = \bar{V}(x, \mathbf{\Pi}(q); \theta^*, \mathbf{\Pi})$. Since $\bar{V}(.; \theta^*, \mathbf{\Pi})$ is a fixed point solution

to the standard contraction mapping problem for the Bellman equation, $\bar{V}(.;\theta',\mathbf{\Pi}')$ generated in lemma 3 is a solution to the contraction mapping in Bellman equation as well. Now we need to show that the decision likelihood is also higher.

$$\begin{aligned}
\mathcal{L}_{dc}(\theta',\mathbf{\Pi}') &= \sum_{i=1}^{N}\sum_{t=1}^{T}\log\hat{p}(d_{it}|x_{it},\mathbf{\Pi}'(q_{it});\theta',\mathbf{\Pi}') \\
&= \sum_{i=1}^{N}\sum_{t=1}^{T}\log\frac{\exp v(x_{it},\mathbf{\Pi}(q_{it}),d_{it};\theta',\mathbf{\Pi}')}{\exp\bar{V}(x_{it},\mathbf{\Pi}(q_{it});\theta',\mathbf{\Pi}')} \\
&= \sum_{i=1}^{N}\sum_{t=1}^{T}\log\frac{\exp v(x_{it},\mathbf{\Pi}(q_{it}),d_{it};\theta',\mathbf{\Pi}')}{\exp\bar{V}(x_{it},\mathbf{\Pi}(q_{it});\theta^*,\mathbf{\Pi})} \\
&= \sum_{i=1}^{N}\sum_{t=1}^{T}\log\Big(\frac{\exp v(x_{it},\mathbf{\Pi}(q_{it}),d_{it};\theta^*,\mathbf{\Pi})}{\exp\bar{V}(x_{it},\mathbf{\Pi}(q_{it});\theta^*,\mathbf{\Pi})}+\Pr(d_{it}|x_{it},\mathbf{\Pi}'(q_{it}))-\Pr(d_{it}|x_{it},\mathbf{\Pi}(q_{it}))\Big)
\end{aligned} \tag{B.12}$$

Please note the equality between line two and line three is driven from equation B.10, and between line three and line four from equation B.7. Using $\log(a+b)=\log(a)+\log(1+b/a)$, we have the following

$$\begin{aligned}
&\mathcal{L}_{dc}(\theta',\mathbf{\Pi}') = \sum_{i=1}^{N}\sum_{t=1}^{T}\log\Big(\frac{\exp v(x_{it},\mathbf{\Pi}(q_{it}),d_{it};\theta^*,\mathbf{\Pi})}{\exp\bar{V}(x_{it},\mathbf{\Pi}(q_{it});\theta^*,\mathbf{\Pi})}+\Pr(d_{it}|x_{it},\mathbf{\Pi}'(q_{it}))-\Pr(d_{it}|x_{it},\mathbf{\Pi}(q_{it}))\Big) \\
&= \sum_{i=1}^{N}\sum_{t=1}^{T}\log\Big(\frac{\exp v(x_{it},\mathbf{\Pi}(q_{it}),d_{it};\theta^*,\mathbf{\Pi})}{\exp\bar{V}(x_{it},\mathbf{\Pi}(q_{it});\theta^*,\mathbf{\Pi})}\Big)+\sum_{i=1}^{N}\sum_{t=1}^{T}\log\Big(1+\frac{\Pr(d_{it}|x_{it},\mathbf{\Pi}'(q_{it}))-\Pr(d_{it}|x_{it},\mathbf{\Pi}(q_{it}))}{\frac{\exp v(x_{it},\mathbf{\Pi}(q_{it}),d_{it};\theta^*,\mathbf{\Pi})}{\exp\bar{V}(x_{it},\mathbf{\Pi}(q_{it});\theta^*,\mathbf{\Pi})}}\Big) \\
&= \mathcal{L}_{dc}(\theta^*,\mathbf{\Pi})+\sum_{i=1}^{N}\sum_{t=1}^{T}\log\Big(1+\frac{\Pr(d_{it}|x_{it},\mathbf{\Pi}'(q_{it}))-\Pr(d_{it}|x_{it},\mathbf{\Pi}(q_{it}))}{\hat{p}(d_{it}|x_{it},\mathbf{\Pi}(q_{it});\theta^*,\mathbf{\Pi})}\Big) \\
&\Rightarrow \mathcal{L}_{dc}(\theta',\mathbf{\Pi}')-\mathcal{L}_{dc}(\theta^*,\mathbf{\Pi}) = \sum_{i=1}^{N}\sum_{t=1}^{T}\log\Big(1+\frac{\Pr(d_{it}|x_{it},\mathbf{\Pi}'(q_{it}))-\Pr(d_{it}|x_{it},\mathbf{\Pi}(q_{it}))}{\hat{p}(d_{it}|x_{it},\mathbf{\Pi}(q_{it});\theta^*,\mathbf{\Pi})}\Big)
\end{aligned} \tag{B.13}$$

We show that the right-hand side of equation B.13 is positive; thus, proving that the likelihood is increasing. Assuming $NT\to\infty$, and our predicted choice probabilities are consistent, concludes $\hat{p}(d_{it}|x_{it},\mathbf{\Pi}(q_{it});\theta^*,\mathbf{\Pi})=\Pr(d_{it}|x_{it},\mathbf{\Pi}(q_{it}))$. Replacing the predicted choice probability with its

counter-part conditional choice probability in equation B.13 yields

$$\begin{aligned}
&\sum_{i=1}^{N}\sum_{t=1}^{T}\log\Big(1+\frac{\Pr(d_{it}|x_{it},\mathbf{\Pi}'(q_{it}))-\Pr(d_{it}|x_{it},\mathbf{\Pi}(q_{it}))}{\hat{p}(d_{it}|x_{it},\mathbf{\Pi}(q_{it});\theta^*,\mathbf{\Pi})}\Big)\\
&=\sum_{i=1}^{N}\sum_{t=1}^{T}\log\Big(1+\frac{\Pr(d_{it}|x_{it},\mathbf{\Pi}'(q_{it}))-\Pr(d_{it}|x_{it},\mathbf{\Pi}(q_{it}))}{\Pr(d_{it}|x_{it},\mathbf{\Pi}(q_{it}))}\Big)\\
&=\sum_{i=1}^{N}\sum_{t=1}^{T}\log\frac{\Pr(d_{it}|x_{it},\mathbf{\Pi}'(q_{it}))}{\Pr(d_{it}|x_{it},\mathbf{\Pi}(q_{it}))} \qquad \text{(B.14)}
\end{aligned}$$

The value inside the summation in equation B.14 is equal to zero for all the observations, except for those where q_{it} lands in the newly split partition. Let us call the partition before the split $\pi_p \in \mathbf{\Pi}$, and the two resulting partitions $\{\pi_l, \pi_r\} \in \mathbf{\Pi}'$. Also let $N(x,\pi)$ denote the number of observations where $x_{it} = x$ and $\mathbf{\Pi}'(q_{it}) = \pi$, and $N(x,\pi,j)$ denote the number where in addition to the aforementioned conditions $d_{it} = j$. We can write B.14 as follow:

$$\begin{aligned}
\sum_{i=1}^{N}\sum_{t=1}^{T}\log\frac{\Pr(d_{it}|x_{it},\mathbf{\Pi}'(q_{it}))}{\Pr(d_{it}|x_{it},\mathbf{\Pi}(q_{it}))} &= \sum_{x\in X}\sum_{\pi\in\{\pi_l,\pi_r\}}\sum_{j\in\mathbf{J}} N(x,\pi,j)\log\big(\Pr(j|x,\pi)-\Pr(j|x,\pi_p)\big)\\
&= \sum_{x\in X}\sum_{\pi\in\{\pi_l,\pi_r\}} N(x,\pi)\sum_{j\in\mathbf{J}}\big(\Pr(j|x,\pi)\log\Pr(j|x,\pi)-\Pr(j|x,\pi)\log\Pr(j|x,\pi_p)\big)
\end{aligned}$$

According to Lemma 2, $\sum_{j\in\mathbf{J}}\Pr(j|x,\pi)\log\Pr(j|x,\pi) \geq \sum_{j\in\mathbf{J}}\Pr(j|x,\pi)\log\Pr(j|x,\pi_p)$. This inequality is strict if there is a $j \in \mathbf{J}$ such that $\Pr(j|x,\pi) \neq \Pr(j|x,\pi_p)$ for any $x \in X$ and $\pi \in \{\pi_l,\pi_r\}$. Thus, equation B.14 is greater or equal than zero, which proves the lemma and inequality B.5 in theorem 1. □

For the second part of theorem 1 we prove a more generalized lemma. First, we define a new relationship in the discretization space.

Definition 2. Discretization $\mathbf{\Pi}$ is a **parent** of discretization $\mathbf{\Pi}'$ if for every $\pi' \in \mathbf{\Pi}'$, there is a partition $\pi \in \mathbf{\Pi}$ such that π' is completely within π. In other words, discretization $\mathbf{\Pi}$ is a *parent* of discretization $\mathbf{\Pi}'$ if one can generate discretization $\mathbf{\Pi}'$ by further splitting discretization $\mathbf{\Pi}$.

According to the definition, $\mathbf{\Pi}$ is the parent of $\mathbf{\Pi}'$ in theorem 1. More generally, any discretization is the parent of all the next split candidate discretization in a recursive partitioning procedure. Now we prove the following lemma.

Lemma 5. For two partitioning $\mathbf{\Pi}$ and $\mathbf{\Pi}'$ such that $\mathbf{\Pi}$ is a parent of $\mathbf{\Pi}'$, the inequality B.6 holds.

Proof. Based on the definition of parent, for every $\pi \in \mathbf{\Pi}$, there is a set of partitions $\{\pi_i\} \in \mathbf{\Pi}'$ such that $\bigcup_i \pi_i = \pi$. Let us call $\{\pi_i\}$ the child set of π in $\mathbf{\Pi}'$ and denote it by $\mathbf{\Pi}'(\pi)$. First we use the log sum inequality [Cover and Thomas, 1991] to prove that for any $\{\pi, \pi'\} \in \mathbf{\Pi}$, $\{x, x'\} \in \mathbf{X}$, and $j \in \mathbf{J}$ the following inequality holds

$$\sum_{\pi_i \in \mathbf{\Pi}'(\pi)} \sum_{\pi'_i \in \mathbf{\Pi}'(\pi')} N(x, \pi_i, x', \pi'_i, j) \log \frac{N(x, \pi_i, x', \pi'_i, j)}{N(x, \pi_i) N(x', \pi'_i, j)} \geq N(x, \pi, x', \pi', j) \log \frac{N(x, \pi, x', \pi', j)}{N(x, \pi) N(x', \pi', j)} \tag{B.15}$$

We prove this inequality by applying the log sum inequality twice. First for a given $\pi_i \in \mathbf{\Pi}'(\pi)$ the following holds according to log sum inequality[2].

$$\sum_{\pi'_i \in \mathbf{\Pi}'(\pi')} N(x, \pi_i, x', \pi'_i, j) \log \frac{N(x, \pi_i, x', \pi'_i, j)}{N(x', \pi'_i, j)} \geq N(x, \pi_i, x', \pi', j) \log \frac{N(x, \pi_i, x', \pi', j)}{N(x', \pi', j)} \tag{B.16}$$

which by subtracting $N(x, \pi_i, x', \pi', j) \log N(x, \pi_i)$ from both sides changes to

$$\sum_{\pi'_i \in \mathbf{\Pi}'(\pi')} N(x, \pi_i, x', \pi'_i, j) \log \frac{N(x, \pi_i, x', \pi'_i, j)}{N(x', \pi'_i, j) N(x, \pi_i)} \geq N(x, \pi_i, x', \pi', j) \log \frac{N(x, \pi_i, x', \pi', j)}{N(x', \pi', j) N(x, \pi_i)}. \tag{B.17}$$

Similarly, according to log sum inequality we have

$$\sum_{\pi_i \in \mathbf{\Pi}'(\pi)} N(x, \pi_i, x', \pi', j) \log \frac{N(x, \pi_i, x', \pi', j)}{N(x, \pi_i)} \geq N(x, \pi, x', \pi', j) \log \frac{N(x, \pi, x', \pi', j)}{N(x, \pi)} \tag{B.18}$$

which by subtracting $N(x, \pi, x', \pi', j) \log N(x', \pi', j)$ from both sides changes to

$$\sum_{\pi_i \in \mathbf{\Pi}'(\pi)} N(x, \pi_i, x', \pi', j) \log \frac{N(x, \pi_i, x', \pi', j)}{N(x, \pi_i) N(x', \pi', j)} \geq N(x, \pi, x', \pi', j) \log \frac{N(x, \pi, x', \pi', j)}{N(x, \pi) N(x', \pi', j)}. \tag{B.19}$$

[2]We have $\sum_{\pi'_i \in \mathbf{\Pi}'(\pi')} N(x, \pi_i, x', \pi'_i, j) = N(x, \pi_i, x', \pi', j)$, and $\sum_{\pi'_i \in \mathbf{\Pi}'(\pi')} N(x', \pi'_i, j) = N(x', \pi', j)$,

By merging inequalities B.17 and B.19 we have

$$\begin{aligned}
&\sum_{\pi_i \in \mathbf{\Pi}'(\pi)} \sum_{\pi'_i \in \mathbf{\Pi}'(\pi')} N(x, \pi_i, x', \pi'_i, j) \log \frac{N(x, \pi_i, x', \pi'_i, j)}{N(x, \pi_i) N(x', \pi'_i, j)} \\
&\qquad \geq \sum_{\pi_i \in \mathbf{\Pi}'(\pi)} N(x, \pi_i, x', \pi', j) \log \frac{N(x, \pi_i, x', \pi', j)}{N(x, \pi_i) N(x', \pi', j)} \\
&\qquad \geq N(x, \pi, x', \pi', j) \log \frac{N(x, \pi, x', \pi', j)}{N(x, \pi) N(x', \pi', j)}
\end{aligned}$$

which concludes inequality B.15. We can prove the lemma by summing this inequality over all $\{\pi, \pi'\} \in \mathbf{\Pi}$, $\{x, x'\} \in \mathbf{X}$ and $j \in \mathbf{J}$. □

As we discussed, according to theorem 1, discretization $\mathbf{\Pi}$ is a parent of $\mathbf{\Pi}'$. Therefore, inequality B.6 holds, and the theorem is proven.

B.2 The Proof for Convergence to a Perfect Discretization

It is essential first to discuss conditions under which no recursive partitioning algorithm would capture all the heterogeneity. It has been shown that some patterns cannot be captured by recursive partitioning, even when the number of observations goes to infinity [Biau et al., 2008]. A simple example of such patterns is presented in Figure B.1. Assume observations are uniformly distributed throughout the covariate space. Also, assume that the flow utility in the crosshatched regions is l and it is h in the non-crosshatched regions. Any split would result in two sub-partitions with a similar number of observations in the crosshatched and non-crosshatched regions. The average utility would be equal to $\frac{h+l}{2}$ in the resulting two sub-partitions. Since the split does not increase $\mathcal{L}_{dc}$, the algorithm does not add it to its discretization. Generally, recursive partitioning cannot capture variational patterns that are symmetric in a way that any splits would lead to two sub-partition with a similar average statistic.

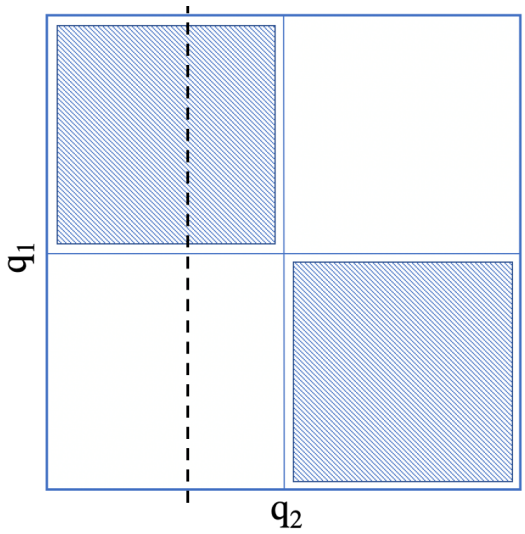


Figure B.1: An example of a pattern that cannot be captured by recursive partitioning. Observations in the white and crosshatched region have different statistics. Any split, such as the black dashed line, result in two sub-partitions that have similar average statistics.

To overcome this shortcoming of recursive partitioning algorithms, we assume no data patterns exist that a single split in a discretization cannot partially capture. Then we can draw the following corollary.

Corollary 5.1. A discretization $\mathbf{\Pi}$ is perfect, or there is a split such that the decision probabilities, average incoming transition probabilities, or outgoing transition probabilities on its two resulting sub-partitions are not equal. Formally, if discretization $\mathbf{\Pi}$ is not perfect, there exist a split that partitions $\pi_p \in \mathbf{\Pi}$ into π_l and π_r such that for a $\{x, x'\} \in \mathbf{X}$, $\pi' \in \mathbf{\Pi}$ and $j \in \mathbf{J}$ at least one of the following inequalities holds:

$$\begin{aligned}
\Pr(j|x, \pi_l) &\neq \Pr(j|x, \pi_r) \\
\Pr(x', \pi'|x, \pi_l, j) &\neq \Pr(x', \pi'|x, \pi_r, j) \\
\frac{\Pr(x, \pi_l|x', \pi', j)}{N(x, \pi_l)} &\neq \frac{\Pr(x, \pi_r|x', \pi', j)}{N(x, \pi_r)}
\end{aligned}$$

We prove the convergence of the recursive partitioning algorithm following this assumption and its resulting corollary.

Theorem 6. The recursive partitioning algorithm discussed in §3.4 create a perfect discretization, i.e., $\mathcal{F}(\mathbf{\Pi}_t) = \mathcal{F}(\mathbf{\Pi}_{t+1})$ if and only if $\mathbf{\Pi}_t$ is perfect.

Similar to theorem 1, we prove this theorem by proving separate lemmas for decision and state transition probabilities. Let us denote the decision and transition part of $\mathcal{F}(\mathbf{\Pi})$ by $\mathcal{F}_{dc}(\mathbf{\Pi})$ and $\mathcal{F}_{tr}(\mathbf{\Pi})$ respectively.

Lemma 7. For any candidate additional split $\{k, q, z\}$ to discretization $\mathbf{\Pi}$, that splits $\pi_p \in \mathbf{\Pi}$ into $\{\pi_l, \pi_r\} \in \mathbf{\Pi}'$, we have $\mathcal{F}_{dc}(\mathbf{\Pi}) = \mathcal{F}_{dc}(\mathbf{\Pi}')$ if and only if for all $x \in \mathbf{X}$ and $j \in \mathbf{J}$ the decision probabilities in π_l and π_r are similar, i.e., $\Pr(j|x, \pi_l) = \Pr(j|x, \pi_r)$.

Proof. We have

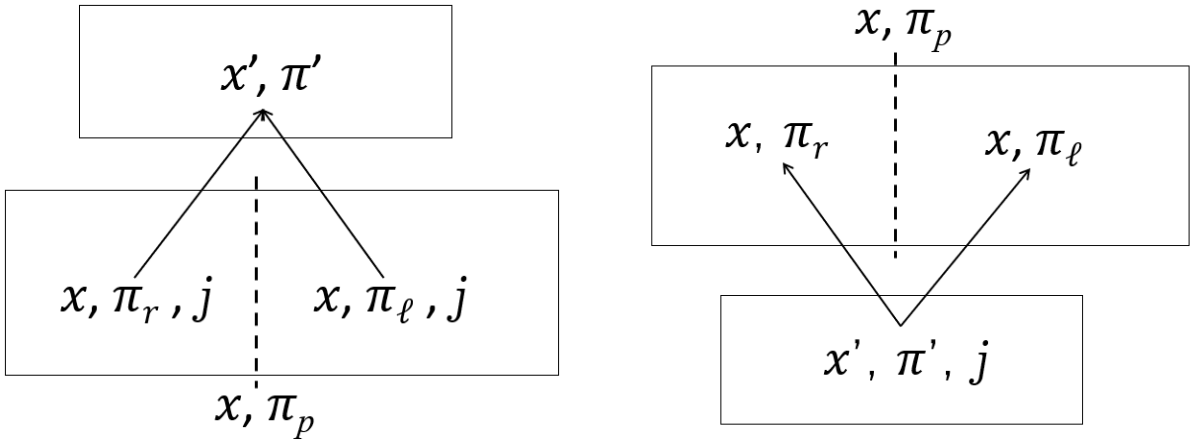
$$\begin{aligned}
\mathcal{F}_{dc}(\mathbf{\Pi}') - \mathcal{F}_{dc}(\mathbf{\Pi}) = N(x, \pi_l, j; \mathbf{\Pi}) \log \frac{N(x, \pi_l, j; \mathbf{\Pi})}{N(x, \pi_l; \mathbf{\Pi})} + N(x, \pi_r, j; \mathbf{\Pi}) \log \frac{N(x, \pi_r, j; \mathbf{\Pi})}{N(x, \pi_r; \mathbf{\Pi})} \\
- N(x, \pi_p, j; \mathbf{\Pi}) \log \frac{N(x, \pi_p, j; \mathbf{\Pi})}{N(x, \pi_p; \mathbf{\Pi})}
\end{aligned}$$

According to log sum inequality the right-hand side of this equality is equal to zero if and only if $\frac{N(x,\pi_r,j;\mathbf{\Pi})}{N(x,\pi_r;\mathbf{\Pi})} = \frac{N(x,\pi_l,j;\mathbf{\Pi})}{N(x,\pi_l;\mathbf{\Pi})}$, which concludes $\Pr(j|x, \pi_l) = \Pr(j|x, \pi_r)$. □

Lemma 8. For any candidate additional split $\{k, q, z\}$ to discretization $\mathbf{\Pi}$, that splits $\pi_p \in \mathbf{\Pi}$ into $\{\pi_l, \pi_r\} \in \mathbf{\Pi}'$, we have $\mathcal{F}_{tr}(\mathbf{\Pi}) = \mathcal{F}_{tr}(\mathbf{\Pi}')$ if and only if for any $\{x, x'\} \in \mathbf{X}$, $\pi' \in \mathbf{\Pi}'$, and $j \in \mathbf{J}$

the following equations hold

$$\Pr(x', \pi' | x, \pi_l, j) = \Pr(x', \pi' | x, \pi_r, j)$$
$$\frac{\Pr(x, \pi_l | x', \pi', j)}{N(x, \pi_l)} = \frac{\Pr(x, \pi_r | x', \pi', j)}{N(x, \pi_r)}$$



(a) Transition from the new sub-partitions

(b) Transition to the new sub-partitions

Figure B.2: The change in likelihood from transition-to and transition-from perspective.

Proof. We proved in appendix B.1 that likelihood, $\mathcal{L}(\theta_t)$, increases with any additional split. We assumed a completely non-parametric form for the state transition part of likelihood function. Therefore the state transition of likelihood function and recursive partitioning objective function are the same, i.e., $\mathcal{L}_{tr}(\mathbf{\Pi}) = \mathcal{F}_{tr}(\mathbf{\Pi})$. Here we prove that the increment in likelihood, and consequently in $\mathcal{F}(\theta_t)$, is equal to zero if and only if the lemma's equations hold. The split changes the likelihood by changing the transition-to and average transition-from probabilities presented in Figure B.2. We can calculate the changes in likelihood with respect to each of these changes separately. First,

according to (a) in Figure B.2 we have

$$\mathcal{F}(\mathbf{\Pi}') - \mathcal{F}(\mathbf{\Pi}) = \sum_{x,x' \in \mathbf{X}} \sum_{\pi' \in \mathbf{\Pi}'} \sum_{j \in \mathbf{J}} N(x', \pi') \Bigg(\frac{N(x', \pi', x, \pi_r, j)}{N(x', \pi')} \log \frac{N(x', \pi', x, \pi_r, j)}{N(x', \pi') N(x, \pi_r, j)} + \frac{N(x', \pi', x, \pi_l, j)}{N(x', \pi')} \log \frac{N(x', \pi', x, \pi_l, j)}{N(x', \pi') N(x, \pi_l, j)} - \frac{N(x', \pi', x, \pi_p, j)}{N(x', \pi')} \log \frac{N(x', \pi', x, \pi_p, j)}{N(x', \pi') N(x, \pi_p, j)} \Bigg)$$

According to log sum inequality the term in the parentheses is greater or equal to zero. The left-hand side is equal to zero if and only if $\frac{N(x',\pi',x,\pi_l,j)}{N(x,\pi_l,j)} = \frac{N(x',\pi',x,\pi_r,j)}{N(x,\pi_r,j)} = \frac{N(x',\pi',x,\pi_p,j)}{N(x,\pi_p,j)}$, for every $\{x, x'\} \in \mathbf{X}$, $j \in \mathbf{J}$ and $\pi' \in \mathbf{\Pi}'$. This concludes the first equation of the lemma.

We can conclude the second equation similarly with (b) in Figure B.2 as the following

$$\mathcal{F}(\mathbf{\Pi}') - \mathcal{F}(\mathbf{\Pi}) = \sum_{x,x' \in \mathbf{X}} \sum_{\pi' \in \mathbf{\Pi}'} \sum_{j \in \mathbf{J}} N(x', \pi', j) \Bigg(\frac{N(x, \pi_r, x', \pi', j)}{N(x', \pi', j)} \log \frac{N(x, \pi_r, x', \pi', j)}{N(x, \pi_r) N(x', \pi', j)} + \frac{N(x, \pi_l, x', \pi', j)}{N(x', \pi', j)} \log \frac{N(x, \pi_l, x', \pi', j)}{N(x, \pi_l) N(x', \pi', j)} - \frac{N(x, \pi_p, x', \pi', j)}{N(x', \pi', j)} \log \frac{N(x, \pi_p, x', \pi', j)}{N(x, \pi_p) N(x', \pi', j)} \Bigg)$$

Again, according to log sum inequality the term in the parentheses is greater or equal to zero. The left-hand side is equal to zero if and only if $\frac{N(x',\pi',x,\pi_l,j)}{N(x,\pi_l)N(x',\pi',j)} = \frac{N(x',\pi',x,\pi_r,j)}{N(x,\pi_r)N(x',\pi',j)} = \frac{N(x',\pi',x,\pi_p,j)}{N(x,\pi_p)N(x',\pi',j)}$, for every $\{x, x'\} \in \mathbf{X}$, $j \in \mathbf{J}$ and $\pi' \in \mathbf{\Pi}'$. This concludes the second equation of the lemma. □

Now we prove theorem 6 using lemmas 8 and 7. Assume that our algorithm stops at iteration t, and other stopping criteria are not met. We prove that any potential split does not decrease $\mathcal{F}(.)$. Since at each iteration of algorithm we choose the split with highest increase in the $\mathcal{F}(.)$, for all the potential next splits that that splits the partition $\pi \in \mathbf{\Pi}$ into $\{\pi_r, \pi_l\} \in \mathbf{\Pi}'$, where $\mathbf{\Pi}' = \mathbf{\Pi}_t + \text{split}$, we have $\mathcal{F}(\mathbf{\Pi}') = \mathcal{F}(\mathbf{\Pi}_t)$. Due to lemmas 8 and 7 for all splits the following equalities hold for all

$\{x, x'\} \in \mathbf{X}$, $\pi' \in \mathbf{\Pi}_t$ and $j \in \mathbf{J}$.

$$\begin{aligned}
\Pr(d|x, \pi_l) &= \Pr(d|x, \pi_r) \\
\Pr(x', \pi'|x, \pi_l, j) &= \Pr(x', \pi'|x, \pi_r, j) \\
\frac{\Pr(x, \pi_l|x', \pi', j)}{N(x, \pi_l)} &= \frac{\Pr(x, \pi_r|x', \pi', j)}{N(x, \pi_r)}
\end{aligned}$$

Therefore, according to 5.1 the discretization $\mathbf{\Pi}_t$ is perfect. Finally, since function $\mathcal{F}(.)$ strictly increasing at each iteration of the algorithm, and it cannot be higher than zero, the algorithm would eventually converge.

B.3 The Random Discretization Generator Algorithm in Second Simulation

This section explains the random discretization generator algorithm that we used in the second simulation study. The intuition for this algorithm is very similar to recursive partitioning – in each step, we randomly select one of the partitions and split it into two partitions along one of the first 10 variables in $\mathbf{Q}$. Please note that we only used the first 10 variables in $\mathbf{Q}$ for partitioning, and the following 20 variables are added as irrelevant variables to show the robustness of the algorithm to irrelevant variables. In addition, to prevent very small or very large partitions, the random partition selection is weighted by the size of the partition: big partitions are more likely to be selected than smaller partitions. Formally, the random partitioning algorithm is as the following.

- Initialize $\mathbf{\Pi}_0$ as one partition equal to the full covariate space
- Do the following for 15 rounds.
  - Randomly select a partition from $\mathbf{\Pi}_t$ weighted by $(\frac{1}{\text{partitions' total splits}})^2$
  - Randomly select a variable from the first 10 variables
  - If possible, split the selected partition into two from the midpoint along the selected variable. The total split for the resulting partitions is the total split of their parents plus one. Repeat this round if the split is not possible.

The total split for each partition is the total of times their parents have selected to be split. If the total split for a partition is 5, it means that it takes five splits from $\mathbf{Q}$ to get to that partition. Note that the total split of a partition is negatively correlated with the size of the partition.

We also vary the replacement cost for each partition to add a decision variation to the model that is not caused by state transition. The replacement cost of a partition is calculated based on the range of its first 10 variables as the following.

$$f_{dc}(\pi) = 5 - \frac{\sum_{i=1}^{10}(v_i^{min}(\pi) + v_i^{max}(\pi))}{10}$$

where $v_i^{min}(\pi)$ and $v_i^{max}(\pi)$ are the minimum and maximum range of i^{th} variable in partition π. We choose this formulation for replacement cost to create a good balance between decision variation caused by state transition or replacement cost. Figure B.3 depicts the distribution of total split and replacement costs in the 1500 generated partitions across all the 100 generated discretizations in the second simulation study.

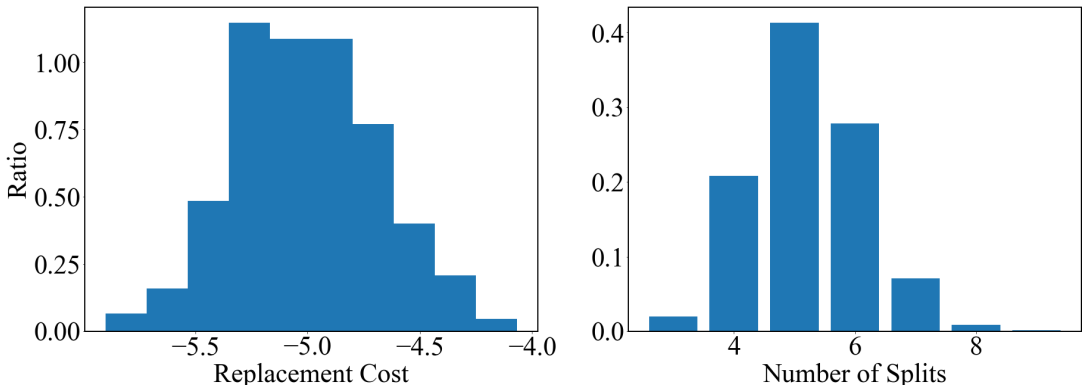


Figure B.3: The histogram of generated partitions' calculated replacement cost, and number of splits in the 100 generated partitioning.