Web Appendix - Design and Evaluation of Optimal Free Trials

Hema Yoganarasimhan [*] University of Washington
Ebrahim Barzegary[*] University of Washington
Abhishek Pani[*] Bright Machines

Appendices

A 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 A1: 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.

[*]We are grateful to an anonymous firm for providing the data. We thank Atanu Lahiri, Simha Mummalaneni, and Omid Rafieian for detailed comments that have improved the paper. Thanks are also due to the participants of the 2018 Marketing Science conference and the Triennial Choice Symposium, 2021 UT Dallas Bass Conference and seminar audiences at the Emory University, Johns Hopkins University, Lehigh University, University of California at Berkeley, University of Houston, University of Maryland, University of Rochester, University of South Carolina, and University of Southern California, for their feedback. Please address all correspondence to: hemay@uw.edu.

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 A2: 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 A3: The number of observations for each trial length in each dataset.

B Appendix for ATE and Randomization Checks

B.1 ATE using Regressions

	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 A4: The coefficients of 7 days and 14 days in regression analysis. In the control case, we control for all the pre-treatment variables.

B.2 Randomization Checks

First, we show the distribution of treatment assignment across the five skill levels in Table A5. 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 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 A5: 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 A6. 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 A7. The treatment effects are very similar to those in Table 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 A6: 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 A7: Effect of trial length calculated by regressing the subscription decision on all the pre-treatment variables and trial length.

C Appendix for Mechanism

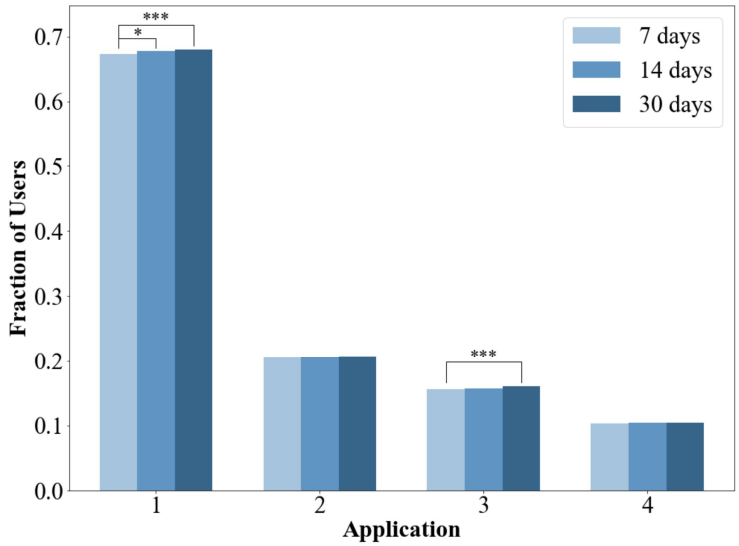

Figure A1: Fraction of users downloading products 1–4 under each trial length. Products 1 and 3 are complements.

D Lasso Implementation Details

Next, we briefly describe our implementation of the lasso model discussed in §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.

E Design of Alternative Personalized Policies

We discuss the design of a series of alternative personalized policies. First, in §E.1, we present with the policies based on other outcome estimators. Next, in §E.2, we discuss policies based on CATE estimators.

E.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:

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

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.

E.1.1 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

[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.

used in the main text addresses the above problem by learning a simpler model that uses fewer variables (Tibshirani, 1996; Friedman et al., 2010).

E.1.2 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., 1984). 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 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

[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.

[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.

superior to earlier implementations both in terms of accuracy and scalability.

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

E.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, $w_j, w_{j'}$.

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

[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.

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.

E.2.1 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.,

[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).

variation in the post-split treatment effect).

E.2.2 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. (2019) 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:

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

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. (2019) 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 §F 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.

F 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., 1984). 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.

G 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 A8: Fraction of users assigned the 7-, 14-, and 30-day trials under counterfactual policies (in test data).

First, in Table A8, 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 A9: 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 A9, 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, 2019). 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 A2, we show the CDFs of the estimated CATEs for the 7 vs. 14 day and 14 vs. 30 treatments for all the methods.

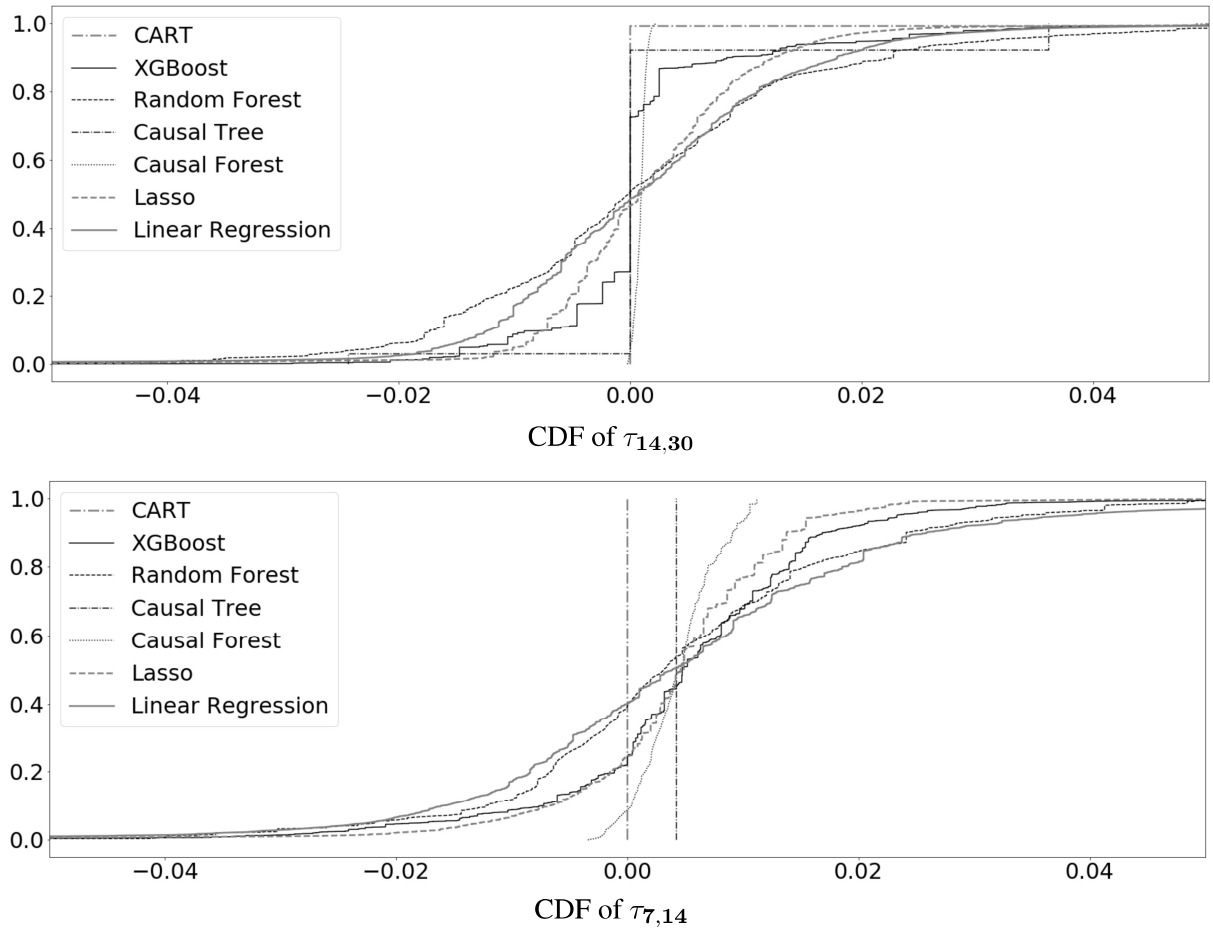

Figure A2: The CDF of estimated CATEs for 7 vs 14 days, and 14 vs 30 days of free trial from using different methods (for test data).

H Appendix for §6

This section is organized as follows. First, in §H.1, we quantify the heterogeneity in the effect of trial length on post-treatment usage. Next, in §H.2, we quantify the heterogeneity in the effect of usage on subscription. Then, in §H.3, we show how these heterogeneous responses are consistent with the user's pre-treatment demographic variables. Finally, in §H.4, we provide additional evidence to rule out the demand cannibalization hypothesis.

H.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 A10: 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 A11: Regression of different usage features on the interaction of trial length, and optimal trial length. Standard errors in parentheses.

We start with Table A10, 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

[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.

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 A11, 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.

H.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 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 3. Table A12 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 A12: 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.

H.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 A13 and A14.

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 A13: 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 A14: 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.

H.4 Additional Evidence to Rule Out Demand Cannibalization

We now provide additional evidence to rule out the demand cannibalization hypothesis. In Figure A3, 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 A15 shows that beginners who use the product more are more likely to subscribe, even after controlling for all the other user-specific variables.

	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 A15: 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.

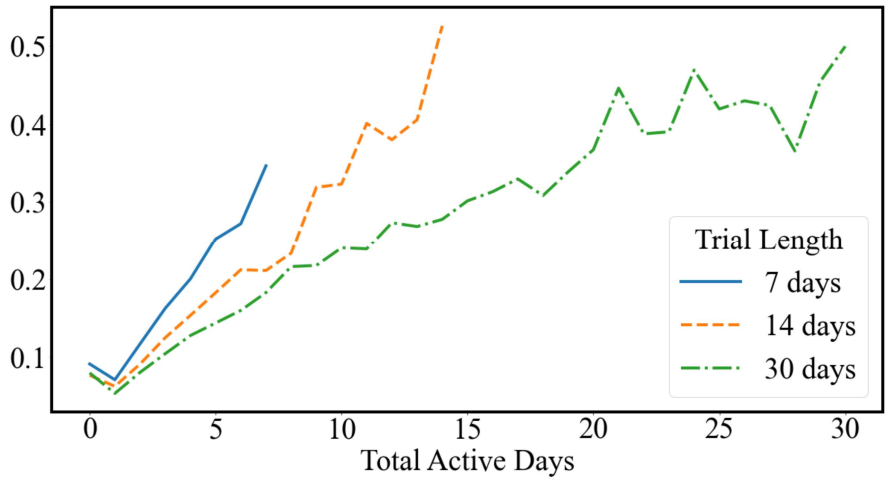

Figure A3: Subscription probability of *beginner* users with different total active days. The subscription probability increases as the total active days increases.

I Appendix for §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 A16: The summary statistics of the subscribed users' total months of subscription.

Trial Length	Subscribed Bundle					Subscription Type		
	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 A17: 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.

J Appendix for §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 A18: The percentage of users allocated to each trial length in the three policies based on: (1) subscription, (2) subscription length, and (3) revenue.

Table A18 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 A16 in Appendix I). 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 A17 in Appendix I). 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.

References

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

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.

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, 1984.

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.

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.

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.

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

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

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

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.

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

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

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

P. M. Robinson. Root-n-consistent semiparametric regression. *Econometrica: Journal of the Econometric*

Society, pages 931–954, 1988.

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

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

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

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.